DYNAMIC ROUTING WITH COMPETITION: FOUNDATIONS AND STRATEGIES

A thesis presented in partial fulfilment
of the requirements for
the degree of Doctor of Philosophy in
Operations Research at
Massey University

Mark Richard Johnston

1999

Errata and Addenda

P 7 line 2	Replace "evaluate these difficulty" by "evaluate the difficulty".
P 17 line 11	Replace '[35]' by '[35, page 475]'.
P 19 line −12	Replace "Brideau and Cavalier" by "Brideau and Cavalier [28, page 1]".
P 21 line −17	Replace "They adopt" by "Desrochers, Lenstra and Savelsbergh [54, page 323] adopt".
P 21 line −1	Replace "which will be useful to" by "that will be useful in".
P 23 line 18	Replace '[60]' by '[60, page 12]'.
P 26 line 13	Replace 'interesting' by 'interestingly'.
P 26 line 14	Replace '[146]' by '[146, page 1105]'.
P 28 line 10	Replace '[7]' by '[7, page 206]'.
P 31 line 14	Replace "Is it possible communicate between decision makers?" by "Is communication possible between decision makers?".
P 34 line −16	Delete 'of these'.
P 39 line −6	Replace "move continuously at the same constant speed in the Euclidean plane" by "move continuously at unit speed in the Euclidean plane and may instantaneously change direction".
P 40 line 3	Replace '[7]' by '[7, pages 11–18]'.
P 43 line −7	Replace "principles make" by "principles that make".
P 46 line 3	Insert after the first sentence: "Let d_{Xi} be the distance from player $\mathcal{X}$'s current location to the location of prize i (at unit speed this is equivalent to the minimum possible time for player $\mathcal{X}$ to reach prize i)."
P 46 line 23	Replace "players prize i" by "players share prize i".
P 47 line −17	Delete 'a'.
P 47 line −15	Delete 'it'.
P 47 line −11	Replace "median prize of move" by "median prize or move".

P 49 line −18	Replace "at least on prize" by "at least one prize".
P 52 line 4	Replace "subpath, but not necessarily committing to entire, but" by "subpath, not necessarily committing to an entire subpath, but".
P 53 line −13	The definition given for *Farthest Insertion* is that of *Most Expensive Insertion*. For **Farthest Insertion**, select vertex k not on the subpath such that d_{ik} (for $k \in$ subpath) is maximized. Insert vertex k between subpath edge (i, j) such that $d_{ik} + d_{kj}$ is minimized.
P 54 line −1	Replace "guaranteed to player $\mathcal{A}$" by "guaranteed to player $\mathcal{A}$ if targeted first".
P 55 line −1	Replace "higher likelihood" by "higher likelihood of capture".
P 58 line 8	Delete 'we'.
P 59 line −14	Insert new paragraph "Let $\wp$ denote a problem instance defined by the number of prizes, the location of each prize, the value of each prize, the initial location of each player, the overall deadline, and the step size. The guaranteed value of a maximal guaranteed subpath for player $\mathcal{A}$ is denoted $\Gamma_A(\wp)$ and the guaranteed value of a maximal guaranteed subpath for player $\mathcal{B}$ is denoted $\Gamma_B(\wp)$."
P 61 line −1	Replace 'paranoid' by 'GUARANTEE-SUBPATH'.
P 63 line −10	Replace "We can calculate the **cooperative value**, Ω, which is" by "For a problem instance $\wp$, we can calculate the **cooperative value**, $\Omega(\wp)$, which is".
P 63 line −5, −1	Replace 'Ω' by '$\Omega(\wp)$'.
P 64 line 8	Replace "select between them." by "select between them?"
P 68 line −15	Replace "is the defined" by "is then defined"
P 68 line −13	Replace 'apply' by 'applying'.
P 93 line 6	Delete comma.
P 93 line 13	Replace "$d_{A1} < d_{B1}$ and $d_{A2} = d_{B2}$" by "$d_{Ai_1} < d_{Bi_1}$ and $d_{Ai_2} = d_{Bi_2}$".
P 93 line 19	Replace "$\mathcal{A}$—$\circ$ i_1 and $\mathcal{B}$—$\circ$ i_1 is a Nash equilibrium in which the players share the sequence $i_1 \rightarrow i_2$" by "$\mathcal{A}$—$\circ$ i_2 and $\mathcal{B}$—$\circ$ i_2 is a Nash equilibrium in which the players share the sequence $i_2 \rightarrow i_1$".

P 93 line 22	Replace "$\mathcal{A}$–$\circ$argmax$\{v_1, v_2\}$ and $\mathcal{B}$–$\circ i_2$, for a reward of $\max\{\frac{1}{2}(v_1 + v_2), v_1\}$ to player $\mathcal{A}$ and $\min\{\frac{1}{2}(v_1 + v_2), v_2\}$ to player $\mathcal{B}$" by "$\mathcal{A}$–$\circ$argmax$\{v_{i_1}, v_{i_2}\}$ and $\mathcal{B}$–$\circ i_2$, for a reward of $\max\{\frac{1}{2}(v_{i_1} + v_{i_2}), v_{i_1}\}$ to player $\mathcal{A}$ and $\min\{\frac{1}{2}(v_{i_1} + v_{i_2}), v_{i_2}\}$ to player $\mathcal{B}$".
P 94 line −12	Replace 'max' by 'min'.
P 94 line −11	Replace 'of' by 'or'.
P 94 line −10	Replace 'max' by 'min'.
P 103 line 18	Replace '$\mathcal{B}$' by '$\mathcal{A}$'.
P 109 line −13	Replace '$\mathcal{B}$safety-window' by '$\mathcal{B}$-safety-window'.
P 117 line −12	Replace last '+' by '$\geq$'.
P 118	Replace '$\Omega(\mathcal{A}, \mathcal{B})$' by '$\Omega(p)$', replace '$\Gamma(\mathcal{A})$' by '$\Gamma_A(p)$', and replace '$\Gamma(\mathcal{B})$' by '$\Gamma_B(p)$'.
P 118 line 15	Replace "every prize of sequence of prizes" by "every prize or sequence of prizes".
P 118 line 17	Replace 'paranoid' by 'guaranteed'.
P 118 line −1	Delete the redundant condition "and $d_{Ai_2} + d_{i_1 i_2} < d_{Bi_1}$".
P 119 line 1	Replace "$\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is accessible but $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is not accessible" by "$\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is accessible but $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is not accessible".
P 119 line 6	Replace '$\Omega(\mathcal{A}, \mathcal{B})$' by '$\Omega(p)$'.
P 133 line −4	Replace '$\Gamma(\mathcal{A}\|\mathcal{B})$' by '$\Gamma_A(p)$' and replace $\Omega(\mathcal{A}, \mathcal{B})$' by '$\Omega(p)$'.
P 136	Replace '$\Gamma(\mathcal{A}\|\mathcal{B})$' by '$\Gamma_A(p)$', replace $\Gamma(\mathcal{B}\|\mathcal{A})$' by '$\Gamma_B(p)$', and replace $\Omega(\mathcal{A}, \mathcal{B})$' by '$\Omega(p)$'.
P 139 line −1	Replace $\Omega(\mathcal{A}, \mathcal{B})$' by '$\Omega(p)$'.
P 146 line 2	Replace 'engine' by 'engines'.
P 170 line −13	Replace 'define' by 'determine'.
P 187 line −12	Replace 'prosed' by 'proposed'.
P 192 line −8	Delete 'the'.
P 196 line 15	Replace '$\Gamma_A(\mathcal{A} \rhd x\mathcal{B} \rhd y)$' by '$\Gamma_A(\mathcal{A} \rhd x, \mathcal{B} \rhd y)$'.
P 198 line 3	Replace "which make" by "that would make".
P 202	In the first table replace '$v_{i_1} < i_2 v_{i_3}$' by '$v_{i_1} < 2v_{i_3}$' and in the second table replace '$v_{i_2} < i_2 v_{i_3}$' by '$v_{i_2} < 2v_{i_3}$'.
P 206 line −20, −18	Replace '$\Gamma(\mathbf{j}, \mathcal{X})$' by '$\Gamma_X(\mathbf{j})$'.
P 206 line −19	Replace '$\Gamma(\mathbf{j})$' by '$\Omega(\mathbf{j})$'.
P 210 line 19	Replace 'that' by 'there'.
P 210 line −15	Replace 'provide' by 'provides'.
P 213 line 10	Replace 'defined' by "defined by".
P 213 line −17	Replace 'local-optima' by "local optimum".
P 217 line 4	Replace "swap to clusters" by "swap two clusters".
P 220 line 22	Delete "in an $\mathcal{X}$-future-family".
P 225 line −10	Before "Table 7.2(d) illustrates" insert the sentence "This exhibits a potential drawback of enforcing the *cluster-no-return* rule."
P 233 line −13	Replace '[29]' by '[29, page 111]'.
P 241 line −10	Replace 'lemma' by 'theorem'.
P 245 line 10	Replace '$\mathcal{A}$→14→12→12' by '$\mathcal{A}$→14→12→13'.
P 245 line 11	Replace '$\mathcal{A}$→11→9→6→7→8' by '$\mathcal{B}$→11→9→6→7→8'.
P 245 line −11	Replace 'playerA→14→12→13' by '$\mathcal{A}$→14→12→13'.
P 245 line −11	Replace '199' by '179'.
P 247	The caption for Table 7.4(d) should read "Player $\mathcal{B}$ FAMILY-PRIZE-GUARANTEE Subpath".
P 251 line −15	Replace 'posible' by 'possible'.
P 254 line 30	Add to the end of the paragraph: "In each case the two player game table indicates the *structure* of the root game table and '□' indicates an entry remaining to be evaluated."
P 261 line −6	Replace "clusters [2] and 35]" by "clusters [2] and [3]".
P 281 line 6	Replace "which is even more confusing" by "which is interesting since both players were previously trying to avoid a tactical conflict".
P 281 line −10	Add to the end of the paragraph: "Cycling occurs when players perpetually retargeting in response to the opponent retargeting. It would eventually be resolved as the players move closer to their respective candidate target clusters but may be detected and resolved earlier by switching to a MAXIMIN evaluator."
P 283 line 10	Replace 'governed' by "governed by".
P 284 line −11	Replace "be following in" by 'follow'.
P 286 line 1	Replace "to straightforward" "to be straightforward".
P 287 line 16	Move □ to the right hand side.
P 302 line −15	Delete "in a player $\mathcal{X}$ future-family".
P 305 line −20	Delete "in a player $\mathcal{A}$ future-family".
P 305 line −11	Replace '$\mathcal{G}_h + 1$' by '$\mathcal{G}_{h+1}$'.

P 337 line −7	Replace 'between' by 'from'.
P 340 line −9	Insert: "In the latter two grid-structures, players are located near to the centre of a grid-cell to minimize the edge effects in the initial planning paths."
P 366 line 7	Replace "measure a strategies" by "measure a strategy's".
P 387 line −10	Replace "but a similarly" by "but are similarly".
P 389	Delete prize 1 from Table 9.7(b) and Table 9.7(d).
P 392 line −10	Replace "is one which is hard to predict" by "is one for which it is hard to predict".
P 393	Delete prize 1 from Table 9.9(b) and Table 9.9(d).
P 400	In Table 10.1 replace '//PRIZE-PARANOID' by '//PRIZE-GUARANTEE'.
P 404 line −14	Replace "A ORIGINAL-DT" by "An ORIGINAL-DT".
P 407	The second part of Figure 10.1 should have label '(b)'.
P 422 line −8	Replace first 'cluster' by 'clusters'.
P 433 line −1	Delete one 'the'.
P 438 line −6	Replace "grid-size. which" by "grid-size, which".
P 462	Rename Section 11.2.1.1 as "Characterisation of Strategies and Problem Instances".
P 462	Add the following paragraph to Section 11.2.1.1: "Problem design (via templates, or more refined bad-case examples) needs to be structurally *catalogued*, if possible. Also, the concept of fairness, methods for 'quickly' estimating the value of a game, and playing against a 'capricious' game designer need to be investigated. It is also necessary to further distinguish between inherently difficult problems, and those which are difficult for the solution strategies."

Bibliography

[12] BALL, M. O. Allocation/routing: models and algorithms. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 199–221.

[51] DELL'AMICO, M., MAFFIOLI, F., AND VÄRBRAND, P. On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research 2* (1995), 297–308.

[202] VOLGENANT, T., AND JONKER, R. On some generalizations of the travelling salesman problem. *Journal of the Operational Research Society 38*, 11 (1987), 1073–1079.

[207] WINSTON, W. L. *Operations Research: Applications and Algorithms*, 3rd ed. Duxbury Press, Belmont, California, 1994.

Glossary of Symbols

The Players

$\mathcal{A}$, $\mathcal{B}$	The two specific players.	39
$\mathcal{C}$	A third specific player.	467
$\mathcal{X}$, $\mathcal{Y}$	Generic players, not specifically stating which is player $\mathcal{A}$ or which is player $\mathcal{B}$.	45
$(x_A, y_A), (x_B, y_B)$	The initial location of the players.	39

Prizes, Clusters and Families

$V = \{1, 2, \ldots, n\}$	The set of all prizes.	39
(x_i, y_i)	The location of prize i.	39
v_i	The value of prize i.	39
$[ci]$	A cluster, elements are prizes $j \in [ci]$.	211
$\eta_{[ci]}$	Required PCTSP value for cluster $[ci]$.	218
$\mathcal{C}$	A clustering, elements are clusters $[ci] \in \mathcal{C}$.	211
$\mathcal{F}$	A family (also a clustering).	216
$\mathbb{F}$	Set of families, elements are families $\mathcal{F} \in \mathbb{F}$.	216

The hierarchy of membership is $j \in [ci] = Q \in \mathcal{Q} \in \mathbb{Q}$.

Distances and Times

λ	Overall deadline.	39
Δ	Step size.	45
d_{ij}	Distance between prize i_1 and prize i_2.	46
d_{Xj}	Distance from player $\mathcal{X}$'s current location to prize j.	46
$D_{[ci][cj]}$	Distance between clusters.	212
$d_{jC_{[ci]}}$	Distance between prize j and the centroid of cluster $[ci]$.	212
τ_{Aj}	Earliest arrival time of player $\mathcal{A}$ at prize j.	149
τ_{Bj}	Earliest arrival time of player $\mathcal{B}$ at prize j.	149

Problem Instances

$\wp$	CPCP problem instance.	59
$P : n\ell\ell vab\lambda$	Prize class (P-class) of problem instances.	372
$C : m\ell\ell v s\kappa ab\lambda$	Cluster class (C-class) of problem instances.	375
$D : hcbssa$	Density class (D-class) of problem instances.	381

Player Strategies

$\mathbb{S}$	Set of player strategies.	381
$\mathbb{S}_\infty$	Infinite set of all possible player strategies.	381
$\mathbb{S}_{A\infty}$	Infinite set of all possible strategies for player $\mathcal{A}$ for the two prize CPCP.	122
$\mathbb{S}_{B\infty}$	Infinite set of all possible strategies for player $\mathcal{B}$ for the two prize CPCP.	122
a, b	Individual player strategies.	122

Game Values

$\Gamma_A(\wp)$	Maximum value guaranteed to player $\mathcal{A}$.	59
$\Gamma_B(\wp)$	Maximum value guaranteed to player $\mathcal{B}$.	59
$\Omega(\wp)$	Cooperative value.	118
$\varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Score for player $\mathcal{A}$.	401
$\varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Score for player $\mathcal{B}$.	401
$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S})$	Effectiveness: either MIN-MIN, MIN-MEAN or MEAN-MEAN.	401
$\varkappa^\diamond(\wp)$	Overall problem MAXIMIN score.	402
$\Xi(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Number of steps.	402
$\tau_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Processor time.	402
$\tau_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Processor time.	402
$\xi(\wp)$	Difficulty of problem instance $\wp$.	402
$\bar{\xi}(\mathbb{P})$	Average difficulty of a problem instance class $\mathbb{P}$.	402
$v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Dynamic simulation battle value to player $\mathcal{A}$.	381
$v_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$	Dynamic simulation battle value to player $\mathcal{B}$.	381
$v_A^*(\wp)$, $v_B^*(\wp)$	Nash equilibrium value of the problem instance.	381
$v_A^\sharp(\wp)$, $v_B^\sharp(\wp)$	Infinite maximin value of the problem instance.	381
$v_A^\diamond(\wp)$, $v_B^\diamond(\wp)$	Computational maximin value of the problem instance.	385

$v^*(\wp)$, $v^H(\wp)$	Optimal and heuristic value of a solution to a combinatorial optimization problem $\wp$.	390
ρ_H	Worst case performance of a heuristic H on a combinatorial optimization problem.	390
$\xi(\wp)$	Difficulty of problem instance $\wp$.	392

Paths, Game Trees and Branch and Bound Trees

j	A game tree node.	155
i	A branch and bound tree node.	205
$\Gamma_A(\mathrm{j})$	The guarantee value for player $\mathcal{A}$ corresponding to game tree node j.	160
$\Gamma_B(\mathrm{j})$	The guarantee value for player $\mathcal{B}$ corresponding to game tree node j.	160
$\Omega(\mathrm{j})$	The cooperative value corresponding to game tree node j.	160
P	Planning subpath.	72
P_A	Planning path for player $\mathcal{A}$.	62
P_B	Planning path for player $\mathcal{B}$.	61
$v_A(P_A, P_B)$	The value of prize on subpath P_A claimed outright by player $\mathcal{A}$ plus half the value of those prizes on P_A shared with player $\mathcal{B}$ on subpath P_B.	155

Step Planning

$\mathcal{X}{\multimap}i$	Player $\mathcal{X}$ moves one step towards prize i.	92
$\mathcal{X}{\multimap}X$	Player $\mathcal{X}$ moves directly to location X.	92
$\mathcal{X} \rhd i$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to prize i.	92
$\mathcal{X} \rhd (i_1 \rightarrow i_2)$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to prize i_1 then all the way to prize i_2.	97
$\mathcal{X} \rhd (i_1 \rightarrow i_2 \rightarrow i_3)$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to prize i_1 then all the way to prize i_2 then all the way to prize i_3.	103

$\mathcal{X} \rhd \{i_1, i_2\}$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to at least one of prizes i_1 and i_2.	98

Tactical Planning

Q	Prizes remaining unclaimed.	49
Q_A, Q_B	Prizes remaining and accessible to each player.	147
P_A, P_B	Planning paths for each player.	148
t	Projected time stamp.	148
t_A, t_B	Projected time stamps.	168
$\mathcal{L}_A, \mathcal{L}_B$	Lead prizes for each player.	169
$\mathcal{D}_A, \mathcal{D}_B$	Direct lead prizes for each player.	169
$\mathcal{R}_A, \mathcal{R}_B$	Follow prizes for each player.	169
$\mathcal{F}_A, \mathcal{F}_B$	Follow pairs for each player.	170
$\odot$	Probe.	148
$\mathcal{X}{\rightarrow}P_X \odot x$	Probe.	148
$\oplus$	Commitment.	169
$\mathcal{X}{\rightarrow}P_X \oplus x$	Commitment.	169
W	Window scenario and corresponding feasibility window.	170
$\mathcal{X} \rhd (W{\rightarrow}\{i_1, i_2\})$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to at least one of prizes i_1 and i_2 via feasibility window W.	171

Strategic Planning

FF	Final family requirement enforced.	219
$\mathcal{Q}_A, \mathcal{Q}_B$	Families remaining and accessible to each player.	220
$\mathcal{Q}_A, \mathcal{Q}_B$	Clusters remaining and accessible to each player.	220
$\mathcal{S}_\mathcal{X}$	Set of target-clusters of player $\mathcal{X}$.	220
$\mathbb{P}_A, \mathbb{P}_B$	Cluster planning paths.	230
$\mathbb{W}$	Cluster window scenario and corresponding cluster window feasibility constraints.	257

$\mathcal{X} \rhd [c_i]$	The 'look through' scenario that player $\mathcal{X}$ is committed to travelling all the way to some prize in cluster $[c_i]$.	231
$\mathcal{B} \rhd \{[c_{y_1}], [c_{y_2}]\}$	The 'look through' scenario that player $\mathcal{B}$ is committed to travelling all the way to at least one prize from cluster $[c_{y_1}]$ or cluster $[c_{y_2}]$.	256
$\mathcal{B} \rhd ([c_{y_0}] \rightarrow \{[c_{y_1}], [c_{y_2}]\})$	The 'look through' scenario that player $\mathcal{B}$ is committed to travelling all the way to at least one of prize from cluster $[c_{y_0}]$ and then travelling all the way to at least one prize from cluster $[c_{y_1}]$ or cluster $[c_{y_2}]$.	257
$\mathcal{A} \rightarrow \mathbb{P}_A \oplus [c_{x_0}] \rhd [c_{x_1}]$	Player $\mathcal{A}$ moves through cluster $[c_{x_0}]$ while committed to cluster $[c_{x_1}]$ as the next look-through cluster.	238
$\mathcal{B} \rightarrow \mathbb{P}_B \rhd \emptyset$	Player $\mathcal{B}$ has no look-through commitment.	243
$\mathcal{B} \rightarrow \mathbb{P}_B \oplus [c_{x_0}] \rhd \emptyset$	Player $\mathcal{B}$ moves through cluster $[c_{x_0}]$ and has no further look-through commitment.	243
$\mathcal{B} \rightarrow \mathbb{P}_B \rhd [c_{x_0}]$	Player $\mathcal{B}$ is committed to cluster $[c_{x_0}]$ as the next look-through cluster.	253

Grid Planning

$\mathbb{G}_A$, $\mathbb{G}_B$	Grid planning paths.	344
$\mathcal{A} \rightarrow \mathbb{G}_A \oplus_D (i, j)$	Commitment to grid cell (i, j) via the intra-cell-direct-path through the current $\mathcal{A}$-grid-cell.	344
$\mathcal{A} \rightarrow \mathbb{G}_A \oplus_H (i, j)$	Commitment to grid cell (i, j) via the intra-cell-harvest-path through the current $\mathcal{A}$-grid-cell.	345
$\mathcal{A} \rightarrow \mathbb{G}_A \oplus_P (x, y)$	Commitment to grid cell (i, j) carrying last-direct, last-harvest and all-direct paths via the current $\mathcal{A}$-grid-cell.	348

Abstract

Operations Research studies a wide range of problems, including long-term, strategic, business planning and short-term, operational, logistical planning. Long-term business decisions revolve around the market demand for goods or services, whereas logistics focuses on efficient scheduling of production and distribution. However, vehicle routing and scheduling problems in a dynamic environment require short-term, operational planning in conjunction with computationally expensive, short-term tactical considerations.

This thesis investigates a model of competition in the distribution of goods to customers, in which a number of independent carriers compete to deliver goods to a fixed set of customers. Assuming that the price and quality of the goods are consistent, each customer is indifferent towards which carrier actually delivers the required goods, but will only accept delivery from the first to arrive at their location. The main source of uncertainty is planning for competition against other independent carriers.

Firstly, we consider the basic elements of competition vehicle routing and scheduling problems, and propose a Reference Model for Competition Routing Problems, synthesising the literature from vehicle routing and game theory. The general problem involves a number of independent decision makers, each representing a carrier company with a private fleet of vehicles, and a fixed set of customers to be serviced. We also formulate the Competitive Prize Collection Problem (CPCP), involving two independent decision makers with one vehicle each. The CPCP encapsulates the core elements of competition within a two player version of the Prize Collecting Travelling Salesman Problem.

Secondly, we consider which strategic, tactical, and operational planning elements are important in the design of strategies for effective performance on the CPCP. We propose a Strategic Planning Architecture (SPA), i.e., a strategy framework based on hierarchical planning at nested planning horizons. This incorporates strategic and tactical planning engines based on modelling the decision problem at each planning horizon as a multiple

stage game. Dynamic monitoring processes match these strategic plans to the predicted and observed movements of the opponent. Strategies which implement the SPA are designed to cover a range of planning horizons and problem sizes.

A series of computational tournaments on problems of different sizes and characteristics suggests that strategies which address contingent planning, cognizance of opponent, and planning based on existing natural structure, are the most effective of those considered. In the process, benchmark sets of robust strategies, and challenging problem instances, are established against which the effectiveness of strategies may be evaluated.

The significant conclusion is that for small problems, strategic considerations are more effective than routing, but for large problems, routing considerations are more effective than strategic. Problems in between require a balance between strategy, response, and routing considerations. Routing only is not sufficient; response requires good strategic information. The CPCP remains a deceptively simple problem which is computationally demanding at all scales of planning, from small problems to large problems. There is considerable scope for the study of further strategies, especially those able to classify, learn from and adapt to, the observed behaviour of the opponent, and for extrapolating these results to a richer set of competition routing problems.

Acknowledgements

When you are a bear of very little brain, and you think of things, you find sometimes that a thing which seemed very thingish inside you is quite different when it gets out into the open and has other people looking at it.

— WINNIE THE POOH (A.A. MILNE)

This thesis is dedicated to the many people who have been supportive, and encouraging, in numerous ways: tiny, small, medium, large and *jumbo*. I am grateful to IBM New Zealand, Landcare Research, and Massey University for their generous financial and technical support.

Thanks to my supervisor, Dr. John Giffin, for his exhortation to "Ph.D. thinking" and his attempts to eschew obfuscation (a.k.a. the red pen). Thanks to Professor Michael Hendy for many (eventually) useful discussions about theorems, the fruit of which is on pages 94 and 241. Thanks also to John Kay for his patience in making sense of this gallimaufry, and to Andy Smith for his unexplainable belief in me.

Thanks to my Mum and Dad, family, and all my friends, for their encouragement and patience, for putting up with my "top secret project", and for friendly "research" into competition (through cricket and rugby seasons). Thanks also to Radio Sport and CricInfo for keeping me in touch with the *real* world.

Finally, thanks must go to my speedy typist, and my dedicated C, UNIX, and MATLAB guru and LaTeXnician, for prompt and excellent technical assistance (that's me).

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Operations Research is the art and science of obtaining bad answers to questions for which otherwise worse answers would be given.

The field of vehicle routing is continually expanded by advances in computing technology. Practical dynamic vehicle routing problems—in which online planning must be able to respond to real-time changes in demands of customers, vehicle breakdowns, traffic flow patterns, and other dynamic elements—are now becoming computationally feasible. This chapter introduces dynamic competition between independent players over a set of customers whose demands (or rewards) are known; uncertainty in route planning is due to the unpredictability of opposing players rather than the customers.

1.1 The Field: Paths, Routes, Schedules and Games

The field of this research is a combination of efficient *path planning* and *tactical planning* by autonomous decision makers in competition. We sketch the context for this thesis with an overview of these existing fields, highlighting the relevant problem models that we will later use to design a new and rich class of problems.

1.1.1 Efficient Path Planning

Planning how to efficiently get "from A to B" can involve many difficult decisions. Following Shah [194], we delineate three general problem classes in path planning: traversing a maze or obstacle course; finding a shortest path; and determining an efficient sequence of required stops.

Path planning problems are concerned with detailed prescription of the exact movements of the object and are common to problems in terrain navigation and robotics (Mitchell [160]). For example, Rowe [188] considers planning a path for a mobile agent across a two-dimensional terrain consisting of an isotropic background region, roads (narrow, low cost, transportation corridors), rivers (narrow features of high crossing cost), and untraversable obstacles. A particularly well known task-level problem in computational robotics is called **FindPath** (Lozano-Pérez [148]). The problem is to determine if a robot system be moved from one configuration to another without colliding with obstacles. Donald [57] edited a special issue of *Algorithmica* on geometrical computational robotics in which many algorithmic problems in computational robotics also fall into the field of Computational Geometry (see, e.g., O'Rourke [170]).

Algorithms for solving various shortest path problems were an early computational test of computer technology. These problems were tackled by such distinguished researchers in the field as Dijkstra [56], Floyd [69] and Lawler [141]. Dreyfus [58] provides a good comparison of these early methods which have since been the subject of incremental improvements in computational complexity.

Routing problems, however, are concerned with sequencing a number of required intermediate customer locations, selecting the most economical from a large number of alternatives. The archetypal routing problem is the **Travelling Salesman Problem** (TSP), whose task is to determine the least cost sequence, departing from a depot, through a set of customers and returning to the depot. A feasible tour is a path that starts at the salesman's depot and visits each city exactly once before ending back at the depot. Lawler, Lenstra, Rinnooy Kan and Shmoys [142] is the authoritative introduction to the TSP, including a discussion the history of the TSP. Laporte [134] surveys a large number of solution methods for the TSP. Orloff [169] and Lenstra and Rinnooy Kan [144] define a **General Routing Problem** (GRP), based on a network, in which the objective is to find a minimum length tour containing a subset of required nodes and a subset of required arcs.

1.1.2 Distribution Management

Distribution Management is concerned with the whole process of storage, trunking and, finally, delivery of goods to customers (Eilon, Watson-Gandy and Christofides [63]). Savelsbergh [189] concisely describes the decision problems associated with distribution management:

> "Distribution management presents a variety of decision making problems at the three levels of strategic, tactical and operational planning. Decisions relating to the location of facilities (plants, warehouses or depots) may be viewed as strategic, while problems of fleet size and fleet mix determination can be termed tactical. On the operational level, two problems prevail: the routing of capacitated vehicles through a collection of customers to pickup or deliver goods, the **Vehicle Routing Problem**, and the scheduling of vehicles to meet time or precedence constraints imposed upon their routes, the **Vehicle Scheduling Problem**."

Complex models often consider two or more components of distribution management in tandem. Examples include:

Inventory/Routing is the combination of managing the stochastic inventory level at each customer location and planning the delivery routes over time (Golden, Assad and Dahl [85]).

Location/Routing is the combination of location of the primary facilities (production plants) and secondary facilities (warehouses and depots) and planning the pickup/delivery routes they service and the deliveries from the plant to minimise both the warehouse costs and the distribution costs (Laporte [133]).

Allocation/Routing involves determining a set of routes for a fleet of vehicles over a multiple day time horizon in which it may be necessary to deliver to particular customers more than once (Ball [12]).

Covering/Routing involves determining a set of vehicle routing such that some reward is gained from travelling to customers *or near to* customers, i.e., customers near to a vehicle route are serviced from the nearest on-route customer (Beasley and Nascimento [15]).

1.1.3 Vehicle Routing and Scheduling Problems

A carrier company is to distribute a commodity to a set of customers, each having a known demand and delivery window, using a fleet of vehicles, each of known capacity, domiciled at a single depot. This is the basic problem setting of the **Vehicle Routing and Scheduling Problem** (VRSP) which is to determine a set of vehicle routes and schedules, where each vehicle is to visit some subset of customers, to optimise some performance or customer satisfaction objective. Although the TSP is generally recognised as the foundation for routing problems, it is the VRSP that is the basic, practical problem in the field of Distribution Management. The VRSP can be regarded as a temporally constrained, capacitated, multiple salesman TSP. The VRSP was originally proposed, almost four decades ago, by Dantzig and Ramser [48]. Since then, researchers have modelled many of the real-world attributes, constraints and limitations of a capacitated vehicle fleet and the varied requirements of customers. Christofides, Mingozzi and Toth [40], Bodin, Golden, Assad and Ball [22] and Golden and Assad [87] survey a variety of vehicle routing models including such practical features such as time windows, pickup and delivery, different vehicle configurations, multiple depots, location-routing and inventory-routing and Laporte [135] surveys some solution methods employed.

1.1.4 Dynamic and Real Time Route Planning

A decision problem is **static** if all the information required to *solve the problem* is available before beginning the physical execution of the solution. Many vehicle routing problems are modelled as static decision problems. However, when the information required is unavailable or imprecise, a **dynamic** decision problem may be a more appropriate model. There are a number of reasons why all the information is not initially available.

Temporal Information. Suppose we have a vehicle dispatching system where not all of the demands are known at the beginning of the day. We cannot stop the vehicles from being dispatched until all the demands are known and hence we must update the scheduling plan throughout the day.

Precision of Information. Some information may be known (perhaps only stochastically) at the time of planning but may become more or less precise with time. An example of this is that when travel times are only known approximately but when a vehicle reaches a city (demand location) we know precisely what the arrival time is. The rapidly reducing cost of global positioning systems (GPS) provides affordable real-time vehicle location information.

State Information. In some systems we may not be able to determine the consequences of our actions upon the subsequent state of the system. This is information we cannot anticipate nor have expectations upon.

Dror and Powell [60] claim that "real-time decision making is putting more emphasis on formulating and solving dynamic models." The **Dynamic Vehicle Routing and Scheduling Problem** (DVRSP) is to determine a tentative vehicle routing schedule through the currently known set of customers, over a planning horizon, given the current state of information including those parts of previous schedules actually executed (Savelsbergh [189]). A DVRSP requires a solution *policy* which is *on-line*, i.e., is responsive to changes in information content or quality and a rolling planning horizon. Also, the carrier may decide to service only a subset of the customers. Realistically sized DVRSPs are now becoming computationally feasible and are being applied to practical problems with real-world constraints (Savelsbergh and Sol [190]). Practical vehicle routing problems are inherently dynamic and static problems can only ever model a snapshot of currently held information.

1.1.5 Games

Game Theory is the study of the behaviour of decision makers whose decisions affect one another. Since von Neumann and Morgenstern [203] put game theory on a firm mathematical foundation, game theory has been applied to a large variety of problems in economics, management, psychology, sociology and the military.

Recreational games have fascinated people for centuries. Many common games have been solved exactly by game theoretic analysis (Shah [194]) including Connect 4, Gomoku (see, e.g., Yakowitz [208]), 3-D Tic-Tac-Toe, Nine Men's Morris. In some games (e.g. Hex) we know who will win if the game is played perfectly (the game theoretic value), but not how to play perfectly.

The application of computers to game playing has parallelled the development and technological advances of computers and the field of **Artificial Intelligence** (AI). Pearl [174] states that

> "The skill of playing games such as chess, checkers, or GO has long been regarded as a distinctive mark of human intelligence. It is natural, therefore, that the general public continues to monitor the success of game-playing programs as a measure of progress in artificial intelligence."

In 1996, and again in 1997, popular public attention focussed on the game of **Chess** with the *ACM Chess Challenge* between world chess champion Garry Kasparov and the IBM supercomputer Deep Blue (IBM [106, 107]). The 1996 Philadelphia match was organised by the Association for Computing Machinery (ACM) to mark the 50th birthday of the first computer. Kasparov won

the six game series by three games to one with two draws. This was the first occasion in which a reigning world chess champion had lost to a chess game to a computer. The 1997 New York rematch was won by Deep Blue by two games to one with three draws. The public fascination with these events is evident from an article in Time Magazine (Krantz [132]).

The ancient Chinese game of Go is much more computationally difficult than chess. With the success of Deep Blue in brute force game tree searching, Go is seen as the next challenging test-bed for research in AI since a more intelligent heuristic approach is required.

Game theory offers well defined solution concepts for problems involving decision makers whose objectives either wholly or partially conflict. Hence it is both natural and necessary to employ these established concepts in the course of this thesis.

1.2 The Proposition: Motivation for Competition Routing Problems

A practical reality of a market economy is that suppliers face competition over the customers to whom they provide goods or services. In a standard logistics system, the customer selects a supplier, based upon the price and quality of the good or service, and the customer collects the good from the supplier, or the supplier delivers the good to customer, or both. Suppose, however, that the suppliers may entice a customer away from another supplier by earlier delivery of the good or service and that, in all other respects, each customer is indifferent between suppliers. Then there is the real possibility of competition between the suppliers for customers based upon the performance of the associated carriers.

A **Competition Routing Problem** (CRP) involves at least two independent carrier companies, a fleet of vehicles (each of know capacity and either privately owned by each company or available for hireage) and a set of customers, each of known demand, require the delivery of a homogeneous commodity. If the carriers were to completely cooperate towards a common objective then we obtain a standard vehicle routing problem, with effectively one decision maker. However, if the carriers have private objectives which partially conflict with those of the other carriers, then no one decision maker completely controls the final result. Private objectives of a carrier are known only to that carrier but may include a component objective common to an group of carriers.

We can now approach the classical VRSP in a new way: a single carrier, for which each vehicle driver as an independent decision maker, who completely cooperates with the central controller, who acts on behalf of all the drivers. Complete cooperation is often relaxed in practice by distributing some of the decision making to the driver, e.g., given an allocation of customers to the vehicle, the driver dynamically adjusts his schedule depending upon traffic congestion. Consider also the situation of owner/drivers in which the vehicle driver has a financial stake in the schedule allocated to the vehicle and the efficiency of implementation of the schedule. We can view a CRP as a classical routing problem with an additional complicating competition component or as a relaxation of the classical routing model in which individual vehicles and alliances of vehicles take on more autonomy and have partially conflicting *private* objectives.

The CRP is novel in several ways. Firstly, although the literature addresses competition in a number of traditional Operations Research problems, to our present knowledge it is yet to considered for vehicle routing type problems, with the exception of Johnston [112], Johnston and Giffin [113, 114], Boyd, Clarke, Gemmell and Miller [26], Holland [101] and Fekete and Schmitt [66]. Secondly, we begin with no understanding of the the balance required between strategic considerations and efficient routing to find an effective solution policy.

Note that there is a field of study also called *competitive routing* that concerns the routing of packets in a computer network (Awerbuch, Azar, Plotkin and Waarts [6]). The problems are primarily concerned with networks of queues, robustness with respect to network failures and maximisation of packet throughput. These competitive routing problems bear little resemblance to what we are considering.

1.3 The Challenge: Research and Thesis Overview

The first research challenge is to model the foundational components of a CRP. The second research challenge is to identify and analyse the fundamental principles which strategies must address, both routing and strategic components, and hence formulate and evaluate an effective and efficient solution paradigm for the class of CRPs.

Dror and Powell [60] identify three dimensions important to research in stochastic and dynamic models in transportation: application area, technical issues, and the degree of optimization or robustness required. The scope and structure of the remainder of this thesis is usefully defined by a paraphrase and reinterpretation of these dimensions. Firstly, model formulation must address important real-world characteristics. Secondly, technical details and difficulties must not be "swept under the carpet". Thirdly, evaluation must be address both efficiency and efficacy for the intended purpose. To these ends, the remainder of this thesis is composed of the following three parts.

Part I: Foundations. Formulates a reference model for CRPs by synthesising and building upon the relevant literature on subset selection, dynamic routing, and game theory. This provides a framework within which CRPs can be modelled and insights can be extrapolated. This first research challenge is to model the foundational components of a CRP. Dror and Powell [60] list a number of stochastic dynamic transportation problems which require additional research. These problems incorporate uncertainty due to customers demands or traffic flows, real-time decision making, coordination of large vehicle fleets, dynamic route choice and cost allocation for competitive pricing. None of these problems, however, addresses dynamic uncertainty from competition. For further computational study, we define a simple, core problem which encapsulates the primitive elements of competition and routing. We also identify essential solution principles and design a solution architecture which provides a backbone for the proposal of strategies.

Part II: Strategies. Goes into an analysis of problem instances of various sizes and natural structures, modelling of $\mathcal{NP}$-hard combinatorial subproblems as they arise and the technically difficult implementation of components for the proposed solution architecture.

Part III: Computational Evaluation. Proposes several classes of problem instances and attempts to computationally evaluate these difficulty of the problem instances and the success of previously designed strategies against robustness and effectiveness criteria. Dror and Powell [60] stress the need for an experimental evaluation of the balance between an attractive model and computational tractability. Finally, we draw some conclusions on the effectiveness of strategies and on the foundations of competition routing, and propose recommendations for future research programme that would build upon this thesis work to further define and expand a new field encompassing competition and routing.

Part I

Foundations

Overview of Part I
Foundations

Part I is concerned with establishing the foundations for Competition Routing Problems. This involves synthesis of the relevant literature, problem formulation and formulation of a solution paradigm.

Chapter 2 provides a Reference Model for Competition Routing Problems (RM-CRP), reviews the relevant literature and formulates a core problem: the Competitive Prize Collection Problem (CPCP). Note that only problem modelling is discussed and there is no attempt to classify solution methods.

Chapter 3 specifies the requirements of a strategy for the CPCP and introduces some elementary strategies which adapt existing combinatorial path planning heuristics. It then derives a set of tactical concepts which a strategy should address to be successful.

Chapter 4 presents a (hierarchical) Strategic Planning Architecture (SPA) framework for the study of four highlighted tactical concepts: aggregation, contingency planning, cognizance of opponent and dynamic response monitoring.

CHAPTER 2

Problem Modelling

Everything should be as simple as possible, but not simpler.

— ALBERT EINSTEIN

The foundations of competition routing need to be laid. We have only the idea of multiple independent carrier companies in competition to deliver a commodity to a set of customers. This chapter draws from the established literature on vehicle routing and game theory to scope and explore the possible modelling choices. Before embarking on the proposal of computational strategies, one needs a clear definition of the problem model. What concrete problem will shed light on the foundational field and be able to be extrapolated across other problems?

2.0 Introduction

In this chapter we design a model for competition routing problems by combining ideas from the two fields of vehicle routing and game theory. Although we have briefly sketched these fields in the previous chapter, it is assumed that the reader has some familiarity with vehicle routing problems and basic principles of game theory. Lawler, Lenstra, Rinnooy Kan and Shmoys [142] and Fudenberg and Tirole [72] provide a good introductions to the TSP and game theory respectively.

The first objective of this chapter is to address the first research challenge of Section 1.3, i.e., to model the foundational components of a CRP and determine how competition fits into

the field of vehicle routing. To achieve this we firstly assemble the building blocks by surveying the relevant literature on subset selection routing problems (Section 2.1), vehicle routing and scheduling problems (Section 2.2), dynamic and real-time decision problems (Section 2.3) and game theoretic problems (Section 2.4). The emphasis here is on describing the classes of problems and how they fit together rather than enumerating every nuance in formulation. In particular, we do not consider solution methods although we do consider some solution concepts. Secondly, Section 2.5 synthesises these building blocks into a descriptive framework for the traditional VRSP and subset selection, dynamic and competition variations which sketches a broad extension of the present field of vehicle routing.

The second objective of this chapter is to formulate a simple problem for computational investigation which is at the core of most of these CRPs, in the sense that we need to know something about the kind of successful behaviour and strategy on this specific problem in order to say anything about a more complex or practical problem. We also need to formulate a core set of questions which may shed light on the broader field. Towards this second objective, Section 2.6 determines a set of minimal specifications for each component of a CRP and formulates the **Competitive Prize Collection Problem** (CPCP) as a core CRP for further computational investigation. Finally, Section 2.7 enumerates a research question strategy for investigating the CPCP and relating insights and conclusions back to the CRP.

2.1 Combinatorial Subset Selection Routing Problems

A **Combinatorial Optimization Problem** (COP) is a problem which requires the determination of a *best solution* from a *finite* set of *feasible solutions* according to some *objective function* which gives a value to each feasible solution. A feasible solution is usually some combinatorial arrangement of elements of some set, e.g., a sequence of customers in the case of the TSP. Although simply stated, the TSP is a prototype of a difficult COP since, in the worst case, a solution method may need to consider all feasible solutions to find the best solution.[1] However, solution concepts from the TSP are often applied to other COPs, e.g., in a VRSP solution method as a subroutine to determine the routes of the individual vehicles.

The **Travelling Salesman Subset-tour Problem** (TSSP) differs from the TSP in that the salesman is not required to visit every city (Mittenthal and Noon [161]). A salesman collects a **prize** (or reward), v_j, in every city j that he visits and pays a fixed **penalty** (or isolation cost), π_k, for every city k that he fails to visit. The salesman travels between cities i and j at cost c_{ij}. Such a subset-tour (or **subtour**) may vary from visiting no cities to visiting all the cities. The salesman must determine both which subset of cities to visit and also in which order to visit those cities; these are two interdependent decisions.

The **Travelling Salesman Subset-tour Problem with One Additional Constraint** (TSSP+1) is just that, a TSSP with a single constraint. Noon, Mittenthal and Pillai [168] and Bowers, Noon and Thomas [25] have applied a TSSP + 1 as a subproblem within a decomposition scheme for solving the classical capacity-constrained VRP. Within a capacity-constrained VRP

[1] This is the so-called *Fundamental Algorithm of Combinatorial Optimization* (Foulds [70]).

the driver's problem is modelled as a TSSP+1. The TSSP+1 decomposition is based on a Lagrangian relaxation and is capable of producing valid lower bounds for the VRP. The best bound attainable is shown to be at least as good as the bound obtained by solving the linear programming relaxation of the classic set partitioning formulation of the VRSP. Pekny and Miller [175] define the **Resource Constrained Travelling Salesman Problem** (RCTSP) which is equivalent to the TSSP+1. They describe an application to scheduling sequence dependent transition costs with respect to an aggregate due date and design a parallel algorithm for problem instances comprising up to 200 cities.

More generally, Beasley and Nascimento [15] consider the possibility of gaining reward by allocating some customers not directly visited by the salesman to some nearby customer who is on the salesman's tour. They provide a framework formulation, the **Single Vehicle Routing Allocation Problem** (SVRAP) for the TSSP class of problems that also incorporates various problems with a covering component. Three types of selection occur: on-route customers, who contribute a fixed reward; off-route (allocated) customers who contribute a benefit dependent upon the on-route customer it is allocated to; and isolated customers who contribute a fixed penalty cost. Not all customers need be visited by the vehicles but customers not visited are either allocated to some customer on one of the vehicle routes (covered) or left isolated. Also, customers may have a preallocated type. The SVRAP also generalizes such problems as the **Covering Tour Problem** (CTP) of Gendreau, Laporte and Semet [77], the **Shortest Covering Path Problem** (SCPP) of Current, Pirkul and Rolland [45], and the **Covering Salesman Problem** (CSP), **Median Tour Problem** (MTP), and **Maximal Covering Tour Problem** (MCTP), of Current and Schilling [46, 47].

The TSSP is characteristic of a class of problem which has evolved in the literature over the last decade or so, which we call **Combinatorial Subset Selection Routing Problems**. We now survey the growing literature for this problem class by categorising the problem models as unconstrained, reward constrained, cost constrained, multiple salesman and time dependent.

2.1.1 Unconstrained TSSP

Keller [126] and Keller and Goodchild [127] propose the **Multiobjective Vending Problem** (MVP): to find all TSSP solutions, S, such that the multiple objective $(v(S), \ell(S))$ is *Pareto optimal* with respect to maximizing $v(S)$ and minimizing $\ell(S)$, i.e., find all those solutions such that there is *no* solution which simultaneously has greater value and shorter length. The MVP also generalizes problems involving shortest paths where each path is associated with two objectives (see Henig [97]).

Another unconstrained TSSP, the salesman wishes to minimise the sum of his travel costs and isolation costs. We call this the Bienstock variation of the **Prize Collecting Travelling Salesman Problem** (Bienstock-PCTSP) considered by Bienstock, Goemans, Simchi-Levi and Williamson [20], Williamson [206] and Goemans and Williamson [83]. Volgenant and Jonker [202] study the same problem, calling it the **Generalized Travelling Salesman Problem** (GTSP),[2]

[2]There are many problems called the Generalized Travelling Salesman Problem (GTSP) in the literature, each a generalization of the TSP.

and a special case called the **Shortest Path Problem with Specified Nodes** (SPPSN) in which a specified set of nodes must be visited exactly once, remaining nodes at most once and large penalties are incurred for not visiting the specified nodes. They give a transformation for the GTSP into an asymmetric TSP with twice the number of vertices and four times the number of edges.

Malandraki and Daskin [152] define a **Maximum Benefit Travelling Salesman Problem** (MBTSP) in which the requirement for a city to be visited at least once is further relaxed. A city may be visited any number of times or not at all, each visit accruing a diminishing additional reward. The objective is to maximize the net profit (reward value collected less travel costs). Malandraki and Daskin also describe a similar variation of the **Chinese Postman Problem** (CPP) in which the requirement for each edge to be traversed at least once is relaxed and a benefit is derived from every traversal of an edge. This is called the **Maximum Benefit Chinese Postman Problem** (MBCPP) which finds a tour of maximum total net benefit. An edge may be traversed more than once or not at all, for decreasing benefit for each traversal. They also consider the multiobjective MBCPP of maximizing the route's benefit and minimizing the route's length.

2.1.2 Reward Constrained TSSP

A **Reward Constrained TSSP** is a TSSP constrained by a single constraint on the sum of prize values on the subtour.

The salesman wishes to minimise the sum of his travel costs and isolation (penalty) costs whilst including in his tour enough cities to collect a prescribed amount v_{min} of prize money. The salesman departs from a given depot and returns to the depot at the end of the subtour. We will call this the Balas variation of the **Prize Collecting Travelling Salesman Problem** (Balas-PCTSP) considered by Aneja and Punnen [4], Awerbuch, Azar, Blum and Vempala [5], Balas [10, 11], Dell'Amico, Maffioli and Värbrand [51] and Fischetti and Toth [68]. The Balas-PCTSP was originally formulated for scheduling the daily operation of a steel rolling mill but has found numerous applications in routing and machine scheduling.

Hamacher and Moll [95] consider a special case of the Balas-PCTSP in which all the prize values are equal, naming it the **Travelling Salesman Selection Problem** which we will call the Hamacher-TSSP. Existing heuristics are based on approximations for the **k-Minimal Spanning Tree Problem** to find the node cluster containing the shortest subtour satisfying the requirement to visit the given number of cities. The Hamacher-TSSP does not include a specific depot that has to be visited by the subtour.

2.1.3 Cost Constrained TSSP

A **Cost Constrained TSSP** is a TSSP constrained by a single constraint on the cost or length of the subtour.

Gensch [78] studies an industrial problem in which the salesman wishes to maximize the net profit (reward value collected less travel costs) whilst not exceeding a prescribed travel cost budget. Gensch names this the **Travelling Salesman's Subtour Problem** for which we will

reserve the acronym (Gensch-TSSP). Kataoka and Morito [125] comment that Gensch's algorithm is not strictly an algorithm (but a heuristic) because it does not necessarily find the optimal solution to one of the component subproblems.

Golden, Levy and Dahl [86] consider a generalization of Gensch-TSSP problem in which there is a net profit, ρ_{ij}, associated with each edge and the salesman wishes to maximize the total net profit over the edges traversed whilst not exceeding a prescribed travel cost budget. This is called the **Time Constrained Travelling Salesman Problem** (TCTSP).[3]

The most popular TSSP+1 studied in the literature is the **Orienteering Problem** (OP), in which the salesman wishes to maximize the reward value collected without exceeding a prescribed travel cost budget, visiting each city at most once. The motivation for this problem is the sport of Orienteering, concisely described by Chao, Golden and Wasil [35]:

> "Orienteering is an outdoor sport usually played in a mountainous or heavily forested area. Armed with a compass and map, competitors start at a specified control point, try to visit as many other control points as possible within a prescribed time limit, and return to a specified control point. Each control point has an associated score, so that the objective of orienteering is to maximize the total score. Competitors who arrive at the finish point after time has expired are disqualified, and the eligible competitor with the highest score is declared the winner. Since time is limited, competitors may not be able to visit all control points. The competitors have to select a subset of control points to visit that will maximize their total score subject to the time restriction."

Hayes and Norman [96] model a real world orienteering event, the 1974 Lake District Mountain Trail, in England, as a dynamic program, to compare optimal paths against the actual routes selected by participants. They also consider the design of the course and the siting of the control points. They do not, however, formulate a combinatorial optimization problem.

Tsiligirides [200] appears to be the first to consider the combinatorial optimization problem formulation now known as the OP, although Tsiligirides called it a **Generalised Travelling Salesman Problem**, i.e., yet another GTSP. Golden, Levy and Vohra [89] were the first to coin the name *Orienteering Problem*. They compared a number of stochastic and deterministic subtour construction and improvement heuristics with those proposed by Tsiligirides. Golden, Wang and Liu [91] improved these heuristic ideas with a multifaceted heuristic including centre of gravity improvement, randomness, subgravity and a learning capability. Keller [126] adds a heuristic for the MVP (but restricted to the OP) to these computational comparisons. Ramesh and Brown [182] propose another heuristic for the OP employing local subtour operations including insertions, deletions and improvements. Wang, Sun, Golden and Jia [205] modify a continuous Hopfield neural network to find solutions for the OP. The neural network finds an initial feasible solution which is then improved using traditional two-exchanges and cheapest insertion. Chao, Golden and Wasil [35] develop a local search heuristic for the OP and compare their heuristic

[3]Note that Baker [9] and Kindervater, Lenstra and Savelsbergh [128] consider problems called the **Time Constrained Travelling Salesman Problem** both of which are equivalent to a **Travelling Salesman Problem with Time Windows** (TSPTW), not a cost constrained TSSP.

with a number of these other authors' heuristics and also some algorithms, concluding that their heuristic is computationally efficient and outperforms most other heuristics on a set of 107 test problems.

Sokkappa [196] also studied the OP, but called it the **Cost Constrained Travelling Salesman Problem** (CCTSP). Golden, Levy and Vohra [89] shows that OP is $\mathcal{NP}$-hard (see Garey and Johnson [73]) but Sokkappa proves that no K-approximation algorithm or fully polynomial approximation scheme exists for the OP, unless $\mathcal{P} = \mathcal{NP}$. Awerbuch, Azar, Blum and Vempala [5] propose the **Bank Robber Problem**, which turns out to be yet another name for the OP. More importantly they are able to provide a polylogarithmic performance guarantee for the OP and the Balas-PCTSP.

Algorithms have also been proposed to find optimal solutions to the OP. Kataoka and Morito [125] introduced the **Maximum Collection Problem** (MCP), which is equivalent to OP, and proposed a branch and bound algorithm with an **Assignment Problem** (AP) relaxation. Laporte and Martello [138] provide an integer linear programming formulation of the **Selective Travelling Salesman Problem** (STSP), also equivalent to the OP, and propose simple greedy heuristics, upper and lower bounding procedures and a branch and bound algorithm. Ramesh, Yoon and Karwan [183] design a branch and bound algorithm using a Lagrangean relaxation solved by a degree-constrained spanning tree procedure. Leifer and Rosenwein [143] contribute a number of strong linear programming relaxations for the OP by adding a sequence of valid inequalities. They determine upper bounds on the optimal value by solving three successive linear programs.

A capacitated TSSP is also possible. Diaby and Ramesh [55] consider the **Distribution Problem with Carrier Service** (DPCS). Each location has a certain demand, the distribution vehicle has a load capacity, and the entire operation should be completed within a certain time. An outside carrier is available for direct service of locations from the central facility. The problem is to determine a feasible tour for the company vehicle and the locations to be served by the outside carrier such that the total cost of the operations is minimised. The features of this problem are feasibility with respect to vehicle load as well as the travel time constraint, penalty costs for not visiting a customer and no rewards.

Finally, Millar [156] and Millar and Kiragu [157] present a novel application of the OP to a fisheries patrol problem in the Scotia-Fundy region of the Atlantic Coast of Canada. The OP serves as a static snapshot of a more dynamic problem; the prize values are used to approximate urgency and importance criteria.

2.1.4 Multiple Salesman TSSP

In the **Multiple Travelling Salesman Problem** (MTSP), m salesman must start from a depot, each visiting a number of prizes and returning to the depot, such that every prize is visited by at least one salesman and the sum of the distances travelled by the salesmen is minimised. The VRSP, then, is simply a capacitated MTSP. We can similarly define multiple salesman versions of the TSSP.

Chao, Golden and Wasil [36] formulate the **Team Orienteering Problem** (TOP). In this

problem, there are m orienteers who compete as a team to maximize the sum of prize values collected by the team members subject to a common time limit. Hence there are three interdependent decisions: which prizes to visit, for each prize which team member to allocate to that prize, and for each team member what sequence of allocated prizes to follow. The paper modifies the local search heuristic developed for the OP in Chao, Golden and Wasil [35] and modifies the stochastic algorithm (heuristic) of Tsiligirides [200].

Butt and Cavalier [31] and Butt and Ryan [30] look at the **Multiple Tour Maximum Collection Problem** (MTMCP), which is equivalent to TOP. Butt and Cavalier [31] provide an integer programming formulation and propose a local search heuristic, whereas Butt and Ryan [30] propose a branch and bound algorithm for finding optimal solutions.

Golden, Assad and Dahl [85] apply the TCTSP of Golden, Levy and Dahl [86] in the context of large scale vehicle routing with an inventory component. The goal of the distribution system is to maintain an adequate level of inventory for all customers. A profit is attached to every customer depending on the urgency of resupplying the customer, which is a nonlinear function of proportion of remaining tank level. The three interdependent decisions of TOP can easily be seen in this application. Customer selection involves identification of the customers to be serviced on a particular day and is intimately related to the inventory component of the problem. Customer–vehicle assignment involves the assignment of customers for service on a particular day to one of the trucks. Finally, routing involves the construction of efficient routes for each truck over the set of its assigned customers.

2.1.5 Time Dependent TSSP

Three forms of **Time Dependent TSSP** have been investigated in the literature: time dependent rewards, time windows and time dependent travel times.

Brideau and Cavalier [28] and Erkut and Zhang [64] look at the **Maximum Collection Problem with Time Dependent Rewards** (MCPTDR). In particular they include service times at the prizes and prize values of the form

$$v_i(t) = \max\{v_i - s_i t, 0\} \tag{2.1}$$

for prize i at time $t \geq 0$ with decay rate $s_i \geq 0$. Brideau and Cavalier attempt to model a CRP by using time-dependent rewards as a static snapshot.

> "A simple example of the MCPTDR involves a salesman in a competitive environment. Consider a salesman who wishes to visit a subset of cities in such a way as to maximize the number of sales made. All potential markets that fall within a restricted radius of travel are candidate cities to be visited by the salesman. From historical data, the salesman knows the potential reward at each city, that is, the number of potential sales that could be made in each market. However, as is common in a competitive market, there are other salesman working the same area, soliciting sales from potential customers. Thus, as the days progress, potential customers purchase goods and services from the salesman's competitors, making the customers unavailable for solicitation. Depending on the number of initial customers and the

> rate at which they make commitments, the number of potential sales decrease in each city until a time is reached when there are no potential sales left to be made. The salesman thus realizes that the longer the delay before visiting a city, the fewer the number of prospective customers remain. The salesman's objective is thus to chose a tour that would maximize the number of potential sales."

This problem indicates that the CRP has real world application for which researchers have proposed models which attempt to capture the competition element without explicitly including dynamic competition in the model.

Problems involving restrictions on when you may visit locations are usually collected together as the class of *Time Window Problems*. For example the **Travelling Salesman Problem with Time Windows** (TSPTW) is the same as the TSP except that each customer may have specified an early time (which you *cannot* visit that location before) and a lateness time (which you *must* visit that location before). Kantor and Rosenwein [124] propose the **Orienteering Problem with Time Windows** (OPTW) in which three types of decision are coordinated: allocation of customers to be serviced; sequencing of customers that are to be serviced; and scheduling of customer deliveries with respect to the time windows at each customer.

Finally, Malandraki and Dial [153] and Malandraki and Daskin [151] consider time dependent travel times in their formulations.

2.2 VRSP Classification

Section 1.1.3 introduced the **Vehicle Routing and Scheduling Problem** (VRSP), namely to determine a set of vehicle routes and schedules, where each vehicle is to visit some subset of customers, to optimise some performance or customer satisfaction objective. In this section we briefly review and critique a description scheme for VRSPs, thus isolating the key components which compose any VRSP.

A **taxonomy** provides a way of structuring existing problems with common features. Current and Min [44] and Current and Marsh [43] provide a problem taxonomy for multiobjective transportation network design and routing problems, which subsumes VRSPs. The taxonomy is based on a hierarchy of major problem type (shortest path, transportation, assignment, transshipment, vehicle routing, optimal network design, spanning tree or network flow), solution technique (algorithm or heuristic) and multiobjective analysis technique. Deo and Pang [53] provide a similar taxonomy restricted to shortest path problems.

We, however, are not concerned with solution techniques for VRSPs but rather with being able to describe the significant features of a particular VRSP or a subclass of VRSPs. Also, we wish to extend a description scheme for VRSPs to one for CRPs and hence the scheme would be used to describe problems which have yet to be defined and, more importantly, to compare the structure of a CRP with a corresponding VRSP.

A **classification scheme** provides a concise notation or language for specifying the salient features of a problem for comparison with other problems. Desrochers, Lenstra and Savelsbergh [54] define a classification scheme for VRSPs and illustrate the scheme with a number of

⟨classification⟩ ::= ⟨addresses⟩/
⟨vehicles⟩/
⟨problem characteristics⟩/
⟨objectives⟩ (2.2)

⟨addresses⟩ ::= ⟨number of depots⟩
⟨type of demand⟩
⟨address scheduling constraints⟩
⟨address selection constraints⟩ (2.3)

⟨vehicles⟩ ::= ⟨number of vehicles⟩
⟨capacity constraints⟩
⟨commodity constraints⟩
⟨vehicle scheduling constraints⟩
⟨route duration constraints⟩ (2.4)

⟨problem characteristics⟩ ::= ⟨type of network⟩
⟨type of strategy⟩
⟨address-address restrictions⟩
⟨vehicle-address restrictions⟩
⟨vehicle-vehicle restrictions⟩ (2.5)

⟨objectives⟩ ::= ⟨objective⟩ ∨ ⟨objective⟩⟨objectives⟩ (2.6)

problems from the literature. Their motivation is to eventually construct a system to manage VRSP models and associated solution methods. They adopt a *formal grammar*, consisting of four fields, to precisely specify the classification language.

> "A number of vehicles, stationed at one or more depots, have to serve a collection of customers in such a way that given constraints are respected and a given objective function is optimised. To define one such problem type in a formal way, our language uses four fields. The first field describes the characteristics and constraints that are relevant only to single addresses (customers and depots). We prefer the term 'address' to 'customer' because of the great variety of customer types: apart from the usual single-address customer, there is also the customer corresponding to an origin–destination pair or to all the addresses located on a street segment. The second field specifies the characteristics relevant only to single vehicles. The third field contains all problem characteristics that cannot be identified with single addresses or vehicles. The fourth field defines one or more objective functions."

Sentence (2.2) shows the four major fields and Sentences (2.3)–(2.6) show the subfields of each major field. These sentences give the top level structure of their classification. We do not intend to reproduce the work of Desrochers, Lenstra and Savelsbergh [54] but rather make some comments which will be useful to formulating the CRP.

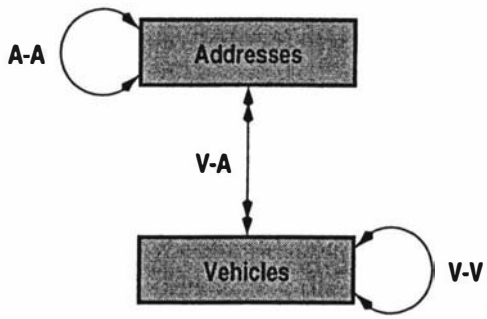


Figure 2.1: Entity-Relationship Model for VRSP

Observation 1. We can identify two entity types, *addresses* and *vehicles*, and three relationship types, *address-address*, *vehicle-address* and *vehicle-vehicle* from the description above. These are illustrated in Figure 2.1. The ⟨**addresses**⟩ and ⟨**vehicles**⟩ fields define the attributes of the addresses and vehicles and the ⟨**address–address restrictions**⟩, ⟨**vehicle–address restrictions**⟩ and ⟨**vehicle–vehicle restrictions**⟩ subfields define the relationships. Hence we have a field for each entity type and a field defining the relationships between the entities.

Critique 1. The ⟨**type of network**⟩ and ⟨**type of strategy**⟩ subfields of ⟨**problem characteristics**⟩ appear logically to be out of place. The ⟨**type of network**⟩ subfield specifies the properties of the underlying network and travel costs. This is an attribute of the addresses and so should logically be a subfield of ⟨**addresses**⟩. The ⟨**type of strategy**⟩ specifies the type of service which is acceptable to both addresses and vehicles as appropriate, i.e, rules for allocation of addresses (customers and depots) to vehicles, e.g., such as splitting of demand and pickup and delivery. This is a component of the relationship between addresses and vehicles and so should logically be incorporated with ⟨**vehicle–address restrictions**⟩.

Observation 2. The token 'o' indicates the empty symbol and is used to indicate a default value, which is usually either the simplest or the most frequently occurring value. Consider the use of the symbol 'o' in the following two examples.

(a). $\langle \delta_1 \rangle ::= o \vee / \vee \div$ in which 'o' stands for "splitting of demand not allowed", '/' stands for "a priori splitting of demand allowed" and '÷' stands for "a posteriori splitting of demands allowed".

(b). $\langle \gamma_1 \rangle ::= o \vee \Delta$ in which 'o' stands for "general costs" and 'Δ' stands for "the costs satisfy the triangle inequality".

Clearly there is a difference in that in case (a) the options appear to be mutually exclusive and 'o' is a default value and in case (b) the options are refinements and 'o' is the most general.

Critique 2. We argue that the latter is preferable for a "classification language" since we should be able to specify classes of problems by not specifying a particular token and the class

contains all problems that satisfy the most general case of that token. Hence the classification 'o/ o / o / o /o' should be the class of all vehicle routing and scheduling problems rather than those with node routing, all deliveries, deterministic demand, no scheduling constraints, etc. Desrochers, Lenstra and Savelsbergh [54] have tried to build a *specification language* in order to be able to describe as tersely as possible the common problems, rather than a more powerful *classification language* which can describe classes of problems.

In Section 2.5 will build upon this classification and include the modifications described above.

2.3 Dynamic Vehicle Routing and Scheduling Problems

Section 1.1.4 introduced the idea of *dynamic* vehicle routing problems. In these problems there is a need to plan tentatively, implementing only the head of the planned vehicle routes, since, potentially at least, new or more precise information is continually arriving or updating. Often a real-time response required.

The *quality* of information is usually good for near-term events and becomes poorer for more distant events. Following Psaraftis [180], at a particular point in time, a problem input may be *deterministic* (if it is known with absolute certainty throughout the duration of the routing process), *probabilistic* (if it follows some probability distribution), *stochastic* (if it evolves by some stochastic process) or may simply have *unknown* value.

Dror and Powell [60] describe the, then, state of current knowledge in stochastic and dynamic models in transportation.

> "Stochastic and dynamic problems are often characterised by the lack of a well-defined objective function, the lack of test data sets, criteria for comparing solutions, and even a standard formulation. Objective functions are not clearly defined, because the formulation of the model is itself an important research area."

Furthermore, the classification scheme of Desrochers, Lenstra and Savelsbergh [54] (reviewed in Section 2.2) does not consider dynamic routing problems although some data is permitted to be stochastic.

2.3.1 Stochastic Problems

It is important not to confuse *dynamic* problems with *stochastic* problems. Elements of both often occur in models of the real world such as that studied by Bertsimas and van Ryzin [18]. However, stochastic problems can also be static, as in the problem considered by Jaillet [108, 109], Jaillet and Odoni [110] and Laporte, Louveaux and Mercure [137] in which an a priori fixed tour through the complete set of customers is required although only a random subset of the customers will eventually have any demand. Gendreau, Laporte and Séguin [76] review the literature on stochastic vehicle routing problems, distinguishing between stochastic demands, stochastic customers and stochastic travel times. A comprehensive overview of stochastic vehicle routing problems and a survey of solution methods can be found in Dror, Laporte and Trudeau [59].

2.3.2 Dynamic Problems

Psaraftis [180] emphasises that the most important consideration in determining whether a problem is static or dynamic is "the way information about a particular routing problem evolves through time and is received by the decision maker". Static inputs are known for the entire duration of the routing process and never require updating, although they may be time dependent. Dynamic inputs will generally be revealed or updated as time goes on. A model must make some assumption as to the sometimes fuzzy characterisation of an input as static or dynamic and its significance in terms of the problem at hand. Note that a dynamic input may also change in *quality* as time progresses.

Psaraftis [179, 180] defines the **Dynamic Travelling Salesman Problem** (DTSP). Demands for service are generated at each node of a network according to a Poisson process of parameter λ, travel times between nodes are known and deterministic and the salesman spends a known service time at each node. The DTSP is to determine a *routing policy* that maximizes the expected number of demands serviced per unit time.

Psaraftis [179, 180] also discusses considerations for general dynamic vehicle routing problems. Lund, Madsen and Rygaard [150] define the **Dynamic Vehicle Routing Problem** (DVRP) in which vehicles are dispatched to satisfy service requests that evolve in real time. Traditionally the solution of such dynamic problems has been based on adaptations of static procedures, i.e., a static VRSP is solved each time an input update has occurred. They argue that the suitability of this approach largely depends upon weak versus strong degrees of dynamism. Bertsimas and Simchi-Levi [17] advocate that the next generation of vehicle routing research must address problems which the uncertainty issues surrounding problem data and provide corresponding robust solution methods.

2.3.3 Real Time Problems

It is important to consider real time decision problems since any real world vehicle routing application that incorporate competition will necessarily involve real time decisions from both dispatchers and drivers.

Séguin, Potvin, Gendreau, Crainic and Marcotte [192] consider the class of **Real Time Decision Problems** (RTDP) in which the objective is to provide responses of a required quality in a continuously evolving environment, within a prescribed time frame, using limited resources and information that is often incomplete or uncertain. They synthesise the literature from AI, dynamic programming and Operations Research through a vehicle dispatching application. Table 2.1 (reproduced from Séguin et al) lists a number of important attributes of RTDPs and demonstrates the diversity of dynamic applications.

Savelsbergh and Sol [190] define the **General Pickup and Delivery Problem** (GPDP) in which transportation requests evolve in real time. They treat the dynamic problem as a sequence of static snapshot problems which must be solved within strict time constraints.

Table 2.1: List of Attributes of RTDPs (reproduced from Séguin et al [192])

Desired solution	Optimal or approximate
Objective	Simple, multiple, conflicting
Response time	Extremely fast to loose, variable or unique
Planning or forecast horizon	Long, medium, short term
Course of action	Unique or multiple (reactive, incremental, deliberative)
Action outcome	Certain or uncertain
Input data	Certain or uncertain, complete or incomplete
Failure recovery mechanisms	Required or not
Future events	Extrapolated, simulated, predicted, ignored
Environment and working conditions	Time invariant to highly time-variant, highly predictable to highly unstable

2.4 Decision Makers and Game Theory

In Section 1.1.5 we considered the popularity of games, especially Chess. In this section we review some concepts of game theory relevant to the remainder of this thesis. We do not attempt to be complete, but rather provide a brief overview of the major concepts we will require.

A **game** is a decision problem involving two or more **decision makers**. The decision makers involved in a game are called **players**. **Game theory** studies the behaviour of players whose decisions affect one another, i.e., where their objective may be in partial or total conflict. When there is only a single player, the corresponding decision problem is a well-defined optimization problem, which may be computationally difficult to solve but only the choices of the single decision maker determine the final outcome. When there are multiple players, no one player completely controls the final outcome, so what is meant by a good decision must be defined before an attempt is made to find one. A game is a description of strategic interaction that includes the constraints on the actions that the players can take and the players' interests, but does not specify the actions that the players *do* take (Osborne and Rubinstein [171]). In particular, a **perfect information** game is one in which each player has complete information about his opponent's position and about the choices available to him (Pearl [174]).

2.4.1 Rationality

Game theorists generally make two basic assumptions about players: that that are rational and that they are intelligent (Myerson [165]). A player is **rational** if he makes decisions consistently in pursuit of his own objectives. A player is **intelligent** if he knows everything that we know

about the game and he can make any inferences about the situation that we can make.

However, Simon and Schaeffer [195] draw the distinction between two forms of rationality. **Substantive rationality** is concerned with choosing the objectively correct or best action in a specified situation, given the objective. **Procedural rationality** is concerned with computational procedures for finding or constructing good actions, taking into account not only the objective and the specified situation, but also the knowledge and the computational capabilities and limitations of the player. The major difficulty in solving our problem is computational difficulty; playing a "good" game involves using the limited available computational resources as effectively as possible. This might mean investing a great deal of computation in examining a few variations, or investing a little computation in each of a large number of variations. Neither approach can come close to exhausting all possible plays of the game, i.e., to achieving substantive rationality.

In an article interesting entitled "How to decide how to decide how to ...: modeling limited rationality", Lipman [146] nicely illustrates the idea of the limited rationality implied by procedural rationality.

> "A person required to risk money on a remote digit of π would, in order to comply fully with the theory of expected utility have to compute that digit, though this would really be wasteful if the cost of computation were more than the prize involved. For the postulates of the theory imply that you should behave in accordance with the logical implications of all that you know. Is it possible to improve the theory in this respect, making allowance within it for the cost of thinking, or would that entail paradox?"

Schoemaker [191] proposes that "strategy, at its core, concerns the development and testing of heuristics for high stake decisions in environments too unstable and complex to be optimized." Also, strategy must incorporate assumptions about the opposing decision makers characterised by variable creativity and rationality, information and resource asymmetry, and differences in history and reputation.

Osborne and Rubinstein [171] assume that each decision-maker is rational in the sense that he is aware of his alternatives, forms expectations about any unknowns, has clear preferences, and chooses his action deliberately after some process of optimization.

2.4.2 A Little Two-Person Game Theory

We are primarily concerned with games involving only two players. Two-person game theory is divided into **constant sum games**, in which the sum of the payoffs is constant over every pair of player actions, and **general sum games**, in which the sum of the payoffs need not be constant over pairs of player actions. Competition is **perfect** in a constant sum game since any payoff which one player does not receive must be received by the other player. However, in a general sum game, competition is not perfect. General sum games are further subdivided into **noncooperative**, in which any type of collusion, such as correlated strategies and side payments, is forbidden, and **cooperative**, in which all such cooperation is permitted. A noncooperative game focuses on sets of possible actions of individual players whereas a cooperative game (or

coalition game) focuses on the sets of possible joint actions of groups of players (Osborne and Rubinstein [171]).

Following Osborne and Rubinstein [171], a **strategic form game** consists of a finite set of **players** and for each player a nonempty set of **actions** and a preference relation on the set of all combinations of actions by the players. A game in **normal form** is the familiar representation as a game table or matrix. An **extensive form game** is an explicit description of the sequential structure of the decision problems encountered by the players in a strategic situation. The model allows us to study solutions in which each player can consider his plan of action not only at the beginning of the game but also at any point of time at which he has to make a decision. By contrast, the model of a strategic form game restricts us to solutions in which each player chooses his plan of action once and for all; this plan can cover unlimited contingencies, but the model of a strategic game does not allow a player to reconsider his plan of action after some events in the game have unfolded.

2.4.3 Games in Operations Research and Management Science

Kirkwood [130] responds to comments by leading operations researchers that Operations Research does not address strategic business or government issues, arguing that "decisions in most organisations form a continuum running from routine operational decisions to major, top level strategic decisions impacting overall organisational direction."

Wang and Parlar [204] give an overview of static game theory applications in management science including examples from production and inventory management, bidding and auctions, marketing, queueing and finance. They note that "some traditional management science areas such as inventory and queueing have not attracted the same amount of attention as others such as marketing and bidding."

Curiel, Pederzoli and Tijs [42] and Potters, Curiel and Tijs [176] consider the problem of how to divide the total cost of a round trip along several institutes among the institutes visited, introducing two types of cooperative travelling salesman games. Similarly, Göthe-Lundgren, Jörnsten and Värbrand [92] consider the allocation of routing costs for each vehicle between the customers allocated to that vehicle in a VRSP.

2.4.4 Dynamic Pursuit, Evasion, Hide and Seek

Various dynamic games have been considered in the literature involving moving players. These can be broadly classified into two types of games: pursuit-evasion games and hide and seek games.

In a **pursuit-evasion game**, the players observe each other perfectly and the pursuer (or team of pursuers) must catch up to the evader by coming within some prescribed distance. Pursuit-evasion games are often modelled as differential games in the Euclidean plane (Chikrij and Glushkov [38]) or grid structures (Dawes [49]). Owen [172] defines a **differential game** as one in which the time interval between stages decreases, until, in the limit, a game is reached in which each player must make a move at each moment in time.

In a **hide and seek game**, the seeker cannot observe the hider until they occupy the same location. Hide and seek games are often played on a graph or network structure and are modelled

as static probabilistic problems. For example, Cao [33] and Cao and von Stengel [34] consider the problem of hiding an object in a graph or tree and a mobile searcher, Anderson and Aramendia [3] and Reijnierse and Potters [186] consider an immobile hider and Thomas and Washburn [199] propose a dynamic search games between an intelligent searcher and an intelligent target moving around a grid.

2.4.5 The Prisoner's Dilemma

Axelrod [7, 8] investigates the question of "when should a person cooperate, and when should a person be selfish, in an ongoing interaction with another person" by looking at a simple game called the iterated Prisoner's Dilemma. The following definition is almost verbatim from Axelrod [7].

Definition 2.4.1

The **Prisoner's Dilemma** is a two player game in which each player can either cooperate or defect. If both cooperate, both get the *reward* r. If both defect, both get the *punishment* p. If one cooperates and the other defects, the first gets the *sucker's* payoff, s, and the other gets the *temptation*, t. The payoffs are ordered $t > r > p > s$, and satisfy $2r > t + s$. The corresponding game matrix is as follows.

		Player $\mathcal{B}$	
		Cooperate	Defect
Player $\mathcal{A}$	Cooperate	(r, r)	(s, t)
	Defect	(t, s)	(p, p)

□

In the **Iterated Prisoner's Dilemma**, the game is played an infinite number of times and each iteration is worth less than the previous iteration by a factor of ω, where $0 < \omega < 1$. Both player's objective is to maximize the cumulative payoff.

The Prisoner's Dilemma is a useful example of a simple game which has been investigated computationally by playing off a number of strategies against one another in a tournament setting.

2.5 Reference Model for Competition Routing Problems

This section proposes a **Reference Model for Competition Routing Problems** (RM-CRP). We do not intend to duplicate the modelling of a VRSP, but rather describe those attributes and relationships which are implied by subset selection, dynamism or competition.

2.5.1 Necessity for a Reference Model

A **Reference Model** is a not a formal definition, but rather a descriptive framework for a class of problems. It provides a structure for describing the components of a problem but is not exhaustive nor exclusive. Since we wish to describe a broad class of competition routing problems, without defining exactly what is and what is not included, a reference model is an important,

if not essential, contribution. It also provides a framework for standardisation of the important aspects of a problem class.

The classification scheme of Desrochers, Lenstra and Savelsbergh [54] can act as a **Reference Model for Vehicle Routing and Scheduling Problems** (RM-VRSP) with the components as in Figure 2.1. The RM-VRSP does not include DVRSP or TSSP but these could easily be incorporated since there are no additional components.

We need a structure for a new class of CRPs so that we can separate out what is new, what needs expanding further and what is inherited from RM-VRSP. A given VRSP can have a number of corresponding natural CRP variations. In this way it may be possible, meaningful or useful to study CRPs analogous to many classical routing problems and also their stochastic and dynamic variants.

The union of Sections 2.5.2–2.5.5 constitutes our proposed **Reference Model for Competition Routing Problems** (RM-CRP). In Section 2.5.6 we provide a number of simple, illustrative examples.

2.5.2 Extended Entity-Relationship Model for CRP

The VRSP classification of Desrochers, Lenstra and Savelsbergh [54], summarised in Section 2.2, provides a succinct descriptive framework for VRSPs. In this section, we extend this classification by building in a natural decision maker entity.

Competition routing problems involve several decision makers. We argue that *decision maker* is an entity type distinct from *addresses* and *vehicles*. A DM makes decisions on the service side of the equation and hence really has nothing to do with the customers or depots. Suppose there is a common fleet of vehicles from which the DMs can rent. In general a DM is distinct from the vehicles rather than an attribute.

Why are decision makers distinct from objectives? Essentially the private objective of a DM is an attribute of that DM. But, what about common objectives between DMs, such as that found in coalitions (temporary teams)? Would these fit as an attribute of the DM class or as a relationship between DMs? When the objective of a DM has a private component and a common component, how do we specify for that DM what the balance should be? We could simply replace the ⟨**objective**⟩ with ⟨**decision makers**⟩ but, essentially for the purpose of clear exposition, we would like to have both. Attributes, and limitations, of the rationality of decision makers as an entity can be grouped with decision makers. However, we may wish to be able to specify which objectives are considered in the problem class without identifying a particular objective with a particular player. In summary, although an objective is basically a subfield of decision maker, for backwards compatibility and clarity, we retain objective as a field. Hence we introduce a new field into the classification: the ⟨**decision makers**⟩ field.

Figure 2.2 shows the entity-relationship model for a CRP with the new **decision maker** entity type. In addition there are four new relationship types:

DM–A Decision-maker–address

DM–V Decision-maker–vehicle

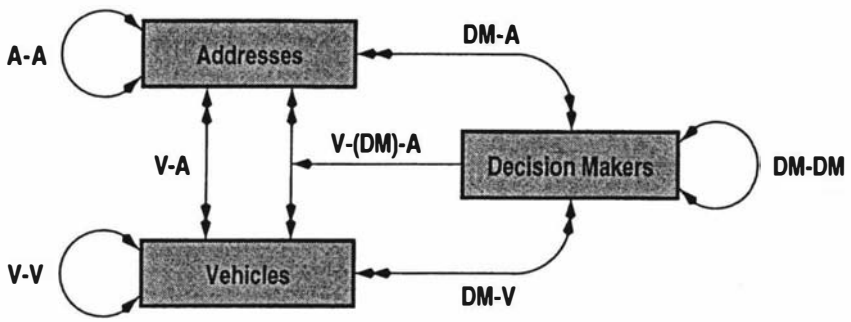


Figure 2.2: Entity-Relationship Model for CRP

$$\begin{aligned}
\langle\textbf{RM-CRP}\rangle \quad ::= \quad & \langle\textbf{addresses}\rangle / \\
& \langle\textbf{vehicles}\rangle / \\
& \langle\textbf{decision makers}\rangle / \\
& \langle\textbf{relationships}\rangle / \\
& \langle\textbf{objectives}\rangle \qquad (2.7)
\end{aligned}$$

$$\begin{aligned}
\langle\textbf{relationships}\rangle \quad ::= \quad & \langle\textbf{A-A restrictions}\rangle \\
& \langle\textbf{V-A restrictions}\rangle \\
& \langle\textbf{V-(DM)-A restrictions}\rangle \\
& \langle\textbf{V-V restrictions}\rangle \\
& \langle\textbf{DM-A restrictions}\rangle \\
& \langle\textbf{DM-V restrictions}\rangle \\
& \langle\textbf{DM-DM restrictions}\rangle \qquad (2.8)
\end{aligned}$$

DM–DM Decision-maker–decision-maker

V–(DM)–A Vehicle–address modified by decision maker, in which we may specify the relationship between a subset of vehicles and a subset of addresses for a particular subset of decision makers.

We replace Sentences (2.2) and (2.5) by Sentences (2.7) and (2.8), in which we have renamed ⟨**problem characteristics**⟩ with ⟨**relationships**⟩. Note that there are now five fields to the classification instead of four. Each of these five fields is called a **component** of the RM-CRP.

To put some flesh on the bones of this model, we must consider the fundamental characteristics of each decision maker

(i). Interaction with other decision makers.

(ii). Individual objectives.

(iii). Interactions with addresses and vehicles (or teams).

Each of these is explored further in the following Sections 2.5.3–2.5.5.

2.5.3 Decision Maker Component

We can now expand the components corresponding to the decision makers.

2.5.3.1 Decision Maker Attributes

Number. How many decision makers are there? Any decision makers who are completely dependent actually constitute a single decision maker.

Computational Resource. We must distinguish between on-line (real time or near to real time) computation and off-line decision making, subject to some kind of computational budget. This is where each player is allocated a computational budget, e.g., an amount of CPU time proportional to the duration of the game.

Observational Ability. Perfect observation may be unrealistic in the real world since the terrain will be either partially unsighted, like an inner city street network, or much too expensive to obtain, e.g., requiring satellite imagery.

2.5.3.2 Decision Maker–Decision Maker Relationships

Communication. Is it possible communicate between decision makers? Is this communication secure between parties or may others eavesdrop on conversations? What knowledge held by decision makers is private or public knowledge?

Coalitions. We assume that decision makers are independent. However, with or without communication, players may enter into permanent or temporary coalitions. This involves non-cooperative game theory considerations. What negotiations are possible and what side payments may be exchanged? How will contracts be enforced when there may be opportunity for a double-cross? What restrictions are placed on coalitions, e.g., permanence, binding rules, duration, some players not being permitted to cooperate, some possibly having to cooperate for the first part of the competition?

Piracy and Pursuit. Is it possible to steal from opponents if associated vehicle locations coincide? Is there any element of incentive to pursue or evade an opponent?

Simultaneous or Sequential Decisions. Is the underlying game a simultaneous or sequential move game, i.e., do all decision makers determine their actions concurrently or one after another?

Supply. There are problems in which there is some degree of autonomy and some degree of one (or more) DM having a great effect upon the outcome. For example, consider a problem in which there may be many salesman competing to deliver to a set of customers but in which there is also some sort of roving depot supplier who is also an independent decision maker with his own objectives to sell wholesale to the salesmen.

2.5.3.3 Decision Maker–Vehicle Relationships

Teams. A **team** is a fixed association of a number of vehicles to a single decision maker. We distinguish between a *team* and a *coalition* where a commitment to a team is binding and permanent (corresponds to a single DM), but a coalition is transient and not binding (corresponds to a DM for each member). We assume that there is at least one vehicle associated with a DM and that an objective is realised by some kind of vehicle routing activity.

Mercenaries. It may be possible to hire vehicles from an outside pool of vehicles, although if there is any competition for a scarce number of hire vehicles, then the rental agency would need to be modelled as a decision maker.

Communications. Can vehicles communicate with their associated decision maker, e.g., a central dispatcher? Does this occur frequently, infrequently or on demand? Are communications secure?

2.5.3.4 Decision Maker–Address Relationships and Decision Maker modified Vehicle–Address Relationships

Unit Task Completion. What are the rules governing servicing a customer, delivery to or collection from a customer or general execution of some task? This is essential since the task is the basic unit of competition and we must have a well-defined understanding of when any rewards or penalties accrue.

2.5.4 Objectives Component

The next component to consider is that of objectives.

2.5.4.1 Public and Private Objectives

A **public objective** is common knowledge between all players. A **private objective** is known only by the player who holds that objective. No assumption is made about the private objective of the opponent. However if an opponent's objective is public then we assume the opponent is rational with respect to that public objective. Note also that a public objective may include some kind of overall constraint such as the time limit since these may be hard or soft. A player's objectives may be **conflicting, partially conflicting** or **cooperative**. Of these, the first two may evolve direct competition, incidental competition or cooperation for mutual benefit.

2.5.4.2 Types of Objective

There are three fundamental components to a player's private objective:

1. value collected (by each player)
2. constraints on the overall time deadline for collection of prizes
3. how many games: one-off or one-of-many

Several **pure** private objectives are possible, assuming that the objective involves prize values:

Win. Claim more in sum of prize values claimed than any opponent. This is compatible with the traditional notion of a game with a winner and the concept of trying to dominate the opponent.

Max. Maximize the sum of prize values claimed, i.e., gaining as much value in prizes as possible but not caring about how much the opponents may claim. This is compatible with the economic interpretation in a market place of trying to improve market share or increase your own income.

Restrict. Minimise the sum of prize values claimed by the opponents.

Goal. Achieve some goal value in sum of prize values claimed in minimal time. However, what happens once that goal is actually achieved, or even before that, when it is guaranteed that is can be achieved by the player providing a verifiable path? Do we continue on until the other player does or cannot achieve his goal value, and in case we do, then what strategy does the first player play? What bonuses may be available for exceeding a goal or from preventing the opponent from achieving his goal?

Spread. Maximize the magnitude of a win or minimise the magnitude of a loss by maximizing the sum of prizes claimed less the sum of prize claimed by the most successful opponent.

Harvest. Maximize the harvest rate, i.e., the prize value claimed per unit time interval of harvesting, over a minimum/maximum time period.

Iterated. Maximize the best, worst or average sum of prize values claimed over a number of iterations of the problem instance.

Although these are the basic pure objectives, many combinations could be constructed to suit the application. In particular is the idea of achieving certain private or public goals throughout the duration of the game that may involve, e.g., cooperation, harvesting and winning at different stages, similar to a cycling points race.

Other useful objectives include minimising the *number of vehicles* required to satisfy all the constraints and *time dependent* objectives, such as the **Travelling Repairman Problem** (TRP) in which we are concerned with the time of visitation to each and every customer, that is their service time (Lucena [149]). This may profitably be combined with a prize value as in the MCPTDR of Brideau and Cavalier [28] (see Section 2.1.5).

Note. An important property of objectives for a CRP is whether the objective is memoryless. For example, the MAX objective remains as trying to claim as much in prize value as possible during the remaining time available. However, the WIN objective varies in consequence depending on how well a player does with respect to its opponent.

2.5.4.3 Constraints and Penalties

Suppose we have a time duration for a game that cannot be exceeded. We associate with each prize $i \in V$ a penalty $\pi_i \geq 0$. If prize i is unclaimed at the end of the game then **both** players have

$$\langle \textbf{objective} \rangle ::= \circ \vee \langle \textbf{operator} \rangle \langle \textbf{function} \rangle \quad (2.9)$$

$$\langle \textbf{function} \rangle ::= T_i \vee C_i \vee P_i(\langle \textbf{vehicle constraints} \rangle) \vee C_j \vee P_j(\langle \textbf{address constraints} \rangle) \quad (2.10)$$

where

T_i	[route duration]
C_i	[vehicle costs]
$P_i(\langle \textbf{vehicle constraints} \rangle)$	[vehicle penalty]
C_j	[address costs]
$P_j(\langle \textbf{address constraints} \rangle)$	[address penalty]
V_j	[path prize value]

π_i deducted from their score. Hence objective maximizing players may need to truly cooperate in ensuring that they can claim the badly penalising prizes between them so that they can both be better off.

Note that in a TSSP, there rewards contribute to the subtour value but penalties contribute to the subtour costs. For the CRP objective component, Sentences (2.9)–(2.10) summarise the major reward, cost and penalty contributors.

2.5.5 Address and Vehicle Components

The last components to consider are those corresponding to addresses and vehicles. The major of these attributes and relationships are inherited from RM-VRSP. Hence we consider those additions arising from the presence of competition.

2.5.5.1 Address Attributes

What are the *attributes* of the ⟨**address**⟩ component?

Tasks. To keep some of combinatorial structure to the CRP, we adopt the **task** as the basic unit of competition and assume that at least some tasks are contestable. A task may be a service, pickup, delivery or pickup-and-delivery. Mosheiov [163] considers the **Travelling Salesman Problem with Pick-up and Delivery** (TSPD) in which *delivery customers* are served by delivery of goods from a central warehouse and *pick-up customers* need to deliver goods from their locations to the warehouse. General pickup and delivery problems (e.g. dial-a-ride) involve a unit task of picking up a load from some point and completing a delivery of that load. However, we allow that not every task need be undertaken, but rather permit the tactical selection of a subset of the tasks.

Prizes. We also assume that there must be some incentive for servicing a particular task, such as a reward, or some disincentive for not servicing a particular task, such as a penalty. Prize values may be time dependent and prizes may require some nonzero time to service.

Environment. The environment is the structure in which addresses are embedded, e.g., the Euclidean plane, a graph or network, street map, Geographic Information Systems (GIS) coverage, Digital Terrain Model (DTM). The environment may model obstacles or a physical terrain and may either be completely known in advance or have some element of exploration required, hence requiring a balance between accumulating prizes, exploring terrain and observation of opponent or prizes, i.e., between gaming tactics and exploration tactics. Also the environment may have dynamic elements such as wind speed and direction.

Selection and Covering. Not all prizes or tasks need to be selected, as in the TSSP. In addition we may consider allocation of off-tour prizes to on-tour prizes as in the SVRAP of Beasley and Nascimento [15] (see Section 2.1).

Calling Process. Tasks or prizes may become available for selection, i.e., arrival, according to some stochastic arrival process of calls from customers, as in the DTSP and DVRSP.

2.5.5.2 Address–Address Constraints

What are the *relationships* between ⟨**addresses**⟩? Location-routing problems, like those described by Laporte [133], can be modelled as relationships between tiers of addresses such that a load must be delivered to a depot address before it can be picked up from the depot address and delivered to the customer address.

2.5.5.3 Vehicle Attributes

What are the *attributes* of the ⟨**vehicle**⟩ component?

Movement. Relative speed, possibly a function of the location on the terrain. Energy constrained by vehicle load and surface traversal, i.e., a tiredness budget.

Capacitated. If prizes are considered to have some weight or volume then capacity of vehicles becomes an important consideration. Also, vehicles may be able to drop off prizes collected either at some *depot* or just drop them somewhere in the environment, henceforth able to be claimed by another vehicle, either from the same team, coalition or an opponent. If a vehicle must return to the depot for some reason then we might have depots; for example, if the vehicle has a capacity and the prizes have some mass, then the vehicle may first collect prizes and then return to the depot to deposit them, and then continue prize collection. We may wish to exchange prizes at a location if we can get a better one in exchange or miss out on it, especially if a vehicle can only go out once (the knapsack version).

Redistribution. On-route redistribution of payload. Roving depots. On-route redistribution of payload may be permitted, especially between vehicles in a team, or between members of a coalition. There is no "currency" so that negotiation of giving away or exchanging a prize is between players.

2.5.5.4 Vehicle–Vehicle Constraints

What are the *relationships* between ⟨**vehicles**⟩? These would be inherited from the RM-VRSP.

2.5.5.5 Vehicle–Address Constraints

What are the *relationships* between ⟨vehicles⟩ and ⟨addresses⟩? What was previously ⟨**type of strategy**⟩ in the classification scheme of Desrochers, Lenstra and Savelsbergh [54] is now incorporated in ⟨**V-A restrictions**⟩. Competition related constraints would be included in ⟨**V-(DM)-A restriction**⟩ of Section 2.5.3 since they must necessarily involve a decision maker. Non-competition related constraints would be inherited from the RM-VRSP.

2.5.6 Illustrative Examples

Having described the components of the RM-CRP, we can now develop a few complete illustrative examples. Some familiar problems can be described succinctly by the new classification scheme: TSP is $1/1/1//T$, OP is $1/1/1, dur//\text{MAX}$, and TOP is $1/1/m, dur//\text{MAX}$.

2.5.6.1 Household Milk Delivery

Up until a few years ago, the delivery of milk in glass bottles to homes was common in New Zealand. However, many people now purchase their milk from the supermarket in plastic containers, although some people still support the milk vendor. Customers would place the exact number of required milk bottles at the gate of their home, together with prepaid plastic tokens. Milk vendors usually employed teenagers, pushing milk trolleys loaded with full bottles, to service the customers. Historically, milk vendors have cooperated with each other in defining districts within which each vendor will operate. Presumably this is because of the low economic margins within which these businesses operate and because demand is low compared with potential supply.

If however, demand was high in comparison with supply, a CRP would be an appropriate formulation of the problem since customers are indifferent between suppliers. Vehicles correspond to the milk trolley and milk truck. Decision makers are the vendors. Vehicles are capacitated but it is also possible to transfer (or replenish) bottles between vehicles.

2.5.6.2 Bicycle Couriers

Bicycle couriers are common in the *central business district* (CBD) of many large cities. These couriers are directed to pickup and deliver small packages among offices by the dispatcher in the courier company. Many small courier businesses operate within the CBD. At present, customers requiring transit of packages contact the courier dispatcher of a particular business who arranges for a bicycle courier to pickup the package. Tight requirements of delivery service are often required by customers.

2.5.6.3 Terrain Navigation

In the presence of an unknown terrain or obstacles even a single prize TSSP may be non-trivial. Taylor [197] considered a variation of the OP that employed a digital elevation model of a physical terrain. Simulations involved both exploring the terrain to discover the prize locations and efficiently sequencing those prizes whose locations were observed from being in the viewshed of some previous location.

A generalization of this problem to a CRP involves a terrain and a number of competitors, much like a true Orienteering event, except that prizes (control points) may only be claimed by one orienteer. An autonomous vehicle must construct a map of the terrain on which it is navigating; it should try to exploit as much knowledge of the terrain as possible in order to make the best decisions about its path (Mitchell [160]) but must also balance considerations of exploration and prize targeting.

2.5.6.4 Airport Shuttles

Boyd, Clarke, Gemmell and Miller [26]model an airport shuttle bus problem as a CRP. Two competing airport shuttle companies each have four vehicles to pickup passengers from the airport and delivery then to their homes or pickup from homes and deliver to the airport. Whenever a new customer arrives in the system, via a stochastic arrival process, both companies determine the best way they can serve the customer, based on their current vehicle routes, and submit a bid price for servicing that customer. The customer chooses a company by considering the bid prices and the reputation of each company; reputation is based on service satisfaction and prize satisfaction. Having determined which company to choose, the customer is assigned to that company and is served accordingly. The goal for each company is to make a profit. Each company's behaviour is dictated by an operating policy, which defines how vehicle routes are selected, the *vehicle routing strategy*, and how bid prices are determined, the *pricing strategy*. A central idea is that of *reputation* as a common measure of each company's perceived level of service. Each carrier is committed to a set of customers and also knows the commitment of each of the other carriers to customers. Also, each customer is assigned to a carrier when the call is placed; no trading or poaching of customers is permitted. The nature of this problem indicates that information about routing and forecasting customers are the two key components. Note that in this model it is sensitive to much information we have about the other carrier's vehicle's locations since then positioning ourselves relative to our opponents becomes very important.

2.5.6.5 Treasure and Pirates

A pirate has buried his lifetime's collection of treasure over a number of rugged terrain islands in the South Pacific. At his death, his crew, each looking after their own interests, descend upon the island. Each of them, having served the pirate over his treasure-burying career, had detailed records of the dig spots. Suppose each pirate had a GPS navigation system and could track the whereabouts of his rival former crew mates. Further, suppose each receives coded communication from one of his rivals offering him a share of a pooled collection if he agrees to concentrate his hunting upon a particular group of islands. Of course, each rival's ship can only hold so much cargo but may return time and time again until no more treasure remains. Unfortunately, another band of pirates have gotten wind of this treasure feast and are en route to the isles from across the seas, estimated time of arrival: one week from today.

2.5.6.6 Discussion

We have suggested five, more or less reasonable, applications which can be formulated as CRPs and in the process shown the richness of the competition model in vehicle routing and scheduling type problems. Even within the range of these simple formulations there is a variable degree of balance between competition aspects and routing aspects of the problem.

- In the airport shuttle problem, there is a strong routing focus and we may be able to ascertain the flexibility of the opponent being able to accommodate additional customers and, especially, the marginal cost to the opponent of doing so. Hence tendering could well be a case of good information generation from our forecasting and routing components. The actual tendering is more a cost exercise than a game exercise since there is very little hard data on future bidding.

- However, in the milk vendor problem, there is a large amount of gaming and the routing is secondary since the set of customers to be serviced is highly dynamic. Also, there is no stochasticity in this case, only dynamism.

2.6 Core Problem: Competitive Prize Collection Problem

We now turn our attention to formulating a simple CRP which, although it is not a model of a practical, real-world problem, it is representative of the core attributes of a CRP.

2.6.1 Purpose

The TSP is the archetypal VRSP; usefully understanding any VRSP requires a good understanding of the sequencing problem inherent to the TSP. Moreover, the TSP invariably appears as a subproblem as part of a solution method for a VRSP.

The RM-CRP considers the components that describe a CRP. The class of CRPs is at least as large as the class of VRSPs and we currently have no analysis of, nor computational experience with, a CRP. We wish to determine a core representative CRP that forms the nucleus of the CRP in a similar role to that which the TSP serves for the VRSP. In this way we wish to be able to extrapolate useful conclusions about the class of CRPs from analysis of, and computational experience with, the core problem. Hence we apply the **Principle of Occam's Razor.**[4]

2.6.2 Fundamental Components

Having determined to design a competition variation of the TSP, or rather the TSSP since one player need not necessarily visit all locations, what are the absolutely essential problem attributes and relationships?

[4]The principle attributed to the English philosopher William of Occam that the fewest possible assumptions are to be made in explaining something.

Addresses. Since we have a competition, we must have some customers or prizes over which to compete, each with some commensurate reward or penalty value. The TSSP is a combinatorial optimization problem and hence there is little difference between a graph or network structure and a Euclidean problem since both are converted to a inter-customer distance matrix. However, in a dynamic problem there is a considerable difference and the Euclidean problem is much less assumptive although the graph embedding is more constrained. Either possibility could be considered. We choose the Euclidean problem due to its similarity with pursuit-evasion games and the diminishing of the routing aspect. The graph embedding is considered further in Section 11.2.2.2.

Vehicles. One uncapacitated vehicle per decision maker.

Decision Makers. It turns out that considering two players is sufficiently complicated without having to analyse the dynamic collusion possibilities of three or more players. Indeed, any attempt at analysing three or more players would require some understanding of the corresponding two player analysis. Also, we choose to start with a noncooperative game, i.e., one in which any type of collusion, such as correlated strategies and side payments, is forbidden. A public objective implies that the players either must cooperate or can make assumptions regarding the opponent's behaviour. However, any analysis is then dependent upon the exact nature of the public objective. Hence we discard public objectives and each player need not make any assumption about the private objective of the opponent.

Objective. The simplest objective is to claim as many prizes as possible. This has the important property that it is memoryless—whatever the current state of play, the objective is the same.

Relationships. Nil, i.e., no restrictions, all prizes are contestable.

2.6.3 Definition

We are now in a position to precisely define a simple CRP for further computational study.

Definition 2.6.1 (Competitive Prize Collection Problem (CPCP))

Let $V = \{1, \ldots, n\}$ be a set of "contestable" **prizes**. Associated with each prize $i \in V$ is a **location** (x_i, y_i) in the Euclidean plane and a **value** $v_i > 0$. For the **Competitive Prize Collection Problem** (CPCP), two independent **players** ($\mathcal{A}$ and $\mathcal{B}$), with given initial locations (x_A, y_A) and (x_B, y_B) respectively, move continuously at the same constant speed in the Euclidean plane. Each player's private objective is to individually collect as much in total prize value as possible up until the overall deadline, λ, expires. The value of each prize is only awarded to the first player who visits; a prize location which is visited simultaneously by both players has its associated value shared equally. At all times, each player has perfect observation of the state of the game position, i.e., where the players are currently located and which prizes remain unclaimed.

□

By considering the simplicity of each component of a CRP, we can see that the CPCP is indeed a core CRP. Evidently, the classification of the CPCP is $2//2/\Re^2, own/V$.

Axelrod [7] makes a number of important assumptions regarding the players in the Prisoner's Dilemma (see Section 2.4.5) which we also apply to the CPCP.

- "There is no mechanism available to the players to make enforceable threats or commitments. Since the players cannot commit themselves to a particular strategy, each must take into account all possible strategies that might be used by the other player. Moreover the players have all possible strategies available to themselves."
- "There is no way to be sure what the other player will do on a given move. This eliminates the possibility of reliable reputations such as might be based on watching the other player interact with third parties. Thus the only information available to the players about each other is the history of their interaction so far."
- "There is no way to eliminate the other player or run away from the interaction."
- "There is no way for the players to change the payoffs."
- "The players can communicate with each other only through the sequence of their own behaviour."
- "There is no need to assume that the players are rational. They need not try to maximize their rewards. Their strategies may simply reflect standard operating procedures, rules of thumb, instincts, habits, or imitation."

When $\lambda = \infty$ the CPCP is a constant sum game (see Section 2.4) since there is always time to claim one more prize. However, when $\lambda < \infty$ the CPCP is a noncooperative general-sum game since it is possible that the overall deadline expires before all prizes are claimed. In the latter case, the players may exhibit qualities of cooperation since both players could be better off by not entering into time consuming conflict. Brandenburger and Nalebuff [27] describe the paradigm of coopetition ("COOPeration" plus "compETITION") in which cooperation between players can sometimes work out to both players' advantage.

2.6.3.1 Competing Salesmen Problem

Fekete and Schmitt [66] study the **Competing Salesmen Problem** (CSP), also a two-player competitive version of the TSP. The players take turns, moving one edge at a time within a graph, with customers located at the vertices of the graph. Both players always know the position of their opponent and the winner is the player who reaches a majority of the customers. A geometric variant of the CSP is also proposed, in which the customers are located in the Euclidean plane and players move with the same maximal speed. This geometric CSP appears to be equivalent to the CPCP with equal prize values but with the objective of gaining more prizes than the opponent and alternative moves.

Fekete and Schmitt focus on establishing whether a player can avoid a loss or force a draw in the graph-based CSP. They draw on the theory of similar combinatorical games, mainly considering the special case where the graph is a tree. This thesis, however, focuses on the design and

comparison of effective strategies rather than on the existence of optimal strategies.

2.7 Research Questions

We have formulated the CRP by building upon the DVRSP, VRSP and TSSP, with the CPCP as a special case of the CRP. This successfully addresses the first research challenge of Section 1.3, namely to formulate a reference model for CRPs. The remaining research questions address the DVRSP, VRSP, TSSP, CRP, CPCP and the relationships between them. The DVRSP, VRSP and TSSP are well established and accepted in the literature. Although the RM-CRP provides a modelling framework for CRPs, the ultimate research challenge is to determine an effective and efficient solution paradigm for the family of CRPs. Since the CPCP is the only problem formulation which we address computationally, each research question necessarily must attempt to extract some insight in this direction. The second research challenge is to identify and analyse the fundamental principles which computational dynamic routing *strategies* must address and to investigate the relationship between the routing and strategic components.

2.7.1 Relationship of CRP/CPCP to DVRSP/VRSP/TSSP

What is different between vehicle routing and competitive routing and how does this assist in the approach to the design of solution methods? Is the DVRSP/VRSP/TSSP solution paradigm applicable to and useful for the CRP/CPCP and, if not, what are the missing principles or techniques?

2.7.2 Understanding the CPCP

What constitutes understanding of the CPCP?

Success. How do you plan computationally in an environment involving selection and sequencing of customers whilst in competition with an opponent? What does strategy need to address to be successful under a scarce computational resource? What are the important new concepts which need to be explored in competition strategies?

Classification. Which solution methods are best applicable to a given subclass of problems and a given computational resource? Do small and large problems require a different strategic approach? Can we distinguish between an average case problem, requiring good performance to be successful, and a bad case problem, requiring good understanding to be successful?

Architecture. What is an appropriate solution architecture for the CPCP? How can the degree of natural structure and decomposability be exploited? What balance is appropriate between strategic and tactical determination and carrying through operations or realising goals?

2.7.3 Relationship of CPCP to CRP

We have formulated a very simple abstract problem, the CPCP, from the class of CRPs. This is because we believe it to be important to study this problem first in the same way as the TSP is fundamental to the class of VRSPs. Is a solution paradigm or architecture designed for the CPCP able to be usefully extrapolated to a general CRP and, if so, is it likely to be effective? Also, is the CPCP a useful subproblem of a general CRP? Finally, is the CPCP a useful exercise for initial understanding of the complexity of modelling competition in vehicle routing research?

Coda

▼ Summary

In this chapter we have surveyed literature on subset selection routing, vehicle routing and scheduling, dynamic vehicle routing, real-time problems and game theory. This was necessary to synthesise a Reference Model for Competition Routing Problems (RM-CRP) and propose the core Competitive Prize Collection Problem (CPCP).

▼ Link

Having formulated the problem, we now need to be able to propose a solution paradigm. In the following chapter we start by applying the *heuristic* paradigm established for the VRSP and DVRP to determine if it is suitable for the CPCP and in the process determine some key difference of CRPs from VRSPs.

CHAPTER 3

Elements of Strategy

Nothing is more fairly distributed than common sense: no one thinks he needs more of it than he already has.

— DESCARTES

Heuristics! Patient rules of thumb,
So often scorned: Sloppy! Dumb!
Yet, slowly, common sense become.

— ODE TO AI

We now turn our attention to evaluating whether the TSSP/DVRSP solution paradigm is suitable for the CPCP.

3.0 Introduction

The aim of this chapter is to determine which principles a "typical solution strategy" needs to address in order to be successful, contrasting those principles make the CPCP different from the TSP. In beginning the study of a new problem, one common approach is to modify basic techniques and paradigms from existing problems, hence we are initially concerned with evaluating how well existing solution methods for the TSSP/DVRSP cross over as potential strategies for the CPCP.

To this end Section 3.1 provides an introduction to the necessary solution concepts. Section 3.2 applies and extends basic heuristic solution methods for the TSSP to the CPCP. These heuristic strategies are based upon the ranking and selection of prizes. Section 3.3 formulates combinatorial

optimization subproblems, similar to the TSSP, as player decision problems whose (heuristic) solution give a strategy for the CPCP. In conclusion, Section 3.4 extracts some observations to show that by restricting solution strategies to combinatorial optimization subproblems, a number of necessary principles of strategy are neglected. Finally, Appendix 3.A is a basic tutorial on local search methods for combinatorial routing subproblem applied to the subproblems designed in Section 3.3.

3.1 Algorithms, Heuristics, Strategies and Simulations

Before embarking on the design of solution strategies for the CPCP, it is useful to review some terminology and solution concepts from combinatorial optimization, game theory and simulation.

3.1.1 Algorithms and Heuristics

The TSP, TSSP, VRSP and similar problems are all combinatorial optimization problems. Solution methods for the problems are either guaranteed to find the best feasible solution or they are not.

An **algorithm** is a "scientific procedure which will converge to the best feasible solution to the problem" (Foulds [70]) which executes in a finite amount of time. Unfortunately, many complex problems require the evaluation of an large number of potential solutions to determine an optimal solution.

A **heuristic** is "a technique which seeks a good (i.e. near-optimal) solution to a problem at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is" (Reeves [185]). However, Pearl [174] warns

> It is often said that heuristic methods are unpredictable; they work wonders *most* of the time, but may fail miserably *some* of the time. Indeed, some heuristics greatly reduce search effort but occasionally fail to find an optimal or even a near-optimal solution. Others work efficiently on most problems in a given domain yet a rare combination of conditions and data may cause the search to continue forever.

Computational complexity is concerned with the computational effort required to solve a given problem as some function of the problem size (Garey and Johnson [73]). It is the benchmark by which problem difficulty and solution efficiency are measured. A method which has polynomial time complexity is deemed **efficient**. We are only concerned with ensuring that any routing component of a strategy for the CPCP employs an efficient heuristic.

3.1.2 Strategies

The CPCP, however, is not a combinatorial optimization problem. A solution to the DTSP or DVRSP is a **policy** (Savelsbergh and Sol [190], Psaraftis [179, 180]) but a solution to the CPCP is a **strategy**.

Osborne and Rubinstein [171] define a **pure strategy** of a player in an extensive game with perfect information with simultaneous moves as a function that assigns an action to to each nonterminal history. Hence a strategy specifies the action chosen by a player for *every* history after which it is his turn to move, even for histories that, if the strategy is followed, are never reached. We say that player **maximinimizes** if she chooses an action that is best for her on the assumption that whatever she does, the opponent will choose his action to restrict her as much as possible.

A **mixed strategy** is a probability distribution over pure strategies. A **Nash equilibrium** is a pair of mixed strategies, one for each player, such that each player's mixed strategy is an optimal response to the other player's mixed strategy. A **pure strategy Nash equilibrium** is a pair of pure strategies, one from each player, which satisfies the same conditions. Every finite strategic form game has a Nash equilibrium. We define an **optimal** strategy as equivalent to a pure strategy Nash equilibrium.

Assumption 3.1.1

Assume that if it is known that a strategy is optimal for a class of problem instances, and if we determine computationally that a particular problem instance is a member of that problem class, then the player will select the optimal strategy. □

There is the difficulty (e.g. in the case where the prize locations and player locations are all collinear) where there is the possibility of an infinite standoff, that is where there is a cycle of positions and neither player want to "give in" to the other. This can only occur when $\lambda = \infty$.

Assumption 3.1.2

Assume that $\lambda < \infty$. □

3.1.3 Simulation Environment

The infinite subtlety resulting from players moving in true continuous time is too difficult to model computationally. Hence we impose the condition that players observe the state of the system and make action-decisions only at discrete time steps. A player is committed to that decision for the duration of that time step.

A computationally efficient way to evaluate a solution procedure for a dynamic problem is via a discrete time **simulation environment.** A strategy implicitly defines what action will be taken for any given situation. For the purposes of simulation, time is discrete and proceeds in steps of duration Δ. Each player may move at most a distance Δ from its current location in one step.

We have seen that a strategy for a player is a mechanism which determines for one player how to move. We now consider an operational definition for a strategy in terms of exactly what a computational method must provide for the simulation controller. A **move** for player $\mathcal{X}$ consists of either a *target-prize* or both a *target-bearing* and a *target-distance.*

- Suppose player $\mathcal{X}$ specifies a *target-bearing*, $\psi_{\mathcal{X}}$ and a *target-distance*, $0 \leq \delta_{\mathcal{X}} \leq \Delta$. Then player $\mathcal{X}$ moves directly to the location a distance $\delta_{\mathcal{X}}$ in the direction of $\psi_{\mathcal{X}}$ from player

$\mathcal{X}$'s current location. No prizes are claimed by player $\mathcal{X}$ en route even if a prize lies directly on the corresponding trajectory.

- Suppose player $\mathcal{X}$ specifies a *target-prize* i. If $d_{Xi} > \Delta$ then player $\mathcal{X}$ moves directly to the location a distance Δ towards prize i from player $\mathcal{X}$'s current location. No prizes are claimed by player $\mathcal{X}$ even if a prize lies directly on the corresponding trajectory. If $d_{Xi} \leq \Delta$ then player $\mathcal{X}$ moves directly to the location of prize i and then consults a secondary-move to determine to which location to move during the time $\Delta - d_{Xi}$ remaining. A **secondary-move** for player $\mathcal{X}$ consists of either a *secondary-target-prize* or both a *secondary-target-bearing* and a *secondary-target-bearing*.
  - Suppose player $\mathcal{X}$ specifies a *secondary-target-bearing*, ψ'_X and a *secondary-target-distance*, $0 \leq \delta'_X \leq \Delta - d_{Xi}$. Then player $\mathcal{X}$ moves directly to the location a distance δ'_X in the direction of ψ_X from the location of prize i.
  - Suppose player $\mathcal{X}$ specifies a *secondary-target-prize*, i'. Player $\mathcal{X}$ moves directly to the location a distance $\Delta - d_{Xi}$ towards prize i' from the location of prize i.

Assumption 3.1.3

We also make the assumption that

$$d_{i_1 i_2} \geq 2\Delta \quad \forall i_1, i_2 \in V$$

Hence a player can only claim a prize if it is specified as the primary *target-prize* and a player can claim at most one prize per step. The 2Δ is required in Assumption 3.1.3 in order to prove Lemma 5.5.2. □

Suppose both players target the same prize, i. If $d_{Ai} \neq d_{Bi}$ then one player will miss out on claiming prize i but the destination of each player is independent of which claims the prize. Otherwise the players prize i and both move away from prize i according to their respective secondary-moves.

In a **real time simulation environment**, players simultaneously compute and enact their movements. Thus there is a tradeoff between computation time required for planning (delay in returning a policy) and actually playing in the short term.

In a **discrete time simulation**, players separate computation from motion. A budget of computational resource is allocated to each player, either in bulk funding form (for the whole duration of the game) or in funding per time step. The general form would be something like Algorithm 3.1.[1]

We provide three examples in order to contrast the differences between optimal strategies and heuristic strategies. Example 3.1.4 gives an optimal strategy for the problem instance in which for each prize the players are equidistant to that prize. Example 3.1.5 gives a heuristic strategy for the one dimensional problem instance. We use the symbol '$\mathcal{A}$—$\circ i$' to indicate that player $\mathcal{A}$ heads in the direction of prize i, and the symbol '$\mathcal{A}$—$\circ\mathcal{B}$' to indicate that player $\mathcal{A}$ heads in the direction of player $\mathcal{B}$'s current location.

[1] Algorithms are presented in a **Pidgin Pascal** style similar to Papadimitriou and Steiglitz [173] but with C++ style comments beginning with '//'. The keyword **target** is used to assign the player's choice of target structure.

Algorithm 3.1 model SIMULATION ENVIRONMENT

// A simulation environment for the CPCP.
$t \leftarrow 0$ *// Time.*
$Q \leftarrow V$ *// All prizes initially available.*
Initialise player prize value totals.
while ($t < \lambda$ **and** $Q \neq \emptyset$) **do**
Players $\mathcal{A}$ and $\mathcal{B}$ select a move simultaneously.
A player within Δ of target prize may optionally select a second move.
Update player locations, remaining prizes Q and player totals.
Increment t.
end

end

Algorithm 3.2 strategy EQUIDISTANT

// When each prize is equidistant to the players.
target $\leftarrow$ median prize

end

Example 3.1.4 (Equidistant Prize Locations)

Consider the a problem in which each prize is equidistant from both players but the players are collinear with the prize locations. Then the players are either identically located or they are mirror images where the mirror line is the line through the prizes; it both cases are equivalent. There exists at least one (and at most two) **median prize**, one for which $\leq$ half the value sum of prizes lies strictly on one side and $\leq$ half the value sume of prizes lies strictly on the other side of the prize. Then the optimal strategy is to target a median prize since the opponent must either also target the median prize of move to one side of the other which is non-optimal for them. Algorithm 3.2 EQUIDISTANT gives the corresponding CPCP strategy.

□

Example 3.1.5 (Collinear Prize and Player Locations)

The **collinear problem** is the special case in which all the prize locations and initial player locations are collinear. This section gives an optimal strategy for the collinear problem. The basic idea is to head towards your opponent unless $d_{AB} \leq 2\Delta$ or we require some tie breaking rule in case of an infinite standoff. Chess has a similar breaking rule in that a state cannot be cycled more than three times in the game.

(i). Suppose the players are coincident. Determine the value sum of the prizes remaining on each side of the players and head towards the side of greater value or if the values are equal then choose either side arbitrarily.

(ii). Suppose $0 < d_{AB} \leq 2\Delta$ and there is no prize between the player locations. Then if the players step towards each other then at the next iteration they may be coincident or have swapped sides. We can evaluate whether this is beneficial by once again determining the value sum of the prizes remaining on each side of the players' locations. It would be beneficial to head towards the opponent if their side is more valuable and not beneficial if our side is more valuable.

(iii). Suppose $0 < d_{AB} \leq 2\Delta$ and there is at least one prize between the players. Since the distance between any two prizes must be at least 2Δ there must be exactly one prize, i, between the players. We must be able to evaluate whether it is beneficial to target that prize. Let $v(A)$ be the value sum of prizes strictly on player $\mathcal{A}$'s side of prize i and let $v(B)$ be the value sum of prizes strictly on player $\mathcal{B}$'s side of prize i. There are several cases assuming we are player $\mathcal{A}$:

(a) $d_{Ai} < d_{Bi}$: We can suppose that both players head towards prize i and hence that player $\mathcal{A}$ claims prize i. Then we will face the previous case, so $\mathcal{A}$—$\circ\mathcal{B}$ if $v(B) > v(A)$ otherwise player $\mathcal{A}$ will reverse direction.

(b) $d_{Ai} > d_{Bi}$: Again we can suppose that both players head towards prize i and hence that player $\mathcal{B}$ claims prize i and again we will face the previous case, so $\mathcal{A}$—$\circ\mathcal{B}$ if $v(B) > v(A)$ otherwise player $\mathcal{A}$ will either reverse direction or simply head away from prize i in the first place.

(c) $d_{Ai} = d_{Bi}$: Player $\mathcal{A}$—$\circ i$ if $v(A) - v(B) < v_i$ in which case $\frac{1}{2}(v(A) + v(B) + v_i > v(A)$ otherwise player $\mathcal{A}$ will move away from prize i.

(iv). Suppose $d_{AB} > 2\Delta$. Then player $\mathcal{A}$ should head towards player $\mathcal{B}$ unless all the prizes are on player $\mathcal{A}$'s side, in which case player $\mathcal{A}$ should head towards the prizes. We must identify under what circumstances the game fails to terminate playing against a general player. If the game is to fail to terminate then eventually we will get a situation in which $d_{AB} \leq 2\Delta$ and both players will withdraw then move together then withdraw then move together in an infinite standoff. We bring in the rule that this can only happen three times after which, if the players again withdraw then they are compelled never to head towards each other again.

□

Example 3.1.6 (Consistent Random Prize)

The simplest heuristic strategy (Algorithm 3.3 CONSISTENT RANDOM PRIZE) is to select a prize at random and consistently target that prize until it is claimed, by either player, and then select another prize at random and so on.

□

Algorithm 3.3 strategy CONSISTENT RANDOM PRIZE

Select a remaining prize at random and consistently target that prize until it is no longer remaining.

end

3.2 Prize Ranking and Selection Strategies

Suppose we wish to select a single target-prize by assigning to each prize a *desirability score* and ranking the prizes accordingly. Essentially, any strategy which selects a target-prize can be thought of in this generic way. Initially, however, we wish to determine a desirability score for a prize without considering any sequence of prizes.

The TSSP, VRSP and variations are inherently formulated as tour or subtour (tour of a subset) problems, in which the vehicle must return to its start location upon completion of the sequence of customers. In applying these heuristic solution concepts to the CPCP, we consider a graph theoretic[2] path structure, in which the vertices are the current player location, called the **root**, and the set of prizes. A **subpath** is a path which begins at the root and visits a subset of the prizes in sequence. A **nontrivial** subpath visits at least on prize. The initial prize on a nontrivial subpath is the **head** and the final prize is the **tail**. Finally, let $Q \subseteq V$ be the set of prizes remaining unclaimed at global time t.

For the purposes of this section, we suppose that $\lambda = \infty$. The extension to the finite λ case is straightforward but would over-complicate the presentation of the ideas with details for checking for deadline feasibility.

3.2.1 Desirability Criteria

We begin by considering criteria which capture the desirability of an individual prize, independent from considering sequences of prizes or proximity to other prizes.

3.2.1.1 Proximity to the Player Location

Nearest Neighbour. Rosenkrantz, Stearns and Lewis [187] originally proposed the *nearest neighbour* heuristic for the TSP. At each iteration, expand the current subpath to the vertex which is closest to its end, i.e., the "nearest neighbour". Algorithm 3.4 NEAREST NEIGHBOUR presents the corresponding CPCP strategy.

Farthest Neighbour. Algorithm 3.5 FARTHEST NEIGHBOUR presents the analogous CPCP strategy in which the farthest prize from the player is selected.

[2]Many TSSP/VRSP heuristics employ graph theoretic terminology for concise definitions. We refer the reader to Bondy and Murty [23] for basic graph theory concepts and definitions as necessary.

Algorithm 3.4 strategy NEAREST NEIGHBOUR

// Target the nearest prize.
target $\leftarrow \text{argmin}_{j \in Q} d_{Aj}$

end

Algorithm 3.5 strategy FARTHEST NEIGHBOUR

// Target the farthest prize.
target $\leftarrow \text{argmax}_{j \in Q} d_{Aj}$

end

3.2.1.2 Value of Prize

Certainly the value of a prize is important and was ignored by considering only proximity. Algorithm 3.6 MAX VALUE and Algorithm 3.7 MIN VALUE reflect two extremes.

However the proximity of a potential target-prize to other valuable prizes can also be an important consideration. Golden, Levy and Vohra [89] and Golden, Wang and Liu [91] develop a heuristic for the OP which inserts vertices into a subpath "based upon a neighbourhood score rather than an individual score." Let the **subgravity**, $s(i)$, of prize $i \in Q$ be given by

$$s(i) = \sum_{j \in Q} v_j e^{-\mu d_{ij}}$$

for some parameter $\mu > 0$. The subgravity value, in conjunction with a centre-of-gravity measure and an ellipse measure, is used to select a prize k not in the subpath to insert, This is an example of *stage setting*, i.e., considering what lies beyond the initial target-prize, but without considering sequences of prizes.

3.2.1.3 Greedy

Greedy heuristics address the tradeoff between potential value and time invested in travel.

Greedy. Laporte and Martello [138] (also Keller [126]) proposed a *greedy* heuristic for the OP: at each iteration, expand the current subpath to the prize which maximizes the ratio of prize value to distance from the end of the subpath such that the total length of the subpath

Algorithm 3.6 strategy MAX VALUE

// Target the maximum value prize.
target $\leftarrow \text{argmax}_{j \in Q} v_j$

end

Algorithm 3.7 strategy MIN VALUE

// Target the minimum value prize.
target $\leftarrow \text{argmin}_{j \in Q} v_j$

end

Algorithm 3.8 strategy GREEDY

// Target the greediest prize.
target $\leftarrow \text{argmax}_{j \in Q} \frac{v_j}{d_{Aj}}$

end

does not exceed the distance bound. Algorithm 3.8 GREEDY presents the corresponding CPCP strategy. Algorithm 3.9 SUBGRAVITY GREEDY uses the subgravity of a prize instead of its value.

S-**algorithm.** A variation was proposed by Tsiligirides [200] in another heuristic for the OP. Suppose prize k is the root or tail of the current subpath. Define a desirability factor

$$a_j = \left(\frac{v_j}{d_{jk}}\right)^4$$

for each prize j not on the current subpath. The four prizes with highest desirability values are normalised and a random number from $U[0,1]$ determines the prize to append to the current subpath.

Observation. If a player targets the nearest prize then this prize certainly remains the nearest to that prize. However, for other methods, the target-prize is not necessarily static. For example suppose prize i_1 lies between prize i_2 and the location of player $\mathcal{X}$ and suppose further that $\frac{v_{i_2}}{d_{\mathcal{X}i_2}} > \frac{v_{i_1}}{d_{\mathcal{X}i_1}}$. Hence if player $\mathcal{X}$ selects prize i_2 as its target-prize according to the greedy criterion, then there may come a point at which player $\mathcal{X}$ switches to prize i_1 as its target-prize by the same criterion. A simple modification is to consider the cost as the distance from the previous prize claimed by that player (or the initial location if no prize has yet been claimed by the player).

Algorithm 3.9 strategy SUBGRAVITY GREEDY

Input: μ *// Discount factor parameter.*

target $\leftarrow \text{argmax}_{i \in Q} \left\{ \frac{1}{d_{Ai}} \left(\sum_{j \in Q} v_j e^{-\mu d_{ij}} \right) \right\}$

end

3.2.2 Subpath Insertions

As an instance of the CPCP is played, each player collects prizes in a sequence, one at a time. We wish to investigate some ways of building enough of a subpath so that we can target the first prize on the subpath, but not necessarily committing to entire, but rather just stepping towards the head of the subpath.

3.2.2.1 Iterative TSSP Construction Heuristics

Several iterative construction heuristics for the TSSP work by iteratively ranking and selecting a vertex to add to a partial solution. Höck [100] proposed a classification of iterative construction heuristics for the TSP into four classes: *ordered sequence*, *increasing path*, *subtour insertion* and *merged multiple subtours*. Each of these operates by adding one or more edges per iteration to a partial solution (collection of edges) until a tour is formed. The construction heuristics proposed in the literature for the TSSP and its variations also fit this classification although some are subpath based rather than subtour based.

A tour construction heuristic begins with an empty partial solution. At each iteration, one or more edges are added to the solution until a complete solution is constructed. The increasing path methods consider those single target-prize desirability criteria as in Section 3.2.1. Although ordered sequence methods rank edges and merged multiple subtours rank mergers of subtours, neither of these classes are applicable to ranking prizes. Merged multiple subtours would be better classified as an "improvement of infeasibility" method rather than a construction method, where infeasibility is due to there being several subtours rather than a single tour. The subtour insertion methods maintain a subtour and at each iteration a vertex is inserted into the subtour according to some criterion by replacing a single edge in the subtour with two edges adjacent to the candidate node. These methods are essentially vertex-oriented since a vertex is selected and inserted at each iteration.

3.2.2.2 Subpath Insertion Criteria

In selecting a single target-prize we may ignore prizes which are close to the direct trajectory to our chosen target-prize. If we wish to target some prize then we may wish to use the time invested in travelling to it more profitably by making a detour to pick up prizes which are "on the way" for little extra effort. Suppose by some method we have a chosen a *seed prize* which we wish to target. We may be able to improve the overall efficacy of this target by inserting prizes onto a subpath with the seed-prize as the subpath tail. Append some prizes *after* the seed-prize wouldn't be helpful since we don't know the next seed-prize. However, it would be useful to sort out which prizes to consider for insertion, which are considered post seed-node, and those which are not considered at all. The correct selection question is whether a node should be inserted *here* as opposed to later or be left out but *not* whether a node should be inserted here or left out. Either way, we only implement the head of the subpath so no distinction between pre-seed and post-seed insertions is then necessary.

Golden, Bodin, Doyle and Stewart [88] and Johnson and Papadimitriou [111] give a number of

Algorithm 3.10 strategy GREEDY EN ROUTE INSERTION

Input: s *// Seed node.*
Input: γ *// Deviation tolerance.*

$P \leftarrow (r, s)$ *// Initial subpath.*
$finished \leftarrow$ **false**
while (**not** $finished$) **do**
  $K \leftarrow \{k \in Q \setminus V(P) : \exists (i_1, i_2) \in E(P)$ such that
    $d_{i_1 k} + d_{k i_2} - d_{i_1 i_2} \leq (1+\gamma) d_{0s} - \ell(P)\}$
  if $(K = \emptyset)$ **then**
    $finished \leftarrow$ **true**
  else
    $k^* \leftarrow \text{argmax}_{k \in K} \max_{(i_1, i_2) \in E(P)} \frac{v_k}{d_{ik} + d_{kj} - d_{ij}}$
    Insert k^* between i and j on subpath P.
  end
end
target $\leftarrow$ head of subpath P.

end

insertion rules including nearest addition, nearest insertion, farthest insertion, cheapest insertion, greedy insertion and, for Euclidean problems, convex hull insertion. Although these were defined in terms of subtours, we will apply them to nontrivial rooted subpaths. The approach is to maintain a candidate subpath and at each iteration some prize is *selected* and *inserted* into the subpath.

Cheapest Insertion. While there is a vertex available, select vertex k not on the subpath and subpath edge (i, j) which *minimizes* $d_{ik} + d_{kj} - d_{ij}$ and insert k between i and j.

Farthest Insertion. While there is a vertex available, select vertex k not on the subpath and subpath edge (i, j) which *maximizes* $d_{ik} + d_{kj} - d_{ij}$ and insert k between i and j. Although the farthest insertion heuristic has produced good results for the TSP we feel it would be inappropriate for the CPCP. It is credited with forming a rough outline of the final subpath and filling in the details but assumes that all of the prizes are to be visited.

Greedy Insertion. Laporte and Martello [138] proposed this insertion criteria for the OP. Select the prize k not on the subpath and subpath edge (i, j) which maximizes $\frac{v_k}{d_{ik}+d_{kj}-d_{ij}}$, such that the length of the subpath does not exceed the distance bound, and insert prize k between i and j.

Algorithm 3.10 GREEDY EN ROUTE INSERTION presents a corresponding strategy for the CPCP based on greedy insertion. Recall that vertex r is the origin vertex representing the location of player $\mathcal{X}$. Let $V(P)$ be the set of vertices of P and $E(P)$ be the set of edges of the subpath P and let $\ell(P)$ be the length of the subpath.

Algorithm 3.11 strategy GREEDY PAIR

// Target the greediest prize.

target $\leftarrow \text{argmax}_{i \in Q} \max_{j \in Q \setminus \{i\}} \left\{ \frac{v_i + v_j}{d_{Ai} + d_{ij}} \right\}$

end

3.2.2.3 Seed Selection

It remains to select a seed prize to initiate the insertion process. Firstly, we can employ any of the desirability criteria from Section 3.2.1.

An alternative is to consider a sequence of two prizes as the seed rather than just one prize and start the insertion process with the subpath $P \leftarrow (r, s_1, s_2)$ instead. Hence choose distinct prizes $i_1, i_2 \in Q$ which maximize $\frac{v_{i_1}+v_{i_2}}{d_{Ai_1}+d_{i_1 i_2}}$. Here we are considering the target-prize and the following prize as a unit. Ramesh and Brown [182] use this for their "double insertion rule" in an OP heuristic. We can actually use this idea as a strategy in its own right for comparison, as in Algorithm 3.11 GREEDY PAIR.

3.2.3 Avoidance and Observation of Opponent

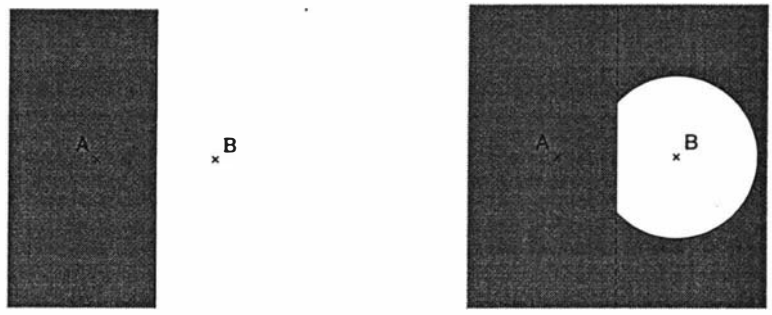
There is a fundamental uncertainty in the CPCP due to its dynamic game nature: each player's current and possible future decisions affect the other. Except for passively noting the disappearance of prizes, so far we have ignored the presence of the opponent. In this section we propose some simple methods to redress this situation.

Acknowledgement of an opponent's location and possible options adds a *temporal* dimension to the planning decisions. Priorities in collecting prizes arise out of criteria such as urgency and likelihood. **Likelihood** is the concept of "if we target a particular prize, are we likely to get there first?" **Urgency** covers "if we target a particular prize, how soon do we need to commit to it?"

There are three basic classes of strategy for dealing with an opponent. The first is outright ignorance, which is the case for the strategies of Sections 3.2.1–3.2.2. The second is simply to avoid the opponent by not considering unlikely prizes. The third is to attempt to predict the opponent's future local movements.

3.2.3.1 Guarantee and Avoidance

Guarantee is the concept that the opponent targets all of the prizes concurrently under the assumption that we cannot observe the opponent for the remainder of the game. Consequently, we conservatively assume that any prize we may visit could be the one that the opponent targets directly. A basic device is the **guarantee status** of each prize, i.e., to which player is each prize closer? In other words, which prizes can player $\mathcal{A}$ guarantee that if player $\mathcal{A}$ targets that prize then player $\mathcal{B}$ cannot get the prize? The shaded region of Figure 3.1(a) represents the set of prize locations which are guaranteed to player $\mathcal{A}$. The prize locations on the boundary (halfway) line



(a) Guarantee ($r = \infty$) (b) Avoidance ($r = \frac{2}{3}d_{AB}$)

Figure 3.1: Guarantee and Avoidance

Algorithm 3.12 strategy GUARANTEED NEAREST NEIGHBOUR

$S \leftarrow \{j \in Q : d_{Aj} \leq d_{Bj}\}$
if $(S = \emptyset)$ **then** $S \leftarrow Q$
target $\leftarrow \text{argmin}_{j \in S} d_{Aj}$

end

are those for which $d_{Aj} = d_{Bj}$.

The methods from Sections 3.2.1 and 3.2.2 can be simply modified to only consider guaranteed prizes, e.g., Algorithm 3.12 GUARANTEED NEAREST NEIGHBOUR and Algorithm 3.13 GUARANTEED GREEDY. Note that if there are no guaranteed prizes, the strategies revert to their nonguaranteed version.

Avoidance relaxes the concept of guarantee by selecting a set of prizes guaranteed to the opponent for which the likelihood that the opponent will target those prizes is high. It is overly drastic to exclude from consideration all prizes not currently guaranteed. What, for example, would you target if there are no prizes guaranteed to you but ample for your opponent? Simple observation of how the guarantee status of prizes change as the players move suggests that we could target those prizes which are not close to player B and hence have a higher likelihood

Algorithm 3.13 strategy GUARANTEED GREEDY

$S \leftarrow \{j \in Q : d_{Aj} \leq d_{Bj}\}$
if $(S = \emptyset)$ **then** $S \leftarrow Q$
target $\leftarrow \text{argmin}_{j \in S} \frac{v_j}{d_{Aj}}$

end

Algorithm 3.14 strategy NEAREST NEIGHBOUR AVOIDANCE

$S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq d_{Bj}\}$
target $\leftarrow \text{argmin}_{j \in S}\{d_{Aj}\}$

end

Algorithm 3.15 strategy GREEDY AVOIDANCE

$S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq d_{Bj}\}$
target $\leftarrow \text{argmax}_{j \in S}\left\{\frac{v_j}{d_{Aj}}\right\}$

end

than those close and guaranteed to player B. Also, urgent prizes are those whose status changes quickly.

Practically, we use a dynamic exclusion zone around the opponent by not considering prizes within a fixed radius of, and guaranteed to, the opponent. The shaded region of Figure 3.1(b) represents the set of prize locations within which player $\mathcal{A}$ would consider target prizes. An alternative, which we do not consider further, is to rule out a particular group of prizes, such as the h nearest guaranteed neighbours of the opponent. In general we can have method which firstly determines a set of prizes, S, to select a target from and then selects a target-prize from S. Then guarantee corresponds to $S \leftarrow \{j \in Q : d_{Aj} \leq d_{Bj}\}$ and radius-and-guarantee corresponds to $S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq d_{Bj}\}$. Note that *guarantee* is equivalent to $r = \infty$. Algorithm 3.14 NEAREST NEIGHBOUR AVOIDANCE and Algorithm 3.15 GREEDY AVOIDANCE are examples of such avoidance strategies.

3.2.3.2 Observation

Observation is simply noting which prize the opponent is currently targeting and selecting a target-prize accordingly. However, observation is not mutually exclusive with avoidance or guarantee but, rather, is complementary. However we could think of guarantee as an extreme realisation of both avoidance and observation.

Consider the simple strategy GUARANTEED NEAREST NEIGHBOUR of Algorithm 3.12 in which we have implicitly assumed that the opponent will target any prize which is closer to them and, consequently, that the only prizes which we will consider to target are those to which we are currently closer. Suppose, instead, we attempt to model the set of prizes which we consider the opponent could be targeting. A **prize-ts** (or **opponent prize target set**) is a subset of remaining prizes which estimates the possible target prizes of the opponent which are not sufficiently distinguishable via observation of the opponent's movements. We assume that the opponent must travel via at least one of the prizes in the *prize-ts* and hence the earliest arrival

Algorithm 3.16 strategy TARGET SET NEAREST NEIGHBOUR AVOIDANCE

Input: $T \subseteq P$ *// prize-ts.*
Input: r *// Avoidance radius.*

// Target the nearest prize conditionally guaranteed via prize-ts.
$S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq \min_{i \in T}\{d_{Bi} + d_{ij}\}\}$
target $\leftarrow \text{argmin}_{j \in S}\ d_{Aj}$

end

Algorithm 3.17 strategy TARGET SET GREEDY AVOIDANCE

Input: $T \subseteq P$ *// prize-ts.*
Input: r *// Avoidance radius.*

// Target the most greedy prize conditionally guaranteed via prize-ts.
$S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq \min_{i \in T}\{d_{Bi} + d_{ij}\}\}$
target $\leftarrow \text{argmax}_{j \in S} \left\{\frac{v_j}{d_{Aj}}\right\}$

end

time of player $\mathcal{B}$ at some prize $j \in Q$ is $\min_{i \in T}\{d_{Bi} + d_{ij}\}$. The corresponding avoidance criterion is $S \leftarrow \{j \in Q : \min\{r, d_{Aj}\} \leq \min_{i \in T}\{d_{Bi} + d_{ij}\}\}$.

The simplest heuristic strategy based upon these ideas is TARGET SET NEAREST NEIGHBOUR AVOIDANCE of Algorithm 3.16. Analogously, we can define the strategy TARGET SET GREEDY AVOIDANCE, as in Algorithm 3.17.

3.2.3.3 Building a prize-ts

A *prize-ts* allows a degree of flexibility in how coarse an estimate of the opponent's target-prize is determined. Clearly, the more refined the *prize-ts* the more potential target-prizes are available. Note that there is also the possibility that we substantially mis-identify the *prize-ts* so we need to be able to *check* whether the opponent's movements are consistent with our *prize-ts* model, so as to be able to *refine* it if necessary and also to be able to *build* a new one. We also require an *evaluator* method which provides a target-prize when given a *prize-ts*.

A straightforward method for estimating which prizes the opponent is targeting at each step

is to consider the current bearing, ψ_B, of player $\mathcal{B}$ at the immediately preceding step. Let θ_{Bj} be the bearing of prize j from the immediately preceding location of player $\mathcal{B}$. Then let the *prize-ts* be the set of remaining prizes whose bearing deviates from ψ_B by at most some threshold parameter θ_{thresh}, i.e.,

$$T = \{j \in Q : |\psi_B - \theta_{Bj}| \leq \theta_{thresh}\}.$$

The least confident *prize-ts* is that consisting of all the remaining prizes, e.g., when $\theta = 2\pi$. Certainly this is a possibility, especially if there is no history or the opponent has just claimed a prize. However, a better idea is to we use the recent historical observation of the opponent to predict which prizes the opponent is targeting.

Some possible approaches for estimating the current bearing, ψ_B, of player $\mathcal{B}$ include the bearing of the immediately preceding step, a moving average of the bearing of the previous few steps and a range of bearings of the previous few steps, i.e., tracking the actual variability in the observed movements of player $\mathcal{B}$. We only implement the bearing of the immediately preceding step, when available, since this is sufficient.

Hence suppose we have estimated the bearing, ψ_B, of player $\mathcal{B}$. Then a *prize-ts* can be constructed via either of the following two methods.

BUILD-PRIZETS (BEARING): Build a prize-ts consisting of those remaining prizes which deviate in bearing from ψ_B by at most θ_{thresh}. Note that this may give an empty prize-ts in which case we may stipulate that the prize closest in bearing to ψ_B must be included.

BUILD-PRIZETS (INSERTION): Let $s \in Q$ (a seed prize) be the prize closest in bearing to ψ_B. Build a prize-ts consisting of those remaining prizes $j \in Q$ such that

$$\min\{d_{Bj} + d_{js} - d_{Bs}, d_{Bs} + d_{sj} - d_{Bj}\} \leq d_{thresh}.$$

The parameters θ_{thresh} and d_{thresh} determine the degree of conservatism in the construction of a *prize-ts*.

3.3 Combinatorial Subproblem Based Strategies

Consider the heuristic strategies outlined in Sections 3.2.1–3.2.3. We can develop complementary subpath improvement heuristics only if we have an objective for the subpath decision problem. Hence it may be more appropriate to formulate combinatorial optimization subproblems based on the TSSP and develop specialised construction and improvement heuristics to solve these. Such subproblem formulations represent a static snapshot of the evolving dynamic situation such as employed to solve DVRSPs.

In the following, we distinguish between three different subpath structures. A **closed subpath** is one in which both the origin and destination locations are fixed, an **open subpath** is one in which the origin location is fixed and the destination location is free, and a **floating subpath** is one in which both the origin and destination are free.

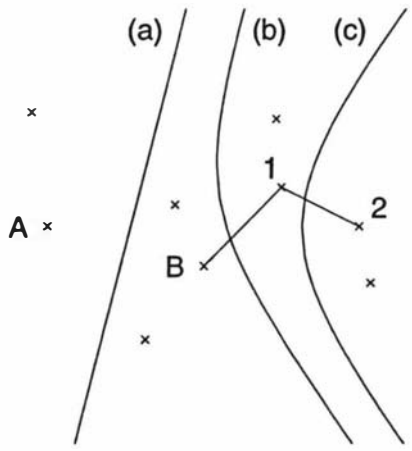


Figure 3.2: A Guarantee Partition.

3.3.1 Guarantee Subpath

A prize $j \in V$ is **guaranteed** to player $\mathcal{A}$ if $d_{Aj} \leq d_{Bj}$ and is **strictly guaranteed** to player $\mathcal{A}$ if $d_{Aj} < d_{Bj}$. Should player $\mathcal{A}$ target a strictly guaranteed prize then player $\mathcal{B}$ cannot possibly intercept that prize. We can extend this concept to an open subpath.

Definition 3.3.1

Let P be the sequence or subpath $(i_1, i_2, \ldots, i_m)$ and let $a_{i_k} = \sum_{q=1}^{k-1} d_{i_q i_{q+1}}$ be the corresponding arrival-time of player $\mathcal{A}$ at each prize $i_k \in \{i_1, i_2, \ldots, i_m\}$. P is **guaranteed** to player $\mathcal{A}$ if $a_j \leq d_{Bj}\ \forall j \in P$ and is **strictly guaranteed** to player $\mathcal{A}$ if $a_j < d_{Bj}\ \forall j \in P$. The **guaranteed value** of P is $\sum_{j \in P: a_j < d_{Bj}} v_j + \frac{1}{2} \sum_{j \in P: a_j = d_{Bj}} v_j$. □

Note that, by the triangle inequality, P is guaranteed if and only if $a_{i_m} \leq d_{Bi_m}$ and P is strictly guaranteed if and only if $a_{i_m} < d_{Bi_m}$.

The **guarantee subpath subproblem** is to determine a subpath guaranteed to player $\mathcal{X}$ of maximal guaranteed value. This is a special case of the open-subpath version of the **Orienteering Problem with Time Windows** (OPTW). The prize guarantee subproblem does not require earliness time windows, only the deadlines which have value $\ell_j = d_{Bj}$. Also the deadlines are significantly special since they satisfy a form of triangle inequality where $\ell_{i_2} \leq \ell_{i_1} + d_{i_1 i_2}\ \forall i_1, i_2 \in V$. Appendix 3.A.3 outlines a local search method for the prize guarantee subproblem. In Figure 3.2, the prizes guaranteed to player $\mathcal{A}$ are those to the left of line (a), while player $\mathcal{B}$ is guaranteed all those prizes to the right of line (a). Thus player $\mathcal{A}$ may choose only to target a prize which is currently guaranteed. Algorithm 3.18 GUARANTEE-SUBPATH presents the corresponding CPCP strategy.

Suppose we have identified a *prize-ts*, T. Then we can consider relaxing the idea of guarantee slightly by assuming that player $\mathcal{B}$ will claim one of the prizes in T prior to claiming any prize not in T. This assumption is denoted by $\mathcal{B} \rhd T$. The standard prize guarantee is equivalent to $T = V$.

Algorithm 3.18 strategy GUARANTEE-SUBPATH

Construct a good guaranteed subpath.
target $\leftarrow$ head of the subpath.

end

Algorithm 3.19 strategy TARGET-SET GUARANTEE-SUBPATH

Input: T *// prize-ts.*

Construct a good guaranteed subpath contingent on $\mathcal{B} \triangleright T$.
if $(T = \emptyset)$ **then**
target $\leftarrow \emptyset$
else
target $\leftarrow$ head of the subpath.
end

end

Definition 3.3.2

Let $\ell_j = \min_{i \in T}\{d_{Bi} + d_{ij}\}\ \forall j \in V$. A subpath $P = (i_1, i_2, \ldots, i_m)$ is **guaranteed contingent on** $\mathcal{B} \triangleright T$ if $a_j \leq \ell_j\ \forall j \in P$. The **guaranteed value of** P **contingent on** $\mathcal{B} \triangleright T$ is $\sum_{j \in P: a_j < \ell_j} v_j + \frac{1}{2} \sum_{j \in P: a_j = \ell_j} v_j$. □

Algorithm 3.19 TARGET-SET GUARANTEE-SUBPATH presents the corresponding CPCP strategy.

Finally, an exact solution to guarantee subpath subproblem is used as the basis of Algorithm 3.20 PRIZE-GUARANTEE (see Section 6.A.2.3).

3.3.2 Horizon OP

Existing combinatorial optimization problems can be employed as subproblems giving directly applicable strategies for the CPCP. The **horizon-OP** is an open subpath OP with the deadline defined as a rolling planning horizon. The closed subpath variation (with specified destination location) is also called the **direct-path** in Section 8.2.2.2.

Algorithm 3.20 strategy PRIZE-GUARANTEE

Construct a maximal guaranteed subpath.
target $\leftarrow$ head of the subpath.

end

Algorithm 3.21 strategy HARVEST PATH

Construct a good harvest subpath.
target ← head of subpath.

end

3.3.3 Harvesting Paths

Algorithm 3.8 GREEDY introduced the idea of maximizing the value of a target-prize divided by the distance that must be travelled to claim it (the "bang-for-buck"). The **harvesting rate** generalizes this bang-for-buck idea to a subpath.

Definition 3.3.3

Let $P = (i_1, i_2, \ldots, i_m)$ be a subpath. The **value** of the subpath is $v(P) = \sum_{j \in P} v_j$ and the **length** of the subpath is $\ell(P) = a_{i_m}$. The **harvesting rate** of the subpath is

$$\hbar(P) = \frac{v(P)}{a_{i_m}}$$

where $v(P)$ is the value sum of the prizes on P and $\ell(P)$ is the length of P. □

We wish to determine a subpath of maximum harvesting rate from the current location of player $\mathcal{A}$. We may optionally specify either an upper bounding or lower bounding constraint on $\ell(P)$ but not both. We consider two types of harvesting rate path; a *harvest path* has a fixed destination location but an *open harvest path* has no such destination location.

Algorithm 3.21 HARVEST PATH presents the corresponding CPCP strategy.

3.3.4 $\mathcal{BA}$-Path and $\mathcal{ABA}$-Path

$\mathcal{BA}$-paths and **$\mathcal{ABA}$-paths** generalize the idea of a guarantee subpath to include some short term response from player $\mathcal{B}$, but remain reasonably conservative; an **$\mathcal{ABA}$-path** further includes an element of contingent planning for player $\mathcal{A}$.

3.3.4.1 $\mathcal{BA}$-Path

Suppose player $\mathcal{B}$ proposes a guaranteed subpath P_B and then player $\mathcal{A}$ responds with a subpath P_A which is *conditionally guaranteed* upon $\mathcal{B} \rightarrow P_B$. Player $\mathcal{A}$'s objective is to find the best value response subpath given player $\mathcal{B}$'s subpath. Player $\mathcal{B}$'s objective is to maximize the value of player $\mathcal{B}$'s subpath less the value of player $\mathcal{A}$'s best response subpath. There is no point in looking at player $\mathcal{B}$'s objective to maximize the value of his path, since then he would just take the best possible prize guarantee subpath. Essentially the tradeoff for player $\mathcal{B}$ here is between his own harvesting and the prevention of player $\mathcal{A}$'s harvesting. Clearly, if there is no guarantee path for player $\mathcal{B}$ then this method is just the same as paranoid for player $\mathcal{A}$.

Algorithm 3.22 strategy $\mathcal{ABA}$-PATH

Construct a good $\mathcal{ABA}$-path or $\mathcal{BA}$-path.
target $\leftarrow$ head of the $\mathcal{A}$-subpath.

end

3.3.4.2 $\mathcal{ABA}$-Path

Suppose player $\mathcal{A}$ proposes a guaranteed path P_A. Then player $\mathcal{B}$ responds with a subpath P_B which is *conditionally guaranteed* upon $\mathcal{A} \rightarrow P_A$. Finally player $\mathcal{A}$ **extends** his subpath from the tail of P_A such that the extension is *conditionally guaranteed* upon $\mathcal{B} \rightarrow P_B$. The objective of the extension subpath phase for player $\mathcal{A}$ is to maximize the overall value of the subpath. The objective of player $\mathcal{B}$ is once again to maximize the value of player $\mathcal{B}$'s path less the value of player $\mathcal{A}$'s path. The overall objective of player $\mathcal{A}$ is to maximize the value of player $\mathcal{A}$'s overall subpath. Essentially the tradeoff for player $\mathcal{A}$ here is between delaying the point of contingency under a tighter set of deadlines and relaxing the response deadlines to player $\mathcal{B}$. Algorithm 3.22 $\mathcal{ABA}$-PATH presents the corresponding CPCP strategy.

More precisely, let r_A be the root vertex corresponding to player $\mathcal{A}$'s current location and let r_B be the root vertex corresponding to player $\mathcal{B}$'s current location. Player $\mathcal{A}$ proposes an initial nonempty guaranteed $\mathcal{A}$-subpath, $P_A = (i_1, i_2, \ldots, i_p)$, corresponding to the deadlines $\ell_{Ai} = d_{Bi}$. Player $\mathcal{B}$ responds with an initial (possibly empty) $\mathcal{B}$-subpath, $P_B = (k_1, k_2, \ldots, k_q)$, which is guaranteed with respect to $\mathcal{A} \rhd P_A$ with corresponding deadlines $\ell_{Bj} = \ell(P_A) + d_{i_p j}$. Finally player $\mathcal{A}$ responds with a (possibly empty) subpath, the $\mathcal{A}$-extension $P'_A = (v_1, v_2, \ldots, v_r)$, which is appended to P_A such that the resulting subpath, $P_A \cdot P_B$, is guaranteed with respect to $\mathcal{B} \rhd P_B$ with corresponding deadlines $\ell'_{Aj} = \ell(P_B) + d_{k_q j}$. Let a'_{Aj} be the corresponding arrival-times of player $\mathcal{A}$ at each prize $j \in P'_A$ given that $\mathcal{A} \rhd P_A$. The corresponding guaranteed value of P'_A is

$$v_G(P'_A : \mathcal{A} \rhd P_A, \mathcal{B} \rhd P_B) = \sum_{j \in P'_A : a'_{Bj} < \ell'_{Aj}} v_j + \frac{1}{2} \sum_{j \in P'_A : a'_{Bj} = \ell'_{Aj}} v_j.$$

Also $\hbar(P'_A : \mathcal{A} \rhd P_A, \mathcal{B} \rhd P_B) = \frac{v_G(P'_A : \mathcal{A} \rhd P_A)}{a_{v_r}}$. We also define

$$\hbar(P_A \cdot P'_A : \mathcal{B} \rhd P_B) = \frac{v_G(P_A) + v_G(P'_A : \mathcal{A} \rhd P_A, \mathcal{B} \rhd P_B)}{a_{v_r}}.$$

Figure 3.3 illustrates the components of a $\mathcal{ABA}$-path. The hollow circles represent the origin-vertices of player $\mathcal{A}$ and player $\mathcal{B}$ respectively. The dashed arrows represent the definition of the deadlines for player $\mathcal{B}$ and the solid arrows represent the definition of deadlines for player $\mathcal{A}$. The objective of player $\mathcal{B}$ is to choose P_B so as to maximize $\hbar(P_B) - \hbar(P'_A : \mathcal{A} \rhd P_A, \mathcal{B} \rhd P_B)$. The objective of player $\mathcal{A}$ is to choose P_B to maximinimaximize $\hbar(P_A \cdot P'_A : \mathcal{B} \rhd P_B)$, i.e.,

$$\text{argmax}_{P_A} \left\{ \min_{P_B} \left\{ \max_{P'_A} \hbar(P_A \cdot P'_A : \mathcal{B} \rhd P_B) \right\} \right\}.$$

Finally, suppose we have constructed a prize-ts, T. Firstly we assume that $\mathcal{B} \rhd T$ and hence the initial deadlines are defined by $\ell_{Aj} = \min_{i \in T}\{d_{Bi} + d_{ij}\}$. Secondly we assume that player

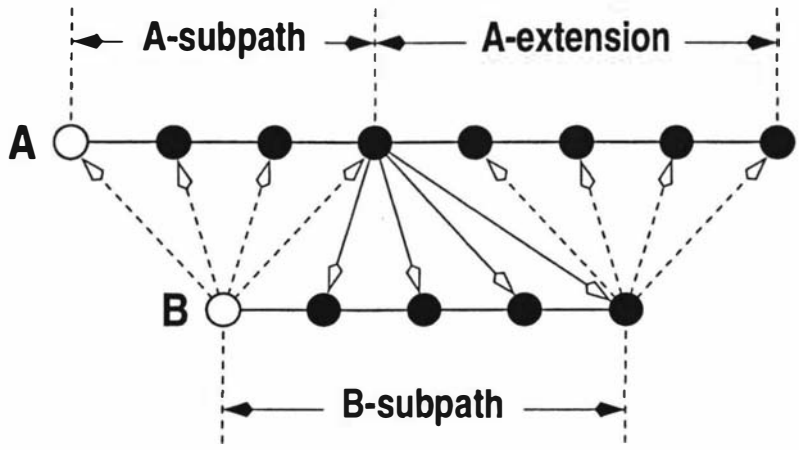


Figure 3.3: Structure of $\mathcal{ABA}$-path Components

Algorithm 3.23 strategy TARGET-SET $\mathcal{ABA}$-PATH

Input: T *// prize-ts.*

Construct a good $\mathcal{ABA}$-path or $\mathcal{BA}$-path contingent on $\mathcal{B} \triangleright T$.
if $(T = \emptyset)$ **then**
    target $\leftarrow \emptyset$
else
    target $\leftarrow$ head of the $\mathcal{A}$-subpath.
end

end

$\mathcal{B}$ must select the first prize on $P_{\mathcal{B}}$ from those prizes in T. The resulting CPCP strategy is TARGET-SET $\mathcal{ABA}$-PATH of Algorithm 3.23.

3.3.5 Cooperate

If the overall deadline is sufficiently restrictive then the players may each be able to claim more value in prizes if they choose to cooperate. We can calculate the **cooperative value**, Ω, which is the maximum value that the players can jointly claim within the overall deadline λ if they cooperate completely. This subproblem is equivalent to an open-subpath variation of the **Multiple Depot Team Orienteering Problem** (MDTOP).

A problem instance is **guarantee determined** (or **static**) if the value of the maximal guaranteed subpaths for each player sum to Ω. Then the guaranteed subpath of each player is an optimal strategy.

Conceivably this could provide the basis of a strategy for the CPCP in which the players choose to cooperate even when the value of the maximal guaranteed subpaths for each player sum to less than Ω. There are two distinct difficulties:

(I). How do they ensure that the opponent does not do a double-cross, that is, how can the players ensure that once they enter into cooperation the cooperation is then binding upon what they agree to when they enter into it? This would involve some kind of rule change to introduce *alliance contracts* that dynamically bind the players into a team for a specific (possibly permanent) period of time.

(II). The more troublesome problem is how the players share the spoils of such cooperation. The cooperative value Ω is the maximum sum of the values the players jointly claim. There may be multiple optimal subpath-pairs so how do we select between them. Also a joint value which is slightly less than Ω may enable the value to be shared more *fairly* in comparison with what the players could expect from a GUARANTEED subpath.

Hence we do not attempt to construct a strategy based upon cooperation with the opponent.

3.4 Necessary Principles of Strategy

Sections 3.2–3.3 have proposed basic strategies based on prize desirability and subproblems. Although these have not been extensively developed, they are useful indicators of what cannot be captured by a combinatorial optimization subproblem formulation of a player's strategic decision problem. From preliminary simulations employing these strategies, we can observe four necessary principles of strategy that must be addressed by a solution paradigm for the CPCP but cannot be adapted from the TSSP/DVRSP solution paradigm.

3.4.1 Dynamism and Uncertainty

Recall that no one player completely controls the final outcome of the game. Hence a strategy must deal *dynamically* with the perpetual *uncertainty* in how the opponent will play the game.

We require strategies which are **robust**, i.e. are designed to perform well in most situations. Planning with little *risk* and a high degree of *security* is *conservative* planning, but what *threshold* of risk is acceptable? This depends upon the degree of *detail* with which we choose or are able to plan; by planning with more detail, we can generate plans with greater security. Just as scale of step length relative to the displacement of prizes determines the coarseness of movements in the game, so does detail of planning determine how well the next one-step move can be planned. Detail involves both "scope" (breadth) and "horizon" (depth) of planning.

Information about future states of the game is unknown or imprecise; hence near-term events should be planned in more detail but longer term guiding policy is essential. Over a longer planning horizon we are representing fewer possible consequences of an action; longer term plans are necessarily less detailed since a player can only roughly estimate the opportunities available from some future position.

Decisions regarding single prizes are simple, but deciding between prize combinations is more difficult. Planning a *guaranteed* (i.e. non-interceptable, riskless) path is increasingly narrow-minded, but could be useful as a static positional evaluator. Considering *direct interception* as a threat in evaluating a plan is not usually effective. Overall, we must evaluate possible plans in such a way as to gauge the possible risks involved.

3.4.2 Hierarchical Stage Setting

Each player is involved in a continual requirement for action but the actual satisfaction of an objective at the end of the game conveys very little information about the overall process required to attain it. The future plays that are possible from any state during a game greatly exceed the few which may satisfy the current objective. The intermediate states encountered usually have no associated payoff or direct information concerning their relative advantage, yet they play a stage-setting role in establishing opportunities for the accumulation of prizes, upon which the objective is based. Of the variety of possible planning horizons available, the short- to medium-term is the most effective for the purposes of evaluating plans. It is not sufficient to plan *myopically* just a few steps at a time without medium term direction since we could not plan for these intermediate positioning steps.

It is clearly not sufficient in general to plan just a few steps ahead in time. Much effort must be given to setting the stage for future opportunities. This must involve long-term planning of credible goals as well as game-like coordination including provocation of and response to the opponent. Since we plan from a given time slice onwards, we must plan at a number of time or goal horizons, each a refinement of the last. In addition, we must manage the coupling of decisions or recursions between horizons. In this way we can design and implement goals at various stages out from our current step-decision which serve to consider future payoffs at all horizons in an overall coordinated way.

3.4.3 Prediction of Opponent's Behaviour

How our opponent will play is the major source of uncertainty in our decision making. If we could perfectly predict exactly how he would respond to any action(s) on our part, then we could completely determine the outcome of the game in terms of our objective. If we could satisfy our objective in such a way that our opponent cannot intercept our path then we could safely ignore our opponent's actions in our planning.

We characterise *cognizance* of our opponent as a continuum ranging from requiring a direct response to his every move through to regarding his activities as essentially independent, over a suitable planning horizon. Optimization is "complete cognizance" in matching our actions to all possible reactions of our opponent and vice versa.

Observation of the opponent's *behaviour* leads to the possibility of *learning*. Information conveyed by intermediate states in the play of a game could be used to construct a representational model of the opponent's behaviour in order to improve our future decisions. **Interaction** debates the merits of coming into close proximity with our opponent given the concomitant sharp increase in uncertainty over the local decision horizon. How does behaviour alter when we are in direct conflict over the local prizes? Are there occasions when conflict is not a choice but a necessity?

3.4.4 Contingency Planning

One must always put oneself in a position to choose between alternatives. Adhering to the traditional substantive rationality represents one extreme, wherein we determine all possible alternative

scenarios and thus clinically exhaust all risk. At the opposite extreme, we may construct a fixed path and evaluate it without the possibility of building in alternative decision forks. For example, consider generating two fixed paths which share the initial steps but fork at some future point. When conservatively evaluating a single path we select some set of representatively "bad" possible scenarios as our opponent's response to that plan. In taking these two paths independently we probably would not consider the same bad scenarios. If, however, we conservatively evaluate the two paths together as a **contingent** plan, then we would select one set of bad response scenarios and, for each such scenario, determine our best choice at the decision fork, rather than implicitly selecting the worst choice when the plans were regarded as independent.

In general, we generate a selection of future partial states at which our plan may fork. In this way we can plan for future decisions when the relevant state information becomes available, instead of attempting to predict it given our present conservative extrapolations. Hence contingent planning produces tree structures where the branching of the tree corresponds to an assessment of the state of play at that time in selecting a subplan to follow. Again, the more detail in planning for contingent forks, the more complex the search becomes.

Threat is the *proactive* form of contingency; it is the not necessarily subtle manipulation of our opponent by maintaining many flexible alternative options. We plan to cover the initial steps of more alternatives than our opponent can cover and whose outcomes he can't resolve. It is this ability to confuse by use of alternative forks of plans that incites uncertainty in our opponent's attempts to defend a position or bank upon a particular sequence of actions on our part. However, threat can be a two-edged sword. Our opponent may also be attempting to manipulate our position by keeping his own options open with respect to his own objective. When threats balance, this could lead to a standoff. Perhaps our opponent is threatening some sequence of actions and we must respond defensively to that threat up until we can counter with a threat of our own. A key question is when should a standoff be initiated: when we have run out of secure alternatives near the perimeter of the prize region, or when we are central to the remaining prizes and have many active alternatives open? This is the tradeoff between *proactively* constructing more alternative plans and *reactively* following some of our existing ones.

Coda

▼ Summary

This chapter has applied the solution concepts of the TSSP and DVRSP solution paradigms to the CPCP. In the process, we have identified a number of key principles which cannot be resolved by adapting the existing combinatorial subproblem approach.

▼ Link

The next chapter designs a solution architecture based around these principles

3.A Appendices to Chapter 3

These appendices consider local neighbourhood search heuristics for the combinatorial optimization subproblems formulated in Section 3.3.

3.A.1 Local Neighbourhood Search Heuristics

Solution methods for combinatorial optimization problems as often classified as follows.

Construction Heuristics. Assemble a solution by selecting increasingly larger subsets of the component objects; once an object is selected is cannot be deselected.

Improvement Heuristics. Iteratively rearrange objects with respect to a set of operators, accepting an improved solution at each iteration until none better can be generated (local optima).

Search Heuristics. Local search methods (or hill climbing methods) make a series of steps, at each stage improving the current solution by moving to a neighbouring solution. This is usually done by considering the neighbouring solutions one at a time and moving to the first one which gives an improvement (a first-improving method) or considering all the neighbouring solutions and moving to the one which gives the greatest improvement (a best-improving method). Anderson [2] considers various such selection strategies.

Optimization Algorithms. A solution method which is guaranteed to find the best solution in finite time, e.g., branch and bound, implicit enumeration. Miller and Pekny [158, 159] have shown that it is possible to use parallel branch and bound algorithms to solve reasonably large instances of the asymmetric TSP up to 250,000 cities.

A **meta-heuristic** is a heuristic which guides another search heuristic. Meta-heuristics have been found to be effective solution methods for a number of combinatorial optimization problems, including the VRP (Gendreau, Hertz and Laporte [75]). **Tabu Search** (TS) (Glover [79, 80] and Glover and Laguna [82]) guides a local search heuristic by short-term prohibition of moves which

would "undo" moves already executed, medium-term search intensification and long-term search diversification. **Simulated Annealing** (SA) (Kirkpatrick, Gelatt and Vecchi [129]) modifies the hill climbing local search heuristic by allowing, with probability decreasing over time, moves which lead to a worse solution than the current solution. **Genetic Algorithms** (GA) (Goldberg [84]) mimic evolutionary genetic processes of random mutation and crossover to combine "good components" of solutions from a pool of solutions to "discover" good solutions. **Threshold Accepting** (TA) (Dueck and Scheuer [62]) is similar to SA except that the probability of accepting a worse solution is constant throughout. Dueck [61] also proposes two other simple meta-heuristics: the **Great Deluge Algorithm** (GDA) and the **Record-to-Record Travel** (RTR). TA, GDA and RTR have been found to outperform SA on some TSPs. **Greedy Randomized Adaptive Search Procedures** (GRASP) (Feo and Resende [67]) is a meta-heuristic that uses randomized restart to search many portions of the solution space. **Scatter Search** (SS) (Glover [81]) is a population-based meta-heuristic similar to GA which uses an initial population of solutions obtained from some basic heuristic and new solutions are formed by taking linear combinations of solutions from the population. The design of meta-heuristics for the subproblems of Section 3.3 is considered beyond the scope of this thesis. However, the following sections of this appendix provide the foundations for such meta-heuristics by designing local neighbourhood search heuristics for these subproblems.

A **local search heuristic** is an improvement heuristic which starts with some solution to a particular problem and searches a subset of the space of all solutions to that problem for better ones. Hence to define a local search heuristic we must define the subspace of all solutions to search about the current solution (the **local neighbourhood**) and a **selection criterion** for determining which solution from the current local neighbourhood to adopt as the next solution.

A **local operator** is an operator which makes a "local" change to a given solution. A local neighbourhood is the defined as all solutions which can be obtained from the current solution by applying one of a given set of local operators. A solution is **locally optimal** with respect to a local operator if apply the local operator cannot improve the solution. Hence, a local search heuristic improves a given initial solution by applying a number of local operators until the solution is locally optimal with respect to each local operator.

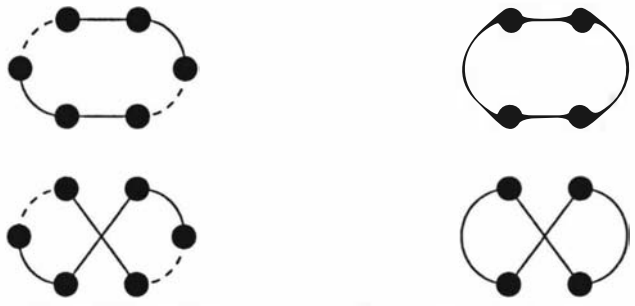
Local search is very intensive (myopic); it just tries to improve the current solution as much as possible at each step (Morton and Pentico [162]). Punnen and Aneja [181] and Aarts and Lenstra [1] survey the field of randomized local search heuristics in combinatorial optimization.

3.A.2 Local Subpath Operators

Local operators for the TSSP and the VRP are almost exclusively defined in terms of tours or subtours. In this sections we adapt these local operators for subpaths via a visual specification scheme.

3.A.2.1 Defining Local Operators

Consider the **two-exchange** local tour operator of Lin and Kernighan [145] in which two non-adjacent edges are removed from the tour and replaced by two different edges such that the tour



(a) Two-Exchange (b) Two-Exchange (minimal)

Figure 3.4: Lin and Kernighan Two-Exchange Definition

remains connected. This is *defined visually* in Figure 3.4(a) in which '•' represents a vertex, '—' represents a subpath edge, and '- - -' represents a *wildcard edge* (either a single vertex or a subpath consisting of at least one edge). The upper tour defines the original tour, marking the significant edges and vertex, and the lower tour defines the tour resulting from applying the two-exchange operator. Figure 3.4(b) illustrates the minimal instance of the two-exchange where every wildcard is collapsed to a single vertex.

3.A.2.2 Subpath Insertion, Deletion and Improvement Operators

Figure 3.5 defines a number of basic subpath operations using the same definition scheme. The leftmost vertex represents the root vertex of the subpath (see the definition in Section 3.2). No other vertex can substitute for the root vertex since it represents the current location of a player. The upper subpath concisely defines the subpath on which the operator can be applied, locating significant vertices and restrictions on the location of vertices with respect to one another and the ends of the subpath. The lower subpath defines the new subpath by giving the edges added and edges removed.

Gendreau, Hertz and Laporte [74, 75] proposed the **generalized insertion** (GENI) operator which inserts a vertex into a subtour together with a local reoptimization of the subtour. Figure 3.6 defines some generalized insertion operators applied to subpaths and their inverse operators, i.e., some generalized deletion operators. Figure 3.7 defines additional subpath two-exchanges in which the subpath root is fixed but the subpath tail is changed; these are derived from equivalent three-exchanges for tours.

These local operators constitute the subpath insertion, subpath deletion and subpath improvement operators that will be employed in Appendices 3.A.3–3.A.6.

To implement a local operator, we must provide an evaluator function, which determines the change in path length, a *feasibility* function, which determines whether the operator constructs a feasible solution, and a *do* procedure, which actually modifies the current solution by applying the operator. A key consideration in designing local search operators is that the evaluator and feasibility functions must have computational complexity much less than the do procedure.

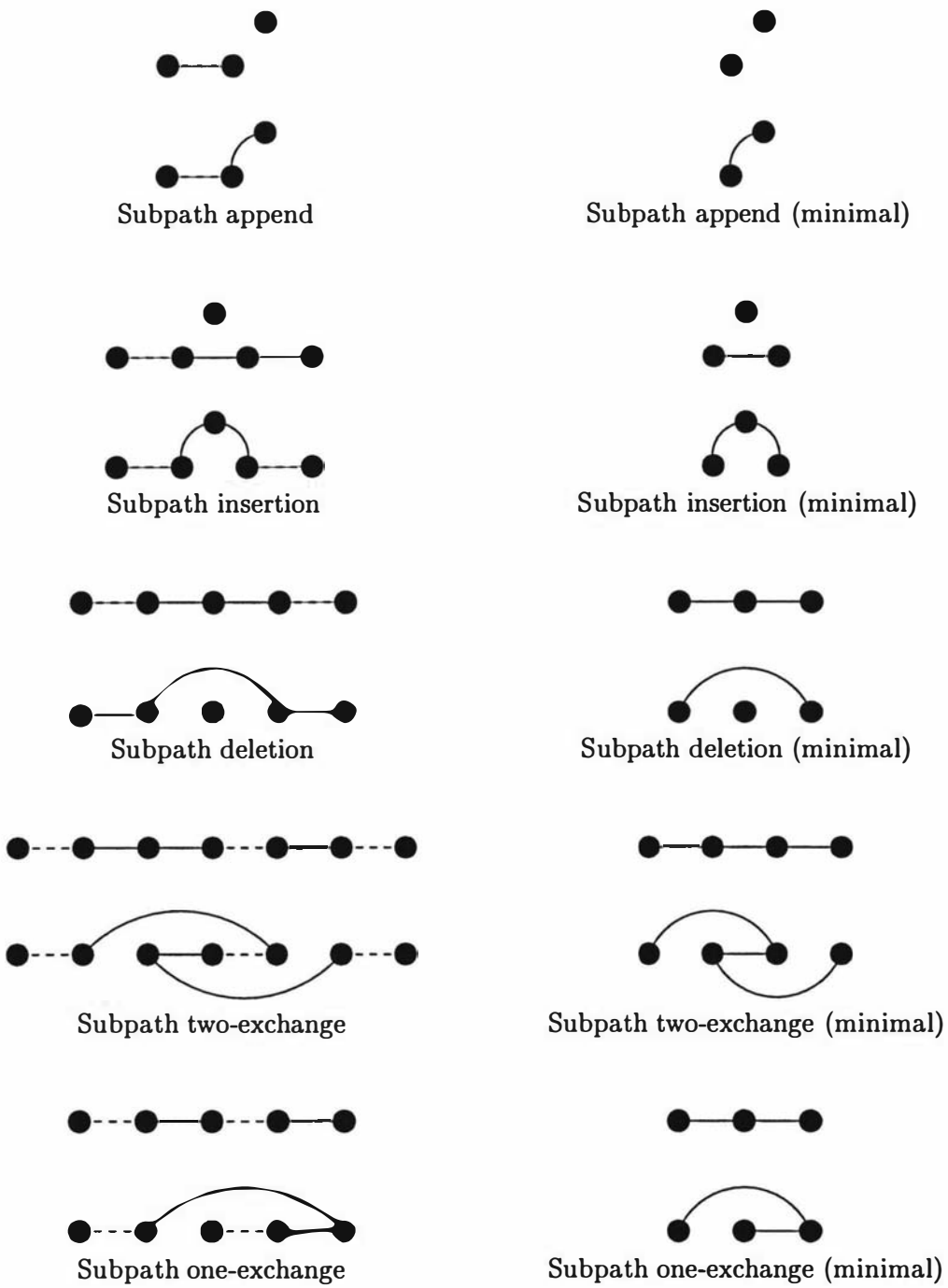


Figure 3.5: Basic Local Subpath Operators

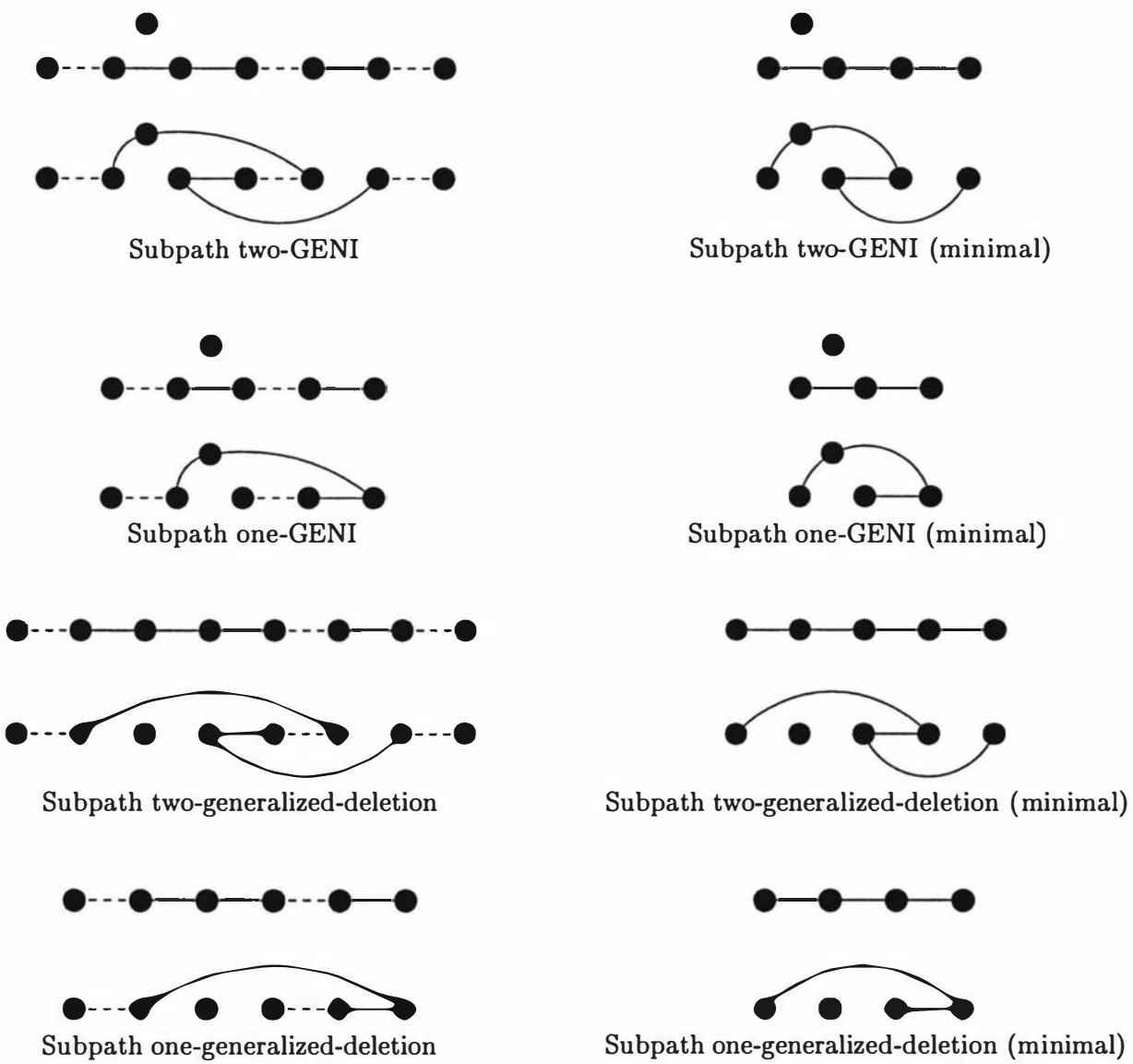


Figure 3.6: Generalized Subpath Insertion and Deletion operators

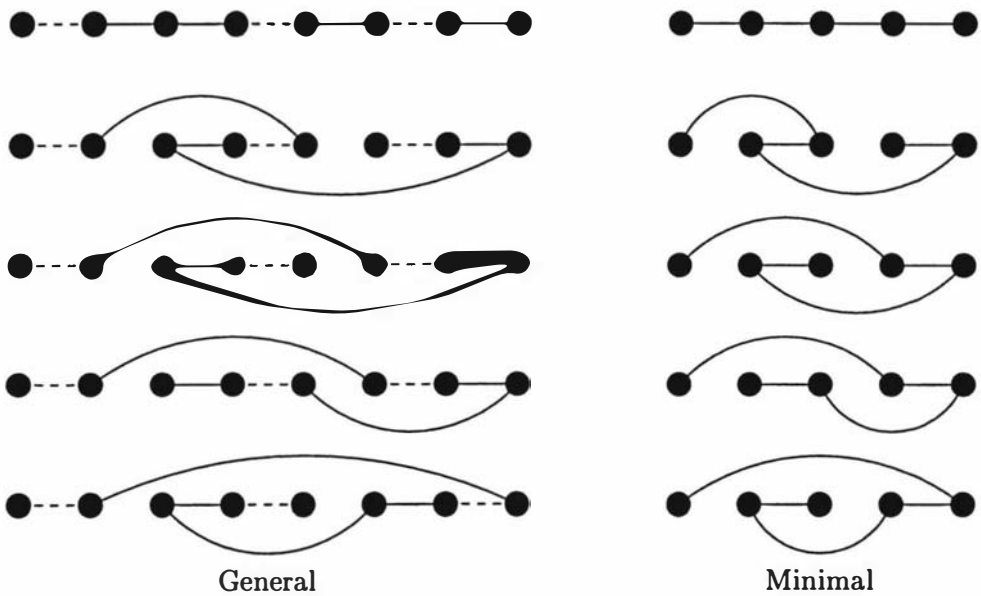


Figure 3.7: Additional Subpath Two-Exchanges

3.A.3 Local Search for PRIZE-GUARANTEE

Algorithm 3.24 presents the FAST PRIZE GUARANTEE local search heuristic for determining a good guaranteed subpath, as required by Section 3.3.1. Suppose the deadlines, ℓ_j $\forall j \in Q$, are set such that $\ell_j = \min\{\lambda, t + d_{Bj}\}$ where t is the current absolute simulation time and λ is the overall deadline. Let P_A be the planning subpath of player $\mathcal{A}$ with root vertex r. In applying a local operator we must ensure that the resulting subpath is feasible with respect to the deadlines. Since all distances are Euclidean distances, by the triangle inequality we need only check that the tail vertex of P_A has arrival time no later than its deadline.

3.A.4 Local Search for HORIZON-OP

Algorithm 3.25 presents the FAST HORIZON-OP local search heuristic for determining a good OP-subpath, as required by Section 3.3.2. The only feasibility concern is with respect to the horizon, $H \leq \lambda$.

3.A.5 Local Search for HARVEST-PATH

Algorithm 3.26 presents the FAST HARVEST PATH local search heuristic for determining a good harvest-path, as required by Section 3.3.3. Suppose we have the constraint $\ell_{min} \leq \ell(P_A) \leq \ell_{max} \leq \lambda$. Assume that ℓ_{min} is no greater than the shortest subpath from the origin through all the vertices and that $\ell_{max} \leq \lambda - t_A$. For harvest-path initialise with $P_A = (r)$. For open-harvest-path initialise with $P_A = (r, s)$, where s is a vertex which represents the destination location, and assume that $\ell_{max} \geq d_{rs}$.

Algorithm 3.24 heuristic FAST PRIZE GUARANTEE

```
// Local search heuristic for PRIZE-GUARANTEE.

finished ← false
P_A ← (r)
while (not finished) do
    if ∃ a feasible insertion then
        perform most-greedy insertion
    else if ∃ a length improving two-exchange then
        perform most-length-improving two-exchange
    else if ∃ a feasible deletion then
        perform least-greedy deletion
    else
        finished ← true
    end
end

end
```

Algorithm 3.25 heuristic FAST HORIZON-OP

```
// Local search heuristic for HORIZON-OP.

Input:
finished ← false
P_A ← (r)
while (not finished) do
    if ∃ a feasible insertion then
        perform most-greedy insertion
    else if ∃ a length improving two-exchange then
        perform most-length-improving two-exchange
    else if ∃ a feasible hr-improving deletion then
        perform least-greedy deletion
    else
        finished ← true
    end
end

end
```

Algorithm 3.26 heuristic FAST HARVEST PATH

```
// Local search heuristic for PRIZE-GUARANTEE.

finished ← false
P_A ← (r)
while (not finished) do
    if ℓ(P_A) < ℓ_min or ∃ a feasible hr-improving insertion then
        perform most-hr-improving insertion
    else if ∃ a length improving two-exchange then
        perform most-length-improving two-exchange
    else if ℓ(P_A) > ℓ_max or ∃ a feasible hr-improving deletion then
        perform most-hr-improving deletion
    else
        finished ← true
    end
end
```

3.A.6 Local Search for $\mathcal{ABA}$-PATH

Figure 3.8 defines the components of the FAST $\mathcal{ABA}$-PATH local search heuristic for determining a good $\mathcal{ABA}$-path, as required by Section 3.3.4. We can now expand each component.

Main. Set deadlines for the $\mathcal{A}$-subpath from player $\mathcal{B}$'s current location.The $\mathcal{A}$-subpath is initialised to the $\mathcal{A}$-origin, and possibly to the $\mathcal{A}$-destination. The $\mathcal{B}$-subpath is initialised to the $\mathcal{B}$-origin. If we are playing $\mathcal{ABA}$-PATH then we call '**$\mathcal{A}$-path**'. If we are playing $\mathcal{BA}$-PATH then we skip directly to '**$\mathcal{B}$-hotstart**'.

$\mathcal{A}$-path. Objective for player $\mathcal{A}$: maximize overall $\mathcal{A}$-harvesting-rate, i.e., of the initial $\mathcal{A}$-subpath and the $\mathcal{A}$-extension. Evaluate existing feasible $\mathcal{A}$-subpath using '$\mathcal{B}$-hotstart'. Evaluate each candidate move using '$\mathcal{B}$-hotstart'. Make a feasible, objective-improving *insertion* move is possible; otherwise make a length-improving *improvement* move if possible; otherwise make an objective-improving *deletion* move if possible; otherwise stop.

$\mathcal{B}$-hotstart. We have just created a new $\mathcal{A}$-subpath. Determine new deadlines for player $\mathcal{B}$'s subpath determined from the end of the $\mathcal{A}$-subpath. Note that the existing $\mathcal{B}$-subpath may be infeasible with respect to the new deadlines and may contain prizes which have now been visited on $\mathcal{A}$-subpath. Hence:

(i). set new deadlines

(ii). remove prizes which are claimed on $\mathcal{A}$-subpath

(iii). while $\mathcal{B}$-subpath is infeasible:

- make a length-improving improvement if possible;

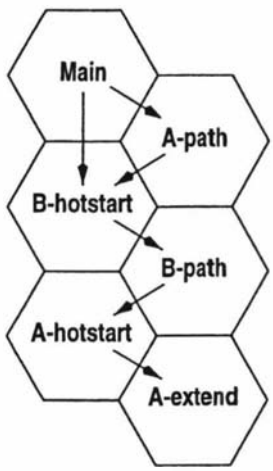


Figure 3.8: Structure of $\mathcal{ABA}$-PATH heuristic

- otherwise make the length-improving deletion which maximizes

$$\frac{-\Delta\ell(P_B)}{v_j}$$

Now we have a feasible $\mathcal{B}$-subpath. Note that if we have a ***prize-ts*** then we must insist that the first vertex on the $\mathcal{B}$-subpath, following the $\mathcal{B}$-origin vertex, is a prize from the ***prize-ts***.

$\mathcal{B}$-path. Objective for player $\mathcal{B}$: maximize overall $\mathcal{B}$-harvesting-rate less overall $\mathcal{A}$-harvesting-rate. Evaluate existing feasible $\mathcal{B}$-subpath using '$\mathcal{A}$-hotstart'. Evaluate each candidate move using '$\mathcal{A}$-hotstart'. Make a feasible, objective-improving ***insertion*** move if possible; otherwise make a length-improving ***improvement*** move if possible; otherwise make an objective-improving ***deletion*** move if possible; otherwise stop.

Note that if we have a ***prize-ts*** then we must insist that the first vertex on the $\mathcal{B}$-subpath, following the $\mathcal{B}$-origin vertex, is a prize from the ***prize-ts***.

$\mathcal{A}$-hotstart. We have just created a new $\mathcal{B}$-subpath. Determine new deadlines for player $\mathcal{A}$'s subpath determined from the end of the $\mathcal{B}$-subpath. Note that the existing $\mathcal{A}$-extension may be infeasible with respect to the new deadlines and may contain prizes which have now been visited on $\mathcal{B}$-subpath. Hence:

(i). set new deadlines

(ii). remove prizes which are claimed on $\mathcal{B}$-subpath

(iii). while $\mathcal{A}$-extension is infeasible:

- make a length-improving improvement if possible;
- otherwise make the length-improving deletion which maximizes

$$\frac{-\Delta\ell(P_A)}{v_j}$$

Now we have a feasible $\mathcal{A}$-extension.

$\mathcal{A}$-extend. Objective for player $\mathcal{A}$: maximize the overall $\mathcal{A}$-harvesting-rate. The initial $\mathcal{A}$-path is fixed (not, however, including the destination vertex) and operators can be applied only to the $\mathcal{A}$-extension. Make a feasible hr-improving insertion if possible; otherwise make a length-improving improvement if possible; otherwise make an hr-improving deletion if possible.

CHAPTER 4

Strategic Planning Architecture

When it is not necessary to make a decision, it is necessary not to make a decision.

— LORD FALKLAND'S RULE

This chapter proposes a Strategic Planning Architecture (SPA) and Dynamic Monitoring System (DMS) to capture and structure the principles identified in the previous chapter. The SPA is a dynamic, hierarchical, flexible, structural framework for computational strategies which considers different levels of aggregation of the prize set. It is based upon the premise of continually setting up for future payoffs. The building blocks focus on the interaction of planning horizons, contingent planning, cognizance of opponent and dynamic response to the opponent's behaviour.

4.0 Introduction

The objective of this chapter is to formulate a solution paradigm incorporating the four principles identified in Section 3.4, namely uncertainty and robustness, stage setting, cognizance of opponent and contingency planning. These concepts are usefully delineated by considering the difference between a scenario, a forecast and an observation.

- A **scenario** is a description of the future and the process of how to get there from the present (Meristö [155]). A scenario need not be an exact future course but rather an approximate representation. Multiple scenario analysis involves developing a representative set of mutually exclusive alternatives and assessing the player's response to each of these possible futures.
- A **forecast** assumes that it is possible to predict the future, assigning probabilities to the occurrence of each possible future. Thus a forecast is an estimate of what is likely to occur.

- An **observation** is the noting of an event as it occurs.

The structure we propose for a solution paradigm is an architecture organised as follows. Section 4.1 considers scenario generation and evaluation, incorporating aggregation, contingency planning, stage setting (planning for the long term) and uncertainty (robust detailed planning for the near term). Section 4.2 considers forecasting and observation, incorporating cognizance of the opponent's behaviour and dynamic scenario selection. The split between scenario generation and scenario selection is necessary to divide and conquer a complex problem (as advocated by Newell and Simon [166]) by modularizing the component problems in order to structure both our understanding and realistic implementation.

4.1 Aggregation and Contingency Planning

A strategy needs to balance the accumulation of prizes over a near term planning horizon, in a localised area, against the need to prepare positionally for continued prize accumulation over longer planning horizons. To be effective, both these extremes require the evaluation of future scenarios contingent upon uncertain events, i.e., the behaviour and movement selections of the opponent.

Contingency is the analysis of the consequences of a set of actions incorporating conditioning of the future plan upon some classification of future state(s). It can be viewed as the effective use of *threat* for the purposes of evaluation, i.e., **if** *state1* **then** *action1* **else if** *state2* **then** *action2*. Contingency cannot be adequately addressed using the subpath based paradigm of Chapter 3.

Ideally we would like to be able to plan for every possible future decision of our opponent. Given a scarce computational resource this is clearly unrealistic since contingency planning is computationally expensive. Hence contingency planning must be balanced against the both **aggregation** (the process of selecting and organising significant groups of prizes) and the **certainty** of the evaluation of each scenario. The issue is how to make effective use of the available computational resource to solve a range of problem sizes and structures such that we consider both the "forest" and the "trees" simultaneously.

These considerations suggest that we require a good understanding of what scenarios, evaluations and decisions are required at a given degree of aggregation (Section 4.1.1) and how those decision problems fit together to address near- and long-term planning (Section 4.1.2).

4.1.1 Framing Decision Problems

The necessity to make decisions implies that there is a context in which a decision problem is formulated. This context, which we will call a **frame**, describes the prizes which are relevant and accessible, the structural aggregation of the prizes, our player and, if deemed relevant, the opponent.

4.1.1.1 Frame Definition

Suppose we fix a planning horizon and restrict the regional focus on prizes considered. The basic decision problem is to determine how the game is likely to evolve over the duration of this planning horizon. The players may also have directives to follow or implement, or surrogate objectives to satisfice. These four attributes are sufficient to define a frame.

Horizon. The planning horizon represents how far into the future to plan and can be *fixed, variable, rolling or dynamic.*

Scope. The scope describes the subset of prizes which may be considered, how the visibility of the players may be limited, how the prizes may be aggregated, the structure of the aggregation (e.g. into clusters) and any stratification of the prizes.

Directives. A directive is a constraint placed upon the decision problem. These may include restricted displacement of the player, target locations, time windows on harvesting within favourable regions, maintaining an avoidance distance from the opponent or some lower bound on harvesting rate.

Surrogates. Surrogate objectives describe the goal or purpose of the decision problem as these may be different from the overall objective of the CPCP. These may include harvesting rate, time to reach a target location, or milestones to be attempted.

4.1.1.2 Frame Components

Kahneman, Slovic and Tversky [122] and Tversky and Kahneman [201] describe three heuristics that are employed in making judgements under uncertainty:

Representativeness. Usually employed when people are asked to judge the probability that an object or event A belongs to a class or process B.

Availability of instances or scenarios. Often employed when people are asked to assess the frequency of a class or the plausibility of a particular development.

Adjustment from an anchor. Usually employed in numerical prediction when a relevant value is available.

Séguin, Potvin, Gendreau, Crainic and Marcotte [192] outline the functional components of a real-time decision process.

(i). Information management and data fusion.

(ii). Situation assessment and evaluation of alternatives.

(iii). Decision: a decision does not necessarily result in an immediate action, rather, the action is incorporated into a plan that extends over a rolling time period known as a planning horizon.

These functional components are implemented as two processes which operate within the context of a frame.

Planner. The planner generates and evaluates at least one scenario, but need not involve contingent planning. This constitutes components (i)–(ii) above. For example, NEAREST NEIGHBOUR could be used as a single scenario planner, or selecting random prizes could be used as a multiple scenario planner.

Selector. The selector decides upon a scenario from those generated by the planner and implements the initial actions of the plan. This constitutes (iii) above. Even if only one scenario is proposed by the planner, the selector may be non-trivial as it must determine how much of the initial plan to implement.

4.1.2 Hierarchical Strategic Planning Architecture

The **Strategic Planning Architecture** (SPA) is a hierarchy of frames. The *global-frame* is the initial frame considered and has infinite planning horizon, full scope (i.e. all prizes are available), no directives and no surrogates. The global-frame selector not only selects a strategic plan but the implementation of that plan is itself a frame spawned by the selector. Hence the decision made by a selector process is another frame.

At each frame the selector defines a decision problem whose solution is a sub-frame of a finer level. The coarseness of the sub-frame could be different depending upon which actual decision was made. In general, the scope is smaller (i.e. fewer available prizes), the planning horizon is shorter, and more specific directives and surrogates are applied.

4.1.2.1 Hierarchical Nature

Hierarchical structure is important in *Systems Thinking* (as advocated by Checkland [37]) together with processes of communication and control. It provides for the decoupling of decisions at different planning horizons into strategic, tactical and operational planning. Moreover, the decisions telescope since we make progressively finer decisions, at decreasing levels of aggregation, until an actual *step* is determined.

4.1.2.2 Dynamic Composition of Frames

The selection of frames also depends upon the stage of the game and the structure of the prizes remaining. At the beginning of the game we need to set up what to do in the long-term carefully but with very imprecise information and uncertain expectations. However, at the end of the game, we may need to put more effort into fewer levels of decision making, perhaps due to more optimality-centred methods.

The decision processes exist concurrently and dynamically adjust to the refinement, redefinition or existence of both the parent and child frame. However, near-term frames are likely to be updated more often but with less intensive computation, while long-term frames may employ fewer more intensive computations. Thus the distribution of computational effort between processes depends upon what is more effective at any configuration of frames.

4.1.2.3 Natural Structure

The aim of aggregation is to isolate a natural strategic structure, local to a particular frame, suitable and sufficient to determine how the game may evolve over the planning horizon of that frame. Generically, the number of frames used, their relative planning horizons and planning scopes are dynamic and should adapt to the natural structure of the evolving game state. However, this depends very much on what structure (or lack thereof) there is to exploit. Hence it may be necessary to impose some structure when it is necessary to decompose the decision problem but no natural structure is apparent. We wish to maintain a balance between planning horizons but know that there is always a tradeoff of scope and detail concomitant with a computational budget. Although the framework is sufficiently flexible to make frame choice and goal-within-frame choice independent, there is a synergy between the choice of frames and the methods used within the adjoining frames. A method at one frame attempts at exploit the structure of the objects within that frame and, hence, any subframe will be based around the object selected. In addition, there is the strong coupling between frames, in that decisions in one frame implement and refine the decisions of the previous frames and model the focussed prize region in more detail.

4.2 Cognizance of Opponent and Response Monitoring

Séguin, Potvin, Gendreau, Crainic and Marcotte [192] distinguish between three types of planning in a real-time decision process.

Reactive planning. Short-term, local effect, small changes in global state.

Incremental planning. Modification, global state does not depart too much from expected state at the time the plan was first devised.

Deliberative planning. Mandatory revision when state of the world departs significantly from its predicated state.

While contingent planning attempts to account for the future actions of the opponent, observation of the opponent is necessary to determine whether reactive, incremental or deliberative planning is required at each frame within the SPA. **Response Monitoring** is a cycle of forecasting the likely targets of the opponent, observation of whether the opponent's behaviour is consistent with our prediction, plan refinement and, when necessary, triggering of deliberative planning. Each player must decide how much cognizance they will take of their opponent's movements and opportunities.

4.2.1 Cognizance of Opponent in a Frame

Consider a particular frame of the SPA. If the opponent is outside the frame's scope then the frame may be classified as *independent* and it is responsible for monitoring the opponent's opportunities and actual movements on an appropriate scale. Otherwise, the frame may be classified as *dependent* and its parent frame is responsible for monitoring the opponent on a scale appropriate to the parent frame; the opponent can be safely ignored within this frame (except indirectly

through surrogate objectives). Loosely, a dependent frame remains dynamic but an independent frame becomes like a static subproblem. In defining and directing a child frame, the decision process must consider the dependence (or independence) of a potential child frame as part of the natural structure of the parent frame's planning scope. There is at least one dependent frame, the global frame, and a child frame of an independent frame is, by definition, also independent.

The approach to the presence of the opponent within the scope of a frame of the SPA may include any of the following features.

Tactical Conflict. Direct competition over the prizes within scope.

Avoidance of Opponent. Recognition that the opponent may have a significant effect in the current scope but the surrogates or directives imply that the best approach is to harvest efficiently whilst staying out of tactical conflict.

Ignorance of Opponent. Trying to focus on efficient harvesting, regardless of the opponent's impact upon the prizes available for harvesting.

Independence of Opponent. The opponent is sufficiently out of scope so that efficiency in satisficing the surrogate objectives is paramount.

4.2.2 Active and Passive Monitoring

There are two distinct roles for monitoring that we can identify:

Passive monitoring. Determining what the opponent is doing or is likely to be able to do and, if necessary, determining effective countering steps.

Active monitoring. Studying the behaviour of the opponent and using this in such a way as to adapt the countering moves to take advantage of said behaviour. Essentially this is equivalent to *learning* from observed actions of the opponent.

An active monitor may actually modify (via parameters or spawning processes) the information engine at that frame. It may also classify the opponent's behaviour as one of several player types: paranoid, conservative, calculated risk-taker, desperado or stalker. Initially an active monitor would classify the problem instance as one of a number of problem classes and apply the most successful or robust strategy from a stored library for that class of problem. Estimation of current progress could then be used to adapt or fine-tune the strategy or to match the strategy component to the phase of game being played.

Classifier Systems are massively parallel, message-passing, rule-based systems that learn through competing hypotheses (expressed as rules), credit assignment and rule discovery (Booker, Goldberg and Holland [24], Holland [103] and Holland, Holyoak, Nisbett and Thagard [104]). Rule discovery employs the **Genetic Algorithm** (GA), a meta-heuristic also useful for combinatorial optimisation problems (Holland [102] and Goldberg [84]). Classifier systems are designed to operate in environments exhibiting perpetually novel events, continual (often real-time) requirements for action, implicitly or inexactly defined goals and sparse payoff obtainable only through long action sequences. Since competitive routing problems often exhibit these characteristics, classifier systems may be a useful component of an active learning monitor given suitable strategic and

tactical building blocks. An active monitor learning module is integral to a full-blown strategic framework but is beyond the scope of this thesis since it more appropriately belongs to the realms of Artificial Intelligence (AI).

4.2.3 Dynamic Monitoring System

A **Dynamic Monitoring System** (DMS) is a particular implementation of the SPA which applies an *scenario engine* as its *planner* module and a *monitor* as its *selector* module.

Scenario Engine. Recommends what action should be taken for each possible target set of the opponent, proposes and evaluates a number of scenarios using a contingent evaluator.

Monitor. Determines which target set the opponent is currently targeting or likely to target and spawns the sub-frame corresponding to that scenario.

Figure 4.1 illustrates the relationships between monitors and frames in a DMS. The GLOBAL-MONITOR operates within the global-frame, selects a scenario generated by a global scenario engine and spawns a corresponding sub-frame and SUB-MONITOR. In turn, that SUB-MONITOR operates within the sub-frame, selects a scenario generated by an appropriate scenario engine and spawns another, simpler sub-frame and SUB-MONITOR. The symbol '⋮' indicates that the number of frames in the DMS is dynamic with the successive choices made by each monitor. Finally, there must exist a STEP-MONITOR whose decision is a *step*. The corresponding frame is called the step-frame.

The composition of the strategic framework hierarchy is dynamic. While deliberative planning may be required at one frame, this may trigger only incremental or reactive planning at its parent frame. Dynamism implies that all frames exist concurrently. Hence a frame may become invalid when the monitor of its parent frames determines that the frame's definition is no longer appropriate. The depth of the hierarchy may change depending upon the global state of the game. In practice the frames exist concurrently and the horizon, scope, directives and surrogates are updated via message passing between the monitoring processes. However a monitor may spawn a completely new frame and monitor.

Coda

▼ Summary

In this chapter we have designed a framework for considering *aggregation* of prizes in a strategy. We have designed a computational, hierarchical, strategic framework for the CPCP. "Solving" the problem requires a balancing of computational requirements against effectively coordinating strategy at a number of different planning frames. Having proposed the RM-CRP and CPCP in Chapter 2 and found the DVRSP solution paradigm wanting as a solution paradigm for the CPCP/CRP in Chapter 3, the current chapter has outlined a computational, hierarchical, strategic solution architecture. We have highlighted the difference between making policy (strategy), implementing it at a lower level (tactics) and efficient movement between significant locations (operations).

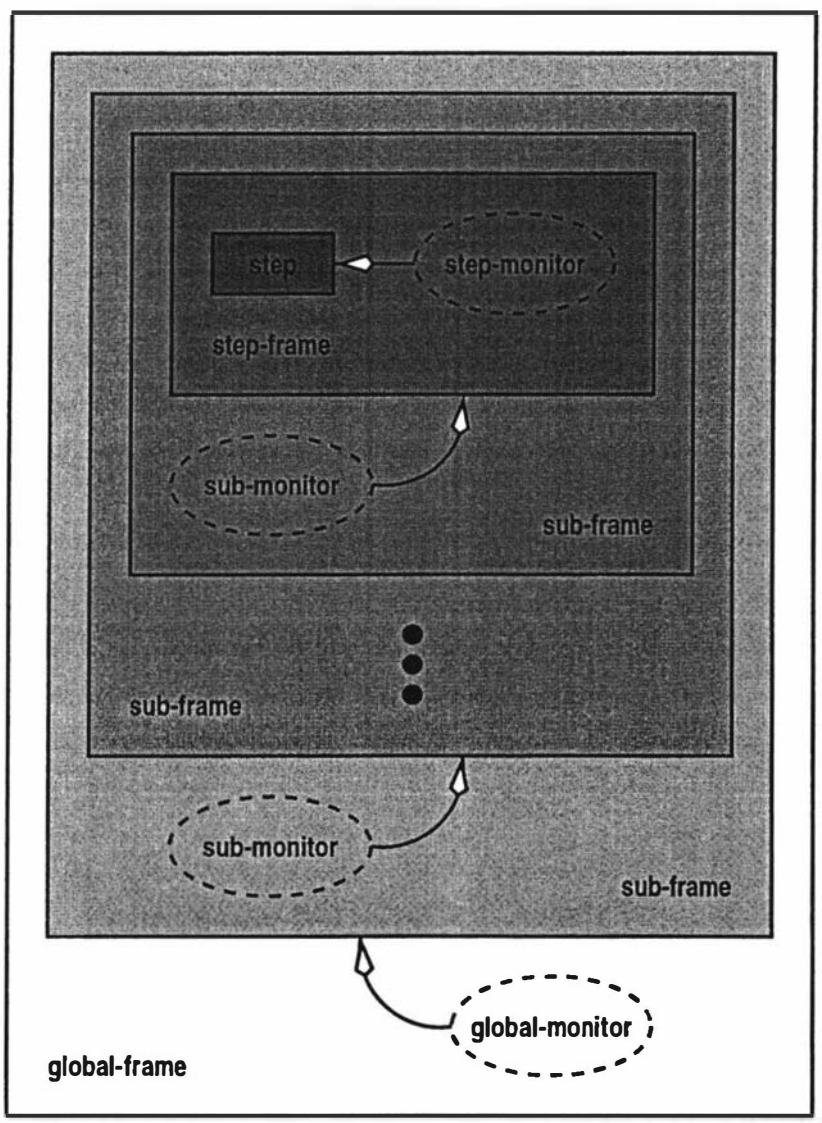


Figure 4.1: Generic Dynamic Monitoring System

▼ Link

In Part II we develop a series of implementations of the SPA/DMS for different levels of aggregation of the prize set. The progression in Chapters 5–8 is from tiny problems involving two or three prizes through to large problems involving hundreds of prizes.

Part II

Strategies

Overview of Part II
Strategies

Part II is concerned with the design and analysis of strategies for the CPCP. Chapters 5–7 incrementally develop an implementation of the SPA/DMS for tiny (two prize), small (few prizes) and medium (exhibiting some natural cluster-like structure) problems. Chapter 8 develops an alternative implementation of the SPA/DMS for large problems in which there are very many prizes and no assumptions made about their locational structure.

CHAPTER 5

Optimal and Heuristic Analysis of Tiny Problems

It's not the size of the dog in the fight that counts, but rather the size of the fight in the dog.

— DWIGHT EISENHOWER

Since the one prize problem is trivial—heading directly towards the prize is indubitably optimal—the smallest problem of interest consists of two prizes. Yet even this deceptively simple problem can be strategically complex.

5.0 Introduction

We begin with a mathematical analysis of contingency planning by investigating tactical cases of the two prize problem (Section 5.1) and the window feasibility subproblem (Section 5.2). These two building blocks are fundamental to almost all the strategies proposed in the remainder of this thesis and form the *step-frame* at the base of the SPA/DMS. We then design strategies for two prize problems (Section 5.3), and two prize problems with a finite deadline (Section 5.4), which compete against one another in a computational tournament (Section 5.5) to determine what result each player can expect from a particular encounter and which strategies can effectively deliver such a result. Finally, we consider the design of a *scenario engine* for the next non-trivial problem, i.e., the three prize problem, by integrating two prize subproblems and window

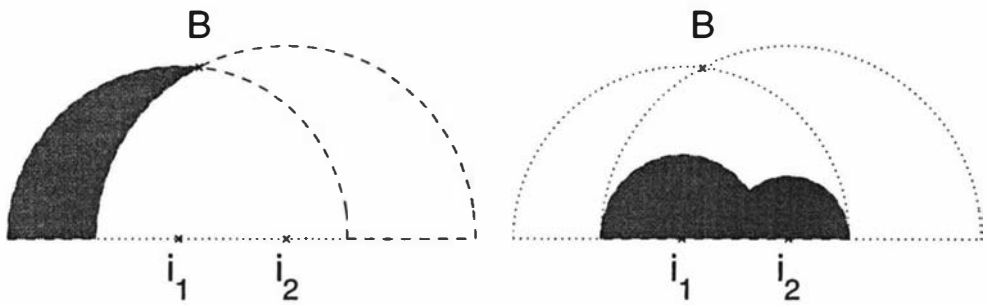


Figure 5.1: Static Cases of the Two Prize Problem

feasibility subproblems (Section 5.6).

5.1 Two Prize Problem

The **two prize problem** is an instance of the CPCP with exactly two prizes $\{i_1, i_2\}$.[1] We assume that $\lambda = \infty$ and consequently we have a *two-person constant-sum game*. The case where $\lambda < \infty$ leads to a *two-person non-cooperative general-sum game* and is discussed in Section 5.4. Without loss of generality, suppose the players are in the same half plane relative to the prizes, $d_{Ai_1} \leq d_{Bi_1}$ and that if $d_{Ai_1} = d_{Bi_1}$ then $d_{Ai_2} \leq d_{Bi_2}$. We also suppose that $|d_{Ai_1} - d_{Ai_2}| \neq d_{i_1 i_2}$, i.e., player $\mathcal{A}$ is neither collinear with nor between the prizes.[2]

We use the notation "$\mathcal{A}$—$\circ$ i" to indicate that player $\mathcal{A}$ takes one step, of size Δ, towards prize i and "$\mathcal{A}$—$\circ$ X" to indicate that player $\mathcal{A}$ move to location X such that $d_{AX} \leq \Delta$. The notation "$\mathcal{A} \rhd i_1$" stands for the hypothesis that player $\mathcal{A}$ is committed to travel all the way to prize i_1.

A move $\mathcal{A} \multimap X$ is **critical** if otherwise there is no possibility that player $\mathcal{A}$ can successfully achieve its current guaranteed value. Hence the current decision is a "knife-edge" decision.

5.1.1 Static Cases

The following cases (i)–(iv) are the **static cases** for which there exists a saddle point equilibrium in which neither player requires a contingent strategy. We say that the static cases are **guarantee determined**.

Case (i): $d_{Ai_1} < d_{Bi_1}$ and $d_{Ai_2} > d_{Bi_2}$. Figure 5.1 shows the possible locations of player $\mathcal{A}$ satisfying case (i) given the locations of player $\mathcal{B}$ and the two prizes. Player $\mathcal{A}$ can guarantee value v_{i_1} since prize i_1 is *guaranteed* to player $\mathcal{A}$, and player $\mathcal{B}$ can guarantee value v_{i_2} since prize i_2 is *guaranteed* to player $\mathcal{B}$. Hence $\mathcal{A}$—$\circ$ i_1 and $\mathcal{B}$—$\circ$ i_2 is a saddle point equilibrium.

[1] We use $\{i_1, i_2\}$ rather than $\{1, 2\}$ as the prize labels since later we apply the two prize problem relative to a larger prize set.

[2] If $|d_{Ai_1} - d_{Ai_2}| = d_{i_1 i_2}$ then the optimal strategy for player $\mathcal{A}$ is NEAREST NEIGHBOUR.

Case (ii): $d_{Ai_1} = d_{Bi_1}$ and $d_{Ai_2} = d_{Bi_2}$. Without loss of generality we may assume that $v_{i_1} \geq v_{i_2}$.

Suppose $v_{i_1} > v_{i_2}$. If $\mathcal{A} \rhd i_1$ then the resulting outcome is at least $\frac{1}{2}(v_{i_1} + v_{i_2})$ to player $\mathcal{A}$. However, $\mathcal{A} \multimap i_1$ is *critical* for player $\mathcal{A}$ since otherwise if $\mathcal{B} \multimap i_1$ then the resulting outcome is $v_{i_2} < \frac{1}{2}(v_{i_1} + v_{i_2})$. Similarly $\mathcal{B} \multimap i_1$ is *critical* for player $\mathcal{B}$. Hence $\mathcal{A} \multimap i_1$ and $\mathcal{B} \multimap i_1$ is a saddle point equilibrium in which the players share the sequence $i_1 \rightarrow i_2$, for a reward of $\frac{1}{2}(v_{i_1} + v_{i_2})$ to each player.

Suppose $v_{i_1} = v_{i_2}$. Any pair of moves which target a prize is a saddle point equilibrium in which the players either share the sequence $i_1 \rightarrow i_2$ or claim one prize each, for a reward of $v_{i_1} = v_{i_2} = \frac{1}{2}(v_{i_1} + v_{i_2})$ to each player.

In both cases a saddle point equilibrium is $\mathcal{A} \multimap \operatorname{argmax}\{v_{i_1}, v_{i_2}\}$ and $\mathcal{B} \multimap \operatorname{argmax}\{v_{i_1}, v_{i_2}\}$, for a reward of $\frac{1}{2}(v_{i_1} + v_{i_2})$ to each player.

Case (iii): $d_{A1} < d_{B1}$ and $d_{A2} = d_{B2}$. Suppose $v_{i_1} \geq v_{i_2}$. If $\mathcal{A} \rhd i_1$ then the resulting outcome is at least v_{i_1} to player $\mathcal{A}$. If $\mathcal{B} \rhd i_2$ then the resulting outcome is at least v_{i_2} to player $\mathcal{B}$. Hence $\mathcal{A} \multimap i_1$ and $\mathcal{B} \multimap i_2$ is a saddle point equilibrium in which the player $\mathcal{A}$ eventually claims prize i_1 for reward v_{i_1} and player $\mathcal{B}$ eventually claims prize i_2 for reward v_{i_2}.

Suppose $v_{i_1} < v_{i_2}$. If $\mathcal{A} \rhd i_2$ then the resulting outcome is at least $\frac{1}{2}(v_{i_1} + v_{i_2})$ to player $\mathcal{A}$. However, $\mathcal{A} \multimap i_2$ is *critical* for player $\mathcal{A}$ since otherwise if $\mathcal{B} \multimap i_2$ then the resulting outcome is $v_{i_1} < \frac{1}{2}(v_{i_1} + v_{i_2})$. Similarly $\mathcal{B} \multimap i_2$ is *critical* for player $\mathcal{B}$. Hence $\mathcal{A} \multimap i_1$ and $\mathcal{B} \multimap i_1$ is a Nash equilibrium in which the players share the sequence $i_1 \rightarrow i_2$, for a reward of $\frac{1}{2}(v_{i_1} + v_{i_2})$ to each player.

In both cases a saddle point equilibrium is $\mathcal{A} \multimap \operatorname{argmax}\{v_1, v_2\}$ and $\mathcal{B} \multimap i_2$, for a reward of $\max\{\frac{1}{2}(v_1 + v_2), v_1\}$ to player $\mathcal{A}$ and $\min\{\frac{1}{2}(v_1 + v_2), v_2\}$ to player $\mathcal{B}$.

Case (iv): $d_{Ai_1} + d_{i_1 i_2} < d_{Bi_2}$ or $d_{Ai_2} + d_{i_1 i_2} < d_{Bi_1}$. Figure 5.1 also shows the possible locations of player $\mathcal{A}$ satisfying case (iv) given the locations of player $\mathcal{B}$ and the two prizes. Player $\mathcal{A}$ can guarantee value $v_{i_1} + v_{i_2}$ since either $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ or $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is *guaranteed.*

5.1.2 The Two Prize Theorem

The cases which are not static that remain are characterised by constraints (5.1)–(5.4) and are called the *dynamic*, cases since there is no optimal strategy for either player that does not involve playing *contingently* upon the observed movements of the opponent.

$$d_{Ai_1} < d_{Bi_1} \tag{5.1}$$
$$d_{Ai_2} < d_{Bi_2} \tag{5.2}$$
$$d_{Ai_1} + d_{i_1 i_2} \geq d_{Bi_2} \tag{5.3}$$
$$d_{Ai_2} + d_{i_1 i_2} \geq d_{Bi_1} \tag{5.4}$$

Constraints (5.1)–(5.2) are known as the **prize constraints** and constraints (5.3)–(5.4) as the **pair constraints**. Player $\mathcal{A}$ is known as the "$\mathcal{A}$-head" player and player $\mathcal{B}$ is known as the "$\mathcal{B}$-hind" player. Figure 5.2 shows the possible locations of player $\mathcal{A}$ satisfying constraints (5.1)–(5.4) given the locations of player $\mathcal{B}$ and the two prizes.

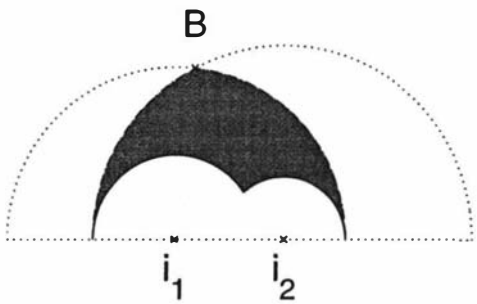


Figure 5.2: Dynamic Case of the Two Prize Problem

Theorem 5.1.1 (Two Prize Theorem)
Suppose the constraints (5.3)–(5.4) hold at some time t_0 with player $\mathcal{A}$ located at A and player $\mathcal{B}$ located at B. Player $\mathcal{A}$ declares that it will arrive at location X at time $t_X \geq t_0 + d_{AX}$. Then $\exists$ a location Y such that

$$d_{AX} \geq d_{BY} \tag{5.5}$$
$$d_{Xi_1} + d_{i_1 i_2} \geq d_{Yi_2} \tag{5.6}$$
$$d_{Xi_2} + d_{i_1 i_2} \geq d_{Yi_1} \tag{5.7}$$

and, hence, if player $\mathcal{B}$ moves directly to location Y, then the pair constraints also hold at time t_X. In particular, if X is the location of prize i_1, then $\exists$ a corresponding Y located on the line segment between B and prize i_2, and if X is the location of prize i_2, then $\exists$ a corresponding Y located on the line segment between B and prize i_1.

Proof:

If $d_{Bi_1} \leq d_{Xi_2} + d_{i_1 i_2}$ or $d_{Bi_2} \leq d_{AX}$ then move player $\mathcal{B}$ a distance $\max\{d_{AX}, d_{Bi_2}\}$ directly towards prize i_2. Alternatively, if $d_{Bi_2} \leq d_{Xi_1} + d_{i_1 i_2}$ of $d_{Bi_1} \leq d_{AX}$ then move player $\mathcal{B}$ a distance $\max\{d_{AX}, d_{Bi_1}\}$ directly towards prize i_1. Henceforth, suppose $d_{Bi_1} > d_{AX}$, $d_{Bi_2} > d_{AX}$, $d_{Bi_1} > d_{Xi_2} + d_{i_1 i_2}$ and $d_{Bi_2} > d_{Xi_1} + d_{i_1 i_2}$.

From the perspective of location A, let ψ_{Ai_1} be the bearing to prize i_1 and let ψ_{Ai_2} be the bearing to prize i_2. From the perspective of location B, let ψ_{Bi_1} be the bearing to prize i_1 and let ψ_{Bi_2} be the bearing to prize i_2. As in Figure 5.3, let θ_A be the acute angle between ψ_{Ai_1} and ψ_{Ai_2}, and let θ_B be the acute angle between ψ_{Bi_1} and ψ_{Bi_2}.

Without loss of generality, we may assume that the location X is on or inside the triangle $\triangle Ai_1 i_2$, since this can only make constraints (5.6)–(5.7) more restrictive. From the perspective of location X, let ψ_{Xi_1} be the bearing to prize i_1. Let θ_{Xi_1} be the acute angle between ψ_{Xi_1} and ψ_{Ai_1}.

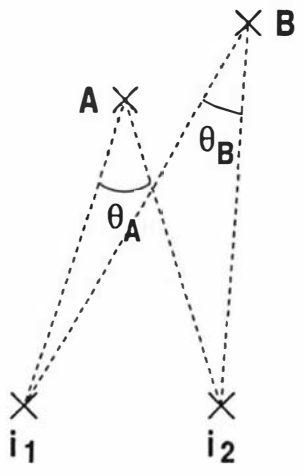


Figure 5.3: Angles θ_A and θ_B.

We now construct the required location Y in each of the following cases.

(I). Suppose $\theta_B \leq \theta_A$. Let ψ_Y be the bearing which is an angle $\frac{\theta_B}{\theta_A}\theta_{Xi_1}$ from ψ_{Bi_2} towards ψ_{Bi_1} via the acute angle between the ψ_{Bi_2} and ψ_{Bi_1}.

- If the distance along bearing ψ_Y from B to the line through the prizes is no greater than d_{AX}, then let Y be the intersection of bearing ψ_Y with this line through the prizes. Then Y satisfies constraints (5.5)–(5.7)
- Move player $\mathcal{B}$ to a location Y, along bearing ψ_Y a distance d_{AX} (but no further than the line between the prizes). Player $\mathcal{B}$ travels at least as close in deviation of bearing from the bearing of each prize for the same distance as does player $\mathcal{A}$ from a location farther from both prizes. Hence Y satisfies constraints (5.5)–(5.7).

(II). Consider the location H such that $d_{Hi_1} = d_{Ai_2} + d_{i_1i_2}$ and $d_{Hi_2} = d_{Ai_1} + d_{i_1i_2}$. Applying the *Cosine Law* to $\triangle Ai_1i_2$ and $\triangle Hi_1i_2$:

$$\begin{aligned} d_{12}^2 &= d_{A1}^2 + d_{A2}^2 - 2d_{A1}d_{A2}\cos\theta_A \qquad (5.8)\\ d_{12}^2 &= d_{H1}^2 + d_{H2}^2 - 2d_{H1}d_{H2}\cos\theta_H \\ &= (d_{A2} + d_{12})^2 + (d_{A1} + d_{12})^2 - 2(d_{A2} + d_{12})(d_{A1} + d_{12})\cos\theta_H \\ &= d_{A1}^2 + d_{A2}^2 - 2d_{A1}d_{A2}\cos\theta_H + \\ &\quad 2d_{12}(d_{A1} + d_{A2} + d_{12})(1 - \cos\theta_H) \qquad (5.9) \end{aligned}$$

Equating (5.8) and (5.9)

$$\begin{aligned} d_{A1}d_{A2}(\cos\theta_H - \cos\theta_A) &= d_{12}(d_{A1} + d_{A2} + d_{12})(1 - \cos\theta_H) \\ &> 0 \quad (\text{since } \theta_H > 0) \end{aligned}$$

Then $\cos\theta_H > \cos\theta_A$ and $\theta_H < \theta_A$ since $\theta_A, \theta_H \leq \pi$. Hence the location H satisfies the requirements for location B in case (I) above. Let $Y = Y_H$ be the point that would be constructed from case (I) corresponding to $B = H$, i.e., where player $\mathcal{B}$ is located at H.

(III). Suppose $\theta_B > \theta_A$. Consider the locations H and Y_H from case (II).

- Suppose $\angle Bi_1i_2 > \angle Y_Hi_1i_2$ and $\angle Bi_2i_1 > \angle Y_Hi_2i_1$. Then $d_{Bi_1} > d_{Y_Hi_1}$ and $d_{Bi_2} > d_{Y_Hi_2}$. Consider the region R in the half-plane bounded by

  C_1: the arc radius $r_1 = d_{Ai_2} + d_{i_1i_2}$ centred at prize i_1

  C_2: the arc radius $r_2 = d_{Ai_1} + d_{i_1i_2}$ centred at prize i_2

  L_1: the line through prize i_1 and location Y_H

  L_2: the line through prize i_2 and location Y_H

  Let P_1 be the intersection of C_1 and L_1, and let P_2 be the intersection of C_2 and L_2. Since Y_H is on or inside the triangle $\triangle Hi_1i_2$, R is a convex region with extreme points $\{H, Y_H, P_1, P_2\}$. Hence the point in R which is farthest from Y_H is one of $\{H, P_1, P_2\}$. We know $d_{HY_H} \leq d_{AX}$, by definition of Y_H. Since L_1 is a normal to C_1, we can consider the circle C_3 centred at Y_H of radius $d_{P_1Y_H} \leq r_1$. Then H cannot lie inside C_3 and, hence, $d_{P_1Y_H} \leq d_{HY_H}$. Similarly, $d_{P_2Y_H} \leq d_{HY_H}$. Hence every point in R is at most a distance d_{AX} from Y_H, and the location $Y = Y_H$ satisfies constraints (5.5)–(5.7).

- Suppose $\angle Bi_1i_2 \leq \angle Y_Hi_1i_2$. Since $d_{Bi_1} > d_{Xi_2} + d_{i_1i_2}$, then $d_{Bi_1} > d_{Y_Hi_1}$. Move player $\mathcal{B}$ to a location Y, a distance $d_{Bi_1} - d_{Y_Hi_1}$ directly towards prize i_1. Since $d_{Yi_1} = d_{Y_Hi_1}$, constraint (5.7) is satisfied. Also $d_{Bi_1} - d_{Y_Hi_1} \leq d_{Hi_1} - d_{Y_Hi_1} = d_{P_1Y_H} \leq d_{AX}$, so that constraint (5.5) is satisfied. Since $\angle Yi_1i_2 = \angle Bi_1i_2 \leq \angle Y_Hi_1i_2$ and $d_{Yi_1} = d_{Y_Hi_1}$, $d_{Yi_2} \leq d_{Y_Hi_2}$ and hence Y also satisfies constraint (5.6).

- Similarly, if $\angle Bi_2i_1 \leq \angle Y_Hi_2i_1$, then player $\mathcal{B}$ can move to a location Y, a distance $d_{Bi_2} - d_{Y_Hi_2}$ directly towards prize i_2, and constraints (5.5)–(5.7) are satisfied.

Hence we have shown that $\exists$ a location Y satisfying constraints (5.5)–(5.7).

Finally, constraint (5.3) states exactly that, if X is the location of prize i_1, then $\exists$ such a Y on the line segment between B and prize i_2, and constraint (5.4) states exactly that, if X is the location of prize i_2, then $\exists$ such a Y on the line segment between B and prize i_2. ■

■ Importance of the Two Prize Theorem

The Two Prize Theorem shows that if player $\mathcal{B}$ were to know exactly to which location player $\mathcal{A}$ is to move, then player $\mathcal{B}$ can move so as to maintain the pair constraints (5.3)–(5.4) intact. However, if player $\mathcal{A}$ were to know exactly to which location player $\mathcal{B}$ is to move, then it is **not true** that player $\mathcal{A}$ can move so as to maintain the prize constraints (5.3)–(5.4) intact (see Section 5.3.8).

5.1.3 Dynamic Cases

The following assumption serves to distinguish between the preferences of each player.

Assumption 5.1.2

Player $\mathcal{A}$ will not (wherever possible) move so as to sacrifice a guaranteed value for some lower guaranteed value. Player $\mathcal{B}$ will not (wherever possible) move so that player $\mathcal{A}$ can improve its guaranteed value. □

This effectively defines **rationality** for the players in a Two Prize Problem.

Case (v): $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} > d_{Bi_1}$. Firstly, $\mathcal{B}\multimap i_2$ is *critical* to player $\mathcal{B}$ since otherwise, if $\mathcal{A}\multimap i_1$, $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is guaranteed. Hence we may assume that $\mathcal{B}\multimap i_2$. Also, player $\mathcal{A}$ can guarantee value $v_{i_1} + \frac{1}{2}v_{i_2}$ from $\mathcal{A} \rhd i_1$ and can guarantee value v_{i_2} from $\mathcal{A} \rhd i_2$.

Suppose $v_{i_1} > \frac{1}{2}v_{i_2}$. Then $\mathcal{A}\multimap i_1$ is *critical* to player $\mathcal{A}$ since otherwise, if $\mathcal{B}\multimap i_2$, then only value $v_{i_1} < v_{i_1} + \frac{1}{2}v_{i_2}$ is guaranteed to player $\mathcal{A}$. Hence $\mathcal{A}\multimap i_1$ and $\mathcal{B}\multimap i_2$ is a saddle point equilibrium.

Suppose $v_{i_1} \leq \frac{1}{2}v_{i_2}$. Then $\mathcal{A}\multimap i_1$ is not critical to player $\mathcal{A}$.

- If $\mathcal{A}\multimap i_1$ then, by Theorem 5.1.1, $\exists$ a location Y such that $d_{BY} \leq \Delta$ and the pair constraints continue to hold if $\mathcal{B}\multimap Y$. Since $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$, this move must be equivalent to $\mathcal{B}\multimap i_2$ and hence constraint (5.3) will continue to hold with equality.

- If $d_{Ai_1} > \Delta$, then $\mathcal{A}\multimap i_2$ is not critical to player $\mathcal{A}$, since $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2} > d_{Ai_2}$ implies that player $\mathcal{A}$ is not located on the line through the prizes and, hence, at the next step, we must have $d_{Ai_2} < d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$. However, if $d_{Ai_1} \leq \Delta$ and $v_{i_1} \leq \frac{1}{2}v_{i_2}$, then it is *critical* that player $\mathcal{A}$ does *not* play $\mathcal{A}\multimap i_1$ since, otherwise, if $\mathcal{B}\multimap i_2$, only value $v_{i_1} + \frac{1}{2}v_{i_2} < v_{i_2}$ is guaranteed to player $\mathcal{A}$.

- We propose that player $\mathcal{A}$ adopt the strategy of $\mathcal{A}\multimap i_2$ since, eventually, player $\mathcal{A}$ must play $\mathcal{A}\multimap i_2$, there is no guaranteed value lost, and there is little to be gained for player $\mathcal{A}$ by both players consistently targeting opposite prizes.

Case (vi): $d_{Ai_1} + d_{i_1 i_2} > d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} = d_{Bi_1}$. Similar to case (v).

Case (vii): $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} = d_{Bi_1}$. Firstly, $\mathcal{B}\multimap i_1$ is *critical* to player $\mathcal{B}$ since otherwise, if $\mathcal{A}\multimap i_2$, $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is guaranteed. Also $\mathcal{B}\multimap i_2$ is *critical* to player $\mathcal{B}$ since otherwise, if $\mathcal{A}\multimap i_1$, $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is guaranteed. Hence player $\mathcal{B}$ cannot move so as to *ensure* that both pair constraints continue to hold.

Player $\mathcal{A}$ can guarantee value $v_{i_1} + \frac{1}{2}v_{i_2}$ from $\mathcal{A} \rhd i_1$ and can guarantee value $v_{i_2} + \frac{1}{2}v_{i_1}$ from $\mathcal{A} \rhd i_2$.

- Consider $\mathcal{A}\multimap i_1$. Then, by Theorem 5.1.1, $\exists$ a location Y such that $d_{BY} \leq \Delta$ and the pair constraints continue to hold if $\mathcal{B}\multimap Y$. Since $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$, this move must be equivalent to $\mathcal{B}\multimap i_2$ and, hence, constraint (5.3) will continue to hold with equality.

- Consider $\mathcal{A}\multimap i_2$. Then, by Theorem 5.1.1 $\exists$ a location Y such that $d_{BY} \leq \Delta$ and the pair constraints continue to hold if $\mathcal{B}\multimap Y$. Since $d_{Ai_2} + d_{i_1 i_2} = d_{Bi_1}$, this move must be equivalent to $\mathcal{B}\multimap i_1$ and, hence, constraint (5.4) will continue to hold with equality.

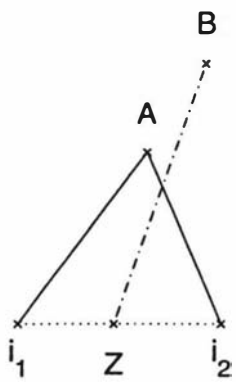


Figure 5.4: Two Prize Median Location

Hence player $\mathcal{A}$ cannot move so as to *ensure* that at least one of the pair constraints does not continue to hold.

Suppose $v_{i_1} \neq v_{i_2}$. Then $\mathcal{A}$—$\circ$ argmax$\{v_{i_1}, v_{i_2}\}$ is *critical* to player $\mathcal{A}$ since, otherwise, if $\mathcal{B}$—$\circ$ argmax$\{v_{i_1}, v_{i_2}\}$, only value min$\{v_{i_1} + \frac{1}{2}v_{i_2}, v_{i_2} + \frac{1}{2}v_{i_2}\}$ is guaranteed to player $\mathcal{A}$. Hence $\mathcal{A}$—$\circ$ argmax$\{v_{i_1}, v_{i_2}\}$ and $\mathcal{B}$—$\circ$ argmin$\{v_{i_1}, v_{i_2}\}$ is a saddle point equilibrium.

Suppose $v_{i_1} = v_{i_2}$. Then the mixed strategy in which player $\mathcal{A}$ selects $\mathcal{A}$—$\circ$ i_1 with probability $\frac{1}{2}$ and $\mathcal{A}$—$\circ$ i_2 with probability $\frac{1}{2}$, and the mixed strategy in which player $\mathcal{B}$ selects $\mathcal{B}$—$\circ$ i_1 with probability $\frac{1}{2}$ and $\mathcal{B}$—$\circ$ i_2 with probability $\frac{1}{2}$, forms a Nash equilibrium.

Case (viii): $d_{Ai_1} + d_{i_1 i_2} > d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} > d_{Bi_1}$. This is the most difficult case since there is no obvious optimal strategy for either player, yet there is some slack in each of constraints (5.1)–(5.4). In Section 5.3 we will return to this case to propose various strategies for it.

5.1.4 Median Feasibility

Suppose constraints (5.1)–(5.4) hold with strict inequality. We wish to determine if player $\mathcal{B}$ can move to some location such that one of the prize constraints definitely does not hold, thus leaving player $\mathcal{A}$ to decide between the prizes. If player $\mathcal{B}$ can reach the line between the prizes with the pair constraints still holding, then there is no location for player $\mathcal{A}$ which is closer to both prizes. Hence player $\mathcal{B}$ can guarantee a prize. This is illustrated in Figure 5.4.

Definition 5.1.3

$\mathcal{B} \rhd \{i_1, i_2\}$ is **median-feasible** if $\exists$ a location Z, on the line through the two prizes, such that constraints (5.10)–(5.11) hold.

$$d_{BZ} + d_{Zi_2} \leq d_{Ai_1} + d_{i_1 i_2} \quad (5.10)$$

$$d_{BZ} + d_{Zi_1} \leq d_{Ai_2} + d_{i_1 i_2} \quad (5.11)$$

□

Such a location Z is called a **median location**.

To determine if $\mathcal{B} \rhd \{i_1, i_2\}$ is median-feasible, overlay Cartesian coordinates with origin at prize i_1 and positive x-axis towards prize i_2; hence prize i_2 is located at $(d_{i_1 i_2}, 0)$. Let $Z = (z, 0)$. Since $d_{Ai_1} < d_{Bi_1}$, then $d_{BZ} + d_{Zi_2} \leq d_{Ai_1} + d_{i_1 i_2} < d_{Bi_1} + d_{i_1 i_2}$ implies $z > 0$, by the triangle inequality. Since $d_{Ai_2} < d_{Bi_2}$, then $d_{BZ} + d_{Zi_1} \leq d_{Ai_2} + d_{i_1 i_2} < d_{Bi_2} + d_{i_1 i_2}$ implies $z < d_{i_1 i_2}$, also by the triangle inequality. Hence a necessary condition is that Z is located between the two prizes.

Let (x, y) be the location of player $\mathcal{B}$ in this coordinate system. Then $d^2_{Bi_1} = x^2 + y^2$ and $d_{Bi_2} = (d_{i_1 i_2} - x)^2 + y^2$ and, therefore,

$$x = \frac{d^2_{i_1 i_2} + d^2_{Bi_1} - d^2_{Bi_2}}{2d_{i_1 i_2}}.$$

Also $d^2_{BZ} = (x - z)^2 + y^2$. Consider constraint (5.10) where $d_{i_1 i_2} - d_{Zi_2} = z$. Substituting, we have

$$\begin{aligned} & \sqrt{(x-z)^2 + y^2} \leq d_{Ai_1} + z \\ \Leftrightarrow \quad & x^2 - 2xz + z^2 + y^2 \leq d^2_{Ai_1} + 2z d_{Ai_1} + z^2 \\ \Leftrightarrow \quad & 2z(x + d_{Ai_1}) \geq d^2_{Bi_1} - d^2_{Ai_1} \\ \Leftrightarrow \quad & z \geq \frac{d^2_{Bi_1} - d^2_{Ai_1}}{2(x + d_{Ai_1})} \end{aligned} \tag{5.12}$$

Consider constraint (5.11) where $d_{Zi_1} = z$. Substituting, we have

$$\begin{aligned} & \sqrt{(x-z)^2 + y^2} \leq d_{Ai_2} + d_{i_1 i_2} - z \\ \Leftrightarrow \quad & x^2 - 2xz + z^2 + y^2 \leq (d_{Ai_2} + d_{i_1 i_2})^2 + z^2 - 2z(d_{Ai_2} + d_{i_1 i_2}) \\ \Leftrightarrow \quad & 2z(d_{Ai_2} + d_{i_1 i_2} - x) \leq (d_{Ai_2} + d_{i_1 i_2})^2 - d^2_{Bi_1} \\ \Leftrightarrow \quad & z \leq \frac{(d_{Ai_2} + d_{i_1 i_2})^2 - d^2_{Bi_1}}{2(d_{Ai_2} + d_{i_1 i_2} - x)} \end{aligned} \tag{5.13}$$

Finally, let

$$z_1 = \frac{d^2_{B1} - d^2_{A1}}{2(x + d_{A1})} \tag{5.14}$$

$$z_2 = \frac{(d_{A2} + d_{12})^2 - d^2_{B1}}{2(d_{A2} + d_{12} - x)} \tag{5.15}$$

Hence we have established the following Lemma.

Lemma 5.1.4

$\mathcal{B} \rhd \{i_1, i_2\}$ is **median-feasible** if and only if $z_1 \leq z_2$. ■

The set of locations $Z = (z, 0)$ satisfying $z_1 \leq z \leq z_2$ is called the **median window** associated with $\mathcal{B} \rhd \{i_1, i_2\}$.

5.1.5 Sensitivity Diagrams for Two Prize Problem

In this section we characterise the sensitivity of the two prize problem to variation in the location of one player or variation in the location of one prize. These characterisations are in terms of the static and dynamic cases of Sections 5.1.1–5.1.3 and the median-feasibility criterion of Section 5.1.4.

5.1.5.1 Player Location Sensitivity

Suppose we fix the location of player $\mathcal{B}$ and the locations of prizes 1 and 2. We wish to characterise the location of player $\mathcal{A}$ by partitioning the plane $\Re^2$. There are eight characteristic curves which are the loci of points satisfying the following:

(i). $d_{A1} = d_{B1}$: Circle centred at prize 1 of radius d_{B1}.

(ii). $d_{A2} = d_{B2}$: Circle centred at prize 2 of radius d_{B2}.

(iii). $d_{A1} + d_{12} = d_{B2}$: Circle centred at prize 1 of radius $d_{B2} - d_{12}$.

(iv). $d_{A2} + d_{12} = d_{B1}$: Circle centred at prize 2 of radius $d_{B1} - d_{12}$.

(v). $d_{B1} + d_{12} = d_{A2}$: Circle centred at prize 2 of radius $d_{B1} + d_{12}$.

(vi). $d_{B2} + d_{12} = d_{A1}$: Circle centred at prize 1 of radius $d_{B2} + d_{12}$.

(vii). **$\mathcal{A}$ median-feasible boundary.** We want to plot the locations of player $\mathcal{A}$ such that $\exists$ a median-location Z between prize 1 and prize 2 for which equations (5.16)–(5.17) hold.

$$d_{B1} + d_{1Z} = d_{AZ} \tag{5.16}$$
$$d_{B2} + d_{2Z} = d_{AZ} \tag{5.17}$$

Since $\Delta B12$ is a triangle, $\exists$ a location Z a distance z from prize 1 towards prize 2 such that $d_{B1} + z = d_{B2} + d_{12} - z$ and $0 < z < d_{12}$, i.e.,

$$z = \tfrac{1}{2}(d_{B2} + d_{12} - d_{B1}).$$

Then the $\mathcal{A}$-median-feasible boundary is the circle centred at Z of radius $d_{B1} + z$.

(viii). **$\mathcal{B}$ median-feasible boundary.** Define Cartesian coordinates with origin at prize 1 and positive x-axis towards prize 2. We require those locations $A = (x, y)$ such that $\exists$ a location $Z = (z, 0)$ for which $d_{A1} + z = d_{BZ} = d_{A2} + d_{12} - z$. Choose $z \in [0, d_{12}]$. This defines d_{BZ}. Now it is simple to construct a location A such that $d_{A1} = d_{BZ} - z$ and $d_{A2} = d_{BZ} + z - d_{12}$. Hence we construct the $\mathcal{B}$-median-feasible boundary from all such locations A by parameterising on $z \in [0, d_{12}]$.

Figure 5.5 shows the upper half-plane of the resulting diagram; the lower half-plane situation is symmetric. The *unshaded* regions represent the locations (in the half-plane above the prizes) for player $\mathcal{A}$ for which the game is *guarantee determined* with outcomes as indicated (i.e. static cases). The shaded regions represent the locations for player $\mathcal{A}$ for which one player is at least as close to both prizes but cannot guarantee either sequence of prizes (i.e. dynamic cases). In particular, the inner darkly shaded region represents the locations for player $\mathcal{A}$ for which $\mathcal{B} \rhd \{1, 2\}$ is *median-feasible* and the outer darkly shaded region represents the locations for player $\mathcal{A}$ for which $\mathcal{A} \rhd \{1, 2\}$ is *median-feasible*.

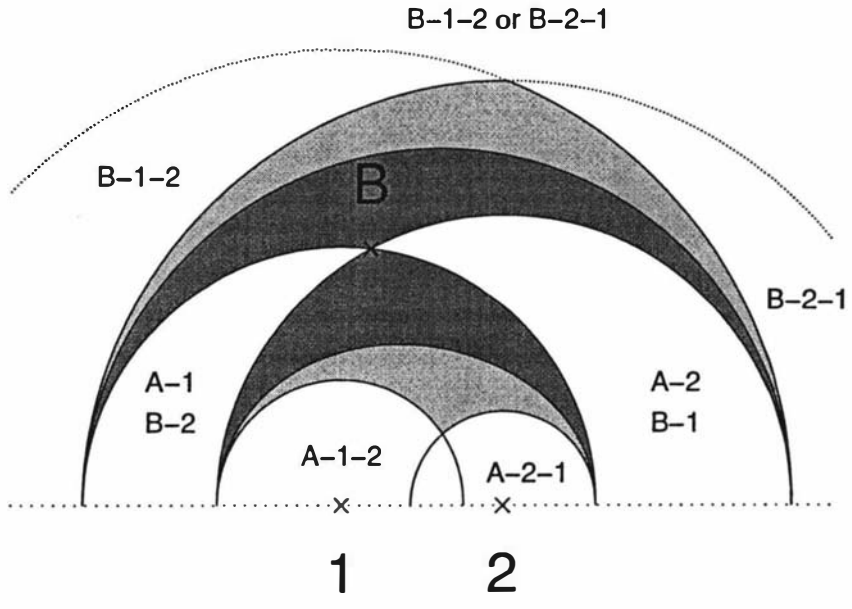


Figure 5.5: Player Location Sensitivity of Two Prize Problem

5.1.5.2 Prize Location Sensitivity

Suppose we fix the location of players $\mathcal{A}$ and $\mathcal{B}$ and the location of prize 1 such that $d_{A1} < d_{B1}$. We wish to characterize the location of prize 2 by partitioning the plane $\Re^2$. There are four characteristic curves which are the loci of points satisfying the following:

(i). $d_{A2} = d_{B2}$: Represented by the line of points equidistant from each player.

(ii). $d_{A1} + d_{12} = d_{B2}$: Represented by one half of a hyperbola with foci at prize 1 and player $\mathcal{B}$ and with d_{A1} as the length of the transverse axis.

(iii). $d_{A2} + d_{12} = d_{B1}$: Represented by an ellipse with foci at prize 1 and player $\mathcal{A}$ and with d_{B1} as the length of the major axis.

(iv). **$\mathcal{B}$ median-feasible boundary.** Determine the locations of prize 2 such that $\exists$ a median-location Z between prize 1 and prize 2 such that equations (5.18)–(5.19) hold.

$$d_{A1} + d_{1Z} = d_{BZ} \tag{5.18}$$
$$d_{A2} + d_{2Z} = d_{BZ} \tag{5.19}$$

Since constraint (5.18) does not involve prize 2, we can generate the locations Z from constraint (5.18) forming one half of a hyperbola.

Given a particular location Z such that $d_{A1} + d_{1Z} = d_{BZ}$, we can use the requirement that $d_{A2} + d_{2Z} = d_{A1} + d_{1Z}$ to construct the location of prize 2. Impose a coordinate system with origin at prize 1 and positive x-axis towards location Z. Suppose that player $\mathcal{A}$ is located at (x, y) and location Z is at $(z, 0)$. Then

$$x = \frac{z^2 + d_{A1}^2 - d_{AZ}^2}{2z}.$$

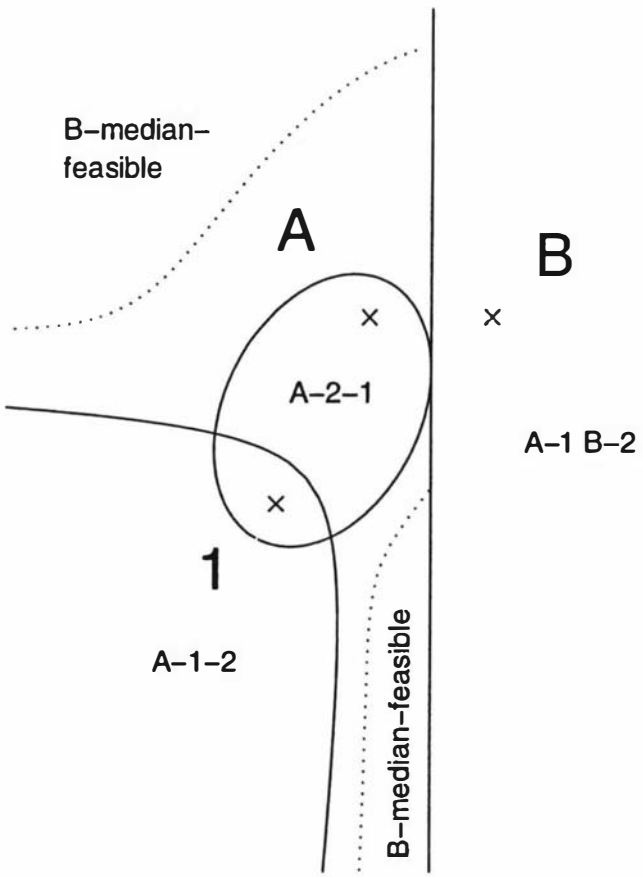


Figure 5.6: Prize Location Sensitivity of Two Prize Problem

Let $(a, 0)$ be the corresponding location of prize 2. Then

$$\begin{aligned}
& d_{A2} + d_2 z = d_{A1} + d_1 z \\
\Leftrightarrow\ & \sqrt{(a-x)^2 + y^2} + a - z = d_{A1} + z \\
\Leftrightarrow\ & (a-x)^2 + y^2 = (d_{A1} + 2z - a)^2 \quad (\text{since } a - z < d_{A1} + z) \\
\Leftrightarrow\ & y^2 + a^2 - 2ax + x^2 = d_{A1}^2 + 4z^2 + a^2 + 4z d_{A1} - 2a d_{A1} - 4ax \\
\Leftrightarrow\ & 2a(d_{A1} + z - x) = d_{A1}^2 - x^2 - y^2 + 4z^2 + 4z d_{A1} \\
\Leftrightarrow\ & 2a(d_{A1} + z - x) = 4z(z + d_{A1}) \quad (\text{since } d_{A1}^2 = x^2 + y^2) \\
\Leftrightarrow\ & a = \frac{2z(z + d_{A1})}{d_{A1} + z - x}
\end{aligned} \tag{5.20}$$

Figure 5.6 shows the resulting sensitivity diagram. The unlabelled regions correspond to those case **(viii)** dynamic game positions of Section 5.1.3 for which $\mathcal{B} \rhd \{1, 2\}$ is $\mathcal{B}$-median-feasible.

5.1.5.3 Discussion

Sensitivity analysis provides a means of studying the structure of a problem instance. From a particular two prize problem instance we can construct four sensitivity diagrams, perturbing the

location of each player and each prize respectively. From the local sensitivity of each diagram about the actual location of the player or prize, we can consider the "knife-edge" decisions of each player. We return to these ideas in Section 9.3 in which they are eventually used to design difficult problem instances.

5.2 Window Feasibility Subproblem

Consider a problem involving three prizes $\{1,2,3\}$ in which player $\mathcal{A}$ is closer to all three prizes. Suppose also that player $\mathcal{A}$ is, for some reason, unconditionally committed to prize 3, for which we already have the notation "$\mathcal{A} \triangleright 3$". Finally suppose that constraints (5.21)–(5.24) hold.

$$d_{A3} + d_{31} < d_{B1} \tag{5.21}$$
$$d_{A3} + d_{32} < d_{B2} \tag{5.22}$$
$$d_{A3} + d_{31} + d_{12} > d_{B2} \tag{5.23}$$
$$d_{A3} + d_{32} + d_{12} > d_{B1} \tag{5.24}$$

That is, player $\mathcal{A}$ can guarantee $\mathcal{A}\triangleright(3\rightarrow 1)$ and $\mathcal{A}\triangleright(3\rightarrow 2)$ but not $\mathcal{A}\triangleright(3\rightarrow 1\rightarrow 2)$ or $\mathcal{A}\triangleright(3\rightarrow 2\rightarrow 1)$. Player $\mathcal{B}$ is committed to attempting to claim one of prizes 1 and 2, for which we have the notation "$\mathcal{B} \triangleright \{1,2\}$".

Suppose that upon claiming prize 3, player $\mathcal{A}$ observes the actual location of player $\mathcal{B}$ at that time and then decides between targeting prize 1 or prize 2. This is a contingent strategy of player $\mathcal{B}$ since a decision is made at some future time (the time at which player $\mathcal{A}$ claims prize 3) about which prize to target. The problem for player $\mathcal{B}$ is to determine how to move during the period in which player $\mathcal{A}$ moves to prize 3 such that, at the time at which player $\mathcal{A}$ claims prize 3, player $\mathcal{B}$ is located at a position at which the pair constraints still hold on the remaining prizes $\{1,2\}$.

We now generalize this scenario and present the construction of a *feasibility window* through which player $\mathcal{B}$ must move in order that the pair constraints on the two remaining prizes continue to hold.

Consider a more general scenario in which player $\mathcal{A}$ unconditionally commits to arriving at some location X at time t_X and player $\mathcal{B}$ unconditionally commits to arriving at some location Y at time t_Y. We can determine the status of the two prizes $\{i_1, i_2\}$ under this scenario. If $t_X = t_Y$ then this is just the two prize problem. Hence, without loss of generality, suppose $t_X > t_Y$.

We generalize the analysis of the two prize problem to this scenario. Analysis of the corresponding *static* cases is similar, but the *dynamic* cases, characterised by constraints (5.25)–(5.28), exhibit a new difficulty.

$$t_X + d_{Xi_1} < t_Y + d_{Yi_1} \tag{5.25}$$
$$t_X + d_{Xi_2} < t_Y + d_{Yi_2} \tag{5.26}$$
$$t_X + d_{Xi_1} + d_{i_1 i_2} \geq t_Y + d_{Yi_2} \tag{5.27}$$
$$t_X + d_{Xi_2} + d_{i_1 i_2} \geq t_Y + d_{Yi_1} \tag{5.28}$$

Note that these constraints generalize (5.21)–(5.24). The corresponding contingent strategy of player $\mathcal{A}$ is to observe the actual location of player $\mathcal{B}$ at time t_X and then decide between targeting prize i_1 or prize i_2.

We wish to determine a location, Z, for player $\mathcal{B}$ at time t_X such that the pair constraints definitely hold at time t_X. Then such a location Z must satisfy constraints (5.29)–(5.31).

$$t_X \geq t_Y + d_{YZ} \tag{5.29}$$
$$d_{Xi_1} + d_{i_1 i_2} \geq d_{Zi_2} \tag{5.30}$$
$$d_{Xi_2} + d_{i_1 i_2} \geq d_{Zi_1} \tag{5.31}$$

Constraint (5.29) states the time available for player $\mathcal{B}$ to travel from location Y to location Z is at most the difference in arrival-times of the players. Hence player $\mathcal{B}$ may travel no further than $t_X - t_Y$ from location Y. Constraints (5.30)–(5.31) are the pair constraints applying at time t_X.

The set of locations Z satisfying constraints (5.29)–(5.31) is called a **feasibility window**, through which player $\mathcal{B}$ must pass at time t_X. If such a location Z exists then the window scenario is called **window feasible**. If no such location Z exists then player $\mathcal{A}$ can safely play the contingent strategy to guarantee either $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ or $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ depending upon the location of player $\mathcal{B}$ at time t_X.

Lemma 5.2.1

The set of locations Z satisfying constraints (5.29)–(5.31) corresponds to the intersection of the following three circles.

C_1: Centre at prize i_1 and radius $r_1 = d_{Xi_2} + d_{i_1 i_2}$
C_2: Centre at prize i_2 and radius $r_2 = d_{Xi_1} + d_{i_1 i_2}$
C_3: Centre at Y and radius $r_3 = t_X - t_Y$

Proof:

Circle C_1 corresponds to constraint (5.31), circle C_2 corresponds to constraint (5.30) and circle C_3 corresponds to constraint (5.29). ■

Hence the feasibility window may be determined via a geometric construction. Algorithm 5.1 WINDOW FEASIBLE presents an algorithm for determining if the three circles overlap (and hence if $\exists$ a feasibility window) by determining if $C_3 \cap C_1 \neq \emptyset$, $C_3 \cap C_2 \neq \emptyset$ and $(C_3 \cap C_1) \cap (C_3 \cap C_2) \neq \emptyset$.

Example 5.2.2

Consider the three prize problem of Figure 5.7(a). At time t_Y player $\mathcal{B}$ is at location $Y = B$ and player $\mathcal{A}$ is at location A. Player $\mathcal{A}$ is unconditionally committed to prize 3 and will arrive at the location of prize 3 (location X) at time $t_X = t_Y + d_{A3}$. Figure 5.7(a) illustrates the geometric construction of the corresponding feasibility window. Since the three circles have a common intersection then the scenario $\mathcal{A} \rhd 3$ and $\mathcal{B} \rhd \{1, 2\}$ is window feasible. Player $\mathcal{B}$ must play so as to reach the feasibility window (the intersection of the three circles) at time t_X.

Figure 5.7(b) illustrates a scenario which is not window feasible since the three circles have no common intersection. □

5.3 Strategies for Two Prize Problem

Suppose that there are two prizes $\{i_1, i_2\}$ and that the prize and pair constraints (5.1)–(5.4) hold with strict inequality as in **case (viii)** of Section 5.1.3. We now turn to the design of strategies

Algorithm 5.1 procedure WINDOW FEASIBLE

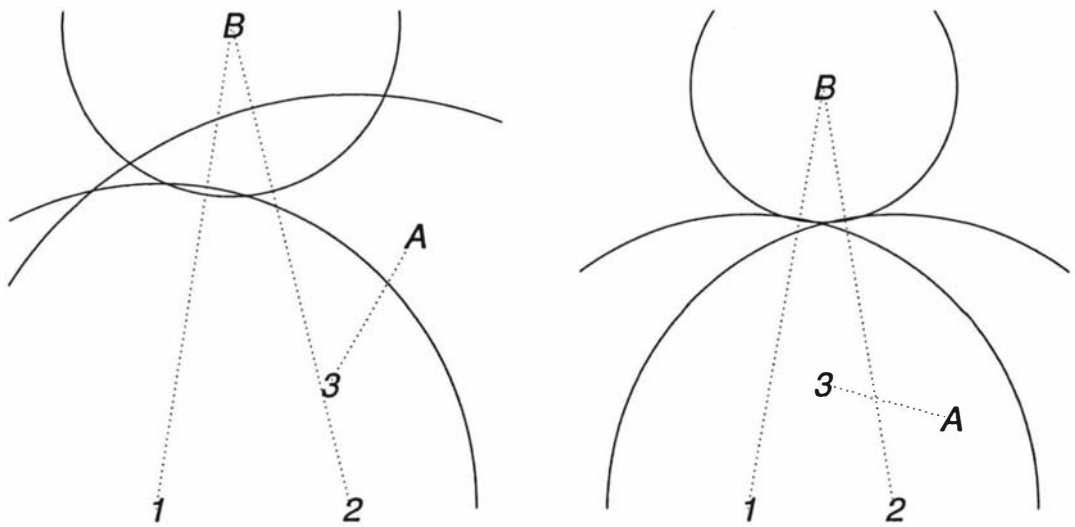
Input: $Y, i_1, i_2, r_1, r_2, r_3$.
Output: infeasible *or (***feasible** *and* $\psi_1, \psi_2, \delta_1, \delta_2$*)*.

if $((r_1 \geq d_{Yi_1})$ **or** $(r_3 \geq d_{Yi_1})$ **or** $(r_2 \geq d_{Yi_2})$ **or** $(r_3 \geq d_{Yi_2}))$ **then**
  return(feasible)
end

// Infeasible if $C_3 \cap C_1 = \emptyset$ *or* $C_3 \cap C_2 = \emptyset$.
if $((r_3 + r_1 < d_{Yi_1})$ **or** $(r_3 + r_2 < d_{Yi_2}))$ **then**
  return(infeasible)
end

// Feasible $\Leftrightarrow (C_3 \cap C_1) \cap (C_3 \cap C_2) \neq \emptyset$.
$\psi_1 \leftarrow$ bearing of prize i_1 from location Y
$\psi_2 \leftarrow$ bearing of prize i_2 from location Y
if $(d_{Yi_1} = r_1 + r_3)$ **then**
  $\delta_1 \leftarrow 0$
else
  $$x \leftarrow \frac{r_3^2 + d_{Yi_1}^2 - r_1^2}{2d_{Yi_1}}$$
  $$y \leftarrow \sqrt{r_3^2 - x^2}$$
  $$\delta_1 \leftarrow \arctan \frac{y}{x}$$
end
if $(d_{Yi_2} = r_2 + r_3)$ **then**
  $\delta_2 \leftarrow 0$
else
  $$x \leftarrow \frac{r_3^2 + d_{Yi_2}^2 - r_2^2}{2d_{Yi_2}}$$
  $$y \leftarrow \sqrt{r_3^2 - x^2}$$
  $$\delta_2 \leftarrow \arctan \frac{y}{x}$$
end
if $(\delta_1 + \delta_2 <$ acute angle between bearings ψ_1 and $\psi_2)$ **then**
  return(infeasible)
else
  return(feasible)
end

end



(a) Feasibility Construction (b) No Common Intersection

Figure 5.7: Three Prize Problem

for this dynamic case of the Two Prize Problem. Such strategies are STEP-MONITORS since the decision problem is to determine which one-step move to make rather than which prize to target. The case in which $d_{Ai_1} \leq \Delta$ or $d_{Ai_2} \leq \Delta$ is considered separately in Section 5.3.8; until then, suppose that $d_{Ai_1} > \Delta$ and $d_{Ai_2} > \Delta$. Shortly we will propose a *framework* for such two prize strategies, but firstly some preliminaries are necessary.

5.3.1 Slack Variables

Define the following **slack variables** corresponding to constraints (5.1)–(5.4).

$$\begin{aligned} s_1 &= d_{Bi_1} - d_{Ai_1} \\ s_2 &= d_{Bi_2} - d_{Ai_2} \\ s_3 &= d_{Ai_1} + d_{i_1 i_2} - d_{Bi_2} \\ s_4 &= d_{Ai_2} + d_{i_1 i_2} - d_{Bi_1} \end{aligned}$$

Hence the two prize case in which we are interested is characterised by $s_1 > 0$, $s_2 > 0$, $s_3 > 0$ and $s_4 > 0$. By simple substitutions we obtain the following result.

Result 5.3.1

$$\begin{aligned} s_1 + s_2 + s_3 + s_4 &= 2d_{i_1 i_2} \\ s_1 + s_3 &= d_{i_1 i_2} + d_{Bi_1} - d_{Bi_2} \\ s_1 + s_4 &= d_{i_1 i_2} + d_{Ai_2} - d_{Ai_1} \\ s_2 + s_3 &= d_{i_1 i_2} + d_{Ai_1} - d_{Ai_2} \\ s_2 + s_4 &= d_{i_1 i_2} + d_{Bi_2} - d_{Bi_1} \end{aligned}$$

■

5.3.1.1 Objective and Constraints

The objective of each player is to maximize the value in prizes claimed. We can now formulate this objective in terms of Assumption 5.1.2 and the slack variables.

Player $\mathcal{A}$ *must ensure* that the prize constraints continue to hold after making a one-step move; hence $s_1 > 0$ and $s_2 > 0$ are constraints. If this is possible, then player $\mathcal{A}$'s objective is to achieve either $s_3 \leq 0$ or $s_4 \leq 0$. If this is not possible then $\mathcal{A}\multimap$ argmax$\{v_{i_1}, v_{i_2}\}$ is *critical.*

Player $\mathcal{B}$ *must ensure* that the pair constraints continue to hold after making a one-step move; hence $s_3 > 0$ and $s_4 > 0$ are constraints. If this is possible then player $\mathcal{B}$'s objective is to achieve either $s_1 \leq 0$ or $s_2 \leq 0$. If this is not possible then $\mathcal{B}\multimap$ argmin$\{v_{i_1}, v_{i_2}\}$ is *critical.*

Result 5.3.1 has several consequences in terms of the tradeoff between both players attempting to satisfy their objective. Since $s_1 + s_2 + s_3 + s_4 = 2d_{i_1 i_2}$ is a constant, minimizing $s_3 + s_4$ is equivalent to maximizing $s_1 + s_2$ and vice versa. In this way the players' objectives are opposed. However, if $s_3 \ll s_4$ then player $\mathcal{B}$ can "defend" s_3 via $\mathcal{B}\multimap i_2$ and similarly if $s_3 \gg s_4$. Suppose player $\mathcal{A}$ converges to prize i_1 and the prize and pair constraints continue to hold. Then $d_{Ai_2} - d_{Ai_1}$ converges to $d_{i_1 i_2}$, $s_2 + s_3$ converges to 0 and hence s_2 and s_3 both converge to 0. Thus as player $\mathcal{A}$ anticipates achieving its objective, player $\mathcal{B}$ also anticipates achieving its objective.

5.3.1.2 Critical Values of Slack Variables

A one-step move is **safe** for player $\mathcal{A}$ if it ensures that both prize constraints continue to hold. A one-step move is **safe** for player $\mathcal{B}$ if it ensures that both pair constraints continue to hold. We now give some computable bounds on the slack variables such that $\exists$ a safe one-step move for each player.

Critical values of s_1 and s_2. Let θ_A be the acute angle between the prizes from the perspective of player $\mathcal{A}$. The maximum decrease in s_1 from one step occurs when $\mathcal{A}\multimap i_2$ and $\mathcal{B}\multimap i_1$ and hence

$$s_1^{crit} = \Delta + d_{Ai_1} - \sqrt{d_{Ai_1}^2 + \Delta^2 - 2\Delta d_{Ai_1} \cos\theta_A} \tag{5.32}$$

by the *Cosine Law.* Similarly the maximum decrease in s_2 from one step occurs when $\mathcal{A}\multimap i_1$ and $\mathcal{B}\multimap i_2$ and hence

$$s_2^{crit} = \Delta + d_{Ai_2} - \sqrt{d_{Ai_2}^2 + \Delta^2 - 2\Delta d_{Ai_2} \cos\theta_A} \tag{5.33}$$

If $s_1 > s_1^{crit}$ or $s_2 > s_2^{crit}$ then one of $\mathcal{A}\multimap i_1$ or $\mathcal{A}\multimap i_2$ is **safe**, i.e., both prize constraints will continue to hold after the one-step moves have been made. If $s_1 \leq s_1^{crit}$ and $s_2 \leq s_2^{crit}$ then player $\mathcal{A}$ is potentially in danger of relinquishing the claim to one of the prizes as neither $\mathcal{A}\multimap i_1$ nor $\mathcal{A}\multimap i_2$ is *safe.*

Critical values of s_3 and s_4. Let θ_B be the acute angle between the prizes from the perspective of player $\mathcal{B}$. The maximum decrease in s_3 from one step occurs when both $\mathcal{A}\multimap i_1$ and $\mathcal{B}\multimap i_1$ and hence

$$s_3^{crit} = \Delta + d_{Bi_2} - \sqrt{d_{Bi_2}^2 + \Delta^2 - 2\Delta d_{Bi_2} \cos\theta_B} \tag{5.34}$$

by the *Cosine Law*. Similarly the maximum decrease in s_4 from one step occurs when $\mathcal{A}$—$\circ i_2$ and $\mathcal{B}$—$\circ i_2$ and hence

$$s_4^{crit} = \Delta + d_{Bi_1} - \sqrt{d_{Bi_1}^2 + \Delta^2 - 2\Delta d_{Bi_1} \cos\theta_B} \tag{5.35}$$

If $s_3 > s_3^{crit}$ or $s_4 > s_4^{crit}$ then one of $\mathcal{B}$—$\circ i_1$ or $\mathcal{B}$—$\circ i_2$ is **safe**, i.e., both pair constraints will continue to hold after the one-step moves have been made. If $s_3 \leq s_3^{crit}$ and $s_4 \leq s_4^{crit}$ then player $\mathcal{B}$ is potentially in danger of relinquishing the claim to one of the sequences $\mathcal{A} \triangleright (i_1 \rightarrow i_2)$ or $\mathcal{A} \triangleright (i_2 \rightarrow i_1)$ as neither $\mathcal{B}$—$\circ i_1$ nor $\mathcal{B}$—$\circ i_2$ is *safe*.

5.3.2 Safety Windows

When one of the s_i values falls below its critical value we must determine which one-step moves are *safe*. The set of safe one-step moves correspond to a *window* through, which the player must play which we call a **safety window**.

5.3.2.1 Player $\mathcal{A}$ Safety Window

At some time t_0, suppose $s_1 \leq s_1^{crit}$ or $s_2 \leq s_2^{crit}$, but $d_{Ai_1} > \Delta$ and $d_{Ai_2} > \Delta$.

We wish to determine a location, Z, for player $\mathcal{A}$ at time $t_0 + \Delta$ such that $s_1 \geq 0$ and $s_2 \geq 0$ definitely hold at time $t_0 + \Delta$. Then such a location Z must satisfy constraints (5.36)–(5.38).

$$\Delta \geq d_{AZ} \tag{5.36}$$
$$d_{Bi_1} - \Delta \geq d_{Zi_1} \tag{5.37}$$
$$d_{Bi_2} - \Delta \geq d_{Zi_2} \tag{5.38}$$

This is a *window feasibility subproblem* as in Section 5.2 except that it is player $\mathcal{A}$ who wishes to determine a feasibility window and player $\mathcal{B}$ has no committed target location at time $t_0 + \Delta$.

Hence apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the prizes i_1 and i_2, location Y is the location of player $\mathcal{A}$ and

$$\begin{aligned} r_1 &= d_{Bi_1} - \Delta \\ r_2 &= d_{Bi_2} - \Delta \\ r_3 &= \Delta \end{aligned}$$

If such a location Z exists then this game position is $\mathcal{A}$-**safe** and the corresponding feasibility window is called the $\mathcal{A}$-**safety-window**.

5.3.2.2 Player $\mathcal{B}$ Safety Window

At some time t_0, suppose $s_3 \leq s_3^{crit}$ or $s_4 \leq s_4^{crit}$, but $d_{Ai_1} > \Delta$ and $d_{Ai_2} > \Delta$.

We wish to determine a location, Z, for player $\mathcal{B}$ at time $t_0 + \Delta$ such that $s_3 \geq 0$ and $s_4 \geq 0$ definitely hold at time $t_0 + \Delta$. Then such a location Z must satisfy constraints (5.39)–(5.41).

$$\Delta \geq d_{BZ} \tag{5.39}$$
$$(d_{Ai_1} - \Delta) + d_{i_1 i_2} \geq d_{Zi_2} \tag{5.40}$$
$$(d_{Ai_2} - \Delta) + d_{i_1 i_2} \geq d_{Zi_1} \tag{5.41}$$

This is a *window feasibility subproblem* as in Section 5.2 except that player $\mathcal{A}$ has no committed target location at time $t_0 + \Delta$.

Hence apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the prizes i_1 and i_2, location Y is the location of player $\mathcal{B}$ and

$$\begin{aligned} r_1 &= d_{Ai_2} - \Delta + d_{i_1 i_2} \\ r_2 &= d_{Ai_2} - \Delta + d_{i_1 i_2} \\ r_3 &= \Delta \end{aligned}$$

If such a location Z exists then this game position is **$\mathcal{B}$-safe** and the corresponding feasibility window is called the **$\mathcal{B}$-safety-window**.

5.3.3 Two Prize Strategy Framework

We now propose a *framework* for a two prize strategy consisting of two components.

(a). Construction of a **target window** (an interval of bearings) through which the player must play its one-step move. The *default target window* (that which is least restrictive) is the window which spans the two prizes via the acute angle between their bearings.

(b). Selection of a **target location** within the *target window* which the player will move towards as its one-step move.

5.3.3.1 Construction of Target Window

Suppose either $s_1 \leq s_1^{crit}$ or $s_2 \leq s_2^{crit}$. If the game position is $\mathcal{A}$-safe, then the $\mathcal{A}$-target-window is the $\mathcal{A}$-safety-window; otherwise the $\mathcal{A}$-target-window is the default window.
Suppose both $s_1 > s_1^{crit}$ and $s_2 > s_2^{crit}$. Then the $\mathcal{A}$-target-window is the default window.

Suppose either $s_3 \leq s_3^{crit}$ or $s_4 \leq s_4^{crit}$. If the game position is $\mathcal{B}$-safe, then the $\mathcal{B}$-target-window is the $\mathcal{B}$safety-window; otherwise the $\mathcal{B}$-target-window is the default window.
Suppose both $s_3 > s_3^{crit}$ and $s_4 > s_4^{crit}$. If $\mathcal{B} \rhd \{i_1, i_2\}$ is median-feasible, then the $\mathcal{B}$-target-window is the $\mathcal{B}$-median-window of Section 5.1.4; otherwise the $\mathcal{B}$-target-window is the default window.

Note. If player $\mathcal{B}$ plays to the $\mathcal{B}$-median-window then $\mathcal{B} \rhd \{i_1, i_2\}$ remains median-feasible and the $\mathcal{B}$-median-window will continue to enlarge until eventually $s_1 \leq 0$ or $s_2 \leq 0$ is achieved. However, it is not necessarily always true that if a player plays to a safety-window then the resulting game position will also have a safety-window.

5.3.3.2 Selection of Target Location

Sections 5.3.4–5.3.7 present simple heuristic strategies for the selection of a either a target-prize or a target-bearing. If a target-prize is selected then it is translated to the corresponding end of the target-window. If a target-bearing is selected and this is outside the target-window then this also is translated to the corresponding end of the target-window.

Algorithm 5.2 strategy TWO-PRIZE AHEAD PURE-ATTACK

$k^* \leftarrow \text{argmin}_{k\in\{1,2\}} s_{k+2}$
target $\leftarrow i_{k^*}$

end

Algorithm 5.3 strategy TWO-PRIZE AHEAD PURE-DEFEND

$k^* \leftarrow \text{argmin}_{k\in\{1,2\}} s_k$
target $\leftarrow i_{k^*}$

end

5.3.4 A Family of ATTACK and DEFEND Strategies

The basic strategy requirement for player $\mathcal{A}$ is to ensure $s_1 > 0$ and $s_2 > 0$ continue to hold and attempt to force either $s_3 \leq 0$ or $s_4 \leq 0$. This gives rise two strategies for player $\mathcal{A}$: Algorithm 5.2 TWO-PRIZE AHEAD PURE-ATTACK and Algorithm 5.3 TWO-PRIZE AHEAD PURE-DEFEND.

The basic strategy requirement for player $\mathcal{B}$ is to ensure $s_3 > 0$ and $s_4 > 0$ continue to hold and attempt to force either $s_1 \leq 0$ or $s_2 \leq 0$. This gives rise to two strategies for player $\mathcal{B}$: Algorithm 5.4 TWO-PRIZE BEHIND PURE-ATTACK and Algorithm 5.5 TWO-PRIZE BEHIND PURE-DEFEND.

The *pure attack* strategies persistently attack the weakest of two constraints and the *pure defend* strategies continually defend the weakest of two constraints. Rather than attack the weakest of two constraints, an alternative *pure attack* strategy is to attack the strongest of two constraints, thus continually reducing a ceiling on the opponent's overall strength, as in Algorithm 5.6 TWO-PRIZE AHEAD CEILING-ATTACK and Algorithm 5.7 TWO-PRIZE BEHIND CEILING-ATTACK.

Consider the following scenarios.

(i). s_1 *small,* s_2 *small,* s_3 *large,* s_4 *large.* Player $\mathcal{A}$ should defend and player $\mathcal{B}$ should attack.

(ii). s_1 *large,* s_2 *large,* s_3 *small,* s_4 *small.* Player $\mathcal{A}$ should attack and player $\mathcal{B}$ should defend.

(iii). s_1 *large,* s_2 *small,* s_3 *large,* s_4 *small.* Player $\mathcal{A}$—$\circ i_2$ is the obvious attacking or defensive move. Defence implies $\mathcal{B}$—$\circ i_1$ and attack implies $\mathcal{B}$—$\circ i_2$.

Algorithm 5.4 strategy TWO-PRIZE BEHIND PURE-ATTACK

$k^* \leftarrow \text{argmin}_{k\in\{1,2\}} s_k$
target $\leftarrow i_{k^*}$

end

Algorithm 5.5 strategy TWO-PRIZE BEHIND PURE-DEFEND

$k^* \leftarrow \text{argmin}_{k\in\{1,2\}} s_{k+2}$
target $\leftarrow i_{k^*}$

end

Algorithm 5.6 strategy TWO-PRIZE AHEAD CEILING-ATTACK

$k^* \leftarrow \text{argmax}_{k\in\{1,2\}} s_{k+2}$
target $\leftarrow i_{k^*}$

end

(iv). s_1 *large,* s_2 *small,* s_3 *small,* s_4 *large.* Player $\mathcal{B}$—$\circ i_2$ is the obvious attacking or defensive move. Defence implies $\mathcal{A}$—$\circ i_2$ and attack implies $\mathcal{A}$—$\circ i_1$.

In summary, player $\mathcal{A}$ should defend if and only if $\min\{s_1, s_2\}$ is small and player $\mathcal{B}$ should defend if and only if $\min\{s_3, s_4\}$ is small. Algorithm 5.8 TWO-PRIZE AHEAD THRESH-DEFEND and Algorithm 5.9 TWO-PRIZE BEHIND THRESH-DEFEND present two resulting strategies for a threshold parameter s_{thresh}.

5.3.5 A Family of TIT FOR TAT Strategies

Result 5.3.1 shows that $s_1 + s_3$ decreases as player $\mathcal{B}$ gets closer to prize i_1 and $s_2 + s_3$ decreases as player $\mathcal{A}$ gets closer to prize i_1.

Consider what conditions would imply that s_3 consistently decreases in a *prolonged play* of a two prize problem instance. Now $\mathcal{B}$—$\circ i_2$ implies that s_3 cannot decrease, so a necessary condition is that $\mathcal{B}$—$\circ i_1$. Also $\mathcal{B}$—$\circ i_1$ implies that $s_1 + s_3$ consistently decreases at a rate depending only on player $\mathcal{B}$'s location. Eventually player $\mathcal{A}$ must approach prize i_1 to claim the sequence $i_1 \rightarrow i_2$ and $\mathcal{A}$—$\circ i_1$ implies that s_1 cannot decrease. Hence it is reasonable to expect that $\mathcal{A}$—$\circ i_1$ and $\mathcal{B}$—$\circ i_1$ consistently for s_3 to monotonically decrease.

Even the above is by no means certain to guarantee that s_3 approaches zero as it is possible that $s_2 > 0$ may not hold throughout. However, it is enough to suggest a family of two prize strategies for both players based on the following two reasonable expectations for prolonged play:

Algorithm 5.7 strategy TWO-PRIZE BEHIND CEILING-ATTACK

$k^* \leftarrow \text{argmax}_{k\in\{1,2\}} s_k$
target $\leftarrow i_{k^*}$

end

Algorithm 5.8 strategy TWO-PRIZE AHEAD THRESH-DEFEND

Input: s_{thresh} *// Threshold parameter.*
if $(\min\{s_1, s_2\} \geq s_{thresh})$ **then**
// A-attack.
$k^* \leftarrow \text{argmin}_{k \in \{1,2\}} s_{k+2}$
else
// A-defend.
$k^* \leftarrow \text{argmin}_{k \in \{1,2\}} s_k$
end
target $\leftarrow i_{k^*}$

end

Algorithm 5.9 strategy TWO-PRIZE BEHIND THRESH-DEFEND

Input: s_{thresh} *// Threshold parameter.*
if $(\min\{s_3, s_4\} \geq s_{thresh})$ **then**
// B-attack.
$k^* \leftarrow \text{argmin}_{k \in \{1,2\}} s_k$
else
// B-defend.
$k^* \leftarrow \text{argmin}_{k \in \{1,2\}} s_{k+2}$
end
target $\leftarrow i_{k^*}$

end

Algorithm 5.10 strategy TIT FOR TAT BEHIND (BEARING)

// Target opposite to opponent's previous bearing.
target $\leftarrow$ Bearing $\frac{\theta_{Ai_1}}{\theta_{Ai_1}+\theta_{Ai_2}}\theta_B$ from ψ_{Bi_2} towards ψ_{Bi_1}.

end

(i). Player $\mathcal{A}$ would ideally like to consistently target one prize, say prize i_1, and discover that player $\mathcal{B}$ has also targeted prize i_1 over an reasonable length of time.

(ii). On the other hand, if player $\mathcal{A}$ targets one prize, say prize i_1, then player $\mathcal{B}$ would like to ensure that it defends the corresponding slack variable, s_3, by targeting prize i_2.

Each player *does not know* what its opponent's one-step move will be, but can use observation of the previous move in the history of the game as a substitute.

5.3.5.1 TIT FOR TAT BEHIND Strategy

The idea is that player $\mathcal{B}$ can avoid the scenario above by targeting the opposite prize to that which player $\mathcal{A}$ targeted at the previous iteration. This is an example of a strategy which Axelrod [7] calls TIT FOR TAT in the context of the iterated Prisoner's Dilemma game of Section 2.4.5 .

If the players were restricted to travelling only towards prizes then this would be straightforward. As no such restriction applies, we need a definition of an *opposite move* to the opponent's previous move. Such a definition must retain the criterion that if player $\mathcal{A}$ targets a prize then player $\mathcal{B}$ should target the opposite prize.

The idea is to determine what "weighting" of the prizes player $\mathcal{A}$ targeted at the previous iteration and then apply the weighting in reverse. Two possibilities include:

Bearings. Let ψ_{Ai_1} be the bearing of prize i_1 from player $\mathcal{A}$'s immediately preceding location and let ψ_{Ai_2} be the bearing of prize i_2 from player $\mathcal{A}$'s immediately preceding location. Let ψ_{A^-} be the bearing that player $\mathcal{A}$ moved at the immediately preceding step. If ψ_{A^-} is outside the acute angle between ψ_{Ai_1} and ψ_{Ai_2}, then redefine ψ_{A^-} as ψ_{Ai_1} or ψ_{Ai_2}, whichever is the closest bearing to the existing ψ_{A^-}. Let θ_{Ai_1} be the acute angle between ψ_{Ai_1} and ψ_{A^-} and let θ_{Ai_2} be the acute angle between ψ_{Ai_2} and ψ_{A^-}. Let $\gamma = \frac{\theta_{Ai_1}}{\theta_{Ai_1}+\theta_{Ai_2}}$. Let ψ_{Bi_1} be the bearing of prize i_1 from player $\mathcal{B}$'s current location and let ψ_{Bi_2} be the bearing of prize i_2 from player $\mathcal{B}$'s current location. Let θ_B be the acute angle between ψ_{Bi_1} and ψ_{Bi_2}. Let $\psi_{B^\bullet}$ be the bearing an angle $\gamma\theta_B$ from bearing ψ_{Bi_2} towards ψ_{Bi_1}. Player $\mathcal{B}$ then targets bearing $\psi_{B^\bullet}$.

The result is Algorithm 5.10 TIT FOR TAT BEHIND (BEARING).

Location. Let z_{A^-} be the displacement from prize i_1 towards prize i_2 along the "median line" through the two prizes which player $\mathcal{A}$ targeted at the immediately preceding step. If $z_{A^-} < 0$ then set $z_{A^-} = 0$. If $z \geq d_{i_1 i_2}$ then set $z_{A^-} = d_{i_1 i_2}$. Player $\mathcal{B}$ then targets the location a distance z_{A^-} from prize i_2 towards prize i_1.

Algorithm 5.11 strategy TIT FOR TAT BEHIND (LOCATION)

// Target opposite to opponent's previous median location.
target $\leftarrow$ Bearing of location a distance z_{A^-} from i_2 towards i_1.

end

Algorithm 5.12 strategy TIT FOR TAT AHEAD (BEARING)

// Target same as opponent's previous bearing.
target $\leftarrow$ Bearing $\frac{\theta_{Bi_1}}{\theta_{Bi_1}+\theta_{Bi_2}}\theta_A$ from ψ_{Ai_1} towards ψ_{Ai_2}.

end

The result is Algorithm 5.11 TIT FOR TAT BEHIND (LOCATION).

Note that in both cases, if player $\mathcal{A}$ targets prize i_1 then player $\mathcal{B}$ targets prize i_2 and vice versa.

5.3.5.2 TIT FOR TAT AHEAD Strategy

The idea is that player $\mathcal{A}$ targets the same target which player $\mathcal{B}$ targeted at the immediately preceding step.

The two possibilities analogous to the previous TIT FOR TAT strategy are:

Bearings. Let ψ_{Bi_1} be the bearing of prize i_1 from player $\mathcal{B}$'s immediately preceding location and let ψ_{Bi_2} be the bearing of prize i_2 from player $\mathcal{B}$'s immediately preceding location. Let $\psi_{B\bullet}$ be the bearing that player $\mathcal{B}$ moved at the immediately preceding step. If $\psi_{B\bullet}$ is outside the acute angle between ψ_{Bi_1} and ψ_{Bi_2} then redefine $\psi_{B\bullet}$ as ψ_{Bi_1} or ψ_{Bi_2} whichever is the closest bearing to the existing $\psi_{B\bullet}$. Let θ_{Bi_1} be the acute angle between ψ_{Bi_1} and $\psi_{B\bullet}$ and let θ_{Bi_2} be the acute angle between ψ_{Bi_2} and $\psi_{B\bullet}$. Let $\gamma = \frac{\theta_{Bi_1}}{\theta_{Bi_1}+\theta_{Bi_2}}$.

Let ψ_{Ai_1} be the bearing of prize i_1 from player $\mathcal{A}$'s current location and let ψ_{Ai_2} be the bearing of prize i_2 from player $\mathcal{A}$'s current location. Let θ_A be the acute angle between ψ_{Ai_1} and ψ_{Ai_2}. Let ψ_{A^-} be the bearing an angle $\gamma\theta_B$ from bearing ψ_{Ai_1} towards ψ_{Ai_2}. Player $\mathcal{A}$ then targets bearing ψ_{A^-}.

The result is Algorithm 5.12 TIT FOR TAT AHEAD (BEARING).

Location. Let $z_{B\bullet}$ be the displacement from prize i_1 towards prize i_2 along the "median line" through the two prizes which player $\mathcal{B}$ targeted at the immediately preceding step. If $z_{B\bullet} < 0$ then set $z_{B\bullet} = 0$. If $z \geq d_{i_1 i_2}$ then set $z_{B\bullet} = d_{i_1 i_2}$. Player $\mathcal{A}$ then targets the location a distance $z_{B\bullet}$ from prize i_1 towards prize i_2.

The result is Algorithm 5.13 TIT FOR TAT AHEAD (LOCATION).

Note that in both cases, if player $\mathcal{B}$ targets a prize then player $\mathcal{A}$ targets the same prize.

Algorithm 5.13 strategy TIT FOR TAT AHEAD (LOCATION)

// Target same as opponent's previous median location.
target $\leftarrow$ Bearing of location a distance $z_{B^\bullet}$ from i_1 towards i_2.

end

Algorithm 5.14 strategy TIT FOR TWO TATS AHEAD

if *Deviation in player $\mathcal{B}$ bearing over previous two steps* $\leq \theta_{thresh}$ **then**
Target as in TIT FOR TAT AHEAD.
else
target $\leftarrow$ Same $\mathcal{A}$-bearing as previous step.
end

end

5.3.5.3 TIT FOR TWO TATS AHEAD Strategy

Recall that player $\mathcal{A}$ hopes for some long term consistency in which both players target the same prize. Rather than player $\mathcal{A}$ responding immediately to a change in target by player $\mathcal{B}$, suppose we introduce some threshold delay in response. If player $\mathcal{B}$ maintains its bearing (up to a threshold parameter θ_{thresh}), then player $\mathcal{A}$ continues to play TIT FOR TAT AHEAD; otherwise player $\mathcal{A}$ targets the same bearing as at the immediately preceding one-step move. This is analogous to the TIT FOR TWO TATS strategy of Axelrod [7]. The result is Algorithm 5.14 TIT FOR TWO TATS AHEAD.

5.3.6 RANDOM Strategy

Random strategies incorporate some random selection in determining a target-prize or a target-bearing. The simplest random strategy is, at each step, to choose a prize from the set of remaining prizes, with equal probability, and target that prize. We include Algorithm 5.15 RANDOM PRIZE for comparison with the other strategies, due to its unpredictable nature.

Algorithm 5.15 strategy RANDOM PRIZE

Input: Q *// Set of remaining prizes.*
// Target random prize.
target $\leftarrow$ **rand**(Q)

end

Algorithm 5.16 strategy PREVIOUS MEDIAN

// Strategy for player $\mathcal{B}$.
Input: i_1, i_2 *// Prizes.*

$z \leftarrow \frac{1}{2}(d_{Ai_1} + d_{i_1 i_2} - d_{Ai_2})$
$Z \leftarrow$ location distance z from i_1 towards i_2
target $\leftarrow$ bearing of location Z

end

5.3.7 Previous Median Strategy

Suppose $\mathcal{B} \rhd \{i_1, i_2\}$ is not median-feasible. Recall from Section 5.1.5 that, given the locations of player $\mathcal{A}$ and the two prizes, the set of locations B such that $\exists$ a median-location Z for which $d_{Ai_1} + d_{i_1 Z} = d_{BZ} = d_{Ai_2} + d_{i_2 Z}$ form a circle centred at the location Z a distance $z = \frac{1}{2}(d_{Ai_2} + d_{i_1 i_2} - d_{Ai_1})$ from prize i_1 towards i_2. Hence a measure of the "median infeasibility" is $d_{BZ} - (d_{Ai_1} + d_{i_1 Z})$. Practically, this suggests a strategy for player $\mathcal{B}$ whether or not $\mathcal{B} \rhd \{i_1, i_2\}$ is median-feasible. Algorithm 5.16 PREVIOUS MEDIAN is the resulting strategy for player $\mathcal{B}$.

5.3.8 Last Resort Strategies

Finally we consider the case where either $d_{Ai_1} \leq \Delta$ or $d_{Ai_2} \leq \Delta$.

5.3.8.1 Player $\mathcal{B}$'s LAST RESORT MEDIAN Strategy

It is possible that the distance between player $\mathcal{B}$ and the line through the two prizes is $\leq \Delta$. Hence we need to determine if there exists a location Z on this line such that $d_{BZ} \leq \Delta$. This is the only occasion for which we must explicitly consider the discreteness of the one-step movements. There is no benefit in player $\mathcal{B}$ moving to the other side of the line so player $\mathcal{B}$ moves to some location Z on the line through the two prizes, arriving no later than the end of the current time step.[3] Hence, we wish to determine if $\exists$ a median-location Z such that constraints (5.42)–(5.44) hold.

$$d_{BZ} < \Delta \tag{5.42}$$
$$\Delta \leq d_{Ai_1} + d_{i_1 Z} \tag{5.43}$$
$$\Delta \leq d_{Ai_2} + d_{i_2 Z} \tag{5.44}$$

We also wish to determine if $\exists$ a "standard" median-location Z such that constraints (5.45)–(5.47) hold.

$$d_{BZ} \geq \Delta \tag{5.45}$$
$$d_{BZ} \leq d_{Ai_1} + d_{i_1 Z} \tag{5.46}$$
$$d_{BZ} \leq d_{Ai_2} + d_{i_2 Z} \tag{5.47}$$

[3] In this case player $\mathcal{B}$ moves a distance $\leq \Delta$ in time duration Δ.

Algorithm 5.17 strategy TWO-PRIZE BEHIND LAST-RESORT MEDIAN

$$x \leftarrow \frac{d_{i_1 i_2}^2 + d_{Bi_1}^2 - d_{Bi_2}^2}{2d_{i_1 i_2}}$$
$$y \leftarrow \sqrt{d_{Bi_1}^2 - x^2}$$
$$z_1 \leftarrow \frac{d_{Bi_1}^2 - d_{Ai_1}^2}{2(x + d_{Ai_1})}$$
$$z_2 \leftarrow \frac{(d_{Ai_2} + d_{i_1 i_2})^2 - d_{Bi_1}^2}{2(d_{Ai_2} + d_{i_1 i_2} - x)}$$

if $(y \leq \Delta)$ **then**

$$\delta x \leftarrow \sqrt{\Delta^2 + x^2 - d_{Bi_1}^2}$$
$$w_1 \leftarrow \max\{x - \delta x, \Delta - d_{Ai_1}\}$$
$$w_2 \leftarrow \min\{x + \delta x, d_{Ai_2} + d_{i_1 i_2} - \Delta\}$$

if $(w_1 \leq w_2)$ **then**

$$z_T \leftarrow \tfrac{1}{2}(w_1 + w_2)$$

else if $(z_2 \geq \max\{x + \delta x, z_1\})$ **then**

$$z_T \leftarrow \tfrac{1}{2}(z_2 + \max\{x + \delta x, z_1\})$$

else

$$z_T \leftarrow \tfrac{1}{2}(z_1 + \min\{x - \delta x, z_2\})$$

end

else

$$z_T \leftarrow \tfrac{1}{2}(z_1 + z_2)$$

end

target $\leftarrow$ Bearing of location a distance z_T from i_1 towards i_2.

end

A necessary condition for constraints (5.43)–(5.44) to hold is that $d_{Ai_1} + d_{Ai_2} + d_{i_1 i_2} \geq 2\Delta$. Recall from Assumption 3.1.3 that we assume that $d_{i_1 i_2} \geq \Delta$. Also, by the triangle inequality, $d_{Ai_1} + d_{Ai_2} \geq d_{i_1 i_2} \geq \Delta$ and hence the necessary condition is satisfied. Also, since $d_{Ai_1} + d_{Ai_2} + d_{i_1 i_2}$ then $\exists$ a median-location Z such that constraints (5.43)–(5.44) hold. Hence, by considering the cases $d_{BZ} < \Delta$ and $d_{BZ} \geq \Delta$, $\exists$ a median-location Z such that either constraints (5.42)–(5.44) hold or constraints (5.45)–(5.47).

Algorithm 5.17 TWO-PRIZE BEHIND LAST-RESORT MEDIAN presents the resulting strategy for player $\mathcal{B}$. Note also that this argument shows that the player $\mathcal{B}$ equivalent to the Two Prize Theorem (Theorem 5.1.1) is **not true.**

5.3.8.2 Player $\mathcal{A}$'s FINAL Strategy

Since $\exists$ a median-location Z for player $\mathcal{B}$ such that either constraints (5.42)–(5.44) or constraints (5.45)–(5.47) hold, player $\mathcal{A}$'s one-step move is critical. If $v_{i_1} > v_{i_2}$ then $\mathcal{A}$—$\circ i_1$ is *critical* to player $\mathcal{A}$. If $v_{i_1} < v_{i_2}$ then $\mathcal{A}$—$\circ i_2$ is *critical* to player $\mathcal{A}$. Otherwise if $v_{i_1} = v_{i_2}$ then player $\mathcal{A}$ can arbitrarily decide between $\mathcal{A}$—$\circ i_1$ and $\mathcal{A}$—$\circ i_2$.

5.4 Two Prize Problem with Finite Deadline

In this chapter so far we have analysed the two prize problem, and designed strategies, assuming that $\lambda = \infty$. Now we consider the case where $\lambda < \infty$ and Assumption 5.1.2 still applies.

5.4.1 Tactical Analysis

We must firstly make some definitions. If λ is sufficiently restrictive, then the value sum of the prizes that each player claims may not add up to the total prize pool value. In this case it may be beneficial to *cooperate*.

Definition 5.4.1

The **cooperative value**, $\Omega(\mathcal{A},\mathcal{B})$, is the maximum value that the players can jointly claim within the overall deadline λ if they cooperate completely. □

Definition 5.4.2

A prize i is **accessible** to player $\mathcal{X}$ at time t_0 if $t_0 + d_{\mathcal{X}i} \leq \lambda$. A sequence $\mathcal{X} \triangleright (i_1 \rightarrow i_2)$ is **accessible** to player $\mathcal{X}$ at time t_0 if $t_0 + d_{\mathcal{X}i_1} + d_{i_1 i_2} \leq \lambda$. □

The cooperative value is the sum of a subset of the prize values since each prize is either claimed (possibly shared) or not. Because of the overall deadline, not every prize of sequence of prizes is accessible to a player and hence we must adapt the definition of *guarantee* to include the corresponding change in the nature of the *paranoid value*.

Definition 5.4.3

Player $\mathcal{X}$ **guarantees** prize i_1 at time t_0 if $d_{\mathcal{X}i_1} \leq \min\{d_{\mathcal{Y}i_1}, \lambda - t_0\}$ and **strictly guarantees** prize i_1 if $d_{\mathcal{X}i_1} < d_{\mathcal{Y}i_1}$ and $d_{\mathcal{X}i_1} \leq \lambda - t_0$. Also player $\mathcal{X}$ **guarantees** the sequence $\mathcal{X} \triangleright (i_1 \rightarrow i_2)$ at time t_0 if $d_{\mathcal{X}i_1} + d_{i_1 i_2} \leq \min\{d_{\mathcal{Y}i_2}, \lambda - t_0\}$ and **strictly guarantees** the sequence $\mathcal{X} \triangleright (i_1 \rightarrow i_2)$ if $d_{\mathcal{X}i_1} + d_{i_1 i_2} < d_{\mathcal{Y}i_2}$ and $d_{\mathcal{X}i_1} + d_{i_1 i_2} \leq \lambda - t_0$. □

The *static* cases of Section 5.1.1 are those for which $\Omega(\mathcal{A},\mathcal{B}) = \Gamma(\mathcal{A}) + \Gamma(\mathcal{B})$ and the *dynamic* cases of Section 5.1.3 are those for which $\Omega(\mathcal{A},\mathcal{B}) > \Gamma(\mathcal{A}) + \Gamma(\mathcal{B})$.

If only prize i_1 is accessible to player $\mathcal{X}$ then $\mathcal{X} \multimap i_1$ is certain. Hence there are two cases requiring analysis.

■ One prize inaccessible to one player

Suppose prize i_2 is accessible to both players but prize i_1 is accessible only to player $\mathcal{A}$. Then $\mathcal{B} \multimap i_2$ is certain and the concept of an **optimal** one-step move for player $\mathcal{A}$ is well defined. Also, $d_{\mathcal{A}i_1} < d_{\mathcal{B}i_1}$.

- If $d_{\mathcal{A}i_2} > d_{\mathcal{B}i_2}$ then $\mathcal{A} \multimap i_1$ is *optimal*.
- If $d_{\mathcal{A}i_2} = d_{\mathcal{B}i_2}$ then $\mathcal{A} \multimap \operatorname{argmax}\{v_{i_1}, \frac{1}{2}v_{i_2}\}$ is *optimal*.
- If $\mathcal{A} \triangleright (i_1 \rightarrow i_2)$ is accessible and $d_{\mathcal{A}i_1} + d_{i_1 i_2} < d_{\mathcal{B}i_2}$ then $\mathcal{A} \multimap i_1$ is *optimal*. Alternatively, if $\mathcal{A} \triangleright (i_2 \rightarrow i_1)$ is accessible and $d_{\mathcal{A}i_2} + d_{i_1 i_2} < d_{\mathcal{B}i_1}$ then $\mathcal{A} \multimap i_2$ is *optimal*.

- Suppose $d_{Ai_2} < d_{Bi_2}$, $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} \geq d_{Bi_1}$. Then $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is accessible but $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is not accessible and hence $\mathcal{A} \multimap \operatorname{argmax}\{v_{i_1}, \frac{1}{2}v_{i_2}\}$ is *optimal.*
- Suppose $d_{Ai_2} < d_{Bi_2}$, $d_{Ai_1} + d_{i_1 i_2} > d_{Bi_2}$ and $d_{Ai_2} + d_{i_1 i_2} \geq d_{Bi_1}$. Then $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is not accessible and $\mathcal{A} \multimap \operatorname{argmax}\{v_{i_1}, v_{i_2}\}$ is *optimal.*

■ Both prizes accessible to both players

Suppose now that both prizes are accessible to both players. Then $\Omega(\mathcal{A}, \mathcal{B}) = v_{i_1} + v_{i_2}$. We can see if changes are required to the analysis of Section 5.1. Certainly **cases (i)–(iii)** require no changes. Hence suppose that $d_{Ai_1} < d_{Bi_1}$ and $d_{Ai_2} < d_{Bi_2}$. The changes occur when one or both of the sequences $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ and $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ may become inaccessible to player $\mathcal{A}$.

Suppose neither sequence is accessible to player $\mathcal{A}$. Then $\mathcal{A} \multimap \operatorname{argmax}\{v_{i_1}, v_{i_2}\}$ is optimal but not necessarily critical. If $v_{i_1} \neq v_{i_2}$, then $\mathcal{B} \multimap \operatorname{argmin}\{v_{i_1}, v_{i_2}\}$ is optimal (but not necessarily critical) since Assumption 5.1.2 dictates that player $\mathcal{A}$ will eventually abandon prize $\operatorname{argmin}\{v_{i_1}, v_{i_2}\}$. However, if $v_{i_1} = v_{i_2}$, then Assumption 5.1.2 cannot help player $\mathcal{B}$ since player $\mathcal{A}$ can possibly maintain both prize constraints, until either $\mathcal{A} \multimap i_1$ or $\mathcal{A} \multimap i_2$ is *critical*, before deciding which prize to claim, at which time at least one of the prizes may no longer be accessible to player $\mathcal{B}$. *Hence this case is unresolved for player $\mathcal{B}$.*

Suppose $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is accessible but $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is not accessible.

- Suppose $d_{Ai_1} + d_{i_1 i_2} < d_{Bi_2}$. Then $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is *guaranteed.*
- Suppose $d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2}$. Then $\mathcal{B} \multimap i_2$ is critical since otherwise, if $\mathcal{A} \multimap i_1$, the resulting outcome is $v_{i_1} + v_{i_2}$ to player $\mathcal{A}$. If $v_{i_1} > \frac{1}{2}v_{i_2}$ then $\mathcal{A} \multimap i_1$ is critical since otherwise, if $\mathcal{B} \multimap i_1$, the resulting outcome is $v_{i_2} < v_{i_1} + \frac{1}{2}v_{i_2}$. If $v_{i_1} \leq \frac{1}{2}v_{i_2}$ and $\mathcal{A} \multimap i_2$ is not critical then *this case is unresolved for player $\mathcal{B}$.*
- Suppose $d_{Ai_1} + d_{i_1 i_2} > d_{Bi_2}$. *This case is unresolved for both player $\mathcal{A}$ and player $\mathcal{B}$.*

Suppose both sequences are accessible to player $\mathcal{A}$. *This case is also unresolved for both player $\mathcal{A}$ and player $\mathcal{B}$.*

5.4.2 Revision of Dynamic Cases

We have identified a number of cases for which a critical one-step move is evident for at least one player. For those *unresolved* dynamic cases in which both prizes are accessible to both players and a critical one-step move is not evident for at least one player, we proceed to update the previous analysis of the dynamic case of the two prize problem to account for $\lambda < \infty$. We begin with *median feasibility.*

5.4.2.1 Median Feasibility (Revision)

We need to determine if $\exists$ an **accessible median location,** Z, which is a location on the line through the two players, such that constraints (5.48)–(5.51) hold.

$$d_{Ai_1} + d_{Zi_1} \geq d_{BZ} \tag{5.48}$$

$$d_{Ai_2} + d_{Zi_2} \geq d_{BZ} \quad (5.49)$$
$$d_{BZ} + d_{Zi_1} \leq \lambda - t_B \quad (5.50)$$
$$d_{BZ} + d_{Zi_2} \leq \lambda - t_B \quad (5.51)$$

Constraints (5.48)–(5.49) are equivalent to constraints (5.10)–(5.11) of Section 5.1.4.

Adopting the notation of Section 5.1.4, constraints (5.12)–(5.13) must hold. Additionally we must have $d_{BZ} \leq \lambda - t_B$, $d_{Zi_1} \leq \lambda - t_B$ (i.e. $z \leq \lambda - t_B$) and $d_{Zi_2} \leq \lambda - t_B$ (i.e. $z \geq d_{i_1 i_2} - (\lambda - t_B)$). Each of these places a constraint on z. Since $d_{Zi_1} \leq \lambda - t_B$ then necessarily $x < \lambda - t_B$. Also, since $d_{Zi_2} \leq \lambda - t_B$ then necessarily $d_{i_1 i_2} - x < \lambda - t_B$. Then

$$\begin{aligned}
& d_{BZ} + d_{Zi_1} \leq \lambda - t_B \\
\Leftrightarrow\ & \sqrt{(x-z)^2 + y^2} + z \leq \lambda - t_B \\
\Leftrightarrow\ & x^2 - 2xz + z^2 + y^2 \leq (\lambda - t_B)^2 - 2z(\lambda - t_B) + z^2 \\
\Leftrightarrow\ & 2z(\lambda - t_B - x) \leq (\lambda - t_B)^2 - d_{BZ}^2 \\
\Leftrightarrow\ & z \leq \frac{(\lambda - t_B)^2 - d_{BZ}^2}{2(\lambda - t_B - x)}
\end{aligned}$$

Also

$$\begin{aligned}
& d_{BZ} + d_{Zi_2} \leq \lambda - t_B \\
\Leftrightarrow\ & \sqrt{(x-z)^2 + y^2} + d_{i_1 i_2} - z \leq \lambda - t_B \\
\Leftrightarrow\ & x^2 - 2xz + z^2 + y^2 \leq (\lambda - t_B - d_{i_1 i_2} + z)^2 \\
\Leftrightarrow\ & d_{Bi_1}^2 - (\lambda - t_B - d_{i_1 i_2})^2 \leq 2z(x + \lambda - t_B - d_{i_1 i_2}) \\
\Leftrightarrow\ & z \geq \frac{d_{Bi_1}^2 - (\lambda - t_B - d_{i_1 i_2})^2}{2(x + \lambda - t_B - d_{i_1 i_2})}
\end{aligned}$$

Summarising the applicable constraints, let

$$z_1 = \max\left\{\frac{d_{B1}^2 - d_{A1}^2}{2(x + d_{A1})}, \frac{d_{Bi_1}^2 - (\lambda - t_B - d_{i_1 i_2})^2}{2(x + \lambda - t_B - d_{i_1 i_2})}, d_{i_1 i_2} - \lambda + t_B)\right\}$$

$$z_2 = \min\left\{\frac{(d_{A2} + d_{12})^2 - d_{B1}^2}{2(d_{A2} + d_{12} - x)}, \frac{(\lambda - t_B)^2 - d_{BZ}^2}{2(\lambda - t_B - x)}, \lambda - t_B\right\}$$

Definition 5.4.4

$\mathcal{B} \rhd \{i_1, i_2\}$ is **$\mathcal{B}$-median-accessible** if $z_1 \leq z_2$. □

5.4.2.2 Window Feasibility Subproblem (Revision)

Suppose $t_B + d_{Bi_1} \leq \lambda$ and $t_B + d_{Bi_2} \leq \lambda$. The definitions of *accessible* involving t_A and t_B are analogous to those of Definition 5.4.2. We wish to see if player $\mathcal{B}$ can ensure additionally that each prize is accessible to player $\mathcal{B}$ at time $t_A > t_B$. This requires that the location Z satisfies constraints (5.52)–(5.53).

$$d_{Zi_1} \leq \lambda - t_A \quad (5.52)$$
$$d_{Zi_2} \leq \lambda - t_A \quad (5.53)$$

Also, $\mathcal{A} \rhd (i_2 \rightarrow i_1)$ is accessible if and only if $t_A + d_{Ai_2} + d_{i_1 i_2} \leq \lambda$, i.e., $d_{Ai_2} + d_{i_1 i_2} \leq \lambda - t_A$. Similarly, $\mathcal{A} \rhd (i_1 \rightarrow i_2)$ is accessible if and only if $d_{Ai_1} + d_{i_1 i_2} \leq \lambda - t_A$. Hence constraints (5.52)–(5.53) can *only* further restrict the feasibility window. Finally we redefine r_1, r_2 and r_3.

$$r_1 = \min\{d_{Ai_2} + d_{i_1 i_2}, \lambda - t_A\} \quad (5.54)$$
$$r_2 = \min\{d_{Ai_1} + d_{i_1 i_2}, \lambda - t_A\} \quad (5.55)$$
$$r_3 = t_A - t_B \quad (5.56)$$

5.4.2.3 Player $\mathcal{A}$ Accessibility Window

At some time t_0, suppose that both $\mathcal{A} \triangleright (i_1 \rightarrow i_2)$ and $\mathcal{A} \triangleright (i_2 \rightarrow i_1)$ are accessible to player $\mathcal{A}$. We wish to determine a location, Z, for player $\mathcal{A}$ at time $t_0 + \Delta$ such that both sequences definitely remain accessible to player $\mathcal{A}$ at time $t_0 + \Delta$. Then such a location Z must satisfy constraints (5.57)–(5.59).

$$\Delta \geq d_{AZ} \quad (5.57)$$
$$\lambda - (t_0 + \Delta) \geq d_{Zi_1} + d_{i_1 i_2} \quad (5.58)$$
$$\lambda - (t_0 + \Delta) \geq d_{Zi_2} + d_{i_1 i_2} \quad (5.59)$$

This is a *window feasibility subproblem* as in Section 5.2. We also wish to apply constraints (5.37)–(5.38) to ensure that Z is also *safe*. Hence apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the prizes i_1 and i_2; location Y is the location of player $\mathcal{A}$ and

$$r_1 = \min\{\lambda - (t_0 + \Delta) - d_{i_1 i_2}, d_{Bi_1} - \Delta\}$$
$$r_2 = \min\{\lambda - (t_0 + \Delta) - d_{i_1 i_2}, d_{Bi_2} - \Delta\}$$
$$r_3 = \Delta.$$

If such a location Z exists then this game position is **$\mathcal{A}$-accessible** and the corresponding feasibility window is called the **$\mathcal{A}$-accessibility-window**.

5.4.2.4 Player $\mathcal{B}$ Accessibility Window

At some time t_0, suppose the both prizes are accessible to player $\mathcal{B}$. We wish to determine a location, Z, for player $\mathcal{B}$ at time $t_0 + \Delta$ such that both prizes definitely remain accessible to player $\mathcal{B}$ at time $t_0 + \Delta$. Then such a location Z must satisfy constraints (5.60)–(5.62).

$$\Delta \geq d_{BZ} \quad (5.60)$$
$$\lambda - (t_0 + \Delta) \geq d_{Zi_1} \quad (5.61)$$
$$\lambda - (t_0 + \Delta) \geq d_{Zi_2} \quad (5.62)$$

This is a *window feasibility subproblem* as in Section 5.2. We also wish to apply constraints (5.40)–(5.41) to ensure that Z is also *safe*. Hence apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the prizes i_1 and i_2; location Y is the location of player $\mathcal{B}$ and

$$r_1 = \min\{\lambda - (t_0 + \Delta), d_{Ai_2} - \Delta + d_{i_1 i_2}\}$$
$$r_2 = \min\{\lambda - (t_0 + \Delta), d_{Ai_2} - \Delta + d_{i_1 i_2}\}$$
$$r_3 = \Delta.$$

If such a location Z exists then this game position is **$\mathcal{B}$-accessible** and the corresponding feasibility window is called the **$\mathcal{B}$-accessibility-window**.

5.4.3 Deadline Strategy Framework

When a critical one-step move is not evident for a player then we propose that that player adopt the *framework* of Section 5.3.3. However, we adapt the construction of the *target window* to reflect *accessibility*.

If the game position is $\mathcal{A}$-accessible then the $\mathcal{A}$-target-window is the $\mathcal{A}$-accessibility-window; otherwise the $\mathcal{A}$-target-window is the default window.

Suppose the game position is $\mathcal{B}$-median-accessible. Then the $\mathcal{B}$-target-window is the $\mathcal{B}$-median-accessible-window. Suppose the game position is not $\mathcal{B}$-median-accessible. If the game position is $\mathcal{B}$-accessible then the $\mathcal{B}$-target-window is the $\mathcal{B}$-accessibility-window; otherwise the $\mathcal{B}$-target-window is the default window.

Finally, we directly carry over the selection of a *target location* from each of the two prize strategies presented in Section 5.3. In particular, note that Algorithm 5.17 TWO-PRIZE LAST-RESORT MEDIAN still applies.

5.5 Tiny Tournament

Axelrod [7] conducted two tournaments for the iterated Prisoner's Dilemma (see Section 2.4.5) in which each participant submitted a strategy and the tournament consisted of playing each strategy off against each other strategy, with the overall scores of each game aggregated to determine an overall ranking of the strategies.

We propose a similar **Tiny Tournament** for the two prize problem, comprising the two-prize strategies designed in Section 5.3. Since we have already determined an optimal one-step strategy for each player in each of the static cases, we need only consider test problem instances drawn from the dynamic cases, i.e., those satisfying the prize and pair constraints (5.1)–(5.4). In each case there is a well defined ahead player and a well defined behind player. Therefore, without loss of generality, we compare the set of strategies for player $\mathcal{A}$ against the set of strategies for player $\mathcal{B}$.

5.5.1 Experimental Aims

Let $\mathbb{S}_{A\infty}$ be the infinite set of all possible strategies for player $\mathcal{A}$ for the two prize CPCP and let $\mathbb{S}_{B\infty}$ be the infinite set of all possible strategies for player $\mathcal{B}$ for the two prize CPCP. The **expected value of the game** $\wp$ (or the **expected value of the problem instance** $\wp$), $v^*_A(\wp)$, is defined as the *Nash equilibrium* value (in mixed strategies) to player $\mathcal{A}$ of the infinite two player game in which player $\mathcal{A}$'s pure strategies are $\mathbb{S}_{A\infty}$ and player $\mathcal{B}$'s pure strategies are $\mathbb{S}_{B\infty}$ and the payoff to player $\mathcal{A}$, when player $\mathcal{A}$ selects the pure strategy $\mathsf{a} \in \mathbb{S}_{A\infty}$ and player $\mathcal{B}$ selects the pure strategy $\mathsf{b} \in \mathbb{S}_{B\infty}$, is the total prize value claimed by player $\mathcal{A}$ in the corresponding simulation on problem instance $\wp$, if such a Nash equilibrium exists. If $\lambda = \infty$, then $v^*_A(\wp)$ exists since the game is a constant-sum game.

The purpose of the tiny tournament is to test the following conjecture.

Conjecture 5.5.1

$$v_B^*(\wp) = \Omega(\wp) - \Gamma_A(\wp) \tag{5.63}$$

□

If Conjecture 5.5.1 is true, for a large proportion of problem instances, then we would be able to conclude that $\Omega(\wp) - \Gamma_A(\wp)$ is a good estimate of the expected value, $v_B^*(\wp)$, of the dynamic two prize problem scenario for player $\mathcal{B}$ and, hence, that $\Gamma_A(\wp)$ is a good estimate of the expected value, $v_A^*(\wp)$, of the dynamic two prize problem scenario for player $\mathcal{A}$. Since player $\mathcal{A}$ can always select PRIZE-GUARANTEE as its two-prize strategy, $v_A^*(\wp) \geq \Gamma_A(\wp)$. Also, $\Omega(\wp) \geq v_A^*(\wp) + v_B^*(\wp)$ and, hence, $v_B^*(\wp) \leq \Omega(\wp) - \Gamma_A(\wp)$.

The necessity for investigating Conjecture 5.5.1 is to determine whether it is reasonable to *expect* some return to player $\mathcal{B}$ from a dynamic two prize problem scenario. If so, then this can easily be incorporated into the evaluation of a proposed scenario involving a two prize subproblem in the context of a problem with more than two prizes.

5.5.2 Measures of Performance

Let $\mathbb{S}_A$ be a given finite set of player $\mathcal{A}$ strategies and let $\mathbb{S}_B$ be a given finite set of player $\mathcal{B}$ strategies. A **computational tournament** plays off every player $\mathcal{A}$ strategy against each player $\mathcal{B}$ strategy on a number of representative problem instances. The **computational maximin value**, $v_B^\diamond(\wp)$, of the computational tournament results corresponding to problem instance $\wp$, is defined as the best result for player $\mathcal{B}$ given that player $\mathcal{A}$ selects the best opposing strategy for problem instance $\wp$; $v_B^\diamond(\wp)$ represents an estimate of $v_B^*(\wp)$.

A **simulation battle** between an $\mathcal{A}$-strategy and a $\mathcal{B}$-strategy is a complete play of the CPCP on a given problem instance. We assume that once the game state becomes *static* the players play optimally from that point onwards. For each battle between an $\mathcal{A}$-strategy and a $\mathcal{B}$-strategy we collect the outcome for each player as in Table 5.2, up to equivalence of outcome. When at least one *stochastic* strategy is involved, we take the worst result for each player over 100 simulation battles.

Note that $\Omega(\wp) \geq v_{i_1}$ and $\Gamma_A(\wp) \geq v_{i_1}$. Let $v_A(\wp)$ be the value claimed by player $\mathcal{A}$ and $v_B(\wp)$ be the value claimed by player $\mathcal{B}$ in a single battle on the problem instance $\wp$. Then assign to player $\mathcal{A}$ the score $v_A(\wp) - \Gamma_A(\wp)$ and assign to player $\mathcal{B}$ the score $v_B(\wp) - \Omega(\wp) + \Gamma_A(\wp)$.

Timing for the tiny tournament is not significant since each strategy is $O(1)$ computational complexity per step, and we are not trying to minimize the number of steps required.

5.5.3 Experimental Design

The tiny tournament is conducted on a sample of problem instances, representative of the following two-prize problem classes, between the two-prize strategies proposed in previous sections of this chapter.

5.5.3.1 Participants

The participant player strategies in the tiny tournament are those listed in Table 5.1. The parameters of strategies **A4**, **A7**, **A8**, **A9**, **B4** and **B8** have been conservatively tuned prior to the tournament.

5.5.3.2 The Class of Tiny Problem Instances

We classify the two prize problem instance according to the prize and player locations, and the prize values.

Classification of Prize and Player Locations

Whereas Axelrod [7] required only a single problem instance for the prisoner's dilemma tournament, we require a representative set of problem instances. We propose the following seven subclasses of problems. For subclasses (3)–(7) below, suppose that the pair constraints (5.3)–(5.4) hold with strict inequality, and recall the definitions of '$\mathcal{A}$-accessible', '$\mathcal{B}$-accessible' and '$\mathcal{B}$-median-accessible' from Section 5.4.2. Since the LAST RESORT MEDIAN strategy is optimal for player $\mathcal{B}$ in the case where $\min\{d_{Ai_1}, d_{Ai_2}\} \leq \Delta$, we assume that $\min\{d_{Ai_1}, d_{Ai_2}\} > \Delta$.

(1). Constraints (5.3)–(5.4) hold with equality.

(2). One of constraint (5.3)–(5.4) holds with equality and the other holds with strict inequality.

(3). $\mathcal{A}$-accessible and $\mathcal{B}$-median-accessible.

(4). $\mathcal{A}$-accessible and $\mathcal{B}$-accessible but not $\mathcal{B}$-median-accessible.

(5). $\mathcal{A}$-accessible and not $\mathcal{B}$-accessible.

(6). Not $\mathcal{A}$-accessible and $\mathcal{B}$-median-accessible.

(7). Not $\mathcal{A}$-accessible and $\mathcal{B}$-accessible but not $\mathcal{B}$-median-accessible.

Lemma 5.5.2

There can be no class "not $\mathcal{A}$-accessible and not $\mathcal{B}$-accessible" since these are mutually exclusive.

Proof:

Suppose there exists locations for player $\mathcal{A}$ and player $\mathcal{B}$ which satisfy constraints (5.3)–(5.4) with strict inequality and which are neither $\mathcal{A}$-accessible nor $\mathcal{B}$-accessible.

The locations are not $\mathcal{A}$-accessible only if constraints (5.64)–(5.65) hold:

$$d_{Bi_1} - \Delta < d_{Ai_1} < d_{Bi_1} \tag{5.64}$$
$$d_{Bi_2} - \Delta < d_{Ai_2} < d_{Bi_2} \tag{5.65}$$

The locations are not $\mathcal{B}$-accessible only if constraints (5.66)–(5.67) hold:

$$d_{Ai_1} + d_{i_1 i_2} - \Delta < d_{Bi_2} < d_{Ai_1} + d_{i_1 i_2} \tag{5.66}$$
$$d_{Ai_2} + d_{i_1 i_2} - \Delta < d_{Bi_1} < d_{Ai_2} + d_{i_1 i_2} \tag{5.67}$$

Table 5.1: Tiny Tournament: Participating Strategies

Identifier	Name	Algorithm	Type
G1	GREEDY	3.x	**D**
G2	NEAREST NEIGHBOUR	3.x	**D**
G3	FARTHEST NEIGHBOUR	3.x	**D**
G4	OPPONENT NEAREST NEIGHBOUR	3.x	**D**
G5	OPPONENT FARTHEST NEIGHBOUR	3.x	**D**
G6	MAX PRIZE	3.x	**D**
G7	MIN PRIZE	3.x	**D**
G8	RANDOM PRIZE	5.15	**S**
A1	AHEAD PURE-ATTACK	5.2	**D**
A2	AHEAD CEILING-ATTACK	5.6	**D**
A3	AHEAD PURE-DEFEND	5.3	**D**
A4	AHEAD THRESH-DEFEND	5.8	**DP**
A5	TIT FOR TAT AHEAD (BEARING)	5.12	**D**
A6	TIT FOR TAT AHEAD (LOCATION)	5.13	**D**
A7	TIT FOR TWO TATS AHEAD (BEARING)	5.14	**DP**
A8	TIT FOR TWO TATS AHEAD (LOCATION)	5.14	**DP**
B1	BEHIND PURE-ATTACK	5.4	**D**
B2	BEHIND CEILING-ATTACK	5.7	**D**
B3	BEHIND PURE-DEFEND	5.5	**D**
B4	BEHIND THRESH-DEFEND	5.9	**DP**
B5	TIT FOR TAT BEHIND (BEARING)	5.10	**D**
B6	TIT FOR TAT BEHIND (LOCATION)	5.11	**D**
B7	PREVIOUS-MEDIAN	5.16	**D**

Key

Code	Explanation
G	Applicable to both players
A	Player $\mathcal{A}$ only
B	Player $\mathcal{B}$ only
D	Deterministic
S	Stochastic
P	Parameterised

Table 5.2: Possible Outcomes of Two Prize Problem

Code	$\mathcal{A}$-Value	Description
(8)	$v_{i_1} + v_{i_2}$	Outright
(7)	$v_{i_1} + \frac{1}{2}v_{i_2}$	Large plus half small
(6)	v_{i_1}	Large
(5)	$\frac{1}{2}v_{i_1} + v_{i_2}$	Small plus half large
(4)	$\frac{1}{2}v_{i_1} + \frac{1}{2}v_{i_2}$	Half both
(3)	$\frac{1}{2}v_{i_1}$	Half large
(2)	v_{i_2}	Small
(1)	$\frac{1}{2}v_{i_2}$	Half small
(0)	0	Nil

Hence,

$$\begin{aligned} d_{Ai_1} + d_{i_1 i_2} - \Delta &< d_{Bi_2} < d_{Ai_2} + \Delta \\ d_{Ai_2} + d_{i_1 i_2} - \Delta &< d_{Bi_1} < d_{Ai_1} + \Delta \end{aligned}$$

and, in particular,

$$d_{Ai_1} + 2d_{i_1 i_2} < d_{Ai_2} + d_{i_1 i_2} + 2\Delta < d_{Ai_1} + 4\Delta$$

i.e., $d_{i_1 i_2} < 2\Delta$. However, this is not possible by Assumption 3.1.3. ■

Classification of Prize Values

Suppose $v_{i_1} \geq v_{i_2}$. If $v_{i_1} > v_{i_2}$ then we call prize i_1 the *large* prize, since it is the most valuable, and call prize i_2 the *small* prize, since it is least valuable. If $v_{i_1} = v_{i_2}$ then the two prizes are indistinguishable by value. There are four possible preference relations amongst the resulting outcomes displayed in Table 5.2.

TWO-(a). If $v_{i_1} > 2v_{i_2}$ then (8) $\succ$ (7) $\succ$ (6) $\succ$ (5) $\succ$ (4) $\succ$ (3) $\succ$ (2) $\succ$ (1) $\succ$ (0).

TWO-(b). If $v_{i_1} = 2v_{i_2}$ then (8) $\succ$ (7) $\succ$ (6) $\equiv$ (5) $\succ$ (4) $\succ$ (3) $\equiv$ (2) $\succ$ (1) $\succ$ (0).

TWO-(c). If $v_{i_2} < v_{i_1} < 2v_{i_2}$ then (8) $\succ$ (7) $\succ$ (5) $\succ$ (6) $\succ$ (4) $\succ$ (2) $\succ$ (3) $\succ$ (1) $\succ$ (0).

TWO-(d). If $v_{i_1} = v_{i_2}$ then (8) $\succ$ (7) $\equiv$ (5) $\succ$ (6) $\equiv$ (4) $\equiv$ (2) $\succ$ (3) $\equiv$ (1) $\succ$ (0).

Tiny Problem Class Definition

We require that $\lambda \geq \max\{d_{Bi_1}, d_{Bi_2}\}$ since, otherwise, player $\mathcal{B}$ either has at most one single-step option. This is sufficient to also ensure that $\Omega = v_{i_1} + v_{i_2}$. Let $\delta = d_{i_1 i_2} + \max\{d_{Ai_1}, d_{Ai_2}\} - \min\{d_{Bi_1}, d_{Bi_2}\}$. Then let the overall deadline $\lambda \sim \min\{d_{Bi_1}, d_{Bi_2}\} + U(0, 2\delta)$.

The tiny problem class consists of the eight location subclasses crossed with the four prize value subclasses to make 32 subclasses in total. Each subclass is defined by a descriptor such as '**T-5d**' which indicates a tiny problem subclass of location subclass '**5**' and prize value subclass '**d**'.

Finally, let $\Delta = 0.01$ and ensure that $d_{i_1 i_2} \geq 2\Delta$, in accordance with Assumption 3.1.3.

Representative Prize Values

Standardise the total prize value $v_{i_1} + v_{i_2} = 100$. For prize value subclass (b), $v_{i_1} = 2v_{i_2}$ implies $v_{i_1} = \frac{200}{3}$ and $v_{i_2} = \frac{100}{3}$. Similarly for prize value subclass (d), $v_{i_1} = v_{i_2} = 50$. For comparability of battles within prize value subclasses (a) and (c), we select representative values for v_{i_1} and v_{i_2}. For subclass (a), select $v_{i_1} = 80$ and $v_{i_1} = 20$, and for subclass (c), select $v_{i_1} = 60$ and $v_{i_2} = 40$.

5.5.4 Results and Analysis

We can now present the results from the tiny tournament, by problem subclass and player strategy.

▼ Results by Tiny Problem Subclass

For each tiny problem subclass we wish to determine which strategies were most successful both in the worst case and in the average case. For battles involving a stochastic strategy we conduct 100 trial battles. Both the minimum score and mean score of each player contribute to the corresponding overall result.

Table 5.3 records the results by tiny problem subclass, for 1000 problem instances in each subclass. The best MIN $\mathcal{A}$-strategy and best MIN $\mathcal{B}$-strategy columns define the *robustness* of the best player strategies on each subclass. With the exception of T-(1a), T-(1b), T-(7a), and T-(7b), no strategy, for either player, is robust on a subclass. The main interest, however, is the in the best MEAN $\mathcal{B}$-strategy, for which a positive result supports Conjecture 5.5.1, and a negative result counts against the conjecture. Problem subclass T-(5) is the most difficult for player $\mathcal{B}$. This is expected since this subclass is $\mathcal{A}$-accessible but not $\mathcal{B}$-accessible. Of the other subclasses, the balance of results support the conjecture to a degree sufficient that it gives a tenable approximation.

▼ Results by Two Prize Strategy

We wish to determine the performance of each two-prize strategy. For each strategy, we determine which subclass was its most successful, and which subclass was its least successful. Over the entire tiny problem class, we determine which opponent was its nemesis, and also calculate an overall MEAN performance. Tables 5.4 and 5.5 gives results by the player strategies.

With respect to the worst MIN subclass and the worst MEAN subclass, no strategy is robust. As expected, **G6** MAX PRIZE is the overall most effective strategy for player $\mathcal{A}$. This indicates that player $\mathcal{A}$ should commit early to only target-prize. For player $\mathcal{B}$, however, several strategies stand out as the overall most effective: **G3** FARTHEST NEIGHBOUR, **G5** OPPONENT FARTHEST NEIGHBOUR, **G8** RANDOM PRIZE, and **B7** PREVIOUS-MEDIAN. The first two of these reflect the attempt to target the the prize which the opponent will not target. The presence of RANDOM PRIZE as a best strategy indicates that confusion of an opposing strategy is effective. The last of these best strategies, PREVIOUS-MEDIAN, indicates that attempting to force the opponent to commit to one target-prize is also effective.

Table 5.3: Tiny Tournament: Results by Tiny Subclass

Subclass	Best $\mathcal{A}$-strategy				Best $\mathcal{B}$-strategy			
	MIN		MEAN		MIN		MEAN	
T-(1a)	0.0	**G1**	0.3	**G1**	−50.0	**G1**	−0.3	**G1**
T-(1b)	0.0	**G1**	0.2	**G1**	−50.0	**G1**	−0.2	**G1**
T-(1c)	−20.0	**G1**	0.2	**G1**	−40.0	**G1**	−0.2	**G1**
T-(1d)	−25.0	**G1**	8.9	**A5**	−50.0	**G1**	−9.4	**G4**
T-(2a)	−40.0	**G2**	11.4	**G6**	−50.0	**G1**	3.4	**G3**
T-(2b)	−33.3	**G2**	−2.1	**G6**	−50.0	**G1**	9.2	**G3**
T-(2c)	−30.0	**G1**	0.1	**G1**	−40.0	**G1**	−0.1	**G1**
T-(2d)	−25.0	**G1**	0.1	**G3**	−25.0	**G1**	−0.1	**B7**
T-(3a)	−60.0	**G7**	−2.1	**A5**	−20.0	**G1**	23.9	**B7**
T-(3b)	−33.3	**G7**	−1.1	**A5**	−33.3	**G1**	13.4	**B7**
T-(3c)	−30.0	**G7**	−0.3	**A5**	−40.0	**G1**	7.8	**B5**
T-(3d)	−25.0	**G1**	1.2	**A1**	−50.0	**G1**	−0.1	**B5**
T-(4a)	−60.0	**G7**	0.9	**G6**	−20.0	**G5**	9.9	**G3**
T-(4b)	−33.3	**G7**	2.1	**G6**	−33.3	**G1**	4.4	**G7**
T-(4c)	−20.0	**A5**	3.3	**A4**	−40.0	**G1**	0.6	**G7**
T-(4d)	−25.0	**G1**	7.2	**A1**	−50.0	**G1**	−0.4	**G5**
T-(5a)	−60.0	**G1**	10.7	**A4**	−50.0	**G1**	−9.9	**G7**
T-(5b)	−33.3	**G1**	16.0	**A1**	−50.0	**G1**	−14.6	**G7**
T-(5c)	−20.0	**G1**	17.7	**A1**	−40.0	**G1**	−15.0	**G7**
T-(5d)	−25.0	**G1**	22.1	**A1**	−50.0	**G1**	−13.4	**G6**
T-(6a)	−60.0	**G1**	0.2	**G6**	−40.0	**G8**	33.1	**G6**
T-(6b)	−16.7	**G6**	0.2	**G6**	−33.3	**G8**	17.5	**G6**
T-(6c)	−20.0	**G6**	0.1	**G6**	−40.0	**G1**	9.7	**G6**
T-(6d)	−25.0	**G1**	0.0	**G8**	−50.0	**G3**	0.1	**G8**
T-(7a)	0.0	**G6**	0.4	**G6**	−20.0	**G1**	25.7	**G6**
T-(7b)	0.0	**G6**	0.3	**G6**	−33.3	**G6**	11.8	**B7**
T-(7c)	−10.0	**G6**	0.2	**G6**	−50.0	**B1**	5.9	**B7**
T-(7d)	−25.0	**G1**	0.1	**G8**	−50.0	**G1**	−3.7	**B7**

Table 5.4: Tiny Tournament: Results by Player $\mathcal{A}$ Strategy

Strategy	Best Subclass		Worst Subclass		Nemesis				Overall	
	MIN	MEAN	MIN	MEAN	MIN		MEAN		MEAN	STD DEV
G1	0.0 (1a)	16.3 (5d)	−70.0 (3a)	−13.3 (2b)	−70.0	**G2**	−1.5	**G7**	13.4	52.5
G2	0.0 (1a)	16.3 (5d)	−70.0 (3a)	−28.0 (7a)	−70.0	**G2**	−8.2	**G5**	−73.8	36.9
G3	0.0 (1a)	16.7 (5d)	−70.0 (3a)	−29.4 (6a)	−70.0	**G2**	−9.1	**G6**	−69.5	34.1
G4	0.0 (1a)	18.2 (5d)	−70.0 (3a)	−28.1 (7a)	−70.0	**G2**	−8.1	**G3**	−66.5	44.4
G5	0.0 (1a)	16.3 (5d)	−70.0 (3a)	−29.4 (6a)	−70.0	**G2**	−8.4	**G6**	−71.6	37.6
G6	0.0 (1a)	16.3 (5d)	−70.0 (3a)	−2.3 (3a)	−70.0	**G2**	0.6	**G7**	45.1	32.0
G7	0.0 (1a)	16.3 (5d)	−70.0 (6a)	−56.4 (7a)	−70.0	**G2**	−14.7	**G6**	−178.4	36.0
G8	0.0 (1a)	16.8 (5d)	−70.0 (3a)	−29.3 (7a)	−70.0	**G2**	−9.7	**G6**	−77.8	21.1
A1	0.0 (1a)	22.1 (5d)	−70.0 (3a)	−28.1 (7a)	−70.0	**G2**	−7.0	**G5**	−69.1	48.4
A2	0.0 (1a)	11.3 (5d)	−70.0 (3a)	−29.4 (6a)	−70.0	**G2**	−9.1	**G6**	−78.4	37.2
A3	0.0 (1a)	17.4 (5d)	−80.0 (6a)	−27.6 (7a)	−80.0	**B7**	−5.6	**G3**	−39.8	59.4
A4	0.0 (1a)	22.1 (5d)	−80.0 (6a)	−27.6 (7a)	−80.0	**B7**	−4.9	**G6**	−20.4	70.9
A5	0.0 (1a)	17.5 (5d)	−80.0 (3a)	−23.7 (7a)	−80.0	**G1**	−2.0	**B2**	−5.0	61.5
A6	0.0 (1a)	17.7 (5d)	−80.0 (2a)	−29.3 (7a)	−80.0	**G1**	−9.0	**G7**	−83.5	23.1
A7	0.0 (1a)	17.5 (5d)	−80.0 (3a)	−23.7 (7a)	−80.0	**G1**	−2.0	**G8**	−5.8	61.7
A8	0.0 (1a)	17.8 (5d)	−80.0 (2a)	−29.3 (7a)	−80.0	**G1**	−8.6	**G7**	−80.0	32.0

Table 5.5: Tiny Tournament: Results by Player $\mathcal{B}$ Strategy

Strategy	Best Subclass				Worst Subclass				Nemesis				Overall	
	MIN		MEAN		MIN		MEAN		MIN		MEAN		MEAN	STD DEV
G1	−20.0	(3a)	32.0	(6a)	−75.0	(6d)	−27.6	(5c)	−75.0	**G1**	−9.7	**G6**	−21.4	94.4
G2	−20.0	(3a)	22.8	(6a)	−75.0	(6d)	−22.2	(5c)	−75.0	**G1**	−6.4	**G6**	−29.1	82.6
G3	−20.0	(3a)	25.5	(6a)	−75.0	(7d)	−20.8	(5d)	−75.0	**A2**	−4.6	**G6**	29.6	82.3
G4	−20.0	(3a)	22.9	(6a)	−75.0	(6d)	−23.0	(5d)	−75.0	**G1**	−9.0	**G6**	−24.5	86.9
G5	−20.0	(3a)	25.5	(6a)	−75.0	(7d)	−22.2	(5c)	−75.0	**A6**	−3.1	**G6**	27.3	80.9
G6	−20.0	(3a)	33.1	(6a)	−75.0	(6d)	−27.7	(5c)	−75.0	**G1**	−9.9	**G6**	−15.8	96.3
G7	−20.0	(3a)	18.7	(3a)	−75.0	(6d)	−19.1	(7d)	−75.0	**G1**	−5.1	**A3**	−6.4	76.6
G8	−20.0	(3a)	26.6	(6a)	−66.7	(2b)	−22.0	(5c)	−66.7	**G1**	−5.4	**G6**	28.9	82.1
B1	−20.0	(3a)	18.0	(6a)	−70.0	(6c)	−22.2	(5c)	−70.0	**G2**	−14.3	**A3**	−44.7	73.4
B2	−20.0	(3a)	26.8	(6a)	−75.0	(6d)	−22.9	(5c)	−75.0	**G1**	−7.7	**G6**	15.2	86.4
B3	−20.0	(3a)	22.9	(6a)	−75.0	(6d)	−22.9	(5c)	−75.0	**G1**	−7.7	**G6**	−26.2	85.5
B4	−20.0	(3a)	18.1	(6a)	−70.0	(6c)	−22.9	(5c)	−70.0	**G2**	−14.2	**A4**	−45.9	73.9
B5	−20.0	(3a)	26.8	(6a)	−66.7	(2b)	−23.8	(5c)	−66.7	**G1**	−5.5	**G6**	23.2	86.5
B6	−20.0	(3a)	21.5	(6a)	−75.0	(6d)	−21.6	(5c)	−75.0	**G1**	−5.2	**A4**	−19.4	82.7
B7	−20.0	(3a)	27.0	(6a)	−75.0	(7d)	−24.0	(5c)	−75.0	**G1**	−5.0	**G6**	28.4	83.3

5.5.5 Conclusions from the Tiny Tournament

The tiny tournament has generated sufficient empirical evidence to suggest that Conjecture 5.5.1 holds on average, although not for problem subclass T-(5). Overall, however, we may conclude that the conjecture is a tenable approximation when only the prize and pair constraints are used to characterise a the two prize subproblem within a larger context.

The most effective strategy for player $\mathcal{A}$ is to commit early to the maximum value prize. The most effective strategy for player $\mathcal{B}$ appears to be either to commit to the opposite prize from the player $\mathcal{A}$ (if player $\mathcal{A}$ commits early), or to play PREVIOUS-MEDIAN otherwise.

5.6 Tactical Approach to Three Prize Problem

The purpose of this section is to introduce a tactical game theoretic approach to the Three Prize Problem, which is then generalized to larger problems throughout the following chapters. Initially, we must address the following questions.

(i). How should player $\mathcal{B}$ play in the scenario $\mathcal{B} \rhd \{i_1, i_2\}$, given that $\mathcal{A} \rhd i_1$?

(ii). How should player $\mathcal{B}$ play in the window-feasible scenario $\mathcal{A} \rhd i_3$ and $\mathcal{B} \rhd \{i_1, i_2\}$?

(iii). How can each player estimate the expected reward from a window scenario?

5.6.1 Two Prize Subproblem

Consider the scenario $\mathcal{A} \rhd i_1$ and $\mathcal{B} \rhd \{i_1, i_2\}$ in the context of a larger problem. The **Two Prize Subproblem** is to determine how player $\mathcal{B}$ should play in this scenario so as to arrive at prize i_2 at the earliest possible time. Note that player $\mathcal{A}$ need not necessarily target prize i_2 following prize i_1.

This subproblem may arise in at least three contexts.

(i). Player $\mathcal{A}$ evaluates the scenario $\mathcal{A} \rhd i_1$ and $\mathcal{B} \rhd \{i_1, i_2\}$ and requires an estimate of the earliest possible arrival time of player $\mathcal{B}$ at prize i_2. A conservative estimate assumes that player $\mathcal{B}$ travels directly to prize i_2 and the arrival time is $t_B + d_{Bi_2}$.

(ii). Player $\mathcal{B}$ evaluates the scenario $\mathcal{A} \rhd i_1$ and $\mathcal{B} \rhd \{i_1, i_2\}$ and requires an estimate of the difference between the arrival time of player $\mathcal{A}$ at prize i_1 and the arrival time of player $\mathcal{B}$ at prize i_2. If player $\mathcal{A}$ does go directly to prize i_1 then player $\mathcal{B}$ can target the i_2-end of the $\mathcal{B}$-safety-window throughout and hence an estimate that the arrival time of player $\mathcal{A}$ at prize i_1 is $t_A + d_{Ai_1}$ and the arrival time of player $\mathcal{B}$ at prize i_2 is $t_B + d_{Bi_2}$. If player $\mathcal{A}$ does not go directly to prize i_1 then player $\mathcal{B}$ must still ensure that if either pair constraint initially holds then it continues to hold. However we would expect the difference gap to be no greater than $(t_B + d_{Bi_2}) - (t_A + d_{Ai_1})$, since we can reason that player $\mathcal{B}$ should only deviate from $\mathcal{B}\text{—}\circ\, i_2$ as often as player $\mathcal{A}$ deviates from $\mathcal{A}\text{—}\circ\, i_1$.

(iii). Player $\mathcal{B}$ has determined (exogenously) that player $\mathcal{A}$ is likely to target $\mathcal{A} \rhd i_1$ and that player $\mathcal{B}$'s best response is to play $\mathcal{B} \rhd \{i_1, i_2\}$. Hence player $\mathcal{B}$ needs to determine a one-step move so that the pair constraints are maintained, if possible. In this case player $\mathcal{B}$

Algorithm 5.18 strategy TWO PRIZE MEDIAN BIAS

Input: i_1, i_2 // *Prizes.*

// *Assuming* $\mathcal{A} \rhd i_1$.
if $d_{Ai_1} + d_{i_1 i_2} \leq d_{Bi_2}$ *or* $d_{Ai_2} + d_{i_1 i_2} \leq d_{Bi_1}$ **then**
  target $\leftarrow i_2$
else

$$x \leftarrow \frac{d_{i_1 i_2}^2 + d_{Bi_1}^2 - d_{Bi_2}^2}{2d_{i_1 i_2}}$$

$$z_1 \leftarrow \frac{d_{Bi_1}^2 - d_{Ai_1}^2}{2(x + d_{Ai_1})}$$

$$z_2 \leftarrow \frac{(d_{Ai_2} + d_{i_1 i_2})^2 - d_{Bi_1}^2}{2(d_{Ai_2} + d_{i_1 i_2} - x)}$$

  target $\leftarrow$ Bearing of location a distance $\max\{z_1, z_2\}$ from i_1 towards i_2.
end

end

assumes that $\mathcal{A}$—$\circ i_1$ but that $\mathcal{A} \rhd i_1$ is not absolutely certain. If $d_{Ai_1} + d_{i_1 i_2} \leq d_{Bi_2}$ or $d_{Ai_2} + d_{i_1 i_2} \leq d_{Bi_1}$ then the pair constraints cannot be salvaged and hence $\mathcal{B}$—$\circ i_2$. Thus suppose that the pair constraints hold with strict inequality. If $\mathcal{B} \rhd \{i_1, i_2\}$ is $\mathcal{B}$-median-feasible, then player $\mathcal{B}$ targets the i_2-end of the $\mathcal{B}$-median-window; otherwise player $\mathcal{B}$ plays to maintain $s_3 > 0$. The resulting strategy is encapsulated in Algorithm 5.18 TWO PRIZE MEDIAN BIAS.

5.6.2 Three Prize Window Feasible Subproblem

Consider the window-feasible scenario $\mathcal{A} \rhd i_3$ and $\mathcal{B} \rhd \{i_1, i_2\}$, where $i_3 \notin \{i_1, i_2\}$, in the context of a larger problem. The **Three Prize Window Feasible Subproblem** is to determine how player $\mathcal{B}$ should play in this scenario so as to arrive at prize i_1 or i_2 at the earliest possible time. Note that player $\mathcal{A}$ need not necessarily target either prize i_1 or i_2 following prize i_3.

Similarly to Section 5.6.1, this subproblem may arise in at least three contexts.

(i). Player $\mathcal{A}$ evaluates the scenario $\mathcal{A} \rhd i_3$ and $\mathcal{B} \rhd \{i_1, i_2\}$ and requires an estimate of the earliest possible arrival time of player $\mathcal{B}$ at prize i_2 such that player $\mathcal{B}$ satisfies the feasibility window. By case (i) of Section 5.6.1, a conservative estimate assumes that player $\mathcal{B}$ travels directly to the i_2-end of the feasibility window, W, and the arrival time is $t_B + d_{BW} + d_{Wi_2}$ where $d_{BW} = t_A + d_{Ai_3} - t_B$ and d_{Wi_2} is the shortest distance from the feasibility window to prize i_2.

(ii). Player $\mathcal{B}$ evaluates the scenario $\mathcal{A} \rhd (i_3 \rightarrow i_1)$ and $\mathcal{B} \rhd \{i_1, i_2\}$ and requires an estimate of the difference between the arrival time of player $\mathcal{A}$ at prize i_1 and the arrival time of player $\mathcal{B}$ at prize i_2. By case (ii) of Section 5.6.1, a reasonable estimate is $(t_B + d_{BW} + d_{Wi_2}) - (t_A + d_{Ai_3} + d_{i_2 i_1})$ in which player $\mathcal{B}$ travels directly to the i_2-end of the feasibility window,

W.

(iii). Player $\mathcal{B}$ has determined (exogenously) that player $\mathcal{A}$ is likely to target $\mathcal{A} \rhd i_3$ and that player $\mathcal{B}$'s best response is to play $\mathcal{B} \rhd \{i_1, i_2\}$. Hence player $\mathcal{B}$ needs to determine a one-step move so that window feasibility is maintained. In this case player $\mathcal{B}$ assumes that $\mathcal{A} \multimap i_3$ but that $\mathcal{A} \rhd i_3$ is not absolutely certain. This requires the calculation the bearing, θ_{i_1}, of the i_1-end of the feasibility window from player $\mathcal{B}$'s current location and the bearing, θ_{i_2}, of the i_2-end of the feasibility window. The *framework* for two prize strategies of Section 5.3.3 also applies to this scenario.

Target-Window. If possible we would like to ensure that, at the time player $\mathcal{A}$ claims prize i_3, player $\mathcal{B}$ is located at some location Z such that $\mathcal{B} \rhd \{i_1, i_2\}$ is median-feasible. Then Z satisfies $d_{ZU} \leq d_{i_3 i_1} + u$ where $u = \frac{1}{2}(d_{i_3 i_2} + d_{i_1 i_2} - d_{i_3 i_1})$ and U is the location a distance u from prize i_1 towards prize i_2. Hence we need to find the intersection of the feasibility-window W and circle C_4 where:

$$C_4: \quad \text{centre at } Z \text{ and radius } r_4 = d_{i_3 i_1} + z$$

If $\exists$ such an intersection then let the $\mathcal{B}$-target-window be the intersection; otherwise let the $\mathcal{B}$-target-window be the feasibility-window.

Target-Bearing. We either know exogenously which end of the target-window player $\mathcal{B}$ should target or we do not. If we do then player $\mathcal{B}$ should directly target that end of the target-window. Suppose we do not know. Since $\mathcal{A} \multimap i_3$ we cannot predict which (if either) of prize i_1 and i_2 that player $\mathcal{A}$ may target next. Hence we play similarly to the PREVIOUS MEDIAN strategy of Section 5.3.7. Let $u = \frac{1}{2}(d_{i_3 i_2} + d_{i_1 i_2} - d_{i_3 i_1})$ and let U be the location a distance u from prize i_1 towards prize i_2. Let ψ_U be the bearing of U from player $\mathcal{B}$'s current location. If ψ_U is within the target-window, then the target-bearing is ψ_U; otherwise the target-bearing is the end of the target-window which is closest in bearing to ψ_U.

Finally, Algorithm 5.19 WINDOW DISTANCE calculates d_{Wi_1}, d_{Wi_2}, θ_{i_1} and θ_{i_2}. The notation '$\psi_2 \circlearrowleft \psi_1$' is the counter-clockwise angle from bearing ψ_2 to bearing ψ_1.

5.6.3 Evaluation of a Window Scenario

Consider the window scenario $\mathcal{B} \rhd \{i_1, i_2\}$ in which player $\mathcal{A}$ arrives at location A at time t_A and player $\mathcal{B}$ arrives at location B at time $t_B < t_A$. This window scenario may or may not be window-feasible. We require an estimate of the expected reward, $\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\})$, to player $\mathcal{A}$ from this scenario. Assume that $\lambda = \infty$ for the purpose of illustrating the approach, which will be generalized in Chapter 6. Also assume that once only one prize remains, the value of the game position is $\Gamma(\mathcal{A}|\mathcal{B})$ to player $\mathcal{A}$ and $\Omega(\mathcal{A}, \mathcal{B}) - \Gamma(\mathcal{A}|\mathcal{B})$ to player $\mathcal{B}$, as suggested by the findings of the Tiny Tournament in Section 5.5.

If the window scenario is *not* window-feasible then

$$\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = v_{i_1} + v_{i_2}.$$

Algorithm 5.19 procedure WINDOW DISTANCE

Input: $Y, i_1, i_2, r_1, r_2, r_3$
Output: $dw_{i_1}, dw_{i_2}, \theta_{i_1}, \theta_{i_2}$

// Assume that the window scenario has a feasibility window.

Determine $\psi_1, \psi_2, \delta_1, \delta_2$ from Algorithm 5.1 WINDOW FEASIBLE.
if $\psi_2 \circlearrowleft \psi_1$ *is acute* **then**
  // Assume $\psi_1 > \psi_2$. **if** $(\psi_2 + \delta_2 \geq \psi_1)$ **then**
    $\theta_{i_1} \leftarrow \psi_1$
    $dw_{i_1} \leftarrow d_{Yi_1} - r_3$
  else
    $\theta_{i_1} \leftarrow \psi_2 + \delta_2$
    $dw_{i_1} \leftarrow \sqrt{(x_{i_1} - x_Y - r_3 \cos\theta_{i_1})^2 + (y_{i_1} - y_Y - r_3 \sin\theta_{i_1})^2}$
  end
  if $(\psi_1 - \delta_1 \leq \psi_2)$ **then**
    $\theta_{i_2} \leftarrow \psi_2$
    $dw_{i_2} \leftarrow d_{Yi_2} - r_3$
  else
    $\theta_{i_2} \leftarrow \psi_1 - \delta_1$
    $dw_{i_2} \leftarrow \sqrt{(x_{i_2} - x_Y - r_3 \cos\theta_{i_2})^2 + (y_{i_2} - y_Y - r_3 \sin\theta_{i_2})^2}$
  end
else
  // $\psi_1 \circlearrowleft \psi_2$ *is acute; assume* $\psi_1 < \psi_2$.
  if $(\psi_1 + \delta_1 \geq \psi_2)$ **then**
    $\theta_{i_2} \leftarrow \psi_2$
    $dw_{i_2} \leftarrow d_{Yi_2} - r_3$
  else
    $\theta_{i_2} \leftarrow \psi_1 + \delta_1$
    $dw_{i_2} \leftarrow \sqrt{(x_{i_2} - x_Y - r_3 \cos\theta_{i_2})^2 + (y_{i_2} - y_Y - r_3 \sin\theta_{i_2})^2}$
  end
  if $(\psi_2 - \delta_2 \leq \psi_1)$ **then**
    $\theta_{i_1} \leftarrow \psi_1$
    $dw_{i_1} \leftarrow d_{Yi_1} - r_3$
  else
    $\theta_{i_1} \leftarrow \psi_2 - \delta_2$
    $dw_{i_1} \leftarrow \sqrt{(x_{i_1} - x_Y - r_3 \cos\theta_{i_1})^2 + (y_{i_1} - y_Y - r_3 \sin\theta_{i_1})^2}$
  end
end

end

If the window scenario is window-feasible then let

$$\begin{aligned} s'_1 &= (t_B + d_{Bi_1}) - (t_A + d_{Ai_1}) \\ s'_2 &= (t_B + d_{Bi_2}) - (t_A + d_{Ai_2}) \\ s'_3 &= (t_A + d_{Ai_1}) - (t_B + d_{Bi_2}) \\ s'_4 &= (t_A + d_{Ai_2}) - (t_B + d_{Bi_1}) \end{aligned}$$

Then $s'_3 + d_{i_1 i_2} \geq 0$ and $s'_4 + d_{i_1 i_2} \geq 0$. Suppose, without loss of generality, that $s'_1 \leq s'_2$, by possibly switching prizes $i_1 \leftrightarrow i_2$.

- If $s'_1 < 0$ and $s'_2 < 0$ then

$$\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = \begin{cases} 0 & \text{if } s'_3 < d_{i_1 i_2} \text{ or } s'_4 < d_{i_1 i_2} \\ \frac{1}{2}\min\{v_{i_1}, v_{i_2}\} & \text{if } s'_3 = d_{i_1 i_2} \text{ and } s'_4 = d_{i_1 i_2} \\ \min\{v_{i_2}, \frac{1}{2}v_{i_1}\} & \text{if } s'_3 = d_{i_1 i_2} \text{ and } s'_4 > d_{i_1 i_2} \\ \min\{v_{i_1}, \frac{1}{2}v_{i_2}\} & \text{if } s'_4 = d_{i_1 i_2} \text{ and } s'_3 > d_{i_1 i_2} \\ \min\{v_{i_1}, v_{i_2}\} & \text{otherwise} \end{cases}$$

- If $s'_1 < 0$ and $s'_2 = 0$ then $\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = \min\{\frac{1}{2}(v_{i_1} + v_{i_2}), v_{i_2}\}$.
- If $s'_1 = 0$ and $s'_2 = 0$ then $\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = \frac{1}{2}(v_{i_1} + v_{i_2})$.
- If $s'_1 = 0$ and $s'_2 > 0$ then $\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = \max\{\frac{1}{2}(v_{i_1} + v_{i_2}), v_{i_2}\}$.
- If $s'_1 > 0$ and $s'_2 > 0$ then

$$\tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}) = \begin{cases} \max\{v_{i_1} + \frac{1}{2}v_{i_2}, v_{i_2} + \frac{1}{2}v_{i_1}\} & \text{if } s'_3 = -d_{i_1 i_2} \text{ and } s'_4 = -d_{i_1 i_2} \\ \max\{v_{i_2}, v_{i_1} + \frac{1}{2}v_{i_2}\} & \text{if } s'_3 = -d_{i_1 i_2} \text{ and } s'_4 > -d_{i_1 i_2} \\ \min\{v_{i_1}, v_{i_2} + \frac{1}{2}v_{i_1}\} & \text{if } s'_4 = -d_{i_1 i_2} \text{ and } s'_3 > -d_{i_1 i_2} \\ \min\{v_{i_1}, v_{i_2}\} & \text{otherwise} \end{cases}$$

The value, $\tilde{v}(\mathcal{B} \rhd \{i_1, i_2\}|\mathcal{A})$, which player $\mathcal{B}$ can expect is given by

$$\tilde{v}(\mathcal{B} \rhd \{i_1, i_2\}|\mathcal{A}) = v_{i_1} + v_{i_2} - \tilde{v}(\mathcal{A}|\mathcal{B} \rhd \{i_1, i_2\}).$$

5.6.4 Tactical Engines for Three Prize Problem

We now begin a tactical analysis of the three prize problem in terms of these three building blocks: the two prize subproblem, the three prize window feasible subproblem and the evaluation of a window scenario. Since we can discard from consideration any prize j for which $\min\{d_{Aj}, d_{Bj}\} > \lambda$, we assume that $\min\{d_{Aj}, d_{Bj}\} \leq \lambda \; \forall j \in V$.

A **tactical engine** is an "information engine" which systematically conducts "what if" analyses representative of possible player tactics, providing a strategy with useful information on the basis of which to make decisions.

Table 5.6: Cases of Three Prize Strategic Analysis for $\lambda = \infty$

Case	Definition		
1	$d_{A1} < d_{B1}$	$d_{A2} < d_{B2}$	$d_{A3} < d_{B3}$
2	$d_{A1} < d_{B1}$	$d_{A2} < d_{B2}$	$d_{A3} = d_{B3}$
3	$d_{A1} < d_{B1}$	$d_{A2} = d_{B2}$	$d_{A3} = d_{B3}$
4	$d_{A1} < d_{B1}$	$d_{A2} < d_{B2}$	$d_{A3} > d_{B3}$
5	$d_{A1} < d_{B1}$	$d_{A2} = d_{B2}$	$d_{A3} > d_{B3}$
6	$d_{A1} = d_{B1}$	$d_{A2} = d_{B2}$	$d_{A3} > d_{B3}$
7	$d_{A1} < d_{B1}$	$d_{A2} > d_{B2}$	$d_{A3} > d_{B3}$
8	$d_{A1} = d_{B1}$	$d_{A2} > d_{B2}$	$d_{A3} > d_{B3}$
9	$d_{A1} > d_{B1}$	$d_{A2} > d_{B2}$	$d_{A3} > d_{B3}$

5.6.4.1 Static and Dynamic Cases

If $\Gamma(\mathcal{A}|\mathcal{B}) + \Gamma(\mathcal{B}|\mathcal{A}) = \Omega(\mathcal{A}, \mathcal{B})$ then we have a **static** game position in which the players' paranoid subpaths are a saddle point equilibrium. If $\Gamma(\mathcal{A}|\mathcal{B}) + \Gamma(\mathcal{B}|\mathcal{A}) < \Omega(\mathcal{A}, \mathcal{B})$ then the game position is **dynamic**. This generalizes the classification of the two prize problem in Section 5.1 into *static* and *dynamic* cases.

Suppose that $\Gamma(\mathcal{A}|\mathcal{B}) + \Gamma(\mathcal{B}|\mathcal{A}) < \Omega(\mathcal{A}, \mathcal{B})$ and therefore each player must have at least one *accessible* prize. Suppose, without loss of generality, that $d_{B1} - d_{A1} \geq d_{B2} - d_{A2} \geq d_{B3} - d_{A3}$. The nine generic *dynamic* cases (for which $\lambda = \infty$) are listed in Table 5.6. Note that if each of the prizes is equidistant from the players can play EQUIDISTANT as in Algorithm 3.2.

Hence we assume throughout this section that $d_{Aj} \neq d_{Bj}$ for some prize j. Case 1 corresponds to case 9 in that if player $\mathcal{A}$ faces case 1 then player $\mathcal{B}$ faces case 9. The remaining cases also correspond according to $2 \leftrightarrow 8$, $3 \leftrightarrow 6$, $4 \leftrightarrow 7$ and $5 \leftrightarrow 5$.

We wish to approximate the tactical decisions that a player faces. We know that once at least one prize is claimed we can apply our one prize or two prize analysis to determine what to do. Also, given a particular two prize scenario, we can evaluate the expected result for each player. Thus we wish to approximate the initial tactical decision of each player.

A *tactical engine* builds a tactical summary of the current game position in terms of each player's tactical options and also estimates the expected reward to each player. In Sections 5.6.4.2 and 5.6.4.3 we design two such tactical engines, THREE-ORIGINAL-DT[4] and THREE-PRIZE-DT respectively, for the three prize problem.

5.6.4.2 Tactical Engine: THREE-ORIGINAL-DT

The simplest tactical approximation is to consider which accessible prize each player will initially target. Figure 5.8 illustrates this for the case in which every prize is accessible to both players. This approximation of tactics is itself a *two-person game* in normal form, with the pure one-off strategies for each player corresponding to the accessible target prizes.

[4]The names THREE-ORIGINAL-DT and THREE-PRIZE-DT are inherited from the tactical engines ORIGINAL-DT and PRIZE-DT for small problems in Chapter 6.

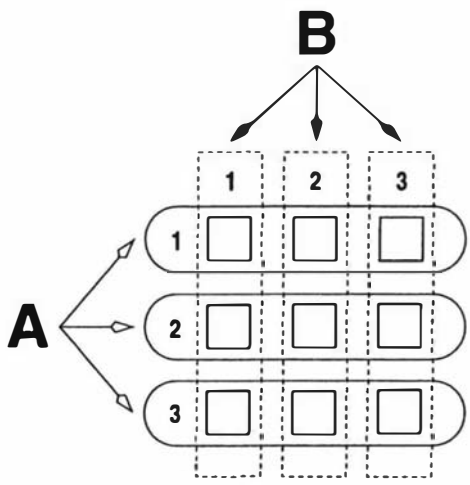


Figure 5.8: Initial target prizes of $\mathcal{A}$ and $\mathcal{B}$.

Suppose each prize is accessible to *both* players. Each of the cases of Table 5.6 gives rise to a 3×3 game table structure in Figure 5.9, in which we arrange the target-prizes along the side for player $\mathcal{A}$ and across the top for player $\mathcal{B}$. The prize labels are replaced by one of three symbols: $\{\circ, \blacksquare, \imath\}$. A prize j for which $d_{Aj} < d_{Bj}$ is labelled '$\circ$' for player $\mathcal{A}$ along the side and labelled '$\imath$' for player $\mathcal{B}$ across the top. A prize j for which $d_{Aj} > d_{Bj}$ is labelled '$\imath$' for player $\mathcal{A}$ along the side and labelled '$\circ$' for player $\mathcal{B}$ across the top. A prize j for which $d_{Aj} = d_{Bj}$ is labelled '$\blacksquare$' for both players.

If there are actually prizes which are not accessible to a player then the corresponding row is omitted for player $\mathcal{A}$ and the corresponding column is omitted for player $\mathcal{B}$.

Each entry in the game-table represents the scenario in which player $\mathcal{A}$ moves directly towards the prize associated with the corresponding row and player $\mathcal{B}$ moves directly towards the prize associated with the corresponding column until either player (or both players simultaneously) reach their target prize. If this scenario is realised then the consequential reward to player $\mathcal{A}$ can be estimated by the value (or share of the value) of the first prize claimed (if player $\mathcal{A}$ claimed the first prize) plus the expected reward to player $\mathcal{A}$ from the remaining one prize or two prize problem.

■ Evaluation of Expected Reward

It remains to tactically estimate the expected reward to player $\mathcal{A}$.

Assumption 5.6.1

The **maximin assumption** is that player $\mathcal{A}$ selects the best possible option, given that player $\mathcal{B}$ *knows the option player* $\mathcal{A}$ *will select.* □

Assumption 5.6.1 gives rise to the (standard) MAXIMIN value for player $\mathcal{A}$: the maximum of the minimum value in each row of the *game in normal form* (the *game-table*).

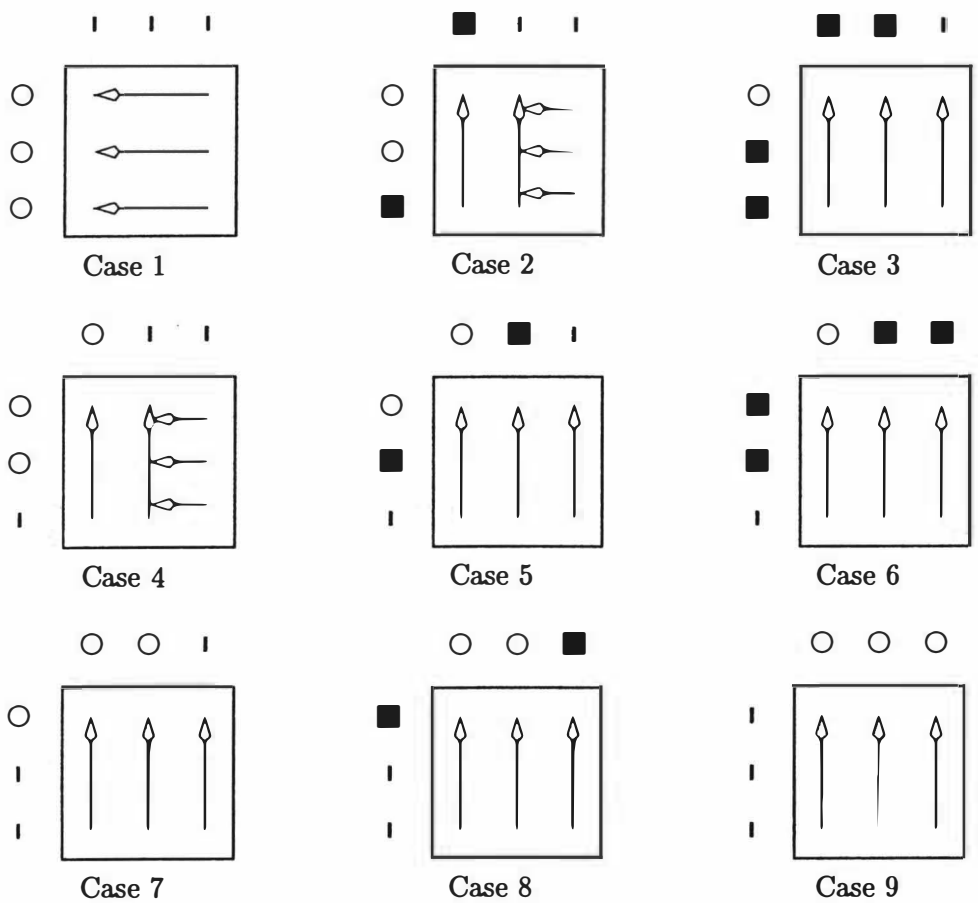


Figure 5.9: Nine cases for THREE-ORIGINAL-DT

Assumption 5.6.2

The **minimax assumption** is that player $\mathcal{A}$ selects the best possible option, given that player $\mathcal{B}$ *selects an option which most restricts player* $\mathcal{A}$*'s result while knowing that player* $\mathcal{A}$ *knows which option player* $\mathcal{B}$ *will select.* □

Assumption 5.6.2 gives rise to the (standard) MINIMAX value for player $\mathcal{A}$: the minimum of the maximum value in each column of the game-table.

These assumptions are ideals since they are based upon scenarios in which the players carry through their commitments—$\mathcal{A} \rhd x$ and $\mathcal{B} \rhd y$—but the practical implementation is just one step at a time with perfect observation of the opponent—$\mathcal{A}$—$\circ$ x and $\mathcal{B}$—$\circ$ y. The maximin assumption is ultra-conservative in that player $\mathcal{A}$ does adapt to what it can observe the opponent's response to be. The minimax assumption is ultra optimistic in that player $\mathcal{A}$ assumes that it can effectively observe which prize player $\mathcal{B}$ is targeting and that player $\mathcal{B}$ does not adapt to whatever player $\mathcal{A}$ is targeting.

Figure 5.9 also illustrates how player $\mathcal{A}$ evaluates the three prize problem in terms of game-table. Left pointing arrows represent minimization across a row and upward pointing arrows represent maximization within a column. Case 1 employs the standard MAXIMIN evaluation and case 3 and cases 5–9 employ the standard MINIMAX evaluation.

Assumption 5.6.3

The **minimaximin assumption** is that player $\mathcal{A}$ selects the best possible option, given that player $\mathcal{B}$ selects an option which most restricts player $\mathcal{A}$'s result, assuming that player $\mathcal{A}$ knows whether player $\mathcal{B}$ will target a prize which is closer to player $\mathcal{A}$ or not, under the two following conditions:

(i). If player $\mathcal{B}$ will target a prize closer to player $\mathcal{A}$, then player $\mathcal{A}$ knows which prize player $\mathcal{B}$ is targeting.

(ii). If player $\mathcal{B}$ will target a prize not closer to player $\mathcal{B}$, then player $\mathcal{A}$ assumes that player $\mathcal{B}$ knows which prize player $\mathcal{A}$ will select.

□

The minimaximin assumption is a hybrid compromise between the maximin and minimax assumptions in which the minimax assumption applies over those prizes for which it is possible for player $\mathcal{A}$ to effectively observe whether or not player $\mathcal{B}$ is targeting a specific prize; the maximin assumption applies over the remaining prizes.

Let $\tilde{v}(\mathcal{A} \rhd x | \mathcal{B} \rhd y)$ be the evaluation of the game-table entry corresponding to $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd y$. Case 2 and case 4 employ a special MINIMAXIMIN evaluation defined by

$$\min \left\{ \min_{y \in Q : d_{\mathcal{A}y} \geq d_{\mathcal{B}y}} \left\{ \max_{x \in Q} \tilde{v}(\mathcal{A} \rhd x | \mathcal{B} \rhd y) \right\}, \max_{x \in Q} \left\{ \min_{y \in Q : d_{\mathcal{A}y} < d_{\mathcal{B}y}} \tilde{v}(\mathcal{A} \rhd x | \mathcal{B} \rhd y) \right\} \right\}.$$

Warning. If both players apply a minimax or minimaximin assumption, then the sum of each player's evaluation of the three prize game position may exceed $\Omega(\mathcal{A}, \mathcal{B})$, since both are

optimistic evaluations. This is an inherent danger in the MINIMAX assumption, since it may be overly optimistic and can lead to logical cycling.

This completes the design of the THREE-ORIGINAL-DT tactical engine for the three prize problem. This is not yet a strategy, since we have not yet specified how to select a one-step move from this tactical information.

5.6.4.3 Tactical Engine: THREE-PRIZE-DT

An alternative tactical approximation is for a player to consider an accessible target prize only if that prize is not closer to the opponent, and to also consider pairs of accessible prizes which are closer to the opponent, incorporating the *three prize window feasible subproblem* of Section 5.6.2.

Suppose each prize is accessible to *both* players. Each of the cases of Table 5.6 gives rise to a 2×2, 2×3, 3×2 or 3×3 game table structure in Figure 5.10, in which we arrange the prizes along the side for player $\mathcal{A}$ and across the top for player $\mathcal{B}$. The symbol 'II' represents a pair of 'I' prizes, i.e., $\mathcal{A} \triangleright \{j,k\}$ for which $d_{Aj} > d_{Bj}$ and $d_{Ak} > d_{Bk}$ is labelled 'II' for player $\mathcal{A}$ along the side and $\mathcal{B} \triangleright \{j,k\}$ for which $d_{Aj} < d_{Bj}$ and $d_{Ak} < d_{Bk}$ is labelled 'II' for player $\mathcal{B}$ across the top.

If there are actually prizes which are not accessible to a player, then the row corresponding to an inaccessible prize, or a pair containing an inaccessible prize, is omitted for player $\mathcal{A}$ and the corresponding column is omitted for player $\mathcal{B}$.

■ Evaluation of Expected Reward

Figure 5.10 also illustrates how player $\mathcal{A}$ evaluates the three prize problem in terms of the game-table. Case 1 employs the standard MAXIMIN evaluation and the remaining cases employ the standard MINIMAX evaluation. Note that in cases 4–6, player $\mathcal{A}$ has only two possible target prizes and in case 3, case 5 and case 7, player $\mathcal{B}$ has only two possible target prizes. This is because the remaining prize is the only prize closer to the opponent and hence the player cannot target a pair of prizes.

Evaluating the entries in the game-table is straightforward. We illustrate this assuming that $\lambda = \infty$. When $\lambda < \infty$, more subcases are required. Again, we generalize this in Chapter 6.

▼ Evaluating $\tilde{v}(\mathcal{A} \triangleright i_1 | \mathcal{B} \triangleright \{i_1, i_2\})$ involves firstly determining the status of $s_3 = d_{Ai_1} + d_{i_1 i_2} - d_{Bi_2}$.

- If $s_3 < 0$, then $\tilde{v}(\mathcal{A} \triangleright i_1 | \mathcal{B} \triangleright \{i_1, i_2\}) = v_{i_1} + v_{i_2} + v_{i_3}$.
- If $s_3 \geq 0$, then we need to evaluate scenario (i) of Section 5.6.1 as follows. Certainly player $\mathcal{A}$ claims prize i_1.
  - If $s_3 > 0$, then player $\mathcal{B}$ also claims prize i_2. To evaluate the remaining one prize subproblem let

$$\tilde{v}(\{i_3\}) = \begin{cases} v_{i_3} & \text{if } d_{Ai_1} + d_{i_1 i_3} < d_{Bi_2} + d_{i_2 i_3} \\ \frac{1}{2} v_{i_3} & \text{if } d_{Ai_1} + d_{i_1 i_3} = d_{Bi_2} + d_{i_2 i_3} \\ 0 & \text{if } d_{Ai_1} + d_{i_1 i_3} > d_{Bi_2} + d_{i_2 i_3} \end{cases}$$

  Hence $\tilde{v}(\mathcal{A} \triangleright i_1 | \mathcal{B} \triangleright \{i_1, i_2\}) = v_{i_1} + \tilde{v}(\{i_3\})$.

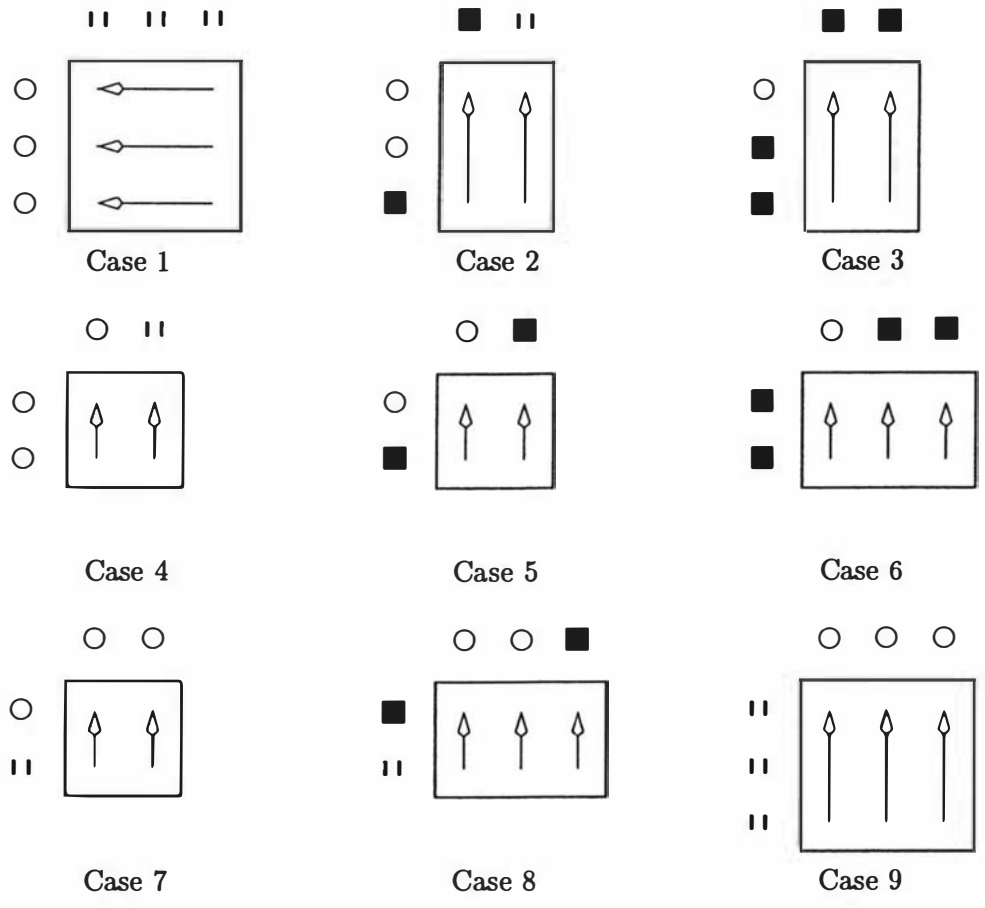


Figure 5.10: Nine cases for THREE-PRIZE-DT

– If $s_3 = 0$, then player $\mathcal{B}$ may not claim prize i_2. The remaining window scenario $\mathcal{A} \rhd \{i_2, i_3\}$ is window-feasible with evaluation

$$\tilde{v}(\mathcal{A} \to i_1 \rhd \{i_2, i_3\} | \mathcal{B} \rhd i_2) = \begin{cases} \frac{1}{2}(v_{i_2} + v_{i_3}) & \text{if } d_{i_1 i_3} = d_{i_1 i_2} + d_{i_2 i_3} \\ \max\{v_{i_3}, \frac{1}{2}(v_{i_2} + v_{i_3})\} & \text{if } d_{i_1 i_3} < d_{i_1 i_2} + d_{i_2 i_3} \end{cases}$$

Hence $\tilde{v}(\mathcal{A} \rhd i_1 | \mathcal{B} \rhd \{i_1, i_2\}) = v_{i_1} + \tilde{v}(\mathcal{A} \to i_1 \rhd \{i_2, i_3\} | \mathcal{B} \rhd i_2)$.

▼ Evaluating $\tilde{v}(\mathcal{A} \rhd i_3 | \mathcal{B} \rhd \{i_1, i_2\})$ involves firstly determining the window-feasible status of the window scenario $\mathcal{A} \rhd i_3$ and $\mathcal{B} \rhd \{i_1, i_2\}$.

- If the window scenario is *not* window-feasible, then

$$\tilde{v}(\mathcal{A} \rhd i_3 | \mathcal{B} \rhd \{i_1, i_2\}) = v_{i_1} + v_{i_2} + v_{i_3}.$$

- If the window scenario is window-feasible, then player $\mathcal{A}$ claims prize i_3 and we must evaluate the remaining window scenario $v(\mathcal{A} \to i_3 | \mathcal{B} \rhd \{i_1, i_2\})$, which is the window scenario of Section 5.6.3 with $t_A = d_{Ai_3}$ and $t_B = 0$. Then

$$\tilde{v}(\mathcal{A} \rhd i_3 | \mathcal{B} \rhd \{i_1, i_2\}) = v_{i_1} + \tilde{v}(\mathcal{A} \to i_3 | \mathcal{B} \rhd \{i_1, i_2\}).$$

▼ Evaluating $\tilde{v}(\mathcal{A} \rhd i_1 | \mathcal{B} \rhd i_1)$ is only required when $d_{Ai_1} = d_{Bi_1}$. Then prize i_1 is shared by the players and, at the instance prize i_1 is claimed, the players will be identically located. The evaluation of the remaining two prize problem is that the remaining two prizes will also be shared. Hence

$$v(\mathcal{A} \rhd i_1 | \mathcal{B} \rhd i_1) = \tfrac{1}{2}(v_{i_1} + v_{i_2} + v_{i_3}).$$

▼ We can similarly define $\tilde{v}(\mathcal{A} \rhd \{i_1, i_2\} | \mathcal{B} \rhd i_1)$ and $\tilde{v}(\mathcal{A} \rhd \{i_1, i_2\} | \mathcal{B} \rhd i_3)$.

This completes the design of the THREE-PRIZE-DT tactical engine. Again, this is not yet a strategy, since we have not yet specified how to select a one-step move from this tactical information. Observe that both these tactical engines employ a "divide and conquer" approach by formulating and approximately evaluating a number of smaller (one or two prize) problems.

Coda

▼ Summary

This chapter has mathematically analysed the two prize CPCP, established the single step strategies, performed a computational tournament to establish the validity of a heuristic tactical assessment and finally shown how these building blocks may be incorporated into a method for evaluating tactics for the three prize CPCP.

▼ Link

The approach we have taken to the three prize problem is to design tactical engines which summarise the evaluation of a number of tactical scenarios. We now require a link between this tactical planning and using the observation of the opponent's movements to select a response. This link is provided by the concept of *monitoring* as introduced in Section 3.2.3.2. Having illustrated the game theoretic approach, without actually designing a complete strategy, we develop these ideas further in the next chapter in which we consider small problems.

CHAPTER 6

Tactical Planning for Small Problems

Any inanimate object, regardless of its position, configuration or purpose, may be expected to perform, at any time, in a totally unexpected manner, for reasons that are either entirely obscure or else completely mysterious.

— FLAP'S LAW

Tactical planning is the systematic analysis of a problem in terms of the decision of determining which prizes to target. Such planning is reasonably coarse in comparison to the fine planning when selecting a one-step move as for the two prize problem. Small problems can be defined as those for which this tactical planning approach is computationally feasible. In this chapter we develop tactical engines for small problems and a hierarchical dynamic monitoring system, thus designing a number of complete strategies.

6.0 Introduction

Section 5.6 introduced a tactical approach to the Three Prize Problem which approximates the initial tactical decision faced by a player, namely which prize to initially target. The objective of this chapter is to expand this tactical planning approach to problems involving up to approximately ten prizes and to design an implementation of the SPA/DMS with two frames. Chapter 5

has provided the *step-frame* and STEP-MONITOR components of the SPA/DMS. This chapter contributes the design of a number of *scenario engine* and a *monitor* for the *global-frame*.

6.0.1 Tactical Planning Problem

To determine tactically robust movements requires a representative coverage of possible scenarios at as fine a level of detail as computational tractability considerations will permit. Therefore, in order to analyse problems consisting of more than two prizes, we restrict consideration to tactical scenarios. The **tactical planning problem** is to determine which prize to target for a range of prize target scenarios of the opponent.

Solving the tactical planning problem necessarily implies *contingent planning*, since an implicit response is required for each tactical scenario of the opponent. A systematic treatment of contingent planning must also consider recursively contingent scenarios in which contingent sequences of prize targets beyond the initial prize targets of each player are evaluated in order to determine the initial scenarios.

A natural formulation for such a planning problem is a multiple stage game with observed actions and simultaneous moves. All players know the actions chosen at all previous stages when choosing their current actions and all players move simultaneously in each stage (each player chooses his or her action at stage k without knowing the stage-k action of any other player). A **game tree** is an explicit representation of all possible plays of the game (Pearl [174]). The game tree corresponds to a contingent strategy since, in selecting an initial target-prize, the player assumes that a subsequent tactical decision, involving the selection of subsequent target-prize, can be resolved by considering the particular game position encountered at that time.

From the initial game position we propose a set of scenarios in which each player selects a target. With each particular scenario we associate a representative *projected game position*. To evaluate this projected game position, we recursively propose a further set of scenarios in which each player selects a target and associates with each one a representative projected game position, and so on. In this way we systematically generate a multistage game in which each stage corresponds to a tactical decision. Such a game tree is called a **tactical game tree** since it approximates the decisions of a tactical planning problem at each stage. The root node corresponds to the initial game position and its children correspond to each representative game position derived from the set of tactical scenarios, and so on. Terminal nodes correspond to game positions in which at most one prize remains.

6.0.2 Tactical Engines

Recall from Section 4.2.3 that a DMS implementation requires both a *scenario engine* as its planner module and a *monitor* as its selector module. Section 6.4 will consider the design of the Small Dynamic Monitoring System, in particular the necessary *global monitor* and *step monitor*. Preparatory to this, we consider tactical game tree based scenario engines which we term **tactical engines**.

Representative Projected Game Positions

Let $Q_A = \{j \in V : t_0 + d_{Aj} \leq \lambda\}$ and let $Q_B = \{j \in V : t_0 + d_{Bj} \leq \lambda\}$. Consider the scenario in which player $\mathcal{A}$ targets prize $x \in Q_A$ and player $\mathcal{B}$ targets prize $y \in Q_B$. We need to select a projected game position to represent the implementation of this scenario. We also need to specify how committed the players are to their respective target prizes. Since a finite game tree is desirable, we reason that the players move towards their respective target-prizes at least until one player reaches its target-prize so that at least one prize is claimed at each depth of the game tree. The possibilities include the following two:

(a). As in THREE-ORIGINAL-DT,[1] move a distance $\min\{d_{Ax_1}, d_{By_1}\}$ until one player reaches its target-prize. We call this type of projection a **probe**.

(b). As in THREE-PRIZE-DT, move until one player reaches its target-prize and, if the other player does not simultaneously reach its target-prize, then that player must select the same prize at its next target-prize. We call this type of projection a **commitment**.

The analysis of Section 5.6 shows that steps alone are too detailed for three or more prizes, but that prizes are reasonable targets of manageable complexity. This does not dismiss methods which rely on bigger "steps" than Δ; these would need to be compared against targeting prizes in future work. Also, targeting prizes offers a computational complexity advantage in that there are only a finite number of prizes to target.

Single Stage Tactical Engine

Section 6.1 considers the design of the PRIZE-PARANOID tactical engine whose tactical game tree consists of only a single stage. Each scenario is a probe and is evaluated by determining the maximal PRIZE-GUARANTEE subpath for each player from the projected game position.

Multiple Stage Tactical Engines

There are three requirements for specification of a multiple stage tactical engine.

Game tree generation. How do we construct the representative game positions?

Evaluating a game tree node. How do we determine an evaluation for player $\mathcal{A}$ of each game tree node given the evaluations of each child node?

Searching of the game tree. How can we use the problem structure and information already gained to most efficiently determine the information required to make a decision at the root node?

The resulting root node game table summarises the tactical response information from the corresponding tactical tree.

Two multiple stage tactical engines are designed in this chapter. Section 6.2 considers the design of the ORIGINAL-DT tactical engine in which each scenario at a given stage of the tactical

[1]The suffix '-DT' stands for **double tree** since each possible path through the game-tree corresponds to a pair of planning paths, one for each player, which we colloquially call a **double path**.

game tree is a probe and the projected position of each player at each stage is at the same instant in future time. Section 6.3 considers the design of the PRIZE-DT tactical engine in which each scenario at a given stage of the tactical game tree is a commitment, and either a projected game position or a projected two-prize window feasible scenario is maintained for each player, possibly at different, though comparable, future instants of time.

6.1 Tactical Engine: PRIZE-PARANOID

The *guarantee subpath subproblem* of Section 3.3.1 was to determine a subpath, guaranteed to a player, of maximal guaranteed value. The PRIZE-PARANOID tactical engine constructs a game table in which each scenario is a single probe that is evaluated by solving a corresponding guarantee path subproblem.

6.1.1 Game Table Generation

A **projected game position** consists of a pair of **projected locations**, A and B; a **projected time-stamp**, t; and a **projected set of remaining prizes**, $Q \subseteq V$. Let Q be the set of remaining prizes and let $Q_A = \{j \in Q : t + d_{Aj} \leq \lambda\}$ and $Q_B = \{j \in Q : t + d_{Bj} \leq \lambda\}$.

Choose $x \in Q_A$ and $y \in Q_B$ and suppose that player $\mathcal{A}$ targets prize x and player $\mathcal{B}$ targets prize y. Let $t' = t + \min\{d_{Ax}, d_{By}\}$. If $d_{Ax} < d_{By}$, then let A' be the location of prize x, let B' be the location which is a distance d_{Ax} from B towards prize y and let $Q' = Q \setminus \{x\}$. If $d_{Ax} = d_{By}$, then let A' be the location of prize x, let B' be the location of prize y and let $Q' = Q \setminus \{x, y\}$. If $d_{Ax} > d_{By}$, then let A' be the location which is a distance d_{By} from A towards prize x, let B' be the location of prize y and let $Q' = Q \setminus \{y\}$. The scenario in which $\mathcal{A} \triangleright A'$ and $\mathcal{B} \triangleright B'$ is called a **probe**. The notation "probe $\mathcal{A} \rightarrow P_A \odot x$ and $\mathcal{B} \rightarrow P_B \odot y$" defines a new game position in which prize x is appended to P_A, prize y is appended to P_B, $A \leftarrow A'$, $B \leftarrow B'$, $t \leftarrow t'$ and $Q \leftarrow Q'$.

We can now define how to generate the PRIZE-PARANOID game table.

- $\forall x \in Q_A$ and $\forall y \in Q_B$ probe $\mathcal{A} \rightarrow P_A \odot x$ and $\mathcal{B} \rightarrow P_B \odot y$.

Each probe is evaluated by determining the maximum guaranteed value attainable from the corresponding projected game position. Assemble the probe evaluations in a game table with the targets, Q_A, for player $\mathcal{A}$ along the left hand side and the targets, Q_B, for player $\mathcal{B}$ along the top.

6.1.2 Evaluating the Root Game Table

For each target $y \in Q_B$ of player $\mathcal{B}$ the best response target of player $\mathcal{A}$ is defined by

$$x^*(\mathcal{B} \odot y) = \text{argmax}_{x \in Q_A} \tilde{v}_A(\mathcal{A} \odot x, \mathcal{B} \odot y) \tag{6.1}$$

where $\tilde{v}_A(\mathcal{A} \odot x, \mathcal{B} \odot y)$ is the maximum guaranteed value attainable from the projected game position corresponding to the probe $\mathcal{A} \rightarrow P_A \odot x$ and $\mathcal{B} \rightarrow P_B \odot y$.

Applying the *minimax assumption* (Assumption 5.6.2) gives rise to the MINIMAX evaluation of the PRIZE-PARANOID game table. Let

$$\tilde{v}_A = \min_{y \in Q_B} \left\{ \max_{x \in Q_A} \tilde{v}_A(\mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \tag{6.2}$$

Let T be a *prize-ts* as in Section 3.2.3.2 and suppose that we cannot determine which target in T player $\mathcal{B}$ will target (this is denoted $\mathcal{B} \rhd T$). Then the best response target of player $\mathcal{A}$ from the PRIZE-PARANOID game table is defined by

$$x^*(\mathcal{B} \odot y) = \text{argmax}_{x \in Q_A} \left\{ \min_{y \in T} \tilde{v}_A(\mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \tag{6.3}$$

The corresponding MINIMAX evaluation is

$$\tilde{v}_A(\mathcal{B} \rhd T) = \min_{y \in T} \left\{ \max_{x \in Q_A} \tilde{v}_A(\mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \tag{6.4}$$

Consider also the natural extension of PRIZE-GUARANTEE to a *prize-ts*, T, in which player $\mathcal{B}$ is restricted to targeting a prize from T before considering any other prize. The corresponding guarantee path subproblem has earliest arrival times of player $\mathcal{B}$ modifed to

$$\tau_{Bj} = \min_{i \in T} \{ d_{Bi} + d_{ij} \}$$

Note that these two target-set variations provide different evaluations since they include different assumptions of what can occur past the first pair of targets. The TARGET-SET PRIZE-PARANOID evaluation assumes that player $\mathcal{A}$ knows what player $\mathcal{B}$ initially targeted past the first target, whereas TARGET-SET PRIZE-GUARANTEE assumes that player $\mathcal{A}$ does not.

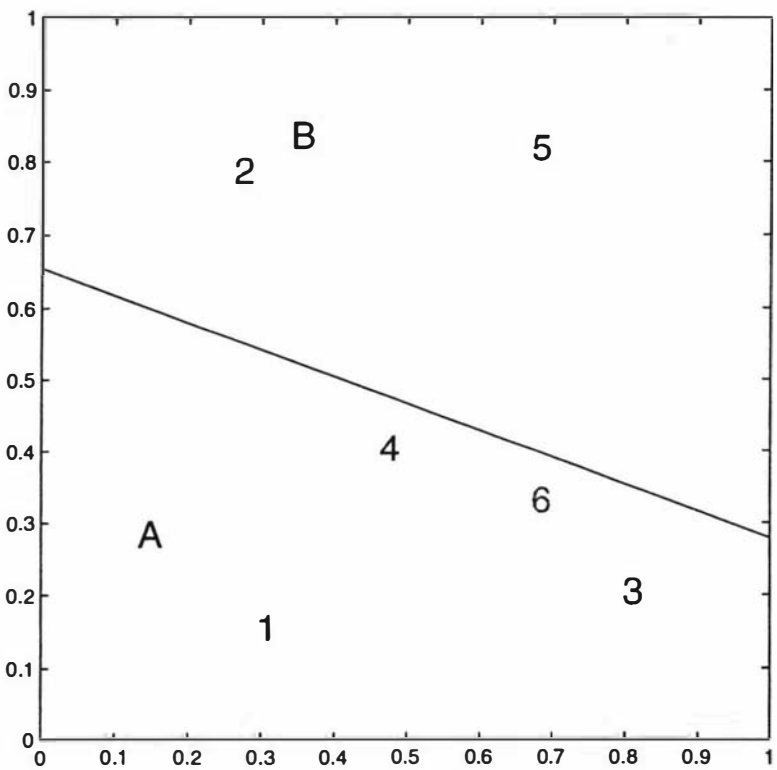
6.1.3 Example Tactical Problem

Consider the example problem of Figure 6.1. Here, the prizes are indexed from $\{1,2,3,4,5,6\}$ with each prize location at the centre of each prize label. The players are labelled $\{\mathcal{A}, \mathcal{B}\}$ with initial player location at the centre of each player label. The solid line partitions the prize set into those prizes initially closer to player $\mathcal{A}$ and those prizes initially closer to player $\mathcal{B}$. The prize values, v, and locations, (x, y), initial player locations and overall deadline λ are defined in the accompanying tables. Note that $\sum_{j \in V} v_j = 1000$, i.e., the prize values sum to 1000, in this example.

6.1.3.1 Analysis of PRIZE-GUARANTEE

Table 6.1 illustrates the tactics determined by PRIZE-GUARANTEE and TARGET-SET PRIZE-GUARANTEE for both players.

- The maximal PRIZE-GUARANTEE subpath for player $\mathcal{A}$ is $\mathcal{A}$→4→6→3 (value 471) as illustrated in Table 6.1(a). In particular, note that $\mathcal{A}$→1→4 (value 355) is not guaranteed, but $\mathcal{A}$→4→1 is guaranteed, $\mathcal{A}$→1→3 (value 429) is guaranteed, but $\mathcal{A}$→1→6→3 is not guaranteed.



Prize	v	x	y
1	202	0.31	0.16
2	137	0.28	0.79
3	227	0.81	0.20
4	153	0.48	0.40
5	190	0.68	0.82
6	91	0.68	0.33
Total	1000		

Player	x	y
$\mathcal{A}$	0.15	0.28
$\mathcal{B}$	0.36	0.84

Deadline	$\lambda = \infty$

Figure 6.1: Example Tactical Problem

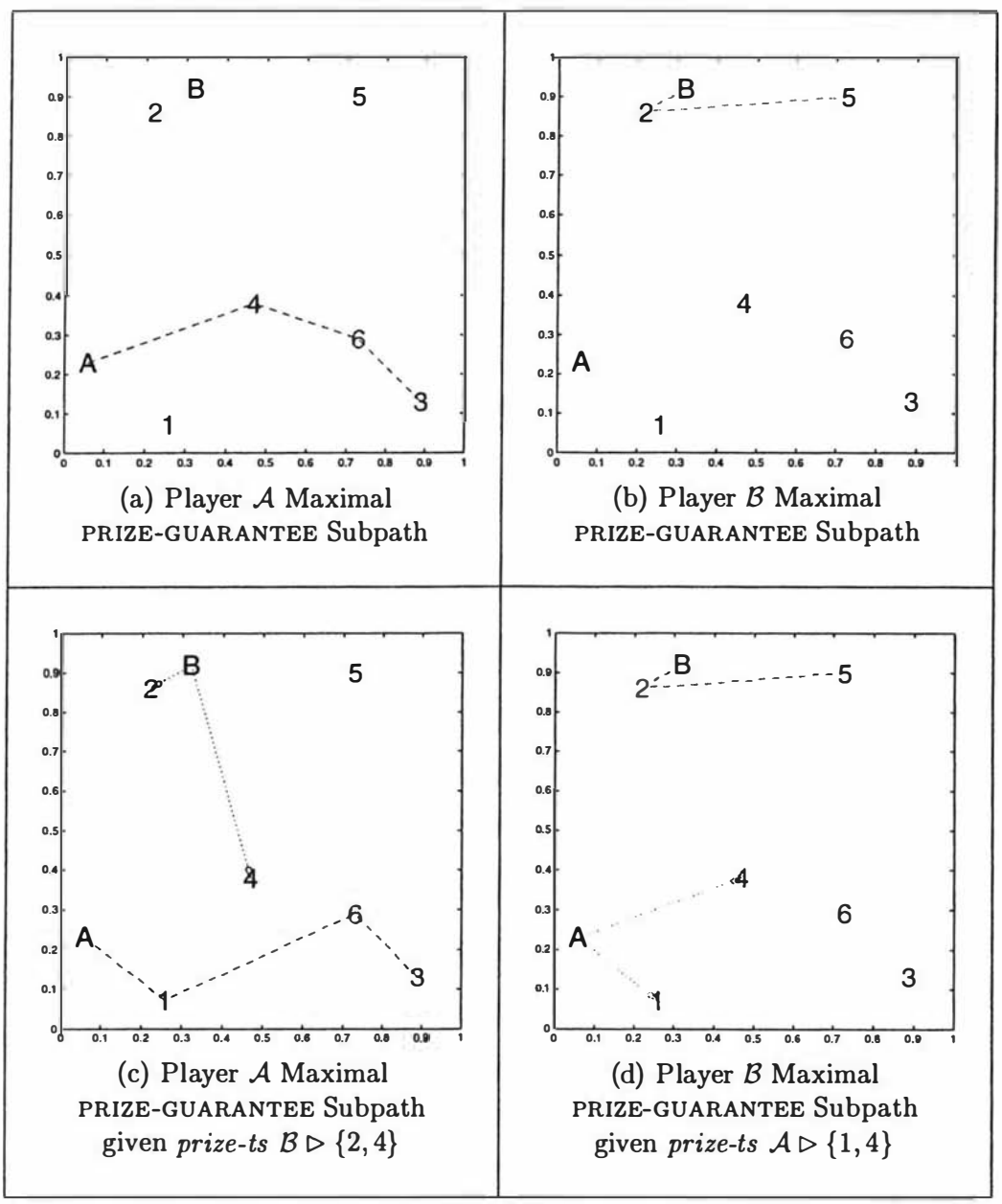
- The maximal PRIZE-GUARANTEE subpath for player $\mathcal{B}$ is $\mathcal{B}\rightarrow 2\rightarrow 5$ (value 327) as illustrated in Table 6.1(b). Also, $\mathcal{B}\rightarrow 5\rightarrow 2$ is guaranteed.
- With TARGET-SET PRIZE-GUARANTEE for player $\mathcal{A}$, suppose that $\mathcal{B} \rhd \{2,4\}$, i.e., that player $\mathcal{B}$ will claim either prize 2 or prize 4 next. The corresponding best guaranteed path is then $\mathcal{A}\rightarrow 1\rightarrow 6\rightarrow 3$ (value 520) as illustrated in Table 6.1(c). This is because the earliest that player $\mathcal{B}$ could then arrive at prize 3 is via prize 2 but player $\mathcal{A}$ can then traverse $\mathcal{A}\rightarrow 1\rightarrow 6\rightarrow 3$ and arrive at prize 3 prior.
- With TARGET-SET PRIZE-GUARANTEE for player $\mathcal{B}$, suppose that $\mathcal{A} \rhd \{1,4\}$, i.e., that player $\mathcal{A}$ will claim either prize 1 or prize 4 next. The corresponding best guaranteed path is unchanged from $\mathcal{B}\rightarrow 2\rightarrow 5$ (value 327) as illustrated in Table 6.1(d).

6.1.3.2 Analysis of PRIZE-PARANOID

Table 6.2 illustrates the tactics determined by PRIZE-PARANOID and TARGET-SET PRIZE-PARANOID for both players.

- Table 6.2(a) gives the PRIZE-PARANOID game table for player $\mathcal{A}$. This is interpreted by applying a MINIMAX evaluation, i.e., minimize the column maximums, which implies that player $\mathcal{B}$ will target $\mathcal{B}\odot 3$, $\mathcal{B}\odot 4$ or $\mathcal{B}\odot 6$ with evaluation 471. These scenarios are highlighted in the game table with boxes, i.e., $\boxed{471}$. In each case the best response tactic for player $\mathcal{A}$ is $\mathcal{A}\odot 4$. The value 471 equates to claiming prizes $\{3, 4, 6\}$. Note that a MAXIMIN evaluation of the player $\mathcal{A}$ game table also implies $\mathcal{A}\odot 4$. Table 6.2(c) illustrates the resulting PRIZE-GUARANTEE subpaths for each player under the probe scenario of $\mathcal{A}\odot 4$ and $\mathcal{B}\odot 3$. Note that prizes $\{1,2\}$ remain unclaimed by this analysis, which shows that PRIZE-PARANOID does not account for all of the prizes in determining its evaluation of a tactical probe scenario. This is simply because it relies on PRIZE-GUARANTEE for scenario evaluation.
- Table 6.2(b) gives the PRIZE-PARANOID game table for player $\mathcal{B}$. A MINIMAX analysis implies that player $\mathcal{A}$ will target $\mathcal{A}\odot 4$ and hence $\mathcal{B}\odot 2$ or $\mathcal{B}\odot 5$ with evaluation 327 (equating to prizes $\{2,5\}$). Table 6.2(d) illustrates the resulting PRIZE-GUARANTEE subpaths for each player under the probe scenario $\mathcal{A}\odot 4$ and $\mathcal{B}\odot 5$. This time, only prize 1 in not accounted for.
- With TARGET-SET PRIZE-PARANOID for player $\mathcal{A}$, suppose that $\mathcal{B} \rhd \{2, 5\}$. Evaluation of the columns of the player $\mathcal{A}$ PRIZE-PARANOID game table corresponding to $\mathcal{B}\odot 2$ and $\mathcal{B}\odot 5$ implies that either option for player $\mathcal{B}$ is tactically likely with evaluation 520 (equating to prizes $\{1,6,3\}$). In the case of $\mathcal{B}\odot 5$, the best response tactic for player $\mathcal{A}$ is $\mathcal{A}\odot 1$. Table 6.2(e) illustrates the resulting PRIZE-GUARANTEE subpaths for each player under this probe scenario. However, in the case of $\mathcal{B}\odot 2$, any of $\mathcal{A}\odot 1$, $\mathcal{A}\odot 3$, $\mathcal{A}\odot 4$ or $\mathcal{A}\odot 6$ is a suitable response tactic for player $\mathcal{A}$, since, in the time $d_{\mathcal{B}2}$ it takes for player $\mathcal{B}$ to reach prize 2, any of these probes takes player $\mathcal{A}$ only a short distance from its current location and only an inconsequential deviation from $\mathcal{A}\rightarrow 1$. Note that if we suppose that $\mathcal{B} \rhd \{2,4\}$ then $\mathcal{B}\odot 4$ is the implied tactic of player $\mathcal{B}$ with evaluation 471, and hence $\mathcal{A}\odot 4$ as before.

Table 6.1: Tactical Example of PRIZE-GUARANTEE



(a) Player $\mathcal{A}$ Maximal PRIZE-GUARANTEE Subpath

(b) Player $\mathcal{B}$ Maximal PRIZE-GUARANTEE Subpath

(c) Player $\mathcal{A}$ Maximal PRIZE-GUARANTEE Subpath given *prize-ts* $\mathcal{B} \rhd \{2, 4\}$

(d) Player $\mathcal{B}$ Maximal PRIZE-GUARANTEE Subpath given *prize-ts* $\mathcal{A} \rhd \{1, 4\}$

- With TARGET-SET PRIZE-PARANOID for player $\mathcal{B}$, suppose that $\mathcal{A} \rhd 1$. Then the corresponding tactic for player $\mathcal{B}$ is any of $\mathcal{B} \odot 1$, $\mathcal{B} \odot 3$, $\mathcal{B} \odot 4$ or $\mathcal{B} \odot 6$ with evaluation 343 (equating to prizes $\{4,5\}$). In each of these probe scenarios, the first prize claimed on the subsequent player $\mathcal{B}$ PRIZE-GUARANTEE subpath is prize 4 and hence the appropriate tactic for player $\mathcal{B}$ is $\mathcal{B} \odot 4$. Table 6.2(f) illustrates the resulting PRIZE-GUARANTEE subpaths for each player under the probe scenario $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 4$. Player $\mathcal{A}$'s resulting PRIZE-GUARANTEE subpath is much worse than in other scenarios since, in being constrained to target prize 1, player $\mathcal{A}$ misses out on prize 4 and then cannot subsequently guarantee both prize 6 and prize 3.

6.1.3.3 Discussion

In comparing the tactics of PRIZE-GUARANTEE and PRIZE-PARANOID, with and without a *prize-ts*, we can see from this example that both players hold to a maximum guaranteed path unless a better path becomes available due to the *prize-ts* of the opponent. The tactics implied from each player's perspective are often different in PRIZE-PARANOID.

Suppose $\mathcal{B} \rhd \{2,4\}$. In TARGET-SET PRIZE-GUARANTEE this implied that player $\mathcal{A}$ target, $\mathcal{A}\rightarrow 1$ but in TARGET-SET PRIZE-PARANOID this implied that $\mathcal{A}\rightarrow 4$. The difference is that in the former case, player $\mathcal{B}$ must commit to prize 4 in order to get to prize 3 as early as possible, but in the latter case once player $\mathcal{A}$ arrives at prize 4, the probe $\mathcal{B} \odot 4$ is completed and player $\mathcal{B}$ may then immediately target prize 3.

This tactical example is considered further in Sections 6.2.4 and 6.3.5 where the tactics are determined by tactical engines ORIGINAL-DT and PRIZE-DT.

6.1.4 Summary

We have designed the PRIZE-PARANOID tactical engine, proposed a MINIMAX evaluator for the corresponding game table with or without a *prize-ts* and presented an illustrative example. The PRIZE-PARANOID tactical engine constitutes the scenario engine component at the global-frame of the SPA/DMS. It is coupled with a PRIZE-MONITOR and STEP-MONITOR in Section 6.4 to construct a full CPCP strategy.

6.2 Tactical Engine: ORIGINAL-DT

The tactical engine ORIGINAL-DT is structured around a game tree approach in which, at each game tree node, the projection of each player towards its proposed target-prize is a *probe*.

6.2.1 Game Tree Generation

A **planning path** consists of a sequence of target-prizes. A **projected game position** consists of a pair of planning paths, P_A for player $\mathcal{A}$ and P_B for player $\mathcal{B}$; a pair of **projected locations**, A and B; a **projected time-stamp**, t; and a **projected set of remaining prizes**, $Q \subseteq V$. Let $Q_A = \{j \in Q : t + d_{Aj} \leq \lambda\}$ and $Q_B = \{j \in Q : t + d_{Bj} \leq \lambda\}$. A **game tree node** is uniquely

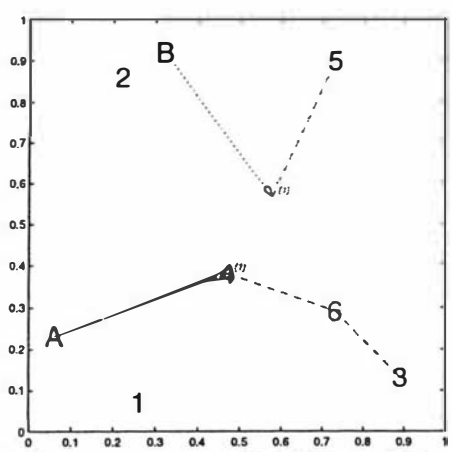
Table 6.2: Tactical Example of PRIZE-PARANOID

	1	2	3	4	5	6
1	520	520	429	429	520	429
2	327	471	137	190	355	327
3	508	520	227	318	471	227
4	471	520	**471**	**471**	471	**471**
5	327	471	327	190	471	327
6	318	520	318	318	471	318

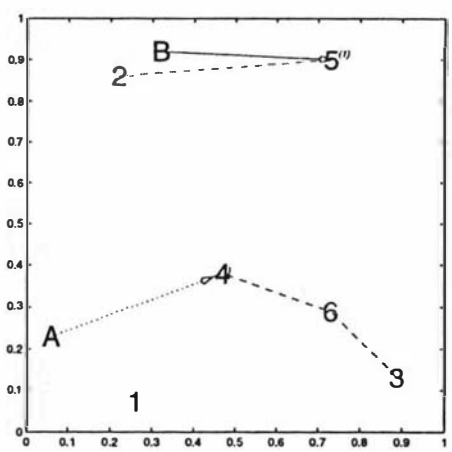
(a) Player $\mathcal{A}$ PRIZE-PARANOID Game Table

	1	2	3	4	5	6
1	343	520	492	137	429	202
2	327	327	327	**327**	327	327
3	343	471	381	190	673	190
4	343	471	343	190	471	355
5	327	508	327	**327**	190	327
6	343	471	381	190	446	290

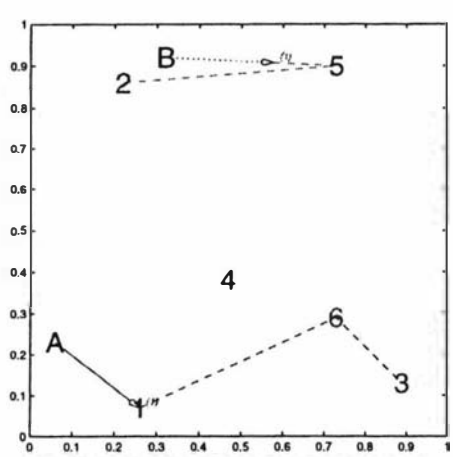
(b) Player $\mathcal{B}$ PRIZE-PARANOID Game Table



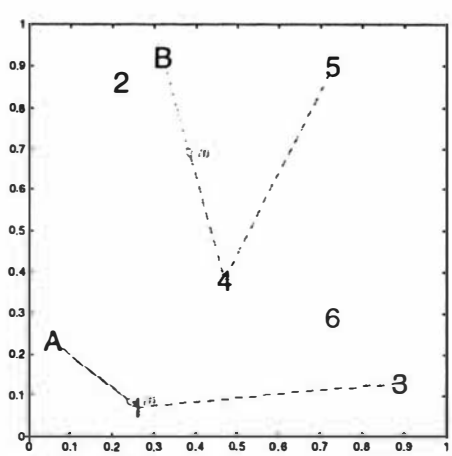
(c) Player $\mathcal{A}$'s Anticipated Tactical Game Evolution



(d) Player $\mathcal{B}$'s Anticipated Tactical Game Evolution



(e) Player $\mathcal{A}$'s Anticipated Tactical Game Evolution with *prize-ts* $\mathcal{B} \triangleright \{2, 5\}$



(f) Player $\mathcal{B}$'s Anticipated Tactical Game Evolution with *prize-ts* $\mathcal{A} \triangleright \{1\}$

defined by a game position. The game tree **root node** corresponds to the current position of the game with $P_A = \emptyset$, $P_B = \emptyset$ and $t = t_0$, where t_0 is the current time on the game clock. Also recall the specific definition of a *probe* from Section 6.1.1.

We can now define how to generate the ORIGINAL-DT game tree by expanding a given game-tree node j to define the game position corresponding to each of its children.

▼ Expansion Rule for Probe

- $\forall x \in Q_A$ and $\forall y \in Q_B$ probe $\mathcal{A} \rightarrow P_A \odot x$ and $\mathcal{B} \rightarrow P_B \odot y$.

6.2.2 Evaluating a Game Tree Node

Let j be a game tree node and let $\tilde{v}_A(j)$ be an evaluation of the prize value player $\mathcal{A}$ expects to claim from the associated game position. Let $\tilde{v}_A(j; \mathcal{A} \odot x, \mathcal{B} \odot y)$ be the evaluation of the child node of j defined by the probe $\mathcal{A} \rightarrow P_A \odot x$ and $\mathcal{B} \rightarrow P_B \odot y$.

A terminal node corresponds to a game position in which at most one prize remains unclaimed. Let $v_A(P_A, P_B)$ be the value of target-prizes on P_A claimed outright by player $\mathcal{A}$ plus half the value of those target-prizes on P_A shared with player $\mathcal{B}$. For a terminal node t we define $\tilde{v}_A(t) = v_A(P_A, P_B)$ plus whatever share of any remaining prize player $\mathcal{A}$ can claim.

For each non-terminal node j we apply one of the assumptions from Section 5.6.4.2 to define an evaluation. These are illustrated in Figure 6.2, in which a left arrow indicates minimizing across a row, an upward arrow indicates maximising within a column, and increasingly darker shading represents aggregation of corresponding maxima and minima. Compare Figure 6.2 with Figure 5.8.

MAXIMIN. Applying the *maximin assumption* (Assumption 5.6.1) gives rise to the MAXIMIN evaluation illustrated Figure 6.2(a). Let

$$\tilde{v}_A(j) = \max_{x \in Q_A} \left\{ \min_{y \in Q_B} \tilde{v}_A(j; \mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \tag{6.5}$$

Player $\mathcal{A}$ takes the initiative and determines a target-prize assuming that player $\mathcal{B}$ will respond as if player $\mathcal{B}$ *knows* which target-prize player $\mathcal{A}$ will take.

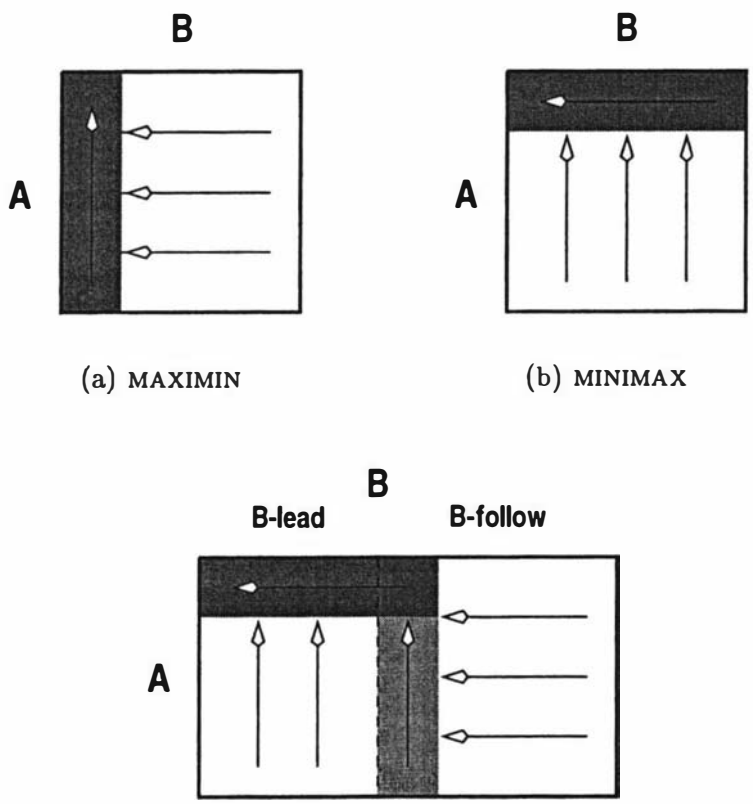
MINIMAX. Applying the *minimax assumption* (Assumption 5.6.2) gives rise to the MINIMAX evaluation illustrated in Figure 6.2(b). Let

$$\tilde{v}_A(j) = \min_{y \in Q_B} \left\{ \max_{x \in Q_A} \tilde{v}_A(j; \mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \tag{6.6}$$

Player $\mathcal{A}$ predicts a target-prize for player $\mathcal{B}$ under the assumption that player $\mathcal{A}$ can respond to whichever prize player $\mathcal{A}$ observes player $\mathcal{B}$ targeting.

MINIMAXIMIN. Applying the *minimaximin assumption* (Assumption 5.6.3) gives rise to the MINIMAXIMIN evaluation illustrated in Figure 6.2(c). Let $Q_L \subseteq Q_B$. Let

$$\tilde{v}_A(j) \quad = \quad \min \left\{ \min_{y \in Q_L} \left\{ \max_{x \in Q_A} \tilde{v}_A(j; \mathcal{A} \odot x, \mathcal{B} \odot y) \right\}, \right. \tag{6.7}$$



(a) MAXIMIN (b) MINIMAX

(c) MINIMAXIMIN

Figure 6.2: Evaluators of an ORIGINAL-DT Game Tree Node

$$\max_{x \in Q_A} \left\{ \min_{y \in Q_B \setminus Q_L} \tilde{v}_A(j; \mathcal{A} \odot x, \mathcal{B} \odot y) \right\} \Bigg\}$$

Note that Equation (6.7) generalizes Equation (6.5) (with $Q_L = \emptyset$) and Equation (6.6) (with $Q_L = Q_B$). The **lead prizes** of player $\mathcal{B}$ are those prizes of which player $\mathcal{B}$ guarantees at least a share, i.e., those prizes $\mathcal{L}_B = \{i \in Q_B : d_{Bi} \leq d_{Ai}\}$. A prize which is not a lead prize for player $\mathcal{B}$ is a **follow prize** of player $\mathcal{B}$. We can dynamically define $Q_L \subseteq Q_B$ at each game tree node j. Two possibilities include $Q_L = \mathcal{L}_B$ or $Q_L = \mathcal{L}_B \cup \{i \in Q_B : d_{Bi} \geq \delta\}$ for some threshold distance parameter δ. We implement only the simpler $Q_L = \mathcal{L}_B$.

6.2.3 Searching the Game Tree

Pearl [174] defines algorithms for searching a game tree in which the players alternate in making decisions. In this section we adapt these search algorithms for searching a game tree in which the players decide simultaneously.

We can search a game tree by working backwards from the end of the game, replacing the prospect of an obligation to play a subgame with the expected value of that subgame and then reason backwards to the present. Once each terminal node is assigned a value, a value for each non-terminal node can be determined by a bottom-up process. This method is alternatively known as **Zermelo's algorithm** (Binmore [21]) or **backwards induction** (Owen [172]). It relies on the fact that the game tree exhibits **subgame perfection**—both players know exactly the projected game position at each stage.

Pearl [174] defines various types of search procedure:

> "A **search procedure** ... is a prescription for determining the order in which nodes are to be generated. We distinguish between a **blind**, or **uninformed**, search and a **informed, guided**, or **directed** search. In the former, the order in which nodes are expanded depends only on information gathered by the search but is unaffected by the character of the unexplored portion of the graph, not even by the goal criterion. The latter uses partial information about the problem domain and about the nature of the goal to help guide the search toward the more promising directions."

We now proceed to design game tree search procedures, both blind and directed, for the ORIGINAL-DT tactical game tree applying each of the MAXIMIN, MINIMAX and MINIMAXIMIN evaluators.

6.2.3.1 Depth-First Game Tree Search

Equations (6.5)–(6.7) define MAXIMIN, MINIMAX and MINIMAXIMIN, respectively, by bottom-up induction. A straightforward translation of the definition of MAXIMIN gives an uninformed, recursive search procedure, Algorithm 6.1 EXHAUSTIVE-MAXIMIN ORIGINAL-DT, which systematically traverses every node of the game tree in a depth-first fashion. This is the "brute force" search of Knuth and Moore [131].

However, it is possible to skip descendants of a node which cannot influence the evaluation of that node, thus making possibly substantial computational savings. Knuth and Moore [131]

Algorithm 6.1 function EXHAUSTIVE-MAXIMIN ORIGINAL-DT

```
// Determine v̄_A(j).

if j is terminal then
    return(v̄_A(j))
else
    lb ← 0  // Lower bound on v(j).
    for x ∈ Q_A do
        rub ← ∞  // Upper bound on row minimum.
        for y ∈ Q_B do
            Evaluate v̄_A(j; A ⊙ x, B ⊙ y).
            rub ← min{rub, v̄(j; A ⊙ x, B ⊙ y)}
        end
        lb ← max{lb, rub}
    end
    return(lb)
end

end
```

give a concise algorithmic definition of **alpha-beta pruning**, with examples of "deep cutoffs", applications, history and analysis of expected performance. The α-β algorithm only calculates sufficient evaluation information to determine the value of the game tree root node. Define a function $\bar{v}_A(j;\alpha,\beta)$ which, given $\alpha < \beta$, returns

$$\bar{v}_A(j;\alpha,\beta) = \begin{cases} \bar{v}_A(j) & \text{if } \alpha < \bar{v}_A(j) < \beta \\ \alpha & \text{if } \bar{v}_A(j) \leq \alpha \\ \beta & \text{if } \bar{v}_A(j) \geq \beta \end{cases}$$

Algorithm 6.2 α-β-MAXIMIN ORIGINAL-DT presents the standard α-β pruning algorithm, adapted from Pearl [174, page 234]. We then evaluate the game tree root node r exactly from

$$\bar{v}_A(r) = \bar{v}_A(r;-\infty,\infty).$$

Many other game tree searching, such as SSS*, are possible, but it is beyond the scope of this thesis to compare and contrast the performance of such algorithms. However, it is worth noting that Bhattacharya and Bagchi [19] provide a unifying framework, GenGame, for search strategies such as α-β and SSS*.

6.2.3.2 Informed Game Tree Node Bounding

A lower or upper bound on $\bar{v}_A(j)$ for a game tree node j is useful since this can be used to further prune the game tree search.

Algorithm 6.2 function α-β-MAXIMIN ORIGINAL-DT

```
// Determine ṽ_A(j; α, β).
// Adapted from Pearl [174, page 234].

Input: α, β
if j is terminal then
    return(ṽ_A(j))
else
    lb ← α  // Lower bound on ṽ_A(j; α, β).
    finished ← false
    for x ∈ Q_A and not finished do
        rub ← β  // Upper bound on row minimum.
        finished_row ← false
        for y ∈ Q_B and not finished_row do
            Evaluate ṽ_A(j, A ⊙ x, B ⊙ y; lb, rub).
            rub ← min{rub, ṽ_A(j, A ⊙ x, B ⊙ y; lb, rub)}
            if (rub ≤ lb) then finished_row ← true
        end
        lb ← max{lb, rub}
        if (lb ≥ β) then finished ← true
    end
    return(lb)
end

end
```

Definition 6.2.1

The **cooperative-value**, $\Omega(j)$, of node j is the maximum joint value of the remaining prizes that the players could collect no later than the global overall deadline λ if both players cooperate perfectly from the game position associated with j. □

If $\lambda = \infty$ then the *cooperative-value* is just the value sum of the remaining prizes.

Definition 6.2.2

The **paranoid-value**, $\Gamma_A(j)$, of player $\mathcal{A}$ at a game tree node j is the maximum value of a guaranteed subpath through prizes in Q_A from the game position associated with j. Similarly for $\Gamma_B(j)$. □

The problem of determining $\Omega(j)$ is called the COOPERATE subproblem and that of determining $\Gamma_A(j)$ or $\Gamma_B(j)$ is called the PARANOID subproblem. Exact Branch and Bound algorithms for solving these subproblems are presented in Appendix 6.A.2.

Then $v_A(P_A, P_B) + \Gamma_A(j)$ is a lower bound on $\bar{v}_A(j)$ and $v_A(P_A, P_B) + \Omega(j) - \Gamma_B(j)$ is an upper bound on $\bar{v}_A(j)$. Algorithm 6.3 NODE-BOUNDED α-β-MAXIMIN ORIGINAL-DT improves on the standard α-β pruning algorithm by employing these bounds at each node examined. Similar algorithms to these can be derived for MINIMAX and MINIMAXIMIN. Algorithm 6.4 NODE-BOUNDED α-β-MINIMAX ORIGINAL-DT and Algorithm 6.5 NODE-BOUNDED α-β-MINIMAXIMIN ORIGINAL-DT presents the MINIMAX and MINIMAXIMIN equivalents, respectively, of Algorithm 6.3.

6.2.3.3 Heuristically Guided Search

A **frontier node** is a game tree node which is either terminal or is non-terminal but none of whose child nodes are generated by a particular game tree search algorithm. The number of frontier nodes that are generated, examined, and evaluated has become a standard measure of the complexity of game searching procedures (Pearl [174]).

Each of the game tree search algorithms we have presented contain phrases like "for $x \in Q_A$" and "for $y \in Q_B$"and "for $y \in Q_L$ and "for $y \in Q_B - Q_L$". The efficiency of Algorithms 6.2–6.5 depends on the order in which the nodes are evaluated. We wish to explore first those branches of the game tree whose value effectively prunes having to search large sections of the game tree. A heuristic can be used to determine which child game tree node to generate next. We sort the prizes in Q_A by minimizing a *score* σ_{Aj} on each prize $j \in Q_A$ and similarly for Q_B. Some possible scoring functions include:

(i). Nearest Neighbour: $\sigma_{Aj} \leftarrow t + d_{Aj}$

(ii). Earliest Deadline: $\sigma_{Aj} \leftarrow \begin{cases} t + d_{Bj} & \text{if } j \in Q_B \\ \lambda & \text{if } j \notin Q_B \end{cases}$

(iii). Loosest Deadline: $\sigma_{Aj} \leftarrow \begin{cases} d_{Aj} - d_{Bj} & \text{if } j \in Q_B \\ t + d_{Aj} - \lambda & \text{if } j \notin Q_B \end{cases}$

Algorithm 6.3 function NODE-BOUNDED α-β-MAXIMIN ORIGINAL-DT

```
// Determine ṽ_A(j;α,β).

if j is terminal then
    return(ṽ_A(j))
end
Evaluate Γ_A(j), Γ_B(j) and Ω(j).
α' ← max{α, v_A(P_A,P_B) + Γ_A(j)}
if (α' ≥ β) then
    return(β)
end
β' ← min{β, v_A(P_A,P_B) + Ω(j) − Γ_B(j)}
if (β' ≤ α') then
    return(α')
end
finished ← false
lb ← α'  // Lower bound on ṽ_A(j;α,β).
for x ∈ Q_A and not finished do
    rub ← β'  // Upper bound on row minimum.
    finished_row ← false
    for y ∈ Q_B and not finished_row do
        Evaluate ṽ_A(j; A ⊙ x, B ⊙ y; lb, rub).
        rub ← min{rub, ṽ_A(j; A ⊙ x, B ⊙ y; lb, rub)}
        if (rub ≤ lb) then finished_row ← true
    end
    lb ← max{lb, rub}
    if (lb ≥ β') then finished ← true
end
return(lb)

end
```

Algorithm 6.4 function NODE-BOUNDED α-β-MINIMAX ORIGINAL-DT

```
// Determine v̄_A(j; α, β).

if j is terminal then
    return(v̄_A(j))
end
Evaluate Γ_A(j), Γ_B(j) and Ω(j).
α' ← max{α, v_A(P_A, P_B) + Γ_A(j)}
if (α' ≥ β) then
    return(β)
end
β' ← min{β, v_A(P_A, P_B) + Ω(j) − Γ_B(j)}
if (β' ≤ α') then
    return(α')
end
finished ← false
ub ← β'  // Upper bound on v̄_A(j; α, β).
for y ∈ Q_B and not finished do
    clb ← α'  // Lower bound on column maximum.
    finished_col ← false
    for x ∈ Q_A and not finished_col do
        Evaluate v̄_A(j; A ⊙ x, B ⊙ y; clb, ub).
        clb ← max{clb, v̄_A(j; A ⊙ x, B ⊙ y; clb, ub)}
        if (clb ≥ ub) then finished_col ← true
    end
    ub ← min{ub, clb}
    if (ub ≤ α') then finished ← true
end
return(ub)

end
```

Algorithm 6.5 function NODE-BOUNDED α-β-MINIMAXIMIN ORIGINAL-DT

```
// Determine ṽ_A(j; α, β).

if j is terminal then
    return(ṽ_A(j))
end
Evaluate Γ_A(j), Γ_B(j) and Ω(j).
α' ← max{α, v_A(P_A, P_B) + Γ_A(j)}
if (α' ≥ β) then
    return(β)
end
β' ← min{β, v_A(P_A, P_B) + Ω(j) − Γ_B(j)}
if (β' ≤ α') then
    return(α')
end
finished ← false
ub ← β'  // Upper bound on ṽ_A(j; α, β).
for y ∈ Q_L and not finished do
    clb ← α'  // Lower bound on column maximum.
    finished_col ← false
    for x ∈ Q_A and not finished_col do
        Evaluate ṽ_A(j; A ⊙ x, B ⊙ y; clb, ub).
        clb ← max{clb, ṽ_A(j; A ⊙ x, B ⊙ y; clb, ub)}
        if (clb ≥ ub) then finished_col ← true
    end
    ub ← min{ub, clb}
    if (ub ≤ α') then finished ← true
end
if (not finished and Q_L ⊂ Q_B) then
    lb ← α'  // Lower bound on block maximum.
    for x ∈ Q_A and not finished do
        rub ← ub  // Upper bound on row minimum.
        finished_row ← false
        for y ∈ Q_B − Q_L and not finished_row do
            Evaluate ṽ_A(j; A ⊙ x, B ⊙ y; lb, rub).
            rub ← min{rub, ṽ_A(j; A ⊙ x, B ⊙ y; lb, rub)}
            if (rub ≤ lb) then finished_row ← true
        end
        lb ← max{lb, rub}
        if (lb ≥ ub) then finished ← true
    end
    ub ← min{ub, lb}
end
return(ub)

end
```

(iv). Tightest Deadline: $\sigma_{Aj} \leftarrow \begin{cases} d_{Bj} - d_{Aj} & \text{if } j \in Q_B \\ \lambda - t - d_{Aj} & \text{if } j \notin Q_B \end{cases}$

(v). Value: $\sigma_{Aj} \leftarrow v_j$.

(vi). Weighted Nearest Neighbour and Tightest Deadline: for $0 \leq \omega \leq 1$

$$\sigma_{Aj} \leftarrow \begin{cases} \omega(t + d_{Aj}) + (1 - \omega)(d_{Bj} - d_{Aj}) & \text{if } j \in Q_B \\ \omega(t + d_{Aj}) + (1 - \omega)(\lambda - t - d_{Aj}) & \text{if } j \notin Q_B \end{cases}$$

We have chosen only to implement the *weighted nearest neighbour and tightest deadline* option with $\omega = \frac{1}{2}$ since "nearest neighbour" is indicative of efficient routing and "tightest deadline" implies that some prizes are more urgent than others for guaranteed collection. In particular we do not consider *value* as a sufficient score since value tactically implies neither efficient routing nor urgency. Note that the selection affects game-tree search *efficiency* but not effectiveness of the strategy.

6.2.4 Example Tactical Problem Revisited

Section 6.1.3 compared the tactics determined by PRIZE-GUARANTEE and PRIZE-PARANOID on the example problem of Figure 6.1. We now consider the tactics determined by ORIGINAL-DT for the same example problem.

6.2.4.1 Analysis of Player $\mathcal{A}$'s ORIGINAL-DT Tactics

Table 6.3(a) gives the ORIGINAL-DT root game table for player $\mathcal{A}$, recursively employing the MINIMAXIMIN evaluator throughout. Applying the MINIMAXIMIN evaluator implies $\mathcal{B} \odot 2$, with evaluation 529 (equating to prizes $\{1, 2, 5\}$). The best response tactic for player $\mathcal{A}$ is $\mathcal{A} \odot 2$ or $\mathcal{A} \odot 5$. Table 6.4(a) illustrates the scenario $\mathcal{A} \odot 2$ and $\mathcal{B} \odot 2$.

Consider further the scenario $\mathcal{A} \odot 2$ and $\mathcal{B} \odot 2$. Table 6.3(c) gives the corresponding ORIGINAL-DT game table for player $\mathcal{A}$. A MINIMAXIMIN analysis implies $\mathcal{B} \odot 5$, with evaluation 545 (equating to prizes $\{1, 4, 5\}$), and hence $\mathcal{A} \odot 1$. This scenario is illustrated in Table 6.4(c).

At the third depth of recursion, consider the scenario $\mathcal{A} \odot 2 \odot 1$ and $\mathcal{B} \odot 2 \odot 5$. Table 6.3(e) gives the corresponding ORIGINAL-DT game table for player $\mathcal{A}$. A MINIMAXIMIN analysis implies $\mathcal{B} \rhd \{4, 6, 3\}$ from the "MAXIMIN" section of the game table. Hence, player $\mathcal{A}$'s best lead is $\mathcal{A} \odot 4$, assuming that $\mathcal{B} \odot 6$ or $\mathcal{B} \odot 3$ with evaluation 343 (equating to prizes $\{4, 5\}$). The latter scenario is illustrated in Table 6.4(e).

At the fourth depth of recursion, consider the scenario $\mathcal{A} \odot 2 \odot 1 \odot 4$ and $\mathcal{B} \odot 2 \odot 5 \odot 3$. Table 6.3(g) gives the corresponding ORIGINAL-DT game table for player $\mathcal{A}$. A MINIMAXIMIN analysis implies $\mathcal{B} \odot 6$ with evaluation 190 (equating to prize 5). The double bar at the right hand side of the game table indicates that all the remaining prizes are currently closer to player $\mathcal{B}$ and hence the evaluation is MINIMAX only. Any of $\mathcal{A} \odot 6$, $\mathcal{A} \odot 3$ or $\mathcal{A} \odot 5$ is sufficient to claim prize 5 and hence we select $\mathcal{A} \rightarrow 5$.

In summary, the 529 evaluation of $\mathcal{B} \odot 2$ and $\mathcal{A} \odot 2$ is derived from the analysis above with player $\mathcal{A}$ collecting prizes $\{1, 4, 5\}$.

Table 6.3: Tactical Example of ORIGINAL-DT MINIMAXIMIN Game Tables

	2	5	4	6	1	3
1	520	682	673	657	545	657
4	520	529	661	661	661	661
6	520	529	645	673	508	520
3	520	529	520	554	554	554
2	**529**	492	529	673	480	529
5	**529**	529	529	508	508	520

(a) Player $\mathcal{A}$ ORIGINAL-DT Root Game Table

	1	4	6	3	2	5
2	480	673	673	673	645	520
5	480	529	529	529	545	529
4	**571**	529	508	520	661	661
6	**571**	529	508	554	529	673
1	480	529	508	492	520	545
3	**571**	529	508	619	529	673

(b) Player $\mathcal{B}$ ORIGINAL-DT Root Game Table

	5	4	6	1	3
1	**545**	545	434	520	434
4	471	661	661	661	661
6	471	508	508	434	508
3	471	508	446	508	508
5	471	392	343	343	520

(c) $\mathcal{A} \odot 2$ and $\mathcal{B} \odot 2$

	3	4	6	2	5
2	327	434	471	471	471
4	471	471	471	471	471
5	471	327	471	471	471
6	**471**	480	480	471	471
3	417	480	417	471	471

(d) $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 4$

	5	4	6	3
4	471	471	**343**	**343**
6	471	318	343	434
3	380	318	380	434
5	471	190	343	318

(e) $\mathcal{A} \odot 2 \odot 1$ and $\mathcal{B} \odot 2 \odot 5$

	3	4	5	2
4	**364**	480	480	480
5	343	327	380	327
2	**364**	327	380	417
3	343	380	364	364

(f) $\mathcal{A} \odot 1 \odot 3$ and $\mathcal{B} \odot 4 \odot 6$

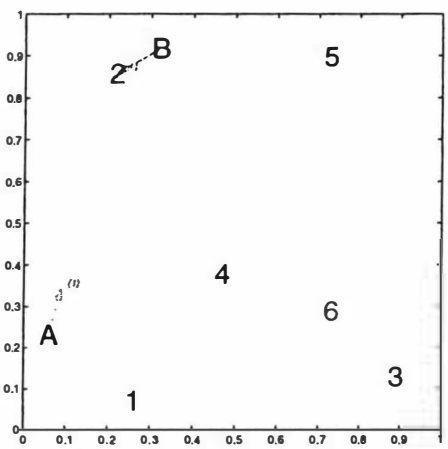
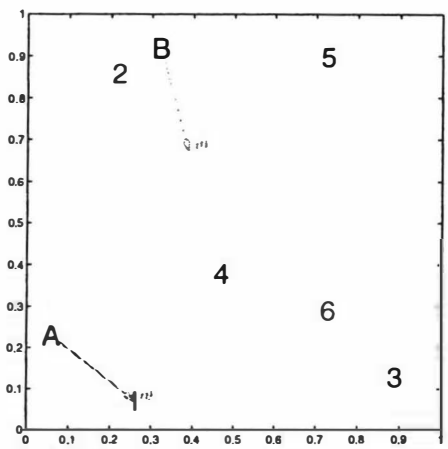
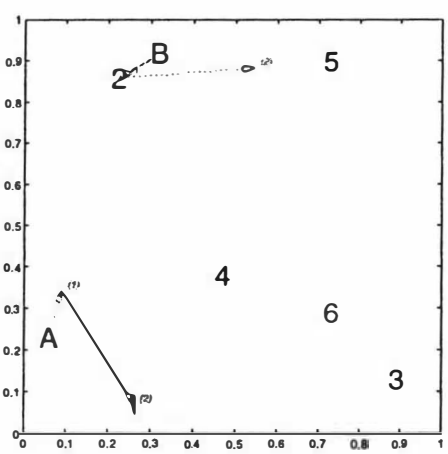
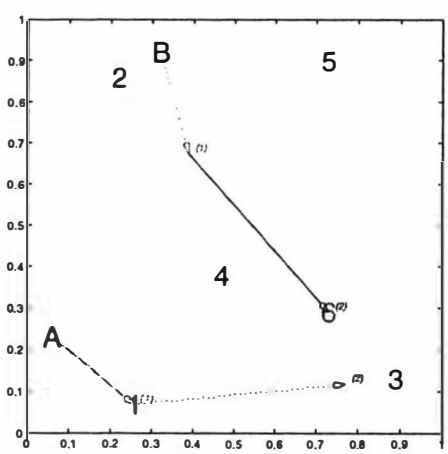
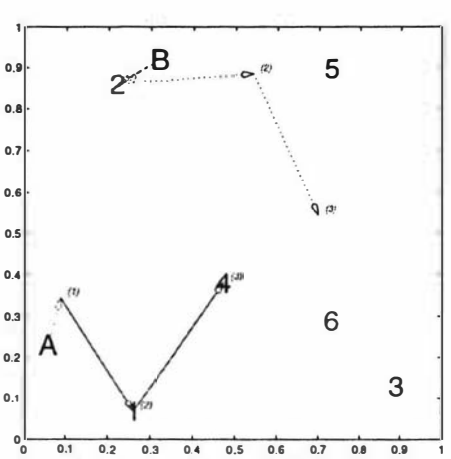
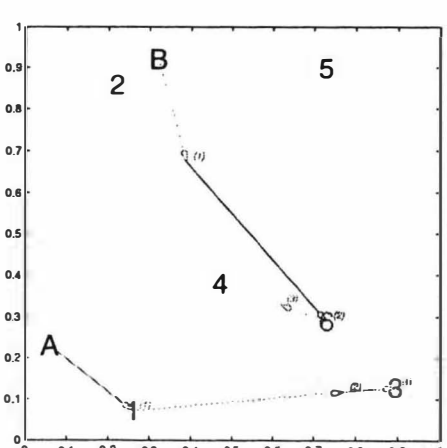
	6	5	3
6	190	318	318
3	190	318	281
5	**190**	318	190

(g) $\mathcal{A} \odot 2 \odot 1 \odot 4$ and $\mathcal{B} \odot 2 \odot 5 \odot 3$

	4	5	2
4	343	343	**343**
5	190	327	327
2	190	290	290

(h) $\mathcal{A} \odot 1 \odot 3 \odot 3$ and $\mathcal{B} \odot 4 \odot 6 \odot 4$

Table 6.4: Tactical Example of ORIGINAL-DT

Player $\mathcal{A}$'s Anticipated Tactical Game Evolution	Player $\mathcal{B}$'s Anticipated Tactical Game Evolution
	
(a) Initial Probe $\mathcal{A} \odot 2$ and $\mathcal{B} \odot 2$	(b) Initial Probe $\mathcal{A} \odot 2$ and $\mathcal{B} \odot 2$
	
(c) Second Probe $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 5$	(d) Second Probe $\mathcal{A} \odot 3$ and $\mathcal{B} \odot 6$
	
(e) Third Probe $\mathcal{A} \odot 4$ and $\mathcal{B} \odot 3$	(f) Third Probe $\mathcal{A} \odot 3$ and $\mathcal{B} \odot 4$

6.2.4.2 Analysis of Player $\mathcal{B}$'s ORIGINAL-DT Tactics

Consider now the tactics implied by ORIGINAL-DT from player $\mathcal{B}$'s perspective. Table 6.3(b) gives the ORIGINAL-DT root game table for player $\mathcal{B}$. Applying the MINIMAXIMIN evaluator implies $\mathcal{A} \odot 1$, with evaluation 571 (equating to prizes $\{2, 4, 5, 6\}$). Any of $\mathcal{B} \odot 4$, $\mathcal{B} \odot 6$ or $\mathcal{B} \odot 3$ is the best response for player $\mathcal{A}$, each of which would then turn to $\mathcal{A} \rightarrow 4$. Hence $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 4$ is the tactical scenario determined by player $\mathcal{B}$, as illustrated in Table 6.4(b).

Consider further the scenario $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 4$. Table 6.3(d) gives the corresponding ORIGINAL-DT game table for player $\mathcal{B}$. A MINIMAXIMIN analysis offers very little tactical insight, since almost all possibilities are evaluated to 471 (equating to prizes $\{3, 4, 6\}$). This shows that the ORIGINAL-DT determined tactics may not always be particularly useful. The problem appears to be that player $\mathcal{B}$ has four lead prizes but cannot sufficiently comprehend the possible responses of the player $\mathcal{A}$ whose subsequent probes function to establish position rather than to claim prizes. For the sake of continuation, suppose that the highlighted scenario $\mathcal{A} \odot 3$ and $\mathcal{B} \odot 6$ is selected since player $\mathcal{B}$ claims a prize in this scenario. This is illustrated in Table 6.4(d).

At the third depth of recursion, consider the scenario $\mathcal{A} \odot 1 \odot 3$ and $\mathcal{B} \odot 4 \odot 6$. Table 6.3(f) gives the corresponding ORIGINAL-DT game table for player $\mathcal{B}$. A MINIMAXIMIN analysis implies $\mathcal{A} \odot 3$ and hence the best response tactic for player $\mathcal{B}$ is $\mathcal{B} \odot 4$ or $\mathcal{B} \odot 2$. The former scenario is illustrated in Table 6.4(f).

At the fourth depth of recursion, consider the scenario $\mathcal{A} \odot 1 \odot 3 \odot 3$ and $\mathcal{B} \odot 4 \odot 6 \odot 4$. Table 6.3(h) gives the corresponding ORIGINAL-DT game table for player $\mathcal{B}$. The double bar at the left hand side of the game table indicates that all the remaining prizes are currently close to player $\mathcal{B}$ and hence the evaluation is MAXIMIN only. A subsequent MAXIMIN analysis implies player $\mathcal{B}$ leads $\mathcal{B} \odot 4$ but cannot distinguish the likely response of player $\mathcal{A}$.

In summary, 571 evaluation of $\mathcal{A} \odot 1$ and $\mathcal{B} \odot 4$ is derived from the analysis above with player $\mathcal{B}$ collecting prizes $\{4, 5, 6\}$.

6.2.4.3 Discussion

It is difficult to compare these tactics to those implied by PRIZE-GUARANTEE and PRIZE-PARANOID. However, the planning for player $\mathcal{B}$ is certainly more insightful using PRIZE-PARANOID in that because player $\mathcal{A}$ must play to protect prize 3, player $\mathcal{B}$ can get prizes 4 and 6. Or player $\mathcal{B}$ can get 6 and 3 if player $\mathcal{A}$ takes 1 and 4. Hence there is more of a tactical comparison which incorporates tactics for an end-game, unlike the previous methods, which were overly conservative in the end-game.

6.2.5 Summary

We have designed the ORIGINAL-DT tactical engine, and proposed MINIMAX, MAXIMIN and MINIMAXIMIN evaluators for the corresponding game tables. We have also compared the tactics determined by ORIGINAL-DT with those determined by PRIZE-GUARANTEE and PRIZE-PARANOID on the same illustrative example.

6.3 Tactical Engine: PRIZE-DT

We have seen in the design of THREE-PRIZE-DT in Section 5.6.4.3 that it is reasonable for player $\mathcal{A}$ to consider a tactically richer set of targeting options than targeting only a prize. These tactical targeting options include the following.

(i). Those prizes of which player $\mathcal{A}$ guarantees a share.

(ii). Those pairs of prizes which are not guaranteed to player $\mathcal{A}$ but for which player $\mathcal{B}$ cannot guarantee a sequence of both prizes. These provide one-prize-contingent look-ahead.

(iii). Those pairs of prizes which are not guaranteed to player $\mathcal{A}$ but which, if player $\mathcal{B}$ were to commit to a distinct third prize, player $\mathcal{B}$ could not thereby guarantee a sequence of the pair of prizes by playing a contingent strategy, as in the window feasibility subproblem of Section 5.2. These triples of prizes provide two-prize-contingent look-ahead.

Player $\mathcal{B}$ also has a similar set of targeting options.

The tactical engine PRIZE-DT is structured around a game tree approach in which, at each game tree node, the players select from the three types of options and the projection of each player through its proposed target is a *commitment*.

6.3.1 Definitions of Tactical Commitment

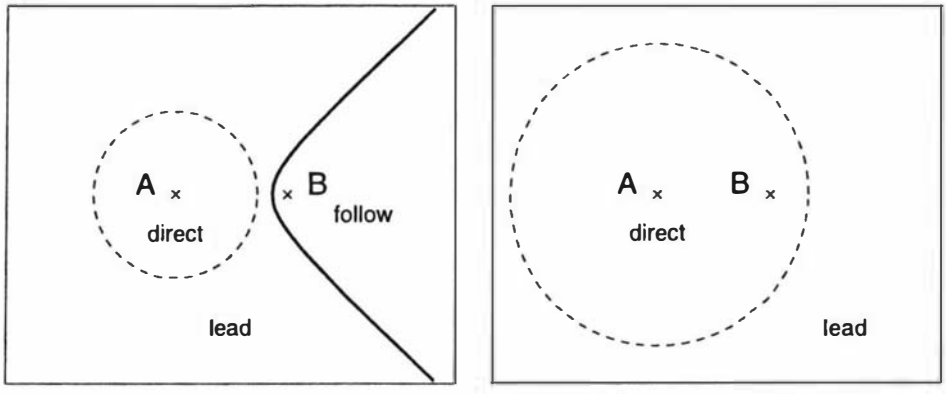
A **planning path** consists of a sequence of prize-targets, possibly interspersed with *feasibility-windows* with at most one feasibility-window between each pair of prizes. A **projected game position** consists of a pair of planning paths, P_A for player $\mathcal{A}$ and P_B for player $\mathcal{B}$; a pair of **projected locations**, A and B; a pair of **projected time-stamps**, t_A and t_B; and a **projected set of remaining prizes**, $Q \subseteq V$. Let $Q_A = \{j \in Q : t_A + d_{Aj} \leq \lambda\}$ and $Q_B = \{j \in Q : t_B + d_{Bj} \leq \lambda\}$. A **game tree node** is uniquely defined by a game position plus a *targeting restriction* on at most one of the players. The game tree **root node** corresponds to the current position of the game with $P_A = \emptyset$, $P_B = \emptyset$ and $t_A = t_B = t_0$ where t_0 is the current time on the game clock.

A feature of PRIZE-DT is that at each game tree node j (other than the root node) we do not necessarily have $t_A = t_B$, i.e., the players will usually have different time-stamps, since we are considering that the players complete their commitment to some target. Contrast this with ORIGINAL-DT in which there is a single time-stamp for each game tree node. If $t_A < t_B$ then at time t_A player $\mathcal{B}$ is *committed* to targeting location B. Similarly, if $t_A > t_B$ then at time t_B player $\mathcal{A}$ is *committed* to targeting location A.

6.3.1.1 Lead Prizes and Commitment

Consider the first targeting option of player $\mathcal{A}$ introduced at the very beginning of Section 6.3: *those prizes of which player $\mathcal{A}$ guarantees a share*, i.e., $\{i \in Q_A : t_A + d_{Ai} \leq t_B + d_{Bi}\}$.

Suppose $t_A = t_B$. Choose $x \in Q_A$ such that $t_B < t_A + d_{Ax} \leq t_B + d_{Bx}$ and suppose that player $\mathcal{A}$ targets prize x. Also, choose $y \in Q_B$ such that $t_A + d_{Ay} \geq t_B + d_{By} > t_A$ and suppose that player $\mathcal{B}$ targets prize y. Let A' be the location of prize x. Let B' be the location



(a). $0 < t_B - t_A < d_{AB}$ (b). $t_B - t_A \geq d_{AB}$

Figure 6.3: Direct-lead, lead and follow subregions for player $\mathcal{A}$.

of prize y. Let $t'_A = t_A + d_{Ax}$. Let $t'_B = t_B + d_{By}$. Let $Q' = Q \setminus \{x, y\}$. The scenario in which $\mathcal{A} \triangleright A'$ and $\mathcal{B} \triangleright B'$ is another type of **commitment** called a **lead-lead**. In general, let $\mathcal{L}_A = \{i \in Q_A : t_B < t_A + d_{Ai} \leq t_B + d_{Bi}\}$ be the **lead prizes** of player $\mathcal{A}$ and let $\mathcal{L}_B = \{i \in Q_B : t_A < t_B + d_{Bi} \leq t_A + d_{Ai}\}$ be the **lead prizes** of player $\mathcal{B}$. For $x \in \mathcal{L}_A$ and $y \in \mathcal{L}_B$, the notation[2] "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$" defines a new game position in which prize x is appended to P_A, prize y is appended to P_B, $A \leftarrow A'$, $B \leftarrow B'$, $t_A \leftarrow t'_A$, $t_B \leftarrow t'_B$ and $Q \leftarrow Q'$.

Suppose $t_A < t_B$. Choose $x \in Q_A$ such that $t_A + d_{Ax} \leq t_B$, and suppose that player $\mathcal{A}$ targets prize x. Let A' be the location of prize x. Let $t'_A = t_A + d_{Ax}$. Let $t'_B = t_B + d_{By}$. Let $Q' = Q \setminus \{x\}$. The scenario in which $\mathcal{A} \triangleright A'$, to work towards catching up to player $\mathcal{B}$'s time stamp, is one type of **commitment** called a **direct-lead**. In general, let $\mathcal{D}_A = \{i \in Q_A : t_A + d_{Ai} \leq t_B\}$ be set of **direct lead prizes** of player $\mathcal{A}$. For $x \in \mathcal{D}_A$, the notation "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B$" defines a new game position in which prize x is appended to P_A, $A \leftarrow A'$, $t_A \leftarrow t'_A$ and $Q \leftarrow Q'$. Only player $\mathcal{A}$ selects a target since it is unreasonable that player $\mathcal{A}$ would be able to select another target give more information than player $\mathcal{B}$ at time t_B.

Suppose $t_A > t_B$. Let $\mathcal{D}_B = \{i \in Q_B : t_B + d_{Bi} \leq t_A\}$ be the set of **direct lead prizes** of player $\mathcal{B}$. For $y \in \mathcal{D}_B$, we can similarly define the notation "**commit** $\mathcal{A} \rightarrow P_A$ and $\mathcal{B} \rightarrow P_B \oplus y$".

The other prizes available to player $\mathcal{A}$ are $\mathcal{R}_A = Q_A - (\mathcal{D}_A \cup \mathcal{L}_A)$. These are the **follow-prizes** of player $\mathcal{A}$. The **follow-prizes** of player $\mathcal{B}$ are $\mathcal{R}_B = Q_B - (\mathcal{D}_B \cup \mathcal{L}_B)$. Figure 6.3 classifies the three types of prize for player $\mathcal{A}$ (direct lead, lead and follow) by prize location where $t_A < t_B$. The dashed circle encloses those prize locations $i \in Q_A$ such that $t_A + d_{Ai} \leq t_B$, i.e., the set $\mathcal{D}_A$. In Figure 6.3(a) the solid curve is one half of a hyperbola such that a point P on the curve satisfies $t_A + d_{AP} = t_B + d_{BP}$. Hence, the curve partitions the prizes into $\mathcal{D}_A \cup \mathcal{L}_A$ on one side and $\mathcal{R}_A$ on the other. In Figure 6.3(b) we have $t_B - t_A \geq d_{AB}$ and hence $\mathcal{R}_A = \emptyset$.

[2]Note that $\oplus$ is used to define a *commitment* whereas $\odot$ was used in Section 6.2 to define a *probe*.

6.3.1.2 Follow Pairs, Feasibility Windows and Commitment

Consider the other two targeting options of player $\mathcal{B}$ introduced at the beginning of Section 6.3: *those pairs of prizes which are not guaranteed to player* $\mathcal{B}$.

Choose $x \in Q_A$ such that $t_B < t_A + d_{Ax} \leq t_B + d_{Bx}$ and suppose that player $\mathcal{A}$ targets prize x. Also, choose $y_1, y_2 \in Q_B$ such that $y_1 \neq y_2$, $t_A + d_{Ay_1} < t_B + d_{By_1}$ and $t_A + d_{Ay_2} < t_B + d_{By_2}$, and suppose that player $\mathcal{B}$ is committed to attempting to claim one of prizes y_1 and y_2, i.e., $\mathcal{B} \rhd \{y_1, y_2\}$. Let A' be the location of prize x. Let $t'_A = t_A + d_{Ax}$. Let $Q' = Q \setminus \{x\}$.

If $x \in \{y_1, y_2\}$, then let W be the default window. If $x \notin \{y_1, y_2\}$, then we have the window scenario in which X is the location of prize x, $t_X = t_A + d_{Ax}$ and $\mathcal{B} \rhd \{y_1, y_2\}$, as in Section 5.2. Let W be the corresponding feasibility window.

The scenario in which $\mathcal{A} \rhd A'$ and $\mathcal{B} \rhd \{y_1, y_2\}$ is another type of **commitment** called a **lead-follow**. In general, let $\mathcal{F}_A = \{\{x_1, x_2\} : x_1, x_2 \in \mathcal{R}_A, x_1 \neq x_2\}$ be the **follow-pairs** of player $\mathcal{A}$ and let $\mathcal{F}_B = \{\{y_1, y_2\} : y_1, y_2 \in \mathcal{R}_B, y_1 \neq y_2\}$ be the **follow-pairs** of player $\mathcal{B}$. For $x \in \mathcal{L}_A$ and $\{y_1, y_2\} \in \mathcal{F}_B$, the notation "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \rhd (W \rightarrow \{y_1, y_2\})$" defines a new game position in which prize x is appended to P_A, $A \leftarrow A'$, $t_A \leftarrow t'_A$, $Q \leftarrow Q'$ and player $\mathcal{B}$ is **restricted** to target only prize y_1 or y_2 and be forced to move through the window W. This is the only type of *targeting restriction* we consider. Note that *neither* y_1 nor y_2 is appended to P_B.

Suppose $t_A > t_B$ and $\mathcal{B} \rhd \{y_1, y_2\}$ for some $\{y_1, y_2\} \in \mathcal{F}_B$. This is also a window scenario in which $X = A$ and $t_X = t_A$. The corresponding feasibility window is called an **early feasibility window**, since player $\mathcal{B}$ arrives at its projected location B earlier than player $\mathcal{A}$ arrives at its projected location A and hence must satisfy a feasibility window at time t_A, regardless of what player $\mathcal{A}$ targets at time t_A. We denote the **early window scenario** by $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \{y_1, y_2\}$.

For these two window scenarios, we now define whether the window scenario is *window-feasible.*

▼ $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \{i_1, i_2\}$

Let $X = A$, $t_X = t_B$, $Y = B$ and $t_Y = t_B$. If $t_Y \geq t_X$ then we require that constraints (6.8)–(6.9) hold.

$$t_X + d_{Xi_1} + d_{i_1 i_2} > t_Y + d_{Yi_2} \tag{6.8}$$

$$t_X + d_{Xi_2} + d_{i_1 i_2} > t_Y + d_{Yi_1} \tag{6.9}$$

If $t_Y < t_X$ then we wish to determine if there exists a *feasibility window* exactly as in Section 5.2 by applying Algorithm 5.1 WINDOW FEASIBLE. Hence, we make the following definition.

Definition 6.3.1

The window scenario $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \{i_1, i_2\}$ is **window-feasible** if either

- $t_Y \geq t_X$ and constraints (6.8)–(6.9) hold; or
- $t_Y < t_X$ and there exists a location Z satisfying constraints (5.29)–(5.31).

□

▼ $\mathcal{A} \triangleright k$ and $\mathcal{B} \triangleright \{i_1, i_2\}$

This case is identical to the previous case except for the following modification. Let X be the location of prize k and let t_X be the earliest possible arrival time of player $\mathcal{A}$ at prize k.

6.3.1.3 Chains of Window Scenarios

Let W_0 be the window feasible scenario $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \{i_1, i_2\}$, with corresponding time-stamps t_A and t_B.[3] Suppose that player $\mathcal{A}$ targets the sequence of prizes $k_1, \ldots, k_m$. Let W_1 be the window scenario $\mathcal{A} \triangleright k_1$ and $\mathcal{B} \triangleright (W_0 \rightarrow \{i_1, i_2\})$. The Two Prize Theorem (Theorem 5.1.1) implies that W_1 is also window feasible. For each $2 \leq j \leq m$ let W_j be the window scenario in which player $\mathcal{A}$ has moved through the subpath $(k_1, \ldots, k_{j-1})$ and $\mathcal{A} \triangleright k_j$ and $\mathcal{B} \triangleright (W_0 \rightarrow \cdots \rightarrow W_{j-1} \rightarrow \{i_1, i_2\})$. The Two Prize Theorem implies that $W_2, \ldots, W_m$ are also window feasible. Thus, for all intents and purposes, the constraints implied by $\mathcal{B} \triangleright (W_0 \rightarrow \cdots \rightarrow W_m \rightarrow \{i_1, i_2\})$ are no more restrictive than the constraints implied by $\mathcal{B} \triangleright (W_0 \rightarrow \{i_1, i_2\})$. Hence we have proved the following Lemma.

Lemma 6.3.2

$\mathcal{B} \triangleright (W_0 \rightarrow \cdots \rightarrow W_m \rightarrow \{i_1, i_2\})$ is **equivalent** to $\mathcal{B} \triangleright (W_0 \rightarrow \{i_1, i_2\})$.

6.3.1.4 Reverse Leads and Commitment

Suppose our observations discern that player $\mathcal{B}$ is targeting a prize $y \in Q_B$ for which $d_{Ay} < d_{By}$.

Choose $x \in Q_A$ such that $t_A + d_{Ax} > t_B + d_{Bx}$ and suppose that player $\mathcal{A}$ targets prize x. Since $d_{By} + d_{yx} > d_{Ay} + d_{yx} \geq d_{Ax}$, player $\mathcal{A}$ guarantees prize x conditional upon $\mathcal{B} \triangleright y$. Let A' be the location of prize x. Let B' be the location of prize y. Let $t'_A = t_A + d_{Ax}$. Let $t'_B = t_B + d_{By}$. Let $Q' = Q \setminus \{x, y\}$. The scenario in which $\mathcal{A} \triangleright A'$ and $\mathcal{B} \triangleright B'$ is another type of **commitment** called a **reverse lead**. In general, $\mathcal{R}_A = \{i \in Q_A : t_B < t_A + d_{Ai} \leq t_B + d_{Bi}\}$ are the **reverse lead prizes** of player $\mathcal{A}$ and $\mathcal{R}_B = \{i \in Q_B : t_A < t_B + d_{Bi} \leq t_A + d_{Ai}\}$ are the **reverse lead prizes** of player $\mathcal{B}$. Note that $\mathcal{R}_A \subseteq \mathcal{D}_B \cup \mathcal{L}_B$ and $\mathcal{R}_B \subseteq \mathcal{D}_A \cup \mathcal{L}_A$.[4] For $x \in \mathcal{R}_A$ and $y \in \mathcal{R}_B$, the notation "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$" defines a new game position in which prize x is appended to P_A, prize y is appended to P_B, $A \leftarrow A'$, $B \leftarrow B'$, $t_A \leftarrow t'_A$, $t_B \leftarrow t'_B$ and $Q \leftarrow Q'$.

Intuitively we would suspect that neither player would complete a reverse lead scenario $\mathcal{A} \triangleright x$ and $\mathcal{B} \triangleright y$ for $x \in \mathcal{R}_A$ and $y \in \mathcal{R}_B$. In comparison with the lead-lead scenario $\mathcal{A} \triangleright y$ and $\mathcal{B} \triangleright x$, the players are, in a sense, switching positions and hence one of the players must be worse off in terms of its expected outcome from the two game positions. Theorem 6.A.2 of Appendix 6.A.1 shows that reverse leading cannot be optimal in the three prize problem. For problems involving more than three prizes, player $\mathcal{A}$ may find that a reverse lead scenario is the best response to player $\mathcal{B}$ targeting a prize in $\mathcal{R}_B$, although computational experience indicates that this only occurs at deep stages of the tactical game tree.

[3] We use the symbol W to stand for both the *window feasible scenario* and the corresponding *feasibility window*.

[4] Hence the term **reverse lead**.

6.3.1.5 Conditional Lead Prizes and Commitment

Prize y is a **lead prize** for player $\mathcal{B}$ **conditional upon** $\mathcal{A} \triangleright x$ if $t_B + d_{By} \leq t_A + d_{Ax} + d_{xy}$. Prize y is a **lead prize** for player $\mathcal{B}$ **conditional upon** $\mathcal{A} \triangleright x$ and $\mathcal{B} \triangleright W$ if $t_B + d_{BW} + d_{Wy} \leq t_A + d_{Ax} + d_{xy}$.

Choose $x \in \mathcal{L}_A$ and suppose that player $\mathcal{A}$ targets prize x. Suppose prize y is a lead prize for player $\mathcal{B}$ conditional upon $\mathcal{A} \triangleright x$ and that player $\mathcal{B}$ targets prize y. Let A' be the location of prize x and let B' be the location of prize y. We can apply the analysis of the *two prize subproblem* in Section 5.6.1 to conservatively estimate player $\mathcal{A}$'s arrival time at prize x as $t'_A = t_A + d_{Ax}$, and player $\mathcal{B}$'s arrival time at prize y as $t'_B = t_B + d_{By}$. Let $Q' = Q \setminus \{x, y\}$. The notation "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$" defines a new game position in which prize x is appended to P_A, prize y is appended to P_B, $A \leftarrow A'$, $B \leftarrow B'$, $t_A \leftarrow t'_A$, $t_B \leftarrow t'_B$ and $Q \leftarrow Q'$.

Choose $x \in \mathcal{L}_A$ and suppose that player $\mathcal{A}$ targets prize x. Suppose prize y is a lead prize for player $\mathcal{B}$ conditional upon $\mathcal{A} \triangleright x$ and $\mathcal{B} \triangleright W$ and that player $\mathcal{B}$ targets prize y. Let A' be the location of prize x and let B' be the location of prize y. We can apply the analysis of the *three prize window feasible subproblem* in Section 5.6.2 to conservatively estimate player $\mathcal{A}$'s arrival time at prize x as $t'_A = t_A + d_{Ax}$ and player $\mathcal{B}$'s arrival time at prize y as $t'_B = t_B + d_{BW} + d_{Wy}$. Let $Q' = Q \setminus \{x, y\}$. The notation "**commit** $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus (W \rightarrow y)$" defines a new game position in which prize x is appended to P_A, prize y is appended to P_B, $A \leftarrow A'$, $B \leftarrow B'$, $t_A \leftarrow t'_A$, $t_B \leftarrow t'_B$ and $Q \leftarrow Q'$.

6.3.2 Game Tree Generation

We now define how to generate the PRIZE-DT game tree by expanding a given game-tree node j to define the game position corresponding to each of its children. We use the notation "$\mathcal{B} \triangleright \emptyset$" to indicate that player $\mathcal{B}$ has no *targeting restrictions* at time t_B and the notation "$\mathcal{B} \triangleright (W \rightarrow \{y_1, y_2\})$ to indicate that player $\mathcal{B}$ is restricted to considering only prizes y_1 and y_2 as target prizes and must travel through the feasibility window W. Similar notation is used for player $\mathcal{A}$. The expansion rules also define a transition between the three different types of game-tree node:

(i). $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \emptyset$.

(ii). $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright (W_1 \rightarrow \{y_1, y_2\})$.

(iii). $\mathcal{A} \triangleright (W_1 \rightarrow \{x_1, x_2\})$ and $\mathcal{B} \triangleright \emptyset$.

▼ Expansion Rule for Direct-Lead and Lead-Lead Scenarios

Case (L1). $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \emptyset$

- $\forall\, x \in \mathcal{D}_A$ commit $\mathcal{A} \rightarrow P_A \oplus x$ (maintaining $\mathcal{B} \rightarrow P_B$).
- $\forall\, y \in \mathcal{D}_B$ commit $\mathcal{B} \rightarrow P_B \oplus y$ (maintaining $\mathcal{A} \rightarrow P_A$).
- $\forall\, x \in \mathcal{L}_A$ and $\forall\, y \in \mathcal{L}_B$ commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$.

▼ Expansion Rules for Lead-Follow Scenarios

Case (F1a). $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \emptyset$ where $t_A \leq t_B$.

- Choose a follow pair $\{y_1, y_2\} \in \mathcal{F}_B$.
- If y_2 is a lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd y_1$ then commit $\mathcal{A} \rightarrow P_A \oplus y_1$ and $\mathcal{B} \rightarrow P_B \oplus y_2$.
- If y_1 is a lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd y_2$ then commit $\mathcal{A} \rightarrow P_A \oplus y_2$ and $\mathcal{B} \rightarrow P_B \oplus y_1$.
- Choose an $x \in \mathcal{L}_A - \{y_1, y_2\}$.
- Let W_1 be the window scenario $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd \{y_1, y_2\})$.
- If W_1 is *window-feasible* then:
  - If $y \in \{y_1, y_2\}$ is a lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd W_1$ then commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus (W_1 \rightarrow y)$.
  - Otherwise commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow \{y_1, y_2\})$.

Case (F2a). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$ where $t_A > t_B$.

- Choose a follow pair $\{y_1, y_2\} \in \mathcal{F}_B$.
- Let W_1 be the early window scenario $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \{y_1, y_2\})$.
- If W_1 is *window-feasible* then:
  - Commit $\mathcal{A} \rightarrow P_A \oplus y_1$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow y_2)$.
  - Commit $\mathcal{A} \rightarrow P_A \oplus y_2$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow y_1)$.
  - Choose an $x \in \mathcal{L}_A - \{y_1, y_2\}$.
  - Let W_2 be the window scenario $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd (W_1 \rightarrow \{y_1, y_2\})$. Then W_2 is *window-feasible.*
  - If $y \in \{y_1, y_2\}$ is a lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd W_1$ then commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow y)$.
  - Otherwise commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow \{y_1, y_2\})$.

Case (F3a). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd (W_1 \rightarrow \{y_1, y_2\})$

- Commit $\mathcal{A} \rightarrow P_A \oplus y_1$ and $\mathcal{B} \rightarrow P_B \oplus (W_1 \rightarrow y_2)$.
- Commit $\mathcal{A} \rightarrow P_A \oplus y_2$ and $\mathcal{B} \rightarrow P_B \oplus (W_1 \rightarrow y_1)$.
- Choose an $x \in Q - \{y_1, y_2\}$.
- Let W_2 be the window scenario $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd (W_1 \rightarrow \{y_1, y_2\})$. Then W_2 is *window-feasible.*
- If $y \in \{y_1, y_2\}$ is a lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd W_1$ then commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus (W_1 \rightarrow y)$.
- Otherwise commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \rhd (W_1 \rightarrow \{y_1, y_2\})$.

Similarly we can define expansion rules for cases (F1b)–(F3b) in which the roles of the players are reversed.

▼ Expansion rule for Reverse-Lead Scenarios

Case (R1). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$

- $\forall\, x \in \mathcal{R}_A$ and $\forall\, y \in \mathcal{R}_B$ commit $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$.

6.3.3 Evaluating a Game Tree Node

We need to be able to evaluate a game tree node so as to provide an evaluation of the corresponding scenario generated from the parent game tree node.

6.3.3.1 Assumptions

When player $\mathcal{A}$ targets a prize $x \in \mathcal{L}_A$ and player $\mathcal{B}$ targets a follow pair $\{y_1, y_2\} \in \mathcal{F}_B$, player $\mathcal{A}$ is most likely to claim prize x well before player $\mathcal{B}$ can claim either prize y_1 or prize y_2. Player $\mathcal{B}$ then has more time to observe which prize player $\mathcal{A}$ is targeting and can respond by selecting an appropriate follow pair from $\mathcal{F}_B$. Hence, for the purposes of evaluation, player $\mathcal{A}$ *must* implicitly assume that, once player $\mathcal{B}$ decides to target a follow pair, player $\mathcal{B}$ knows player $\mathcal{A}$'s choice of target prize. This corresponds to a ***maximin assumption*** over the block of targeting options $\mathcal{L}_A \times \mathcal{F}_B$.

When player $\mathcal{A}$ targets a follow pair $\{x_1, x_2\} \in \mathcal{F}_A$ and player $\mathcal{B}$ targets a prize $y \in \mathcal{L}_B$, player $\mathcal{B}$ is most likely to claim prize y well before player $\mathcal{A}$ can claim either prize x_1 or x_2. Player $\mathcal{A}$ then has more time to observe which prize player $\mathcal{B}$ is targeting and can respond by selecting an appropriate follow pair from $\mathcal{F}_A$. Hence, for the purposes of evaluation, once player $\mathcal{A}$ decides to target a follow pair, player $\mathcal{A}$ *can* implicitly assume that player $\mathcal{B}$'s choice of target prize is known. This corresponds to a ***minimax assumption*** over the block of targeting options $\mathcal{F}_A \times \mathcal{L}_B$.

When player $\mathcal{A}$ targets a prize $x \in \mathcal{L}_A$ and player $\mathcal{B}$ targets a prize $y \in \mathcal{L}_B$ then, for the purposes of evaluation, there are at least two explicit assumptions we can make.

Assumption 6.3.3

The PRIZE-DT **maximin assumption** is that once player $\mathcal{A}$ decides to target a lead prize, player $\mathcal{A}$ selects the best possible target prize, given that player $\mathcal{B}$ *knows the option player $\mathcal{A}$ will select* and player $\mathcal{B}$ can select a target from either $\mathcal{L}_B$ or $\mathcal{F}_B$. □

Assumption 6.3.4

The PRIZE-DT **minimax assumption** is that once player $\mathcal{B}$ decides to target a lead prize, player $\mathcal{A}$ selects the best possible target from either $\mathcal{L}_A$ or $\mathcal{F}_A$, given that player $\mathcal{B}$ *selects an option that most restricts player $\mathcal{A}$'s result while knowing that player $\mathcal{A}$ knows which lead prize player $\mathcal{B}$ will target.* □

As discussed in Section 5.6.4.2, these assumptions are ideals. They represent two extreme possibilities which summarise the tactical information associated with a projected game position. Certainly a MAXIMIN type objective is appropriate when $\mathcal{A} \rhd x \in \mathcal{L}_A$ and $\mathcal{B} \rhd \{y_1, y_2\} \in \mathcal{F}_B$ and a MINIMAX type objective is appropriate when $\mathcal{A} \rhd \{x_1, x_2\} \in \mathcal{F}_A$ and $\mathcal{B} \rhd y \in \mathcal{L}_B$. The strategy design issue is how to evaluate the scenario in which $\mathcal{A} \rhd x \in \mathcal{L}_A$ and $\mathcal{B} \rhd y \in \mathcal{L}_B$. Hence Assumptions 6.3.3 and 6.3.4 are the most elementary possible assumptions, selecting MAXIMIN and MINIMAX based philosophies respectively.

6.3.3.2 Evaluator Definitions

Assumption 6.3.3 gives rise to the GENERALIZED-MAXIMIN evaluator which is a combination of MINIMAX and MAXIMIN over blocks of target options for each player, as in Figure 6.4(a). Assumption 6.3.4 gives rise to the GENERALIZED-MINIMAX evaluator, as in Figure 6.4(b). A left arrow indicates minimizing across a row and an upward arrow indicates maximizing within a column and increasingly dark shading implies increasing aggregation of maximum of minimum values. These two evaluators are defined explicitly below. In particular, Figure 6.4 generalizes Figure 5.10.

Let j be a game tree node and let $\tilde{v}_A(j)$ be the evaluation of the prize value player $\mathcal{A}$ expects to claim from the associated game position. Let $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y)$ be the evaluation of the child node of j defined by the commitment $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B \oplus y$. Similarly we define $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus (W \rightarrow y))$ and $\tilde{v}_A(j; \mathcal{A} \oplus (W \rightarrow x), \mathcal{B} \oplus y)$. Let $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B})$ be the evaluation of the child node of j defined by the commitment $\mathcal{A} \rightarrow P_A \oplus x$ and $\mathcal{B} \rightarrow P_B$. Similarly we define $\tilde{v}_A(j; \mathcal{A}, \mathcal{B} \oplus y)$.

■ $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \emptyset$

The following GENERALIZED-MINIMAX and GENERALIZED-MAXIMIN evaluators apply when $\mathcal{A} \triangleright \emptyset$ and $\mathcal{B} \triangleright \emptyset$.

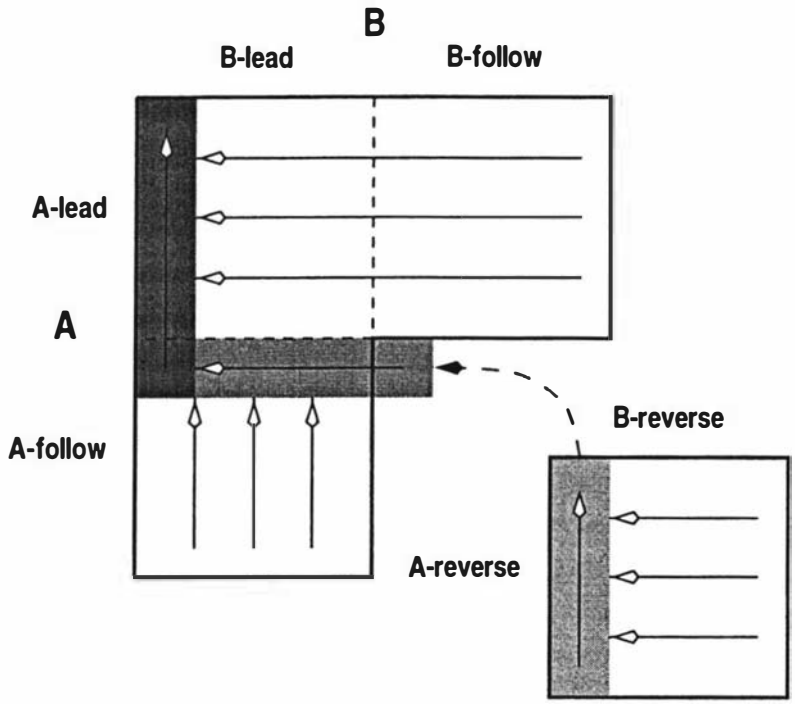
▼ GENERALIZED-MINIMAX.

If $\mathcal{F}_B \neq \emptyset$ then let

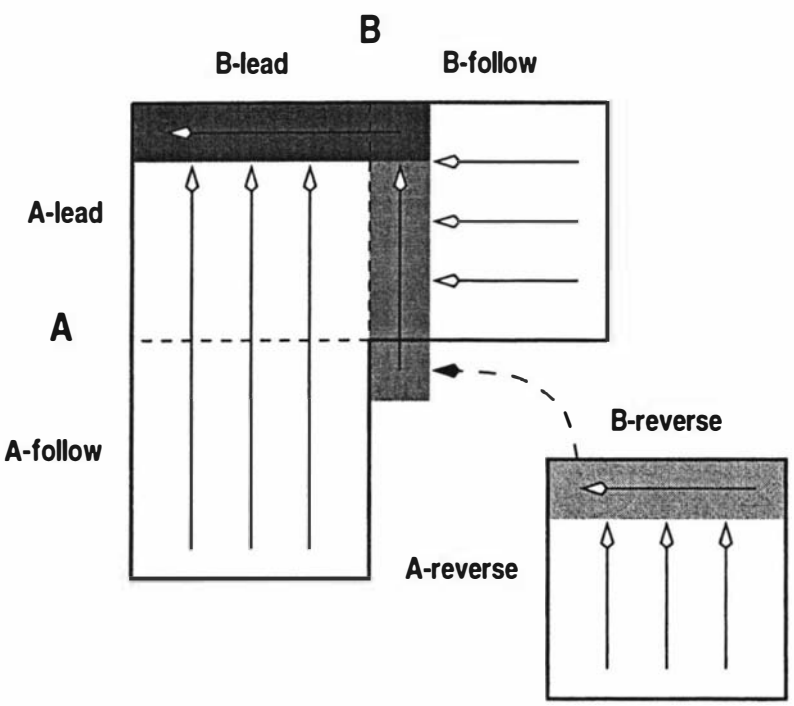
$$\begin{aligned}
\tilde{v}_A(j) \;=\; \max\Bigg\{ & \max_{x \in \mathcal{D}_A} \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B}), \\
& \min\Bigg\{ \min_{y \in \mathcal{D}_B} \tilde{v}_A(j; \mathcal{A}, \mathcal{B} \oplus y), \\
& \quad \min_{y \in \mathcal{L}_B} \Bigg\{ \max_{x \in \mathcal{L}_A} \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y), \\
& \qquad \max_{\{x_1, x_2\} \in \mathcal{F}_A} \tilde{v}_A(j; \mathcal{A} \triangleright \{x_1, x_2\}, \mathcal{B} \triangleright y) \Bigg\}, \\
& \quad \max\Bigg\{ \max_{x \in \mathcal{L}_A} \Bigg\{ \min_{\{y_1, y_2\} \in \mathcal{F}_B} \tilde{v}_A(j; \mathcal{A} \triangleright x, \mathcal{B} \triangleright \{y_1, y_2\}) \Bigg\}, \\
& \qquad \min_{y \in \mathcal{R}_B} \Bigg\{ \max_{x \in \mathcal{R}_A} \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y) \Bigg\} \Bigg\} \Bigg\} \Bigg\}
\end{aligned} \tag{6.10}$$

Otherwise $\mathcal{F}_B = \emptyset$ so let

$$\begin{aligned}
\tilde{v}_A(j) \;=\; \max\Bigg\{ & \max_{x \in \mathcal{D}_A} \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B}), \\
& \min\Bigg\{ \min_{y \in \mathcal{D}_B} \tilde{v}_A(j; \mathcal{A}, \mathcal{B} \oplus y), \\
& \quad \min_{y \in \mathcal{L}_B} \Bigg\{ \max_{x \in \mathcal{L}_A} \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y), \\
& \qquad \max_{\{x_1, x_2\} \in \mathcal{F}_A} \tilde{v}_A(j; \mathcal{A} \triangleright \{x_1, x_2\}, \mathcal{B} \triangleright y) \Bigg\} \Bigg\} \Bigg\}
\end{aligned} \tag{6.11}$$



(a) GENERALIZED-MAXIMIN



(b) GENERALIZED-MINIMAX

Figure 6.4: Alternative Evaluators for PRIZE-DT

▼ GENERALIZED-MAXIMIN.

If $\mathcal{F}_A \neq \emptyset$ then let

$$\begin{aligned}\tilde{v}_A(j) \;=\; \min\Bigg\{ & \min_{y\in\mathcal{D}_B} \tilde{v}_A(j;\mathcal{A},\mathcal{B}\oplus y),\\ & \max\Bigg\{ \max_{x\in\mathcal{D}_A} \tilde{v}_A(j;\mathcal{A}\oplus x,\mathcal{B}),\\ & \quad \max_{x\in\mathcal{L}_A}\Bigg\{ \min_{y\in\mathcal{L}_B} \tilde{v}_A(j;\mathcal{A}\oplus x,\mathcal{B}\oplus y),\\ & \qquad \min_{\{y_1,y_2\}\in\mathcal{F}_B} \tilde{v}_A(j;\mathcal{A}\rhd x,\mathcal{B}\rhd\{y_1,y_2\})\Bigg\},\\ & \quad \min\Bigg\{ \min_{y\in\mathcal{L}_B}\Bigg\{ \min_{\{x_1,x_2\}\in\mathcal{F}_A} \tilde{v}_A(j;\mathcal{A}\rhd\{x_1,x_2\},\mathcal{B}\rhd y)\Bigg\},\\ & \qquad \max_{x\in\mathcal{R}_A}\Bigg\{ \min_{y\in\mathcal{R}_B} \tilde{v}_A(j;\mathcal{A}\oplus x,\mathcal{B}\oplus y)\Bigg\}\Bigg\}\Bigg\}\Bigg\}\end{aligned} \tag{6.12}$$

Otherwise $\mathcal{F}_A = \emptyset$ so let

$$\begin{aligned}\tilde{v}_A(j) \;=\; \min\Bigg\{ & \min_{y\in\mathcal{D}_B} \tilde{v}_A(j;\mathcal{A},\mathcal{B}\oplus y),\\ & \max\Bigg\{ \max_{x\in\mathcal{D}_A} \tilde{v}_A(j;\mathcal{A}\oplus x,\mathcal{B}),\\ & \quad \max_{x\in\mathcal{L}_A}\Bigg\{ \min_{y\in\mathcal{L}_B} \tilde{v}_A(j;\mathcal{A}\oplus x,\mathcal{B}\oplus y),\\ & \qquad \min_{\{y_1,y_2\}\in\mathcal{F}_B} \tilde{v}_A(j;\mathcal{A}\rhd x,\mathcal{B}\rhd\{y_1,y_2\})\Bigg\}\Bigg\}\Bigg\}\end{aligned} \tag{6.13}$$

■ $\mathcal{A}\rhd\emptyset$ and $\mathcal{B}\rhd(W_1\to\{y_1,y_2\})$

When $\mathcal{A}\rhd\emptyset$ and $\mathcal{B}\rhd(W_1\to\{y_1,y_2\})$, player $\mathcal{B}$ essentially has no targeting choices and hence the following GENERALIZED-MAX evaluator applies.

▼ GENERALIZED-MAX.

$$\begin{aligned}&\tilde{v}_A(j;\mathcal{A}\rhd\emptyset,\mathcal{B}\rhd(W\to\{y_1,y_2\}))\\ &\quad= \max\{\tilde{v}_A(j;\mathcal{A}\oplus y_1,\mathcal{B}\oplus(W\to y_2)),\tilde{v}_A(j;\mathcal{A}\oplus y_2,\mathcal{B}\oplus(W\to y_1)),\\ &\qquad\qquad \max_{x\in Q_A\setminus\{y_1,y_2\}} \tilde{v}_A(j;\mathcal{A}\rhd x,\mathcal{B}\rhd(W\to\{y_1,y_2\}))\}\end{aligned} \tag{6.14}$$

■ $\mathcal{A}\rhd(W_1\to\{x_1,x_2\})$ and $\mathcal{B}\rhd\emptyset$

When $\mathcal{A}\rhd(W_1\to\{x_1,x_2\})$ and $\mathcal{B}\rhd\emptyset$, player $\mathcal{A}$ essentially has no targeting choices and hence the following GENERALIZED-MIN evaluator applies.

▼ GENERALIZED-MIN.

$$\begin{aligned}&\tilde{v}_A(j;\mathcal{A}\rhd(W_1\to\{x_1,x_2\}),\mathcal{B}\rhd\emptyset)\\ &\quad= \min\{\tilde{v}_A(j;\mathcal{A}\oplus(W\to x_1),\mathcal{B}\oplus x_2),\tilde{v}_A(j;\mathcal{A}\oplus(W\to x_2),\mathcal{B}\oplus x_1),\\ &\qquad\qquad \min_{y\in Q_B\setminus\{x_1,x_2\}} \tilde{v}_A(j;\mathcal{A}\rhd(W\to\{x_1,x_2\}),\mathcal{B}\rhd y)\}\end{aligned} \tag{6.15}$$

6.3.3.3 Evaluation of Three Prize Subproblem

Two quantities remain to be defined: $\tilde{v}_A(j; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\})$ and $\tilde{v}_A(j; \mathcal{A} \rhd \{x_1, x_2\}, \mathcal{B} \rhd y)$.

Consider $\tilde{v}_A(j; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\})$. Note that this does not define a game position or a game tree node, but rather is the evaluation of the scenario in which player $\mathcal{A}$ intends to target $\mathcal{A} \rhd x$ and player $\mathcal{B}$ intends to target $\mathcal{B} \rhd \{y_1, y_2\}$. However $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \rhd \{y_1, y_2\})$ is the evaluation of a game position or game tree node as defined in Section 6.3.1.2. The difference is that in the former case "$\mathcal{A} \rhd x$" and in the latter case "$\mathcal{A} \oplus x$".

Section 5.6.3 presents an evaluation of a window scenario $\mathcal{B} \rhd \{y_1, y_2\}$ in which prizes y_1 and y_2 are the only two prizes remaining. We now generalize this evaluation to the case where at least these two prizes remain and player $\mathcal{A}$ may not necessarily choose to target either prize y_1 or y_2. Algorithm 6.6 defines $\tilde{v}_A(j; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\})$.

The case where $x \notin \{y_1, y_2\}$ and the window scenario W is window-feasible, requires some explanation. The essential question is: at the time t_B, does player $\mathcal{A}$ have sufficient information to be able to target either the y_1-end or the y_2-end of W? If *either* prize y_1 or prize y_2 is a lead prize for player $\mathcal{B}$ conditional on $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd W$, then we can *split* the evaluation of $\tilde{v}_A(j; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\})$ into components $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus (W \rightarrow y_1)$ and $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus (W \rightarrow y_2)$. If *neither* prize y_1 nor prize y_2 is a lead prize for player $\mathcal{B}$ conditional on $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd W$, then we cannot split the evaluation and instead must rely on the recursive evaluation of $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \rhd \{y_1, y_2\})$ to predict the best end of W to target. Hence the complexity of the definition of $\tilde{v}_A(j; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\})$ is reflected in the expansion rules of Section 6.3.2.

The definition of $\tilde{v}_A(j; \mathcal{A} \rhd \{x_1, x_2\}, \mathcal{B} \rhd y)$ is similar, with the 'max' operator replacing the 'min' operator and return zero instead of ∞.

6.3.3.4 Selecting GENERALIZED-MAXIMIN or GENERALIZED-MINIMAX

Finally, we may actually switch between employing the GENERALIZED-MAXIMIN and GENERALIZED-MINIMAX evaluators. If $t_A \ll t_B$, then most prizes will be direct leads for player $\mathcal{A}$. If $t_A \leq t_B$, then at time t_A player $\mathcal{A}$ cannot observe what player $\mathcal{B}$ is targeting at time t_B and hence cannot satisfy the *minimax assumption* (Assumption 6.3.4). If $t_A > t_B$, then at time t_A player $\mathcal{A}$ can observe what player $\mathcal{B}$ is targeting at time t_B and hence can satisfy the *minimax assumption.* If $t_A \gg t_B$, then most prizes will be direct leads for player $\mathcal{B}$. This implies that we use the GENERALIZED-MAXIMIN evaluator whenever $t_A \leq t_B$ and use the GENERALIZED-MINIMAX evaluator whenever $t_A > t_B$. This provides for the dynamic selection of an appropriate evaluator for each game tree node based upon the relative time stamps.

In general let $\kappa \in \Re$ be a parameter such that, if $t_A < t_B + \kappa$, then we employ the GENERALIZED-MAXIMIN evaluator and, if $t_A \geq t_B + \kappa$, then we employ the GENERALIZED-MINIMAX evaluator. We call this **κ-switching**. If $\kappa = \infty$ then GENERALIZED-MAXIMIN is always used; if $\kappa = -\infty$ then GENERALIZED-MINIMAX is always used.

Algorithm 6.6 definition $\tilde{v}_A(j; \mathcal{A} \triangleright x, \mathcal{B} \triangleright \{y_1, y_2\})$

```
if (x ∈ {y1, y2}) then
    if (tA > tB) then
        Let W be the early window scenario A ▷ ∅ and B ▷ {y1, y2}.
        if W is window-feasible then
            if (x = y1) then
                return(ṽA(j; A ⊕ x, B ⊕ (W→y2)))
            else
                return(ṽA(j; A ⊕ x, B ⊕ (W→y1)))
            end
        else
            // W is not window-feasible.
            return(∞)
        end
    else
        // tA ≤ tB.
        if (x = y1) then
            return(ṽA(j; A ⊕ x, B ⊕ y2))
        else
            return(ṽA(j; A ⊕ x, B ⊕ y1))
        end
    end
else
    // x ∉ {y1, y2}.
    if (tA > tB) then
        Let W be the early window scenario A ▷ ∅ and B ▷ {y1, y2}.
    else
        // tA ≤ tB.
        Let W be the window scenario A ▷ x and B ▷ {y1, y2}.
    end
    if W is window-feasible then
        if y1 and y2 are B-leads conditional on A ▷ x and B ▷ W then
            return(min{ṽA(j; A ⊕ x, B ⊕ (W→y1)), ṽA(j; A ⊕ x, B ⊕ (W→y2))})
        else if y1 is B-lead conditional on A ▷ x and B ▷ W then
            return(ṽA(j; A ⊕ x, B ⊕ (W→y1)))})
        else if y2 is B lead conditional on A ▷ x and B ▷ W then
            return(ṽA(j; A ⊕ x, B ⊕ (W→y2)))})
        else
            return(ṽA(j; A ⊕ x, B ▷ (W→{y1, y2})))
        end
    else
        // W is not window-feasible.
        return(∞)
    end
end

end
```

6.3.4 Searching the Game Tree

The **cooperative value**, $\Omega(j)$, as defined in Definition 6.2.1 and the **paranoid values**, $\Gamma_A(j)$ for player $\mathcal{A}$ and $\Gamma_B(j)$ for player $\mathcal{B}$, as defined in Definition 6.2.2 can be modified by noting only that the players may have different starting times and that the definitions apply only to a game tree node j such that $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$. Appendix 6.A.2 presents Branch and Bound algorithms for determining these values.

Algorithm 6.7 α-β-GENERALIZED-MINIMAX PRIZE-DT and Algorithm 6.8 α–β-GENERALIZED-MAXIMIN PRIZE-DT present α–β game tree search algorithms to determine $\tilde{v}_A(j; \mathcal{A} \rhd \emptyset, \mathcal{B} \rhd \emptyset; \alpha, \beta)$ by employing the GENERALIZED-MINIMAX and GENERALIZED-MAXIMIN evaluators respectively. Algorithm 6.9 α-β-GENERALIZED-MAX PRIZE-DT presents an α–β game tree search algorithm to determine $\tilde{v}_A(j; \mathcal{A} \rhd \emptyset, \mathcal{B} \rhd (W_1 \rightarrow \{y_1, y_2\}); \alpha, \beta)$ by employing the GENERALIZED-MAX evaluator. Algorithm 6.10 α-β-GENERALIZED-MIN PRIZE-DT presents an α–β game tree search algorithm to determine $\tilde{v}_A(j; \mathcal{A} \rhd (W_1 \rightarrow \{x_1, x_2\}), \mathcal{B} \rhd \emptyset; \alpha, \beta)$ by employing the GENERALIZED-MIN evaluator.

6.3.5 Example Tactical Problem Revisited

We now consider the tactics determined by PRIZE-DT for the example problem of Figure 6.1. Sections 6.1.3 and 6.2.4 discussed the tactics determined by PRIZE-GUARANTEE, PRIZE-PARANOID and ORIGINAL-DT.

6.3.5.1 Analysis of Player $\mathcal{A}$'s PRIZE-DT Tactics

Table 6.5(a) gives the PRIZE-DT root game table for player $\mathcal{A}$. An evaluation of ∞ for a "lead vs follow" scenario, or an evaluation of 0 for a "follow vs lead" scenario, implies that the lead player guarantees both prizes of the pair contingent upon the lead player's lead. A MINIMAX analysis implies that either $\mathcal{B} \oplus 2$ or $\mathcal{B} \rhd \{1, 3, 4, 5\}$ with evaluation 661. In both cases, the best response for player $\mathcal{A}$ is $\mathcal{A} \oplus 4$—in the former case as a response and in the latter case as a lead.

Consider the scenario $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 2$. Table 6.5(b) gives the corresponding PRIZE-DT game table for player $\mathcal{A}$. A MINIMAX analysis implies $\mathcal{B} \rhd \{1, 3, 6\}$ from the "MAXIMIN" section of the game table. Hence the best lead for player $\mathcal{A}$ is either $\mathcal{A} \oplus 3$ or $\mathcal{A} \oplus 6$ with evaluation 508. In both of these cases, either $\mathcal{B} \rhd \{1, 3\}$ or $\mathcal{B} \rhd \{1, 6\}$.

At the third depth of recursion, consider the scenario $\mathcal{A} \oplus 6$ and $\mathcal{B} \rhd \{1, 6\}$. Since $\mathcal{A} \oplus 6$, we assume $\mathcal{B} \oplus 1$. Table 6.5(c) gives the corresponding PRIZE-DT game table for player $\mathcal{A}$. Since $\mathcal{L}_B = \emptyset$, a MAXIMIN analysis implies player $\mathcal{A}$'s best lead is $\mathcal{A} \oplus 3$ with $\mathcal{B} \rhd \{3, 5\}$ and hence $\mathcal{B} \oplus 5$.

In summary, Table 6.5(d) illustrates a likely tactical game evolution anticipated by player $\mathcal{A}$.

6.3.5.2 Analysis of Player $\mathcal{B}$'s PRIZE-DT Tactics

Table 6.6(a) gives the PRIZE-DT root game table for player $\mathcal{B}$. A MINIMAX analysis implies that $\mathcal{A} \oplus 4$ and hence player $\mathcal{B}$'s best response is either $\mathcal{B} \rhd \{1, 3\}$ or $\mathcal{B} \rhd \{1, 6\}$. Table 6.6(d) illustrates the commitment $\mathcal{A} \oplus 4$ and $\mathcal{B} \rhd \{1, 6\}$. The solid bar represents the feasibility window through which player $\mathcal{B}$ must play in order to claim at least one prize from $\{1, 6\}$.

Algorithm 6.7 function α-β-GENERALIZED-MINIMAX PRIZE-DT

```
// Determine ṽ_A(j; A ▷ ∅, B ▷ ∅; α, β).
if j is terminal then
    return(ṽ_A(j))
end
Evaluate Γ_A(j), Γ_B(j) and Ω(j).
α' ← max{α, v_A(P_A, P_B) + Γ_A(j)}
if (α' ≥ β) then
    return(β)
end
β' ← min{β, v_A(P_A, P_B) + Ω(j) − Γ_B(j)}
if (β' ≤ α') then
    return(α')
end
finished ← false
for x ∈ D_A and not finished do
    Evaluate ṽ_A(j; A ⊕ x, B; α', β').
    α' ← max{α', ṽ_A(j; A ⊕ x, B; α', β')}
    if (α' ≥ β') then finished ← true
end
for y ∈ D_B and not finished do
    Evaluate ṽ_A(j; A, B ⊕ y; α', β').
    β' ← min{β', ṽ_A(j; A, B ⊕ y; α', β')}
    if (β' ≤ α') then finished ← true
end
if (finished) then
    return(α')
end
ub ← β'  // Upper bound on ṽ_A(j; α, β).
for y ∈ L_B and not finished do
    clb ← α'  // Lower bound on column maximum.
    finished_col ← false
    for x ∈ L_A and not finished_col do
        Evaluate ṽ_A(j; A ⊕ x, B ⊕ y; clb, ub).
        clb ← max{clb, ṽ_A(j; A ⊕ x, B ⊕ y; clb, ub)}
        if (clb ≥ ub) then finished_col ← true
    end
    for {x_1, x_2} ∈ F_A and not finished_col do
        Evaluate ṽ_A(j; A ▷ {x_1, x_2}, B ▷ y; clb, ub).
        clb ← max{clb, ṽ_A(j; A ▷ {x_1, x_2}, B ▷ y; clb, ub)}
        if (clb ≥ ub) then finished_col ← true
    end
    ub ← min{ub, clb}
    if (ub ≤ α') then finished ← true
end

(continued ...)

end
```

Algorithm 6.7 *(continued)*

```
    if (not finished and L_A ≠ ∅ and F_B ≠ ∅) then
        lb ← α'  // Lower bound on block maximum.
        for x ∈ L_A and not finished do
            rub ← ub  // Upper bound on row minimum.
            finished_row ← false
            for {y1, y2} ∈ F_B and not finished_row do
                Evaluate ṽ_A(j; A ▷ x, B ▷ {y1, y2}; lb, rub).
                rub ← min{rub, ṽ_A(j; A ▷ x, B ▷ {y1, y2}; lb, rub)}
                if (rub ≤ lb) then finished_row ← true
            end
            lb ← max{lb, rub}
            if (lb ≥ ub) then finished ← true
        end
        if (not finished and R_A ≠ ∅) then
            revub ← ub  // Upper bound on reverse block minimum.
            for y ∈ R_B and not finished do
                clb ← lb  // Lower bound on reverse column maximum.
                finished_col ← false
                for x ∈ R_A and not finished_col do
                    Evaluate ṽ_A(j; A ⊕ x, B ⊕ y; clb, revub).
                    clb ← max{clb, ṽ_A(j; A ⊕ x, B ⊕ y; clb, revub)}
                    if (clb ≥ revub) then finished_col ← true
                end
                revub ← min{revub, clb}
                if (revub ≤ lb) then finished ← true
            end
            lb ← max{lb, revub}
            if (lb ≥ ub) then finished ← true
        end
        ub ← min{ub, lb}
    end
    return(ub)

end
```

Algorithm 6.8 function α-β-GENERALIZED-MAXIMIN PRIZE-DT

```
// Determine ṽ_A(j; A ▷ ∅, B ▷ ∅; α, β).
if j is terminal then
    return(v(j))
end
Evaluate Γ_A(j), Γ_B(j) and Ω(j).
α' ← max{α, v_A(P_A, P_B) + Γ_A(j)}
if (α' ≥ β) then
    return(β)
end
β' ← min{β, v_A(P_A, P_B) + Ω(j) − Γ_B(j)}
if (β' ≤ α') then
    return(α')
end
finished ← false
for x ∈ D_A and not finished do
    Evaluate ṽ_A(j; A ⊕ x, B; α', β').
    α' ← max{α', ṽ_A(j; A ⊕ x, B; α', β')}
    if (α' ≥ β') then finished ← true
end
for y ∈ D_B and not finished do
    Evaluate ṽ_A(j; A, B ⊕ y; α', β').
    β' ← min{β', ṽ_A(j; A, B ⊕ y; α', β')}
    if (β' ≤ α') then finished ← true
end
if (finished) then
    return(α')
end
lb ← α'  // Lower bound on ṽ_A(j; α, β).
for x ∈ L_A and not finished do
    rub ← β'  // Upper bound on row minimum.
    finished_row ← false
    for y ∈ L_B and not finished_row do
        Evaluate ṽ_A(j; A ⊕ x, B ⊕ y; lb, rub).
        rub ← min{rub, ṽ_A(j; A ⊕ x, B ⊕ y; lb, rub)}
        if (rub ≤ lb) then finished_row ← true
    end
    for {y1, y2} ∈ F_B and not finished_row do
        Evaluate ṽ_A(j; A ▷ x, B ▷ {y1, y2}; lb, rub).
        rub ← min{rub, ṽ_A(j; A ▷ x, B ▷ {y1, y2}; lb, rub)}
        if (rub ≤ lb) then finished_row ← true
    end
    lb ← max{lb, rub}
    if (lb ≥ β') then finished ← true
end

(continued ...)

end
```

Algorithm 6.8 *(continued)*

if (not $finished$ and $\mathcal{L}_B \neq \emptyset$ and $\mathcal{F}_A \neq \emptyset$) **then**
  $ub \leftarrow \beta'$ *// Upper bound on block minimum.*
  for $y \in \mathcal{L}_B$ **and not** $finished$ **do**
    $clb \leftarrow lb$ *// Lower bound on column maximum.*
    $finished\_col \leftarrow$ **false**
    for $\{x_1, x_2\} \in \mathcal{F}_A$ **and not** $finished\_col$ **do**
      Evaluate $\tilde{v}_A(j; \mathcal{A} \triangleright \{x_1, x_2\}, \mathcal{B} \triangleright y; clb, ub)$.
      $rub \leftarrow \min\{rub, \tilde{v}_A(j; \mathcal{A} \triangleright \{x_1, x_2\}, \mathcal{B} \triangleright y; clb, ub)\}$
      if $(clb \geq ub)$ **then** $finished\_col \leftarrow$ **true**
    end
    $ub \leftarrow \min\{ub, clb\}$
    if $(ub \leq lb)$ **then** $finished \leftarrow$ **true**
  end
  if (not $finished$ and $\mathcal{R}_B \neq \emptyset$) **then**
    $revlb \leftarrow lb$ *// Lower bound on reverse block maximum.*
    for $x \in \mathcal{R}_A$ **and not** $finished$ **do**
      $rub \leftarrow ub$ *// Upper bound on reverse row minimum.*
      $finished\_row \leftarrow$ **false**
      for $y \in \mathcal{R}_B$ **and not** $finished\_row$ **do**
        Evaluate $\tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y; revlb, rub)$.
        $rub \leftarrow \min\{rub, \tilde{v}_A(j; \mathcal{A} \oplus x, \mathcal{B} \oplus y; revlb, rub)\}$
        if $(rub \leq revlb)$ **then** $finished\_row \leftarrow$ **true**
      end
      $revlb \leftarrow \max\{revlb, rub\}$
      if $(revlb \geq ub)$ **then** $finished \leftarrow$ **true**
    end
    $ub \leftarrow \min\{ub, revlb\}$
    if $(ub \leq lb)$ **then** $finished \leftarrow$ **true**
  end
  $lb \leftarrow \max\{lb, ub\}$
end
return(lb)

end

Algorithm 6.9 function α-β-GENERALIZED-MAX PRIZE-DT

```
// Determine ṽ_A(j; A ▷ ∅, B ▷ (W_1 → {y_1, y_2}); α, β).
finished ← false
lb ← α  // Lower bound on ṽ_A(j; α, β).
Evaluate ṽ_A(j; A ⊕ y_1, B ⊕ (W_1 → y_2); lb, β).
lb ← max{lb, ṽ_A(j; A ⊕ y_1, B ⊕ (W_1 → y_2); lb, β)}
if (lb ≥ β) then finished ← true
if (not finished) then
    Evaluate ṽ_A(j; A ⊕ y_2, B ⊕ (W_1 → y_1); lb, β).
    lb ← max{lb, ṽ_A(j; A ⊕ y_2, B ⊕ (W_1 → y_1); lb, β)}
    if (lb ≥ β) then finished ← true
end
for x ∈ Q_A \ {y_1, y_2} and not finished do
    Evaluate ṽ_A(j; A ⊕ x, B ▷ (W_1 → {y_1, y_2}); lb, β).
    lb ← max{lb, ṽ_A(j; A ⊕ x, B ▷ (W_1 → {y_1, y_2}); lb, β)}
    if (lb ≥ β) then finished ← true
end
return(lb)

end
```

Algorithm 6.10 function α-β-GENERALIZED-MIN PRIZE-DT

```
// Determine ṽ_A(j; A ▷ (W_1 → {x_1, x_2}), B ▷ ∅; α, β).
finished ← false
ub ← β  // Upper bound on ṽ_A(j; α, β).
Evaluate ṽ_A(j; A ⊕ (W_1 → x_1), B ⊕ x_2; α, ub).
ub ← min{ub, ṽ_A(j; A ⊕ (W_1 → x_1), B ⊕ x_2; α, ub)}
if (ub ≤ α) then finished ← true
if (not finished) then
    Evaluate ṽ_A(j; A ⊕ (W_1 → x_2), B ⊕ x_1; α, ub).
    ub ← min{ub, ṽ_A(j; A ⊕ (W_1 → x_2), B ⊕ x_1; α, ub)}
    if (ub ≤ α) then finished ← true
end
for y ∈ Q_B \ {x_1, x_2} and not finished do
    Evaluate ṽ_A(j; A ▷ (W_1 → {x_1, x_2}), B ⊕ y; α, ub).
    ub ← min{ub, ṽ_A(j; A ▷ (W_1 → {x_1, x_2}), B ⊕ y; α, ub)}
    if (ub ≤ α) then finished ← true
end
return(ub)

end
```

Table 6.5: Tactical Example of Player $\mathcal{A}$ PRIZE-DT Game Tables

	2	5	{4,6}	{4,3}	{4,1}	{6,3}	{6,1}	{3,1}
4	661	608	∞	∞	∞	∞	661	661
6	508	471	508	508	508	∞	571	508
1	545	673	429	520	520	429	682	∞
3	417	429	417	429	417	429	429	508
{2,5}	0	339						

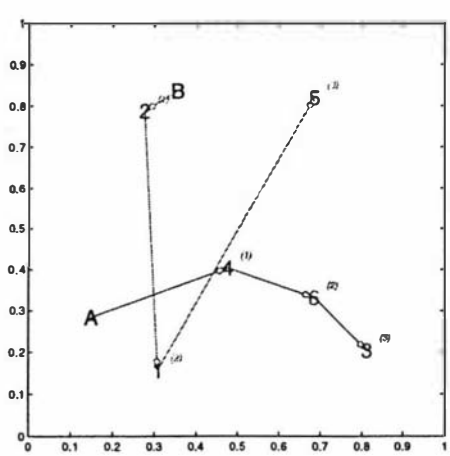
(a) Player $\mathcal{A}$ Root PRIZE-DT Game Table

	5	{1,6}	{1,3}	{6,3}
6	520	508	508	∞
1	520	392	483	429
3	520	508	508	∞

(b) $\mathcal{A} \oplus 4$ ($t_A = 0.35$) and $\mathcal{B} \oplus 2$ ($t_B = 0.09$)

	{3,5}
3	227
5	190

(c) $\mathcal{A} \oplus 4 \oplus 6$ ($t_A = 0.57$) and $\mathcal{B} \oplus 2 \oplus 1$ ($t_B = 0.73$)



(d) Player $\mathcal{A}$'s Anticipated Tactical Game Evolution

Consider further the scenario $\mathcal{A} \oplus 4$ and $\mathcal{B} \rhd \{1,6\}$. Table 6.6(b) gives the PRIZE-DT game table for player $\mathcal{B}$. Analysis implies that $\mathcal{A} \oplus 1$. Since already $\mathcal{B} \rhd \{1,6\}$, player $\mathcal{B}$ can then respond with $\mathcal{B} \oplus 6$.

At the third depth of recursion, consider the scenario $\mathcal{A} \rhd W \oplus 6$ and $\mathcal{B} \oplus 4 \oplus 1$ where W is the feasibility window associated with $\mathcal{A} \oplus 4$ and $\mathcal{B} \rhd \{1,6\}$. Table 6.6(c) gives the PRIZE-DT game table for player $\mathcal{B}$. Since $\mathcal{L}_A = \emptyset$, a MAXIMIN analysis implies player $\mathcal{B}$'s best lead is $\mathcal{B} \oplus 3$ with player $\mathcal{A}$ following with $\mathcal{A} \rhd \{2,5\}$ or $\mathcal{A} \rhd \{3,5\}$. In either case, player $\mathcal{A}$ eventually claims prize 5 and player $\mathcal{B}$ eventually claims prize 2.

In summary, Table 6.6(e) illustrates a likely tactical game evolution anticipated by player $\mathcal{B}$.

6.3.5.3 Discussion

In PRIZE-PARANOID both players basically committed to their PRIZE-GUARANTEE paths. In ORIGINAL-DT, both players were hesitant in the initial stages and the tactics determined by each player were sufficiently different that player $\mathcal{A}$ could not anticipate a player $\mathcal{B}$ threat on prize 4 and player $\mathcal{B}$ could not figure out an end-game on prizes $\{2,4,5,6\}$.

By comparison, PRIZE-DT appears to be more tactically robust in that both players can evaluate the possible end-games and the resulting initial tactics are very similar and hence realistic. Significantly, player $\mathcal{B}$ is able to effectively plan in situations involving targeting prizes with are not leads.

Note also that, by design, the depth of recursion for PRIZE-DT is often much less than for ORIGINAL-DT since the former often eliminates two prizes at each depth. This partially accounts for the hesitant nature of the probes in ORIGINAL-DT in that a player often does not continue through to the targeted prize.

In conclusion, it appears from this tactical example that PRIZE-DT provides a superior tactical analysis for small problems than can be determined by the previously considered tactical engines.

6.3.6 Summary

We have designed the PRIZE-DT tactical engine, prosed MINIMAX, MAXIMIN and evaluators for the corresponding game tables. We have also compared PRIZE-DT favourably against the tactics determined by PRIZE-GUARANTEE, PRIZE-PARANOID and ORIGINAL-DT.

6.4 Dynamic Monitoring: Small-DMS

According to Pearl [174], a game tree is "solved" when the root node is labelled with a value and "an associated optimal playing strategy which prescribes how that label can be guaranteed regardless of how [the opponent] plays." We have defined several evaluators and search algorithms which determine a value for the root node game position and imply "optimal" tactical decisions which guarantee the root value *if* the assumptions of that evaluator are realised *and if* a game position at least as good as the anticipated game position representative of the tactical scenario *is actually realised.* In short, the tactical engines ORIGINAL-DT and PRIZE-DT provide only an *estimate* of the value of game position to player $\mathcal{A}$ under a number of *idealised assumptions.*

Table 6.6: Tactical Example of Player $\mathcal{B}$ PRIZE-DT Game Tables

	4	6	1	3	{2,5}
2	327	492	327	583	∞
5	327	480	327	571	661
{4,6}	0	492	480	571	
{4,3}	0	492	480	571	
{4,1}	0	492	480	571	
{6,3}	0	0	480	571	
{6,1}	**339**	429	318	571	
{3,1}	**339**	492	0	492	

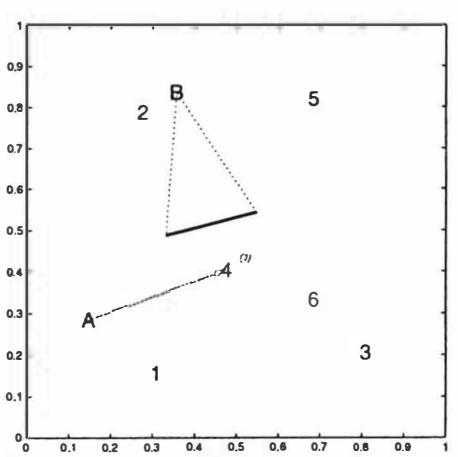
(a) Player $\mathcal{B}$ Root PRIZE-DT Game Table

	6	1	3	5	2
{6,1}	339	**318**	418	520	520

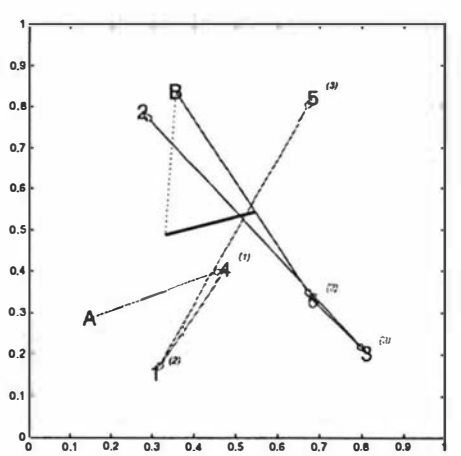
(b) $\mathcal{A} \triangleright \{6,1\}$ ($t_A = 0$) and $\mathcal{B} \oplus 4$ ($t_B = 0.35$)

	{2,3}	{2,5}	{3,5}
3	417	**364**	**364**
2	327	137	327
5	327	417	327

(c) $\mathcal{A} \triangleright W \oplus 6$ ($t_A = 0.60$) and $\mathcal{B} \oplus 4 \oplus 1$ ($t_B = 0.65$)



(d) Initial Commitment $\mathcal{A} \oplus 4$ and $\mathcal{B} \triangleright \{1,6\}$



(e) Player $\mathcal{B}$'s Anticipated Tactical Game Evolution

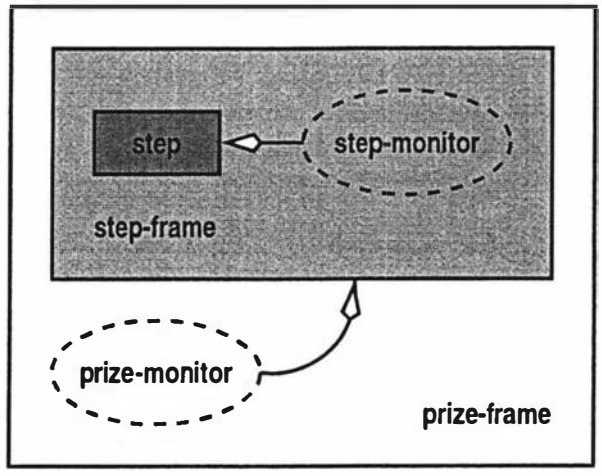


Figure 6.5: Monitors and Frames for Small DMS

Accuracy of evaluators is investigated in Section 9.2. Monitoring is the process which observes the incremental changes in the game position, refines estimates of the opponent's intended prize target, requests new tactical information if necessary and improves the selection of a target prize. Hence a tactically projected game position may not be realised but rather is viewed as an objective which itself may incrementally change.

Chapter 4 introduced the concept of a Dynamic Monitoring System (DMS). In this section we design a Small-DMS for small problems, incorporating the tactical engines, ORIGINAL-DT, PRIZE-DT and PRIZE-PARANOID, designed in this chapter, and the two prize strategies, designed in Chapter 5. Firstly, we define the frames and monitors, and look at methods for checking and refining a *prize-ts*. Secondly, we design a generic PRIZE-MONITOR and propose specific modules incorporating the tactical engines ORIGINAL-DT and PRIZE-DT. Thirdly, we propose a STEP-MONITOR based on Sections 5.6.1 and 5.6.2. Finally, we discuss how the tactical engines could incorporate *prize-ts* concepts in their evaluations.

6.4.1 System of Frames and Monitors

A tactical engine provides tactical information about matching prize targeting scenarios of the players. The two prize strategies of Chapter 5 provide one-step move for targeting a pair of prizes, neither of which is guaranteed. The Small DMS is designed to couple together the sequential decisions in terms of targeting prizes (the **prize-frame**) and in terms of individual steps (the **step-frame**).

One *dynamic monitor* operates at each of the two frames. Figure 6.5 illustrates the relationship between the monitors and frames in the Small DMS.

- At the prize-frame level, a PRIZE-MONITOR estimates which prize, or set of prizes, the opponent *is* targeting or *is likely* to target, i.e., a *prize-ts* as in Section 3.2.3.2. A tactical engine determines a *tactical response* encapsulated in a step-frame, which consists of a

prize-ts and a target-prize (or a target-pair).

- At the step-frame level, a STEP-MONITOR determines an initial one-step move implementing the target-prize or target-pair.

6.4.2 Target Sets

Consider the ORIGINAL-DT and PRIZE-DT tactical engines. The MINIMAX evaluators assumed that player $\mathcal{A}$ could observe player $\mathcal{B}$'s movements and respond contingent upon which prize we estimate that player $\mathcal{B}$ is targeting. They also assumed that player $\mathcal{B}$ commits to a target prize and that player $\mathcal{A}$ could subsequently select a response without fear of player $\mathcal{B}$ "changing its mind". Hence there remains considerable difficulty in translating the observation of player $\mathcal{B}$'s movement into defining a step-frame.

The approach we take is based on the idea of a set of possible opponent target prizes called a **prize-target-set** (or **prize-ts**). The idea is that we hypothesise an initial set of prizes which we use to make an initial response and as play develops over the next few steps we incrementally test and reject or refine the target set. As a *prize-ts* is refined we place more confidence in our estimate and hence make closer matched responses. A *prize-ts* is rejected if we have little confidence in our estimate.

Section 3.2.3.2 presents methods for building an initial *prize-ts* by observation of the opponent. We now give methods for refining (contracting) a *prize-ts* (by only discarding prizes and not adding any further prizes) and for checking a *prize-ts* to determine if new prizes should be added or the entire *prize-ts* discarded and a new *prize-ts* built. This makes for a simple system for dynamically maintaining a *prize-ts*. Ideally such a system should refine and augment the *prize-ts* with some associated measure of confidence in the current *prize-ts* which changes with the observed consistency (or lack of consistency) of the opponent relative to our predictions.

6.4.2.1 Refine prize-ts

Refinement of a given *prize-ts* may only discard prizes, not add further prizes. The idea is that we attempt to narrow down the *prize-ts* until we are confident that the opponent is targeting a specific prize. Suppose we have estimated the bearing, ψ_B, of player $\mathcal{B}$. Then any of the following three methods are applicable:

REFINE-PRIZETS (BEARING): Discard those prizes from the current *prize-ts* which deviate in bearing from ψ_B by more than θ_{thresh}, for some parameter $\theta_{thresh} > 0$.

REFINE-PRIZETS (INSERTION): Let $s \in T$ (a seed prize) be the prize closest in bearing to ψ_B. Discard those prizes from the current *prize-ts* such that $\min\{d_{Bj} + d_{js} - d_{Bs}, d_{Bs} + d_{sj} - d_{Bj}\} > d_{thresh}$, for some parameter d_{thresh}.

REFINE-PRIZETS (CONFIDENCE): Refine the *prize-ts* according to the consistency of the opponent by setting the threshold parameter as a decreasing function of the time, t_{conf}, since the *prize-ts* was rebuilt. As long as CHECK-PRIZETS (BEARING) does not discard the current *prize-ts*, the confidence in that *prize-ts* increases. Set the threshold θ_{thresh} according to

$\theta_{thresh} = \theta_0 e^{-kt_{conf}}$, or similarly for d_{thresh}. When the current *prize-ts* is discarded we have less confidence in the the new *prize-ts*; however, if the rate of decrease in the threshold value is too great, we will discard the *prize-ts* and rebuild more often. Hence, there is a balance to be found between confidence and variability.

6.4.2.2 Check prize-ts

Finally we need to determine if player $\mathcal{B}$'s observed movements are still consistent with the current *prize-ts*. Suppose we have estimated the bearing, ψ_B, of player $\mathcal{B}$.

CHECK-PRIZETS (BEARING): Check to see if ψ_B is pointing towards the current *prize-ts*, i.e., if $\exists$ two prizes which have angular gap less than 180° through which ψ_B passes. Also, if player $\mathcal{B}$ is targeting a prize not in the current *prize-ts* to within θ_{thresh} of its current bearing, then *prize-ts* is invalid.

6.4.2.3 Multiple Target Sets

The methods described thus far for building and refining a *prize-ts* are based on *observation* of the opponent. Another approach is to build several candidate *prize-ts* based on *prediction* of the opponent's likely targets by employing a tactical engine. A **prize-mts** (or **prize multiple target set**) is a family of target sets. Note that a *prize-mts* is not necessarily a partition of the prizes. We present a possible method for building a *prize-mts* without reference to a tactical engine.

BUILD-PRIZEMTS-SWEEP: Sort the prizes radially, centred on the current location of player $\mathcal{B}$. Choose the m largest angular gaps between the prizes; these partition the set of prizes into m distinct sets.

Further methods, specialised to a specific tactical engine, are presented in the next section. Methods for constructing a *prize-ts* from a *prize-mts* include:

MERGE. The *prize-ts* is the union of the component sets of the *prize-mts*.

SELECT. For each component set of the *prize-mts*, use a tactical engine to evaluate the *likelihood* that the opponent will target that set of prizes. Select that component set with the greatest likelihood as the *prize-ts*.

MERGE AND SELECTION. Use a tactical engine to evaluate each component set from the *prize-mts* and let the *prize-ts* be the union of those component sets which evaluate up to some threshold above the minimum evaluation.

6.4.3 Small DMS

Algorithms 6.11–6.12 together constitute a generic PRIZE-MONITOR for the Small DMS.

A practical consideration is that we do not wish to employ the tactical engine unnecessarily, since we assume that the tactical engine is the most computationally expensive procedure of those in the Small DMS. After each step, if the existing *prize-ts* is still valid then it is refined, but if

Algorithm 6.11 monitor SMALLDMS-PRIZE-MONITOR

```
// Build new prize-ts.
if ∄ step-frame or prize just claimed then
    module: Build prize-mts.
    module: Select or construct prize-ts from prize-mts.
else
    BUILD-PRIZE-TS
end

// Determine tactical response.
module: Determine target-prize or target-pair.

end
```

Algorithm 6.12 monitor SMALLDMS-PRIZE-REFINE

```
// Refine current prize-ts.
REFINE-PRIZETS
if prize-ts = ∅ then
    // Build new step-frame.
    SMALLDMS-PRIZE-MONITOR
else if change made to prize-ts then
    // Update tactical response.
    module: Determine target-prize or target-pair.
else
    // No changes to step-frame necessary.
end

end
```

the *prize-ts* is not valid (the step-frame is out of scope), or is refined to empty, then it is rebuilt from scratch. The tactical engine is only engaged to redetermine the target-prize or target-pair structure if the *prize-ts* is altered from the previous step. We employ a tactical engine to predict a *prize-ts* until the *prize-ts* is refined to null, then use the generic BUILD-PRIZE-TS based solely on the observation.

There are three specialised *modules* that a specific PRIZE-MONITOR can provide:

module: Build a *prize-mts* consisting of at least one component-set, e.g., use BUILD-PRIZEMTS-SWEEP. The *default* is to construct a *prize-mts* consisting of a single component-set which contains every remaining prize.

module: Select or construct a nonempty *prize-ts* from a *prize-mts*, e.g., use MERGE, SELECT or MERGE-AND-SELECTION. The *default* is to use MERGE, which requires no evaluations.

module: Determine target-prize or target-pair (as appropriate). There is no *default*.

We now design modules for the PRIZE-MONITOR using the tactical engines ORIGINAL-DT, PRIZE-DT and PRIZE-PARANOID. The three monitors are based upon two steps:

(i). Compute the appropriate root game table using a tactical engine.

(ii). Using the table or vector information to build a *prize-mts*, evaluate a candidate *prize-ts* and determine a target-prize or target-pair.

6.4.3.1 Monitor: ORIGINAL-DT PRIZE-MONITOR

The ORIGINAL-DT tactical engine can be used as the basis for all three required modules. Let r be the root node of the game tree ORIGINAL-DT and let $\tilde{v}_A$ be any one of the evaluators of Section 6.2.2.

▼ module: Build prize-mts

The *minimaximin assumption* (Assumption 5.6.3) implies that if player $\mathcal{B}$ targets a prize $y \in \mathcal{L}_B$ then player $\mathcal{A}$ can select the best possible response. Hence, for each $y \in \mathcal{L}_B$,

$$\tilde{v}_A(i; \mathcal{B} \rhd y) = \max_{x \in Q_A} \tilde{v}_A(i; \mathcal{A} \odot x, \mathcal{B} \odot y)$$

provides a measure of the *likelihood* that player $\mathcal{B}$ will target prize y; the lower the evaluation the more likely it is that player $\mathcal{B}$ will target that prize. The idea is that we can use each of these likely target prizes as a seed for a candidate *prize-ts* consisting of all prizes "close" in bearing from the opponent.

Evaluate $\tilde{v}_A(i)$ employing the MINIMAXIMIN evaluator with $Q_L = \mathcal{L}_B$. For some parameter $\gamma > 1$, let

$$S = \{y \in \mathcal{L}_B : \tilde{v}_A(i; \mathcal{B} \rhd y) \leq \gamma \tilde{v}_A(i)\}.$$

Expand each seed prize $s \in S$ to a component-set of the *prize-mts* by adding each prize which deviates in bearing from player $\mathcal{B}$'s current location by at most θ_{thresh} from the bearing to prize s. Finally, if $\max_{x \in Q_A} \left\{\min_{y \in Q_B \setminus \mathcal{L}_B} \tilde{v}_A(i; \mathcal{A} \odot x, \mathcal{B} \odot y)\right\} \leq \gamma \tilde{v}_A(i)$ then also add the component-set $Q_B \setminus \mathcal{L}_B$ to the *prize-mts*.

▼ module: Select prize-ts

The *minimax assumption* assumes that we can distinguish which prize the opponent may be targeting. By definition, the prizes in a *prize-ts* are indistinguishable as target prizes from current observations. Hence, we select a *prize-ts* from the component-sets of the *prize-mts* by choosing the component-set with the least evaluation applying the *maximin assumption* over the prizes in the component-set.

For a set $T \subseteq Q_B$, let

$$\tilde{v}_A(i; \mathcal{B} \rhd T) = \max_{x \in Q_A} \left\{\min_{y \in T} \tilde{v}_A(i; \mathcal{A} \odot x, \mathcal{B} \odot y)\right\}.$$

Select the component-set T from the *prize-mts* which minimizes $\tilde{v}_A(i; \mathcal{B} \rhd T)$.

▼ module: **Determine target-prize**

Let T be a *prize-ts*. Target the prize x^* by applying the *maximin assumption* over the *prize-ts*.

$$x^* = \text{argmax}_{x \in Q_A} \left\{ \min_{y \in T} \tilde{v}_A(i; \mathcal{A} \odot x, \mathcal{B} \odot y) \right\}.$$

6.4.3.2 Monitor: PRIZE-DT PRIZE-MONITOR

The PRIZE-DT tactical engine can be used as the basis for all three required modules. Let r be the root node of the game tree PRIZE-DT and let $\tilde{v}_A$ be any one of the evaluators of Section 6.3.3.

▼ module: **Select prize-ts**

Suppose $T \subseteq \mathcal{L}_B$ is a component of the *prize-mts* and suppose that player $\mathcal{A}$ is considering targeting a lead prize. Since we cannot distinguish which prize in T player $\mathcal{B}$ is targeting, we evaluate this scenario applying the *maximin assumption*. Let

$$\begin{aligned} &\tilde{v}_A(i; \mathcal{A} \rhd \mathcal{L}_A, \mathcal{B} \rhd T \subseteq \mathcal{L}_B) \\ &\quad = \max_{x \in \mathcal{L}_A} \left\{ \min_{y \in T} \tilde{v}_A(i; \mathcal{A} \oplus x, \mathcal{B} \oplus y) \right\} \end{aligned} \tag{6.16}$$

Suppose $T \subseteq Q_B$ is a component of the *prize-mts* and suppose that player $\mathcal{A}$ is considering targeting a lead prize $x \in \mathcal{L}_A$. Which follow-pairs from $\mathcal{F}_B$ should be considered in the evaluation of this scenario? A follow-pair $\{y_1, y_2\} \in \mathcal{F}_B$ for which $y_1 \notin T$ and $y_2 \notin T$ certainly must not be considered. Also a follow-pair $\{y_1, y_2\} \in \mathcal{F}_B$ for which $y_1 \in T$ and $y_2 \in T$ certainly must be considered. The observation of player $\mathcal{B}$ targeting prize $y_1 \in T \cap \mathcal{R}_B$ is consistent with player $\mathcal{B}$ targeting $\mathcal{B} \rhd \{y_1, y_2\}$ for any prize $y_2 \in \mathcal{R}_B \setminus \{y_1\}$. Hence we also consider any follow-pair $\{y_1, y_2\} \in \mathcal{F}_B$ for which $y_1 \in T$ *or* $y_2 \in T$. Applying the *maximin assumption*, let

$$\begin{aligned} &\tilde{v}_A(i; \mathcal{A} \rhd \mathcal{R}_A, \mathcal{B} \rhd T \subseteq Q_B) \\ &\quad = \max_{x \in \mathcal{L}_A} \left\{ \min \left\{ \min_{y \in \mathcal{L}_B \cap T} \tilde{v}_A(i; \mathcal{A} \oplus x, \mathcal{B} \oplus y), \right. \right. \\ &\qquad \left. \left. \min_{\{y_1, y_2\} \in \mathcal{F}_B : y_1 \in T \text{ or } y_2 \in T} \tilde{v}_A(i; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\}) \right\} \right\} \end{aligned} \tag{6.17}$$

be the evaluation of this scenario.

Suppose $T \subseteq \mathcal{L}_B$ is a component of the *prize-mts* and suppose that player $\mathcal{A}$ is considering targeting a follow-pair. Again, we cannot distinguish which prize in T player $\mathcal{B}$ is targeting and we can therefore only evaluate this scenario by assuming that player $\mathcal{B}$ knows which follow-pair player $\mathcal{A}$ will select. Note that this is an even more conservative assumption than the *maximin assumption*. Let

$$\begin{aligned} &\tilde{v}_A(i; \mathcal{A} \rhd \mathcal{F}_A, \mathcal{B} \rhd T \subseteq \mathcal{L}_B) \\ &\quad = \max_{\{x_1, x_2\} \in \mathcal{F}_A} \left\{ \min_{y \in T} \tilde{v}_A(i; \mathcal{A} \rhd \{x_1, x_2\}, \mathcal{B} \rhd y) \right\} \end{aligned} \tag{6.18}$$

be the evaluation of this scenario.

Suppose $T \subseteq Q_B$ is a component of the *prize-mts* and suppose that player $\mathcal{A}$ is considering targeting a follow-pair. Again we assume that player $\mathcal{B}$ knows which follow-pair $\{x_1, x_2\} \in \mathcal{F}_A$

that player $\mathcal{A}$ will select. The observation of player $\mathcal{B}$ targeting a prize $y \in T \cap \mathcal{R}_B$ can be matched against both reverse-leads $\mathcal{A} \oplus x_1$ and $\mathcal{A} \oplus x_2$. However, since player $\mathcal{A}$ cannot distinguish which prize in T player $\mathcal{B}$ is targeting, player $\mathcal{A}$ cannot choose between $\mathcal{A} \oplus x_1$ and $\mathcal{A} \oplus x_2$ when matching against each individual $\mathcal{B} \oplus y \in T \cap \mathcal{R}_B$. Hence, let

$$\begin{aligned}
&\tilde{v}_A(i; \mathcal{A} \rhd \mathcal{F}_A, \mathcal{B} \rhd T \subseteq Q_B) \\
&\quad = \max_{\{x_1, x_2\} \in \mathcal{F}_A} \left\{ \min \left\{ \min_{y \in \mathcal{L}_B \cap T} \tilde{v}_A(i; \mathcal{A} \rhd \{x_1, x_2\}, \mathcal{B} \rhd y), \right. \right. \\
&\qquad \left. \left. \min_{y \in \mathcal{R}_B \cap T} \{\min \{\tilde{v}_A(i; \mathcal{A} \oplus x_1, \mathcal{B} \oplus y), \tilde{v}_A(i; \mathcal{A} \oplus x_2, \mathcal{B} \oplus y)\}\} \right\} \right\} \qquad (6.19)
\end{aligned}$$

be the evaluation of this scenario.

Finally, let

$$\begin{aligned}
&\tilde{v}_A(i; \mathcal{B} \rhd T \subseteq Q_B) \\
&\quad = \tilde{v}_A(i; \mathcal{A} \rhd \mathcal{R}_A, \mathcal{B} \rhd T \subseteq Q_B) + \tilde{v}_A(i; \mathcal{A} \rhd \mathcal{F}_A, \mathcal{B} \rhd T \subseteq Q_B) \qquad (6.20)
\end{aligned}$$

Select the component-set T from the *prize-mts* which maximizes (6.20).

▼ module: Build prize-mts

Similar to the ORIGINAL-DT PRIZE-MONITOR, for each $y \in \mathcal{L}_B$,

$$\begin{aligned}
&\tilde{v}_A(i; \mathcal{B} \rhd y) \\
&\quad = \max \left\{ \max_{x \in \mathcal{L}_A} \tilde{v}_A(i; \mathcal{A} \oplus x, \mathcal{B} \oplus y), \right. \\
&\qquad \left. \max_{\{x_1, x_2\} \in \mathcal{F}_A} \tilde{v}_A(i; \mathcal{A} \rhd \{x_1, x_2\}, \mathcal{B} \rhd y) \right\} \qquad (6.21)
\end{aligned}$$

provides a measure of the *likelihood* that player $\mathcal{B}$ will target prize y; the lower the evaluation the more likely it is that player $\mathcal{B}$ will target that prize. Then evaluate $\tilde{v}_A(i)$ employing the GENERALIZED-MINIMAX evaluator. For some parameter $\gamma > 1$, let

$$S = \{y \in \mathcal{L}_B : \tilde{v}_A(i; \mathcal{B} \rhd y) \leq \gamma \tilde{v}_A(i)\}.$$

Expand each seed prize $s \in S$ to a component-set of the *prize-mts* by adding each prize which deviates, in bearing from player $\mathcal{B}$'s current location, by at most θ_{thresh} from the bearing to prize s. Let

$$\begin{aligned}
&\tilde{v}_A(i; \mathcal{B} \rhd \mathcal{R}_B) \\
&\quad = \max \left\{ \max_{x \in \mathcal{L}_A} \left\{ \min_{\{y_1, y_2\} \in \mathcal{F}_B} \tilde{v}_A(i; \mathcal{A} \rhd x, \mathcal{B} \rhd \{y_1, y_2\}) \right\}, \right. \\
&\qquad \left. \min_{y \in \mathcal{R}_B} \left\{ \max_{x \in \mathcal{R}_A} \tilde{v}_A(i; \mathcal{A} \oplus x, \mathcal{B} \oplus y), \right\} \right\}. \qquad (6.22)
\end{aligned}$$

If $\tilde{v}_A(i; \mathcal{B} \rhd \mathcal{R}_B) \leq \gamma \tilde{v}_A(i)$, then also add the component-set $\mathcal{R}_B$ to the *prize-mts*.

▼ module: Determine target-prize or target-pair

Let T be a *prize-ts*. If $\tilde{v}_A(i; \mathcal{A} \rhd \mathcal{R}_A, \mathcal{B} \rhd T \subseteq Q_B) \geq \tilde{v}_A(i; \mathcal{A} \rhd \mathcal{F}_A, \mathcal{B} \rhd T \subseteq Q_B)$ then target the prize x^* argument which maximizes (6.17); otherwise target the follow-pair $\{x_1^*, x_2^*\}$ argument which maximizes (6.19).

6.4.3.3 Monitor: PRIZE-PARANOID PRIZE-MONITOR

The PRIZE-PARANOID method can also be used as the basis for all three required modules.

Let $\Gamma_A(\mathcal{B} \rhd y)$ be the paranoid value of player $\mathcal{A}$ conditional upon $\mathcal{B} \rhd y$. Similarly, let $\Gamma_A(\mathcal{A} \rhd x, \mathcal{B} \rhd y)$ be the paranoid value of player $\mathcal{A}$ conditional upon $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd y$.

▼ module: Build prize-mts

For each $y \in Q_B$, $\Gamma_A(\mathcal{B} \rhd y)$ provides a measure of the *likelihood* that player $\mathcal{B}$ will target prize y; the lower the evaluation the more likely it is that player $\mathcal{B}$ will target that prize. For some parameter $\gamma > 1$, let

$$S = \{y \in Q_B : \Gamma_A(\mathcal{B} \rhd y) \leq \gamma \Gamma_A\}.$$

Expand each seed prize $s \in S$ to a component-set of the *prize-mts*, as for the ORIGINAL-DT PRIZE-MONITOR.

▼ module: Select prize-ts

Select the component-set $T \subseteq Q_B$ from the *prize-mts* which minimizes

$$\max_{x \in Q_A} \left\{ \min_{y \in T} \Gamma_A(\mathcal{A} \rhd x, \mathcal{B} \rhd y) \right\}.$$

Note that, in general, $\Gamma_A(\mathcal{A} \rhd x, \mathcal{B} \rhd T) \neq \min_{y \in Q_B} \Gamma_A(\mathcal{A} \rhd x\mathcal{B} \rhd y)$. On the one hand, $\Gamma_A(\mathcal{B} \rhd T)$ measures the best guarantee given that player $\mathcal{A}$ will never be able to discern which prize $y \in T$ player $\mathcal{B}$ first visits. However, $\min_{y \in Q_B} \Gamma_A(\mathcal{B} \rhd y)$ reflects that player $\mathcal{B}$ selects a prize $y \in Q_B$ to target given that player $\mathcal{A}$ actually knows which prize this is.

▼ module: Determine target-prize

Let T be a *prize-ts*. Target the prize x^* by applying a maximin assumption.

$$x^* = \text{argmax}_{x \in Q_A} \left\{ \min_{y \in T} \Gamma_A(\mathcal{A} \rhd x, \mathcal{B} \rhd y) \right\}.$$

6.4.4 Step Monitor

The prize-monitors of Section 6.4.3 each build a *step-frame* consisting of a *prize-ts* and a *target-prize* or a *target-pair* structure. Algorithm 6.13 presents a simple SMALLDMS-STEP-MONITOR. The STEP-MONITOR cannot change the step-frame but must trigger the PRIZE-MONITOR to do so. This guards against engaging the tactical engine at every step.

The component *tiny problem* strategies are:

TWO-PRIZE-BIAS: Determine a two-prize target-window as in Section 5.3.3 and always play to the opposite side of the window from the *prize-ts* predicted target. Note that this is not equivalent to TIT FOR TAT since we go with our prediction rather than our observation.

THREE-PRIZE-WINDOW: Determine a feasible-window as in Section 5.6.2.

Algorithm 6.13 monitor SMALLDMS-STEP-MONITOR

```
// Check if step-frame exists and is still valid.
if ∄ step-frame or prize just claimed then
    SMALLDMS-PRIZE-MONITOR // Build new step-frame

else
    CHECK-PRIZETS
    if prize-ts invalid then
        SMALLDMS-PRIZE-MONITOR // Build new step-frame

    else
        SMALLDMS-PRIZE-REFINE // Refine current step-frame

    end
end

// Determine step.
if ∃ a target-prize x then
    target ← x
else
    // Target target-prize-pair {x1, x2}.
    if (|T| = 1) then
        if (T ⊂ {x1, x2}) then
            TWO-PRIZE-BIAS
        else
            THREE-PRIZE-WINDOW
        end
    else
        COMPROMISE-WINDOW.
    end
end

end
```

COMPROMISE-WINDOW: Sort the prizes in $T \setminus \{x_1, x_2\}$ in order of distance from player $\mathcal{B}$. Starting with the closest window, determine the intersection of feasible-windows, discarding any subsequent window which make the intersection empty (at worst we are left with the closest feasible window). The compromise is that we wish to satisfy the earlier feasible-windows; the later ones may still be feasible if we can quickly narrow the *prize-ts*. Determine a compromise window and play TIT FOR TAT (BEARING) through it.

6.4.5 Discussion

We have employed a tactical engine as a "black box" distinct from any knowledge of the *prize-ts* under investigation. To evaluate a *prize-ts* or determine a target-prize or target-pair we have applied the MAXIMIN principle only on the table or vector entries without consideration for the structure of the options considered. Two examples should suffice:

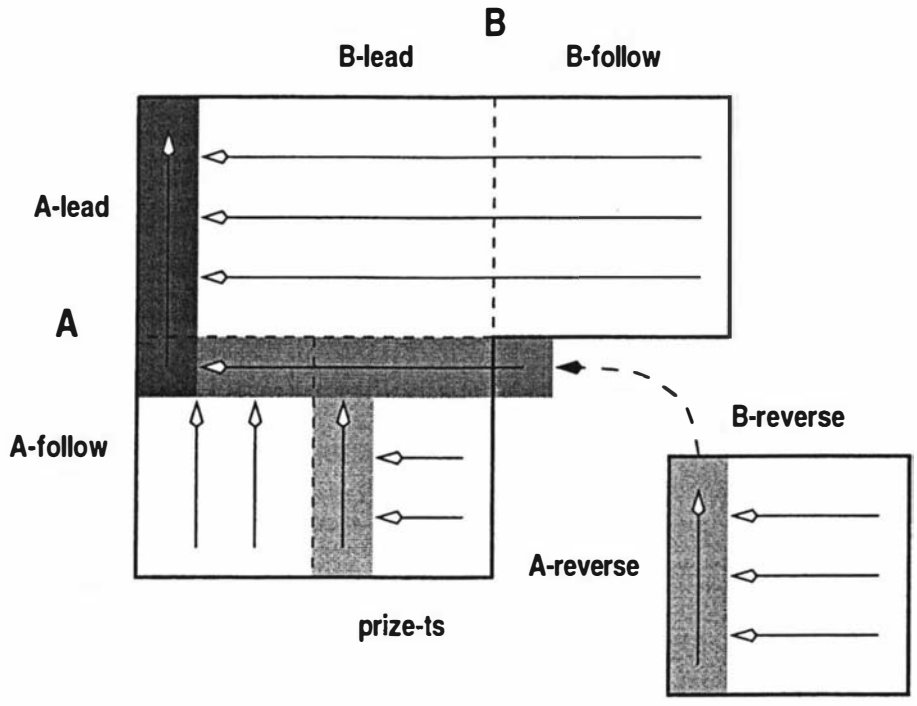
(a). Suppose we have a *prize-ts* and apply PRIZE-PARANOID-MONITOR. We select the target-prize i according to $\text{argmax}_{i \in Q} \min_{j \in T} \omega_i(j)$, i.e., what prize i should player $\mathcal{A}$ target such that player $\mathcal{B}$ chooses the correspondingly most restrictive choice of prize j? Let $\omega_i(T)$ be the $\mathcal{A}$-paranoid-value given that player $\mathcal{B}$ is committed to targeting a prize from T, as in Algorithm 3.19 TARGET SET PARANOID. Clearly $\omega_i(T) \leq \min_{j \in T} \omega_i(j)$ $\forall i \in Q$.

(b). Suppose we have a *prize-ts* and apply PRIZE-DT-MONITOR. Choose $\{x_1, x_2\} \in \mathcal{F}_A$. We have used

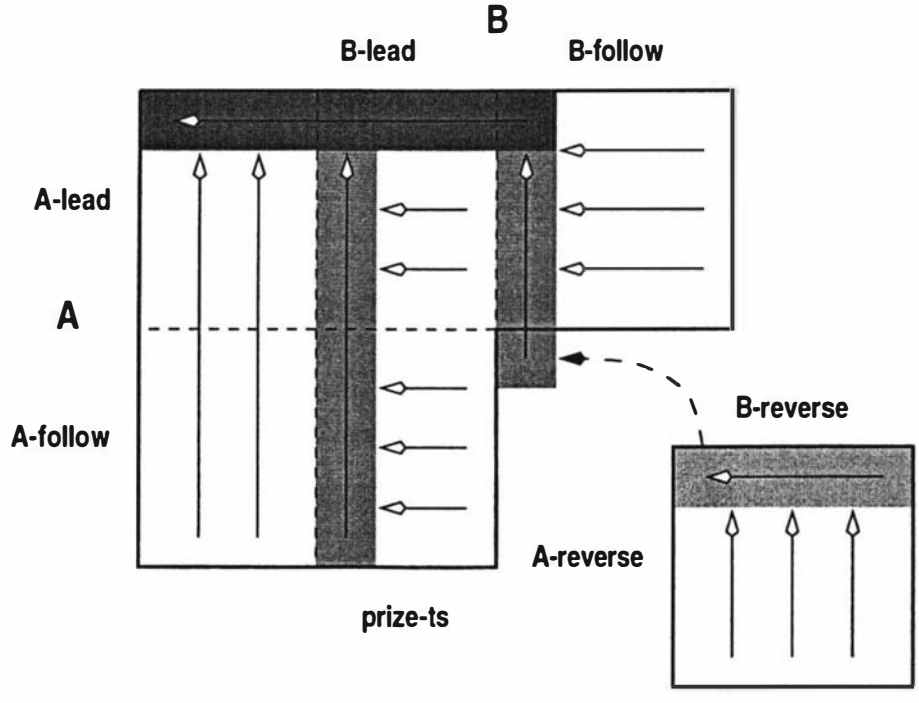
$$\omega_{\{x_1,x_2\}}(T) = \min \left\{ \min_{y \in \mathcal{L}_B \cap T} v(i, \{x_1, x_2\}, y), \min_{y \in \mathcal{R}_B \cap T} \left\{ \min \{v(i, x_1, y), v(i, x_2, y)\} \right\} \right\}.$$

This assumes that, if we select the target-pair $\{x_1, x_2\}$, we can determine which single prize the opponent is targeting so as to determine the appropriate feasible-window. In Section 6.4.4 we have already seen that we can determine a *compromise window*. Conservatively we would first determine if the scenario $\mathcal{A} \triangleright \{x_1, x_2\}$ and $\mathcal{B} \triangleright T$ is window-feasible, i.e., there exists a common feasible-window, and evaluate $\omega_{\{x_1,x_2\}}(T)$ by assuming that player $\mathcal{A}$ plays through this window initially for every player $\mathcal{B}$ target in T. Essentially we have not used the tactical engine to evaluate the *prize-ts*, but rather used the tactical engine to evaluate the prizes and then compromised the information to evaluate the *prize-ts*.

Another potential weakness of the "black box" compromise is that we apply a *prize-ts* or *prize-mts* only at the root game tree level. An alternative is to embed a *prize-mts* in evaluating each game tree node in which case all evaluations would be over estimated meta-options rather than individual prizes. Figure 6.6 illustrates two possible evaluators involving a *prize-ts* at a general game tree node of PRIZE-DT. The GENERALIZED-TS-MAXIMIN and GENERALIZED-TS-MINIMAX evaluators modify the evaluators of Section 6.3.3 by taking maximin across those $\mathcal{B}$-lead prizes which overlap a *prize-ts* constructed specifically for that game tree node.



(a) GENERALIZED-TS-MAXIMIN



(b) GENERALIZED-TS-MINIMAX

Figure 6.6: PRIZE-DT evaluators incorporating a *prize-ts*

Coda

▼ Summary

In this chapter, we have considered contingent planning (without aggregation) by targeting prizes and formulated the Small-DMS as a two-frame implementation of the SPA/DMS. The tactical example traced through this chapter has compared the various tactical planning options which show that the nature of the planning is very different from pure routing considerations.

▼ Link

Tactical planning involves planning at the scale of targeting prizes. When problems are larger, and thus it is no longer computationally feasible to search the associated game trees, we must either consider only a subset of the prizes or consider agglomerating the prizes into clusters. We develop these ideas further in the next chapter in which we consider *medium problems.*

6.A Appendices to Chapter 6

6.A.1 Three Prize Reverse Lead Theorem

We consider applying the PRIZE-DT analysis to the following three prize problem with prizes $\{i_1, i_2, i_3\}$. Suppose $\mathcal{L}_A = \mathcal{R}_B = \{i_1, i_2\}$ and $\mathcal{R}_A = \mathcal{L}_B = \{i_3\}$ where $d_{Ai_1} < d_{Bi_1}$, $d_{Ai_2} < d_{Bi_2}$, $d_{Ai_3} > d_{Bi_3}$ and $\lambda = \infty$.

Then we can determine explicitly the evaluation of each child node of the root game tree node.

$$\tilde{v}_A(j; \mathcal{A} \oplus i_1, \mathcal{B} \oplus i_2) = \begin{cases} v_{i_1} + v_{i_2} + v_{i_3} & \text{if } d_{Ai_1} + d_{i_1 i_2} < d_{Bi_2} \\ v_{i_1} + \frac{1}{2}(v_{i_2} + v_{i_3}) & \text{if } d_{Ai_1} + d_{i_1 i_2} = d_{Bi_2} \\ v_{i_1} + v_{i_3} & \text{if } d_{Ai_1} + d_{i_1 i_3} < d_{Bi_2} + d_{i_2 i_3} \\ v_{i_1} + \frac{1}{2} v_{i_3} & \text{if } d_{Ai_1} + d_{i_1 i_3} = d_{Bi_2} + d_{i_2 i_3} \\ v_{i_1} & \text{if } d_{Ai_1} + d_{i_1 i_3} > d_{Bi_2} + d_{i_2 i_3} \end{cases}$$

$$\tilde{v}_A(j; \mathcal{A} \oplus i_2, \mathcal{B} \oplus i_1) = \begin{cases} v_{i_2} + v_{i_1} + v_{i_3} & \text{if } d_{Ai_2} + d_{i_1 i_2} < d_{Bi_1} \\ v_{i_2} + \frac{1}{2}(v_{i_1} + v_{i_3}) & \text{if } d_{Ai_2} + d_{i_1 i_2} = d_{Bi_1} \\ v_{i_2} + v_{i_3} & \text{if } d_{Ai_2} + d_{i_2 i_3} < d_{Bi_1} + d_{i_1 i_3} \\ v_{i_2} + \frac{1}{2} v_{i_3} & \text{if } d_{Ai_2} + d_{i_2 i_3} = d_{Bi_1} + d_{i_1 i_3} \\ v_{i_2} & \text{if } d_{Ai_2} + d_{i_2 i_3} > d_{Bi_1} + d_{i_1 i_3} \end{cases}$$

$$\tilde{v}_A(j; \mathcal{A} \oplus i_3, \mathcal{B} \oplus i_1) = \begin{cases} v_{i_3} + v_{i_2} & \text{if } d_{Ai_3} + d_{i_3 i_2} < d_{Bi_1} + d_{i_1 i_2} \\ v_{i_3} + \frac{1}{2} v_{i_2} & \text{if } d_{Ai_3} + d_{i_3 i_2} = d_{Bi_1} + d_{i_1 i_2} \\ v_{i_3} & \text{if } d_{Ai_3} + d_{i_3 i_2} > d_{Bi_1} + d_{i_1 i_2} \end{cases}$$

$$\tilde{v}_A(j; \mathcal{A} \oplus i_3, \mathcal{B} \oplus i_2) = \begin{cases} v_{i_3} + v_{i_1} & \text{if } d_{Ai_3} + d_{i_3 i_1} < d_{Bi_2} + d_{i_2 i_1} \\ v_{i_3} + \frac{1}{2} v_{i_1} & \text{if } d_{Ai_3} + d_{i_3 i_1} = d_{Bi_2} + d_{i_2 i_1} \\ v_{i_3} & \text{if } d_{Ai_3} + d_{i_3 i_1} > d_{Bi_2} + d_{i_2 i_1} \end{cases}$$

Also let

$$\begin{aligned} c &= \max\{\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2),\tilde{v}_A(j;\mathcal{A}\oplus i_2,\mathcal{B}\oplus i_1)\} \\ r &= \min\{\tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_1),\tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2)\} \end{aligned}$$

Lemma 6.A.1

$r \leq c$.

Proof:

Suppose that $r > c$. Then $\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2) < \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2)$ and $\tilde{v}_A(j;\mathcal{A}\oplus i_2,\mathcal{B}\oplus i_1) < \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_1)$.

- Suppose $d_{Ai_1} + d_{i_1i_2} < d_{Bi_2}$. Then $\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2) = v_{i_1} + v_{i_2} + v_{i_3} > v_{i_3} + v_{i_1} \geq \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2)$.
- Suppose $d_{Ai_1} + d_{i_1i_2} = d_{Bi_2}$. Then

$$\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2) = \max\{v_{i_1} + \tfrac{1}{2}(v_{i_2} + v_{i_3}), v_{i_1} + v_{i_3}\}.$$

Certainly $\tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2) \leq v_{i_3} + v_{i_1}$ and thus $\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2) \geq \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2)$.

Therefore $d_{Ai_1} + d_{i_1i_2} > d_{Bi_2}$. Similarly $d_{Ai_2} + d_{i_1i_2} > d_{Bi_1}$. Let

$$\begin{aligned} w_1 &= d_{Ai_1} + d_{i_1i_3} - d_{Bi_2} + d_{i_2i_3} \\ w_2 &= d_{Ai_2} + d_{i_2i_3} - d_{Bi_1} + d_{i_1i_3} \\ w_3 &= d_{Ai_3} + d_{i_3i_2} - d_{Bi_1} + d_{i_1i_2} \\ w_4 &= d_{Ai_3} + d_{i_3i_1} - d_{Bi_2} + d_{i_1i_2} \end{aligned}$$

- Consider the inequality $\tilde{v}_A(j;\mathcal{A}\oplus i_1,\mathcal{B}\oplus i_2) < \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_2)$:

	$w_4 < 0$	$w_4 = 0$	$w_4 > 0$
$w_1 < 0$	**false**	**false**	**false**
$w_1 = 0$	**true**	$v_{i_1} < v_{i_3}$	$v_{i_1} < \frac{1}{2}v_{i_3}$
$w_1 > 0$	**true**	$v_{i_1} < i_2 v_{i_3}$	$v_{i_1} < v_{i_3}$

Therefore $w_1 \geq 0$ is necessary.

- Consider the inequality $\tilde{v}_A(j;\mathcal{A}\oplus i_2,\mathcal{B}\oplus i_1) < \tilde{v}_A(j;\mathcal{A}\oplus i_3,\mathcal{B}\oplus i_1)$:

	$w_3 < 0$	$w_3 = 0$	$w_3 > 0$
$w_2 < 0$	**false**	**false**	**false**
$w_2 = 0$	**true**	$v_{i_2} < v_{i_3}$	$v_{i_2} < \frac{1}{2}v_{i_3}$
$w_2 > 0$	**true**	$v_{i_2} < i_2 v_{i_3}$	$v_{i_2} < v_{i_3}$

Therefore $w_2 \geq 0$ is also necessary.

Hence we have established that $d_{Ai_1} + d_{i_1i_3} - d_{Bi_2} - d_{i_2i_3} = w_1 \geq 0$ and $d_{Ai_2} + d_{i_2i_3} - d_{Bi_1} - d_{i_1i_3} = w_2 \geq 0$. Thus $d_{Ai_1} + d_{Ai_2} + d_{i_1i_3} + d_{i_2i_3} - d_{Bi_1} - d_{Bi_2} - d_{i_1i_3} - d_{i_2i_3} = d_{Ai_1} + d_{Ai_2} - d_{Bi_1} - d_{Bi_2} \geq 0$. However, this is impossible since $d_{Ai_1} < d_{Bi_1}$ and $d_{Ai_2} < d_{Bi_2}$.

Therefore $r \leq c$, as required. ■

An interesting observation from this lemma is that, even if $v_{i_3} \gg v_{i_1} + v_{i_2}$ and player $\mathcal{B}$ does not target prize i_3, player $\mathcal{A}$ does not target prize i_3 either.

Theorem 6.A.2

Reverse leading for player $\mathcal{A}$ is not necessary for three prize problems under the GENERALIZED-MINIMAX objective.

Proof:

Assume that player $\mathcal{A}$ has at least as many leads as player $\mathcal{B}$.

For reverse leading to be possible we must have $\mathcal{R}_A \neq \emptyset$, $\mathcal{R}_B \neq \emptyset$ and $\mathcal{F}_B \neq \emptyset$ (i.e. $|\mathcal{R}_B| \geq 2$). Since always $\mathcal{R}_A \cap \mathcal{R}_B = \emptyset$ and $\mathcal{L}_A \cup \mathcal{L}_B = (\mathcal{L}_A \cap \mathcal{L}_B) \cup \mathcal{R}_A \cup \mathcal{R}_B$ there is only one possible scenario:

- $\mathcal{L}_A = \mathcal{R}_B = \{i_1, i_2\}$ and $\mathcal{R}_A = \mathcal{L}_B = \{i_3\}$.

Therefore the assertion is true by Lemma 6.A.1. ■

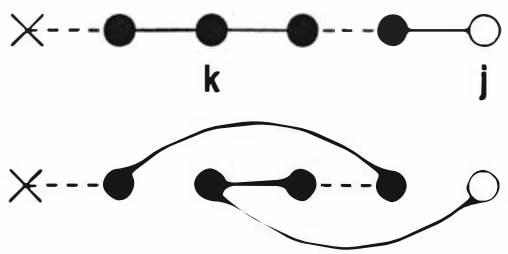


Figure 6.7: Subpath appendage two-exchange

6.A.2 Solving COOPERATE and GUARANTEE Subproblems

We now consider Branch-and-Bound (B&B) algorithms for the COOPERATE and GUARANTEE subproblems of Sections 6.2 and 6.3. To define a B&B algorithm we must specify a *bounding criterion* and a *branching rule*. Firstly we investigate a branching scheme in which we need not consider all remaining prizes.

6.A.2.1 Two-Optimal Appendages

Consider a depth-first B&B algorithm. Let Ψ_X be the working subpath through prizes from Q, let t_X be the projected arrival-time at the end of Ψ_X and suppose that $|\Psi_X| \geq 2$. We propose to append prize $j \in Q \setminus \Psi_X$ to subpath Ψ_X. Choose $k \in \Psi_X$ such that k is not the last prize on Ψ_X. Then consider the two-exchange defined by Figure 6.7, in which $\times$ denotes the vertex corresponding to the current player location, with change in distance $\delta(j,k)$. If $|\Psi_X| \leq 1$ or $\nexists\, k$ such that $\delta(j,k) < 0$, then j is a **two-optimal-appendage** of Ψ_X.

Lemma 6.A.3

We need only branch to those prizes $j \in Q \setminus \Psi_X$ which are two-optimal-appendages of Ψ_X.

Proof:

1. Ψ_X is path-two-optimal if and only if Ψ_X is constructed such that each prize appended is a two-optimal-appendage.

2. Consider the overall deadline λ. Let τ_i be the arrival time of player $\mathcal{X}$ at each prize i on Ψ_X. Certainly $t_X \leq \lambda$ if and only if $\tau_i \leq \lambda \; \forall \; i \in \Psi_X$. That is, we need only consider the arrival-time of player $\mathcal{X}$ at the end node of Ψ_X to establish guarantee of Ψ_X with respect to overall deadline λ.

3. Consider a set of deadlines ℓ_i on prizes $i \in Q$ determined by earliest arrival-times of player $\mathcal{Y}$ at each $i \in Q$. The *triangle inequality* holds on Q, i.e., $\ell_i \leq \ell_k + d_{ki} \; \forall \; i,k \in Q$.

   Suppose $\Psi_X \neq \emptyset$ and let j be the last prize on Ψ_X. Then $\tau_j \leq \ell_j$ if and only if $\tau_i \leq \ell_i \; \forall \; i \in \Psi_X$. That is, we need only consider arrival-time of player $\mathcal{X}$ at the end node of Ψ_X to establish guarantee of Ψ_X with respect to deadlines ℓ_i.

4. Hence, if we append prize $j \in Q \setminus \Psi_X$ to Ψ_X, where prize j is *not* a two-optimal-appendage of Ψ_X, then somewhere else in the B&B tree there will exist a B&B tree node (which we *will* process) at which Ψ_X will contain the same prizes in some different sequence and satisfying any deadlines as required, *but with an earlier* t_X, which can only give a better objective function value involving arrival-time or guaranteed value.

■

Checking if a prize is a two-optimal-appendage to Ψ_X requires only $O(n)$ operations. It may turn out that we end up doing more work by only branching to two-optimal-appendages than if we had branched to some non-two-optimal-appendages. Due to the nature of B&B, we might have established a better v_{best} earlier in the search process, hence pruning off more of the B&B tree earlier.

Similarly, it is possible to implement a three-optimal appendage rule, although it is yet to be determined whether the computational overhead is profitable in terms of fewer branches to explore.

We can now design B&B algorithms for the COOPERATE and GUARANTEE subproblems.

6.A.2.2 B&B: COOPERATE

We present a depth-first B&B algorithm to determine $\Omega(j)$ for game tree node j. Associate a value $\beta_{coop} = \sum_{j \in Q: \min\{t_A + d_{Aj}, t_B + d_{Bj}\} \leq \lambda} v_j$ with j; if we can show that $\Omega(j) \geq \beta_{coop}$, then we need not search any further. We implicitly search a B&B tree. Let i be a general B&B node defined by a pair of subpaths Ω_A and Ω_B with projected time-stamps t_A and t_B and accumulated values $v(\Omega_A)$ and $v(\Omega_B)$. Initialise with $v_{best} = 0$, $\Omega_A = \emptyset$ and $\Omega_B = \emptyset$; t_A and t_B are inherited from game tree node j.

Bounding Criteria. Fathom B&B node i if

$$v(\Omega_A) + v(\Omega_B) + \sum_{j \in Q: \min\{t_A + d_{Aj}, t_B + d_{Bj}\} \leq \lambda} v_j \leq v_{best} \tag{6.23}$$

Targeting. Lemma 6.A.3 shows that player $\mathcal{A}$ need only target those prizes $j \in Q \setminus (\Omega_A \cup \Omega_B)$ which are two-optimal-appendages of Ω_A and player $\mathcal{B}$ need only target those prizes $j \in Q \setminus (\Omega_A \cup \Omega_B)$ which are two-optimal-appendages of Ω_B. Hence let

$$Q_A = \{j \in Q \setminus (\Omega_A \cup \Omega_B) : t_A + d_{Aj} \leq \lambda \text{ and } j \text{ is a two-optimal-appendage of } \Omega_A\} \tag{6.24}$$

and let

$$Q_B = \{j \in Q \setminus (\Omega_A \cup \Omega_B) : t_B + d_{Bj} \leq \lambda \text{ and } j \text{ is a two-optimal-appendage of } \Omega_B\}. \tag{6.25}$$

Branching Rule. Branch to each prize $x \in Q_A$ and $y \in Q_B$ in turn. Algorithm 6.14 COOP_SLAVE(i) provides the details. The phrase "$x \in Q_A$ **or** $y \in Q_B$" requires some explanation: essentially it does not matter in which order we branch, but we only commit one player to a target-prize at each B&B node. What we actually do is to *sort* the prizes by one of the methods from Section 6.2.3.3; in particular, we adopt scoring function (v): *weighted nearest neighbour and tightest deadline.* We then select either $x \in Q_A$ or $y \in Q_B$ according to the minimum score.

Algorithm 6.14 procedure COOP_SLAVE(i)

Input: i // *B&B tree node.*
$Q' \leftarrow Q \setminus (\Omega_A \cup \Omega_B)$
$Q_A \leftarrow \{j \in Q' : t_A + d_{Aj} \leq \lambda$ and j is a two-optimal-appendage of $\Omega_A\}$
$Q_B \leftarrow \{j \in Q' : t_B + d_{Bj} \leq \lambda$ and j is a two-optimal-appendage of $\Omega_B\}$
$ub \leftarrow v(\Omega_A) + v(\Omega_B) + \sum_{j \in Q' : \min\{t_A + d_{Aj}, t_B + d_{Bj}\} \leq \lambda} v_j$
if $(ub > v_{best})$ **then**
  $v_{best} \leftarrow \max\{v_{best}, v(\Omega_A) + v(\Omega_B)\}$
  for $((x \in Q_A$ **or** $y \in Q_B)$ **and** $(v_{best} < \min\{\beta_{coop}, ub\}))$ **do**
    COOP_SLAVE$((i, \mathcal{A} \oplus x))$ **or** COOP_SLAVE$((i, \mathcal{B} \oplus y))$
  end
end

end

6.A.2.3 B&B: GUARANTEE

The PRIZE-GUARANTEE subproblem is an **Orienteering Problem with Deadlines** (OPD). We present a depth-first branch-and-bound (B&B) algorithm to determine $\Gamma(j, \mathcal{X})$ for game tree node j, assuming deadlines $\ell_j = t_Y + d_{Yj}$. Associate a value $\beta_{\text{guarantee}} = \Gamma(j) - v(P_X)$; if we can show that $\Gamma(j, \mathcal{X}) \geq \beta_{\text{guarantee}}$, then we need not search any further. We implicitly search a B&B tree. Let i be a general B&B node defined by a guaranteed subpath Γ_X with projected time-stamp t_X and accumulated value $v(\Gamma_X)$. Initialise with $v_{best} = 0$, $\Gamma_X = \emptyset$; t_X is inherited from game tree node j.

Bounding criterion. Fathom B&B node i if

$$v(\Gamma_X) + \sum_{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}} v_j \leq v_{best} \tag{6.26}$$

Targeting. Lemma 6.A.3 shows that player $\mathcal{X}$ need only target those prizes $j \in Q \setminus \Gamma_X$ which are two-optimal-appendages of Γ_X. Hence let

$$Q_X = \{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\} \text{ and } j \text{ is a two-optimal-appendage of } \Gamma_X\}. \tag{6.27}$$

Branching Rule. Branch to each prize $x \in Q_X$. Algorithm 6.15 GUARANTEE_SLAVE(i) provides the details. To implement "$x \in Q_X$" we *sort* the prizes by one of the methods from Section 6.2.3.3; in particular, we adopt scoring function (v): ***weighted nearest neighbour and tightest deadline.***

Heuristic Improvment of the Bound. Whenever a new best guaranteed subpath is found, we apply the local search scheme of Section 3.A.3 to heuristically improve the lower bound as much as possible before continuing with the B&B algorithm.

This concludes this appendix on the design of algorithms to solve the COOPERATE and GUARANTEE subproblems.

Algorithm 6.15 procedure GUARANTEE_SLAVE(i)

Input: i *// B&B tree node.*
$Q_X \leftarrow \{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}$ and j is a two-optimal-appendage of $\Gamma_X\}$
$ub \leftarrow v(\Gamma_X) + \sum_{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}} v_j$
if $(ub > v_{best})$ **then**
$v_{best} \leftarrow \max\{v_{best}, v(\Gamma_X)\}$
for $x \in Q_X$ **and** $v_{best} < \min\{ub, \beta_{\text{guarantee}}\}$ **do**
GUARANTEE_SLAVE$((i, \mathcal{X} \oplus x))$
end
end

end

CHAPTER 7

Strategic Planning for Medium Problems

Tactics is knowing what to do when there is something to do.
Strategy is knowing what to do when there is nothing to do.
— SAVIELLY TARTAKOVER, CHESS GRAND MASTER

Medium problems are those problems which have too many prizes to apply a tactical engine to the whole prize set, but which exhibit some useful, natural, cluster-like structure. The focus of this chapter is the design of the strategic engine CLUSTER-DT, a generalization of the tactical engine PRIZE-DT, which strategically analyses contingent sequences of clusters. We begin by gradually establishing the component building blocks of the CLUSTER-DT engine by analogy to the component building blocks of PRIZE-DT. Finally we extend the small DMS of Chapter 6 to incorporate CLUSTER-DT as a strategic engine which coordinates hierarchically with a tactical engine.

7.0 Introduction

Effective planning involves a tradeoff between planning horizon and the computational complexity of contingent planning, as identified by the SPA in Chapter 4. The PRIZE-DT tactical engine is routinely capable of tactical planning for up to approximately ten prizes. For problems involving

more prizes, we wish to take advantage of any natural structure exhibited in the layout or values of the prizes, such that approximately the same degree of strategic information can be made available. Since all possible sequences of prizes cannot be considered, it is necessary to aggregate the prizes. Hence the focus of targeting is scaled up from prizes to clusters of prizes in this chapter.

The **strategic planning problem** is to determine which cluster to target for a range of cluster target scenarios of the opponent. The difference between tactical planning and strategic planning is that the former focuses on prizes and the latter focuses on clusters. Clusters can be thought of as "exploded prizes." Thus a principal design consideration is the interaction of intra-cluster versus inter-cluster planning. Indeed the approach we take in this chapter is to use a tactical engine to handle much of the intra-cluster planning and generalize the concepts of PRIZE-GUARANTEE, PRIZE-PARANOID and PRIZE-DT to cluster targeting. This involves many technical design difficulties related to scaling the concepts from prizes to clusters.

A scenario engine which proposes scenario in terms of contingent sequences of clusters is called a **strategic engine**. Once an initial cluster target or sequence of clusters has been determined by a *global-monitor*, it remains to solve the tactical planning problem restricted to targeting some prize from the initial cluster target. Following the decision of which prize to target comes the choice of a single step. In this way we design an implementation of the SPA/DMS with three fixed hierarchical frames. Although this modular design is conceptually simple, in practice that are many technicalities and cases to develop.

We begin with types and structures of clusters and methods for construction of a clustering in Section 7.1. Then we consider the tactical planning problems of Chapter 6 with the additional constraint of visiting prizes within a cluster contiguously in Section 7.2. These define a suitable formulation for the *intra-cluster* component of the SPA implementation we are attempting.

Section 7.3 (together with Appendix 7.A.4 and Appendix 7.A.5) provide a precise technical specification of the the concepts required for players to target clusters and for the prizes within a cluster to be allocated between the players in a strategic scenario. We then have sufficient resolution of the technical difficulties to generalize PRIZE-PARANOID to CLUSTER-PARANOID (Section 7.4) and PRIZE-DT to CLUSTER-DT (Section 7.5).

Finally, Section 7.6 constructs a Medium-DMS, and hence, we have completed a fully-fledged implementation of the SPA.

7.1 Cluster and Family Structures

We begin with a discussion of a family-cluster structure, which is a clustering of the prizes such that the clusters may also be grouped into families of clusters. Clustering methods are presented, together with methods for possibly "thinning" the prize set by dropping insignificant prizes from consideration.

The primary motivation for considering clustering prizes is the necessity of pursuing computational tractability. For problems involving a sufficiently large number of prizes, we can no longer consider sufficient contingent sequences of prizes for a tactical analysis of the whole prize set.

The options are to discard some of the prizes in the short-term, or to undertake some strategic analysis and represent the prizes at a more appropriate level of detail or aggregation.

7.1.1 Clusters

This following defines a cluster for our purposes.

Definition 7.1.1

A **cluster** is a subset of prizes likely to be visited contiguously either by one player or by both players in a local conflict. □

The **prize region** is the smallest convex subset of the plane containing all the prizes. A **subregion** is a connected subset of the prize region. A cluster does not necessarily correspond to a subregion since a cluster need not contain all the prizes that are contained in the smallest convex subregion that does contain all the prizes in the cluster. Also a single subregion may contain two or more overlapping clusters, e.g., one cluster may contain prizes of stratified greater value (a "hot-spot") while another may be of negligible value (the "background").

Although traditionally the prizes in a cluster are likely to be spatially close, e.g. the "spherical" cluster, this is not necessarily the case since we could consider a string of prizes of good harvesting potential, a **stringy cluster**, to also fit our definition of a cluster.

The field of *Cluster Analysis* is fundamentally concerned with finding the natural structure of the prizes by location, particularly how many clusters best represent this natural structure. It does not consider the importance or value of individual prizes. Ideally we would like to be able to dynamically select or partition the prizes relative to player locations, granularity, isolation and concentration of prize value. In practice, however, we are interested only in the effectiveness of the clustering for later purposes. Hence, we do not intend to estimate the natural number of clusters, but rather impose a clustering on the prize set involving some given number of clusters. The method chosen for clustering the prizes is highly likely to influence the strategic analysis, and hence the performance, of the CLUSTER-DT strategic engine. However, our aim is to provide a variety of types of clusterings and infer which types are useful to the purposes of CLUSTER-DT. Recently, combinatorial meta-heuristics (see Section 3.A.1), such as simulated annealing and tabu search, have been applied to clustering (De Amorim, Barthélemy and Ribeiro [50] and Selim and Alsultan [193]) but these are beyond the scope of this thesis.

7.1.1.1 Traditional Agglomerative Clustering

The traditional objective of clustering is to maximize the intra-cluster cohesion (the **diameter**) and minimize the inter-cluster coupling (the **split**). We begin with a review of some simple clustering methods, based on the presentation of Everitt [65].

Cluster Notation. We use the notation '$[ci]$' for cluster, indexed by ci, whose elements are prizes $j \in [ci]$, and the notation '$\mathcal{C}$' for a clustering whose elements are clusters $[ci] \in \mathcal{C}$.

Algorithm 7.1 heuristic AGGLOMERATIVE CLUSTERING

Input: n *// number of prizes*
Input: k *// number of required clusters*

$[ci] \leftarrow \{i\}\ \forall i \in \{1, \ldots, n\}$ *// initially singleton clusters*
$m \leftarrow n$
while $(m > k)$ **do**
Find the "nearest" pair of distinct clusters and merge them.
end

end

Algorithm 7.1 gives the generic AGGLOMERATIVE CLUSTERING heuristic that starts with singleton clusters and successively merges the two *nearest* clusters until the required number of clusters is attained.

A specific clustering heuristic is defined by the providing the definition of a *distance*, $D_{[ci][cj]}$, between a distinct pair of clusters $[ci]$ and $[cj]$. Everitt [65] gives several simple possibilities including the following.

Single Linkage.

$$D_{[ci][cj]} = \min_{k \in [ci], \ell \in [cj]} d_{k\ell}$$

Complete Linkage.

$$D_{[ci][cj]} = \max_{k \in [ci], \ell \in [cj]} d_{k\ell}$$

Group Average.

$$D_{[ci][cj]} = \frac{1}{|[ci]||[cj]|} \sum_{k \in [ci]} \sum_{\ell \in [cj]} d_{k\ell}$$

Ward's Method. This is based upon minimizing an intra-cluster *Sum of Squares Error* (SSE). Let $\mathcal{C}$ be a clustering of all the prizes. The *centroid*, $C_{[ci]}$, of cluster $[ci]$ is the mean location of the prizes $j \in [ci]$. Let

$$SSE(\mathcal{C}) = \sum_{[ci] \in \mathcal{C}} \sum_{j \in [ci]} d^2_{jC_{[ci]}}.$$

Let $\mathcal{C}_{[ci][cj]}$ be the clustering formed from $\mathcal{C}$ by merging two distinct clusters $[ci]$ and $[cj]$ and define

$$D_{[ci][cj]} = SSE(\mathcal{C}_{[ci][cj]}) - SSE(\mathcal{C}).$$

These basic clustering methods use only inter prize distance and centroidal distance as objectives. The CPCP has other features which are equally as important: prize value and length of cluster traversal.

7.1.1.2 Prize Value Weighted Clustering

We can use the prize values in the clustering process. We now propose a simple adaptation of Ward's Method to incorporate prize values. The aim is to obtain clusters of prizes grouped around high valued prizes and to then add in the more widely dispersed prizes of lower value. Define the *weighted centroid*, $W_{[ci]}$, of cluster $[ci]$ as the weighted mean location of all the prizes $j \in [ci]$ where each prize $j \in [ci]$ has weight v_j, i.e., its prize value. Then redefine

$$SSE(\mathcal{C}) = \sum_{[ci]\in\mathcal{C}} \sum_{j\in[ci]} v_j d^2_{jC_{[ci]}} \tag{7.1}$$

Hence, the clustering of the prizes is a balance between closely grouping high valued prizes together and the opportunity of incorporating prizes of smaller value further from the cluster centroid.

Any clustering $\mathcal{C}$ can be improved with respect to $SSE(\mathcal{C})$, as defined Equation (7.1), and with respect to infeasibility with respect to the number of required clusters k. Let M be a large penalty value. We wish to minimize

$$\sum_{[ci]\in\mathcal{C}} \sum_{j\in[ci]} v_j d^2_{jC_{[ci]}} + M \min\{0, k - |\mathcal{C}|\} \tag{7.2}$$

We propose using the following local search heuristic with two local operators: ***shifting*** a prize from one cluster to another and ***swapping*** two prizes from different clusters. For these two operators, the change in (7.2) can be calculated in $O(1)$ complexity. A shift may merge a cluster with a singleton cluster (resulting in one less cluster) or create a singleton cluster by removing a prize from a cluster (resulting in one more cluster). Note that, for a sufficiently large M, a locally optimal clustering with respect to (7.2) must consist of exactly k clusters. A simple "hill-climbing" heuristic has been implemented in which the best shift or swap, from those which improve (7.2), is made until a local-optima is reached.

Cluster improvement can be applied to any clustering $\mathcal{C}$ including a *random clustering* consisting of either a specified number of clusters, or of an unspecified number of clusters, such that all possible clusterings are equally probable. Algorithms for constructing a random clustering are presented in Appendix 7.A.1. The cluster improvement heuristic can also be applied to the clustering consisting of all singleton clusters and the clustering consisting of exactly one cluster containing all the prizes.

7.1.1.3 Grid Clustering

An alternative initial clustering for subsequent cluster improvement can be constructed by partitioning the prizes. Section 8.2.1 and Appendix 8.A.1 present methods for building a grid-structure which is defined as a regular, rectilinear partition of the prize region into non-overlapping rectangles. A corresponding clustering of the prizes is obtained by defining the set of prizes in a nonempty grid-cell to be a cluster. Note that we do not know *a priori* exactly how many clusters will be created although it can be no more than the number of grid-cells. However, the initial clustering can be improved with respect to both the required number of clusters and $SSE(\mathcal{C})$. Although a grid-structured partition can always be imposed upon the prizes, the resulting initial (non-improved) clustering may be arbitrary or unnatural.

7.1.1.4 Stringy Clusters

Since players move from prize to prize, it is not necessarily wise to consider only spherical clusters. If an efficient harvest path is significant compared to the "background" of prizes, then it may be useful to extract it as a separate cluster. In this way, we simultaneously consider the length of cluster traversal and the total prize value, by considering naturally occurring **stringy clusters**, i.e., clusters which have some inherent path-like structure. We do not attempt to impose such a structure on a set of prizes but rather attempt to highlight that clustering methods also need to adapt to the unique elements of the CPCP.

Thus far we have considered only clustering methods which build spherical-like clusters which group prizes for explanatory purposes rather than for sequencing purposes. When harvesting a subset of prizes from a cluster a player moves from prize to prize along a subpath. Hence it is reasonable to consider defining harvesting subpaths which are "likely to be visited contiguously by one player or by both players in a local conflict" as a cluster, e.g., a subpath of valuable prizes which stands out from a background subregion of prizes of relatively insignificant value.

We propose two combinatorial subproblems, which are variations of the VRP, to construct k stringy clusters. We assume that the number k is given, i.e., we do not need to determine how many stringy clusters to construct.

Floating Path VRP (FPVRP). Determine a set of k floating[1] subpaths such that every prize is a member of exactly one subpath and the sum (or maximum) of the lengths of the subpaths is minimized.

Floating Path Prize Collecting VRP (FPPCVRP). Determine a set of k floating subpaths where each prize is a member of at most one subpath, each subpath has value at least $v_{thresh} > 0$ and the sum (or maximum) of the lengths of the subpaths is minimized. This problem is a generalization of the PCTSP to multiple 'vehicles'.

We have implemented a simple local search heuristic for the FPPCVRP subproblem which uses the same prize insertion, deletion and edge-exchange operators as in Appendix 3.A.2, plus inter-path operators *join* (append one subpath to another) and *transfer* (remove a prize from one subpath and insert it onto another subpath). Note, however, that we have not attempted to discern if the stringy clusters stand out from their background subregions.

In summary, stringy clusters offer a richer set of possibly significant strategic options, although these ideas may be more applicable to "dense" problems involving a large set of prizes.

7.1.1.5 Cluster Stratification

Not every prize needs to be considered in analysing a problem in terms of sequences of clusters. It is the value density of significant locations which needs to be represented by the clustering. Prizes which are isolated, have comparatively insignificant value or are loosely included in a cluster may be dropped from planning considerations.

[1] Recall from Section 3.3 that a *closed subpath* is one in which both the origin and destination locations are fixed, an *open subpath* is one in which the origin location is fixed and the destination location is free and a *floating subpath* is one in which both the origin and destination are free.

The strategic planning advantage of being able to remove some prizes from a clustering is that the new clustering (composed of the same number of clusters) may better represent the natural structure of the remaining prizes. The disadvantage is that, if too many prizes are removed, then the remaining prizes may not accurately reflect the natural structure of the full prize set and hence the strategic planning on the remaining prizes will be suspect and possibly vulnerable. Hence removing prizes is potentially beneficial in terms of both reduced computational effort and increased strategic accuracy. On the other hand, stratification could introduce tactics in which an insignificant prize, which is directly on the trajectory of a player, is overlooked and, hence, some post-tactic analysis of omitted prizes may be necessary.

Stratification methods. Pre-clustering stratification may involve removing the least significant prizes overall, up to some proportion of the mean prize value. Post-clustering stratification may involve removing the least significant prizes from each cluster, up to some proportion of the total cluster value either the poorest prizes or those prizes $j \in [ci] \in \mathcal{C}$ for which $\frac{v_j}{d_{jC_{[ci]}}}$ is sufficiently small. Cluster stratification can proceed incrementally and iteratively with cluster improvement.

Prize reconsideration. When should prizes which have been thinned out be reconsidered? The possibilities include the following three.

(i). Only when a new clustering is constructed.

(ii). Once the strategic plan (in terms of clusters) has been determined and before we consider a tactical plan (in terms of prizes). This would involve associating a thinned out prize with one or more clusters.

(iii). Once a tactical plan (in terms of prizes) has been determined. This would involve inserting thinned out prizes into the tactical plan.

We adopt option (i) since then thinned out prizes need not be considered in the strategic or tactical planning and because we assume that a similar computational budget is available throughout the play of the CPCP, regardless of how many prizes remain.

7.1.1.6 A Composite Clustering Strategy

The composite clustering strategy we have implemented is to firstly construct k_1 stringy clusters, by finding a good solution to an FPPCVRP, and to secondly construct k_2 spherical clusters, from the remaining prizes, perform a few iterations of improving (and removing some prizes from) those spherical clusters. The balance between the number of stringy clusters and the number of spherical clusters which is useful to CLUSTER-DT is of particular interest. We assume that the total number of clusters required is given but that the choice of how many stringy clusters versus how many spherical clusters is left to the strategy.

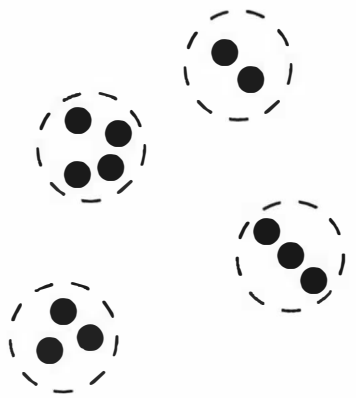


Figure 7.1: A Family-Cluster Structure

7.1.2 Families

The schematic problem structure of Figure 7.1—in which the shaded circles represent clusters of prizes and the larger dashed circles represent "natural" groupings of the clusters—suggests that clusters may be further aggregated.

Definition 7.1.2

A **family-cluster structure** is a clustering such that, for each player, a subset of clusters is itself grouped into sets (or families) of clusters. □

For the planning purposes of a single strategy, two family-cluster structures are maintained, one for each player. However, the two family-cluster structures share the same clustering of the prizes.

Example 7.1.3

Clustering: $\mathcal{C} = \{[1], [2], [3], [4]\}$.
$\mathcal{A}$-family: $\mathbb{F}_A = \{\{[1], [2]\}, \{[3], [4]\}\}$.
$\mathcal{B}$-family: $\mathbb{F}_B = \{\{[1]\}, \{[2], [3]\}\}$. □

There are two motivations for requiring a family-cluster structure. The first is that we may wish to exploit some natural structure involving clustering clusters into families such that the clusters in a family are likely to be visited contiguously by one player (or by both players) in strategic conflict. In this case the families for each player would be the same. In this context, families are not to be imposed artificially. The second is that CLUSTER-DT will require the solution of tactical subproblems involving a sequence of families. In this case the families for each player are likely to be different. Families are not evaluated as potential targets but are only a mechanism for grouping sets of clusters so as to reduce the number of clusters to consider as potential targets.

We wish to build a family-cluster structure which maps onto the natural structure of the problem. Suppose a clustering $\mathcal{C}$ is given (using any clustering method and any types of cluster) and suppose we wish to construct a family-cluster structure for both players consisting of f families. A straightforward approach is to use the same agglomerative clustering methods presented in Section 7.1.1. We begin with each family consisting of a single cluster. We can think of a

family $\mathcal{F}$ as a super-cluster consisting of the prizes $j \in [ci]$ such that $[ci] \in \mathcal{F}$. Merging two families is equivalent to merging the super-clusters and we continue to merge families until only f families remain. The cluster improvement method cannot be applied, however, since we would have to be able to shift a cluster of prizes from one super-cluster to another or swap to clusters of prizes.

We assume that the number of required families is given, rather than attempting to determine an appropriate number.

7.1.3 Summary

This section has reviewed basic clustering methods for traditional spherical clustering, developed improvement methods for prize value weighted spherical clustering, proposed a different criterion for a cluster (namely the stringy cluster) suitable for highlighting promising harvesting subpaths and considered further aggregation of clusters to families. Hence, given the number of required clusters and families, we are able to apply the composite clustering strategy and a simple agglomerative family method to construct a family-cluster structure.

7.2 Family-Cluster Precedence Constrained Tactical Subproblems

Cluster constrained problems are not new to vehicle routing. One of the main heuristic concepts for the VRP is the so-called "cluster-first route-second" approach (see, e.g., Hiquebran, Alfa, Shapiro and Gittoes [99]). Chisman [39], Jongens and Volgenant [121], Laporte, Potvin and Quilleret [140], Lokin [147] and Potvin and Guertin [177] study the **Clustered Travelling Salesman Problem** (CTSP) in which the cities are divided into clusters and each cluster of cities must be visited contiguously. Laporte and Nobert [139] consider yet another problem called the Generalized Travelling Salesman Problem (GTSP) in which the salesman must visit at least one city from each cluster. Laporte, Asef-Vaziri and Sriskandarajah [136] investigate the problem problem of determining a Hamiltonian circuit which passes through each cluster either at least once or exactly once—yet another problem called the GTSP.

The fundamental CRP subproblems that naturally arise in the context of clusters are those of local, tactical conflict on a cluster and harvesting of a cluster under little or no tactical conflict. The tactical engines ORIGINAL-DT and PRIZE-DT of Chapter 6 are now extended to the strategic scenario where a sequence of clusters is given for each player to follow in such a way that *a player cannot return to a cluster already visited* and may have to fulfill certain harvesting criteria on each cluster. This is a central idea throughout this chapter since we assume, for planning purposes, that a player plans to visit a cluster only once.

7.2.1 Tactical Planning Subproblem Formulation

Suppose, for planning purposes, a family-cluster structure is given. We now impose a number of cluster precedence and harvesting rules upon a small problem.

Rule 1 Cluster-No-Return.

Recall the definition for a *cluster* as a set of prizes that are *likely* to be visited contiguously. Taking this definition literally, apply a **cluster-no-return** rule to each player:

- Once a player has claimed a prize from some cluster, that player may not subsequently claim any prize from any previously visited cluster, i.e., those prizes claimed by a player from a cluster *must be* visited contiguously.

Rule 2 Family-No-Return.

Suppose that the clusters in a family are also to be visited contiguously. This is enforced by applying a **family-no-return** rule to each player:

- Once a player has claimed a prize from a cluster from some family, that player may not subsequently claim any prize from any family which contains a prize from any previously visited family.

Rule 3 ANY–ALL–PCTSP Cluster Harvesting

As a player moves through a cluster there are several possible requirements which we can impose on the subset of prizes player $\mathcal{A}$ must visit from that cluster. We impose one of three alternative harvesting requirements on the subset of prizes a player must visit from a cluster should that player claim any prize from that cluster.

ANY: Need not claim any further prizes from the cluster.

ALL: Must contiguously claim every prize from the cluster.

PCTSP: Must claim sufficient prizes from cluster $[ci]$ to contiguously claim at least $\eta_{[ci]}$ in value from cluster $[ci]$.

A cluster contained in both players' family-structure *must* be classified as an ANY cluster. We use the shorthand "ANY–ALL–PCTSP requirement" as applying whichever definition is specified. The most general application of these ANY–ALL–PCTSP requirements is that we could specify the requirement for each player for each cluster. However, we assume that one requirement applies to both players and all clusters and, if the PCTSP requirement is employed, then $\eta_{[ci]}$ is specified individually for each cluster $[ci]$.

Rule 4 Family-Sequence

Independently for each player, we may prescribe a strict sequence on the set of families, defining an *optional* **family-sequence** rule:

- The player must visit at least one prize from one cluster from a family before having the option of moving on to a cluster from the next family in the sequence.

We use the shorthand "$\mathcal{X}$-family-sequence" to denote that we apply the family-sequence requirement to player $\mathcal{X}$, since one player may have a family-sequence requirement while the other does

not. Note that there is *never* a prescribed sequence of clusters within a family, nor prizes within a cluster.

Rule 5 Final-Family

Suppose player $\mathcal{X}$ has a family-sequence rule. Then we may ***optionally*** impose an additional **final-family** rule on player $\mathcal{X}$:

- Player $\mathcal{X}$ must claim at least one prize from some cluster in the last cluster of the $\mathcal{X}$-family-sequence.

We use the shorthand "$\mathcal{X}$-final-family" (or the subscript '$_{FF}$') to denote that we apply the final-family requirement to player $\mathcal{X}$, since one player may have a final-family requirement while the other does not.

▼ Notation

The following examples illustrate the notation we will use for specifying a family-sequence and final-family.

Example 7.2.1

Single family: $\mathcal{A}\rightarrow\{[1],[2],[3],[4],[5]\}$.
No sequencing of families: $\mathcal{A}\rightarrow\{\{[1],[2]\},\{[3],[4]\},\{[5]\}\}$.
Families of a single cluster, no sequencing: $\mathcal{A}\rightarrow\{\{[1]\},\{[2]\},\{[3]\},\{[4]\},\{[5]\}\}$.
Families of a single cluster, $\mathcal{A}$-family-sequence, no-final-family: $\mathcal{A}\rightarrow\{[1]\}\rightarrow\{[3]\}\rightarrow\{[4]\}$.
$\mathcal{A}$-family-sequence, no final-family: $\mathcal{A}\rightarrow\{[1],[2],[3]\}\rightarrow\{[4],[5]\}$.
$\mathcal{A}$-family-sequence, final-family: $\mathcal{A}\rightarrow\{[1],[3]\}\rightarrow\{[4],[5]\}_{FF}$. □

▼ Comment

A small problem, with compulsory cluster-no-return, family-no-return and ANY–ALL-PCTSP requirements, stands on its own as a tactical planning problem for which we can propose simple extensions to the tactical engines ORIGINAL-DT and PRIZE-DT. Indeed these tactical engines (developed in Section 7.2.2) can immediately be used with the existing Small-DMS of Section 6.4.

The optional family-sequence and final-family requirements require *exogenous* specification. We propose these requirements solely for the purposes of the specification of the strategic engine CLUSTER-DT in Section 7.5. In this case we solve the prize-planning-problem within a *prize-frame* proposed by CLUSTER-DT. The final-family requirement is useful for forcing the player to move through the family-sequence to reach a target family, or to determine if this is possible. In this context we may need to record the exit-prize, exit-stamp (time-stamp) and exit-value for each player from the first cluster they visit.

7.2.2 Tactical Engines

We now discuss the straightforward changes required to ORIGINAL-DT and PRIZE-DT tactical engines in applying the cluster-no-return, family-no-return, family-sequence, final-family and ANY–ALL–PCTSP requirements. Several subproblems are investigated further in Appendix 7.A.2.

7.2.2.1 Cluster Targeting

Consider game tree node, j, of ORIGINAL-DT or PRIZE-DT. We associate with j the (projected) set of families $\mathbb{Q}_A$ and clusters $\mathcal{Q}_A$ which player $\mathcal{A}$ may still visit, and the (projected) set of families $\mathbb{Q}_B$ and clusters $\mathcal{Q}_B$ which player $\mathcal{B}$ may still visit, according to the family-no-return and cluster-no-return rules respectively.[2]

Suppose the planning path, P_X, is nonempty and the last prize claimed by player $\mathcal{X}$ on P_X was from cluster $[ci]$. We need to be able to determine if the ANY–ALL–PCTSP requirement of cluster $[ci]$ has been *satisfied* by player $\mathcal{X}$. Since the last prize claimed by player $\mathcal{X}$ was from cluster $[ci]$ then the ANY requirement has already been satisfied. For the ALL requirement, this involves checking if any prizes from cluster $[ci]$ remain unclaimed. For the PCTSP requirement, this involves recording the value accumulated from cluster $[ci]$ thus far and checking if this is at least $\eta_{[ci]}$.

Algorithm 7.2 TARGET-CLUSTERS defines $\mathcal{S}_X$: the set of **target-clusters** of player $\mathcal{X}$. Then let $Q_X = \{j \in Q : d_{Xj} \leq \lambda\} \cap \bigcup_{[ci]\in\mathcal{S}_X} [ci]$ be the set of **target-prizes** of player $\mathcal{X}$.

7.2.2.2 Prize Distances

Let us define the DISTANCE subproblem.

DISTANCE: Determine the earliest possible arrival-time of player $\mathcal{X}$ at each remaining prize in each cluster available to player $\mathcal{X}$ in an $\mathcal{X}$-future-family satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements of player $\mathcal{X}$.

Appendix 7.A.2.3 presents an algorithm for solving the distance subproblem. The arrival-time of player $\mathcal{X}$ at prize j is τ_{Xj}. Let $\tau_{Xj} = \infty$ for each prize j not available to player $\mathcal{X}$ and $\tau_{Xj} = t_X + d_{Xj}\ \forall\ j \in Q_X$.

7.2.2.3 Game Tree Bounding and Pruning Subproblems

Suppose player $\mathcal{X}$ has an $\mathcal{X}$-final-family requirement and hence, necessarily, an $\mathcal{X}$-family-sequence. The subproblem POSSIBLE determines if player $\mathcal{X}$ can still satisfy the $\mathcal{X}$-final-family requirement from a given game tree node j; if not, then the game tree search can be fathomed at that node.

POSSIBLE: Does there exist a subpath for player $\mathcal{X}$ which satisfies the family-sequence, cluster-no-return, family-no-return and ANY-ALL-PCTSP requirements and arrives at some prize from the final family in the family-sequence of player $\mathcal{X}$ no later than the global overall deadline λ?

[2]Note on nomenclature: we use i for a tree node, i for a prize, $[ci]$ for a cluster, Q for a set of prizes, $\mathcal{Q}$ for a family of clusters and $\mathbb{Q}$ for a set of families.

Algorithm 7.2 definition TARGET-CLUSTERS

// Determine $\mathcal{S}_X$, the set of target-clusters of player $\mathcal{X}$.
Input: j *// Game tree node.*
if $(P_X \neq \emptyset)$ **then**
  Let $[cx_0]$ be the cluster from which $\mathcal{X}$ last claimed a prize.
  if ANY–ALL–PCTSP *requirement on* $[cx_0]$ *not satisfied* **then**
    return$(\mathcal{S}_X \leftarrow \{[cx_0]\})$
  end
end
if $\mathcal{X}$*-family-sequence applies* **then**
  // Suppose the $\mathcal{X}$-family-sequence is $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$.
  if $\mathcal{X}$ *has claimed a prize from family* $\mathcal{G}_m$ **then**
    $\mathbb{S}_X \leftarrow \{\mathcal{G}_m\}$
  else if $(P_X = \emptyset)$ **then**
    $\mathbb{S}_X \leftarrow \{\mathcal{G}_1\}$
  else
    Let k be the least k such that no prize has been claimed by player $\mathcal{X}$ from family $\mathcal{G}_k$.
    $\mathbb{S}_X \leftarrow \{\mathcal{G}_{k-1}, \mathcal{G}_k\}$
  end
else
  $\mathbb{S}_X \leftarrow \mathbb{Q}_X$
end
return$(\mathcal{S}_X \leftarrow \bigcup_{\mathcal{G} \in \mathbb{S}_X} \mathcal{G})$

end

An algorithm to solve the subproblem POSSIBLE is detailed in Appendix 7.A.2.

At a game tree node j, the definitions of the cooperative-value $\Omega(j)$ (Definition 6.2.1), and the paranoid-value $\Gamma_{\mathcal{X}}(j)$, of player $\mathcal{X}$, need to modified to account for the cluster-no-return and family-no-return rules and the ANY–ALL–PCTSP requirement on each cluster visited. Algorithms that solve the subproblems COOPERATIVE VALUE and PARANOID VALUE are also detailed in Appendix 7.A.2.

7.2.2.4 Tactical Subproblem: FAMILY-PRIZE-GUARANTEE

This is the generalization of PRIZE-GUARANTEE to the family-cluster structure to handle cluster-no-return, family-no-return, family-sequence, final-family, and ANY–ALL-PCTSP requirements. The prize deadlines are determined using the DISTANCE subproblem of Appendix 7.A.2.3 and the prize-guarantee path is determined using the GUARANTEE subproblem of Appendix 7.A.2.6. Note that the DISTANCE subproblem incorporates the $\mathcal{Y}$-family-sequence information and the PARANOID subproblem incorporates the $\mathcal{X}$-family-sequence information.

7.2.2.5 Tactical Engines: FAMILY-PRIZE-PARANOID and FAMILY-ORIGINAL-DT

There is one further modification to PRIZE-PARANOID and ORIGINAL-DT required. Redefine

$$\mathcal{L}_B = \{i \in Q_B : \tau_{Bi} \leq \tau_{Ai}\}$$

so that a prize is only a lead-prize for player $\mathcal{B}$ if player $\mathcal{B}$ is able to arrive at that prize prior to player $\mathcal{A}$. The resulting tactical engines are called FAMILY-PRIZE-PARANOID and FAMILY-ORIGINAL-DT.

7.2.2.6 Tactical Engine: FAMILY-PRIZE-DT

Further modifications to PRIZE-DT required. Firstly, redefine

$$\begin{aligned}
\mathcal{D}_A &= \{i \in Q_A : \tau_{Ai} \leq t_B\} \\
\mathcal{L}_A &= \{i \in (Q_A \setminus \mathcal{D}_A) : \tau_{Ai} \leq \tau_{Bi}\} \\
\mathcal{R}_A &= Q_A \setminus (\mathcal{D}_A \cup \mathcal{L}_A) \\
\mathcal{F}_A &= \{\{i_1, i_2\} : i_1, i_2 \in \mathcal{R}_A, i_1 \neq i_2\}
\end{aligned}$$

and the corresponding sets for player $\mathcal{B}$.

Secondly, consider $\mathcal{B} \rhd \{y_1, y_2\} \in \mathcal{F}_B$. We know that both prizes y_1 and y_2 are eventually targetable by player $\mathcal{A}$ since $\tau_{Ai_1} < \tau_{Bi_1}$ and $\tau_{Ai_2} < \tau_{Bi_2}$. However, it may not be true that either sequence $y_1 \rightarrow y_2$ or $y_2 \rightarrow y_1$ is available to player $\mathcal{A}$ due to the cluster-no-return, family-no-return or family-sequence constraints.

- Suppose $\mathcal{A} \rhd \emptyset$. Since $\{y_1, y_2\} \in \mathcal{F}_B$, both prizes are available to player $\mathcal{A}$. Rather than rewrite all the expansion rules of PRIZE-DT for one-sided windows, we simple redefine *window feasibility*:

- If both $y_1 \rightarrow y_2$ and $y_2 \rightarrow y_1$ are available to player $\mathcal{A}$ then, as in Section 5.2,[3] we must determine if there exists a location, Z, for player $\mathcal{B}$ at time t_B such that neither $y_1 \rightarrow y_2$ or $y_2 \rightarrow y_1$ are feasible to player $\mathcal{A}$ at time t_A. Such a location Z must satisfy constraints (7.3)–(7.5).

$$t_A \geq t_B + d_{BZ} \tag{7.3}$$
$$d_{Ay_1} + d_{y_1y_2} \geq d_{Zy_2} \tag{7.4}$$
$$d_{Ay_2} + d_{y_1y_2} \geq d_{Zy_1} \tag{7.5}$$

  - If $y_2 \rightarrow y_1$ is available to player $\mathcal{A}$ but $y_1 \rightarrow y_2$ is not, then we must determine if there exists a location, Z, for player $\mathcal{B}$ at time t_B such that $y_2 \rightarrow y_1$ is not feasible to player $\mathcal{A}$ at time t_A. Such a location Z must satisfy constraints (7.3) and (7.5). In particular Z need not satisfy constraint (7.4). The set of locations Z satisfying constraints (7.3) and (7.5) is called a **y_1-sided feasibility window**.
  - If $y_1 \rightarrow y_2$ is available to player $\mathcal{A}$ but $y_2 \rightarrow y_1$ is not, then we must determine if there exists a location, Z, for player $\mathcal{B}$ at time t_B such that $y_1 \rightarrow y_2$ is not feasible to player $\mathcal{A}$ at time t_A. Such a location Z must satisfy constraints (7.3)–(7.4). In particular Z need not satisfy constraint (7.5). The set of locations Z satisfying constraints (7.3)–(7.4) is called a **y_2-sided feasibility window**.
  - If neither $y_1 \rightarrow y_2$ or $y_2 \rightarrow y_1$ are available to player $\mathcal{A}$ then no feasibility window is required. We say that the window-scenario is window-feasible even though it imposes no constraint on the passage of player $\mathcal{B}$.

- Suppose $\mathcal{A} \rhd x$. We must first determine which of y_1 and y_2 is available to player $\mathcal{A}$ before determining which sequences are available to player $\mathcal{A}$.

These modifications are all that are required to define the tactical engine: FAMILY-PRIZE-DT.

7.2.3 Example Tactical Problem Revisited

Chapter 6 compared the strategies determined by PRIZE-GUARANTEE and PRIZE-PARANOID (Section 6.1.3), ORIGINAL-DT (Section 6.2.4) and PRIZE-DT (Section 6.3.5) on the example problem of Figure 6.1. We now consider the tactics determined by FAMILY-PRIZE-GUARANTEE and FAMILY-PRIZE-DT for the same example problem, but with the clusters defined in Figure 7.2.

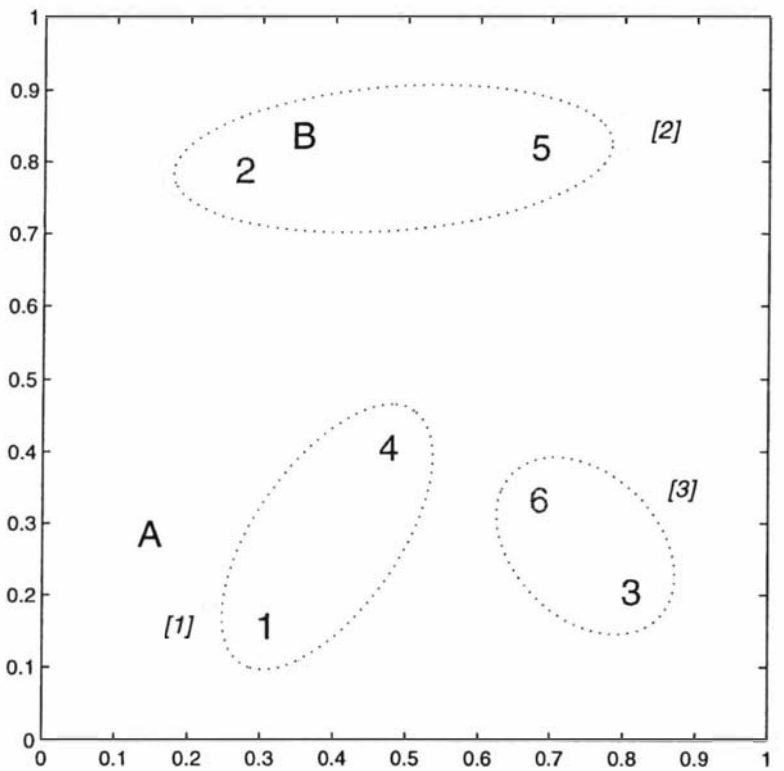
7.2.3.1 Analysis of FAMILY-PRIZE-GUARANTEE Tactics

Table 7.1 presents the maximal FAMILY-PRIZE-GUARANTEE subpaths for each player, which are identical to the maximal PRIZE-GUARANTEE subpaths.

7.2.3.2 Analysis of Player $\mathcal{A}$'s FAMILY-PRIZE-DT Tactics

Table 7.2(a) gives the FAMILY-PRIZE-DT root game table for each player $\mathcal{A}$. A MINIMAX analysis implies that player $\mathcal{B}$ will select a target from $\{1, 3, 4, 6\}$, i.e., from the MAXIMIN section of the

[3] Constraints (7.3)–(7.5) correspond to Constraints (5.29)–(5.31) respectively.



Prize	v	x	y
1	202	0.31	0.16
2	137	0.28	0.79
3	227	0.81	0.20
4	153	0.48	0.40
5	190	0.68	0.82
6	91	0.68	0.33
Total	1000		

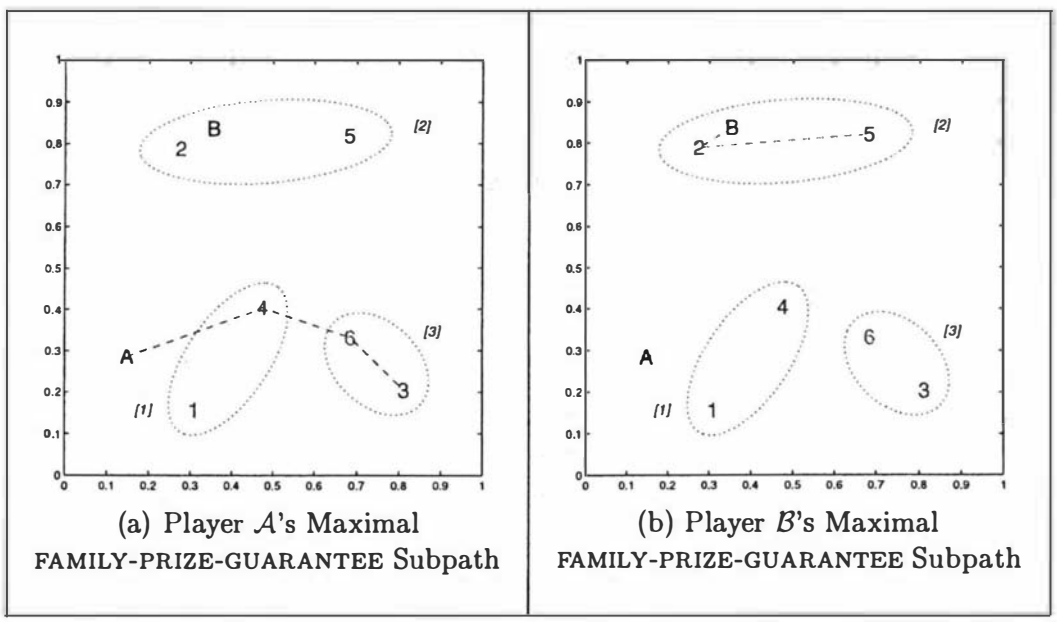
Player	x	y
$\mathcal{A}$	0.15	0.28
$\mathcal{B}$	0.36	0.84

Deadline	$\lambda = \infty$

Cluster	Prizes
[1]	$\{1, 4\}$
[2]	$\{2, 5\}$
[3]	$\{3, 6\}$

Figure 7.2: Example Tactical Problem with Clusters

Table 7.1: Tactical Example of FAMILY-PRIZE-GUARANTEE



(a) Player $\mathcal{A}$'s Maximal FAMILY-PRIZE-GUARANTEE Subpath

(b) Player $\mathcal{B}$'s Maximal FAMILY-PRIZE-GUARANTEE Subpath

game table. Hence player $\mathcal{A}$'s best lead prize is $\mathcal{A} \oplus 4$ assuming $\mathcal{B} \rhd \{6, 1\}$ or $\mathcal{B} \rhd \{3, 1\}$ with evaluation 661. Table 7.2(c) illustrates the commitment $\mathcal{A} \oplus 4$ and $\mathcal{B} \rhd \{6, 1\}$. This tactical scenario was previously explored in Section 6.3.5 with the same resulting evaluation.

There are significant differences between the player $\mathcal{A}$ root game tables of FAMILY-PRIZE-DT and PRIZE-DT (see Table 6.5(a) on page 186). For example, consider the scenario $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 2$. Table 7.2(b) gives the corresponding FAMILY-PRIZE-DT game table for player $\mathcal{A}$ whilst Table 6.5(b) gives the corresponding PRIZE-DT game table for player $\mathcal{A}$. A MINIMAX evaluation of the FAMILY-PRIZE-DT game table implies $\mathcal{B} \oplus 5$ and hence player $\mathcal{A}$'s best response is $\mathcal{A} \oplus 6$ or $\mathcal{A} \oplus 3$. The inference from the PRIZE-DT game table is also $\mathcal{A} \oplus 6$ or $\mathcal{A} \oplus 3$, but for player $\mathcal{B}$ it is different, namely $\mathcal{B} \rhd \{1, 6\}$ or $\mathcal{B} \rhd \{1, 3\}$. This difference in tactics between FAMILY-PRIZE-DT and PRIZE-DT is due to the *cluster-no-return* rule, which, in this case, prevents player $\mathcal{B}$ from claiming prize 5 unless it is claimed immediately following player $\mathcal{B}$ claiming prize 2. Table 7.2(d) illustrates the likely tactical game evolution anticipated by player $\mathcal{A}$ under the initial commitment to $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 2$. Note that player $\mathcal{B}$ eventually claims prize 1 since the *cluster-no-return* rule prevents player $\mathcal{A}$ from returning to cluster [1] following $\mathcal{A} \rightarrow 4 \rightarrow 6$.

7.2.3.3 Analysis of Player $\mathcal{B}$'s FAMILY-PRIZE-DT Tactics

Table 7.3(a) gives the FAMILY-PRIZE-DT root game table for each player $\mathcal{B}$. A MINIMAX analysis implies that player $\mathcal{A}$ will target $\mathcal{A} \oplus 4$ and, hence, player $\mathcal{B}$'s best response is $\mathcal{B} \oplus 5$ with evaluation 392.

Consider further the tactical scenario $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 55$. Table 7.3(b) gives the corresponding FAMILY-PRIZE-DT root game table for each player $\mathcal{B}$. A MINIMAX analysis implies that player $\mathcal{A}$

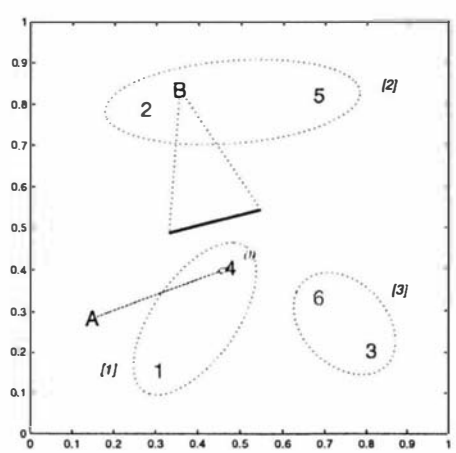
Table 7.2: Tactical Example of Player $\mathcal{A}$ FAMILY-PRIZE-DT Game Tables

	2	5	{4,6}	{4,3}	{4,1}	{6,3}	{6,1}	{3,1}
4	471	673	∞	∞	∞	∞	**661**	**661**
6	318	520	508	508	508	∞	571	508
1	673	673	520	520	520	520	682	∞
3	227	429	508	364	508	429	508	508
{2,5}	0	339						

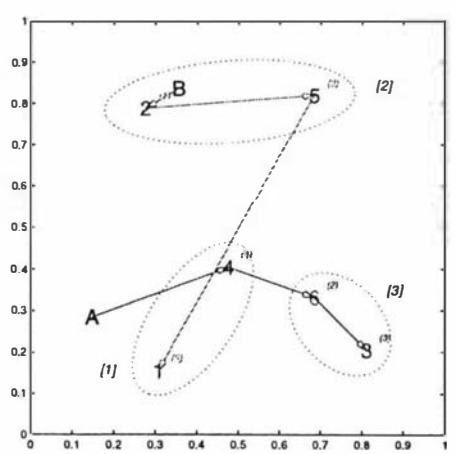
(a) Player $\mathcal{A}$ Root FAMILY-PRIZE-DT Game Table

	5	{1,6}	{1,3}	{6,3}
6	**471**	661	661	471
1	446	545	636	545
3	**471**	380	661	380

(b) $\mathcal{A} \oplus 4$ ($t_A = 0.35$) and $\mathcal{B} \oplus 2$ ($t_B = 0.09$)



(c) Initial Commitment $\mathcal{A} \oplus 4$ and $\mathcal{B} \rhd \{1,6\}$



(d) Player $\mathcal{A}$'s Anticipated Tactical Game Evolution Assuming $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 2$

will target $\mathcal{A} \oplus 1$ and, hence, player $\mathcal{B}$'s best response is $\mathcal{B} \rhd \{6,3\}$. Table 7.3(c) illustrates the subsequent likely tactical game evolution anticipated by player $\mathcal{B}$.

7.2.4 Summary

We have designed the tactical engines FAMILY-PRIZE-PARANOID, FAMILY-ORIGINAL-DT and FAMILY-PRIZE-DT by adapting the tactical engines PRIZE-PARANOID, ORIGINAL-DT and PRIZE-DT to incorporate family-cluster structures and a number of cluster precedence and harvesting constraints. In comparison with the ORIGINAL-DT and PRIZE-DT on the same set of prizes, we have reduced the breadth of the game tree since all planning paths which return to a cluster are now excluded; further reduction is also due to the streamlining effect of some undisputed clusters.

Two further modifications are necessary to implement CLUSTER-DT. These are motivated and defined in Section 7.3.5 and Appendix 7.A.5.

- Earliest possible arrival-times, $\tau_{\mathcal{X}j}$, at each prize in the first family of an $\mathcal{X}$-family-sequence may be specified—see Sections 7.3.5.3–7.3.5.4 and Appendix 7.A.5. The notation "$\tau_{\mathcal{X}} \ltimes [ci]$" means that player $\mathcal{X}$ arrives at each prize $i \in [ci]$ at time $\tau_{\mathcal{X}i}$.
- Dynamic and static cluster window distance correction (see Appendix 7.A.5). We use the notation "$\{[ci]\}_{\langle \mathbb{W}_1 \rangle} \rightarrow \{[cj]\}$" to mean that we apply a dynamic cluster window distance correction to d_{ij} $\forall\, i \in [ci]$ and $j \in [cj]$.

7.3 Strategic Cluster Building Blocks

The overall focus of this chapter is to design a PRIZE-DT-like method which strategically analyses the problem in terms of contingent sequences of *clusters*. In this section we design the necessary *strategic building blocks* of CLUSTER-DT analogously to the corresponding building blocks of PRIZE-DT. Subproblems involving planning paths of prizes through sequences of clusters arise naturally in this context. Several variations of the TSP in which the cities are partitioned into clusters have been studied previously. Each of these problems arise in various guises throughout this section.

▼ Building Blocks Road Map

We begin, in Section 7.3.1, by considering a generalization of the idea of *guarantee* of a prize by a player to domination of a cluster. A prize can be collected by visiting its single point location. Since there are usually more than one prize in a cluster, decisions must be made as to which subset of prizes from a cluster are selected, and in which sequence. It is useful to make these prize decisions in the light of which cluster is to be the next target, i.e., to *look through* a cluster in targeting the next cluster and *move through* a cluster with the next target cluster fixed. Section 7.3.2 develops these concepts. Then follow the three key building blocks: Sections 7.3.3–7.3.5 generalize PRIZE-GUARANTEE, the two prize problem and window feasibility to clusters and to the look-through/move-through paradigm. These building blocks are the CLUSTER-GUARANTEE strategic subproblem, the two cluster problem and the cluster feasibility windows.

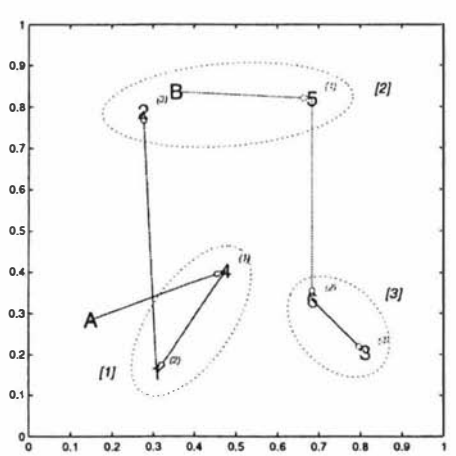
Table 7.3: Tactical Example of Player $\mathcal{B}$ FAMILY-PRIZE-DT Game Tables

	4	6	1	3	{2,5}
2	339	492	455	583	∞
5	[392]	545	327	545	661
{4,6}	0	492	480	571	
{4,3}	0	492	480	492	
{4,1}	0	492	480	492	
{6,3}	0	0	480	571	
{6,1}	339	339	318	571	
{3,1}	339	492	0	492	

(a) Player $\mathcal{B}$ Root FAMILY-PRIZE-DT Game Table

	1	6	3
2	327	327	327
{6,1}	508	529	620
{6,3}	[645]	508	508
{1,3}	417	327	392

(b) $\mathcal{A} \oplus 4$ and $\mathcal{B} \oplus 5$



(c) Player $\mathcal{B}$'s Anticipated Tactical Game Evolution

7.3.1 Dominance

In the PRIZE-DT method we classified each remaining prize j according to

- $t_A + d_{Aj} < t_B + d_{Bj}$ — *guaranteed* to player $\mathcal{A}$.
- $t_A + d_{Aj} = t_B + d_{Bj}$ — *sharable* prize.
- $t_A + d_{Aj} > t_B + d_{Bj}$ — *guaranteed* to player $\mathcal{B}$.

We wish to generalize these ideas to remaining clusters. If a cluster is a singleton then the cluster definitions should be equivalent to the prize definitions.

Definition 7.3.1

A cluster is **dominated** by player $\mathcal{X}$ if we *estimate* that player $\mathcal{Y}$ expects an *insignificant* value of prizes from a *tactical engagement* on that cluster. A cluster is **non-dominated** if neither player dominates that cluster. □

There are two keys to the definition of a *dominated* cluster. A *tactical engagement* on cluster $[ci]$ is playing on that cluster in isolation of other remaining clusters. We require an evaluation method for *estimating* the value that the opponent expects from the tactical engagement and a threshold value which defines what is a *significant* value. At most one player should be able to dominate each cluster.

7.3.1.1 Operational Definitions of Dominance

There are two operational definitions of *dominance* that we consider in for CLUSTER-DT.

PRIZE-GUARANTEE: If player $\mathcal{Y}$ cannot guarantee a prize from cluster $[ci]$, then player $\mathcal{X}$ *dominates* $[ci]$.

PRIZE-DT: If player $\mathcal{Y}$ has zero evaluation from cluster $[ci]$ applying PRIZE-DT, then player $\mathcal{X}$ *dominates* $[ci]$.

If a cluster is dominated by player $\mathcal{X}$ under the PRIZE-DT definition, then it is also dominated by player $\mathcal{X}$ under the PRIZE-GUARANTEE definition. The reverse is not necessarily true.

Some alternative operational definitions of *dominance* are possible.

ORIGINAL-DT: If player $\mathcal{Y}$ has zero evaluation from cluster $[ci]$ applying ORIGINAL-DT, then player $\mathcal{X}$ *dominates* $[ci]$.

PRIZE-DT-THRESHOLD: Associate with each cluster $[ci]$ a value $\zeta_{[ci]}$, where $0 \leq \zeta_{[ci]} \leq v([ci])$ and $v([ci])$ is the value sum of the prizes $j \in [ci]$. If both players have evaluation $\geq \zeta_{[ci]}$ from cluster $[ci]$ applying PRIZE-DT, then $[ci]$ is *non-dominated*. If player $\mathcal{X}$ has evaluation $\geq \zeta_{[ci]}$ from cluster $[ci]$ applying PRIZE-DT and player $\mathcal{Y}$ cannot, then player $\mathcal{X}$ *dominates*. If neither player has evaluation $\geq \zeta_{[ci]}$ from $[ci]$ applying PRIZE-DT, then, if player $\mathcal{Y}$ has zero evaluation from $[ci]$ applying PRIZE-DT, then player $\mathcal{X}$ *dominates* $[ci]$.

ORIGINAL-DT-THRESHOLD: Similarly.

PRIZE-GUARANTEE-THRESHOLD: Similarly.

PRIZE-DT-ALL: If player $\mathcal{X}$ has evaluation $v([ci])$ from cluster $[ci]$ applying PRIZE-DT, then player $\mathcal{X}$ *dominates* $[ci]$.

ORIGINAL-DT-ALL: Similarly.

PRIZE-GUARANTEE-ALL: Similarly.

These are not considered further since the PRIZE-GUARANTEE and PRIZE-DT are sufficiently illustrative of the black box approach and the application of further definitions would become repetitive. The two chosen were preferred since they are straightforward to compute and reflect the fact that those clusters deemed non-dominated are those clusters on which a tactical conflict can occur.

7.3.1.2 Dominance and Cluster-lead

We now reveal the relationship between *dominance* and the generalization of *lead-prize* to clusters.

Definition 7.3.2

Cluster $[ci]$ is a **cluster-lead** for player $\mathcal{X}$ if cluster $[ci]$ is available to player $\mathcal{X}$ (in the family-no-return sense) and either

- cluster $[ci]$ is not available to player $\mathcal{Y}$ (in the family-no-return sense); or
- cluster $[ci]$ is available to player $\mathcal{Y}$ (in the family-no-return sense) and player $\mathcal{Y}$ does not dominate cluster $[ci]$.

□

If cluster $[ci]$ is non-dominated and available to both players (in the family-no-return sense), then it is a *cluster-lead* for both players. It is the definition of *cluster-lead* that generalizes the definition *guaranteed prize* and *shareable prize*.

7.3.2 Look-through

A **cluster planning path**, $\mathbb{P}_A$, associated with player $\mathcal{A}$, consists of a sequence of clusters, a projected location A and a projected time-stamp t_A. A prize planning path, P_A, associated with $\mathbb{P}_A$, consists of a sequence of prizes visited from the sequence of clusters visited in $\mathbb{P}_A$. These are initial definitions suitable for our present purposes but will be expanded as necessary to enable the design of the component building blocks of CLUSTER-DT.

At each decision node of the PRIZE-DT game tree we have a *projected location* and *projected time-stamp* for each player associated with the planning paths P_A and P_B. Consider the scenario $\mathcal{A}{\rightarrow}P_A{\oplus}i$ and $\mathcal{B}{\rightarrow}P_B{\oplus}j$, i.e., where each player commits to a target-prize. We project player $\mathcal{A}$ to the location of prize i and increase the time-stamp by the direct distance. Similarly for player $\mathcal{B}$.

When generalizing from prize-objects to cluster-objects, we generalize this process of updating the player locations and time-stamps in *extending* the players' corresponding *cluster planning paths*. Below are three possible generalizations.

1. **Direct generalization.** Suppose player $\mathcal{A}$ commits to cluster $[ci]$. We determine an intra-cluster subpath with a corresponding exit-prize from cluster $[ci]$, update the player $\mathcal{A}$ projected location to that of the exit-prize, and update the projected time-stamp by the length of that subpath.

2. **One cluster look-through.** When determining an exit-prize from a cluster it would be useful to know which cluster is likely to be targeted next, so that the exit-prize from the cluster is essentially "optimized" for the next cluster target; it would also be useful to determine when the players move together from cluster to cluster in *tactical conflict* on each cluster. This means at each decision node we **look through** a committed cluster and propose scenarios for which clusters should be considered next.

3. **Entire cluster planning path look-through** or **m-cluster look-through.** We could **look through** a sequence of two or more clusters in our planning path. The extreme of looking through the entire cluster planning path is that we never update the location and time-stamp of each player and look through the entire sequence of prizes thus far to propose scenarios for which clusters should be considered next.

The difficulty with the *direct generalization* is that, if $t_B \ll t_A$, then it remains possible for a tactical conflict to occur on cluster $[ci]$ which player $\mathcal{A}$ is planning to move through. Hence we must have a facility for remembering that player $\mathcal{A}$ is committed to cluster $[ci]$ but has not yet determined a subpath through that cluster.

The difficulty with the *m-cluster look-through* and the *entire cluster planning path look-through* generalizations is the discrepancy in the projected time-stamp of each player. The decisions made which are based upon those projected locations and time-stamps must reflect the information that is available at the time when the decision is made. Player $\mathcal{A}$'s subpath through the preceding clusters is too flexible for this purpose since, in planning this subpath, player $\mathcal{A}$ assumes that it knows too much in advance about which sequence of clusters player $\mathcal{B}$ intends to follow, which player $\mathcal{A}$ would not know in making the initial steps of such a subpath. Also, the repetitive computation of such long subpaths through the cluster planning paths would be computationally expensive.

Hence, the *one cluster look-through* generalization is a useful compromise and is adopted throughout this chapter. It remains to further generalize *dominance* of a cluster when either player looks through some cluster (see Section 7.3.2.1) and to generalize the process by which each player moves through the cluster they target, updating the player's projected location and time-stamp (see Sections 7.3.2.5 and 7.3.2.6).

We use the notation "$\mathcal{A} \rhd [ci]$" to indicate that player $\mathcal{A}$ looks through cluster $[ci]$.

7.3.2.1 Dominance with Look-through

We now turn to defining *dominance* of a cluster when either or both players look through a cluster. Figure 7.3 illustrates the three general cases of one cluster look-through. In each case we require a definition of dominance of cluster [1]. As $\mathcal{A} \rhd$ [2] in cases (i) and (iii) there are several

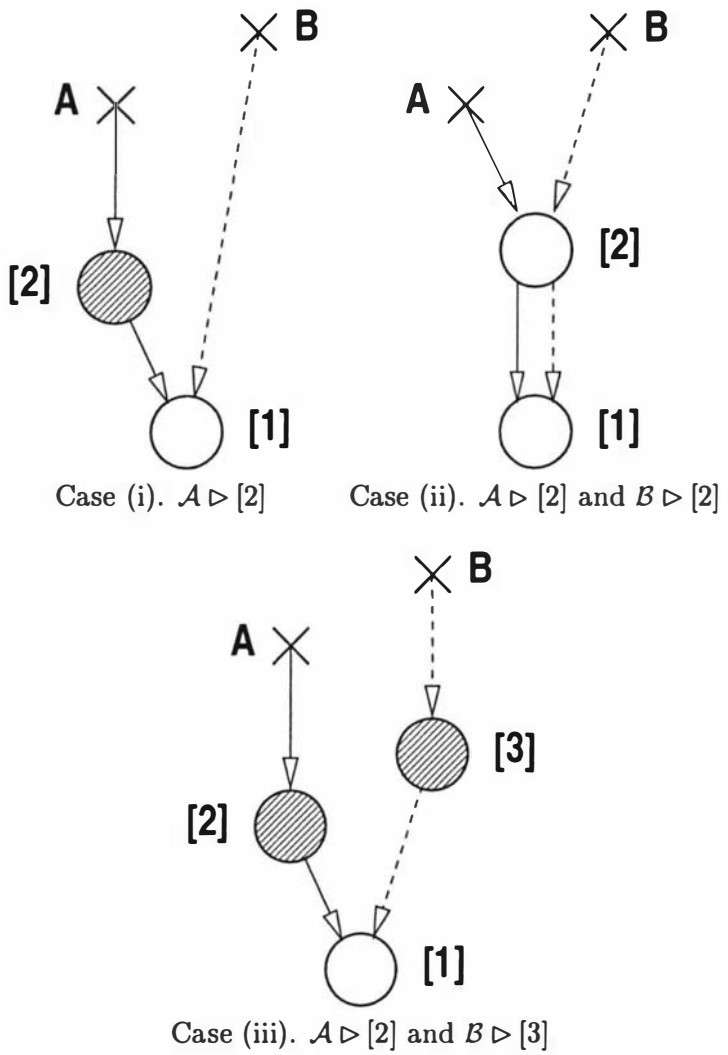


Figure 7.3: Cases of one cluster look-through dominance

possible requirements which could be imposed on the subset of prizes player $\mathcal{A}$ must visit from cluster [2] en route to cluster [1]; similarly as $\mathcal{B} \triangleright [3]$ in case (iii). These clusters are shaded.

In general, if player $\mathcal{A}$ looks through cluster $[cx_0]$ and either player $\mathcal{B}$ does not look through any cluster, or player $\mathcal{B}$ looks through cluster $[cy_0]$ (where $[cx_0] \neq [cy_0]$), we apply an ANY–ALL-PCTSP harvesting requirement as in Section 7.2.1, the same requirement on both players.

Operational Definitions of Dominance

The two operational definitions of dominance from Section 7.3.1.1, PRIZE-GUARANTEE and PRIZE-DT, are adapted slightly by replacing PRIZE-GUARANTEE with FAMILY-PRIZE-GUARANTEE (see Section 7.2.2.4) and replacing PRIZE-DT with FAMILY-PRIZE-DT (see Section 7.2.2.6).

Case (i). $\mathcal{A} \triangleright [2]$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A} \rightarrow \{[2]\} \rightarrow \{[1]\}_{FF}$ and $\mathcal{B} \rightarrow \{[1]\}_{FF}$. If player $\mathcal{Y}$ has zero evaluation then, $\mathcal{X}$ *dominates* cluster [1].

Case (ii). $\mathcal{A} \triangleright [2]$ and $\mathcal{B} \triangleright [2]$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[2]\}{\rightarrow}\{[1]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[2]\}{\rightarrow}\{[1]\}_{FF}$. If player $\mathcal{Y}$ has zero evaluation then, $\mathcal{X}$ *dominates* cluster [1].

Case (iii). $\mathcal{A} \triangleright [2]$ and $\mathcal{B} \triangleright [3]$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[2]\}{\rightarrow}\{[1]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[3]\}{\rightarrow}\{[1]\}_{FF}$. If player $\mathcal{Y}$ has zero evaluation then, $\mathcal{X}$ *dominates* cluster [1].

7.3.2.2 Preview of Move-through

We now preview the process of player $\mathcal{X}$ moving through cluster $[ci]$ when player $\mathcal{X}$ looks through cluster $[ci]$ to some other cluster. The primary motivation is that a cluster is moved through only once. We consider either one player moves through the cluster or both players move through the cluster simultaneously. That cluster is subsequently not available for either player to target, even if prizes from that cluster remain unclaimed. This is the essence of considering of cluster-targets.

For player $\mathcal{X}$ to move through cluster $[ci]$ we require that either both players look through cluster $[ci]$ or player $\mathcal{X}$ dominates cluster $[ci]$. It turns out that this is not quite restrictive enough, as a *deadlock* problem can occur. Consider the scenario in which player $\mathcal{A}{\rightarrow}[1]{\rightarrow}[2]$ and player $\mathcal{B}{\rightarrow}[2]{\rightarrow}[1]$. Suppose that player $\mathcal{A}$ does not dominate cluster [1] and player $\mathcal{B}$ does not dominate cluster [2] according to the ANY requirement. Can player $\mathcal{A}$ move through cluster [1] or player $\mathcal{B}$ move through cluster [2]?

Definition 7.3.3

Deadlock occurs when the players do not look through the same cluster but *neither* player dominates their look-through-cluster given their proposed cluster-lead and the proposed cluster-lead of the opponent. □

Deadlock is common in situations where resources are shared, such as computer Operating Systems. Brookshear [29] shows that deadlock cannot occur *unless* all three of the following conditions are satisfied.

1. There is competition for non-shareable resources.
2. The resources are requested on a partial basis; that is, having received some resources, a program will return later to request more.
3. Once a resource has been delegated, it cannot forcibly be retrieved.

Hence deadlock can be prevented by removing any one of them.

We now turn to algorithms which resolve this potential problem of deadlock so that one player can move through a cluster.

7.3.2.3 Look-through: Deadlock Resolution

There are potentially three scenarios in which *deadlock* can occur when employing either the ANY or PCTSP requirement.

(i). $\mathcal{A}{\rightarrow}[1]{\rightarrow}[2]$ and $\mathcal{B}{\rightarrow}[2]{\rightarrow}[1]$ (see Figure 7.4); or

(ii). $\mathcal{A}$→[1]→[3] and $\mathcal{B}$→[2]→[3] (see Figure 7.5); or

(iii). $\mathcal{A}$→[1]→[3] and $\mathcal{B}$→[2]→[4] (see Figure 7.6)

The following lemma is essential in the resolution of deadlock.

Lemma 7.3.4

If the ALL definition of domination is used then *deadlock* cannot occur.

Proof:

For player $\mathcal{A}$ to be able to claim a prize outside of cluster [1] there must be no prizes remaining available in cluster [1], by definition of the ALL requirement. Similarly, for player $\mathcal{B}$ to be able to claim a prize outside of cluster [2] there must be no prizes remaining available in cluster [2], by definition of the ALL requirement. Hence, we cannot simultaneously have player $\mathcal{A}$ claiming a prize from cluster [2] and player $\mathcal{B}$ claiming a prize from cluster [1], regardless of which other clusters they may be required to travel through. ■

We employ a *locking mechanism* to ensure that deadlock can be resolved. When considering any of the cases (i)–(iii) above, we firstly place a **lock** on each player. The **deadlock problem** is the problem of determining which player to unlock, or possibly whether to unlock both players, thus allowing the possibility of one or both to move through their respective look-through-cluster.

Lemma 7.3.4 implies that we can always apply the ALL requirement to resolve a deadlock problem. Algorithm 7.3 resolves the case in Figure 7.4, Algorithm 7.4 resolves the case in Figure 7.5 and Algorithm 7.5 resolves the case in Figure 7.6. In each of these figures, the dashed arrows show the presumed, continued cluster-path for the purposes of determining dominance.

7.3.2.4 Cluster-lead and Direct-cluster-lead with Look-through

We now define **cluster-lead** and **direct-cluster-lead** where at least one player looks through a cluster. For cases (i) and (ii) of Figure 7.3 the definition of *cluster-lead* is the same as in Definition 7.3.2. Direct-cluster-leads are not required for cases (i) and (ii). By design, case (i) never occurs in CLUSTER-DT (see Section 7.5.1), but case (i) does occur in CLUSTER-PARANOID (see Section 7.3.3), in which case there is no need to distinguish between cluster-lead and direct-cluster-lead. In case (ii) the cluster must have an ANY requirement and, hence, there is no meaningful sense of direct-lead for clusters.

For case (iii) we need to make some modifications. Suppose $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$ where $[cx_0] \neq [cy_0]$. We wish to be able to classify each cluster for each cluster as either a *direct-cluster-lead*, a *cluster-lead* or neither.

If cluster $[cy_0]$ is available to player $\mathcal{A}$ (in the family-no-return sense) and cluster $[cx_0]$ is available to player $\mathcal{B}$ (in the family-no-return sense), then cluster $[cy_0]$ is a candidate direct-cluster-lead for player $\mathcal{A}$, and cluster $[cx_0]$ is a candidate direct-cluster-lead for player $\mathcal{B}$. Apply Algorithm 7.3 RESOLVE DEADLOCK TWO CLUSTER.

- if *only* player $\mathcal{A}$ is unlocked then, cluster $[cy_0]$ is a direct-cluster-lead for player $\mathcal{A}$.
- if *only* player $\mathcal{B}$ is unlocked then, cluster $[cx_0]$ is a direct-cluster-lead for player $\mathcal{B}$.

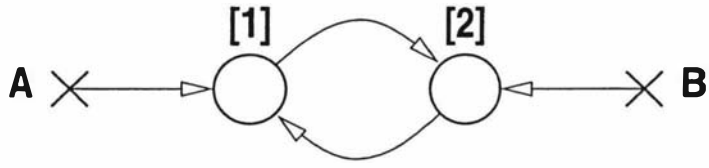


Figure 7.4: Possible deadlock scenario: $\mathcal{A}\rightarrow[1]\rightarrow[2]$ and $\mathcal{B}\rightarrow[2]\rightarrow[1]$.

Algorithm 7.3 procedure RESOLVE DEADLOCK TWO CLUSTER

// Determine which players to unlock, resolving deadlock situation if
// necessary.

Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[2]$ according to the prescribed ANY–ALL–PCTSP definition.
if *neither dominates* **then**
    // $\exists$ Deadlock. Break by replacement with:
    Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[2]$ according to the ALL definition.
end
if *both dominate* **then**
    Unlock both player $\mathcal{A}$ and player $\mathcal{B}$.
else if *player $\mathcal{A}$ dominates* [1] **then**
    Unlock player $\mathcal{A}$ only.
else
    Unlock player $\mathcal{B}$ only.
end

end

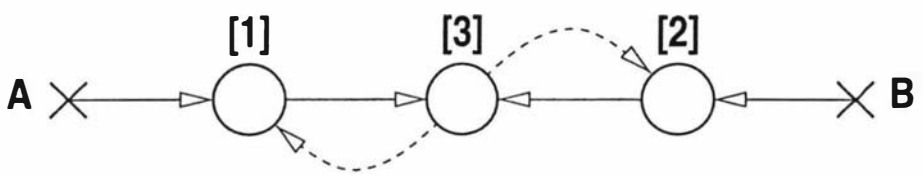


Figure 7.5: Possible deadlock scenario: $\mathcal{A}\rightarrow[1]\rightarrow[3]$ and $\mathcal{B}\rightarrow[2]\rightarrow[3]$.

Algorithm 7.4 procedure RESOLVE DEADLOCK THREE CLUSTER

// Determine which players to unlock, resolving deadlock situation if
// necessary.

Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[3]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the prescribed ANY–ALL–PCTSP definition.
if *neither dominates* **then**
    // $\exists$ Deadlock. Break by replacement with:
    Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[3]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the ALL definition.
end
if *both dominate* **then**
    Unlock both player $\mathcal{A}$ and player $\mathcal{B}$.
else if *player $\mathcal{A}$ dominates* [1] **then**
    Unlock player $\mathcal{A}$ only.
else
    Unlock player $\mathcal{B}$ only.
end

end

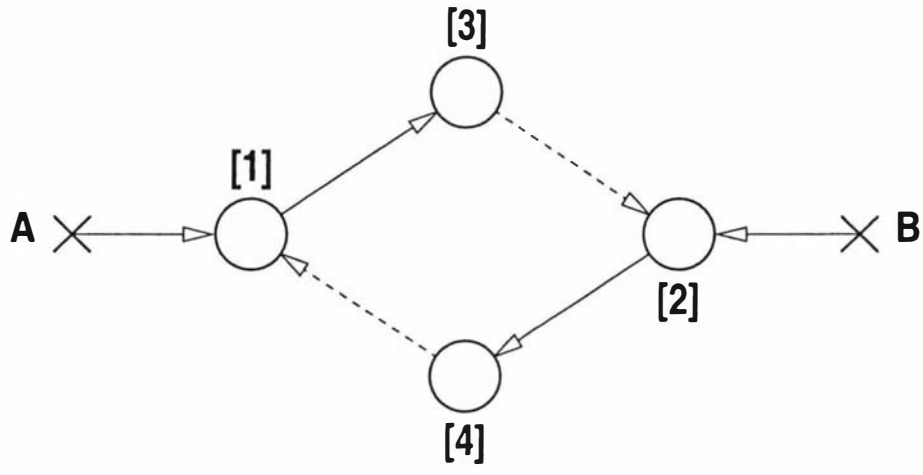


Figure 7.6: Possible deadlock scenario: $\mathcal{A}\rightarrow[1]\rightarrow[3]$ and $\mathcal{B}\rightarrow[2]\rightarrow[4]$.

Algorithm 7.5 procedure RESOLVE DEADLOCK FOUR CLUSTER

// Determine which players to unlock, resolving deadlock situation if
// necessary.

Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[4]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the prescribed ANY–ALL–PCTSP definition.
if *neither dominates* **then**
    // $\exists$ Deadlock. Break by replacement with:
    Determine if player $\mathcal{A}$ dominates [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[4]\rightarrow[1]$ or if player $\mathcal{B}$ dominates [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the ALL definition.
end
if *both dominate* **then**
    Unlock both player $\mathcal{A}$ and player $\mathcal{B}$.
else if *player $\mathcal{A}$ dominates* [1] **then**
    Unlock player $\mathcal{A}$ only.
else
    Unlock player $\mathcal{B}$ only.
end

end

- if *both* players are unlocked then, *neither* cluster $[cx_0]$ nor cluster $[cy_0]$ are direct-cluster-leads nor cluster-leads.

Now consider a cluster $[ci] \notin \{[cx_0], [cy_0]\}$. Cluster $[ci]$ is a candidate cluster-lead for player $\mathcal{X}$ if cluster $[ci]$ is available to player $\mathcal{X}$ (in the family-no-return sense) and either

- cluster $[ci]$ is not available to player $\mathcal{Y}$ (in the family-no-return sense); or
- cluster $[ci]$ is available to player $\mathcal{Y}$ (in the family-no-return sense) and player $\mathcal{Y}$ does not dominate cluster $[ci]$ according to the ANY–ALL-PCTSP requirement.

There are three cases to consider:

(i). Suppose cluster $[cy_0]$ is available to player $\mathcal{A}$ (in the family-no-return sense), cluster $[cx_0]$ is available to player $\mathcal{B}$ (in the family-no-return sense), and cluster $[ci]$ is a potential-cluster-lead for both players. To determine if cluster $[ci]$ is a direct-cluster-lead or a cluster-lead, we apply Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER.

- if *only* player $\mathcal{A}$ is unlocked, then cluster $[ci]$ is a direct-cluster-lead for player $\mathcal{A}$ and a cluster-lead for player $\mathcal{B}$.
- if *only* player $\mathcal{B}$ is unlocked, then cluster $[ci]$ is a direct-cluster-lead for player $\mathcal{B}$ and a cluster-lead for player $\mathcal{A}$.
- if *both* players are unlocked, then cluster $[ci]$ is a cluster-lead for both player $\mathcal{A}$ and player $\mathcal{B}$.

(ii). Suppose cluster $[ci]$ is a potential-cluster-lead only for player $\mathcal{A}$. Let τ_B be the earliest possible exit time for player $\mathcal{B}$ from cluster $[cy_0]$, given that player $\mathcal{B}$ must visit ALL prizes from cluster $[cy_0]$. Cluster $[ci]$ is a direct-cluster-lead for player $\mathcal{A}$ if player $\mathcal{A}$ can determine a path through ALL prizes from cluster $[cx_0]$ followed by ALL prizes from cluster $[ci]$ and exit from cluster $[ci]$ no later than time τ_B. We apply the ALL definition rather than the ANY–ALL–PCTSP definition for cluster $[ci]$ since we cannot *a priori* determine which prizes player $\mathcal{A}$ will eventually claim from cluster $[ci]$ and, hence, the most restrictive assumption is that player $\mathcal{A}$ must be able to claim all the prizes from cluster $[ci]$.

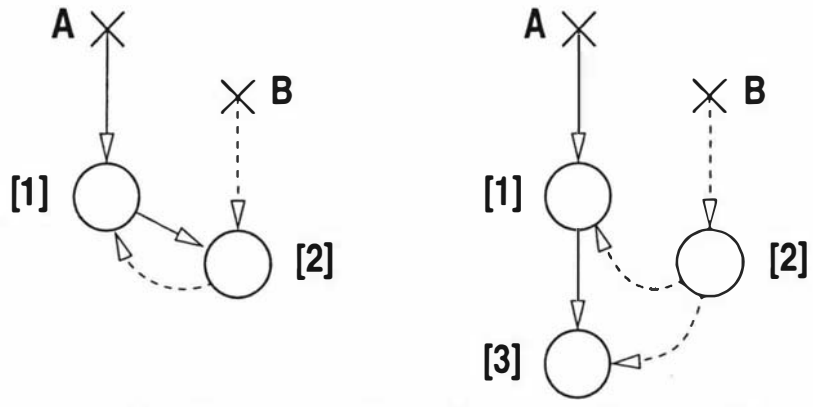
Similarly for player $\mathcal{B}$.

(iii). If cluster $[ci]$ is a potential-cluster-lead for player $\mathcal{X}$, and neither of cases (i)–(ii) apply, then cluster $[ci]$ is a cluster-lead for player $\mathcal{X}$.

7.3.2.5 Direct-cluster-lead Move-through

We use the notation "$\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$" to mean that player $\mathcal{A}$ moves through cluster $[cx_0]$ while committed to cluster $[cx_1]$ as the next look-through cluster.

Suppose $\mathcal{A} \rhd [1]$ and $\mathcal{B} \rhd [2]$. Figure 7.7 illustrates the two cases for player $\mathcal{A}$ moving through cluster [1] to a direct-cluster-lead. Cluster [2] is a direct-cluster-lead of player $\mathcal{A}$ in case (a) and cluster[3] is a direct-cluster-lead of player $\mathcal{A}$ in case (b).



(a) $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [1] \rhd [2]$ while $\mathcal{B} \rhd [2]$ (b) $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [1] \rhd [3]$ while $\mathcal{B} \rhd [2]$

Figure 7.7: Direct-cluster-lead move-through cases.

Case (a). Cluster [2] is a direct-cluster-lead of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[1]\}{\rightarrow}\{[2]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[2]\}{\rightarrow}\{[1]\}$ to determine the exit-prize, exit-time and exit-value from cluster [1] of player $\mathcal{A}$. Note there is no $\mathcal{B}$-final-family requirement.

Case (b). Cluster [3] is a direct-cluster-lead of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[1]\}{\rightarrow}\{[3]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[2]\}{\rightarrow}\{[1],[3]\}$ to determine the exit-prize, exit-time and exit-value from cluster [1] of player $\mathcal{A}$. Note there is no $\mathcal{B}$-final-family requirement.

7.3.2.6 Look-through: Move-through Cluster-lead

Suppose $\mathcal{A} \rhd [cx_0]$ and cluster $[cx_1]$ is a cluster-lead of player $\mathcal{A}$, but not a direct-cluster-lead. Also suppose $\mathcal{B} \rhd [cy_0]$ and cluster $[cy_1]$ is a cluster-lead of player $\mathcal{B}$, but not a direct-cluster-lead. Since $[cx_1]$ and $[cy_1]$ are not direct-cluster-leads, we have $[cx_0] \neq [cy_1]$ and $[cx_1] \neq [cy_0]$.

There are four general cases as illustrated in Figure 7.8. The solid arrows in the figure represent the proposed sequence of clusters for player $\mathcal{A}$, and the dashed arrows represent the proposed sequence of clusters for player $\mathcal{B}$, plus those "worst possible" continuations which assist in determining player $\mathcal{A}$'s subpath to move through cluster [1].

Case (i): $[cx_0] = [cy_0]$ *and* $[cx_1] = [cy_1]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}_{FF}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$, and the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

Case (ii): $[cx_0] = [cy_0]$ *and* $[cx_1] \neq [cy_1]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}_{FF}$ and $\mathcal{B}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cy_1]\}{\rightarrow}\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from

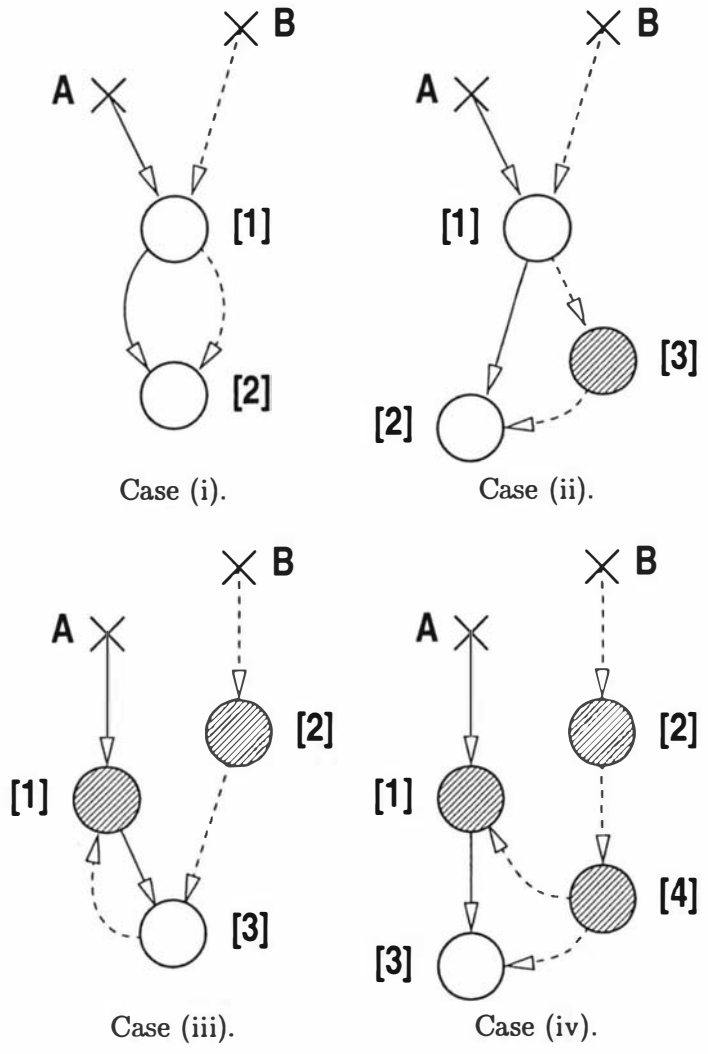


Figure 7.8: Cluster-lead move-through cases from player $\mathcal{A}$'s perspective.

cluster $[cx_0]$ of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_1]\}$ and $\mathcal{B}\rightarrow\{[cx_0]\}\rightarrow\{[cy_1]\}_{FF}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

Case (iii): $[cx_0] \neq [cy_0]$ *and* $[cx_1] = [cy_1]$

Since $[cx_1]$ is a cluster-lead for both players, and is *not* a direct-cluster-lead for either player, there can be no problem with *deadlock*.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cx_0]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_0]\}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cx_1]\}_{FF}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

Case (iv): $[cx_0] \neq [cy_0]$ *and* $[cx_1] \neq [cy_1]$

Theorem 7.3.5, below, shows that there can be no problem with *deadlock*.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cy_1]\}\rightarrow\{[cx_0],[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_0],[cy_1]\}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cy_1]\}_{FF}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

There is the potential that a problem with deadlock might arise in case (iv). Applying Algorithm 7.5 RESOLVE DEADLOCK FOUR CLUSTER would determine which players may move through their look-through cluster. The situation in which one player is committed to a sequence of two clusters is potentially awkward. However, the definitions of cluster-lead and direct-cluster-lead in Section 7.3.2.4 have been specifically crafted so that this cannot occur since, in such a situation, one of the cluster-leads would actually be a direct-cluster-lead and so the matchup never occurs. The argument is made more precise in the following lemma.

Theorem 7.3.5

Suppose $\mathcal{A} \triangleright [1]$, cluster [3] is a cluster-lead of player $\mathcal{A}$, $\mathcal{B} \triangleright [2]$ and cluster [4] is a cluster-lead of player $\mathcal{B}$, as in case (iv) of Figure 7.8. Then Algorithm 7.5 RESOLVE DEADLOCK FOUR CLUSTER would unlock *both* players and hence player $\mathcal{A}$ can move through cluster [1] and player $\mathcal{B}$ can move through cluster [2].

Proof:

We show that we can satisfy the conclusion of Algorithm 7.5 RESOLVE DEADLOCK FOUR CLUSTER by applying what we can deduce from applying Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER.

- Suppose player $\mathcal{A}$ does *not* dominate cluster [1] given that $\mathcal{B}\rightarrow[2]\rightarrow[4]\rightarrow[1]$ according to the prescribed ANY–ALL–PCTSP definition.
  - Since cluster [4] is *not* a direct-cluster-lead for player $\mathcal{B}$, Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER must unlock both players and hence player $\mathcal{B}$ does *not* dominate cluster [2] given that $\mathcal{A}\rightarrow[1]\rightarrow[4]\rightarrow[2]$ according to the prescribed ANY–ALL–PCTSP definition.
  - Hence, Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER would apply the deadlock breaking criterion. Again, since cluster [4] is *not* a direct-cluster-lead for player $\mathcal{B}$, Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER must unlock both players. Hence, player $\mathcal{A}$ *does* dominate cluster [1], given that $\mathcal{B}\rightarrow[2]\rightarrow[4]\rightarrow[1]$ according to the ALL definition.
- Similarly, if player $\mathcal{B}$ does *not* dominate cluster [2], given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the prescribed ANY–ALL–PCTSP definition then, by Algorithm 7.4 RESOLVE DEADLOCK THREE CLUSTER, player $\mathcal{B}$ *does* dominate cluster [2], given that $\mathcal{A}\rightarrow[1]\rightarrow[3]\rightarrow[2]$ according to the ALL definition.

Hence, both players would be *unlocked* by Algorithm 7.5 RESOLVE DEADLOCK FOUR CLUSTER, as required. ■

7.3.3 Strategic Subproblem: CLUSTER-GUARANTEE

We now design a generalization of the PRIZE-GUARANTEE method to clusters using the building blocks we have constructed thus far. Several initialisation scenarios are possible in which either player looks through a cluster which is a cluster-lead for that player. The resulting *strategic subproblem* — to determine a sequence of clusters, each of which is a cluster-lead from the previous scenario — is called CLUSTER-GUARANTEE.

A **cluster planning path**[4] is a sequence of cluster-targets. A **projected game position** consists of a pair of cluster planning paths, $\mathbb{P}_A$ for player $\mathcal{A}$ and $\mathbb{P}_B$ for player $\mathcal{B}$; a pair of **projected locations**, A and B; a **projected set of remaining clusters**, $\mathbb{Q}$; and a pair of **sets of projected arrival-times** at each prize, τ_{Aj} and τ_{Bj} $\forall j \in \mathbb{Q}$.

A **decision tree node** is defined by a projected game position plus, possibly, a lookthrough cluster for either or both players. The decision tree **root node** corresponds to the current position of the game with $P_A = \emptyset$, $P_B = \emptyset$ and $t_A = t_B = t_0$, where t_0 is the current time on the game clock.

7.3.3.1 Decision Tree Generation

We now define how to generate the CLUSTER-GUARANTEE decision tree by expanding a given decision tree node j to define the game position corresponding to each of its children. The expansion rules define transitions between five different types of decision tree node. At some

[4] We use $\mathbb{P}_A$ for a *cluster* planning path for player $\mathcal{A}$ and P_A for a *prize* planning path for player $\mathcal{A}$.

decision tree node j let $\mathcal{D}_A$ be the set of *direct cluster leads* and let $\mathcal{L}_A$ be the set of *cluster leads* of player $\mathcal{A}$.

Case (CP1). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$.

- $\forall [cx_1] \in \mathcal{D}_A \cup \mathcal{L}_A$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cx_1]$.

Case (CP2). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd \emptyset$.

- $\forall [cx_1] \in \mathcal{D}_A \cup \mathcal{L}_A$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ which involves moving player $\mathcal{A}$ through cluster $[cx_0]$, as in case (I) of Section 7.3.3.2 below. This is illustrated in Figure 7.9.

Case (CP3). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd [cy_0]$.

- If $[cy_0] \in \mathcal{D}_A \cup \mathcal{L}_A$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cy_0]$.
- $\forall [cx_1] \in (\mathcal{D}_A \cup \mathcal{L}_A) \setminus \{[cy_0]\}$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cx_1]$.

Case (CP4). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cx_0]$.

- $\forall [cx_1] \in \mathcal{D}_A \cup \mathcal{L}_A$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd \emptyset$, which involves moving both players through cluster $[cx_0]$, as in case (II) of Section 7.3.3.2 below.

Note that this is the *only* case of CLUSTER-GUARANTEE in which player $\mathcal{B}$ moves through a cluster. This is necessary since each cluster can be moved through at most once, either by a single player or jointly by both players.

Case (CP5). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$ where $[cx_0] \neq [cy_0]$.

- If $[cy_0] \in \mathcal{L}_A \cup \mathcal{D}_A$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_0]$.
- $\forall [cx_1] \in (\mathcal{D}_A \cup \mathcal{L}_A) \setminus \{[cy_0]\}$, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$.

7.3.3.2 Move-through Operations

There are two specialist CLUSTER-GUARANTEE move-through procedures that require definition.

(I). $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd \emptyset$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}_{FF}$ with $\mathcal{B} \rightarrow \{[cx_0], [cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

(II). $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd \emptyset$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}_{FF}$ with $\mathcal{B} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

We must also move player $\mathcal{B}$ through cluster $[cx_0]$. Since player $\mathcal{B}$ has no (secondary) target cluster, from the conservative perspective of player $\mathcal{A}$, we can consider each possible target cluster for player $\mathcal{B}$ in turn. For each $[cy_1] \in \mathbb{Q} \setminus \{[cx_0], [cx_1]\}$ apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B} \rightarrow \{[cx_0]\} \rightarrow \{[cy_1]\}_{FF}$ with

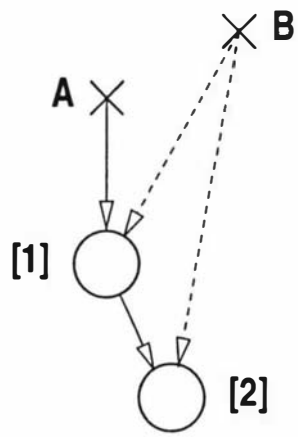


Figure 7.9: Specialist Case (I) Move-through for CLUSTER-GUARANTEE

$\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_1]\}$ to determine the exit-prize $j \in [cx_0]$ and exit-time χ_{Bj} from cluster $[cx_0]$ of player $\mathcal{B}$ and define $\tau_{Bk} = \chi_{Bj} + d_{jk}$ $\forall k \in [cy_1]$. Now let $\mathcal{B} \rhd \tau_B \ltimes \emptyset$, which means that, although player $\mathcal{B}$ has no commitment to look through a cluster, we use τ_{Bk} as the earliest possible entry time of player $\mathcal{B}$ at any remaining prize and t_B is no longer employed.

7.3.3.3 Evaluation of terminal decision node

If a decision node is terminal (i.e. has no children, since there are no cluster-leads for player $\mathcal{A}$ from that decision node) then, if $\mathcal{A} \rhd [cx_0]$, we need to estimate the remaining claim of player $\mathcal{A}$ on:

(I). $\mathcal{B} \rhd \emptyset$ or $\mathcal{B} \rhd [cx_0]$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}$ with $\mathcal{B}\rightarrow\{[cx_0]\}$ to determine the exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

(II). $\mathcal{B} \rhd [cy_0] \neq [cx_0]$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}$ with $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cx_0]\}$ to determine the exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

7.3.3.4 Searching the Decision Tree

To use a Branch and Bound (B&B) algorithm, we need to supply an upper bound on the possible value remaining. One possibility is the sum of the values of all prizes in the clusters in $\mathcal{D}_A \cup \mathcal{L}_A$ and also $[cx_1]$ if $\mathcal{A} \rhd [cx_1]$. Hence, we can search the decision tree by considering the best branch at each decision tree node. However, we may do better by stopping at that cluster, i.e., if $\mathcal{A} \rhd [cx_0]$, then we may also consider no further branching as the evaluation.

Suppose $\mathcal{A} \rhd [cx_0]$. If $\mathcal{B} \rhd \emptyset$ or $\mathcal{B} \rhd [cx_0]$, apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}$ with $\mathcal{B}\rightarrow\{[cx_0]\}$ to determine the exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

If $\mathcal{B} \rhd [cy_0] \neq [cx_0]$, apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}$ with $\mathcal{B}\rightarrow\{[cx_0]\}$ to determine the exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

7.3.3.5 Example Strategic Problem

The strategic example problem of Figure 7.10 is used throughout this chapter to illustrate the strategic analysis proposed by various strategic subproblems and strategic engines. The clusters are predefined for this problem and there is only one family.

▼ Analysis of $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$

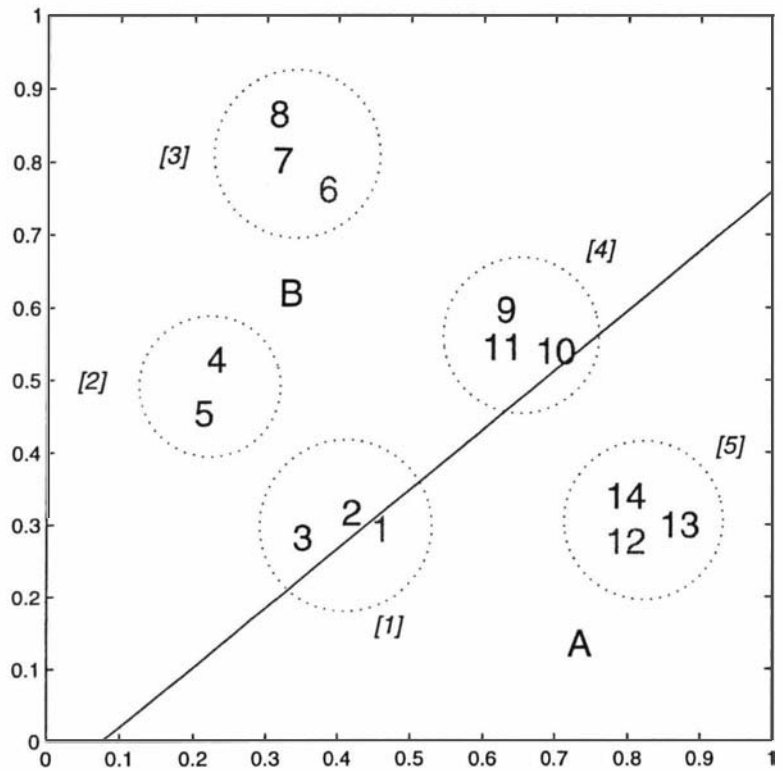
Table 7.4(a) shows that the lead clusters for player $\mathcal{A}$ are $\mathcal{L}_A = \{[1], [5]\}$ and Table 7.4(b) shows that the lead clusters for player $\mathcal{B}$ are $\mathcal{L}_B = \{[1], [2], [3], [4]\}$. Table 7.4(c) shows the maximal player $\mathcal{A}$ FAMILY-PRIZE-GUARANTEE subpath, $\mathcal{A}\rightarrow 14\rightarrow 12\rightarrow 12$, with prize total 179, and Table 7.4(d) shows the maximal player $\mathcal{B}$ FAMILY-PRIZE-GUARANTEE subpath, $\mathcal{A}\rightarrow 11\rightarrow 9\rightarrow 6\rightarrow 7\rightarrow 8$, with prize total 373.

Player $\mathcal{A}$'s CLUSTER-GUARANTEE Subpath

- Consider the scenario $\mathcal{A} \rhd [1]$. Table 7.5(a) shows a hyperbolic curve which are those points of equal earliest arrival time give that $\mathcal{A} \rhd 1$, i.e., player $\mathcal{A}$ must travel via prize 1. This shows that if $\mathcal{A} \rhd [1]$ then there are no following cluster leads for player $\mathcal{A}$. Thus, the maximal cluster guarantee path, given $\mathcal{A} \rhd [1]$, is $\mathcal{A}\rightarrow[1]$ as in Table 7.5(c).
- Consider the scenario $\mathcal{A} \rhd [5]$. Table 7.5(b) shows another equal earliest arrival time curve, given that player $\mathcal{A}$ must collect all the prizes from cluster [5]. However, the curve is piecewise hyperbolic (with breakpoints shown with a small '*') since the three sections of the curve correspond to the three possible exit prizes of player $\mathcal{A}$ from cluster [5]. Hence, if $\mathcal{A} \rhd [5]$, then there are no following cluster leads for player $\mathcal{A}$. Thus the maximal cluster guarantee path, given $\mathcal{A} \rhd [1]$, is $\mathcal{A}\rightarrow[5]$ as in Table 7.5(d).
- In summary, the maximal cluster guarantee path for player $\mathcal{A}$ is $\mathcal{A}\rightarrow[5]$, i.e., playerA$\rightarrow 14\rightarrow 12\dashrightarrow 13$, with prize total 199.

Player $\mathcal{B}$'s CLUSTER-GUARANTEE Subpath

- Consider the scenario $\mathcal{B} \rhd [1]$. Table 7.6(a) shows the equal earliest arrival curve, given that player $\mathcal{B}$ must claim at least one prize from cluster [1]. Note that $\mathcal{B}\rightarrow 2\rightarrow 3$ is guaranteed, but there is no other guaranteed subpath which includes both prize 2 and prize 3. Thus the maximal cluster guarantee path, given $\mathcal{B} \rhd [1]$, is $\mathcal{B}\rightarrow 2\rightarrow 3$, with prize total 243, as in Table 7.6(c).
- Consider the scenario $\mathcal{B} \rhd [2]$. Table 7.6(b) shows the equal earliest arrival curve, given that player $\mathcal{B}$ must claim both prizes from cluster [2]. This suggests that player $\mathcal{B}$ can subsequently target cluster [3]. Thus, the maximal cluster guarantee path, given $\mathcal{B} \rhd [2]$, is $\mathcal{B}\dashrightarrow 5\rightarrow 4\rightarrow 6\rightarrow 7\rightarrow 8$, with prize total 372, as in Table 7.6(d).



Prize	v	x	y
1	43	0.46	0.29
2	68	0.42	0.32
3	80	0.35	0.28
4	64	0.23	0.53
5	99	0.21	0.45
6	35	0.39	0.76
7	82	0.32	0.80
8	92	0.32	0.87
9	66	0.63	0.60
10	94	0.70	0.54
11	98	0.63	0.55
12	46	0.80	0.28
13	60	0.87	0.30
14	73	0.80	0.34
Total	1000		

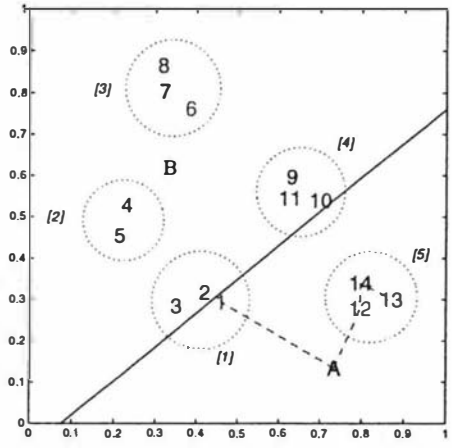
Player	x	y
$\mathcal{A}$	0.73	0.13
$\mathcal{B}$	0.33	0.62

Deadline	$\lambda = \infty$

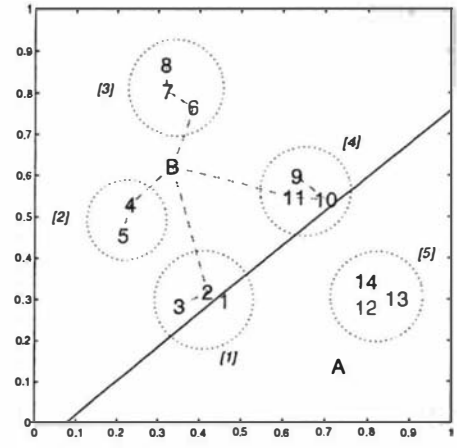
Cluster	Prizes
[1]	$\{1, 2, 3\}$
[2]	$\{4, 5\}$
[3]	$\{6, 7, 8\}$
[4]	$\{9, 10, 11\}$
[5]	$\{12, 13, 14\}$

Figure 7.10: Example Strategic Problem

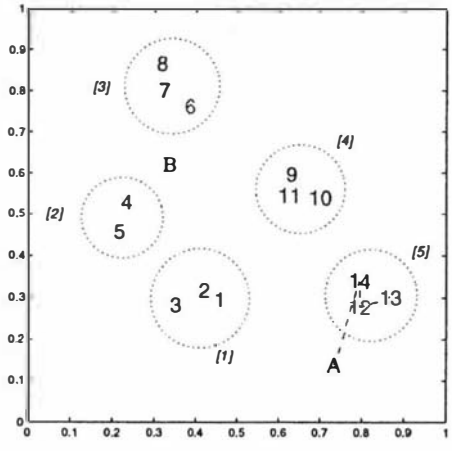
Table 7.4: Strategic Example of Player $\mathcal{A}$ CLUSTER-GUARANTEE



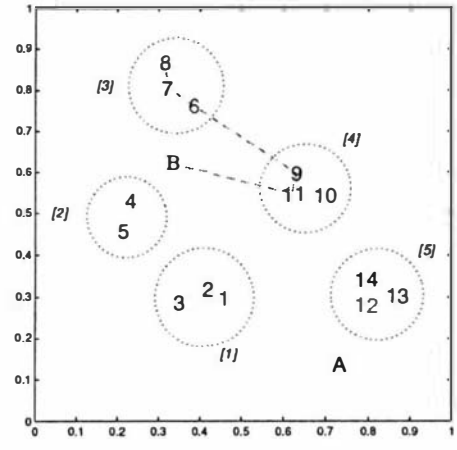
(a) $\mathcal{L}_A = \{[1], [5]\}$



(b) $\mathcal{L}_B = \{[1], [2], [3], [4]\}$

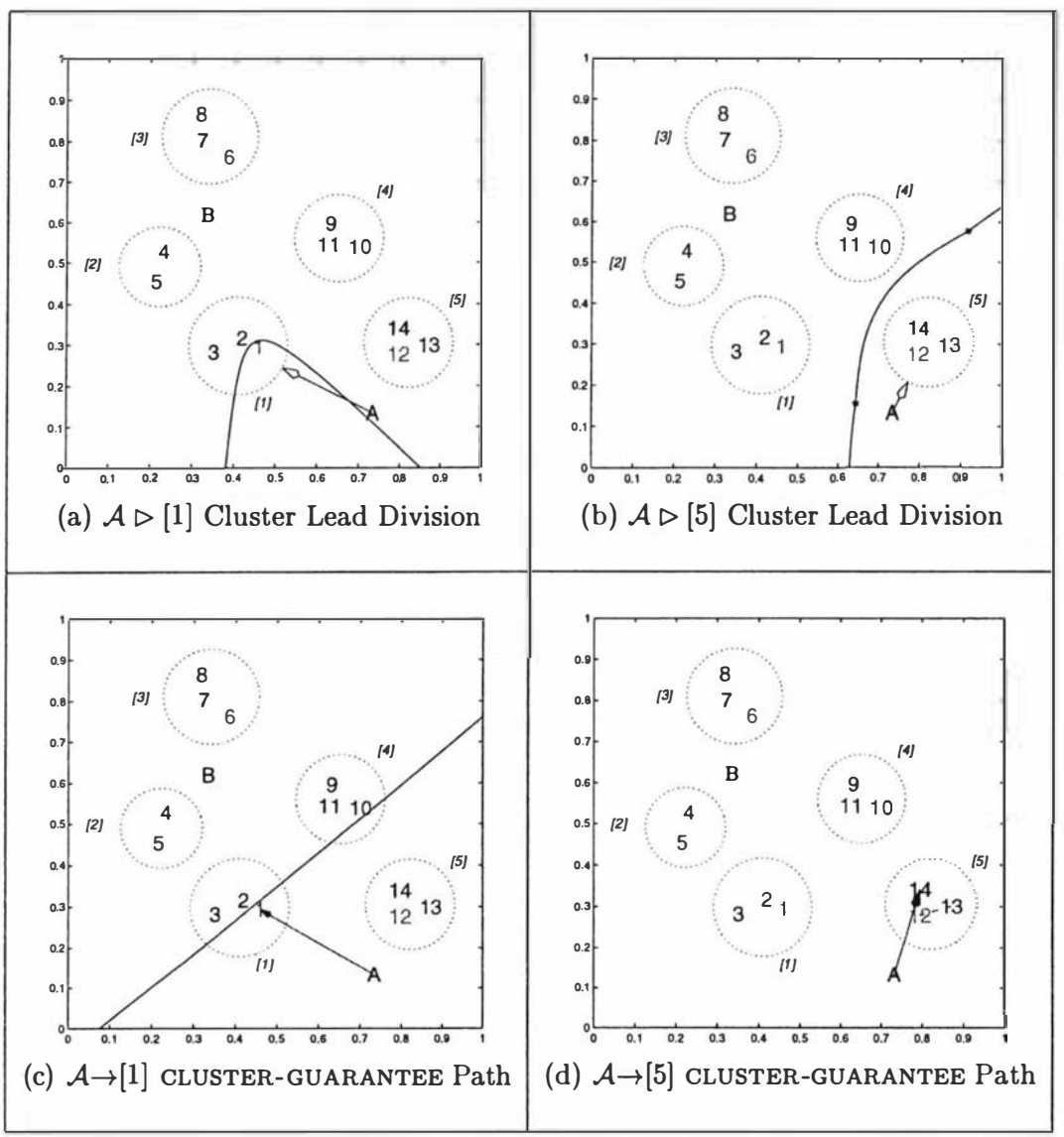


(c) Player $\mathcal{A}$ Maximal FAMILY-PRIZE-GUARANTEE Subpath



(d) Player $\mathcal{A}$ Maximal FAMILY-PRIZE-GUARANTEE Subpath

Table 7.5: Strategic Example of Player $\mathcal{A}$ CLUSTER-GUARANTEE



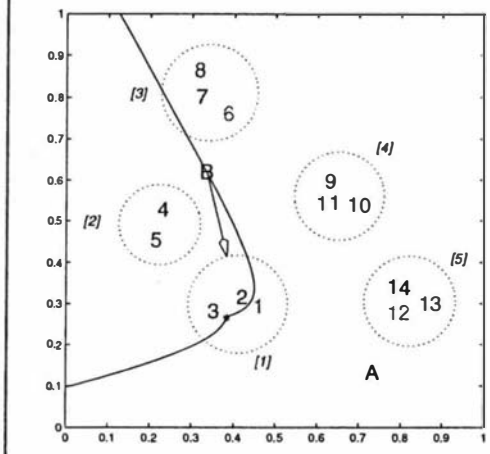
(a) $\mathcal{A} \triangleright$ [1] Cluster Lead Division

(b) $\mathcal{A} \triangleright$ [5] Cluster Lead Division

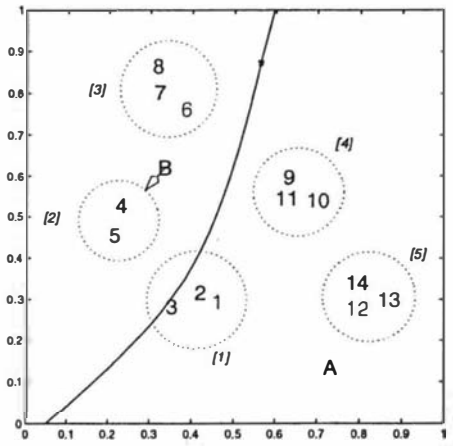
(c) $\mathcal{A} \rightarrow$[1] CLUSTER-GUARANTEE Path

(d) $\mathcal{A} \rightarrow$[5] CLUSTER-GUARANTEE Path

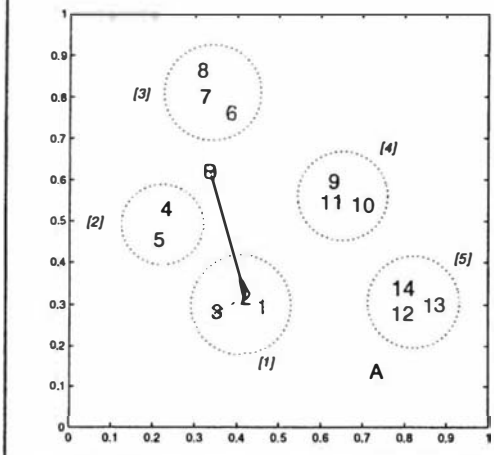
Table 7.6: Strategic Example of Player $\mathcal{B}$ CLUSTER-GUARANTEE



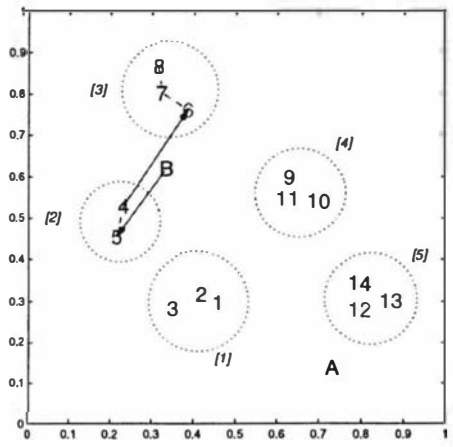
(a) $\mathcal{B} \triangleright$ [1] Cluster Lead Division



(b) $\mathcal{B} \triangleright$ [2] Cluster Lead Division

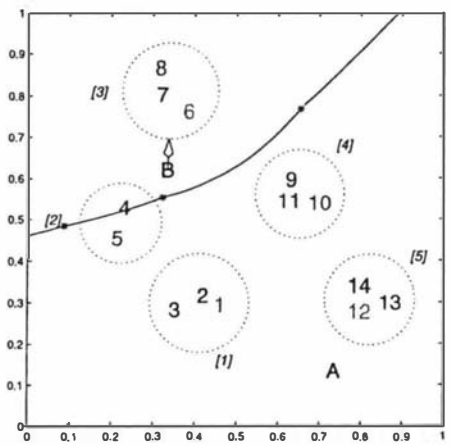


(c) $\mathcal{B} \rightarrow$ [1] CLUSTER-GUARANTEE Path

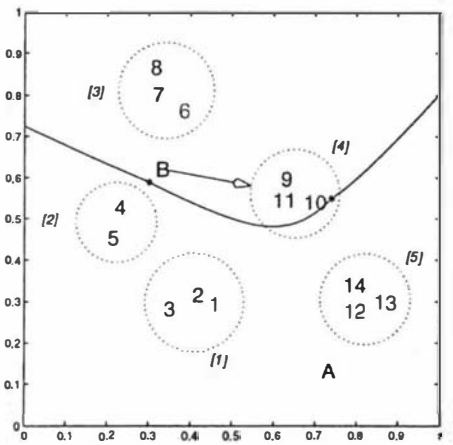


(d) $\mathcal{B} \rightarrow$ [2] $\rightarrow$ [3] CLUSTER-GUARANTEE Path

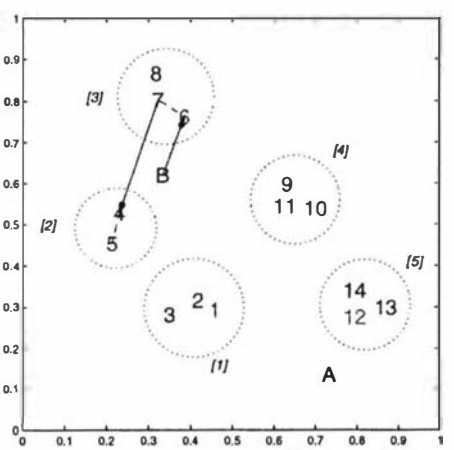
Table 7.7: Strategic Example of Player $\mathcal{B}$ CLUSTER-GUARANTEE



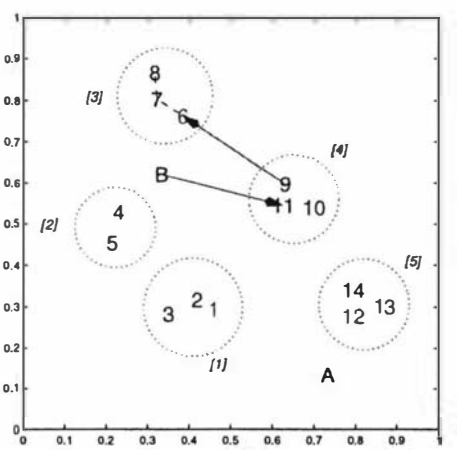
(a) $\mathcal{B} \triangleright$ [3] Cluster Lead Division



(b) $\mathcal{B} \triangleright$ [4] Cluster Lead Division



(c) $\mathcal{B} \rightarrow$[3] $\rightarrow$[2] CLUSTER-GUARANTEE Path



(d) $\mathcal{B} \rightarrow$[4] $\rightarrow$[3] CLUSTER-GUARANTEE Path

- Consider the scenario $\mathcal{B} \rhd [3]$ Table 7.7(a) shows the equal earliest arrival curve, given that player $\mathcal{B}$ must claim all the prizes from cluster [3]. This suggests that player $\mathcal{B}$ can subsequently target cluster [2]. Thus the maximal cluster guarantee path, given $\mathcal{B} \rhd [3]$, is $\mathcal{B}\rightarrow 6\rightarrow 7\rightarrow 4\rightarrow 5$, with prize total 280, as in Table 7.7(c).
- Consider the scenario $\mathcal{B} \rhd [4]$ Table 7.7(b) shows the equal earliest arrival curve, given that player $\mathcal{B}$ must claim all the prizes from cluster [4]. This suggests that player $\mathcal{B}$ can subsequently target cluster [3]. Thus the maximal cluster guarantee path, given $\mathcal{B} \rhd [4]$, is $\mathcal{B}\rightarrow 11\rightarrow 9\dashrightarrow 6\rightarrow 7\rightarrow 8$, with prize total 373, as in Table 7.7(d).
- In summary, the maximal cluster guarantee path for player $\mathcal{B}$ is $\mathcal{B}\dashrightarrow[4]\rightarrow[3]$, i.e., $\mathcal{B}\rightarrow 11\rightarrow 9\rightarrow 6\rightarrow 7\dashrightarrow 8$, with prize total 373.

▼ Analysis of $\mathcal{A} \rhd [1]$ and $\mathcal{B} \rhd [1]$

Consider the scenario $\mathcal{A} \rhd [1]$ and $\mathcal{B} \rhd [1]$. Table 7.8(a) shows the equal earliest arrival curve, given that *both players* must claim at least one prize from cluster [1]. Table 7.8(c) shows the maximal cluster guarantee paths for each player. The possible cluster guarantee paths for player $\mathcal{A}$ are $\mathcal{A}\rightarrow[1]\rightarrow[5]$, of value 176, and $\mathcal{A}\rightarrow[1]\rightarrow[4]$, of value 207. The possible cluster guarantee paths for player $\mathcal{B}$ are $\mathcal{B}\rightarrow[1]\rightarrow[3]$, of value 277, and $\mathcal{B}\rightarrow[1]\rightarrow[2]\rightarrow[3]$, of value 417.

▼ Analysis of $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [3]$

Consider the scenario $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [3]$. Table 7.8(b) shows the equal earliest arrival curve, given that player $\mathcal{A}$ must claim all the prizes from cluster [5] and player $\mathcal{B}$ must claim all the prizes from cluster [5]. Table 7.8(d) shows all the posible cluster guarantee paths for each player. The possible cluster guarantee paths for player $\mathcal{A}$ are $\mathcal{A}\rightarrow[5]\rightarrow[1]$, of value 237, and $\mathcal{A}\dashrightarrow[5]\rightarrow[4]$, of value 227. The possible cluster guarantee paths for player $\mathcal{B}$ are $\mathcal{B}\dashrightarrow[3]\rightarrow[2]$, of value 372, and $\mathcal{B}\dashrightarrow[3]\rightarrow[4]$, of value 101.

▼ Discussion

We have provided an example for each of the possible initialisation scenarios of CLUSTER-GUARANTEE. These will be employed extensively in illustrating the strategic scenarios of CLUSTER-PARANOID (Section 7.4.2) and CLUSTER-DT (Section 7.5.4).

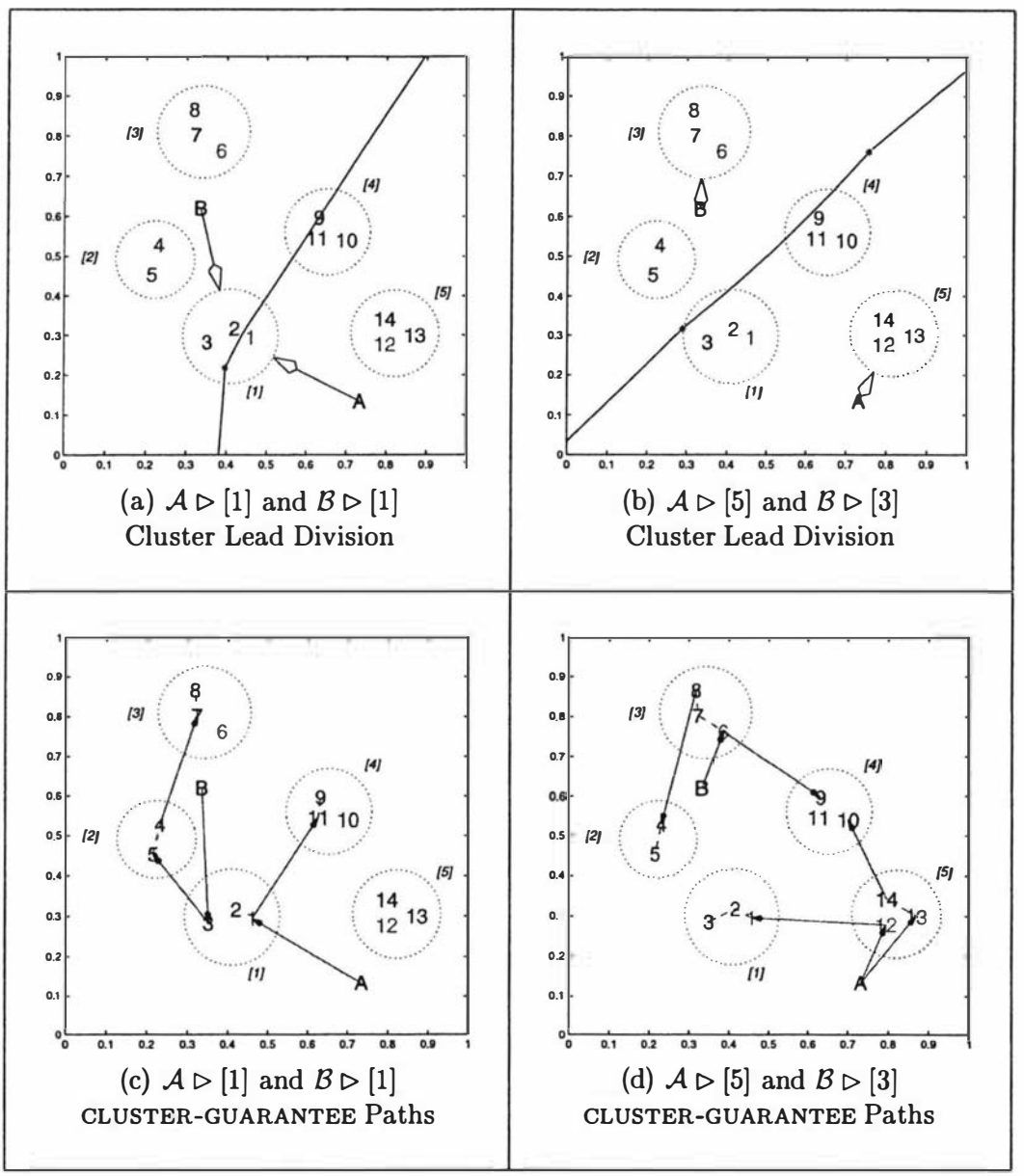
7.3.4 Two Cluster Problem

We now strategically analyse the **Two Cluster Problem**, which is analogous to the *two prize problem* of Section 5.1, using the tools of cluster-lead, look-through, move-through and CLUSTER-GUARANTEE developed in the previous sections.

7.3.4.1 One Cluster Problem

Once one cluster has been moved through from a two cluster problem, we have only one cluster remaining, cluster $[ci]$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[ci]\}$ with

Table 7.8: Strategic Example of CLUSTER-GUARANTEE



(a) $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [1]$ Cluster Lead Division

(b) $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [3]$ Cluster Lead Division

(c) $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [1]$ CLUSTER-GUARANTEE Paths

(d) $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [3]$ CLUSTER-GUARANTEE Paths

$\mathcal{B}\rightarrow\{[ci]\}$ to determine the exit-value from cluster $[ci]$ of player $\mathcal{A}$. This exit-value is the player $\mathcal{A}$ evaluation of the one cluster problem.

7.3.4.2 Additional Move-through Cases

We will require three additional specialist move-through cases as follows. These arise because player $\mathcal{B}$ has no further possible clusters to target but either both players or only player $\mathcal{A}$ must move through a cluster.

■ $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [ci_1]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[ci_1]\}\rightarrow\{[ci_2]\}_{FF}$ with $\mathcal{B}\rightarrow\{[ci_1]\}\rightarrow\{[ci_2]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[ci_1]$ of player $\mathcal{A}$. Also apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{[ci_1]\}$ with $\mathcal{A}\rightarrow\{[ci_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[ci_1]$ of player $\mathcal{B}$.

■ $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [ci_2]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[ci_1]\}$ with $\mathcal{B}\rightarrow\{[ci_2]\}\rightarrow\{[ci_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[ci_1]$ of player $\mathcal{A}$. Also apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{[ci_2]\}$ with $\mathcal{A}\rightarrow\{[ci_1]\}\rightarrow\{[ci_2]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[ci_2]$ of player $\mathcal{B}$.

■ $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd [ci_2]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[ci_1]\}\rightarrow\{[ci_2]\}_{FF}$ with $\mathcal{B}\rightarrow\{[ci_2]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[ci_1]$ of player $\mathcal{A}$.

7.3.4.3 Evaluation of Scenarios

Determine the cluster-leads of each player with no look-through.

- Suppose cluster $[ci_1]$ is a cluster-lead of both players. We must define an evaluation of the scenario $\mathcal{A} \rhd [ci_1]$ and $\mathcal{B} \rhd [ci_1]$. Determine if cluster $[ci_2]$ is a cluster-lead for one or both players given both players look through cluster $[ci_1]$.
  - If cluster $[ci_2]$ is a cluster-lead for both players, then move both players through cluster $[ci_1]$ according $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [ci_1] \rhd [ci_2]$ (case (i) of Section 7.3.2.6).
  - If cluster $[ci_2]$ is a cluster-lead for player $\mathcal{A}$ only, then move both players through cluster $[ci_1]$ according to $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [ci_1]$.
  - If cluster $[ci_2]$ is a cluster-lead for player $\mathcal{B}$ only, then move both players through cluster $[ci_1]$ according to $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [ci_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [ci_1] \rhd [ci_2]$.

  The evaluation is the exit-value of player $\mathcal{A}$ from cluster $[ci_1]$ plus the player $\mathcal{A}$ evaluation of the remaining *one cluster problem* on $[ci_2]$.

- Suppose cluster $[ci_1]$ is a cluster-lead of player $\mathcal{A}$ and cluster $[ci_2]$ is a cluster-lead of player $\mathcal{B}$. We must define an evaluation of the scenario $\mathcal{A} \rhd [ci_1]$ and $\mathcal{B} \rhd [ci_2]$. Apply Algorithm 7.3 RESOLVE DEADLOCK TWO CLUSTER to determine if cluster $[ci_2]$ is a direct-cluster-lead of player $\mathcal{A}$, or cluster $[ci_1]$ is a direct-cluster-lead of player $\mathcal{B}$, or neither.
  - If only player $\mathcal{A}$ is unlocked, then cluster $[ci_2]$ is a direct-cluster-lead for player $\mathcal{A}$, so move player $\mathcal{A}$ through cluster $[ci_1]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd [ci_2]$. The evaluation is the exit-value of player $\mathcal{A}$ from cluster $[ci_1]$ plus the player $\mathcal{A}$ evaluation of the remaining *one cluster problem* on $[ci_2]$.
  - If only player $\mathcal{B}$ is unlocked, then cluster $[ci_1]$ is a direct-cluster-lead for player $\mathcal{B}$, so move player $\mathcal{B}$ through cluster $[ci_2]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [ci_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [ci_2] \rhd [ci_1]$. The evaluation is the player $\mathcal{A}$ evaluation of the remaining *one cluster problem* on cluster $[ci_1]$.
  - If both players are unlocked, then move player $\mathcal{A}$ through cluster $[ci_1]$ and move player $\mathcal{B}$ through cluster $[ci_2]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [ci_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [ci_2]$. The evaluation is the exit-value of player $\mathcal{A}$ from cluster $[ci_1]$.
- Suppose both clusters are cluster-leads for player $\mathcal{A}$, but neither cluster is a cluster-lead for player $\mathcal{B}$. We must define an evaluation of the scenario $\mathcal{A} \rhd [ci_1]$ and $\mathcal{B} \rhd \{[ci_1], [ci_2]\}$. Move player $\mathcal{A}$ through cluster $[ci_1]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [ci_1] \rhd [ci_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd [ci_2]$. The evaluation is the exit-value of player $\mathcal{A}$ from cluster $[ci_1]$ plus the player $\mathcal{A}$ evaluation of the remaining *one cluster problem* on $[ci_2]$.
- Suppose both clusters are cluster-leads for player $\mathcal{B}$, but neither cluster is a cluster-lead for player $\mathcal{A}$. We must define an evaluation of the scenario $\mathcal{A} \rhd [ci_1]$ and $\mathcal{B} \rhd \{[ci_1], [ci_2]\}$. Move player $\mathcal{B}$ through cluster $[ci_2]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [ci_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [ci_2] \rhd [ci_1]$. The evaluation is the player $\mathcal{A}$ evaluation of the remaining *one cluster problem* on $[ci_1]$.

We can now evaluate each of the targeting scenarios. It remains to determine which target to select.

7.3.4.4 Strategic Analysis for Two Cluster Problem

We can now design a strategy for player $\mathcal{A}$ and a corresponding player $\mathcal{A}$ evaluation. The following cases generalize those for the two prize problem and the following strategies proposed for each case parallel those optimal or critical strategies for each case of the two prize problem.

Case (i). *Both clusters are cluster-leads for both players.*

	$\mathcal{B} \rhd [ci_1]$	$\mathcal{B} \rhd [ci_2]$
$\mathcal{A} \rhd [ci_1]$	□	□
$\mathcal{A} \rhd [ci_2]$	□	□

Player $\mathcal{A}$ plays a MAXIMIN strategy and the evaluation is the MAXIMIN value of the table above.

Case (ii). *Cluster* $[ci_1]$ *is a cluster-lead for player* $\mathcal{A}$ *only and cluster* $[ci_2]$ *is a cluster-lead for both players.*

	$\mathcal{B} \rhd [ci_2]$	$\mathcal{B} \rhd [ci_1]$
$\mathcal{A} \rhd [ci_1]$	□	∞
$\mathcal{A} \rhd [ci_2]$	□	□

Player $\mathcal{A}$ plays a MINIMAX strategy and the evaluation is the MINIMAX value of the table above.

Case (iii). *Cluster* $[ci_1]$ *is a cluster-lead for player* $\mathcal{B}$ *only and cluster* $[ci_2]$ *is a cluster-lead for both players.*

	$\mathcal{B} \rhd [ci_1]$	$\mathcal{B} \rhd [ci_2]$
$\mathcal{A} \rhd [ci_2]$	□	□
$\mathcal{A} \rhd [ci_1]$	0	□

Player $\mathcal{A}$ targets cluster $[ci_2]$ and the evaluation is MAXIMIN value of the table above.

Case (iv). *Cluster* $[ci_1]$ *is a cluster-lead for player* $\mathcal{A}$ *only and cluster* $[ci_2]$ *is a cluster-lead for player* $\mathcal{B}$ *only.*

Player $\mathcal{A}$ targets cluster $[ci_1]$ and the evaluation is the evaluation of $\mathcal{A} \rhd [ci_1]$ and $\mathcal{B} \rhd [ci_2]$.

Case (v). *Both clusters are cluster-leads for player* $\mathcal{A}$ *only.*

	$\mathcal{B} \rhd \{[ci_1], [ci_2]\}$
$\mathcal{A} \rhd [ci_1]$	□
$\mathcal{A} \rhd [ci_2]$	□

Player $\mathcal{A}$ plays a MAXIMIN strategy on the table above and the evaluation is the maximum value of the table above.

Case (vi). *Both clusters are cluster-leads for player* $\mathcal{B}$ *only.*

	$\mathcal{B} \rhd [ci_1]$	$\mathcal{B} \rhd [ci_2]$
$\mathcal{A} \rhd \{[ci_1], [ci_2]\}$	□	□

Player $\mathcal{A}$ plays a TWO CLUSTER MEDIAN strategy, the details of which are presented in Appendix 7.A.3. The evaluation is the minimum value of the table above.

This completes the design of a strategic solution to the two cluster problem. In the same way that the two prize problem is embedded into PRIZE-DT, these strategic concepts for two clusters will be embedded into CLUSTER-DT.

7.3.5 Cluster Feasible Windows

Recall that for PRIZE-DT we considered the window scenario $\mathcal{A} \rhd x$ and $\mathcal{B} \rhd \{y_1, y_2\}$ as a fundamental building block. We now generalize this to clusters. Initially we consider the cluster window scenario $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$ and then motivate several variations arising from look-through. This section is closely linked with two appendices. Appendix 7.A.4 makes precise definitions for cluster-window-feasible in each of the cases considered and Appendix 7.A.5 makes precise definitions for moving players through clusters with respect to a cluster window scenario.

7.3.5.1 The Basic Case

We firstly look at the basic case in which player $\mathcal{A}$ targets a cluster-lead (not a direct-cluster-lead) and player $\mathcal{B}$ targets a cluster-pair.

■ $\mathcal{A} \rhd [cx_1]$ **and** $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$

Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$.

Cluster-lead. Is cluster $[cy] \in \{[cy_1], [cy_2]\}$ a cluster-lead for player $\mathcal{B}$ conditional upon $\mathcal{A} \rhd [cx_1]$? Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B} \rightarrow \{[cy]\}$ with $\mathcal{A} \rightarrow \{[cx_1]\} \rightarrow \{[cy]\}$ to determine if player $\mathcal{B}$ has a non-zero evaluation. Note that there is no potential problem with deadlock since $[cx_1]$ is a cluster-lead and any deadlock situation would imply that $[cx_1]$ should have been a direct-cluster-lead of player $\mathcal{A}$ instead.

Cluster Window Feasibility. Suppose player $\mathcal{A}$ dominates both clusters $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd [cx_1]$. Consider the following contingent strategy of player $\mathcal{A}$. Player $\mathcal{A}$ moves through cluster $[cx_1]$ and then determines which of either cluster $[cy_1]$ or cluster $[cy_2]$ to target next, contingent upon the observed location of player $\mathcal{B}$ at that time. It is possible that player $\mathcal{A}$ then dominates cluster $[cy_2]$ via $\mathcal{A} \rhd [cy_1]$ or dominates cluster $[cy_1]$ via $\mathcal{A} \rhd [cy_2]$. The scenario $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$ is cluster-window-feasible if such a contingent strategy for player $\mathcal{A}$ cannot exist. That is, once player $\mathcal{A}$ has claimed prizes from cluster $[cx_1]$, player $\mathcal{B}$ will be in a position to claim some prizes from either cluster $[cy_1]$ (if player $\mathcal{A}$ then targets cluster $[cy_2]$) or cluster $[cy_2]$ (if player $\mathcal{A}$ then targets cluster $[cy_1]$).

Suppose player $\mathcal{A}$ exits from cluster $[cx_1]$ by exit-prize $k \in [cx_1]$ at some time $\chi_{\mathcal{A}k}$. Let $i_1 \in [cy_1]$ be an entry-prize to cluster $[cy_1]$ and $i_2 \in [cy_2]$ be an entry-prize to cluster $[cy_2]$. Let $W(i_1, i_2, k)$ be the prize window scenario $\mathcal{A} \rhd k$ and $\mathcal{B} \rhd \{i_1, i_2\}$ where player $\mathcal{A}$ arrives at prize k at time $\chi_{\mathcal{A}k}$ and the distance between i_1 and i_2 must satisfy the ANY-ALL-PCTSP requirement. Since player $\mathcal{B}$ cannot predict the exit-prize k of player $\mathcal{A}$ from cluster $[cx_1]$, we must determine the existence of i_1 and i_2 such that $W(i_1, i_2, k)$ is window feasible for every possible exit-prize $k \in [cx_1]$.

We employ the prize window feasibility concepts since they capture the ideas required and are simple to apply using our existing procedures. Also, while the exit-prize $k \in [cx_1]$ is unknown, we can apply a COMPROMISE WINDOW as in SMALLDMS-STEP-MONITOR of Section 6.4.4 to determine how to initially navigate a cluster window when required.

Definition 7.A.2—case (i) of Appendix 7.A.4—makes these ideas more precise. We use the symbol $\mathbb{W}$ to stand for a cluster window scenario *and* for the implied cluster window feasibility constraints.

Move-through. Recall that, to move a player through a cluster, that player must have a secondary target cluster. Hence there is no associated move-through operation in this case.

7.3.5.2 Initial Player $\mathcal{B}$ Look-Through Cases

We consider two initial cases in which player $\mathcal{B}$ looks through a cluster.

■ $\mathcal{A} \rhd [cx_1]$ **and** $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\}$

Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$, conditional on $\mathcal{B} \rhd [cy_0]$.

Cluster Window Feasibility. Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd [cx_1]$, conditional on $\mathcal{B} \rhd [cy_0]$. This is essentially the same as the previous case considered, except that player $\mathcal{B}$ now initially looks through cluster $[cy_0]$ and so must carefully select an exit-prize from cluster $[cy_0]$. Definition 7.A.3 (page 321) of Appendix 7.A.4 (case (ii)) precisely defines whether this cluster window scenario is cluster-window-feasible.

Move-through. There is no associated move-through operation since neither player has a secondary target cluster.

■ $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ **and** $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$

Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd [cx_0]$, conditional on $\mathcal{B} \rhd [cx_0]$.

Cluster-lead. Is cluster $[cy] \in \{[cy_1], [cy_2]\}$ a cluster-lead for player $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd [cx_0]$? Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B} \rightarrow \{[cx_0]\} \rightarrow \{[cy]\}_{FF}$ with $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\} \rightarrow \{[cy]\}$ to determine if player $\mathcal{B}$ has a non-zero evaluation.

Cluster Window Feasibility. Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ conditional on $\mathcal{B} \rhd [cx_0]$. Player $\mathcal{B}$ must carefully select an exit-prize from cluster $[cx_0]$. Case (iii) of Appendix 7.A.4 precisely defines whether this cluster window scenario is cluster-window-feasible.

Move-through. We wish to move player $\mathcal{A}$ through cluster $[cx_0]$ but, since both players look through cluster $[cx_0]$, we must move both players through, even though player $\mathcal{B}$ has not chosen a secondary target cluster. Let W_1 be the cluster window feasible scenario $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$. The move-through operation is:

- $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ — case (x) of Appendix 7.A.5.

The move-through operation must ensure that the resulting scenario is also cluster window feasible.

7.3.5.3 Chains of Cluster Window Feasible Scenarios

We now look at two cases which involve a chain of cluster window feasible scenarios.

■ $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ **and** $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

Let $\mathbb{W}_1$ be a previous cluster window feasible scenario such that $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$. Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd [cx_0]$, conditional on $\mathcal{B} \rhd \mathbb{W}_1$.

Cluster-lead. Is cluster $[cy] \in \{[cy_1], [cy_2]\}$ a cluster-lead for player $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd \mathbb{W}_1$? Since $\mathcal{B} \rhd \mathbb{W}_1$, associate with $\mathbb{W}_1$ an earliest arrival-time, τ_{Bi}, at each prize $i \in [cy_1] \cup [cy_2]$, such that player $\mathcal{B}$ moves with respect to the constraints imposed by $\mathbb{W}_1$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B} \rightarrow \{\tau_B \ltimes [cy]\}$ with $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\} \rightarrow \{[cy]\}$ to determine if player $\mathcal{B}$ has a non-zero evaluation. (Recall that the notation "$\tau_B \ltimes [cy]$" means that player $\mathcal{B}$ arrives at each prize $i \in [cy]$ at time τ_{Bi}.)

Cluster Window Feasibility. Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$, conditional on $\mathcal{B} \rhd \mathbb{W}_1$. Let $\mathbb{W}_2$ be the cluster window scenario $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$. Loosely, $\mathbb{W}_2$ is cluster-window-feasible if player $\mathcal{B}$ can move such that the constraints imposed by $\mathbb{W}_1$ are satisfied and, once player $\mathcal{A}$ has claimed prizes from cluster $[cx_1]$, player $\mathcal{B}$ will be in a position to claim some prizes from either cluster $[cy_1]$ (if player $\mathcal{A}$ then targets cluster $[cy_2]$) or cluster $[cy_2]$ (if player $\mathcal{A}$ then targets cluster $[cy_1]$). Since player $\mathcal{A}$ is essentially making a detour to target cluster $[cx_1]$ and player $\mathcal{B}$ is already able to satisfy the constraints imposed by $\mathbb{W}_1$, then $\mathbb{W}_2$ should also be cluster-window-feasible.

Definition 7.A.4 (page 324) of Appendix 7.A.4 (case (v)) precisely defines whether $\mathbb{W}_2$ is cluster-window-feasible and Lemma 7.A.5 confirms that $\mathbb{W}_2$ is cluster-window-feasible. Furthermore, there are no tighter constraints imposed by $\mathbb{W}_2$ on the earliest arrival-time of player $\mathcal{B}$ at each prize $i \in [cy_1] \cup [cy_2]$. Hence $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \mathbb{W}_2 \rightarrow \{[cy_1], [cy_2]\})$ is equivalent to $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$. In general, the constraints imposed by a chain of cluster window scenarios are equivalent to the constraints imposed by the first window in the chain.

Move-through. In the related move-through operation, player $\mathcal{A}$ moves through cluster $[cx_0]$.

- $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\}$ — case (iii) of Appendix 7.A.5.

■ $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ **and** $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

Let $\mathbb{W}_1$ be a previous cluster window feasible scenario such that $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$. Suppose player $\mathcal{A}$ dominates both $[cy_1]$ and $[cy_2]$ via $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$, conditional on $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1)$.

Static cluster window distance correction. Let χ_{Bj} be the earliest possible exit-time from prize $j \in [cy_0]$ of player $\mathcal{B}$ from cluster $[cy_0]$ satisfying the ANY-ALL-PCTSP requirement on cluster $[cy_0]$. Player $\mathcal{B}$ must move from prize $j \in [cy_0]$ to a prize $i \in [cy_1] \cup [cy_2]$ with respect to the constraints imposed by $\mathbb{W}_1$. Choose an exit-prize $j \in [cy_0]$ of player $\mathcal{B}$ from cluster $[cy_0]$ and an entry-prize $i_1 \in [cy_1]$ of player $\mathcal{B}$ to cluster $[cy_1]$. We wish to calculate the shortest possible distance from j to i with respect to $\mathbb{W}_1$, given player $\mathcal{B}$'s exit-time χ_{Bj} from prize j. Suppose player $\mathcal{A}$ were to exit cluster $[cx_1]$ from exit-prize $k \in [cx_1]$ at exit-time χ_{Ak} and let $W(j, i_1, i_2, k)$ be the associated prize window scenario $\mathcal{A} \triangleright k$ and $\mathcal{B} \triangleright (j \rightarrow \{i_1, i_2\})$. Let $I_{j,i_1} \subseteq [cy_2]$ be the set of prizes $i_2 \in [cy_2]$ such that $\forall k \in [cx_1]$ $\exists$ a feasibility window $W(j, i_1, i_2, k)$. If $I_{j,i_1} = \emptyset$ then let $\delta_{ji_1} = \infty$ otherwise let

$$\delta_{ji_1} = \min_{i_2 \in I_{j,i_1}} \max_{k \in [cx_1]} \left\{ d_{j,W(j,i_1,i_2,k)} + d_{W(j,i_1,i_2,k),i_1} \right\}.$$

This is called the **static cluster window distance correction**.

Cluster-lead. Is cluster $[cy_1]$ a cluster-lead for player $\mathcal{B}$ conditional upon $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright ([cy_0] \rightarrow \mathbb{W}_1)$? Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B} \rightarrow \{[cy_0]\}_{\mathbb{W}_1} \rightarrow \{[cy_2]\}_{FF}$ with $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\} \rightarrow \{[cy_2]\}$. Note that the ANY-ALL-PCTSP requirement is inherited to hold for player $\mathcal{B}$ on cluster $[cy_0]$ and $\mathcal{A} \nrightarrow [cy_0]$.

The notation "$\{[cy_0]\}_{\mathbb{W}_1} \rightarrow \{[cy_2]\}$" means that we apply the *static cluster window distance correction* by temporarily assigning

$$d_{ji_2} \leftarrow \delta_{ji_2} \; \forall \, j \in [cy_0], i_2 \in [[cy_2].$$

We can similarly determine if cluster $[cy_2]$ is a cluster-lead for player $\mathcal{B}$ conditional upon $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright ([cy_0] \rightarrow \mathbb{W}_1)$.

Cluster Window Feasibility. This is essentially the same as the previous case considered, except that player $\mathcal{B}$ now initially looks through cluster $[cy_0]$ and so must carefully select an exit-prize from cluster $[cy_0]$. Definition 7.A.6 (page 326) of Appendix 7.A.4 (case (vi)) precisely defines whether this cluster window scenario is cluster-window-feasible and Lemma 7.A.7 confirms that it is. Hence $\mathcal{B} \triangleright ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \mathbb{W}_2 \rightarrow \{[cy_1], [cy_2]\})$ is equivalent to $\mathcal{B} \triangleright ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$. In general, the constraints imposed by a chain of cluster window scenarios following $\mathcal{B} \triangleright [cy_0]$ are equivalent to the constraints imposed by the first window in the chain.

Move-through. In the related move-through operation, player $\mathcal{A}$ moves through cluster $[cx_0]$.

- $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \triangleright [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \triangleright ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\}$ — case (iv) of Appendix 7.A.5.

7.3.5.4 Look-through Previous Cluster Feasible Window

Let $\mathbb{W}_1$ be the cluster window feasible scenario $\mathcal{A} \triangleright [cx_1]$ and $\mathcal{B} \triangleright ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$. Suppose that player $\mathcal{A}$ targets cluster $[cy_2]$. We move player $\mathcal{A}$ through $[cx_1]$ and player $\mathcal{B}$ through cluster $[cy_0]$ according to $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_1] \triangleright [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \triangleright (\mathbb{W}_1 \rightarrow [cy_1])$—case (vi) of Appendix 7.A.5. The resulting scenario is $\mathcal{A} \triangleright [cy_2]$ and $\mathcal{B} \triangleright (\mathbb{W}_1 \rightarrow [cy_1])$, i.e., player $\mathcal{B}$ looks

through a cluster, but must satisfy the constraints imposed by $\mathbb{W}_1$ to arrive at that cluster. We now know the exit-prize $j \in [cy_0]$ and exit-time χ_{Bj} of player $\mathcal{B}$ from cluster $[cy_0]$. Hence, we can calculate the earliest possible entry-time τ_{Bi_2} of player $\mathcal{B}$ at each prize $i_2 \in [cy_2]$ satisfying the constraints imposed by $\mathbb{W}_1$.

Note that even though we now know the exit-prize $k \in [cx_1]$ of player $\mathcal{A}$ from cluster $[cx_1]$, the constraints imposed by $\mathbb{W}_1$ are $\forall\ k \in [cx_1]$ since at time χ_{Bj} player $\mathcal{B}$ cannot necessarily predict k.

In general, for every $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0])$, we can pre-compute the earliest possible entry-time, τ_{Bj}, to each prize $j \in [cy_0]$ such that player $\mathcal{B}$ satisfies the constraints imposed by $\mathbb{W}_0$, and for every $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0])$, we can precompute the earliest possible entry-time, τ_{Aj}, to each prize $j \in [cx_0]$ such that player $\mathcal{A}$ satsifies the constraints imposed by $\mathbb{W}_0$.

The remaining cluster window scenarios that are necessary for CLUSTER-DT are equivalent to those cases discussed thus far, but with either $\mathcal{A} \rhd \mathbb{W}_0$ or $\mathcal{B} \rhd \mathbb{W}_0$. Appendix 7.A.4 provides the remaining details.

7.3.6 Summary

We now have in place all the building blocks necessary to generalize PRIZE-DT to clusters using the look-through/move-through paradigm. The milestones established in this section were dominance, look-through, move-through, deadlock resolution, cluster-lead, direct-cluster-lead, CLUSTER-GUARANTEE, the two cluster problem and cluster feasible windows.

7.4 Strategic Engine: CLUSTER-PARANOID

The CLUSTER-PARANOID strategic engine generalizes the concept of the PRIZE-PARANOID tactical engine. The concept of a commitment is easily generalized to lookthrough. However, there is no obvious generalization of a probe, thus we cannot attempt to generalize ORIGINAL-DT which employs probes exclusively. Hence, we must recast the idea of PRIZE-PARANOID in terms of commitments rather than probes.

7.4.1 Game Table Generation

We can now define how to generate the CLUSTER-PARANOID game table.

- $\forall [cx_0] \in \mathcal{L}_A$ and $\forall [cy_0] \in \mathcal{L}_B$ commit $\mathcal{A} \rightarrow P_A \rhd [cx_0]$ and $\mathcal{B} \rightarrow P_B \rhd [cy_0]$.

Evaluate each commitment $\mathcal{A} \rightarrow P_A \rhd [cx_0]$ and $\mathcal{B} \rightarrow P_B \rhd [cy_0]$ by determining the CLUSTER-GUARANTEE cluster-path corresponding to $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$. The familiar MINIMAX and MAXIMIN evaluators can easily be defined on the game table.

7.4.2 Example Strategic Problem Revisited

Section 7.3.3.5 compared the strategies determined by CLUSTER-GUARANTEE on the example problem of Figure 7.10. We now consider the strategy determined by CLUSTER-PARANOID for

Table 7.9: Strategic Example of CLUSTER-PARANOID Game Tables

	[1]	[2]	[3]	[4]	[5]
[1]	207	222	222	290	563
[2]	0	0	0	163	372
[3]	209	0	0	0	209
[4]	258	325	325	0	467
[5]	391	311	237	267	∞

(a) Player $\mathcal{A}$ Root Game Table

	[1]	[2]	[3]	[4]	[5]
[1]	417	∞	628	354	354
[2]	372	∞	∞	372	372
[3]	309	∞	∞	372	372
[4]	467	502	∞	∞	467
[5]	179	377	377	179	0

(b) Player $\mathcal{B}$ Root Game Table

the same example problem. Tables 7.9(a)–(b) present the CLUSTER-PARANOID game tables, for player $\mathcal{A}$ and player $\mathcal{B}$ respectively, employing the PRIZE-GUARANTEE tactical engine throughout.

7.4.2.1 Analysis of Player $\mathcal{A}$'s CLUSTER-PARANOID Game Table

A MINIMAX analysis of the player $\mathcal{A}$ game table, Table 7.9(a), implies $\mathcal{B} \triangleright [4]$ and, hence, player $\mathcal{B}$'s best response is $\mathcal{A} \triangleright [1]$. Consider further the scenario $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [4]$.

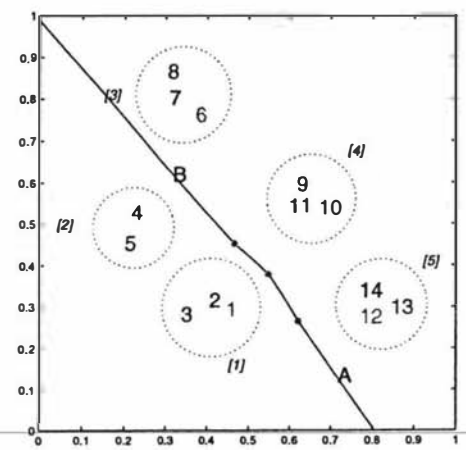
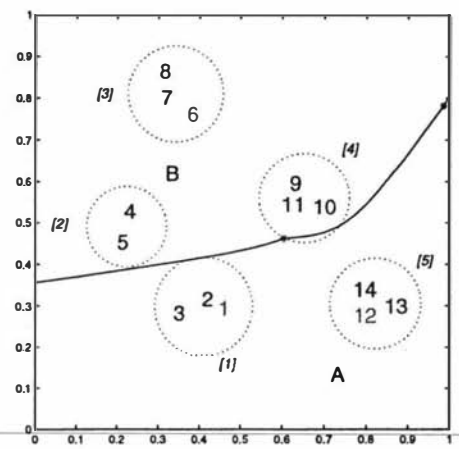
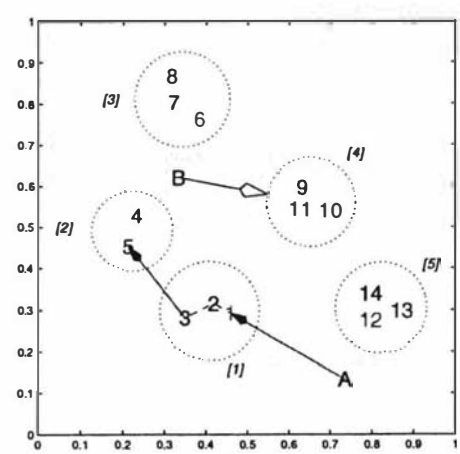
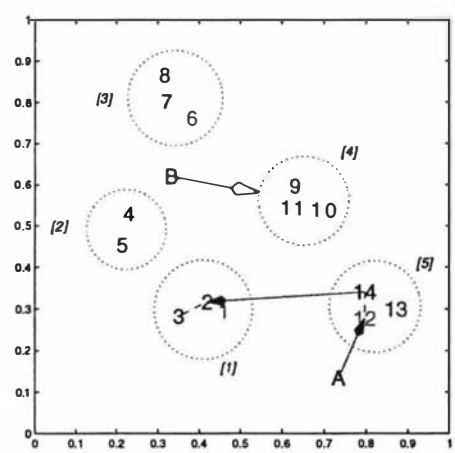
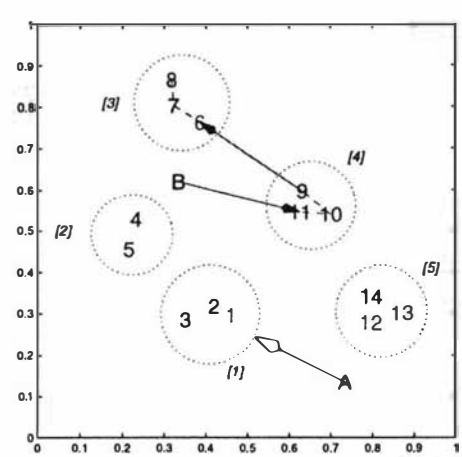
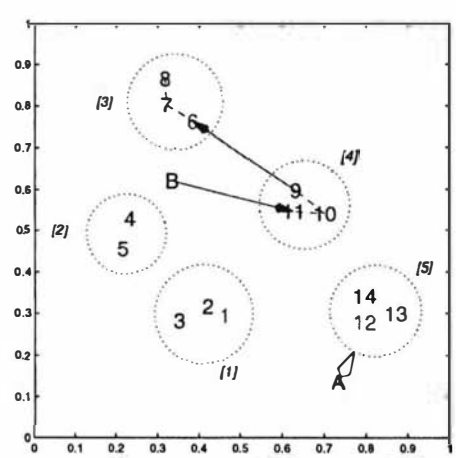
- Table 7.10(a) shows the equal earliest arrival curve given that player $\mathcal{A}$ must claim all the prizes from cluster [1] and player $\mathcal{B}$ must claim all the prizes from cluster [4]. This indicates that cluster [2] is then a cluster lead for player $\mathcal{A}$, and that both clusters [3] and [5] are cluster leads for player $\mathcal{B}$.
- Table 7.10(c) illustrates player $\mathcal{A}$'s maximal cluster guarantee path given that $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [4]$ with $\mathcal{A} \rightarrow [1] \rightarrow [2]$ for an total of 290.
- Table 7.10(e) illustrates player $\mathcal{B}$'s maximal cluster guarantee path given that $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [4]$ with $\mathcal{B} \rightarrow [4] \rightarrow [3]$ for a total of 467.

7.4.2.2 Analysis of Player $\mathcal{B}$'s CLUSTER-PARANOID Game Table

A MINIMAX analysis of the player $\mathcal{B}$ game table, Table 7.9(b), implies $\mathcal{A} \triangleright [5]$ or $\mathcal{A} \triangleright [1]$ and, in either case, the best response for player $\mathcal{B}$ is $\mathcal{B} \triangleright [4]$. We have already considered the scenario $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [4]$. Instead, consider the scenario $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [4]$.

- Table 7.10(b) shows the equal earliest arrival curve given that player $\mathcal{A}$ must claim all the prizes from cluster [5] and player $\mathcal{B}$ must claim all the prizes from cluster [4]. This indicates that cluster [1] is then a cluster lead for player $\mathcal{A}$ and both clusters [2] and 35] are cluster leads for player $\mathcal{B}$.
- Table 7.10(d) illustrates player $\mathcal{A}$'s maximal cluster guarantee path given that $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [4]$ with $\mathcal{A} \rightarrow [5] \rightarrow [1]$ for an total of 267.
- Table 7.10(f) illustrates player $\mathcal{B}$'s maximal cluster guarantee path given that $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [4]$ with $\mathcal{B} \rightarrow [4] \rightarrow [3]$ for a total of 467.

Table 7.10: Strategic Example of CLUSTER-PARANOID

Player $\mathcal{A}$	Player $\mathcal{B}$
	
(a) $\mathcal{A} \triangleright$ [1] and $\mathcal{B} \triangleright$ [4] Lead	(b) $\mathcal{A} \triangleright$ [5] and $\mathcal{B} \triangleright$ [4] Lead
	
(c) $\mathcal{A} \rightarrow$[1]$\rightarrow$[2] and $\mathcal{B} \triangleright$ [4]	(d) $\mathcal{A} \rightarrow$[5]$\rightarrow$[1] and $\mathcal{B} \triangleright$ [4]
	
(e) $\mathcal{A} \triangleright$ [1] and $\mathcal{B} \rightarrow$[4]$\rightarrow$[3]	(f) $\mathcal{A} \triangleright$ [5] and $\mathcal{B} \rightarrow$[4]$\rightarrow$[3]

7.4.2.3 Discussion

The preceding analysis of the CLUSTER-PARANOID game tables suggests that $\mathcal{A}\rightarrow[1]$ and $\mathcal{B}\rightarrow[4]$. However, in Section 7.3.3.5, the maximal CLUSTER-GUARANTEE paths suggested $\mathcal{A}\rightarrow[5]$ and $\mathcal{B}\rightarrow[4]$. Indeed, player $\mathcal{B}$ selects its maximal cluster guarantee path under both strategic analyses. Together, though, the maximal cluster guarantee paths of the players exhaust only clusters $\{[3],[4],[5]\}$, whereas in the likely CLUSTER-PARANOID scenario, four clusters are exhausted in the evaluation.

7.4.3 Summary

We have designed the CLUSTER-PARANOID strategic engine and compared the strategies determined by CLUSTER-PARANOID with those determined by CLUSTER-GUARANTEE on the same illustrative example.

7.5 Strategic Engine: CLUSTER-DT

We now turn to the specification of the CLUSTER-DT strategy, a generalization of the PRIZE-DT strategy, scaled for cluster-objects instead of prize-objects. The aim of CLUSTER-DT is to strategically analyse the problem at a level of detail appropriate to a natural cluster structure and to then focus on tactically analysing a few chosen clusters in terms of individual prizes.

Suppose that a family-cluster structure is given. We apply the *family-no-return* and ANY-ALL-PCTSP cluster harvesting rules of Section 7.2.1 but specifically not the *family-sequence* or *final-family rules*. The *cluster-no-return* rule is implicit in that a cluster is only moved through once, either by one player only or by both players simultaneously.

7.5.1 Strategic Targeting and Strategic Commitment

The following strategic cluster targeting options of player $\mathcal{A}$ parallel the tactical prize targeting options for PRIZE-DT, as in Section 6.3.

(i). Those cluster-leads of player $\mathcal{A}$.

(ii). Those pairs of clusters, $\{[ci_1],[ci_2]\}$, which are not cluster-leads of player $\mathcal{A}$ but either:

- cluster $[ci_2]$ is a cluster-lead for player $\mathcal{A}$ if player $\mathcal{B}$ looks through cluster $[ci_1]$; or
- cluster $[ci_1]$ is a cluster-lead for player $\mathcal{A}$ if player $\mathcal{B}$ looks through cluster $[ci_2]$.

This provides a one-cluster-contingency lookahead option.

(iii). Those pairs of clusters, $\{[ci_1],[ci_2]\}$, which are not cluster-leads of player $\mathcal{A}$, nor are they cluster-leads if player $\mathcal{B}$ looks through the other cluster in the pair, but, if player $\mathcal{B}$ were to look through a distinct third cluster, then player $\mathcal{B}$ could not play a contingent strategy as in Section 7.3.5. This provides a two-cluster-contingency lookahead option.

Similarly for player $\mathcal{B}$'s set of options.

The strategic engine CLUSTER-DT is also structured around a game tree similar to that of PRIZE-DT. At each game tree node, the players select from the three types of strategic options. The projection of each player through its proposed target is similar to the *commitment* of PRIZE-DT but must account for cluster look-through.

Clusters are much better suited to commitments than to probes. We do not generalize ORIGINAL-DT to clusters since it would be very difficult to generalize the concept of a *probe* to clusters. However, a commitment does not imply a specific path through the sequence of prizes in a cluster as, in the case of a look-through cluster, we must determine a sequence of prizes for each possible target cluster following.

A **cluster planning path**, $\mathbb{P}_A$, associated with player $\mathcal{A}$, consists of a sequence of clusters, possibly interspersed with cluster feasibility windows. A **projected game position** consists of a pair of cluster planning paths, $\mathbb{P}_A$ for player $\mathcal{A}$ and $\mathbb{P}_B$ for player $\mathcal{B}$; a pair of **projected locations**, A and B; a pair of **projected time-stamps**, t_A and t_B; and a **projected set of remaining clusters**, $\mathcal{Q}$.

Each player may also look through a cluster. A look-through cluster is a *commitment* to move through that cluster next, but the projected location of a player is that prior to moving through the look-through cluster. A CLUSTER-DT **game tree node** is uniquely defined by a game position, plus any committed look-through of each player.

The game tree *root node* is defined by $\mathbb{P}_A = \emptyset$, $\mathcal{A} \rhd \emptyset$, $\mathbb{P}_B = \emptyset$ and $\mathcal{B} \rhd \emptyset$. If $t_A \neq t_B$ at the root node, then there can be no direct-cluster leads from the root node and, hence, every subsequent child node must be such that both players look through a cluster. This considerably simplifies the game tree since we cannot have the situation in which only one looks through a cluster. Hence, we impose the important restriction that $t_A = t_B$ at the root node.

Let $\mathcal{Q}_A$ be the set of clusters available to player $\mathcal{A}$ (in the family-no-return sense) and let $\mathcal{Q}_B$ be the set of clusters available to player $\mathcal{B}$ (in the family-no-return sense). Let $\mathcal{D}_A$ be the set of *direct-cluster-leads* of player $\mathcal{A}$ and let $\mathcal{D}_B$ be the set of *direct-cluster-leads* of player $\mathcal{B}$. Let $\mathcal{L}_A$ be the set of *cluster-leads* of player $\mathcal{A}$ and let $\mathcal{L}_B$ be the set of *cluster-leads* of player $\mathcal{B}$. Let $\mathcal{F}_A = \{\{[cx_1],[cx_2]\} : [cx_1],[cx_2] \in \mathcal{Q}_A - (\mathcal{D}_A \cup \mathcal{L}_A), [cx_1] \neq [cx_2]\}$ be the **cluster-follow-pairs** of player $\mathcal{A}$. Let $\mathcal{F}_B = \{\{[cy_1],[cy_2]\} : [cy_1],[cy_2] \in \mathcal{Q}_B - (\mathcal{D}_B \cup \mathcal{L}_B), [cy_1] \neq [cy_2]\}$ be the **cluster-follow-pairs** of player $\mathcal{B}$.

The following notation is employed, overloading the $\rhd$ and $\oplus$ symbols with similar interpretations as employed in PRIZE-DT

- "**commit** $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cx_1]$" defines a child game tree node in which player $\mathcal{A}$ will have the same cluster planning path $\mathbb{P}_A$ and will look through cluster $[cx_1]$.
- "**commit** $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0]$" defines a child game tree node in which cluster $[cx_0]$ is appended to $\mathbb{P}_A$, player $\mathcal{A}$ will move through cluster $[cx_0]$ and will not look through a cluster.
- "**commit** $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$" defines a child game tree node in which cluster $[cx_0]$ is appended to $\mathbb{P}_A$, player $\mathcal{A}$ will move through $[cx_0]$ with respect to targeting $[cx_1]$ as the next target cluster, and will then look through cluster $[cx_1]$.

Note that $\mathbb{P}_A$ includes the sequence of clusters which player $\mathcal{A}$ *has moved through* but not the look-through cluster which is committed to *but not yet* moved through.

Reverse Leads. We do not consider reverse lead scenarios in CLUSTER-DT. Since neither player would dominate the target-cluster, at some subsequent game position, the deadlock resolution may fail, introducing many complicated cases.

7.5.2 Game Tree Generation

We now describe how to generate the game-tree by showing how to generate the children of any given game-tree decision-node. There are many more cases for CLUSTER-DT than for PRIZE-DT since there are many combinations of look-through clusters, lead clusters and cluster-follow-pairs to consider.

▼ Expansion Rules for Cluster-Lead vs Cluster-Lead Scenarios

Case (L1). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$

- $\forall\, [cx_1] \in \mathcal{L}_A$ and $\forall\, [cy_1] \in \mathcal{L}_B$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \rhd [cy_1]$.

Case (L2). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$

- $\forall\, [cx_1] \in \mathcal{D}_A$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ (maintaining $\mathcal{B}{\rightarrow}\mathbb{P}_B \rhd [cy_0]$).
- $\forall\, [cy_1] \in \mathcal{D}_B$, commit $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cy_0] \rhd [cy_1]$ (maintaining $\mathcal{A}{\rightarrow}\mathbb{P}_A \rhd [cx_0]$).
- $\forall\, [cx_1] \in \mathcal{L}_A$ and $\forall\, [cy_1] \in \mathcal{L}_B$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cy_0] \rhd [cy_1]$.

Case (L3a). $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0])$ and $\mathcal{B} \rhd [cx_0]$

- $\forall\, [cx_1] \in \mathcal{D}_A$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cx_1]$ (maintaining $\mathcal{B}{\rightarrow}\mathbb{P}_B \rhd [cy_0]$).
- $\forall\, [cy_1] \in \mathcal{D}_B$, commit $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cy_0] \rhd [cy_1]$ (maintaining $\mathcal{A}{\rightarrow}\mathbb{P}_A \rhd (\mathbb{W}_0{\rightarrow}[cx_0])$).
- $\forall\, [cx_1] \in \mathcal{L}_A$ and $\forall\, [cy_1] \in \mathcal{L}_B$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cy_0] \rhd [cy_1]$.

Case (L3b). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0])$

- $\forall\, [cx_1] \in \mathcal{D}_A$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ (maintaining $\mathcal{B}{\rightarrow}\mathbb{P}_B \rhd (\mathbb{W}_0{\rightarrow}[cy_0])$).
- $\forall\, [cy_1] \in \mathcal{D}_B$, commit $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus (\mathbb{W}_0{\rightarrow}[cy_0]) \rhd [cy_1]$ (maintaining $\mathcal{A}{\rightarrow}\mathbb{P}_A \rhd [cx_0]$).
- $\forall\, [cx_1] \in \mathcal{L}_A$ and $\forall\, [cy_1] \in \mathcal{L}_B$, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus (\mathbb{W}_0{\rightarrow}[cy_0]) \rhd [cy_1]$.

▼ Expansion Rules for Cluster-Lead vs Cluster-Follow-Pair Scenarios

This section is closely linked with Appendix 7.A.4, which provides details of the definitions and algorithms to determine if various cluster-window scenarios are cluster-window-feasible.

Case (F1). $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$

- Choose a follow pair $\{[cy_1], [cy_2]\} \in \mathcal{F}_B$.
- If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd [cy_1]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd [cy_2]$.
- If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd [cy_2]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd [cy_1]$.
- Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$ (see case (i) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \mathbb{W}_1$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow [cy])$.
  - Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F2). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cx_0]$

- Choose a follow pair $\{[cy_1], [cy_2]\} \in \mathcal{F}_B$.
- If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_1])$ and $\mathcal{B} \rhd [cx_0]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd [cy_2]$.
- If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_2])$ and $\mathcal{B} \rhd [cx_0]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd [cy_1]$.
- Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$ (see case (iii) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1 \rightarrow [cy])$.
  - Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F3). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$ where $[cx_0] \neq [cy_0]$

- Choose a follow pair $\{[cy_1], [cy_2]\} \in \mathcal{F}_B$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$ (see case (ii) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy_2])$.
  - If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_2])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy_1])$.

- Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ (see case (vi) of Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.7.
- If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy])$.
- Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F4a). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0])$ and $\mathcal{B} \rhd [cx_0]$

- Choose a follow pair $\{[cy_1], [cy_2]\} \in \mathcal{F}_B$.
- If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cy_1])$ and $\mathcal{B} \rhd [cx_0]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd [cy_2]$.
- If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cy_2])$ and $\mathcal{B} \rhd [cx_0]$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd [cy_1]$.
- Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$ (see case (ix) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1 \rightarrow [cy])$.
  - Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F4b). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0])$

- Choose a follow pair $\{[cy_1], [cy_2]\} \in \mathcal{F}_B$.
- If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0])$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cy_2]$.
- If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cy_2])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0])$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd [cy_1]$.
- Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow \{[cy_1], [cy_2]\})$ (see case (viii) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cx_0]) \rhd (\mathbb{W}_1 \rightarrow [cy])$.

- Otherwise, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd (\mathbb{W}_1{\rightarrow}\{[cy_1],[cy_2]\})$.

Case (F5a). $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0])$ and $\mathcal{B} \rhd [cy_0]$ where $[cx_0] \neq [cy_0]$

- Choose a follow pair $\{[cy_1],[cy_2]\} \in \mathcal{F}_B$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0])$ and $\mathcal{B} \rhd ([cy_0]{\rightarrow}\{[cy_1],[cy_2]\})$ (see case (x) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0]{\rightarrow}[cy_1])$ and $\mathcal{B} \rhd ([cy_0]{\rightarrow}\mathbb{W}_1)$, then commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cy_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus ([cy_0]) \rhd (\mathbb{W}_1{\rightarrow}[cy_2])$.
  - If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0]{\rightarrow}[cy_2])$ and $\mathcal{B} \rhd ([cy_0]{\rightarrow}\mathbb{W}_1)$, then commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cy_2]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus ([cy_0]) \rhd (\mathbb{W}_1{\rightarrow}[cy_1])$.
  - Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1],[cy_2]\}$.
  - Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0]{\rightarrow}[cx_1])$ and $\mathcal{B} \rhd ([cy_0]{\rightarrow}\mathbb{W}_1{\rightarrow}\{[cy_1],[cy_2]\})$ (see case (xi) of Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.7.
  - If cluster $[cy] \in \{[cy_1],[cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd (\mathbb{W}_0{\rightarrow}[cx_0]{\rightarrow}[cx_1])$ and $\mathcal{B} \rhd ([cy_0]{\rightarrow}\mathbb{W}_1)$, then commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1{\rightarrow}[cy])$.
  - Otherwise, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus (\mathbb{W}_0{\rightarrow}[cx_0]) \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \rhd ([cy_0]{\rightarrow}\mathbb{W}_1{\rightarrow}\{[cy_1],[cy_2]\})$.

Case (F5b). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0])$ where $[cx_0] \neq [cy_0]$

- Choose a follow pair $\{[cy_1],[cy_2]\} \in \mathcal{F}_B$.
- Let $\mathbb{W}_1$ be the cluster-window scenario: $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0]{\rightarrow}\{[cy_1],[cy_2]\})$ (see case (iv) of Appendix 7.A.4).
- If $\mathbb{W}_1$ is *cluster-window-feasible* then
  - If cluster $[cy_2]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0]{\rightarrow}[cy_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0]{\rightarrow}\mathbb{W}_1)$, then commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus (\mathbb{W}_0{\rightarrow}[cy_0]) \rhd (\mathbb{W}_1{\rightarrow}[cy_2])$.
  - If cluster $[cy_1]$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0]{\rightarrow}[cy_2])$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0]{\rightarrow}\mathbb{W}_1)$, then commit $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus (\mathbb{W}_0{\rightarrow}[cy_0]) \rhd (\mathbb{W}_1{\rightarrow}[cy_1])$.
  - Choose a cluster $[cx_1] \in \mathcal{L}_A \setminus \{[cy_1],[cy_2]\}$.
  - Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0]{\rightarrow}[cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0{\rightarrow}[cy_0]{\rightarrow}\mathbb{W}_1{\rightarrow}\{[cy_1],[cy_2]\})$ (see case (vii) of Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.7.

- If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cy_0]) \rhd (\mathbb{W}_1 \rightarrow [cy])$.
- Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F6). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow [cy_2])$.
- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow [cy_1])$.
- Choose a cluster $[cx_1] \in \mathcal{Q}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ (see case (v) Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.5.
- If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd \mathbb{W}_1$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow [cy])$.
- Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F7). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ where $]cx_0] \neq [cy_0]$.

- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy_2])$.
- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy_1])$.
- Choose a cluster $[cx_1] \in \mathcal{Q}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ (see case (vi) of Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.7.
- If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1 \rightarrow [cy])$.
- Otherwise, commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$.

Case (F8). $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ where $]cx_0] \neq [cy_0]$.

- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cy_0]) \rhd (\mathbb{W}_1 \rightarrow [cy_2])$.
- Commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cy_2]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cy_0]) \rhd (\mathbb{W}_1 \rightarrow [cy_1])$.
- Choose a cluster $[cx_1] \in \mathcal{Q}_A \setminus \{[cy_1], [cy_2]\}$.
- Let $\mathbb{W}_2$ be the cluster-window scenario: $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ (see case (vii) of Appendix 7.A.4). Then $\mathbb{W}_2$ is *cluster-window-feasible* by Lemma 7.A.7.
- If cluster $[cy] \in \{[cy_1], [cy_2]\}$ is a cluster-lead for $\mathcal{B}$ conditional upon $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1)$, then commit $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cy_0]) \rhd (\mathbb{W}_1 \rightarrow [cy])$.

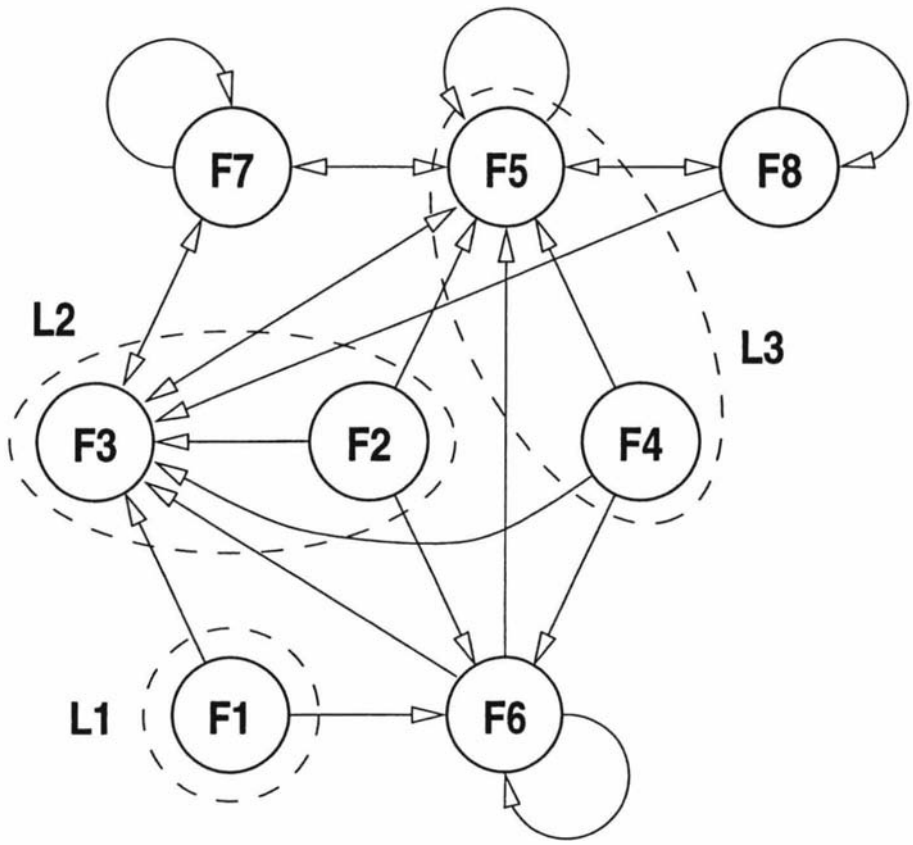


Figure 7.11: State transitions between states (F1)–(F8).

- Otherwise, commit $\mathcal{A}{\rightarrow}\mathbb{P}_A{\oplus}[cx_0]{\triangleright}[cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B{\triangleright}(\mathbb{W}_0{\rightarrow}[cy_0]{\rightarrow}\mathbb{W}_1{\rightarrow}\{[cy_1],[cy_2]\})$.

Expansion rules analogous to (F1)–(F8) can be defined in which we choose a follow pair $\{[cx_1],[cx_2]\} \in \mathcal{F}_A$ and choose a cluster $[cy_1] \in \mathcal{Q}_B \setminus \{[cx_1],[cx_2]\}$. We can continue to apply the expansion rules until $\mathcal{D}_A = \emptyset$, $\mathcal{L}_A = \emptyset$, $\mathcal{F}_A = \emptyset$, $\mathcal{D}_B = \emptyset$, $\mathcal{L}_B = \emptyset$ and $\mathcal{F}_B = \emptyset$ which is then a **terminal node** of the game tree.

▼ State Transitions

The state transitions between cases (F1)–(F8) by applying the expansion rules (F1)–(F8) are summarised in Figure 7.11, in which case (L1) is identified with case (F1), case (L2) is identified with cases (F2) and (F3), and case (L3) is identified with cases (F4) and (F5). The transitions between cases (L1)–(L3) and cases (F6)–(F8) are summarised in Figure 7.12, in which the dashed arrows indicate the application of expansion rules (L1)–(L3) and the solid arrows indicate the application of expansion rules (F1)–(F8).

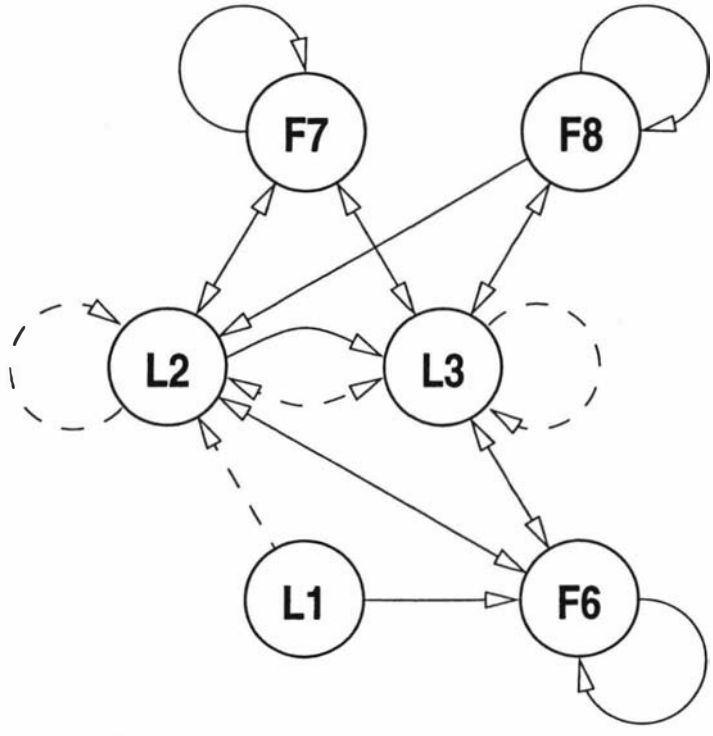


Figure 7.12: State transitions between states (L1)–(L3) and (F6)–(F8).

7.5.3 Evaluating and Searching the CLUSTER-DT Game Tree

We use the same evaluators and α–β search as in PRIZE-DT except that there is are no *reverse lead* scenarios. States (F1)–(F5) apply the GENERALIZED-MINIMAX or GENERALIZED-MAXIMIN evaluator and states (F6)–(F8) apply the GENERALIZED-MAX or GENERALIZED-MIN evaluator. Although CLUSTER-GUARANTEE does not give a strict lower bound on the value of a projected game position, it does give a lower bound on the value of the game position evaluated by searching the subtree from that game tree node.

7.5.3.1 Evaluation of a Terminal Game Tree Node

Let j be a game tree lead node. If j is the root node, then $v(j) = 0$. Otherwise, $\mathcal{A} \triangleright [cx_0]$ and $\mathcal{B} \triangleright [cy_0]$ and neither player has a direct-cluster-lead or cluster-lead. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A} \rightarrow \{[cx_0]\}$ and $\mathcal{B} \rightarrow \{[cx_0], [cy_0]\}$ to determine the exit-value from cluster $[cx_0]$ of player $\mathcal{A}$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A} \rightarrow \{[cx_0], [cy_0]\}$ and $\mathcal{B} \rightarrow \{[cy_0]\}$ to determine the exit-value from cluster $[cy_0]$ of player $\mathcal{B}$.

7.5.3.2 Evaluation of a Non-Terminal Game Tree Node

Suppose player $\mathcal{A}$ moves through cluster $[cx_0]$ whilst targeting cluster $[cx_1]$, i.e., $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \triangleright [cx_1]$. We note the exit-prize, exit-time and exit-value from cluster $[cx_0]$. The exit-prize becomes the new projected player $\mathcal{A}$ location and the exit-time becomes the new projected player $\mathcal{A}$

Table 7.11: Strategic Example of CLUSTER-DT Root Game Tables

	[1]	[2]	[3]	[4]
[1]	480	320	354	533
[5]	470	**437**	469	370
{[2],[3]}	372	209	354	372
{[2],[4]}	437	**437**	437	372
{[3],[4]}	437	**437**	437	372

(a) Player $\mathcal{A}$ Root Game Table

	[1]	[5]	{[2],[3]}	{[2],[4]}	{[3],[4]}
[1]	485	429	628	487	487
[2]	600	520	763	533	533
[3]	536	452	646	563	563
[4]	467	**536**	623	623	628

(b) Player $\mathcal{B}$ Root Game Table

time-stamp. The evaluation uses the exit-value as per:

$$\upsilon(\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]) = \upsilon(\mathbb{P}_A \rhd [cx_0]) + \text{exit-value} \tag{7.6}$$

This evaluation also depends upon the look-through and targeting scenario of player $\mathcal{B}$.

This completes all that is necessary to define the CLUSTER-DT strategic engine.

7.5.4 Example Strategic Problem Revisited

Section 7.4.2 compared the strategies determined by CLUSTER-GUARANTEE and CLUSTER-PARANOID on the example problem of Figure 7.10. We now consider the strategy determined by CLUSTER-DT for the same example problem.

Table 7.11(a)–(b) present the root CLUSTER-DT game tables, for player $\mathcal{A}$ and player $\mathcal{B}$ respectively, applying the MINIMAX evaluator and employing the FAMILY-PRIZE-GUARANTEE tactical engine throughout.

7.5.4.1 Analysis of Player $\mathcal{A}$ MINIMAX

Table 7.11(a) gives the CLUSTER-DT game table for player $\mathcal{A}$. The MINIMAX analysis implies $\mathcal{B} \rhd [2]$ and hence either $\mathcal{A} \rhd [5]$, $\mathcal{A} \rhd \{[2],[4]\}$ or $\mathcal{A} \rhd \{[3],[4]\}$. Each of these strategic scenarios has a MINIMAX evaluation of 437 which equates to player $\mathcal{A}$ claiming all the prizes from clusters [4] and [5] but in different sequences.

- Consider the scenario $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [2]$.
  - Table 7.12(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.12(a) gives the corresponding game table for player $\mathcal{A}$ and Table 7.12(b) gives the corresponding game table for player $\mathcal{B}$. In both cases a MINIMAX analysis implies $\mathcal{A} \rhd [4]$ and $\mathcal{B} \rhd [1]$.
  - Table 7.12(d) illustrates how the game evolves with $\mathcal{A} \rightarrow [5] \rightarrow [4]$ and $\mathcal{B} \rightarrow [2] \rightarrow [1] \rightarrow [3]$.

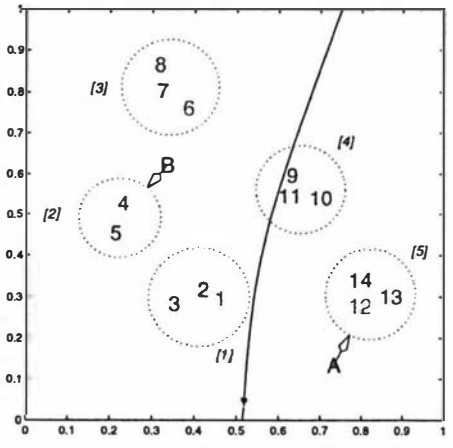
Table 7.12: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [2]$

	[1]	[3]
[4]	437	482
{[1], [3]}	328	370

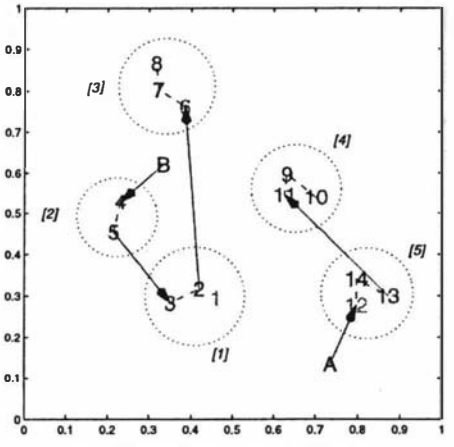
(a) Player $\mathcal{A}$ Game Table

	[4]	{[1], [3]}
[1]	520	612
[3]	372	472

(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division



(d) Strategic Game Evolution

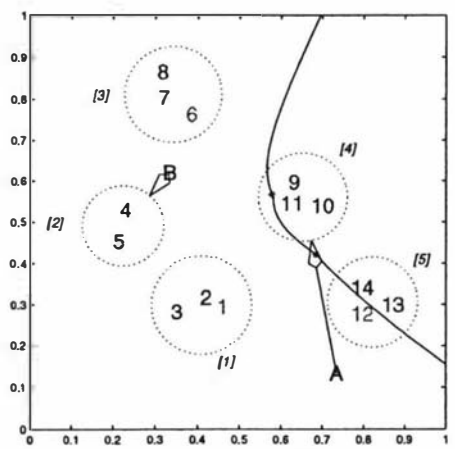
Table 7.13: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \triangleright [4]$ and $\mathcal{B} \triangleright [2]$

	[1]	[3]
[5]	371	437
{[1], [3]}	467	383

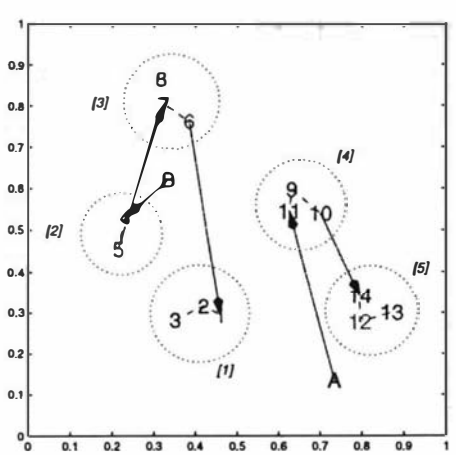
(a) Player $\mathcal{A}$ Game Table

	[5]	{[1], [3]}
[1]	563	533
[3]	471	372

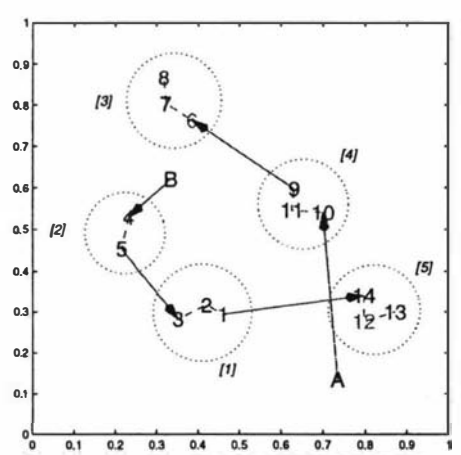
(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division



(d) Player $\mathcal{A}$'s Anticipated Strategic Game Evolution



(e) Player $\mathcal{B}$'s Anticipated Strategic Game Evolution

- Consider the scenario $\mathcal{A} \rhd \{[2],[4]\}$ (in which actually $\mathcal{A} \rhd [4]$) and $\mathcal{B} \rhd [2]$.
  - Table 7.13(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.13(a) gives the corresponding game table for player $\mathcal{A}$ for which a MINIMAX analysis implies $\mathcal{B} \rhd [3]$ and hence $\mathcal{A} \rhd [5]$. Table 7.13(b) gives the corresponding game table for player $\mathcal{B}$ for which a MINIMAX analysis implies $\mathcal{A} \rhd \{[1],[3]\}$ and hence $\mathcal{B} \rhd [1]$, which in turn implies $\mathcal{A} \rhd [3]$. In summary, player $\mathcal{A}$ assumes $\mathcal{B} \rhd [3]$ and hence selects $\mathcal{A} \rhd [5]$, whilst player $\mathcal{B}$ assumes $\mathcal{A} \rhd \{[1],[3]\}$ and hence selects $\mathcal{B} \rhd [1]$. These strategic differences come about since both game tables are evaluated using MINIMAX. If MAXIMIN is applied, then $\mathcal{A} \rhd [3]$ and $\mathcal{B} \rhd [1]$ is implied.
  - Table 7.13(d) illustrates how player $\mathcal{A}$ anticipates the game will evolve with $\mathcal{A}\rightarrow[4]\rightarrow[5]$ and $\mathcal{B}\rightarrow[2]\rightarrow[3]\rightarrow[1]$. Table 7.13(e) illustrates how player $\mathcal{B}$ anticipates the game will evolve with $\mathcal{A}\rightarrow[4]\rightarrow[3]$ and $\mathcal{B}\rightarrow[2]\rightarrow[1]\rightarrow[5]$.
- Similarly, in the scenario $\mathcal{A} \rhd \{[3],[4]\}$ and $\mathcal{B} \rhd [2]$, also $\mathcal{A} \rhd [4]$, so the analysis will be the same as that above.

7.5.4.2 Analysis of Player $\mathcal{B}$ MINIMAX

Table 7.11(b) gives the CLUSTER-DT game table for player $\mathcal{B}$. The MINIMAX analysis implies $\mathcal{A} \rhd [5]$ and hence $\mathcal{B} \rhd [4]$ with evaluation 536.

- Consider the scenario $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [4]$.
  - Table 7.14(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.14(a) gives the corresponding game table for player $\mathcal{A}$ and Table 7.14(b) gives the corresponding game table for player $\mathcal{B}$. In both cases, a MINIMAX or MAXIMIN analysis implies $\mathcal{A} \rhd [1]$ and $\mathcal{B} \rhd [2]$.
  - Table 7.14(d) illustrates how the game evolves with $\mathcal{A}\rightarrow[5]\rightarrow[1]$ and $\mathcal{B}\rightarrow[4]\rightarrow[2]\rightarrow[3]$.

7.5.4.3 Analysis of Other Interesting Scenarios

Several other interesting scenarios are considered below.

- Consider the scenario $\mathcal{A} \rhd [1]$ and $\mathcal{B} \rhd [1]$.
  - Table 7.15(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.15(a) gives the corresponding game table for player $\mathcal{A}$ and Table 7.15(b) gives the corresponding game table for player $\mathcal{B}$. In both cases, a MINIMAX analysis implies $\mathcal{A} \rhd [4]$ and $\mathcal{B} \rhd [2]$.

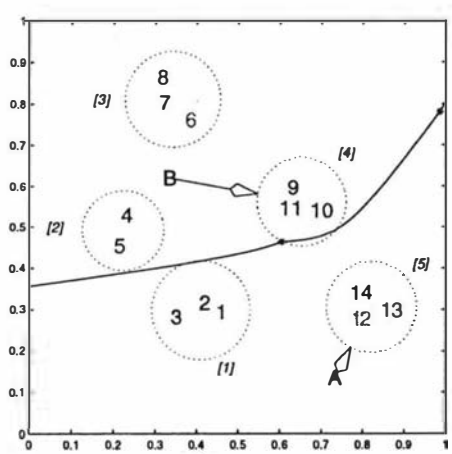
Table 7.14: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [4]$

	[2]	[3]
[1]	370	469
{[2], [3]}	342	473

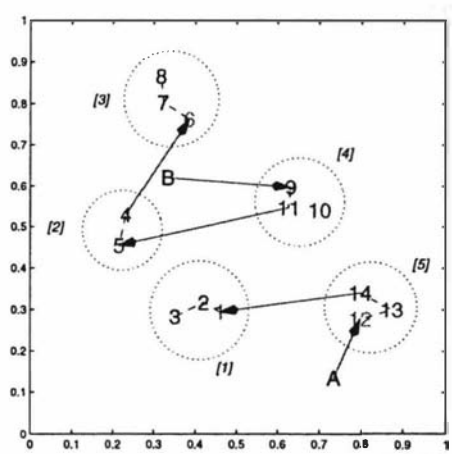
(a) Player $\mathcal{A}$ Game Table

	[1]	{[2], [3]}
[2]	536	612
[3]	467	467

(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division



(d) Strategic Game Evolution

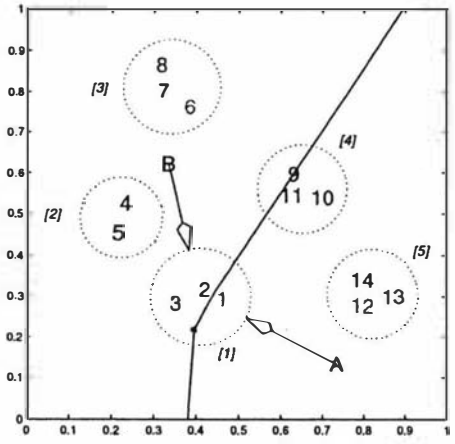
Table 7.15: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [1]$

	[2]	[3]	{[4],[5]}
[4]	480	480	510
[5]	480	480	325
{[2],[3]}	257	597	

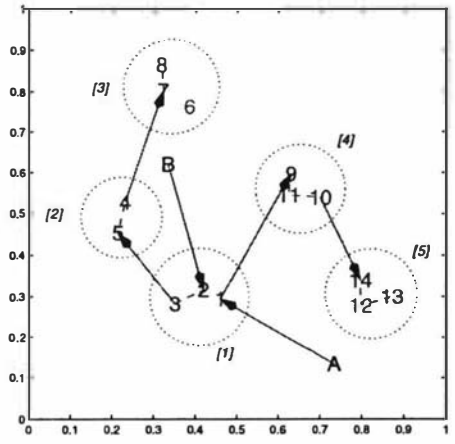
(a) Player $\mathcal{A}$ Game Table

	[4]	[5]	{[2],[3]}
[2]	485	520	572
[3]	444	428	322
{[4],[5]}	392	615	

(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division



(d) Strategic Game Evolution

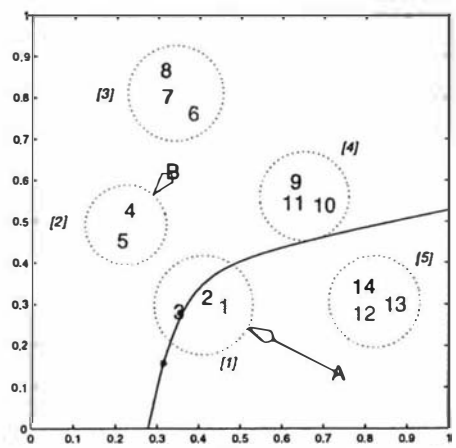
Table 7.16: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [2]$

	[3]	[4]
[5]	370	290
{[3], [4]}	548	320

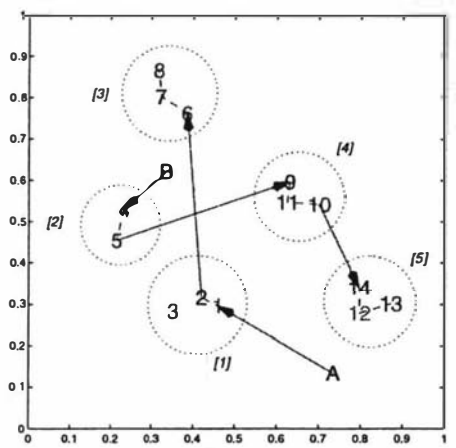
(a) Player $\mathcal{A}$ Game Table

	[5]	{[3], [4]}
[3]	538	372
[4]	630	600

(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division



(d) Strategic Game Evolution

  - Table 7.15(d) illustrates how the game evolves with $\mathcal{A} \rightarrow [1] \rightarrow [4] \rightarrow [5]$ and $\mathcal{B} \rightarrow [1] \rightarrow [2] \rightarrow [3]$.

- Consider the scenario $\mathcal{A} \triangleright [1]$ and $\mathcal{B} \triangleright [2]$.
  - Table 7.16(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.16(a) gives the corresponding game table for player $\mathcal{A}$ and Table 7.16(b) gives the corresponding game table for player $\mathcal{B}$. In both cases, a MINIMAX or MAXIMIN analysis implies $\mathcal{A} \triangleright \{[3], [4]\}$ and $\mathcal{B} \triangleright [4]$.
  - Table 7.16(d) illustrates how the game evolves with $\mathcal{A} \rightarrow [1] \rightarrow [3]$ and $\mathcal{B} \rightarrow [2] \rightarrow [4] \rightarrow [5]$.

- Consider the scenario $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [3]$. In this case, player $\mathcal{B}$ targets the cluster farthest from player $\mathcal{A}$.
  - Table 7.17(c) shows the equal earliest arrival time curve, which indicates the dividing line.
  - Table 7.17(a) gives the corresponding game table for player $\mathcal{A}$ for which a MINIMAX analysis implies $\mathcal{B} \triangleright [4]$ and hence $\mathcal{A} \triangleright [1]$. Table 7.17(b) gives the corresponding

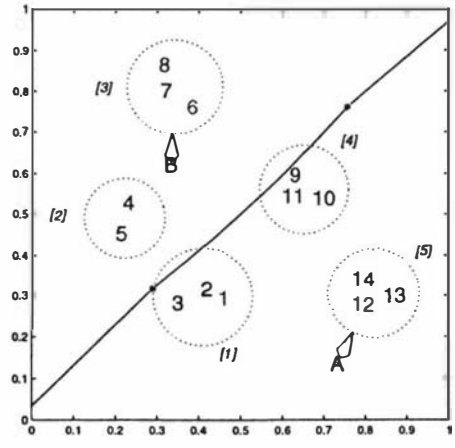
Table 7.17: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \rhd [5]$ and $\mathcal{B} \rhd [3]$

	[2]	[4]
[1]	488	**469**
[4]	437	390

(a) Player $\mathcal{A}$ Game Table

	[1]	[4]
[2]	372	**452**
[4]	467	390

(b) Player $\mathcal{B}$ Game Table



(c) Cluster Lead Division

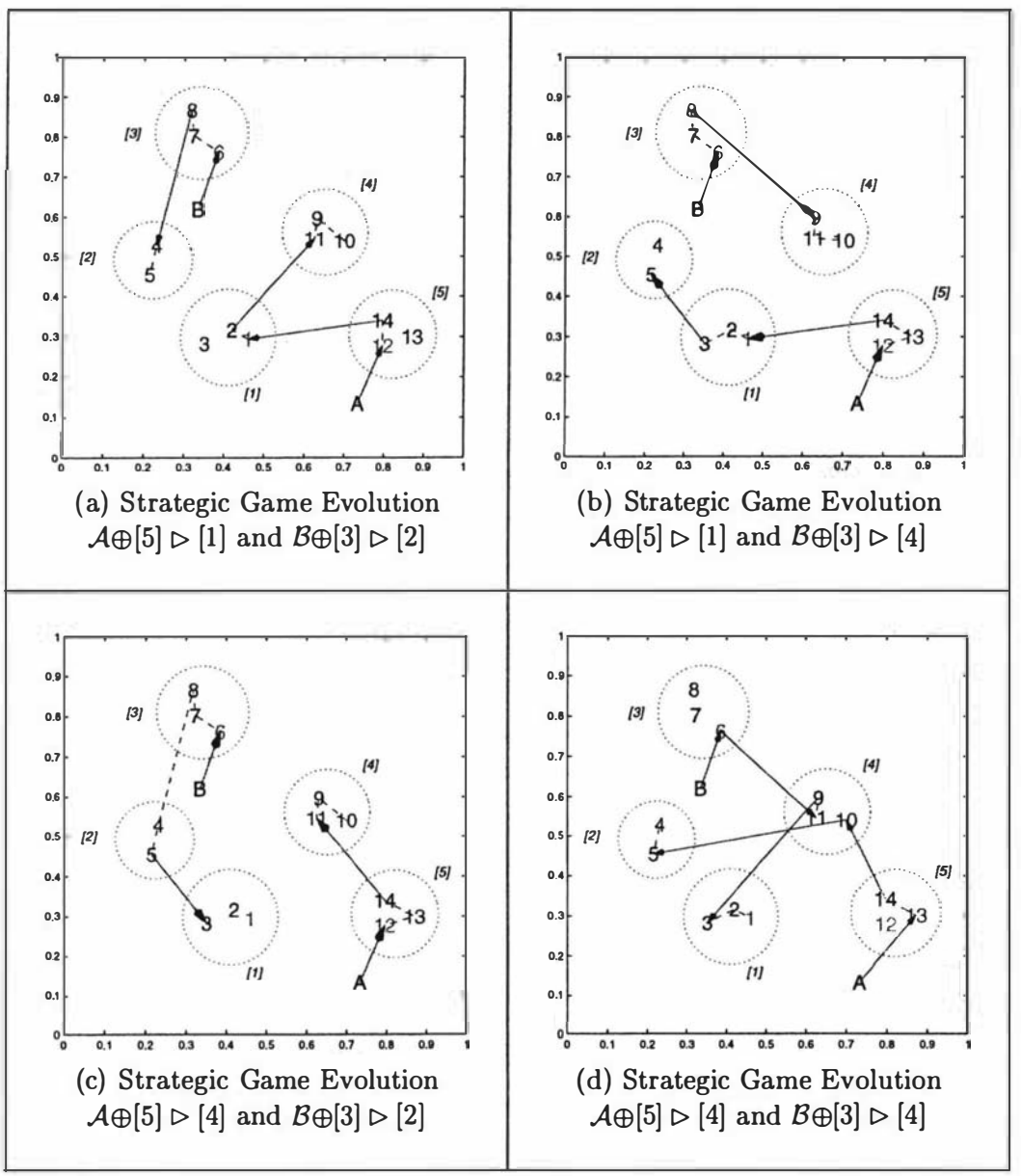
	[1]	[2]
{[1], [2]}	**390**	418

(d) Player $\mathcal{A}$ Game Table
$\mathcal{A} \oplus [5] \rhd [4]$ and $\mathcal{B} \oplus [3] \rhd [4]$

	{[1], [2]}
[1]	**390**
[2]	362

(e) Player $\mathcal{B}$ Game Table
$\mathcal{A} \oplus [5] \rhd [4]$ and $\mathcal{B} \oplus [3] \rhd [4]$

Table 7.18: Strategic Example of CLUSTER-DT Scenario $\mathcal{A} \triangleright [5]$ and $\mathcal{B} \triangleright [3]$ (continued)



(a) Strategic Game Evolution $\mathcal{A} \oplus [5] \triangleright [1]$ and $\mathcal{B} \oplus [3] \triangleright [2]$

(b) Strategic Game Evolution $\mathcal{A} \oplus [5] \triangleright [1]$ and $\mathcal{B} \oplus [3] \triangleright [4]$

(c) Strategic Game Evolution $\mathcal{A} \oplus [5] \triangleright [4]$ and $\mathcal{B} \oplus [3] \triangleright [2]$

(d) Strategic Game Evolution $\mathcal{A} \oplus [5] \triangleright [4]$ and $\mathcal{B} \oplus [3] \triangleright [4]$

game table for player $\mathcal{B}$ for which a MINIMAX analysis implies $\mathcal{A} \triangleright [4]$ and hence $\mathcal{B} \triangleright [2]$. In summary, player $\mathcal{A}$ assumes $\mathcal{B} \triangleright [4]$ and hence selects $\mathcal{A} \triangleright [5]$ (thus avoiding unprofitable tactical conflict), whilst player $\mathcal{B}$ assumes $\mathcal{A} \triangleright [4]$ and hence selects $\mathcal{B} \triangleright [2]$ (also attempting to avoid unprofitable tactical conflict). These strategic differences come about since both game tables are evaluated using MINIMAX. If MAXIMIN is applied then $\mathcal{A} \triangleright [4]$ and $\mathcal{B} \triangleright [4]$ which is even more confusing.

- Table 7.18(a) illustrates how player $\mathcal{A}$ anticipates the game will evolve with $\mathcal{A} \rightarrow [5] \rightarrow [1] \rightarrow [4]$ and $\mathcal{B} \rightarrow [3] \rightarrow [2]$. Table 7.18(b) illustrates how player $\mathcal{A}$ anticipates the game will evolve with $\mathcal{A} \rightarrow [5] \rightarrow [1] \rightarrow [2]$ and $\mathcal{B} \rightarrow [3] \rightarrow [4]$. Table 7.18(c) illustrates how player $\mathcal{B}$ anticipates the game will evolve with $\mathcal{A} \rightarrow [5] \rightarrow [4]$ and $\mathcal{B} \rightarrow [3] \rightarrow [2] \rightarrow [1]$.
- Finally, consider the case involving tactical conflict $\mathcal{A} \oplus [5] \triangleright [4]$ and $\mathcal{B} \oplus [3] \triangleright [4]$. Table 7.17(d) gives the corresponding game table for player $\mathcal{A}$ and Table 7.17(e) gives the corresponding game table for player $\mathcal{B}$. In both cases, an analysis implies $\mathcal{A} \triangleright \{[1], [2]\}$ and $\mathcal{B} \triangleright [1]$. Table 7.18(d) illustrates how the game will evolve with $\mathcal{A} \rightarrow [5] \rightarrow [4] \rightarrow [2]$ and $\mathcal{B} \rightarrow [3] \rightarrow [4] \rightarrow [1]$.

7.5.4.4 Discussion

To summarise the strategies determined for the example strategic problem:

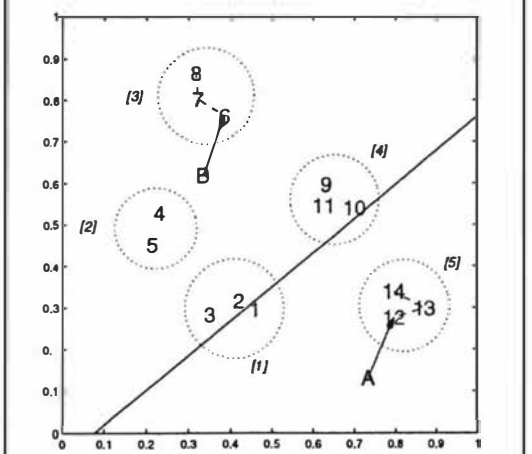
- CLUSTER-GUARANTEE suggests $\mathcal{A} \rightarrow [5]$ and $\mathcal{B} \rightarrow [4] \rightarrow [3]$.
- CLUSTER-PARANOID suggests $\mathcal{A} \rightarrow [1] \rightarrow [2]$ and $\mathcal{B} \rightarrow [4] \rightarrow [3]$.
- CLUSTER-DT suggests that there is a strategic "lead and best response" cycle: $\mathcal{B} \triangleright [2] \Rightarrow \mathcal{A} \triangleright [5] \Rightarrow \mathcal{B} \triangleright [4] \Rightarrow \mathcal{A} \triangleright [1] \Rightarrow \mathcal{B} \triangleright [2] \Rightarrow \ldots$

The good news is that the strategic example we have considered suggests that CLUSTER-DT is quite effective at analysing the strategic interactions of the players, right through to completion of the end-game. Also, CLUSTER-DT appears to be more effective for situations in which a response is required, albeit with some possibility of cycling.

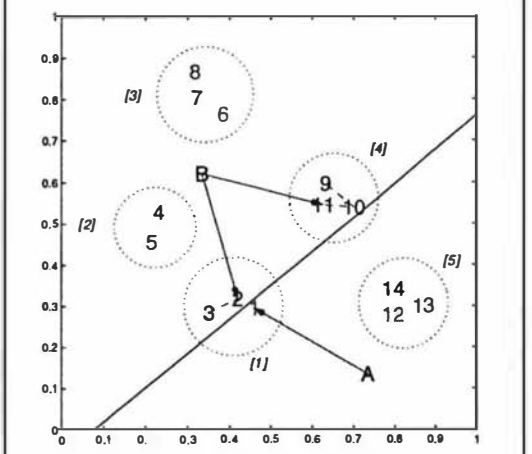
▼ Strategic and Tactical Conflict

We may define **conflict** as direct competition over a specific set of prizes in which the players' maximal PRIZE-GUARANTEE subpaths over that prize set do not exhaust that set of prizes. Individually, scenarios may be classified as *conflict* or *non-conflict* and conflicts as either *strategic* or *tactical*. Table 7.19(a) illustrates a non-conflict (of strategic independence) scenario. These arise when each player dominates (if not guarantees) their initial targets. A strategic conflict (Table 7.19(b)) is essentially a conflict over a set of clusters in which clusters can be considered as a whole, whereas a tactical conflict (Table 7.19(c)) is a conflict where, predominantly, sequences of individual prizes need to be considered.

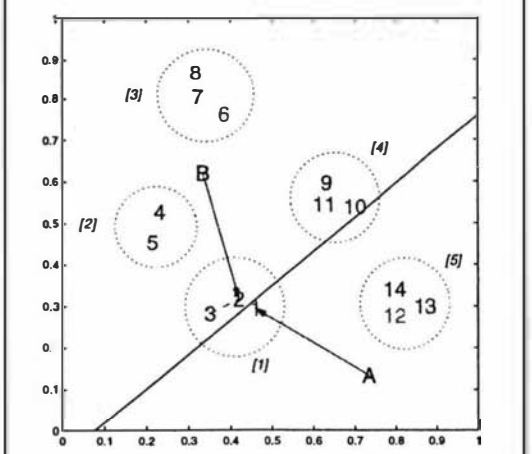
Table 7.19: Examples of Strategic Concepts



(a) Strategic Independence



(b) Strategic Conflict



(c) Tactical Conflict

7.5.5 Summary

We have designed the CLUSTER-DT strategic engine and compared the strategies determined by CLUSTER-DT with with those determined by CLUSTER-GUARANTEE and CLUSTER-PARANOID on the same illustrative example. It only remains to design the Medium-DMS, with which we then have a full three-frame implementation of the SPA/DMS.

7.6 Dynamic Monitoring: Medium-DMS

Chapter 4 introduced the idea of a Dynamic Monitoring System (DMS) and Section 6.4 presented a Small DMS for small problems. The strategic engines designed in this chapter, CLUSTER-DT and CLUSTER-PARANOID, are now incorporated into a Medium DMS for medium problems. The design of a CLUSTER-MONITOR is governed the principle: observe as much as possible but assume as little as possible.

7.6.1 System of Frames and Monitors

Medium problems are assumed to exhibit some natural cluster-like structure. We begin the design of the Medium DMS by defining the frames and monitors in terms of decision problems involving clusters, prizes and steps.

7.6.1.1 Frames

The CLUSTER-DT strategic engine provides strategic information about matching cluster targeting scenarios of the players. The PRIZE-DT tactical engine provides tactical information about matching prize targeting scenarios of the players. The Medium DMS is designed to couple together the sequential decisions in terms of clusters, in terms of the prizes within those clusters, and in terms of individual steps.

Recall that the **global-frame** has infinite planning horizon, full scope, no directives and the original objective to maximize the total value in prizes collected. No special structure is assumed of the prizes at the global-frame. The decision problem constituting the **cluster-frame** is formulated in terms of a family-cluster structure. Given a family-cluster structure, observe or strategically predict which clusters the opponent will target and hence strategically determine a sequence of clusters to target. The resulting decision is encapsulated as a **prize-frame** at which the planning horizon and scope are formulated in terms of the prizes constituting the clusters each player is predicted or determined to target. The remainder of the decision hierarchy is the same as the Small DMS, with the resulting decision from the prize-frame encapsulated as a **step-frame**. The global, cluster, prize and step frames are nested as in Figure 7.13.

Although the planning horizon, scope and directive of each frame is dynamic, the Medium DMS is fixed with these four frames. In general, the strategic framework may have an arbitrary and dynamic number of frames throughout.

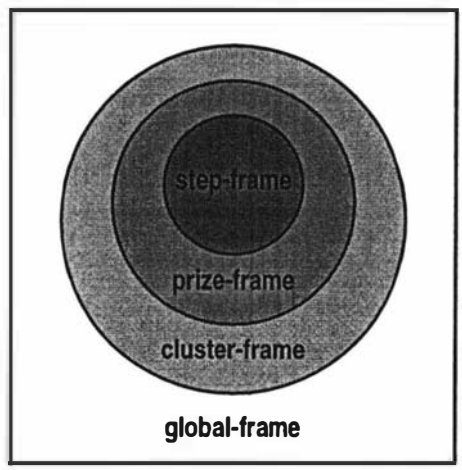


Figure 7.13: Nested Frames for Medium DMS

7.6.1.2 Monitors

One *dynamic monitor* operates at each frame. Figure 7.14 illustrates the relationship between the the monitors and the frames in the Medium DMS.

- At the *global-frame* we apply a GLOBAL-MONITOR that determines a *cluster-frame*. A **cluster-frame** is simply a family-cluster structure imposed on a subset of the prizes. The scope is reduced by thinning insignificant prizes and the directive is imposed to use the family-cluster structure constructed. Any method for constructing a family-cluster structure (such as those of Section 7.1) constitutes a GLOBAL-MONITOR.

- At the *cluster-frame* we apply a CLUSTER-MONITOR (incorporating a strategic engine) which determines a *prize-frame*. The planning horizon and scope are reduced to targeting those prizes in a few strategically selected clusters. The directive is that the family-cluster structure must be following in a strategically planned sequence. The strategic engines employed are CLUSTER-DT and CLUSTER-PARANOID.

- At the *prize-frame* we apply a PRIZE-MONITOR (incorporating a tactical engine) which determines a *step-frame*. Recall from Section 6.4 that a step-frame is a *prize-ts* plus either a *target-prize* or a *target-pair*. The Medium-DMS PRIZE-MONITOR employs either FAMILY-PRIZE-DT or FAMILY-PRIZE-GUARANTEE whereas PRIZE-DT and PRIZE-GUARANTEE were used in the Small DMS.

- Lastly, at the *step-frame* we apply a STEP-MONITOR which determines a *step*.

7.6.2 Cluster Target Sets

The decision problem of a cluster-frame is to prescribe a prize-frame. In the design of the Small DMS in Section 6.4 the idea of a *prize-ts* (prize target set) was central. It would appear

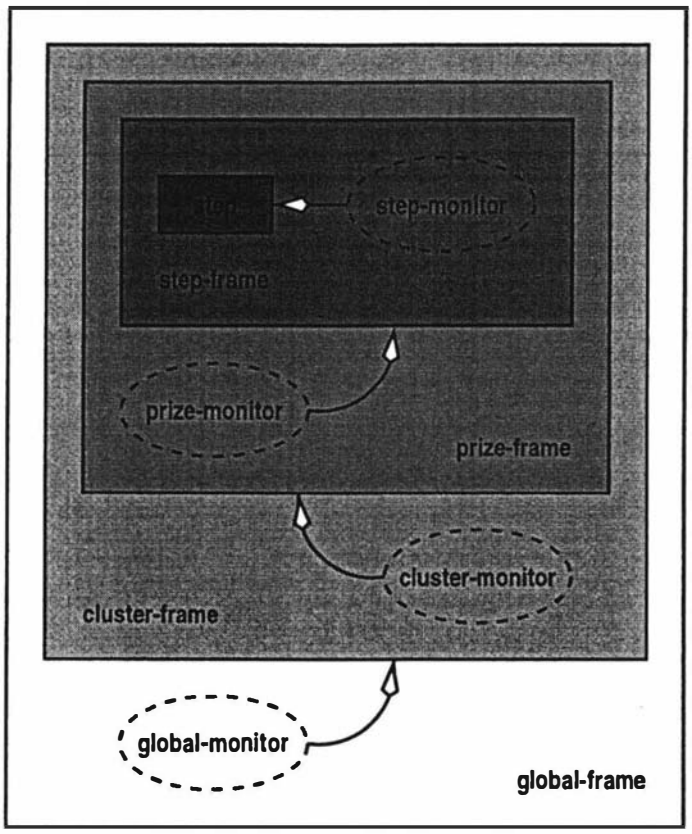


Figure 7.14: Monitors and Frames for Medium DMS

to straightforward to directly generalize the components and methods designed for the PRIZE-MONITOR. However, the matter is complicated by the look-through/move-through paradigm. We wish to determine a (primary) target-cluster or target-cluster-pair. Moreover, we also wish to know which (secondary) cluster to target next and, preferably, similar cluster targeting of the opponent.

7.6.2.1 Primary Cluster Target Sets

We begin with the task of a generalizing the concept of a *prize-ts*.

Definition 7.6.1

A **primary-cluster-target-set** (or **primary-ts**) is a subset of clusters which estimates the possible target clusters of the opponent which are not sufficiently distinguishable by observation or prediction of the opponent's movements. □

A *primary-ts* reflects the current best estimate of which cluster the opponent is currently targeting. A singleton *primary-ts* reflects that we are very confident of which specific cluster that is. If the opponent is located 'within' a cluster, then this cluster should be an element of the *primary-ts*. The cluster that contains the last prize claimed is deemed to be the cluster which the opponent is 'within'.

We can use a strategic engine to construct a *primary-ts* or we can use *observation*. The *prize-ts* build, refine and check procedures for the PRIZE-MONITOR can simply be generalized to a *primary-ts* as follows.

Build primary-ts. This is used to construct a new *primary-ts*.

BUILD-CLUSTERTS-ANGLE: Build a *prize-ts* and let the *primary-ts* be the set of clusters each of which has at least one prize in the *prize-ts*.

Estimating the cluster-sequence insertion cost of a cluster is reasonably expensive, since it involves calculating an intra-cluster route through the prizes, and hence we do not consider the insertion of a cluster into a sequence of clusters.

Refine primary-ts. Refine the current *primary-ts* by discarding those clusters no longer considered likely targets of the opponent.

REFINE-CLUSTERTS-ANGLE: Determine a *prize-ts* and discard those from the current *primary-ts* clusters no longer with a representative prize in the *prize-ts*.

Check primary-ts. Check the current *primary-ts* to determine if new clusters should be added or the entire *primary-ts* discarded. Are the opponent's observed movements still consistent with the current *primary-ts*?

CHECK-CLUSTERTS: Suppose we have estimated the bearing, ψ_B, of player $\mathcal{B}$. Determine if ψ_B is pointing towards the current *primary-ts*, i.e., if $\exists$ two prizes from the *primary-ts* which have angular gap less than 180° through which ψ_B passes.

7.6.2.2 Secondary Cluster Target Sets

The *primary-ts* models the observed or predicted primary target of the opponent. We can apply methods akin to the Small DMS PRIZE-MONITOR to select a *primary $\mathcal{A}$-target-cluster* or *primary $\mathcal{A}$-target-cluster-pair* relative to the *primary-ts*.

The modelling and design complications are about whether to (or how to) determine a secondary $\mathcal{A}$-target or a secondary cluster target set for player $\mathcal{B}$. The two guiding principles are that a prize-frame should consist of as much accurate information, and as few assumptions as possible, and that the players retain similar planning horizons.

If the *primary-ts* is a singleton, then the task is reasonably straightforward since we can repeat the primary exercise from the initial scenario where player $\mathcal{A}$ looks through the primary $\mathcal{A}$-target and player $\mathcal{B}$ looks through the singleton *primary-ts*. The definition of a *secondary-ts* follows this idea.

Definition 7.6.2

A **secondary-cluster-target-set** (or **secondary-ts**) is a set of clusters, not in the *primary-ts*, which predicts the possible secondary target clusters of the opponent following the *primary-ts*. □

The *secondary-ts* is *not* observation-based and, thus, a strategic engine must be employed for its construction.

If, however, the *primary-ts* is not a singleton, then on which primary $\mathcal{B}$-target do we base the *secondary-ts*? We could maintain a *secondary-ts* for each cluster in the *primary-ts*, but this would be inordinately expensive. We must determine just one secondary $\mathcal{A}$-target; how would the compromises in prediction be reconciled? Regardless, the facilities available in FAMILY-PRIZE-DT cannot implement this scheme. Alternatively we could highlight a specific primary $\mathcal{B}$-target from the *primary-ts* upon which to base a *secondary-ts*. This is a strong assumption to hold player $\mathcal{B}$ to at a prize-frame, but it may be useful for constructing a possible secondary $\mathcal{A}$-target.

The model we *do* adopt is to have **no** *secondary-ts* when the *primary-ts* is not a singleton. Any more complicated models would involve extensions to the capabilities of FAMILY-PRIZE-DT.

A *secondary-ts* is never checked or refined, only rebuilt, since it can never be observed, only predicted. Since the *secondary-ts* is a package with a non-singleton *primary-ts*, we update the whole package, never just the *secondary-ts*.

7.6.2.3 Prize Frame

We now have all the elements necessary to define a prize-frame. This defines a prize-frame in terms of a shorter planning horizon, scope of a few clusters, directive to follow some family-cluster sequence and surrogate to maximize prize value over this horizon.

Definition 7.6.3

A **prize frame** is a specification of which family-cluster sequence to follow and a prediction of which sets of clusters the opponent will target. A prize-frame consists of a *primary-ts* and a *primary-target-cluster-pair* or a *primary-target-cluster* (in which case a *secondary-target-cluster*

Algorithm 7.6 monitor MEDIUMDMS-GLOBAL-MONITOR

```
if ∄ cluster-frame then
    Build a family-cluster structure.
else
    // A prize has just been claimed.
    CHECK-FAMILY-CLUSTER
    if family-cluster structure invalid then
        Build a family-cluster structure.
    else
        // No changes made to family-cluster structure.
    end
end

end
```

or *secondary-target-cluster-pair* may also be specified). A *secondary-ts* is required if and only if the *primary-ts* is a singleton. □

This definition of a prize-frame is independent of the particular PRIZE-MONITOR, and tactical engine in particular, which will implement the strategic targeting. In practice, we are restricted by the services provided by a tactical engine as to what can be specified as a prize-frame.

The family-cluster structure for CLUSTER-DT and CLUSTER-PARANOID has no family-sequence constraint. All the clusters (and cluster-pairs) are available as primary and secondary cluster-targets, since only at the third recursion can the family-no-return rule exclude clusters from consideration.

7.6.3 Medium DMS

Algorithms 7.6–7.9, Algorithm 6.12 (renamed) MEDIUMDMS-PRIZE-REFINE and Algorithm 6.13 (renamed) MEDIUMDMS-STEP-MONITOR together present a generic Medium DMS.

A practical consideration in this design is that we do not wish to employ the strategic engine or the tactical engine unnecessarily, since we assume that these are the most computationally expensive procedures in the DMS. At each iteration, if the existing *prize-ts* is still valid, then it is refined, but if the *prize-ts* is not valid (the step-frame is out of scope), or has been refined to null then the corresponding step-frame is rebuilt by firstly checking the validity of the current prize-frame. If the *prize-ts* is refined, then the step-frame is updated, but the prize-frame is not checked. If the existing *primary-ts* is still valid, then it is refined, but if the *primary-ts* is not valid (the prize-frame is out of scope), or has been refined to null, then the corresponding prize-frame is rebuilt. Note that a *prize-ts* need not necessarily be a subset of the *primary-ts*.

7.6.3.1 Dynamic Monitoring

Aggregation is not arbitrary, nor just based upon saving computational complexity, but must be of some *strategic* or *tactical* value. There is always a tradeoff between aggregation (efficiency) and

Algorithm 7.7 monitor MEDIUMDMS-CLUSTER-MONITOR

```
// Check if cluster-frame exists and is still valid.
if ∄ cluster-frame or prize just claimed then
    // Check cluster-frame and possibly rebuild.
    MEDIUMDMS-GLOBAL-MONITOR
end

// Build new prize-frame.
if ∄ prize-frame or prize just claimed then
    Build specialist primary-mts.
    Select primary-ts from primary-mts.
else
    Build generic primary-ts
end
// Determine strategic response.
Determine primary-target-cluster or primary-target-cluster-pair.
Determine secondary-ts and secondary-target as appropriate.

end
```

Algorithm 7.8 monitor MEDIUMDMS-CLUSTER-REFINE

```
// Refine cluster-ts.
REFINE-CLUSTERTS
if ∄ primary-ts then
    // Build new prize-frame.
    MEDIUMDMS-CLUSTER-MONITOR
else if change made to cluster-ts then
    // Update primary strategic response.
    Determine primary-target-cluster or primary-target-cluster-pair.
    Determine secondary-ts and secondary-target as appropriate.
else
    // No changes made to prize-frame.
end

end
```

Algorithm 7.9 monitor MEDIUMDMS-PRIZE-MONITOR

```
    // Check if prize-frame exists and is still valid.
    if ∄ prize-frame or prize just claimed then
        // Build new prize-frame.
        MEDIUMDMS-CLUSTER-MONITOR
    else
        CHECK-CLUSTERTS
        if primary-ts invalid then
            // Build new prize-frame.
            MEDIUMDMS-CLUSTER-MONITOR
        else
            // Refine current prize-frame.
            MEDIUMDMS-CLUSTER-REFINE
        end
    end

    // Build new step-frame.
    if ∄ step-frame or prize just claimed then
        Build specialist prize-mts.
        Select prize-ts from prize-mts.
    else
        Build generic prize-ts
    end
    Determine target-prize or target-pair.

end
```

usefulness (effectiveness). A monitor must endeavour to dynamically map the natural structure at a level of aggregation from individual families—through clusters with family-no-return, individual clusters, prizes with family-no-return (and cluster-no-return), and prizes with cluster-no-return only—to individual prizes, to maximize the effectiveness of information which can be efficiently calculated for the number and structure of prizes remaining.

A clustering need not be static, indeed as prizes are claimed by the players the natural structure of the prize set will change and hence our clustering should reflect this. The procedure CHECK-FAMILY-CLUSTER (required in Algorithm 7.6) determines that we re-cluster whenever the last prize of an existing cluster is claimed. At what stage of the play of a game are there sufficiently few prizes that we can consider the problem a small problem, drop the global-frame and cluster-frame and take up the Small DMS? The procedure CHECK-FAMILY-CLUSTER triggers the construction of a family-cluster structure with a single cluster at a critical number of prizes and once this occurs, MEDIUMDMS-PRIZE-MONITOR reverts to SMALLDMS-PRIZE-MONITOR.

We can now describe two specific cluster monitors: CLUSTER-PARANOID-MONITOR and CLUSTER-DT-MONITOR. Note that either of these can function with a FAMILY-PRIZE-DT or FAMILY-PRIZE-GUARANTEE *submonitor*.

7.6.3.2 Medium Monitor: CLUSTER-DT CLUSTER-MONITOR

We define **cluster-multiple-target-sets (cluster-mts)** analogously to a *prize-mts*. The following modules are straightforward adaptations of the PRIZE-DT PRIZE-MONITOR equivalents in Section 6.4.3.2. These modules are applicable both to primary and secondary (generically 'cluster') constructions and selections.

module: select cluster-ts

module: build cluster-mts

module: determine target-cluster or target-cluster-pair

We can now specifically define a *prize-frame* (and a *secondary-ts* if required) using the CLUSTER-DT strategic engine. Let $\mathcal{T}_1$ be a *primary-ts*, constructed via observation (as in Section 7.6.2.1) or via prediction (as in Section 7.6.3.2). If $|\mathcal{T}_1| = 1$, then we are very confident that we know exactly which cluster the opponent is currently targeting.

▼ Case I. $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd \mathcal{T}_1$ where $\mathcal{T}_1 = \{[cy_0]\}$

Consider the full game table corresponding to $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$.

- If an $\mathcal{A}$-direct-cluster-lead, $[cx_1]$, has best evaluation, then define the prize-frame as $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}$ and $\mathcal{B} \rightarrow \{[cy_0]\} \rightarrow \{[cx_0], [cx_1]\} \setminus \{[cy_0]\}$. No *secondary-ts* is necessary since the prize-frame planning horizon of the players is similar.
- If a $\mathcal{B}$-direct-cluster-lead, $[cy_1]$, has best evaluation, then define the prize-frame as $\mathcal{A} \rightarrow \{[cx_0]\}$ and $\mathcal{B} \rightarrow \{[cy_0]\} \rightarrow \{[cy_1]\} \rightarrow \{[cx_0]\} \setminus \{[cy_1]\}$. The *secondary-ts* is then $\mathcal{T}_2 = \{[cy_1]\}$.
- Otherwise

- Construct a *secondary-ts*, $\mathcal{T}_2$, from the game table, using the build cluster-mts and select cluster-ts modules but ignoring (wlog) any $\mathcal{B}$-direct-cluster-leads.
- Use the **module** to select a secondary $\mathcal{A}$-target, x_{sec}, relative to $\mathcal{T}_2$, where x_{sec} is either a cluster-lead $[cx_1]$ or a cluster-pair $\{[cx_1],[cx_2]\}$.
  * If any $\mathcal{A}$-direct-cluster-lead has a better evaluation than x_{sec} relative to $\mathcal{T}_2$, then find the best $\mathcal{A}$-direct-cluster-lead, $[cx_1]$, and define the prize-frame as $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cx_0],[cx_1]\}\setminus\{[cy_0]\}$. Since the secondary $\mathcal{A}$-target is a direct-cluster-lead, holding player $\mathcal{B}$ to $\mathcal{T}_2$ would make the prize-frame planning horizon disproportionately long for player $\mathcal{B}$.
  * Otherwise, define the prize-frame as $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow x_{sec}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\mathcal{T}_2\rightarrow(\{[cx_0]\}\cup x_{sec})\setminus(\{[cy_0]\cup\mathcal{T}_2)$.

▼ Case II. $\mathcal{A}\rhd\{[cx_1],[cx_2]\}$ and $\mathcal{B}\rhd\mathcal{T}_1$ where $\mathcal{T}_1=\{[cy_0]\}$

(a). If $[cy_0]\in\{[cx_1],[cx_2]\}$ (say $[cy_0]=[cx_1]$) then

- We can assume that it is safe for player $\mathcal{A}$ to target cluster $[cx_2]$.
- Hence, do the same as in Case I with $[cx_0]\leftarrow[cx_2]$ and $[cy_0]\leftarrow[cx_1]$.

(b). If $[cy_0]\notin\{[cx_1],[cx_2]\}$ then

- Construct a *secondary-ts*, $\mathcal{T}_2$, from the game table corresponding to $\mathcal{A}\rhd(\mathbb{W}_0\rightarrow\{[cx_1],[cx_2]\})$ and $\mathcal{B}\rhd[cy_0]$ where $\mathbb{W}_0$ is the (necessarily cluster-window-feasible) scenario $\mathcal{A}\rhd\{[cx_1],[cx_2]\}$ and $\mathcal{B}\rhd[cy_0]$.
- We cannot assume that, if $[cx_2]\notin\mathcal{T}_2$, then it is safe for player $\mathcal{A}$ to target $[cx_2]$, since player $\mathcal{B}$ may yet target $[cx_2]$ following one of the cluster in $\mathcal{T}_2$. We need more refinement of $\mathcal{T}_2$ before choosing between clusters $[cx_1]$ and $[cx_2]$. Player $\mathcal{A}$ should first work on being able to establish something from at least one of the clusters, then that cluster will become a cluster-lead.
- Hence, define the prize-frame as $\mathcal{A}\rightarrow\{[cx_1],[cx_2]\}$ and $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\mathcal{T}_2\rightarrow\{[cx_1],[cx_2]\}\setminus\mathcal{T}_2$.

▼ Case III. $\mathcal{A}\rhd[cx_0]$ and $\mathcal{B}\rhd\mathcal{T}_1$ where $|\mathcal{T}_1|\geq 2$

Since $|\mathcal{T}_1|\geq 2$, there is no "official" *secondary-ts*. However, we wish to select a secondary $\mathcal{A}$-target. This involves making an assumption about player $\mathcal{B}$'s primary $\mathcal{B}$-target and constructing an "unofficial" *secondary-ts*, but *not* incorporating the *secondary-ts* into the prize-frame.

(a). If $[cx_0]\in\mathcal{T}_1$ then

- Assume that $\mathcal{B}\rhd[cx_0]$ for the purpose of selecting a secondary $\mathcal{A}$-target.
- Since both players look through cluster $[cx_0]$, there can be no direct-leads.
- Construct a *secondary-ts*, $\mathcal{T}_2$, from the game table corresponding to $\mathcal{A}\rhd[cx_0]$ and $\mathcal{B}\rhd[cx_0]$.

- Select a secondary $\mathcal{A}$-target, x_{sec}, relative to $\mathcal{T}_2$.
- Define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}x_{sec}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}x_{sec} \setminus \mathcal{T}_1$. Note that we use $\mathcal{T}_2$ only to determine x_{sec} and we *don't* use $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\mathcal{T}_2{\rightarrow}x_{sec} \setminus (\mathcal{T}_1 \cup \mathcal{T}_2)$.

(b). If $[cx_0] \notin \mathcal{T}_1$ then

- From the row $\mathcal{A} \rhd [cx_0]$ of the game table corresponding to $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \emptyset$, choose a minimum entry, y_{prim}, amongst the options which overlap $\mathcal{T}_1$, where y_{prim} is either a cluster-lead $[cy_1]$ or a cluster-pair $\{[cy_1],[cy_2]\}$.
- Consider the game table corresponding to $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd y_{prim}$.
- If y_{prim} is the cluster-lead $[cy_1]$ then
  - If an $\mathcal{A}$-direct-cluster-lead, $[cx_1]$, has best evaluation, then define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\{[cx_0],[cx_1]\} \setminus \mathcal{T}_1$.
  - If a $\mathcal{B}$-direct-cluster-lead has best evaluation, then define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\{[cx_0]\}$.
  - Otherwise
    - Construct an "unofficial" *secondary-ts*, $\mathcal{T}_2$, from the table, ignoring any $\mathcal{B}$-direct-cluster-leads.
    - Select a secondary $\mathcal{A}$-target, x_{sec}, relative to $\mathcal{T}_2$.
    - If any $\mathcal{A}$-direct-cluster-lead has a better evaluation than x_{sec} relative to $\mathcal{T}_2$, then find the best $\mathcal{A}$-direct-cluster-lead, $[cx_1]$, and define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\{[cx_0],[cx_1]\} \setminus \mathcal{T}_1$.
    - Otherwise, define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}x_{sec}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}(\{[cx_0]\} \cup x_{sec}) \setminus \mathcal{T}_1$.
- Otherwise, y_{prim} is the cluster-pair $\{[cy_1],[cy_2]\}$.
  - Select the best $\mathcal{A}$-cluster-lead, $[cx_1]$.
  - Define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\{[cx_0],[cx_1]\} \setminus \mathcal{T}_1$.

▼ Case IV. $\mathcal{A} \rhd \{[cx_1],[cx_2]\}$ and $\mathcal{B} \rhd \mathcal{T}_1$ where $|\mathcal{T}_1| \geq 2$

We cannot assume that, if $[cx_2] \notin \mathcal{T}_1$, then it is safe for player $\mathcal{A}$ to target $[cx_2]$, since player $\mathcal{B}$ may yet target $[cx_2]$ following one of the cluster in $\mathcal{T}_1$. We need more refinement of $\mathcal{T}_2$ before choosing between clusters $[cx_1]$ and $[cx_2]$. Player $\mathcal{A}$ should first attempt to establish something from at least one of the clusters, then that cluster will become a cluster-lead.

- Define the prize-frame as $\mathcal{A}{\rightarrow}\{[cx_1],[cx_2]\}$ and $\mathcal{B}{\rightarrow}\mathcal{T}_1{\rightarrow}\{[cx_1],[cx_2]\} \setminus \mathcal{T}_1$.

▼ Prize Frame, Final-Family and ANY-ALL-PCTSP Requirements

At the prize-frame, the final-family requirement is applied, as is the existing ANY–ALL–PCTSP requirement for each cluster. The principle here is that CLUSTER-DT provides strategic targeting and scenario information. The prize-frame is the only mechanism to show the PRIZE-MONITOR how to convert the strategic targeting to actuality. Hence, if the final-family or ANY–ALL–PCTSP requirements are dropped, then the strategic analysis may never be achieved.

7.6.3.3 Medium Monitor: CLUSTER-PARANOID CLUSTER-MONITOR

The select cluster-ts, build cluster-mts and determine target-cluster modules are straightforward adaptations of the PRIZE-PARANOID PRIZE-MONITOR equivalents in Section 6.4.3.3. These modules are applicable both to primary and secondary constructions and selections. We can now specifically define a *prize-frame* (and a *secondary-ts* if required) using the CLUSTER-PARANOID strategic engine.

▼ Case I. $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd \mathcal{T}_1$ where $\mathcal{T}_1 = \{[cy_0]\}$

Consider the full CLUSTER-PARANOID vector corresponding to initialising with $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$. Apply the modules to construct a *secondary-ts*, $\mathcal{T}_2$, and to select a secondary $\mathcal{A}$-target, $[cx_1]$.

- Define the prize-frame as $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}$ and $\mathcal{B} \rightarrow \{[cy_0]\} \rightarrow \mathcal{T}_2 \rightarrow \{[cx_0], [cx_1]\} \setminus \mathcal{T}_2$.

▼ Case II. $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd \mathcal{T}_1$ where $|\mathcal{T}_1| \geq 2$

Again, since $|\mathcal{T}_1| \geq 2$, there is no "official" *secondary-ts*.

(a). If $[cx_0] \in \mathcal{T}_1$ then

- Assume that $\mathcal{B} \rhd [cx_0]$ for the purpose of selecting a secondary $\mathcal{A}$-target.
- Construct a *secondary-ts*, $\mathcal{T}_2$, from the full CLUSTER-PARANOID vector corresponding to initialising with $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cx_0]$, and select a secondary $\mathcal{A}$-target, $[cx_1]$.
- Define the prize-frame as $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}$ and $\mathcal{B} \rightarrow \mathcal{T}_1 \rightarrow \{[cx_1]\} \setminus \mathcal{T}_1$.

(b). If $[cx_0] \notin \mathcal{T}_1$ then

- Construct a partial CLUSTER-PARANOID vector with entries corresponding to $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [ci]$ for each $[ci] \in \mathcal{T}_1$. Choose a minimum element, indexed as $\mathcal{B} \rhd [cy_0]$, and assume that $\mathcal{B} \rhd [cy_0]$ for the purpose of selecting a secondary $\mathcal{A}$-target.
- Construct a *secondary-ts*, $\mathcal{T}_2$, from the full CLUSTER-PARANOID vector corresponding to initialising with $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd [cy_0]$, and select a secondary $\mathcal{A}$-target, $[cx_1]$.
- Define the prize-frame as $\mathcal{A} \rightarrow \{[cx_0]\} \rightarrow \{[cx_1]\}$ and $\mathcal{B} \rightarrow \mathcal{T}_1 \rightarrow \{[cx_0], [cx_1]\} \setminus \mathcal{T}_1$.

7.6.4 Summary

In this section we have completed the design of the Medium-DMS, a full three-frame implementation of the SPA/DMS.

More complicated monitoring schemes would require more sophisticated services from both CLUSTER-DT and from FAMILY-PRIZE-DT or FAMILY-PRIZE-GUARANTEE. In family sequences in which there are tree structured sequences, the families which follow are dependent upon which clusters are selected from the first family. For example, we might actually like to consider the scenario $\mathcal{A} \rightarrow \{[cx_1] \rightarrow \{[cx_3, cx_4]\}, [cx_2] \rightarrow \{[cx_5]\}\}$, i.e., player $\mathcal{A}$ initially considers either $[cx_1]$ or $[cx_2]$ but then has specific family sequence depending upon which is selected; or, something similar

to primary-ts evaluation built into CLUSTER-DT to enable the selection of a compromise cluster to be selected when a player has one of a whole set of possible targets.

Coda

▼ Summary

In this chapter we have successfully scaled the tactical engines of Chapter **6** to strategic engines CLUSTER-PARANOID and CLUSTER-DT. The principal difficulties and key contributions were:

- Resolving the deadlock issue with Lemma 7.3.4 allowed us to employ the simple operating principle that a cluster is unlocked for move-through only once, either to both players to move through simultaneously or to only one player to move through, with the other missing out.
- This principle establishes that the two-cluster-lookahead can be maintained consistently throughout CLUSTER-DT. The two-cluster-lookahead was originally proposed from the intra-cluster/inter-cluster strategy/routing, but was continually derailed by the deadlock problem.
- The two-cluster-lookahead rule allows for some useful coordination of intra-cluster routing and inter-cluster routing by variation of the path through a cluster depending upon which cluster (or pair of clusters and perhaps also cluster-feasibility-window) is to be targeted next.
- The formulation of a simple description of family-cluster structure with family-sequence and final-family allowed us to compactly specify a cluster targeting scenario. The PRIZE-DT and ORIGINAL-DT tactical engines were easily modified to be able to evaluate these cluster scenarios in terms of overall value and exit-value, exit-time and exit-location from the first cluster (for move-through purposes).
- The intricacy of the Medium-DMS is warranted since it is important to determine how we should use the strategic information, although this was only possible by resolving how to operate the two-cluster-lookahead rule with a primary and secondary target set.

In conclusion, strategic planning involving clusters makes the design of an effective strategy much more technical but is able to return useful strategic information, which is much less conservative and less myopic than PRIZE-DT.

▼ Link

We have considered tactical and strategic planning at the scale of prizes and cluster respectively. However, larger problems, with no natural family-cluster structure, require a more heuristic approach. This is the subject of the next chapter.

7.A Appendices to Chapter 7

7.A.1 Random Clustering

Random clustering is equivalent to randomly allocating n objects (prizes) into containers (clusters). This appendix is concerned with two problems.

Cluster a given set of prizes. Since the prizes already have attributes (location, value), the prizes are distinguishable. However, the clusters are defined only by the prizes they are to contain and hence are indistinguishable containers. The problem is to randomly allocate distinguishable objects to indistinguishable containers. Nijenhuis and Wilf [167] provide an algorithm for the case were the number of containers is not specified, presented in Section 7.A.1.1. In Section 7.A.1.2 we adapt this algorithm to the case where exactly k non-empty containers are required.

Generation of a test problem. Randomly allocate the number of prizes for each cluster given the total number of prizes, a number of clusters and the total prize value required for each cluster. Since the clusters have an attribute (total prize value), the clusters are distinguishable. However, the prizes are indistinguishable since we are only interested in how many prizes for each cluster—the attributes of a prize are to be generated dependent upon to which cluster that prize is allocated. The problem is to randomly allocated indistinguishable prizes to distinguishable containers.

7.A.1.1 Random Partition of n Objects

A **partition** of a set V is a family of sets $T_1, T_2, \ldots, T_k$ satisfying

$$\begin{aligned} T_i \cap T_j &= \emptyset \quad (i \neq j) \\ \cup_{i=1}^{k} T_i &= V \\ T_i &\neq \emptyset \quad (i = 1, \ldots, k) \end{aligned}$$

Algorithm 7.10 procedure RANDOM PARTITION

Input: n *// number of prizes*
Output: ℓ *// number of clusters*
Output: q *// cluster membership vector*
Calculate $a_1, \ldots, a_n$.
$m \leftarrow n$
$\ell \leftarrow 0$
while $(m > 0)$ **do**
Choose $j \in \{1, \ldots, m\}$ with probability

$$Prob(j) = \begin{pmatrix} m-1 \\ j-1 \end{pmatrix} \frac{a_{m-j}}{a_m}$$

$\ell \leftarrow \ell + 1$
Assign ℓ to $q_{m-j+1}, \ldots, q_m$.
$m \leftarrow m - j$
end
Perform a random permutation on $(q_1, \ldots, q_n)$.

end

Nijenhuis and Wilf [167] give an algorithm (reproduced here as Algorithm 7.10) for constructing a random partition on the set $V = \{1, \ldots, n\}$ with the property that every such partition is equally likely. The variable ℓ is the number of cells allocated so far, m is the number of objects which remain to be allocated, and j is the number of objects chosen to be allocated to the next cell.

The numbers a_i are the **Bell numbers** denoting the number of partitions of a set of n distinguishable objects. They can be calculated using the recursion defined by $a_0 = 1$ and $\forall\, n \geq 1$

$$a_n = \sum_{k=0}^{n-1} \begin{pmatrix} m-1 \\ k-1 \end{pmatrix} a_k$$

Hence $a_1 = 1$, $a_2 = 2$, $a_3 = 5$, $a_4 = 15$, etc.

7.A.1.2 Random Partition of n objects into k nonempty cells

The **Stirling numbers of the Second kind**, $S(n, k)$, denote the number of partitions of a set of n distinguishable objects into k non-empty, indistinguishable subsets, where k is a positive integer and $n \geq k$.

Hillman, Alexanderson and Grassl [98] state that the number of allocations of n distinguishable objects to k *distinguishable* containers (after which each container ends up non-empty) is $k!S(n, k)$. Tables 7.20 and 7.21 provide an example for the case were $n = 6$ and $k = 3$.

In general, let j be the size of the first container. Then j must be at least 1 and at most $n + 1 - k$ (in which case all the other containers will have exactly one object each). There will be $\begin{pmatrix} n \\ j \end{pmatrix}$ choices for the j objects to fill the first container. For each of these choices there will remain $n - j$ objects to allocate to $k - 1$ containers, of which there are $(k-1)!S(n-j, k-1)$

Arrangement	Frequency			
$4+1+1$	$\binom{6}{4}$	$\binom{2}{1}$	$=$	30
$3+2+1$	$\binom{6}{3}$	$\binom{3}{2}$	$=$	60
$3+1+2$	$\binom{6}{3}$	$\binom{3}{1}$	$=$	60
$2+3+1$	$\binom{6}{2}$	$\binom{4}{3}$	$=$	60
$2+2+2$	$\binom{6}{2}$	$\binom{4}{2}$	$=$	90
$2+1+3$	$\binom{6}{2}$	$\binom{4}{1}$	$=$	60
$1+4+1$	$\binom{6}{1}$	$\binom{5}{4}$	$=$	30
$1+3+2$	$\binom{6}{1}$	$\binom{5}{3}$	$=$	60
$1+2+3$	$\binom{6}{1}$	$\binom{5}{2}$	$=$	60
$1+1+4$	$\binom{6}{1}$	$\binom{5}{1}$	$=$	30
total		$3!S(6,3)$	$=$	540

Table 7.20: Counting the number of distributions of six distinguishable objects into three distinguishable containers for all possible sizes of the containers.

Size of First Container	Frequency
4	30
3	120
2	210
1	180
total	540

Table 7.21: Counting the number of distributions of six distinguishable objects into three distinguishable containers for all possible sizes of the first container.

Algorithm 7.11 procedure RANDOM k-PARTITION

Input: n *// number of prizes.*
Input: k *// number of clusters.*
Output: q *// cluster membership vector*
$m \leftarrow n$
$\ell \leftarrow k$
while $(m > 0)$ **do**
Choose $j \in \{1, \ldots, m+1-\ell\}$ with probability

$$Prob(j) = \left(\begin{array}{c} m \\ j \end{array} \right) \frac{S(m-j, \ell-1)}{\ell S(m, \ell)}$$

Assign ℓ to $q_{m-j+1}, \ldots, q_m$.
$\ell \leftarrow \ell - 1$
$m \leftarrow m - j$
end
Perform a random permutation on $(q_1, \ldots, q_n)$.

end

such allocations. Hence, the number of allocations in which there are j objects allocated to the first container is

$$\left(\begin{array}{c} n \\ j \end{array} \right) (k-1)! S(n-j, k-1)$$

For example, taking $n = 6, k = 3, j = 4$ gives $\left(\begin{array}{c} 6 \\ 4 \end{array} \right) (6-4)! S(2,2) = 30$ as required. Hence, the relative frequency that the size of the first container is $j \in \{1, \ldots, n+1-k\}$ is given by

$$\frac{\left(\begin{array}{c} n \\ j \end{array} \right) (k-1)! S(n-j, k-1)}{k! S(n,k)} = \left(\begin{array}{c} n \\ j \end{array} \right) \frac{S(n-j, k-1)}{k S(n,k)}$$

Algorithm 7.11 constructs a random allocation of n distinguishable objects to k non-empty, distinguishable containers such that all possible allocations are equally likely. The variable ℓ is the number of containers remaining to be allocated, m is the number of objects which remain to be allocated, and j is the number of objects chosen to be allocated to the next container. By subsequently sorting the containers by the least index object in each container we automatically have an algorithm for constructing a random partition of n distinguishable objects into k non-empty, *indistinguishable* subsets such that all such partitions are equally likely.

7.A.1.3 Random Cluster Sizes

Hillman, Alexanderson and Grassl [98, page 260] show that the number of allocations of n indistinguishable objects into k distinguishable non-empty containers is

$$\left(\begin{array}{c} n-1 \\ k-1 \end{array} \right).$$

Algorithm 7.12 procedure RANDOM CLUSTER SIZES

Input: n *// number of prizes.*
Input: k *// number of clusters.*
Output: q *// vector of cluster sizes.*
$m \leftarrow n$
$i \leftarrow k$
while $(i \geq 2)$ **do**
Choose $q_i \in \{1, \ldots, m+1-i\}$ with probability

$$P(q_i = j) = \frac{\binom{m-j-1}{i-2}}{\binom{m-1}{i-1}}$$

$m \leftarrow m - q_i$
$i \leftarrow i - 1$
end
$q_1 \leftarrow m$

end

Suppose we wish to determine the sizes, q_k, of k distinguishable non-empty clusters such that every partition of the n currently indistinguishable prizes is equally likely. The reason that the clusters are distinguishable is that we wish to be able to assign a value first to the cluster and then allocate the cluster value between the prizes in the cluster. Then we simply sample $q_k \in \{1, \ldots, n+1-k\}$ with probability

$$P(q_k = j) = \frac{\binom{n-j-1}{k-2}}{\binom{n-1}{k-1}}$$

and continue as in Algorithm 7.12 RANDOM CLUSTER SIZES.

7.A.2 Family-Cluster Precedence Subproblems

This appendix presents algorithmic details for solving the following five game tree bounding and pruning subproblems from Section 7.2.2.3.

POSSIBLE. Does there exist a path for player $\mathcal{X}$ which satisfies the family-sequence, cluster-no-return, family-no-return and ANY-ALL-PCTSP requirements of player $\mathcal{X}$, and arrives at some prize from the final family in the family-sequence of player $\mathcal{X}$ no later than the global overall deadline λ?

COOPERATE. Determine the **cooperative-value** $\Omega(j)$ of game tree node j. This is the maximum joint value of the remaining prizes that the players could collect if both players cooperate perfectly. Players must satisfy their family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements and the global overall deadline λ.

If $\lambda = \infty$ then $\Omega(j)$ is the value sum of the remaining prizes in clusters, either: available to player $\mathcal{A}$ and belonging to an available player $\mathcal{A}$ family; or, available to player $\mathcal{B}$ and belonging to an available player $\mathcal{B}$ family.

GUARANTEE. Determine the **guarantee-value** $\Gamma(j, \mathcal{X})$ of player $\mathcal{X}$ at a game tree node j. This is the maximum value of a guaranteed subpath through the remaining prizes satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements of player $\mathcal{X}$ and the global overall deadline λ.

There are two versions of GUARANTEE depending upon whether or not player $\mathcal{X}$ must claim a prize from the final family of its family-sequence. To implement a solver for GUARANTEE we propose and solve the DISTANCE subproblem.

DISTANCE. Determine the earliest possible arrival-time of player $\mathcal{Y}$ at each remaining prize in a cluster available to player $\mathcal{X}$ in a player $\mathcal{X}$ future-family satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements of player $\mathcal{Y}$.

FEASIBLE. Determine if there exists a guaranteed subpath through the remaining prizes satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL-PCTSP requirements of player $\mathcal{X}$, the deadlines ℓ_j and λ and claiming a prize from the $\mathcal{X}$-final-family.

Two types of algorithm are considered in this appendix: the depth-first-search (DFS) branch-and-bound algorithm and the dynamic programming (DP) algorithm (see, e.g., Denardo [52]). Also, COOPERATE, GUARANTEE, FEASIBLE are prize-based algorithms, POSSIBLE is cluster-based and DISTANCE is a combination of prize-based and cluster-based.

7.A.2.1 Two-Optimal Appendage Revisited

We extend the argument regarding *two-optimal-appendages* from Appendix 6.A.2.1 to B&B algorithms involving multiple clusters. Let Ψ_X be the working subpath through prizes from Q and let t_X be the projected arrival-time at the end of Ψ_X.

Suppose $|\Psi_X| \geq 2$ and the last two prizes on Ψ_X are from cluster $[cx]$. We propose to append prize $j \in (Q \cap [cx]) \setminus \Psi_X$ to subpath Ψ_X. If there is a prize on Ψ_X which is not from cluster

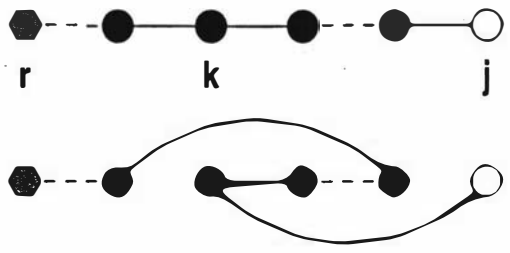


Figure 7.15: Subpath appendage two-exchange

$[cx]$, then let r denote the last such prize on Ψ_X; otherwise, let r be the origin of the subpath Ψ_X. Choose $k \in \Psi_X \cap [cx]$ such that k is not the last prize on Ψ_X. Consider the two-exchange defined by Figure 7.15 with change in distance $\delta(j,k)$. If $|\Psi_X| \leq 1$ or either of the last two prizes on Ψ_X are not from cluster $[cx]$ or $\nexists\, k$ such that $\delta(j,k) < 0$, then $j \in (Q \cap [cx]) \setminus \Psi_X$ is a **two-optimal-appendage** of Ψ_X.

Lemma 7.A.1

We need only target those prizes $j \in Q \setminus \Psi_X$ which are two-optimal-appendages of Ψ_X.

Proof:

1. The subpath of Ψ_X from r onwards is path-two-optimal if and only if Ψ_X is constructed such that each prize appended after r is a two-optimal-appendage.
2. Consider the overall deadline λ. Let τ_i be the arrival time of player $\mathcal{X}$ at each prize i on Ψ_X and, if $\Psi_X \cap [cx] = \emptyset$, then let τ_r be the arrival-time of $\mathcal{X}$ at the origin of Ψ_X. Certainly $t_X \leq \lambda$ if and only if $\tau_r \leq \lambda$ and $\tau_i \leq \lambda \; \forall \; i \in \Psi_X \cap [cx]$. That is, we need only consider the arrival-time of player $\mathcal{X}$ at the end node of Ψ_X to establish a guarantee for Ψ_X with respect to the overall deadline λ.
3. Consider a set of deadlines ℓ_i on prizes $i \in Q \cap [cx]$ determined by solving the DISTANCE subproblem. Since the DISTANCE problem calculates earliest arrival-times subject to family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements of player $\mathcal{Y}$, then the *triangle inequality* holds on cluster $Q \cap [cx]$, i.e., $\ell_i \leq \ell_k + d_{ki} \; \forall \; i,k \in [cx]$.

   Suppose $\Psi_X \neq \emptyset$ and let j be the last prize on Ψ_X. Then $\tau_j \leq \ell_j$ if and only if $\tau_i \leq \ell_i \; \forall \; i \in \Psi_X \cap [cx]$ and either $\Psi_X \cap [cx] = \emptyset$ or $\tau_r \leq \ell_r$. That is, we need only consider the arrival-time of player $\mathcal{X}$ at the end node of Ψ_X to establish a guarantee for Ψ_X with respect to deadlines ℓ_i.
4. Hence, if we append prize $j \in (Q \cap [cx]) \setminus \Psi_X$ to Ψ_X, where prize j is *not* a two-optimal-appendage of Ψ_X, then somewhere else in the B&B tree there will exist a B&B tree node at which Ψ_X will contain the same prizes in some different sequence and satisfying any deadlines as required, but with an earlier t_X; this node can only give a better objective function value involving arrival-time or guaranteed value.

■

7.A.2.2 Prize-DFS: COOPERATE

In this section we modify the B&B algorithm of Appendix 6.A.2.2 to handle the cluster-no-return, family-no-return, family-sequence and ANY-ALL-PCTSP requirements.

Bounding Criteria. Let

$$\begin{aligned} S_1 &= Q_A \cap \bigcup_{\mathcal{G} \in \mathbb{Q}_A} \mathcal{G} & S_1 &= \bigcup_{[ci] \in S_1} [ci] \\ S_2 &= Q_B \cap \bigcup_{\mathcal{G} \in \mathbb{Q}_B} \mathcal{G} & S_2 &= \bigcup_{[ci] \in S_2} [ci] \end{aligned}$$

$$\begin{aligned} Q_0 &= \{j \in Q \cap S_1 \cap S_2 : \min\{t_A + d_{Aj}, t_B + d_{Bj}\} \leq \lambda\} \\ Q_1 &= \{j \in Q \cap (S_1 \setminus S_2) : t_A + d_{Aj} \leq \lambda\} \\ Q_2 &= \{j \in Q \cap (S_2 \setminus S_1) : t_B + d_{Bj} \leq \lambda\} \end{aligned}$$

Fathom B&B node i if

$$v(\Omega_A) + v(\Omega_B) + \sum_{j \in Q_0 \cup Q_1 \cup Q_2} v_j \leq v_{best} \tag{7.7}$$

Targeting. Lemma 7.A.1 shows that player $\mathcal{A}$ need only target those prizes $j \in Q \setminus (\Omega_A \cup \Omega_B)$ which are two-optimal-appendages of Ω_A and player $\mathcal{B}$ need only target those prizes $j \in Q \setminus (\Omega_A \cup \Omega_B)$ which are two-optimal-appendages of Ω_B. Hence, use Algorithm 7.2 TARGET-CLUSTERS to determine $\mathcal{S}_A$ and $\mathcal{S}_B$.

- If $\Omega_A = \emptyset$ then

$$Q_A = \{j \in (Q \setminus \Omega_B) \cap \textstyle\bigcup_{[ci] \in \mathcal{S}_A} [ci] : t_A + d_{Aj} \leq \lambda\}. \tag{7.8}$$

- If $\Omega_A \neq \emptyset$ then let $[cx_0]$ be the cluster from which $\mathcal{A}$ last claimed a prize and

$$\begin{aligned} Q_A = \{&j \in Q \setminus (\Omega_A \cup \Omega_B) : t_A + d_{Aj} \leq \lambda\} \cap \\ &\left(\{j \in [cx_0] : j \text{ is a two-optimal-appendage of } \Omega_A\} \cup \bigcup_{[ci] \in \mathcal{S}_A \setminus \{[cx_0]\}} [ci] \right). \end{aligned}$$

- If $\Omega_B = \emptyset$ then

$$Q_B = \{j \in (Q \setminus \Omega_A) \cap \textstyle\bigcup_{[ci] \in \mathcal{S}_B} [ci] : t_B + d_{bj} \leq \lambda\}. \tag{7.9}$$

- If $\Omega_B \neq \emptyset$ then let $[cy_0]$ be the cluster from which $\mathcal{B}$ last claimed a prize and

$$\begin{aligned} Q_B = \{&j \in Q \setminus (\Omega_A \cup \Omega_B) : t_B + d_{Bj} \leq \lambda\} \cap \\ &\left(\{j \in [cy_0] : j \text{ is a two-optimal-appendage of } \Omega_B\} \cup \bigcup_{[ci] \in \mathcal{S}_B \setminus \{[cy_0]\}} [ci] \right). \end{aligned}$$

Branching Rule. Use the same branching rule as per Appendix 6.A.2.2. Algorithm 7.13 FAMILY_COOP_SLAVE(i) provides the details.

Algorithm 7.13 procedure FAMILY_COOP_SLAVE(i)

Input: i // *B&B tree node.*
Determine Q_0, Q_1, Q_2, Q_A and Q_B.
$ub \leftarrow v(\Omega_A) + v(\Omega_B) + \sum_{j \in Q_0 \cup Q_1 \cup Q_2} v_j$
if $(ub > v_{best})$ **then**
$v_{best} \leftarrow \max\{v_{best}, v(\Omega_A) + v(\Omega_B)\}$
for $((x \in Q_A$ **or** $y \in Q_B)$ **and** $(v_{best} < \min\{\beta_{coop}, ub\}))$ **do**
FAMILY_COOP_SLAVE$((i, \mathcal{A} \oplus x))$ **or** FAMILY_COOP_SLAVE$((i, \mathcal{B} \oplus y))$
end
end

end

7.A.2.3 Cluster-DP: DISTANCE

Recall that the DISTANCE subproblem is defined as follows.

DISTANCE. Determine the earliest possible arrival-time of player $\mathcal{B}$ at each remaining prize in a cluster available to player $\mathcal{A}$ in a player $\mathcal{A}$ future-family satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL–PCTSP requirements of player $\mathcal{B}$.

If prize j is not available to $\mathcal{B}$ then the earliest possible arrival-time is set to ∞.

Let ℓ_j be the earliest possible arrival-time of player $\mathcal{B}$ at each prize in Q. Algorithm 7.14 DISTANCE_DP(j) presents a cluster-depth-first Dynamic Programming (DP) algorithm to determine the ℓ_j values. Where the $\mathcal{B}$-family-sequence requirement applies, a family corresponds to a *stage* of the DP and a cluster corresponds to a *state*. Let χ_i be the earliest possible exit-times from each prize $i \in Q \cap [ci]$ which satisfy the ANY-ALL-PCTSP requirement on cluster $[ci]$. Suppose that a $\mathcal{B}$-family-sequence applies and suppose that the family-sequence is $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m)$. We determine the values ℓ_j for each prize in a cluster in family $\mathcal{G}_h + 1$ according to:

$$\ell_j = \min_{i \in H}\{\chi_i + d_{ij}, \lambda\} \tag{7.10}$$

where $H = Q \cap \bigcup_{[ci] \in \mathcal{Q}_A \cap \mathcal{G}_h} [ci]$. This inter-cluster distance calculation is illustrated in Figure 7.16.

7.A.2.4 Prize-DFS: ALL-DISTANCE and PCTSP-DISTANCE

To determine intra-cluster distances from earliest entry-times τ_j to earliest exit-times χ_i we define the following two subproblems. Note that we use τ_j since these subproblems are re-used elsewhere, beyond solving the DISTANCE problem.

ALL-DISTANCE.

Determine the earliest possible exit-time from every $i \in Q \cap [ci] \neq \emptyset$, satisfying an ALL requirement on cluster $[ci]$ and given arrival-times τ_j.

PCTSP-DISTANCE.

Algorithm 7.14 procedure DISTANCE_DP(j)

Input: j // *Game tree node.*
// *Determine the earliest arrival-times for player $\mathcal{B}$ at each prize $j \in Q$.*
$\ell_j \leftarrow \infty \; \forall \; j \in Q$
if *$\mathcal{B}$-family-sequence applies* **then**
Let k be the least k such that no prize has been visited from family $\mathcal{G}_k$.
$\mathcal{H} \leftarrow \mathcal{G}_k$
else
$\mathcal{H} \leftarrow \mathcal{Q}_B \cap \bigcup_{\mathcal{G} \in \mathbb{Q}_B} \mathcal{G}$
end
if *player $\mathcal{B}$ has not yet claimed a prize* **then**
$\ell_j \leftarrow t_B + d_{Bj} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{H}} [ci]$
else
Let $y \in [cy]$ be the last prize claimed by player $\mathcal{B}$.
if ANY-ALL-PCTSP *requirement satisfied on* $[cy]$ **then**
$\ell_j \leftarrow t_B + d_{Bj} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{H}} [ci]$
else
$\chi_i \leftarrow$ Earliest possible exit-time from $i \in Q \cap [cy]$ satisfying
ALL-PCTSP requirement on cluster $[cy]$
$\ell_j \leftarrow \min_{i \in Q \cap [ci]} \{\chi_i + d_{ij}\} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{H}} [ci]$
end
end
if *$\mathcal{B}$-family-sequence applies* **then**
for $h = k$ **to** $m - 1$ **do**
$K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_{h+1}} [ci]$
for ($[ci] \in \mathcal{G}_h : [ci]$ is an ANY cluster) **do**
// *Determine exit-times χ_i.*
$\chi_i \leftarrow \ell_i \; \forall \; i \in Q \cap [ci]$
// *Update arrival-times ℓ_j on family $\mathcal{G}_{h+1}$.*
$\ell_j \leftarrow \min\{\ell_j, \min_{i \in Q \cap [ci]} \{\chi_i + d_{ij}\}\} \; \forall \; j \in K$
end
for ($[ci] \in \mathcal{G}_h : [ci]$ is an ALL-PCTSP cluster) **do**
// *Initialise χ_i significant upper bounds.*
$\chi_i \leftarrow \max_{j \in K} \{\ell_j - d_{ij}\}$
// *Determine earliest possible exit-times χ_i from every*
// *$i \in Q \cap [ci]$, satisfying* ALL-PCTSP *requirement on $[ci]$ and*
// *given arrival-times ℓ_j.*
if *$[ci]$ is an* ALL *cluster* **then**
ALL_DISTANCE_SLAVE(i)
else
PCTSP_DISTANCE_SLAVE(i)
end
// *Update arrival-times ℓ_j on family $\mathcal{G}_{h+1}$.*
$\ell_j \leftarrow \min\{\ell_j, \min_{i \in Q \cap [ci]} \{\chi_i + d_{ij}\}\} \; \forall \; j \in K$
end
end
end

end

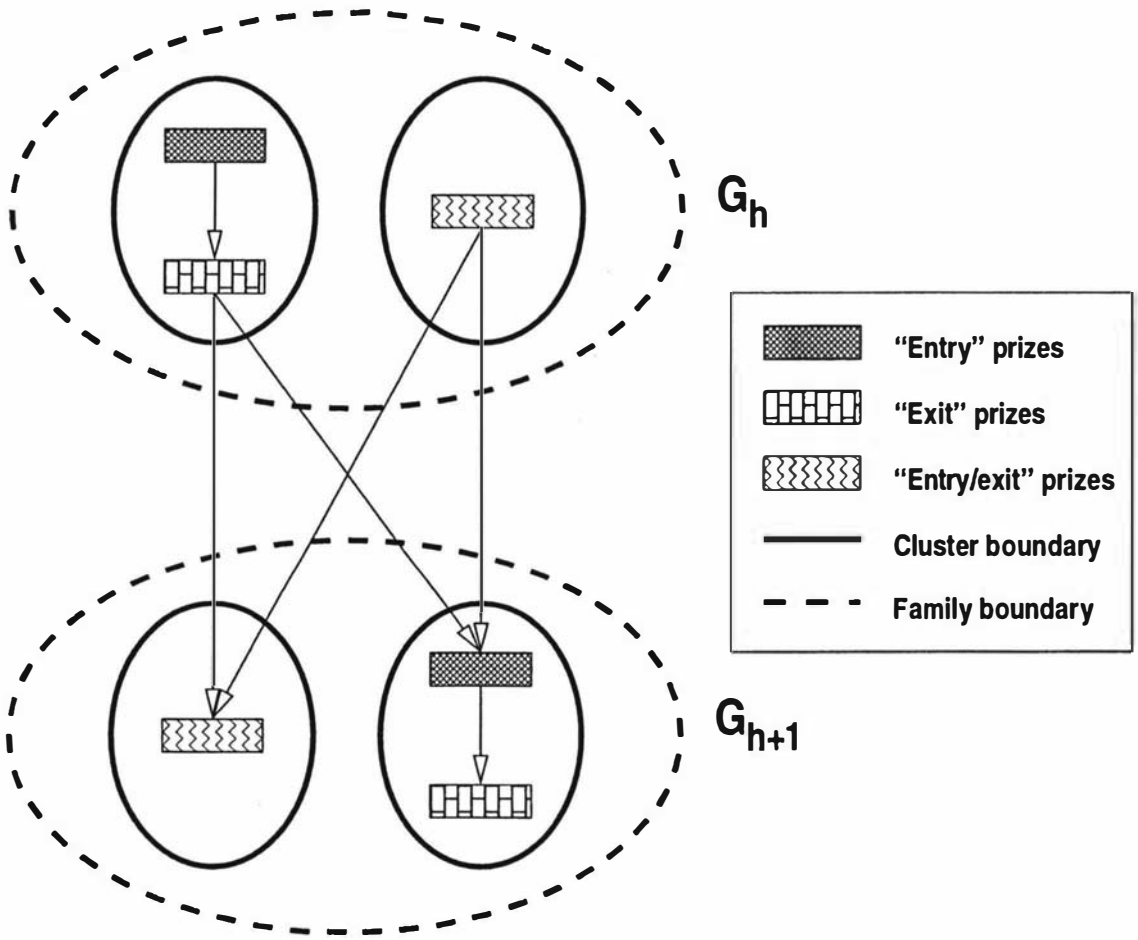


Figure 7.16: Inter-cluster distance calculation

Determine the earliest possible exit-time from every $i \in Q \cap [ci] \neq \emptyset$, satisfying a PCTSP requirement on cluster $[ci]$ and given arrival-times τ_j, where there is a value η remaining to be claimed from cluster $[ci]$.

In a *vector-B&B* algorithm we calculate all the χ values using a single B&B tree rather than calculate each χ value individually with its own B&B tree. Algorithm 7.15 ALL_DISTANCE_SLAVE presents an intra-cluster vector-B&B algorithm which solves the ALL-DISTANCE problem by calculating the optimal *vector* χ of earliest possible exit-times for prizes in cluster $[ci]$ with a vector-based bounding criterion. Similarly, Algorithm 7.16 PCTSP_DISTANCE_SLAVE presents an intra-cluster vector-B&B algorithm for the PCTSP-DISTANCE problem.

Suppose we have a set of upper bounds χ_i on the earliest exit-time of player $\mathcal{B}$ from prizes $i \in Q \cap [ci]$, given the arrival-times τ_j at prizes $j \in Q \cap [ci]$. Initially set $\chi_i = \infty \; \forall \; i \in Q \cap [ci]$. Let Ψ_B be the working subpath through $Q \cap [ci]$ and let t_B be the projected arrival-time at the end of Ψ_B. Initially $\Psi_B = \emptyset$ and t_B is not applicable.

ALL-DISTANCE **bounding criteria.** Fathom B&B node i if

(i). $\Psi_B = \emptyset$ **and** $\nexists \, i \in (Q \cap [ci]) \setminus \Psi_B : \forall \; k \in (Q \cap [ci]) \setminus \Psi_B, \tau_k + d_{ki} < \chi_i$; or

(ii). $\Psi_B \neq \emptyset$ **and** $\nexists i \in (Q \cap [ci]) \setminus \Psi_B : \forall k \in (Q \cap [ci]) \setminus \Psi_B, t_B + d_{Bk} + d_{ki} < \chi_i$.

PCTSP-DISTANCE **bounding criteria.** Fathom B&B node i if

(i). $\Psi_B = \emptyset$ **and** $\nexists i \in (Q \cap [ci]) \setminus \Psi_B : v(\Psi_B) + \sum_{j \in (Q \cap [ci]) \setminus \Psi_B : \tau_j + d_{ji} < \chi_i} v_j \geq \eta$; or

(ii). $\Psi_B \neq \emptyset$ **and** $\nexists i \in (Q \cap [ci]) \setminus \Psi_B : v(\Psi_B) + \sum_{j \in (Q \cap [ci]) \setminus \Psi_B : t_B + d_{Bj} + d_{ji} < \chi_i} v_j \geq \eta$.

Targeting. Lemma 7.A.1 shows that player $\mathcal{B}$ need only target those prizes $j \in Q \setminus \Psi_B$ which are two-optimal-appendages of Ψ_B. Hence let

$$Q_B = \{j \in (Q \cap [ci]) \setminus \Psi_B : j \text{ is a two-optimal-appendage of } \Omega_B\}. \tag{7.11}$$

Branching. Sort Q'_B, i.e., those prizes *not* in Q_B, according to *weighted nearest neighbour and tightest deadline* of Section 6.2.3.3.

For the DISTANCE problem we can employ a useful additional bounding scheme. For setting the distances ℓ_j, the exit-times χ_i need not be computed exactly if we can show that they cannot impact upon the earliest (i.e. optimal) entry-times at prizes in the following family of the $\mathcal{B}$-family-sequence. That is, we are only being interested in the value of χ_i if

$$\exists \; j \in \left(Q \cap \bigcup_{[ci] \in \mathcal{G}_{h+1}} [ci] \right) \text{ such that } \chi_i + d_{ij} < \ell_j.$$

Hence, we initialise χ_i to be the maximum significant exit time from prize i which would be able to improve one of the entry times at the next family, i.e.,

$$\chi_i = \max_{j \in H} \{\ell_j - d_{ij}\} \tag{7.12}$$

Algorithm 7.15 procedure ALL_DISTANCE_SLAVE(i)

Input: i // *B&B tree node.*
$H \leftarrow (Q \cap [ci]) \setminus \Psi_B$
if ($|Q \cap [ci]| = 1$) **then**
  Suppose $Q \cap [ci] = \{i\}$.
  $\chi_i \leftarrow \tau_i$
else if ($|H| = 0$) **then**
  Let i be the last prize on Ψ_B.
  $\chi_i \leftarrow \min\{\chi_i, t_B\}$
else if ($|H| = 1$) **then**
  Suppose $H = \{i\}$.
  $\chi_i \leftarrow \min\{\chi_i, t_B + d_{Bi}\}$
else
  $Q'_B \leftarrow \{j \in H : j$ is a two-optimal-appendage of $\Psi_B\}$
  if ($\Psi_B = \emptyset$) **then**
    for $y \in Q'_B$ **and** $(\exists\, i \in H : \forall\, k \in H, \tau_k + d_{ki} < \chi_i)$ **do**
      ALL_DISTANCE_SLAVE($(i, \mathcal{B} \oplus y)$)
    end
  else
    for $y \in Q'_B$ **and** $(\exists i \in H : \forall k \in H, t_B + d_{Bk} + d_{ki} < \chi_i)$ **do**
      ALL_DISTANCE_SLAVE($(i, \mathcal{B} \oplus y)$)
    end
  end
end

end

where $H = Q \cap \bigcup_{[ci] \in \mathcal{G}_{h+1}} [ci]$. Now, this implies that we need a search strategy for ordering "**for** $[ci] \in \mathcal{G}_h$". We wish to prune the B&B trees for ALL-DISTANCE and PCTSP-DISTANCE as much as possible, so within algorithm 7.14 DISTANCE_DP we branch, at each stage, to the ANY clusters before the ALL-PCTSP clusters.

7.A.2.5 Cluster-DP: POSSIBLE

The POSSIBLE subproblem is similar to the DISTANCE subproblem. Algorithm 7.17 POSSIBLE_DP(i) presents a DP algorithm for the POSSIBLE subproblem, from the perspective of player $\mathcal{A}$, where the final-family requirement applies and hence, necessarily, the family-sequence requirement also applies. Suppose the family-sequence is $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m)$. Each *stage* of the DP corresponds to a family and each *state* corresponds to a cluster.

Algorithm 7.18 POSSIBLE_HEURISTIC(k, τ) presents a heuristic which concludes one of *possible*, *impossible* or *no conclusion* by solving an approximation of the POSSIBLE problem by alternatively avoiding all ALL-PCTSP clusters (to determine *possibility*) and assuming that every cluster is an ANY cluster (to determine *impossibility*).

Algorithm 7.16 procedure PCTSP_DISTANCE_SLAVE(i)

Input: i // *B&B tree node.*
$H \leftarrow (Q \cap [ci]) \setminus \Psi_B$
if $(\upsilon(\Psi_B) \geq \eta)$ **then**
  if $(\Psi_B = \emptyset)$ **then**
    $\chi_i \leftarrow \tau_i \; \forall \; i \in H$
  else
    $\chi_i \leftarrow \min\{\chi_i, t_B + d_{Bi} \; \forall \; i \in H$
    Let j be the last prize on Ψ_B.
    $\chi_j \leftarrow \min\{\chi_j, t_B\}$
  end
else if $(|H| = 1)$ **then**
  Suppose $H = \{i\}$.
  if $(\upsilon(\Psi_B) + \upsilon_i \geq \eta)$ **then**
    if $(\Psi_B = \emptyset)$ **then**
      $\chi_i \leftarrow \tau_i$
    else
      $\chi_i \leftarrow \min\{\chi_i, t_B + d_{Bi}\}$
    end
  end
else
  $Q'_B \leftarrow \{j \in H : j$ is a two-optimal-appendage of $\Psi_B\}$
  found $\leftarrow$ **true**
  for $y \in Q'_B$ **and** *found* **do**
    found $\leftarrow$ **false**
    for $i \in H$ **and not** *found* **do**
      if $(\Psi_B = \emptyset)$ **then**
        $ub_i \leftarrow \sum_{j \in H : \tau_j + d_{ji} < \chi_i} \upsilon_j \; \forall \; i \in H$
      else
        $ub_i \leftarrow \sum_{j \in H : t_B + d_{Bj} + d_{ji} < \chi_i} \upsilon_j \; \forall \; i \in H$
      end
      if $(\upsilon(\Psi_B) + ub_i \geq \eta)$ **then** *found* $\leftarrow$ **true**
    end
    if (*found*) **then** PCTSP_DISTANCE_SLAVE($(i, \mathcal{B} \oplus y)$)
  end
end

end

Algorithm 7.17 function POSSIBLE_DP(i)

Input: i // *B&B tree node or game tree node.*
// *MUST start from satisfied* ANY–ALL-PCTSP *state.*
Let k be the least k such that no prize has been visited from family $\mathcal{G}_k$.
$\tau_j \leftarrow \infty \; \forall \; j \in Q$
if *player* $\mathcal{A}$ *has not yet claimed a prize* **then**
  $\tau_j \leftarrow t_A + d_{Aj} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_k} [ci]$
else
  Let $x \in [cx]$ be the last prize claimed by player $\mathcal{A}$.
  if ANY-ALL-PCTSP *requirement satisfied on* $[cx]$ **then**
    $\tau_j \leftarrow t_A + d_{Aj} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_k} [ci]$
  else
    $\chi_i \leftarrow$ Earliest possible exit-time from $i \in Q \cap [cx]$ satisfying ALL-PCTSP requirement on cluster $[cx]$
    $\tau_j \leftarrow \min_{i \in Q \cap [ci]} \{\chi_i + d_{ij}\} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_k} [ci]$
  end
end
if (POSSIBLE_HEURISTIC(k,τ) = *possible*) **then**
  return(true)
else if (POSSIBLE_HEURISTIC(k,τ) = *impossible*) **then**
  return(false)
end
// *Hence possible_heuristic*(k,τ) = *no conclusion*
for $h = k$ **to** $m - 1$ **do**
  feasible $\leftarrow$ **false**
  for $[ci] \in \mathcal{G}_h$ **do**
    // *Determine exit-times* χ_i.
    $\chi_i \leftarrow$ Earliest possible exit-time from $i \in Q \cap [ci]$ given arrival-times τ_j satisfying ANY-ALL-PCTSP requirement on cluster $[ci]$
    if $(\exists \; i \in [ci] : \chi_i \leq \lambda)$ **then** *feasible* $\leftarrow$ **true**
  end
  if (**not** *feasible*) **then return(false)**
  $K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_h} [ci]$
  // *Determine arrival-times* τ_j *on family* $\mathcal{G}_{h+1}$.
  $\tau_j \leftarrow \min_{i \in K} \{\chi_i + d_{ij}\} \; \forall \; j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_{h+1}} [ci]$
  if (POSSIBLE_HEURISTIC($h+1,\tau$) = *possible*) **then**
    return(true)
  else if (POSSIBLE_HEURISTIC($h+1,\tau$) = *impossible*) **then**
    return(false)
  end
end

end

Algorithm 7.18 **function** POSSIBLE_HEURISTIC(k, τ)

if ($k = m$) **then**
for $j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_m} [ci]$ **do**
if ($t_A + d_{Aj} \leq \lambda$) **then** **return**(*possible*)
end
return(*impossible*)
else
// *Determine if possible.*
$feasible \leftarrow$ **true**
if ($\nexists\, i \in \mathcal{G}_k : \tau_i \leq \lambda$) **then** $feasible \leftarrow$ **false**
for $h = k + 1$ **to** $m - 1$ **and** $feasible$ **do**
$\mathcal{H} \leftarrow \{[ci] \in \mathcal{G}_h : [ci]$ ANY requirement only$\}$
$K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_{h-1}} [ci]$
for $j \in Q \cap \bigcup_{[ci] \in \mathcal{H}} [ci]$ **do**
$\tau_j \leftarrow \min\{\tau_j, \min_{i \in K}\{\tau_i + d_{ij}\}\}$
end
if ($\nexists\, i \in \mathcal{G}_h : \tau_i \leq \lambda$) **then** $feasible \leftarrow$ **false**
end
if ($feasible$) **then**
$\mathcal{H} \leftarrow \{[ci] \in \mathcal{G}_m : [ci]$ ANY requirement only$\}$
$K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_{m-1}} [ci]$
for $j \in Q \cap \bigcup_{[ci] \in \mathcal{H}} [ci]$ **do**
if ($\min_{i \in K}\{\tau_i + d_{ij}\} \leq \lambda$) **then** **return**(*possible*)
end
end
// *Determine if impossible.*
for $h = k + 1$ **to** m **do**
$K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_{h-1}} [ci]$
for $j \in Q \cap \bigcup_{[ci] \in \mathcal{G}_h} [ci]$ **do**
$\tau_j \leftarrow \min\{\tau_j, \min_{i \in K}\{\tau_i + d_{ij}\}\}$
end
if ($\tau_i > \lambda\ \forall\, i \in \mathcal{G}_h$) **then** **return**(*impossible*)
end
end
return(*no conclusion*)

end

7.A.2.6 Prize-DFS: GUARANTEE

We present a depth-first branch-and-bound (B&B) algorithm to determine $\Gamma(j, \mathcal{X})$ for game tree node j, modifying the basic Algorithm 6.15 GUARANTEE_SLAVE to handle the family-sequence, final-family, cluster-no-return, family-no-return and ANY–ALL-PCTSP requirement as necessary. Assume that deadlines ℓ_j have been determined by solving the DISTANCE subproblem.

We implicitly search a B&B tree. Let i be a general B&B node and let Γ_X be the corresponding guarantee subpath of accumulated value $v(\Gamma_X)$. Initialise with $v_{best} = 0$, $\Gamma_X = \emptyset$ and t_X is inherited from game tree node j.

Bounding criterion. Fathom B&B node i if

$$v(\Gamma_X) + \sum_{j \in Q: t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}} v_j \leq v_{best} \tag{7.13}$$

Targeting. Lemma 7.A.1 shows that player $\mathcal{X}$ need only target those prizes $j \in Q \setminus \Gamma_X$ which are two-optimal-appendages of Γ_X. Hence use Algorithm 7.2 TARGET-CLUSTERS to determine $\mathcal{S}_X$.

- If $\Gamma_X = \emptyset$, then

$$Q_X = \{j \in Q \cap \bigcup_{[ci] \in \mathcal{S}_X} [ci] : t_X + d_{Xj} \leq \max\{\ell_j, \lambda\}\} \tag{7.14}$$

- If $\Gamma_X \neq \emptyset$, then let $[cx_0]$ be the cluster from which $\mathcal{X}$ last claimed a prize and

$$Q_X = \{j \in Q \setminus \Gamma_X : t_X + d_{Xj} \leq \max\{\ell_j, \lambda\}\} \cap \left(\{j \in [cx_0] : j \text{ is a two-optimal-appendage of } \Gamma_X\} \cup \bigcup_{[ci] \in \mathcal{S}_X \setminus \{[cx_0]\}} [ci] \right).$$

Final-Family Fathoming. Suppose the final-family requirement applies. Apply procedure FEASIBLE(i) to determine if there is a feasible way of getting through to any prize in the final-cluster satisfying the deadlines ℓ_j and λ.

Branching Rule. Use the same branching rule as per Appendix 6.A.2.3. Algorithm 7.19 FAMILY_GUARANTEE_SLAVE(i) provides the details.

7.A.2.7 Prize-DFS: FEASIBLE

FEASIBLE. Determine if there exists a guaranteed subpath through the remaining prizes satisfying the family-sequence, cluster-no-return, family-no-return and ANY–ALL-PCTSP requirements of player $\mathcal{X}$, the deadlines ℓ_j and λ and claiming a prize from the $\mathcal{X}$-final-family.

We present a depth-first branch-and-bound (B&B) algorithm to solve the FEASIBLE subproblem. Assume that deadlines ℓ_j have been determined by solving the DISTANCE subproblem.

We implicitly search a B&B tree. Let i be a general B&B node and let Γ_X be the corresponding paranoid subpath of accumulated value $v(\Gamma_X)$. Initialise with t_X inherited from game tree node or B&B tree node j, and $\Gamma_X = \emptyset$ if j is a game tree node or Γ_X is inherited from B&B tree node j.

Algorithm 7.19 procedure FAMILY_GUARANTEE_SLAVE(i)

Input: i *// B&B tree node.*
// Assume deadlines ℓ_j.
if *final-family doesn't apply* **or** FAMILY_FEASIBLE_SLAVE(i) **then**
$Q_X \leftarrow \{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}$ and j is a two-optimal-appendage of $\Gamma_X\}$
$ub \leftarrow v(\Gamma_X) + \sum_{j \in Q : t_X + d_{Xj} \leq \min\{\ell_j, \lambda\}} v_j$
if $(ub > v_{best})$ **then**
$v_{best} \leftarrow \max\{v_{best}, v(\Gamma_X)\}$
for $j \in Q_X$ **and** $v_{best} < \min\{ub, \beta_{guarantee}\}$ **do**
FAMILY_GUARANTEE_SLAVE($(i, \mathcal{X} \oplus j)$)
end
end
end

end

Targeting. Lemma 7.A.1 shows that player $\mathcal{X}$ need only target those prizes $j \in Q \setminus \Gamma_X$ which are two-optimal-appendages of Γ_X. Hence use Algorithm 7.2 TARGET-CLUSTERS to determine $\mathcal{S}_X$.

- If $\Gamma_X = \emptyset$, then

$$Q_X = \{j \in Q \cap \bigcup_{[ci] \in \mathcal{S}_X} [ci] : t_X + d_{Xj} \leq \max\{\ell_j, \lambda\}\} \tag{7.15}$$

- If $\Gamma_X \neq \emptyset$, then let $[cx_0]$ be the cluster from which $\mathcal{X}$ last claimed a prize.
  - If $\mathcal{S}_X = \{[cx_0]\}$ then, let

$$\begin{aligned} Q_X \quad = \quad & \{j \in (Q \cap [cx_0]) \setminus \Gamma_X : t_X + d_{Xj} \leq \max\{\ell_j, \lambda\} \text{and} \\ & \quad j \text{ is a two-optimal-appendage of } \Gamma_X\} \end{aligned} \tag{7.16}$$

  - Otherwise, let

$$\begin{aligned} Q_X \quad = \quad & \{j \in Q \setminus \Gamma_X : t_X + d_{Xj} \leq \max\{\ell_j, \lambda\}\} \cap \\ & \bigcup_{[ci] \in \mathcal{S}_X \setminus \{[cx_0]\}} [ci]. \end{aligned}$$

Branching Rule. Same as Appendix 7.A.2.6. Algorithm 7.20 FAMILY_FEASIBLE_SLAVE(i) provides the details.

Algorithm 7.20 **function** FAMILY_FEASIBLE_SLAVE(i)

Input: i // *B&B tree node.*
// *Assume deadlines* ℓ_j.
$K \leftarrow Q \cap \bigcup_{[ci] \in \mathcal{G}_m} [ci]$ **if** $(\Gamma_X \cap K \neq \emptyset)$ **then**
return(true)
end
result $\leftarrow$ **false**
if $(\exists\, i \in K$ **and** $j \in Q_X : t_X + d_{Xj} \leq \ell_j$ **and** $t_X + d_{Xj} + d_{ji} \leq \min\{\ell_i, \lambda\})$ **then**
for $x \in Q_X$ **and not** *result* **do**
result $\leftarrow$ FAMILY_FEASIBLE_SLAVE$((i, \mathcal{X} \oplus x))$
end
end
return(*result*)

end

7.A.3 TWO CLUSTER MEDIAN strategy

In case (vi) of the two cluster problem of Section 7.3.4, both clusters are cluster-leads for player $\mathcal{A}$ only. It is suggested that player $\mathcal{B}$ play a TWO CLUSTER MEDIAN strategy. Section 5.1.4 derived the x, z_1 and z_2 values used in the TWO PRIZE MEDIAN strategy for the *two prize problem*. In this appendix we derive the equivalent expression for the following cluster-window-feasible scenario.

Suppose player $\mathcal{A}\rightarrow[cx_1]\rightarrow\{[cy_1],[cy_2]\}$ and player $\mathcal{B}\rightarrow\{[cy_1],[cy_2]\}$. Let τ_{Ai} be the earliest possible arrival time for player $\mathcal{A}$ at prize $i \in ([cy_1] \cup [cy_2])$, given that player $\mathcal{A}$ looks through $\mathcal{A}\rightarrow[cx_1]$, and satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$.

Alternatively suppose player $\mathcal{A}\rightarrow\{[cy_1],[cy_2]\}$, i.e., player $\mathcal{A}$ does not look through a cluster. Let τ_{Ai} be the earliest possible arrival time for player $\mathcal{A}$ at prize $i \in ([cy_1] \cup [cy_2])$.

Let t_B be the time-stamp of player $\mathcal{B}$.

Choose $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ and let $d_{i_1\rightarrow i_2}$, $d_{i_2\rightarrow i_1}$ and $d_{i_1 i_2}$ be defined as in Appendix 7.A.4.

Suppose

$$\begin{aligned} \tau_{Ai_1} &< t_B + d_{Bi_1} && (7.17)\\ \tau_{Ai_2} &< t_B + d_{Bi_2} && (7.18)\\ \tau_{Ai_1} + d_{i_1\rightarrow i_2} &> t_B + d_{Bi_2} && (7.19)\\ \tau_{Ai_2} + d_{i_2\rightarrow i_1} &> t_B + d_{Bi_1} && (7.20) \end{aligned}$$

We now show how the x, z_1 and z_2 values are derived.

We want to determine if $\exists$ a location Z on the line through the two prizes, such that

$$\begin{aligned} t_B + d_{BZ} + d_{Zi_2} &\leq \tau_{Ai_1} + d_{i_1\rightarrow i_2} \quad \text{and} && (7.21)\\ t_B + d_{BZ} + d_{Zi_1} &\leq \tau_{Ai_2} + d_{i_2\rightarrow i_1}. && (7.22) \end{aligned}$$

Overlay Cartesian coordinates with origin at prize i_1 and positive x-axis towards prize i_2; hence prize i_2 is located at $(d_{i_1 i_2}, 0)$. Let $Z = (z, 0)$. Let (x, y) be the location of player $\mathcal{B}$ in this coordinate system Then $d^2_{Bi_1} = x^2 + y^2$ and $d_{Bi_2} = (d_{i_1 i_2} - x)^2 + y^2$ and hence

$$x = \frac{d^2_{i_1 i_2} + d^2_{Bi_1} - d^2_{Bi_2}}{2d_{i_1 i_2}}.$$

Also $d^2_{BZ} = (x - z)^2 + y^2$.

- Consider constraint (7.21) where $d_{Zi_2} = d_{i_1 i_2} - z$. Substituting, we have

$$\begin{aligned} & \sqrt{(x-z)^2 + y^2} \leq \tau_{Ai_1} - t_B + d_{i_1\rightarrow i_2} - d_{i_1 i_2} + z \\ \Leftrightarrow\ & x^2 - 2xz + z^2 + y^2 \leq (\tau_{Ai_1} - t_B + d_{i_1\rightarrow i_2} - d_{i_1 i_2})^2 + z^2 + \\ & \qquad 2z(\tau_{Ai_1} - t_B + d_{i_1\rightarrow i_2} - d_{i_1 i_2}) \\ \Leftrightarrow\ & z \geq \frac{d^2_{Bi_1} - (\tau_{Ai_1} - t_B + d_{i_1\rightarrow i_2} - d_{i_1 i_2})^2}{2(x + \tau_{Ai_1} - t_B + d_{i_1\rightarrow i_2} - d_{i_1 i_2})} \end{aligned}$$

- Consider constraint (7.22) where $d_{Zi_1} = z$. Substituting, we have

Algorithm 7.21 strategy TWO PRIZE MEDIAN BIAS

// Median location assuming we are $\mathcal{A}$ and $\mathcal{B}\rightarrow i_1$
Input: $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$
$x \leftarrow \dfrac{d_{i_1 i_2}^2 + d_{Ai_1}^2 - d_{Ai_2}^2}{2d_{i_1 i_2}}$
Compute τ_{Bi} $\forall i \in ([cy_1] \cup [cy_2])$.
Compute $d_{i_1 \rightarrow i_2}$ and $d_{i_2 \rightarrow i_1}$ satisfying the ANY–ALL–PCTSP requirement.
$z_1 \leftarrow \dfrac{d_{Ai_1}^2 - (\tau_{Bi_1} - t_A + d_{i_1 \rightarrow i_2} - d_{i_1 i_2})^2}{2(x + \tau_{Bi_1} - t_A + d_{i_1 \rightarrow i_2} - d_{i_1 i_2})}$
$z_2 \leftarrow \dfrac{(\tau_{Bi_2} - t_A + d_{i_2 \rightarrow i_1})^2 - d_{Ai_1}^2}{2(\tau_{Bi_2} - t_A + d_{i_2 \rightarrow i_1} - x)}$
target $\leftarrow$ Bearing of $\max\{z_1, z_2\}$.

end

$$\begin{aligned} & \sqrt{(x-z)^2 + y^2} \leq \tau_{Ai_2} - t_B + d_{i_2 \rightarrow i_1} - z \\ \Leftrightarrow \quad & x^2 - 2xz + z^2 + y^2 \leq (\tau_{Ai_2} - t_B + d_{i_2 \rightarrow i_2})^2 + z^2 - \\ & \qquad 2z(\tau_{Ai_2} - t_B + d_{i_2 \rightarrow i_1}) \\ \Leftrightarrow \quad & z \leq \frac{(\tau_{Ai_2} - t_B + d_{i_2 \rightarrow i_1})^2 - d_{Bi_1}^2}{2(\tau_{Ai_2} - t_B + d_{i_2 \rightarrow i_1} - x)} \end{aligned}$$

The Strategy. See Algorithm 7.21 TWO PRIZE MEDIAN BIAS.

7.A.3.1 $\mathcal{A}\rightarrow[cx_0]\rightarrow[cx_1]\rightarrow\{[cy_1],[cy_2]\}$ and $\mathcal{B}\rightarrow[cy_0]\rightarrow\{[cy_1],[cy_2]\}$

Let τ_{Ai} be the earliest possible arrival time for player $\mathcal{A}$ at prize $i \in ([cy_1] \cup [cy_2])$ given that player $\mathcal{A}$ looks through $\mathcal{A}\rightarrow[cx_0]\rightarrow[cx_1]$, satisfying the ANY–ALL–PCTSP requirement on both clusters $[cx_0]$ and $[cx_1]$. Note: if $[cx_0] = [cy_0]$ then each player need take only one prize from that cluster.

Next, we determine what to do with player $\mathcal{B}$ needing to select an exit-prize from cluster $[cy_0]$. Suppose player $\mathcal{B}$ exits from cluster $[cy_0]$ at exit-prize j at exit-time χ_{Bj} (satisfying the ANY–ALL–PCTSP requirement on cluster $[cy_0]$ if $[cy_0] \neq [cx_0]$). A similar analysis to the above gives:

$$z_1 = \frac{d_{ji_1}^2 - (\tau_{Ai_1} - \chi_{Bj} + d_{i_1 \rightarrow i_2} - d_{i_1 i_2})^2}{2(x + \tau_{Ai_1} - \chi_{Bj} + d_{i_1 \rightarrow i_2} - d_{i_1 i_2})} \tag{7.23}$$

$$z_2 = \frac{(\tau_{Ai_2} - \chi_{Bj} + d_{i_2 \rightarrow i_1})^2 - d_{ji_1}^2}{2(\tau_{Ai_2} - \chi_{Bj} + d_{i_2 \rightarrow i_1} - x)} \tag{7.24}$$

Hence we select such an exit-prize $j \in [cy_0]$ so as to maximize $z_2 - z_1$.

7.A.4 Cluster Feasible Window Mechanisms

Section 7.3.5 introduced the *cluster window scenario* and motivated several cases which are now defined more precisely in this appendix. The cases considered are as follows.

(i). $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$

(ii). $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$

(iii). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$

(iv). $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \{[cy_1], [cy_2]\})$

(v). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

(vi). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

(vii). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

(viii). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow \{[cy_1], [cy_2]\})$

(ix). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$

(x). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$

(xi). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

Figure 7.17 illustrates cases (i)–(iv), Figure 7.18 illustrates cases (v)–(viii) and Figure 7.19 illustrates cases (ix)–(xi).

Notation. Let "$\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_0] \rtimes k \rhd [cx_1]$" mean that player $\mathcal{A}$ moves through cluster $[cx_0]$ with exit-prize $k \in [cx_0]$ and then is looking through cluster $[cx_1]$. Also, we distinguish three distances between prize $i_1 \in [cy_1]$ and prize $i_2 \in [cy_2]$.

- Let $d_{i_1 \rightarrow i_2}$ be the shortest distance from prize i_1 to prize i_2 via sufficient prizes in cluster $[cy_1]$ to satisfy the ANY–ALL–PCTSP requirement on cluster $[cy_1]$.
- Let $d_{i_2 \rightarrow i_1}$ be the shortest distance from prize i_2 to prize i_1 via sufficient prizes in cluster $[cy_2]$ to satisfy the ANY–ALL–PCTSP requirement on cluster $[cy_2]$.
- Let $d_{i_1 i_2}$ be the direct distance between prize i_1 and prize i_2.

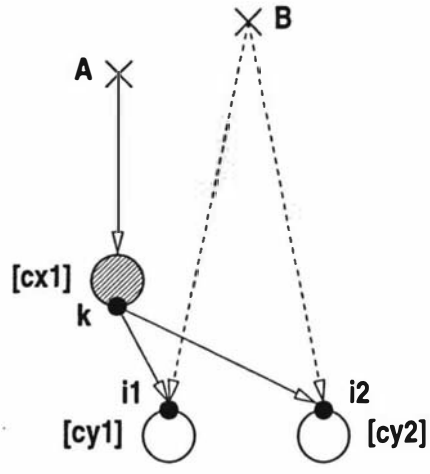
■ Case (i). $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$

Choose $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time from prize k satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$. Let Y be the location of player $\mathcal{B}$ and let t_Y be the current time of player $\mathcal{B}$. We wish to determine how player $\mathcal{B}$ can play to ensure that:

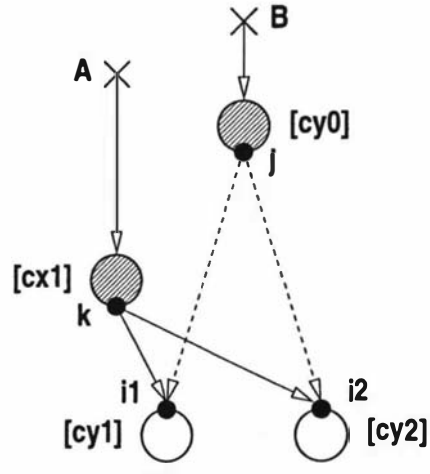
(a). $\mathcal{A}$ does not dominate cluster $[cy_2]$ should $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_1] \rtimes k \rhd [cy_1]$; and

(b). $\mathcal{A}$ does not dominate cluster $[cy_1]$ should $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_1] \rtimes k \rhd [cy_2]$.

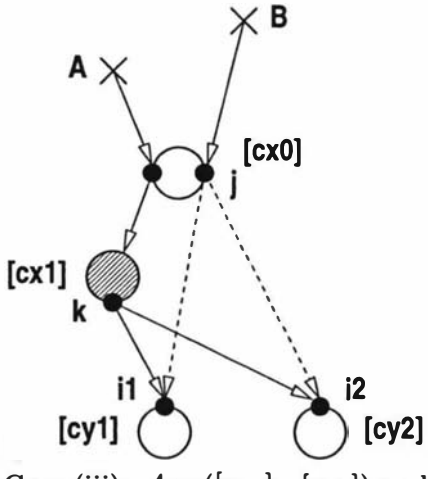
Choose $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$. We need to modify the construction from Section 5.2 since $d_{i_1 \rightarrow i_2}$, $d_{i_2 \rightarrow i_1}$ and $d_{i_1 i_2}$ are not necessarily equal.



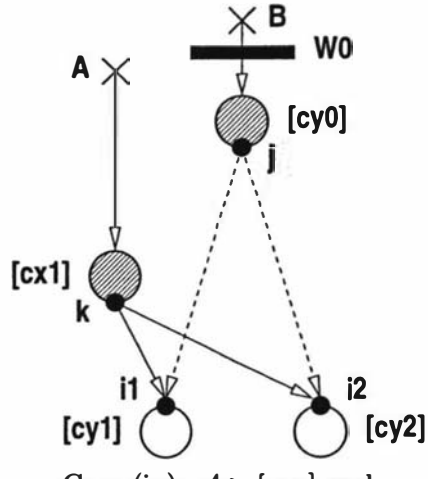
Case (i). $\mathcal{A} \rhd [cx_1]$ and
$\mathcal{B} \rhd \{[cy_1], [cy_2]\}$



Case (ii). $\mathcal{A} \rhd [cx_1]$ and
$\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$



Case (iii). $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and
$\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$



Case (iv). $\mathcal{A} \rhd [cx_1]$ and
$\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \{[cy_1], [cy_2]\})$

Figure 7.17: Cluster Feasible Window Cases (i)-(iv).

- If $t_Y \geq \chi_{Ak}$ then, we require that constraints (7.25)–(7.26) hold.

$$\chi_{Ak} + d_{ki_1} + d_{i_1 \to i_2} > t_Y + d_{Yi_2} \quad (7.25)$$
$$\chi_{Ak} + d_{ki_2} + d_{i_2 \to i_1} > t_Y + d_{Yi_1} \quad (7.26)$$

- If $t_Y < \chi_{Ak}$, then we wish to determine if $\exists$ a *feasibility window* through which player $\mathcal{B}$ can play, i.e., determine whether there exists a location Z such that constraints (7.27)–(7.29) hold.

$$\chi_{Ak} \geq t_Y + d_{YZ} \quad (7.27)$$
$$d_{ki_1} + d_{i_1 \to i_2} \geq d_{Zi_2} \quad (7.28)$$
$$d_{ki_2} + d_{i_2 \to i_1} \geq d_{Zi_1} \quad (7.29)$$

Apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the chosen prizes i_1 and i_2, location Y is the location of player $\mathcal{B}$ and

$$\begin{aligned} r_1 &= d_{ki_2} + d_{i_2 \to i_1} \\ r_2 &= d_{ki_1} + d_{i_1 \to i_2} \\ r_3 &= \chi_{Ak} - t_Y \end{aligned}$$

Since player $\mathcal{B}$ must initially move without knowledge of which exit-prize $k \in [cx_1]$ player $\mathcal{A}$ will exit cluster $[cx_1]$ from, player $\mathcal{B}$ must nominate i_1 and i_2 such that $\forall\ k \in [cx_1]$ the prize window scenario is window feasible.

Definition 7.A.2

$\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$ is **cluster-window-feasible** if $\exists\ i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that $\forall\ k \in [cx_1]$ either:

- $t_Y \geq \chi_{Ak}$ and constraints (7.25)–(7.26) hold; or
- $t_Y < \chi_{Ak}$ and there exists a Z satisfying constraints (7.27)–(7.29).

□

Let $\mathbb{W}_1$ be the cluster window feasible scenario $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$. We can calculate the earliest arrival-times τ_{Bi} for $i \in [cy_1] \cup [cy_2]$ such that player $\mathcal{B}$ moves to prize i in such a way as to satisfy the constraints imposed by $\mathbb{W}_1$.

Choose $i_1 \in [cy_1]$. Let $I_{i_1} \subseteq [cy_2]$ be the set of prizes $i_2 \in [cy_2]$ such that $\forall k \in [cx_1]\ \exists$ a feasibility window $W(i_1, i_2, k)$. If $I_{i_1} = \emptyset$ then let $\tau_{Bi_1} = \infty$; otherwise let

$$\tau_{Bi_1} = \min_{i_2 \in I_{i_1}} \max_{k \in [cx_1]} \left\{ d_{B,W(i_1,i_2,k)} + d_{W(i_1,i_2,k),i_1} \right\}.$$

Similarly define $\tau_{Bi_2}\ \forall\ i_2 \in [cy_2]$.

■ **Case (ii).** $\mathcal{A} \triangleright [cx_1]$ **and** $\mathcal{B} \triangleright ([cy_0] \rightarrow \{[cy_1], [cy_2]\}$

Choose exit-prize $j \in [cy_0]$ of player $\mathcal{B}$ from cluster $[cy_0]$ and let χ_{Bj} be the earliest possible exit-time from exit-prize j satisfying the ANY–ALL–PCTSP requirement on cluster $[cy_0]$. Choose $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from exit-prize k satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$. We wish to determine how player $\mathcal{B}$ can play after exiting from prize $j \in [cy_0]$ at time χ_{Bj} to ensure that:

(a). $\mathcal{A}$ does not dominate cluster $[cy_2]$ should $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_1] \bowtie k \triangleright [cy_1]$; and

(b). $\mathcal{A}$ does not dominate cluster $[cy_1]$ should $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_1] \bowtie k \triangleright [cy_2]$.

Choose $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$.

- If $\chi_{Bj} \geq \chi_{Ak}$, then we require that constraints (7.30)–(7.31) hold.

$$\chi_{Ak} + d_{ki_1} + d_{i_1 \rightarrow i_2} > \chi_{Bj} + d_{ji_2} \tag{7.30}$$
$$\chi_{Ak} + d_{ki_2} + d_{i_2 \rightarrow i_1} > \chi_{Bj} + d_{ji_1} \tag{7.31}$$

- If $\chi_{Bj} < \chi_{Ak}$, then we wish to determine if there exists a *feasibility window* through which player $\mathcal{B}$ can play, i.e., determine whether there exists a location Z such that constraints (7.32)–(7.34) hold.

$$\chi_{Ak} \geq \chi_{Bj} + d_{jZ} \tag{7.32}$$
$$d_{ki_1} + d_{i_1 \rightarrow i_2} \geq d_{Zi_2} \tag{7.33}$$
$$d_{ki_2} + d_{i_2 \rightarrow i_1} \geq d_{Zi_1} \tag{7.34}$$

Apply Algorithm 5.1 WINDOW FEASIBLE to determine if such a location Z exists. The parameters required are the chosen prizes i_1 and i_2, location Y is the location of prize j and

$$\begin{aligned} r_1 &= d_{ki_2} + d_{i_2 \rightarrow i_1} \\ r_2 &= d_{ki_1} + d_{i_1 \rightarrow i_2} \\ r_3 &= \chi_{Ak} - \chi_{Bj} \end{aligned}$$

Definition 7.A.3

$\mathcal{A} \triangleright [cx_1]$ and $\mathcal{B} \triangleright ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$ is **cluster-window-feasible** if $\exists\ j \in [cy_0]$, $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that $\forall\ k \in [cx_1]$ either:

- $\chi_{Bj} \geq \chi_{Ak}$ and constraints (7.30)–(7.31) hold; or
- $\chi_{Bj} < \chi_{Ak}$ and there exists a Z satisfying constraints (7.32)–(7.34).

□

■ **Case (iii).** $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ **and** $\mathcal{B} \triangleright ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$

This case is identical to case (ii) except:

- Choose $j \in [cx_0]$ and let χ_{Bj} be the entry-time of player $\mathcal{B}$ at prize $j \in [cx_0]$. Since both players look through cluster $[cx_0]$, then $[cx_0]$ must be designated an ANY cluster.

- Choose $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from prize $k \in [cx_1]$ via one prize from cluster $[cx_0]$ and satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$.

Hence apply Definition 7.A.3.

■ **Case (iv).** $\mathcal{A} \rhd [cx_1]$ **and** $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \{[cy_1], [cy_2]\})$

The existing cluster window feasible scenario, $\mathbb{W}_0$, implies an earliest possible entry-time τ_{Bj} of player $\mathcal{B}$ at each prize $j \in [cy_0]$ satisfying the constraints imposed by $\mathbb{W}_0$. Hence, this case is identical to case (ii) except:

- Choose $j \in [cy_0]$ and let χ_{Bj} be the earliest possible exit-time of player $\mathcal{B}$ from prize j from cluster $[cy_0]$, satisfying the ANY–ALL–PCTSP requirement on cluster $[cy_0]$, where the entry-time to prize $\ell \in [cy_0]$ is $\tau_{B\ell}$.

Hence apply Definition 7.A.3.

■ **Case (v).** $\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ **and** $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

This case arises from case (F6) of Section 7.5.2. By unwrapping the recursions of (F6) (see Figure 7.11) the base case must have been one of cases (F1), (F2) or (F4b) of Section 7.5.2.

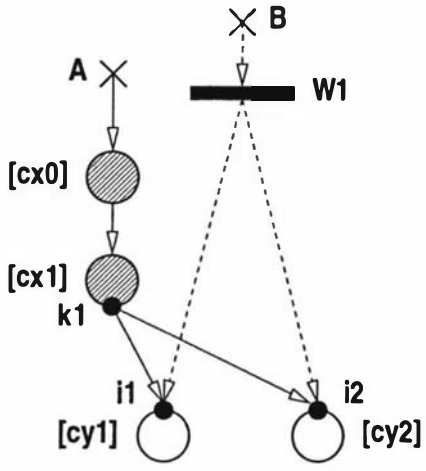
(a). Base case (F1): $\exists\, m \leq -1$ and clusters $\{[cx_{m+1}], [cx_{m+2}], \ldots, [cx_{-1}]\}$ all distinct from each other and clusters $[cx_0]$, $[cx_1]$, $[cy_1]$ and $[cy_2]$ such that $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd [cx_{m+1}]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$—case (i).

(b). Base case (F2): $\exists\, m \leq -1$ and clusters $\{[cx_m], [cx_{m+1}], \ldots, [cx_{-1}]\}$ all distinct from each other and clusters $[cx_0]$, $[cx_1]$, $[cy_1]$ and $[cy_2]$ such that $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd ([cx_m] \rightarrow [cx_{m+1}])$ and $\mathcal{B} \rhd ([cx_m] \rightarrow \{[cy_1], [cy_2]\})$—case (iii). However, the move-through procedure (case (x) of Appendix 7.A.5) associated with $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_m] \rhd [cx_{m+1}]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus [cx_m] \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ updates $\mathbb{W}_1$ to reflect the known exit-prizes of the players from cluster $[cx_m]$. Subsequently, $\mathbb{W}_1$ is the cluster-window-feasible scenario $\mathcal{A} \rhd [cx_{m+1}]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$—case (i) also.

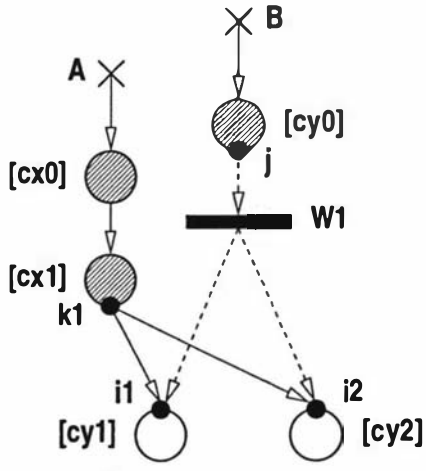
(c). Base case (F4b): $\exists\, m \leq -1$ and clusters $\{[cx_m], [cx_{m+1}], \ldots, [cx_{-1}]\}$ all distinct from each other and clusters $[cx_0]$, $[cx_1]$, $[cy_1]$ and $[cy_2]$ such that $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd ([cx_m] \rightarrow [cx_{m+1}])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \rightarrow [cx_m] \rightarrow \{[cy_1], [cy_2]\})$—case (viii) below. However the move-through procedure (case (xi) Appendix 7.A.5) associated with $\mathcal{A} \rightarrow \mathbb{P}_A \oplus [cx_m] \rhd [cx_{m+1}]$ and $\mathcal{B} \rightarrow \mathbb{P}_B \oplus (\mathbb{W}_0 \rightarrow [cx_m]) \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ updates $\mathbb{W}_1$ to reflect the known exit-prizes of the players from cluster $[cx_m]$. Subsequently, $\mathbb{W}_1$ is the cluster-window-feasible scenario $\mathcal{A} \rhd [cx_{m+1}]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$—case (i) also.

Hence, we may assume that $\mathbb{W}_1$ is the case (i) cluster window feasible scenario $\mathcal{A} \rhd [cx_{m+1}]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$.

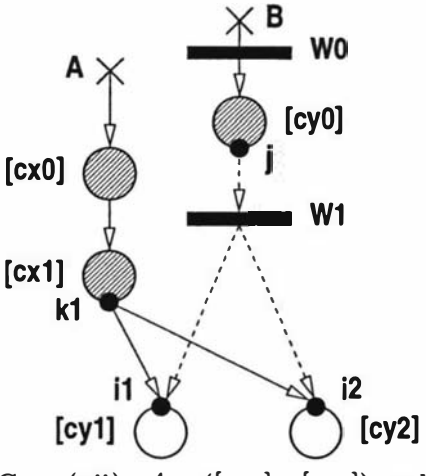
Let χ_{Ak_0} be the earliest possible exit-time of player $\mathcal{A}$ from $k_0 \in [cx_{m+1}]$ satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_0]$. Let $d_{k_0 \rightarrow k_1}$ be the shortest distance from exit-prize



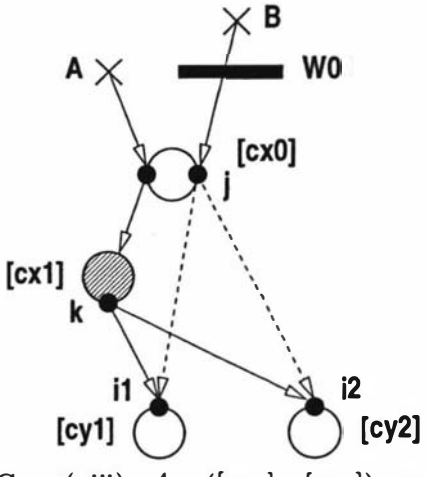
Case (v). $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$



Case (vi). $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$



Case (vii). $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright (\mathbb{W}_0 \rightarrow [cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$



Case (viii). $\mathcal{A} \triangleright ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \triangleright (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow \{[cy_1], [cy_2]\})$

Figure 7.18: Cluster Feasible Window Cases (v)–(viii).

$k_0 \in [cx_{m+1}]$ to exit-prize $k_1 \in [cx_1]$ via the cluster sequence $\{[cx_{m+2}], \ldots, [cx_1]\}$ which satisfies the ANY-ALL-PCTSP requirement on each of these clusters.

Definition 7.A.4

$\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ is **cluster-window-feasible** if $\exists\ i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that $\forall\ k_0 \in [cx_{m+1}]$ and $\forall\ k_1 \in [cx_1]$ either:

- $t_Y \geq \chi_{Ak_0} + d_{k_0 \rightarrow k_1}$ and constraints (7.35)–(7.36) hold;

$$\chi_{Ak_0} + d_{k_0 \rightarrow k_1} + d_{k_1 i_1} + d_{i_1 \rightarrow i_2} > t_Y + d_{Y i_2} \tag{7.35}$$
$$\chi_{Ak_0} + d_{k_0 \rightarrow k_1} + d_{k_1 i_2} + d_{i_2 \rightarrow i_1} > t_Y + d_{Y i_1} \tag{7.36}$$

- or $\chi_{Ak_0} \leq t_Y < \chi_{Ak_0} + d_{k_0 \rightarrow k_1}$ and constraints (7.37)–(7.38) hold

$$\chi_{Ak_0} + d_{k_0 i_1} + d_{i_1 \rightarrow i_2} > t_Y + d_{Y i_2} \tag{7.37}$$
$$\chi_{Ak_0} + d_{k_0 i_2} + d_{i_2 \rightarrow i_1} > t_Y + d_{Y i_1} \tag{7.38}$$

  and $\exists\ Z_1$ satisfying constraints (7.39)–(7.41);

$$\chi_{Ak_0} + d_{k_0 \rightarrow k_1} \geq t_Y + d_{Y Z_1} \tag{7.39}$$
$$d_{k_1 i_1} + d_{i_1 \rightarrow i_2} \geq d_{Z_1 i_2} \tag{7.40}$$
$$d_{k_1 i_2} + d_{i_2 \rightarrow i_1} \geq d_{Z_1 i_1} \tag{7.41}$$

- or $t_Y < \chi_{Ak_0}$ and $\exists\ Z_0$ and Z_1 satisfying constraints (7.40)–(7.45).

$$\chi_{Ak_0} \geq t_Y + d_{Y Z_0} \tag{7.42}$$
$$d_{k_0 i_1} + d_{i_1 \rightarrow i_2} \geq d_{Z_0 i_2} \tag{7.43}$$
$$d_{k_0 i_2} + d_{i_2 \rightarrow i_1} \geq d_{Z_0 i_1} \tag{7.44}$$
$$d_{k_0 \rightarrow k_1} \geq d_{Z_0 Z_1} \tag{7.45}$$

□

Lemma 7.A.5

$\mathcal{A} \rhd ([cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$ is **cluster-window-feasible.**

Proof:

By Definition 7.A.2, choose $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that $\forall\ k_0 \in [cx_{m+1}]$ either: $t_Y \geq \chi_{Ak_0}$ and constraints (7.37)–(7.38) hold; or $t_Y < \chi_{Ak_0}$ and $\exists\ Z_0$ satisfying constraints (7.42)–(7.44), where χ_{Ak_0} is the earliest possible exit-time of player $\mathcal{A}$ from k_0 satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_0]$.

Choose $k_0 \in [cx_{m+1}]$ and $k_1 \in [cx_1]$.

- Suppose $t_Y \geq \chi_{Ak_0} + d_{k_0 \rightarrow k_1}$. Then $t_Y \geq \chi_{Ak_0}$ and so constraints (7.37)–(7.38) hold. Hence

$$\begin{aligned} t_Y + d_{Y i_2} &< \chi_{Ak_0} + d_{k_0 i_1} + d_{i_1 \rightarrow i_2} \\ &\leq \chi_{Ak_0} + d_{k_0 \rightarrow k_1} + d_{k_1 i_1} + d_{i_1 \rightarrow i_2} \end{aligned}$$

satisfying constraint (7.35) and

$$\begin{aligned} t_Y + d_{Yi_1} &< \chi_{Ak_0} + d_{k_0 i_2} + d_{i_2 \to i_1} \\ &\leq \chi_{Ak_0} + d_{k_0 \to k_1} + d_{k_1 i_2} + d_{i_2 \to i_1} \end{aligned}$$

satisfying constraint (7.36).

- Suppose $\chi_{Ak_0} \leq t_Y < \chi_{Ak_0} + d_{k_0 \to k_1}$. Then constraints (7.37)–(7.38) hold. Then, by Lemma 6.3.2, $\exists\ Z_1$ as required.
- Suppose $t_Y < \chi_{Ak_0}$. Choose Z_0 satisfying constraints (7.42)–(7.44). Then, by Lemma 6.3.2, $\exists\ Z_1$ as required.

Thus $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \to \{[cy_1], [cy_2]\})$ is ***cluster-window-feasible*** by Definition 7.A.4. ■

Let $\mathbb{W}_2$ be the cluster window scenario $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_1 \to \{[cy_1], [cy_2]\})$. The proof of Lemma 7.A.5 implies that the constraints imposed by $\mathbb{W}_2$ are no more constraining than those imposed by $\mathbb{W}_1$. The entry-times, $\tau_{Bj}\ \forall\ j \in [cy_1] \cup [cy_2]$, associated with $\mathbb{W}_1$ remain unchanged.

■ Case (vi). $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$

This case arises from either case (F3) or case (F7) of Section 7.5.2.

(a). Case (F7) of Section 7.5.2: By unwrapping the recursions of (F7) (see Figure 7.11) the base case must have been either a case (F3) or a case (F5a).

- Base case (F3): $\exists\ m \leq -1$ and clusters $\{[cx_m], [cx_{m+1}], \ldots, [cx_{-1}]\}$ all distinct from each other and clusters $[cx_0]$, $[cx_1]$, $[cy_0]$, $[cy_1]$ and $[cy_2]$ such that $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd [cx_m]$ and $\mathcal{B} \rhd ([cy_0] \to \{[cy_1], [cy_2]\})$—case (ii).
- Base case (F5a): $\exists\ m \leq -1$ and clusters $\{[cx_m], [cx_{m+1}], \ldots, [cx_{-1}]\}$ all distinct from each other and clusters $[cx_0]$, $[cx_1]$, $[cy_0]$, $[cy_1]$ and $[cy_2]$ such that $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd (\mathbb{W}_0 \to [cx_m])$ and $\mathcal{B} \rhd ([cy_0] \to \{[cy_1], [cy_2]\})$—case (x).

(b). Case (F3) of Section 7.5.2: $\mathbb{W}_1$ is the cluster-window scenario $\mathcal{A} \rhd [cx_0]$ and $\mathcal{B} \rhd ([cy_0] \to \{[cy_1], [cy_2]\})$—case (ii). Hence let $m = 0$.

Let χ_{Bj} be the earliest possible exit-time of player $\mathcal{B}$ from prize $j \in [cy_0]$, satisfying the ANY–ALL–PCTSP requirement on cluster $[cy_0]$.

From the location of player $\mathcal{A}$ at the time at which $\mathcal{A} \rhd [cx_m]$, let χ_{Ak_0} be the earliest possible exit-time of player $\mathcal{A}$ from prize $k_0 \in [cx_m]$, satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_m]$ and in base case (F5a) the entry-time to prize $\ell \in [cx_m]$ is $\tau_{A\ell}$. Note that in case (F7), player $\mathcal{A}$ has already moved through cluster $[cx_m]$ and, hence, $\chi_{Ak_0} \leq t_A$, but the knowledge of the actual exit-prize of player $\mathcal{A}$ from cluster $[cx_m]$ may not be known to player $\mathcal{B}$ at time t_B.

Let $d_{k_0 \to k_1}$ be the shortest distance from exit-prize $k_0 \in [cx_m]$ to exit-prize $k_1 \in [cx_1]$ via the cluster sequence $\{[cx_{m+1}], \ldots, [cx_1]\}$ which satisfies the ANY-ALL-PCTSP requirement on each of these clusters.

Definition 7.A.6

$\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$ is **cluster-window-feasible** if $\exists\ j \in [cy_0]$, $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that $\forall\ k_0 \in [cx_m]$ and $\forall\ k_1 \in [cx_1]$ either:

- $\chi_{Bj} \geq \chi_{Ak_0} + d_{k_0 \to k_1}$ and constraints (7.46)-(7.47) hold;

$$\chi_{Ak_0} + d_{k_0 \to k_1} + d_{k_1 i_1} + d_{i_1 \to i_2} > \chi_{Bj} + d_{ji_2} \tag{7.46}$$
$$\chi_{Ak_0} + d_{k_0 \to k_1} + d_{k_1 i_2} + d_{i_2 \to i_1} > \chi_{Bj} + d_{ji_1} \tag{7.47}$$

- or $\chi_{Ak_0} \leq \chi_{Bj} < \chi_{Ak_0} + d_{k_0 \to k_1}$ and constraints (7.48)-(7.49) hold

$$\chi_{Ak_0} + d_{k_0 i_1} + d_{i_1 \to i_2} > \chi_{Bj} + d_{ji_2} \tag{7.48}$$
$$\chi_{Ak_0} + d_{k_0 i_2} + d_{i_2 \to i_1} > \chi_{Bj} + d_{ji_1} \tag{7.49}$$

  and $\exists\ Z_1$ satisfying constraints (7.40)-(7.41) and (7.50);

$$\chi_{Ak_0} + d_{k_0 \to k_1} \geq \chi_{Bj} + d_{jZ_1} \tag{7.50}$$

- or $\chi_{Bj} < \chi_{Ak}$ and $\exists\ Z_0$ and Z_1 satisfying constraints (7.40)-(7.41), (7.43)-(7.45) and (7.51).

$$\chi_{Ak_0} \geq \chi_{Bj} + d_{jZ_0} \tag{7.51}$$

□

Lemma 7.A.7

$\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$ is **cluster-window-feasible.**

Proof:

By Definition 7.A.3, choose $j \in [cy_0]$, $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$ such that for every $k_0 \in [cx_m]$ either: $\chi_{Bj} \geq \chi_{Ak_0}$ and constraints (7.48)-(7.49) hold; or $\chi_{Bj} < \chi_{Ak_0}$ and $\exists\ Z_0$ satisfying constraints (7.43)-(7.44) and (7.51).

Choose $k_0 \in [cx_m]$ and $k_1 \in [cx_1]$.

- Suppose $\chi_{Bj} \geq \chi_{Ak_0} + d_{k_0 \to k_1}$. Then $\chi_{Bj} \geq \chi_{Ak_0}$ and so constraints (7.48)-(7.49) hold. Hence

$$\begin{aligned} \chi_{Bj} + d_{ji_2} &< \chi_{Ak_0} + d_{k_0 i_1} + d_{i_1 \to i_2} \\ &\leq \chi_{Ak_0} + d_{k_0 \to k_1} + d_{k_1 i_1} + d_{i_1 \to i_2} \end{aligned}$$

  satisfying constraint (7.46) and

$$\begin{aligned} \chi_{Bj} + d_{ji_1} &< \chi_{Ak_0} + d_{k_0 i_2} + d_{i_2 \to i_1} \\ &\leq \chi_{Ak_0} + d_{k_0 \to k_1} + d_{k_1 i_2} + d_{i_2 \to i_1} \end{aligned}$$

  satisfying constraint (7.47).

- Suppose $\chi_{Ak_0} \leq \chi_{Bj} < \chi_{Ak_0} + d_{k_0 \to k_1}$. Then constraints (7.48)–(7.49) hold. Then by Lemma 6.3.2, $\exists\ Z_1$ as required.
- Suppose $\chi_{Bj} < \chi_{Ak_0}$. Choose Z_0 satisfying constraints (7.43)–(7.44) and (7.51). Then by Lemma 6.3.2, $\exists\ Z_1$ as required.

Thus $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$ is *cluster-window-feasible* by Definition 7.A.6.

■

Let $\mathbb{W}_2$ be the cluster window scenario $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$. The proof of Lemma 7.A.7 implies that the constraints imposed by $\mathbb{W}_2$ are no more constraining than those imposed by $\mathbb{W}_1$.

■ Case (vii). $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \to [cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$

Identical to case (vi) except:

- Let χ_{Bj} be the earliest possible exit-time of player $\mathcal{B}$ from prize $j \in [cy_0]$ from cluster $[cy_0]$, satisfying the ANY–ALL–PCTSP requirement on cluster $[cy_0]$, where the entry-time to prize $\ell \in [cy_0]$ is $\tau_{B\ell}$.

Hence apply Definition 7.A.6.

■ Case (viii). $\mathcal{A} \rhd ([cx_0] \to [cx_1])$ and $\mathcal{B} \rhd (\mathbb{W}_0 \to [cx_0] \to \{[cy_1], [cy_2]\})$

Identical to case (iii) except:

- Choose $j \in [cx_0]$ and let $\chi_{Bj} = \tau_{Bj}$.

Hence apply Definition 7.A.3.

■ Case (ix). $\mathcal{A} \rhd (\mathbb{W}_0 \to [cx_0] \to [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \to \{[cy_1], [cy_2]\})$

Identical to case (iii) except:

- Choose $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from prize $k \in [cx_1]$ via one prize from cluster $[cx_0]$ and satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$, where the entry-time to prize $\ell \in [cx_1]$ is $\tau_{A\ell}$.

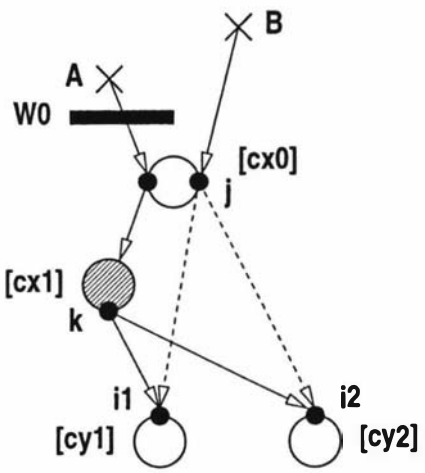
Hence apply Definition 7.A.3.

■ Case (x). $\mathcal{A} \rhd (\mathbb{W}_0 \to [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \to \{[cy_1], [cy_2]\})$

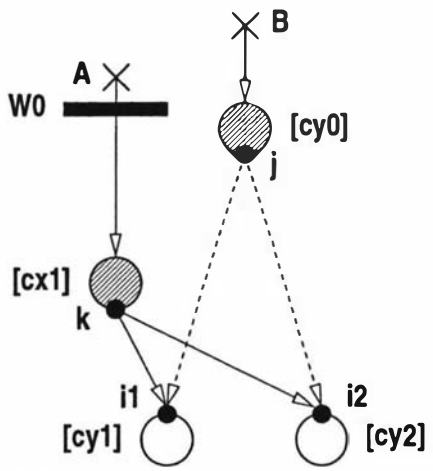
Identical to case (ii) except:

- Choose $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from prize k satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$, where the entry-time to prize $\ell \in [cx_1]$ is $\tau_{A\ell}$.

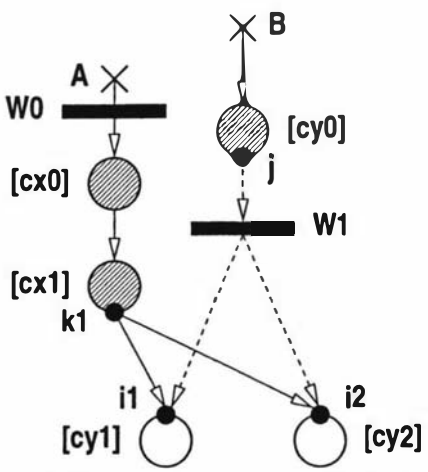
Hence apply Definition 7.A.3.



Case (ix). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cx_0] \rightarrow \{[cy_1], [cy_2]\})$



Case (x). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \{[cy_1], [cy_2]\})$



Case (xi). $\mathcal{A} \rhd (\mathbb{W}_0 \rightarrow [cx_0] \rightarrow [cx_1])$ and $\mathcal{B} \rhd ([cy_0] \rightarrow \mathbb{W}_1 \rightarrow \{[cy_1], [cy_2]\})$

Figure 7.19: Cluster Feasible Window Cases (ix)–(xi).

■ **Case (xi).** $\mathcal{A} \triangleright (\mathbb{W}_0 \to [cx_0] \to [cx_1])$ **and** $\mathcal{B} \triangleright ([cy_0] \to \mathbb{W}_1 \to \{[cy_1], [cy_2]\})$

Identical to case (vi) except:

- Let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from prize $k \in [cx_1]$ *via cluster* $[cx_0]$ and satisfying the ANY–ALL–PCTSP requirement on clusters $[cx_0]$ and $[cx_1]$, where the entry-time to prize $\ell \in [cx_0]$ is $\tau_{A\ell}$.

Hence apply Definition 7.A.6.

7.A.5 Cluster Window Move-Through Mechanisms

There are cases in which only one player moves through a cluster and cases in which both players move through a cluster.

7.A.5.1 Cases in which only one player moves through

(i). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_0\rightarrow[cy_0])$.

(ii). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus (\mathbb{W}_0\rightarrow[cx_0]) \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \rhd [cy_0]$.

(iii). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

(iv). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \rhd ([cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

(v). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_0\rightarrow[cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

■ **Case (i).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_0\rightarrow[cy_0])$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$. The notation "$\tau_B \ltimes [cy_0]$" indicates that we specify the entry-times τ_B on cluster $[cy_0]$ which were determined from $\mathbb{W}_0$. Note that player $\mathcal{B}$ does not include cluster $[cx_0]$ in the $\mathcal{B}$-family-sequence since player $\mathcal{A}$ must satisfy the ANY-ALL-PCTSP requirement on cluster $[cx_0]$ and in addition to claiming some value from cluster $[cx_1]$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_0\rightarrow[cy_0])$.

■ **Case (ii).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus (\mathbb{W}_0\rightarrow[cx_0]) \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd [cy_0]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{\tau_A \ltimes [cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{[cy_0]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd [cy_0]$.

■ **Case (iii).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_1], \tau_B \ltimes [cy_2]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

■ **Case (iv).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd ([cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{[cy_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1],[cy_2]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$. Recall that "$\{[cy_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1],[cy_2]\}$" means that we apply the *static cluster window distance correction* by temporarily assigning

$$d_{ji} \leftarrow \delta_{ji} \; \forall \; j \in [cy_0], i \in [cy_1] \cup [cy_2].$$

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd ([cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

■ **Case (v).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \rhd (\mathbb{W}_0\rightarrow[cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1],[cy_2]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_0\rightarrow[cy_0]\rightarrow\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

7.A.5.2 Cases in which both players move through

(vi). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1\rightarrow[cy_1])$.

(vii). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1\rightarrow[cy_1])$.

(viii). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cy_0]) \rhd [cy_1]$.

(ix). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cy_0]) \rhd (\mathbb{W}_1\rightarrow[cy_1])$.

(x). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

(xi). $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cx_0]) \rhd (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$.

■ **Case (vi).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [cy_0] \rhd (\mathbb{W}_1\rightarrow[cy_1])$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{[cy_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Since the exit-time, χ_{Bj}, of player $\mathcal{B}$ from exit-prize $j \in [cy_0]$ depends on how player $\mathcal{B}$ moves through cluster $[cy_0]$, we can only determine the cluster window distance correction, δ_{ji_1}, to satisfy the constraints imposed by $\mathbb{W}_1$, once χ_{Bj} is known. Hence the distance correction must be dynamic.

Dynamic cluster window distance correction. The cluster window distance correction for player $\mathcal{B}$ is dependent upon the exit-time of player $\mathcal{B}$ from prize j. A **dynamic cluster window distance correction** procedure inside FAMILY-PRIZE-GUARANTEE and FAMILY-PRIZE-DT stores the original χ_{Ak} exit-times associated with $\mathbb{W}_1$ and uses these to calculate the distance δ_{ji_1}, for $j \in [cx_0]$ and $i_1 \in [cy_1]$, when requested by player $\mathcal{B}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}{\rightarrow}\{[cy_0]\}_{\langle \mathbb{W}_1 \rangle}{\rightarrow}\{[cy_1]\}_{FF}$ with $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}{\rightarrow}\{[cy_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cy_0]$ of player $\mathcal{B}$. We use the notation $\langle \mathbb{W}_1 \rangle$ to denote the application of a dynamic cluster window distance correction which ensures that $\mathbb{W}_1$ is cluster window feasible.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_1{\rightarrow}[cy_1])$. Let $[cx]$ be the cluster associated with $\mathbb{W}_1$ from when we considered the exit-prize $k \in [cx]$ for player $\mathcal{A}$. The entry-times τ_{Bi_1} for each entry-prize $i_1 \in [cy_1]$ are calculated as

$$\tau_{Bi_1} = t_B + \min_{i_2 \in [cy_2]} \left\{ \max_{k \in [cx]} \left\{ d_{B,W(i_1,i_2,k)} + d_{W(i_1,i_2,k),i_1} \right\} \right\}$$

where $W(i_1, i_2, k)$ is the prize feasibility window associated with the prize window scenario $\mathcal{A} \rhd k$ and $\mathcal{B} \rhd \{i_1, i_2\}$ and τ_{Ak} is the earliest possible exit time from $[cx]$. Note that player $\mathcal{A}$ may have already moved through cluster $[cx]$, but we assume that player $\mathcal{B}$ has no knowledge of the exit-prize.

■ Case (vii). $\mathcal{A}{\rightarrow}\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ and $\mathcal{B}{\rightarrow}\mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1{\rightarrow}[cy_1])$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}_{FF}$ with $\mathcal{B}{\rightarrow}\{[cx_0]\}_{\mathbb{W}_1}{\rightarrow}\{[cy_1]\}{\rightarrow}\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$. Note that each player does not know the exit-prize of the other player from cluster $[cx_0]$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}{\rightarrow}\{[cx_0]\}_{\langle \mathbb{W}_1 \rangle}{\rightarrow}\{[cy_1]\}_{FF}$ with $\mathcal{A}{\rightarrow}\{[cx_0]\}{\rightarrow}\{[cx_1]\}{\rightarrow}\{[cy_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cy_0]$ of player $\mathcal{B}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_1{\rightarrow}[cy_1])$. Now that the players have moved through cluster $[cx_0]$ we now know the exit-prize of each player from cluster $[cx_0]$. Hence, redefine $\mathbb{W}_1$ to be the cluster window scenario $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd \{[cy_1], [cy_2]\}$, where $[cy_2]$ is the other half of the cluster-pair defined by the original $\mathbb{W}_1$. The entry-times τ_{Bi_1} for each entry-prize $i_1 \in [cy_1]$ are therefore calculated as

$$\tau_{Bi_1} = t_B + \min_{i_2 \in [cy_2]} \left\{ d_{B,W(i_1,i_2)} + d_{W(i_1,i_2),i_1} \right\}$$

where $W(i_1, i_2)$ is the prize feasibility window associated with the prize window scenario $\mathcal{A} \rhd \emptyset$ and $\mathcal{B} \rhd \{i_1, i_2\}$.

■ **Case (viii).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cy_0]) \rhd [cy_1]$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}\rightarrow\{[cy_1]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}\rightarrow\{[cy_1]\}_{FF}$ with $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd [cy_1]$.

■ **Case (ix).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cy_0]) \rhd (\mathbb{W}_1\rightarrow[cy_1])$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{\tau_B \ltimes [cy_0]\}_{\langle\mathbb{W}_1\rangle}\rightarrow\{[cy_1]\}$ with $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}\rightarrow\{[cy_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{B}$.

Result: $\mathcal{A} \rhd [cx_1]$ and $\mathcal{B} \rhd (\mathbb{W}_1\rightarrow[cy_1])$. Calculate the entry-times τ_{Bi_1} for each entry-prize $i_1 \in [cy_1]$ as in case (vii).

■ **Case (x).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \rhd [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus [cx_0] \rhd (\mathbb{W}_1\rightarrow\{[cy_1], [cy_2]\})$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{[cx_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1], [cy_2]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

Since both players look through cluster $[cx_0]$, we *must* also move player $\mathcal{B}$ through cluster $[cx_0]$, even though player $\mathcal{B}$ has no secondary target cluster. What objective should player $\mathcal{B}$ have? A conservative objective is to do as well as possible under the worst possible circumstances. Hence an option is to maximize the value claimed from $[cx_0]$ plus the minimum of the value claimed from $[cy_1]$ (should player $\mathcal{A}$ target $[cy_2]$ next) and the value claimed from $[cy_2]$ (should player $\mathcal{A}$ target $[cy_1]$ next). It would require substantial changes to the existing FAMILY-PRIZE-DT engine to implement this objective for player $\mathcal{B}$. Instead, we have designed the following procedure which mimics the changes that would be necessary to FAMILY-PRIZE-GUARANTEE by initially moving player $\mathcal{B}$ through cluster $[cx_0]$ using a paranoid subpath.

Without knowing the exit-prize of player $\mathcal{A}$ from cluster $[cx_0]$, player $\mathcal{A}$ must select a subpath through $[cx_0]$ such that the cluster window scenario remains feasible. Suppose player $\mathcal{B}$ selects a paranoid subpath S through cluster $[cx_0]$ with exit-prize $j \in [cx_0]$ and exit-time t_S. Choose entry-prizes $i_1 \in [cy_1]$ and $i_2 \in [cy_2]$. Choose exit-prize $k \in [cx_1]$ and let χ_{Ak} be the earliest possible exit-time of player $\mathcal{A}$ from prize k satisfying the ANY–ALL–PCTSP requirement on cluster $[cx_1]$. Determine a feasibility window $W(S, i_1, i_2, k)$.

Temporarily move player $\mathcal{A}$ to the location of prize k at time χ_{Ak}.

(a). Temporarily move $\mathcal{B}$ to the location of prize i_1 at time $t_S + d_{j,W(S,i_1,i_2,k)} + d_{W(S,i_1,i_2,k),i_1}$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{[cy_1]\}$ with $\mathcal{A}\rightarrow\{[cy_2]\}\rightarrow\{[cy_1]\}$ to determine the exit-value, $\mu_1(S,i_1,i_2,k)$, from cluster $[cy_1]$ of player $\mathcal{B}$.

(b). Temporarily move $\mathcal{B}$ to the location of prize i_2 at time $t_S + d_{j,W(S,i_1,i_2,k)} + d_{W(S,i_1,i_2,k),i_2}$. Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{B}\rightarrow\{[cy_2]\}$ with $\mathcal{A}\rightarrow\{[cy_1]\}\rightarrow\{[cy_2]\}$ to determine the exit-value, $\mu_2(S,i_1,i_2,k)$, from cluster $[cy_2]$ of player $\mathcal{B}$.

Finally choose the subpath S to maximize

$$v(S) + \max_{i_1\in[cy_1],i_2\in[cy_2]} \left\{ \min_{k\in[cx_1]} \{\min\{\mu_1(S,i_1,i_2,k),\mu_2(S,i_1,i_2,k)\}\} \right\}.$$

Result: $\mathcal{A} \triangleright [cx_1]$ and $\mathcal{B} \triangleright (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$. Now that the players have moved through cluster $[cx_0]$ we now know the exit-prize of each player from cluster $[cx_0]$. Hence redefine $\mathbb{W}_1$ to be the cluster window scenario $\mathcal{A} \triangleright [cx_1]$ and $\mathcal{B} \triangleright \{[cy_1],[cy_2]\}$.

■ **Case (xi).** $\mathcal{A}\rightarrow\mathbb{P}_A \oplus [cx_0] \triangleright [cx_1]$ **and** $\mathcal{B}\rightarrow\mathbb{P}_B \oplus (\mathbb{W}_0\rightarrow[cx_0]) \triangleright (\mathbb{W}_1\rightarrow\{[cy_1],[cy_2]\})$

Apply FAMILY-PRIZE-GUARANTEE or FAMILY-PRIZE-DT to $\mathcal{A}\rightarrow\{[cx_0]\}\rightarrow\{[cx_1]\}_{FF}$ with $\mathcal{B}\rightarrow\{\tau_B \ltimes [cx_0]\}_{\mathbb{W}_1}\rightarrow\{[cy_1],[cy_2]\}\rightarrow\{[cx_1]\}$ to determine the exit-prize, exit-time and exit-value from cluster $[cx_0]$ of player $\mathcal{A}$.

To move player $\mathcal{B}$ through cluster $[cx_0]$ we use the same procedure as in case (x), except that the entry-times, τ_{Bj}, of player $\mathcal{B}$ to prizes $j \in [cx_0]$ are implied by $\mathbb{W}_0$.

CHAPTER 8

Strategic Planning for Large Problems

Once you open a can of worms, the only way to recan them is to use a larger can.

— ZYMURGY'S FIRST LAW OF EVOLVING SYSTEM DYNAMICS

The strategic engine CLUSTER-DT of the previous chapter assumes that the prizes have a discernible, natural, cluster-like structure which is strategically useful. To consider problems involving a very large number of prizes which do not satisfy these criteria, a transition from strategies founded on the logical analysis of many contingencies to coarse strategies founded on heuristic intuition is necessary. The philosophy we adopt with large problems is that of strategic, but efficient, harvesting.

8.0 Introduction

A **large problem instance** can be defined as a problem instance which does not fit into the tiny, small or medium classifications of the previous chapters. There are two basic ways that medium problems can 'decay' as the number of prizes increases. The first is that there is insufficient useful structure at the global-frame. The second is that there is still some useful, discernible, family-cluster-like structure at the global-frame, but that there are either too many clusters in each family or too many prizes in each cluster to apply CLUSTER-DT.

There are several possible approaches to large problems.

Approximate-CLUSTER-DT. Suppose there is still some useful family-cluster structure, but either there are too many clusters to apply the full CLUSTER-DT branching, or the size of each cluster is too big to apply PRIZE-DT or PARANOID dominance criteria. Hence, we may wish to approximate the CLUSTER-DT game tree, or heuristically prune the CLUSTER-DT game tree (e.g., with an approximate α–β scheme), or solve many of the combinatorial subproblems heuristically using local search.

Family-frame. Continue the progression from tiny to small to medium problems by inserting a *family-frame* between the global-frame and the *cluster-frame*. Hence, a global-monitor determines a family-frame which is essentially a coarse clustering of the prizes. Then a family-monitor determines a cluster-frame with a finer family-cluster structure defined on some subregion of the prizes. Strategies applied on the family-frame may be very simple in terms of complexity or contingency and are essentially designed to coarsely reduce the scope and planning horizon sufficiently so that CLUSTER-DT is computationally tractable.

Grid-structured. Replace the family-cluster-structure with a grid-structure which, rather than trying to identify a natural structure, imposes a division of the prizes at the global-frame into cells. The focus switches from travelling between discrete clusters and then harvesting a cluster to harvesting within a grid-cell whilst moving towards some target grid-cell. The significant property of a large problem is the dense and near continuous—rather than sparse and discrete—prize collection. The SPA of Chapter 4 can be retained, but the methods operating at each frame are more heuristic.

We do not consider the 'approximate CLUSTER-DT' approach further since it essentially involves heuristically relaxing the existing CLUSTER-DT. This in itself would be an interesting investigation, to determine the tradeoffs between heuristically relaxing various elements of the search and evaluation. However, we have already invested considerably in the design of CLUSTER-DT so we add nothing new to the CLUSTER-DT paradigm via tweaking the implementation.

We do not consider the 'family-frame' approach either, since this simply adds another layer to the Medium-DMS designed in Chapter 7, i.e., too much computational complexity and too much accuracy in CLUSTER-DT. We need to undertake some kind of evaluation of where the good harvesting regions are without "worrying too much" about the opponent.

Instead, we attempt to use the 'grid-structured' approach to design a different realisation of the SPA/DMS, i.e., another extended design and implementation of the strategic framework, which is looser at the coarsest scale with respect to contingency. The driving issue is continuous harvesting, presuming there are many prizes (dense) and essentially all routing is intra-cluster, with little or no inter-cluster routing. We essentially need to show that the strategic framework also contends with non-clustered problems. Here the agglomeration of prizes is *divisional*, rather than natural, and, since we assume that the prize locations are reasonably dense, there is no realistic inter-cluster routing. Concentration is primarily on harvesting of prizes, but strategically we must be able to balance harvesting *en route to* a harvesting region against the potential harvesting *within* a region.

We firstly generalize the grid paradigm to a strategic approach for large problems in Section 8.1. The resulting paradigm, incorporating the 'family frame' approach, suggests a rich set of framework-based strategies for large problems whose further investigation is recommended for future work.

Secondly, we consider methods based around a grid-structure in Sections 8.2–8.5. The grid paradigm is based on dividing up the prize region rather than seeking natural clusters of prizes.

8.1 Strategic Approach to Large Problems

The large number of prizes and lack of natural structure implies a near continuous harvesting of prizes rather than discrete harvesting of clusters. We require a paradigm for such large problems which guides and refines the SPA. To this end, we firstly propose five tactical issues that require consideration and, secondly, propose a paradigm that involves identifying the important harvesting features and determining an effective strategic sequence of these features.

8.1.1 Tactical Issues

The SPA of Chapter 4 highlighted the need for a *macro strategy* that takes a global view of the situation and determines *intermediate goals* to strive for. A goal is likely to span a reasonable duration of time and incorporate some decoupling of that period from setup for future periods. Thus, a strategy level process can narrow the focus by outlining the objectives and requirements of a subordinate route planner at a more tactical level of detail. We call the resulting package of requirements a *task*. Often these tasks can be modelled as problems similar in character to existing selection, sequencing and scheduling problems, amenable to modifications of existing heuristic search techniques.

We propose five basic tactical issues as follows.

Travel and Harvest. Two extremes of activity are *harvesting* prizes in some local subregion and *travelling* urgently to some specific destination, with a continuum of possibilities in between. The general scenario would be to harvest whilst travelling to some subregion. There is a tradeoff between the harvesting of prizes that may be developed on the way and the urgency with which the destination needs to be reached. Thus, we consider travelling and harvesting with purpose.

Conflict with Opponent. The opponent can impinge upon tactical options over the local planning horizon anywhere between "not-at-all" through to "entirely". Chapters 5–7 concentrated on local contingent planning, extensively involving the opponent. In this chapter, however, we choose to concentrate efforts upon strategies which do not consider the opponent in such rigour over the local planning horizon. Such strategies apply subset selection and sequencing subproblems rather than game tree searching.

Harvesting Profile. A **harvesting profile** is a tradeoff between prize value and distance travelled. The optimal profile would be equivalent to the set of non-dominated solutions to

the **Multiobjective Vending Problem** (MVP) of Keller and Goodchild [127] (see Section 2.1.1). What we intend to generate is a guideline for expected prize yield throughout the time invested in harvesting. This generalizes both the budget of time to complete the task and the expectation in terms of prize yield, that could vary in dominant influence over the planning period. Requirements may be either hard or soft, but need to be realistic. The profile could define a specific time period with interim performance expectations or, alternatively, it could be open-ended, with the expected time of arrival being a (dynamic) decision variable.

Subregions. The **prize region** is the smallest convex subset of the plane containing all the prizes. A **subregion** is a connected subset of the prize region. The subregion is the equivalent of the grid-cell in our generalized approach. The *subregions* component is a set of requirements concerning ways of travelling through subregions, either to get to a destination or for open ended harvesting. We specify which subregions either must not, should not, could, should or must be visited. We may specify precedence constraints between subregions, together with time windows. The reasoning is that we will often need to cover strategic requirements with respect to what is happening in the global scene and the opponent's possible alternative opportunities. This could also be interpreted as a general guideline or rough outline of alternatives for the final path, for which we must then fill in the details.

Hedging. The final modelling issue is **hedging**, which relaxes the strict sequencing considerations of contingent planning. Because we are planning the implementation of a strategic task, we must be aware that such a task could be defined loosely or exactly. Its nature could be a very tight "window of opportunity" or a "wait and see" exercise. These are extremes, but the *uncertainty* in the requirements contrasts with the precisely defined combinatorial optimization problems with which we are familiar. The effect on modelling subproblems is that the requirements of the task may change dynamically as we progress through the local planning period and that, often, we may be aware of possible scenarios for that change. The realism is that the guiding strategic process will be dynamically hedging against the opponent's activities on the global scene. We should plan to be able to adapt proactively to refinements or relaxations of requirements in that *satisficing*, robust foundations of paths which *cover* similar requirements (which may infer more urgency or greater prospects currently beyond reach) should be favoured over squeezing out every efficiency. Thus, the hedging component of a subproblem specification consists of any known or suspected alternative packages of requirements (possibly weighted or prioritised), or likely alternative scenarios for the dynamically evolving requirements specifications.

Having considered the principal tactical issues involved in planning for large problems, we now propose a solution paradigm which addresses these issues. Concomitant with the SPA/DMS, we wish to construct a scenario engine. In Chapter 6, a scenario corresponded to a sequence of prizes and in Chapter 7, a scenario corresponded to a sequence of clusters. We propose that a sequence of subregions of significant harvesting potential is the logical candidate for large problem

planning scenarios.

8.1.2 Identifying Promising Harvesting Subregions

The first stage in constructing a large problem scenario is to determine candidate subregions of significant harvesting potential. Rather than explicitly constructing a harvesting profile for some set of candidate subregions which cover the prize region, we construct a sample set of efficient sample harvesting (open) subpaths from the set of prizes. This circumvents the need to define subregions but a good sample harvesting subpath indicates that the local subregion around that path has good harvesting potential.

There are several possible harvesting paths we can sample from a subregion. Suppose we wish to maximize the harvesting rate. Possible constraints on the harvesting path include: lower or upper bounds on the subpath length; specified origin location or subregion; and specified destination location or subregion. Alternatively we could try to find a shortest or longest harvesting path which satisfies some goal harvesting rate.

Ideally, we wish to find those subregions which support a sustained period of good prize harvesting; this is established by sampling harvesting subpaths. For large subregions, i.e., those with many prizes, this should be sufficient. However, for small subregions, i.e., those with few prizes, the tradeoff between harvesting and travelling to the next subregion is more significant. We call those subregions which are good subregions for sustained harvesting **hotspots**. Similarly **coldspots** are those subregions which do not readily support prolonged harvesting but nevertheless may support quite good transportation routes for collecting the very good prizes along the way.

8.1.3 Sequencing of Subregions

The second stage in constructing a large problem scenario is to strategically consider sequences of hotspots. If there exist a number of large hotspots of comparable sustained harvesting potential then it is likely that both players will travel to one of those hotspots since, if necessary, any one of these hotspots is likely to be able to support both players in the medium-term, and the time taken to travel to a hotspot should be negligible compared to the time available for harvesting on that hotspot. In this case, the players need not be concerned with strategy, but merely concentrating on beginning to harvest in a good harvesting regions. However, as the hotspots diminish in size, i.e., as the length of time the hotspot can support good harvesting reduces, players must simultaneously consider the movement to other hotspots, between hotspots, and evaluating the harvesting potential under tactical conflict on a hotspot.

In summary, the key strategic questions are as follows. How long will a hotspot support individual harvesting? How does this compare with the time it would take to get there? Can the initial travel time be balanced with some harvesting along the way? Do we have enough information regarding the sustainability of the hotspot if it has to host two harvesting players as opposed to one, and, if so, is there a competitive advantage associated with getting there first and dominating? At what stage should we move on to another hotspot as the current hotspot's residual harvesting potential decreases?

8.2 Grid Structures

We now attempt to implement the ideas discussed in the previous section by proposing a strategic planning paradigm based on a division of the prize region into a regular grid. The idea is to determine an initial sequence of grid-cells at a coarse, strategic level of planning detail and to evaluate a candidate sequence of grid-cells by combining an evaluation of the *harvesting potential* of each individual grid-cell in the sequence under various scenarios for intra-grid-cell traversal.

8.2.1 Building a Grid Structure

There are several possible grid structures we could experiment with: the rectilinear grid, the overlapping grid, or the hierarchical subdivisions such as the *quadtree* (Mitchell [160] and Kambhampati and Davis [123]). We define a **grid-structure** as a regular, rectilinear partition of the prize region into non-overlapping rectangles. A grid-structure is therefore a plane partition and, in particular, is not a clustering of the prizes, although it does imply one (c.f. Section 7.1.1.3). The grid-structure parallels the family-cluster structure used in Chapter 7, although we do not assume that all prizes in a particular grid-cell will be collected when targeting that grid-cell.

Appendix 8.A.1 presents the details for constructing three specific grid-structures:

Static Grid. Partition the plane so that the boundary of the grid-structure is the smallest rectangle aligned with the existing axes which encloses the prizes and players. This grid-structure is static in the sense that, if the players are never outside the smallest rectangle aligned with the existing axes which encloses the prizes, then the grid-structure need never be updated.

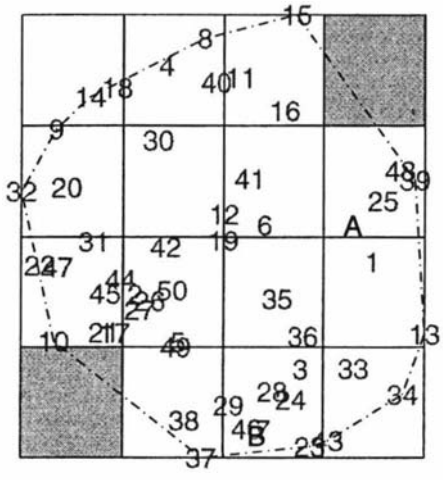
Single Player Dynamic Grid. Partition the plane so that the boundary of the grid-structure is the smallest rectangle aligned with the existing axes which encloses the prizes and players and is such that player $\mathcal{A}$ is located at the centre of a grid-cell. This grid-structure is dynamic in the sense that the prize membership of each grid-cell may change with the location of player $\mathcal{A}$.

Two Player Dynamic Grid. Partition the plane so that the boundary of the grid-structure is the smallest rectangle aligned with the existing axes which encloses the prizes and players and is such that both players are located near the centre of a grid-cell.

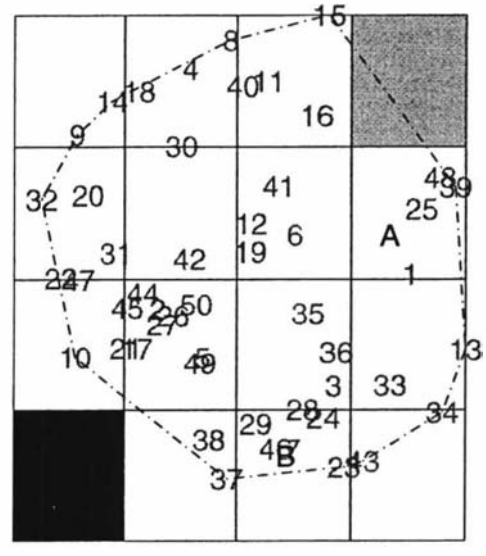
Figure 8.1 provides an example of each grid-structure on a sample problem instance in which a 4×4 grid is required. The convex hull of the prize and player locations is given by the '–·–·' line. A grid-cell is **legal** if it overlaps the *convex hull* (see, e.g., O'Rourke [170]) of the prize and player locations, although a legal grid-cell does not necessarily contain any prizes. The grid-cells that are not *legal* are blacked out and those *legal* grid-cells containing no prizes are lightly shaded.

8.2.2 Intra Grid-Cell Harvesting Potential

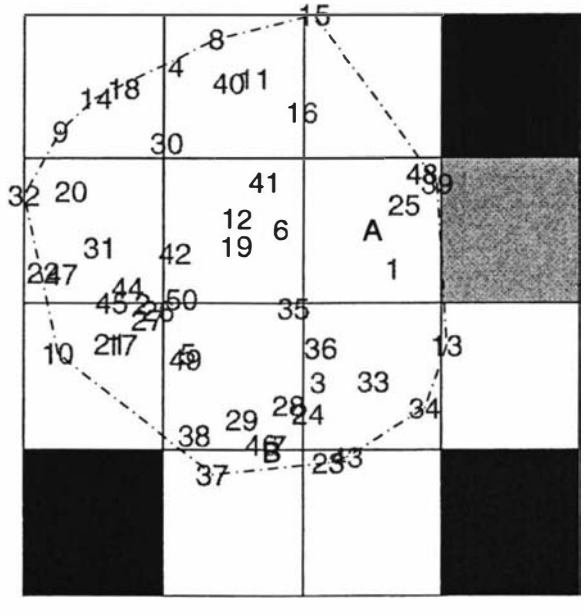
The harvesting potential of a grid-cell is the expected return in terms of prize value for investing time in harvesting the prizes within the grid-cell. One possible surrogate evaluation, which captures this idea, is the value per unit distance or *harvesting rate*. Considering an individual



(i). Static Grid

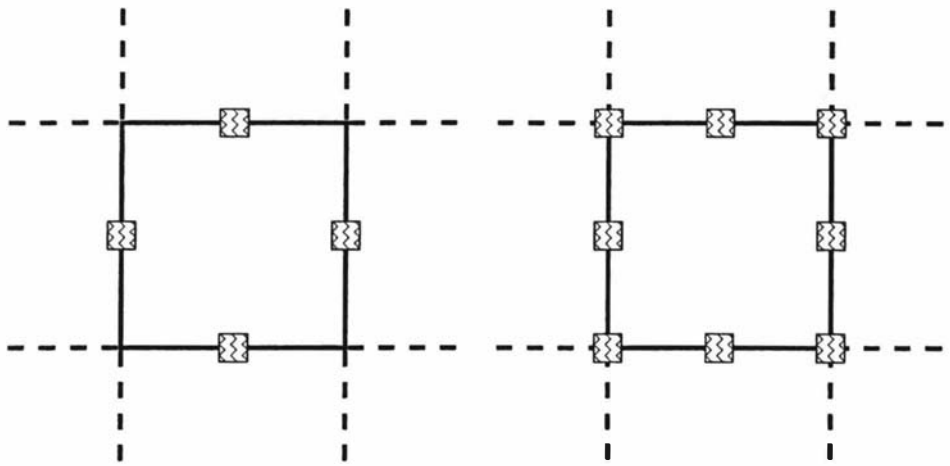


(iii). Two Player Dynamic Grid



(ii). Single Player Dynamic Grid

Figure 8.1: Examples of 4 × 4 grid-structures



(a) Four entry/exit locations (b) Eight entry/exit locations

Figure 8.2: Grid-cell entry/exit locations

grid-cell as part of a sequence of grid-cells suggests that we need to specify the entry and exit locations of a grid-cell.

8.2.2.1 Grid-cell Entry and Exit

In this section we examine the task of planning an intra-cell path, travelling from a specified entry location to a specified exit location. Computational complexity considerations (both in terms of sequences of grid-cells and number of procedure calls for an intra-cell path) encourage us to limit the number of entry/exit locations considered. Figure 8.2 shows two possibilities: four or eight entry/exit locations. Four entry/exit locations are the least that would be practical so that a player can enter from one of the four adjacent grid-cells. Also, diagonal movement can be approximated by a brief excursion into a vertical or horizontally adjacency grid-cell, perhaps claiming no prizes. Hence, we adopt the four entry/exit locations for each grid-cell.

8.2.2.2 Types of intra-cell path

We consider three types of intra grid-cell path.

Direct Path. A closed subpath of maximum value, from the entry location of a given grid-cell, through prizes from that grid-cell, to the exit location, of length at most α times the direct distance from the entry location to the exit location, for some parameter $\alpha \geq 1$.

Harvest Path. A closed subpath of maximum harvesting rate, from the entry location of a given grid-cell, through prizes from that grid-cell, to the exit location.

Open Harvest Path. An open subpath of maximum harvesting rate, from the entry location of a given grid-cell, through prizes from that grid-cell. Note that there is no specified exit location.

If there are no prizes in a grid-cell, then the first two paths are the direct passage from the entry-location to the exit-location and the open-harvest-path consists only of the entry-location. Section 3.3.2 gives further details for the direct-path and Section 3.3.3 for the harvest-path and open-harvest-path.

8.3 Strategic Grid Planning

A grid scenario is simply a sequence of adjacent grid-cells. We assume, for planning purposes, that a player moves from one grid-cell to an *adjacent, legal* grid-cell, where adjacency is defined as side-by-side, not diagonal. This section presents two methods for determining an initial sequence of grid-cells to target. The first, GRID-DT, is another application of the game-tree approach similar to ORIGINAL-DT, PRIZE-DT and CLUSTER-DT. The second, GRID-PATH, determines a sequence of grid-cells independent of the opponent, i.e., with no contingency.

8.3.1 Strategic Plan

A grid engine determines strategic targeting information in terms of an initial sequence of grid-cells, with an indication of how to traverse each grid-cell. Specifically, the strategic plan determined by the grid engines GRID-DT and GRID-PATH consists of:

(i). *$\mathcal{A}$-target-grid-cell*: The adjacent grid-cell to target next.

(ii). *$\mathcal{A}$-current-cell-type*: The type of path (intra-cell-harvest or intra-cell-direct) to traverse the current $\mathcal{A}$-grid-cell to the exit-location associated with the $\mathcal{A}$-target-grid-cell.

(iii). *$\mathcal{A}$-secondary-target-grid-cell*: The grid-cell adjacent to the $\mathcal{A}$-target-grid-cell to target following.

(iv). *$\mathcal{A}$-target-cell-type*: The type of path to traverse the $\mathcal{A}$-target-grid-cell to the exit-location associated with the $\mathcal{A}$-secondary-target-grid-cell.

Figure 8.3 illustrates the strategic plan for the most general case in which there are four adjacent grid-cells to the current $\mathcal{A}$-grid-cell (shaded darkly). The $\mathcal{A}$-target-grid-cell is selected from the lightly shaded grid-cells. The $\mathcal{A}$-secondary-target-grid-cell then determines which of the four exit-locations from the $\mathcal{A}$-target-grid-cell is selected. Overall, the strategic plan specifies one of the 16 exit-locations as a target location.

Response Assumption. Even though the player $\mathcal{B}$ equivalents may be explicitly or implicitly available, we *do not* consider response to observation of the opponent's target grid-cell. This is because such information would require a long period of observation to discern and, hence, not be sufficiently timely to enable a response.

Note. If an $\mathcal{A}$-target-grid-cell is not specified, then we determine an intra-cell-open-harvest-path for the current $\mathcal{A}$-grid-cell. Similarly, if an $\mathcal{A}$-target-grid-cell is specified but an $\mathcal{A}$-secondary-target grid-cell is not specified, then we determine an intra-cell-open-harvest-path for the $\mathcal{A}$-target-grid-cell.

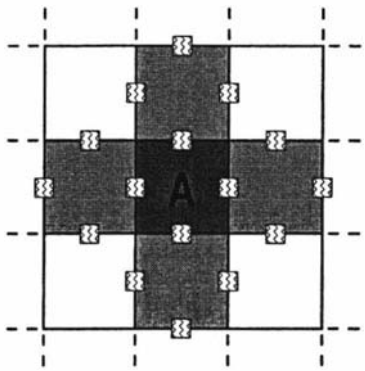


Figure 8.3: Generic Strategic Plan

8.3.2 Grid Engine: GRID-DT

The grid engine GRID-DT is structured around a game tree approach in which, at each game tree node, each player considers harvesting from, or directly traversing, a grid-cell.

The strategic engine CLUSTER-DT attempts to accurately sequence interactions of the players from cluster to cluster. Because of the size and the assumed lack of natural cluster structure of large problems, we do not attempt this. Instead, exploration terminates when the players meet on the same grid-cell and so the aim of GRID-PATH is to coarsely plan the convergence of the players up until local conflict can possibly occur on a grid-cell.

8.3.2.1 Definitions

A grid-planning-path, $\mathbb{G}_A$, associated with player $\mathcal{A}$, consists of a sequence of grid-cells, a projected location A and a projected time-stamp t_A. Similarly define $\mathbb{G}_B$ for player $\mathcal{B}$. A prize-planning-path, P_A, associated with $\mathbb{G}_A$, consists of the sequence of prizes and entry/exit locations visited from the sequence of grid-cells visited in $\mathbb{G}_A$.

A general game tree node j is defined the following three attributes.

- A pair of grid-planning-paths $(\mathbb{G}_A, \mathbb{G}_B)$.
- The current grid-cell and entry-location of each player.
- The values $v(P_A)$ and $v(P_B)$ and projected time-stamps t_A and t_B corresponding to the arrival times at the ends of P_A and P_B respectively.

The game tree root node is defined by: $\mathbb{G}_A = \emptyset$ and $\mathbb{G}_B = \emptyset$; the entry location of each player is that player's current location; the current grid-cell of each player is the grid-cell in which that player is currently located; all legal grid-cells are available, $P_A = \emptyset$ and $P_B = \emptyset$.

Consider player $\mathcal{A}$ and a remaining, legal grid-cell (i, j) adjacent to the current $\mathcal{A}$-grid-cell which player $\mathcal{A}$ is about to enter. The operation "$\mathcal{A} \rightarrow \mathbb{G}_A \oplus_D (i, j)$" defines a child game tree node with:

- The centre of the common edge of the two grid-cells as the entry-location to grid-cell (i, j) and hence the exit-location of the current $\mathcal{A}$-grid-cell.
- P_A appended with the intra-cell-direct-path from entry to exit through the current $\mathcal{A}$-grid-cell.

Similarly the operation "$\mathcal{A}\rightarrow\mathbb{G}_A \oplus_H (i, j)$" defines a child game tree node by appending the intra-cell-harvest-path, from entry to exit through the current $\mathcal{A}$-grid-cell, to P_A.

Direct-only target-grid-cell. If the intra-cell-harvest-path has harvesting-rate less than some parameter hr_{thresh}, or has length and harvesting-rate at most that of the intra-cell-direct-path, then the target-grid-cell is denoted *direct-only*.

Early target-grid-cell. An $\mathcal{A}$-target-grid-cell is denoted *direct-early* if the $\mathcal{A}$-intra-cell-direct-path, through the current $\mathcal{A}$-grid-cell to the exit-location defined by the $\mathcal{A}$-target-grid-cell, arrives at its exit-location no later than t_B. An $\mathcal{A}$-target-grid-cell that is not direct-only is denoted *harvest-early* if the $\mathcal{A}$-intra-cell-harvest-path, through the current $\mathcal{A}$-grid-cell to the exit-location defined by the $\mathcal{A}$-target-grid-cell, arrives at its exit-location no later than t_B. Similarly for $\mathcal{B}$-target-grid-cells.

8.3.2.2 Game Tree Generation

A game tree node is **terminal** if the players share the same current grid-cell, share the same location, one player has no remaining, legal, adjacent grid-cells, or the game tree has reached a pre-specified maximum depth. We now define how to generate the game tree by showing how to generate the children of any non-terminal game tree node j.

Expansion rule (G1).

- For each remaining, legal direct-early $\mathcal{A}$-target-grid-cell (i_A, j_A) adjacent to the current $\mathcal{A}$-grid-cell: $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_D (i_A, j_A)$.
- For each remaining, legal harvest-early $\mathcal{A}$-target-grid-cell (i_A, j_A) adjacent to the current $\mathcal{A}$-grid-cell: $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_H (i_A, j_A)$.
- For each remaining, legal direct-early $\mathcal{B}$-target-grid-cell (i_B, j_B) adjacent to the current $\mathcal{B}$-grid-cell: $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_D (i_B, j_B)$.
- For each remaining, legal harvest-early $\mathcal{B}$-target-grid-cell (i_B, j_B) adjacent to the current $\mathcal{B}$-grid-cell: $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_H (i_B, j_B)$.
- For each remaining, legal $\mathcal{A}$-target-grid-cell (i_A, j_A) adjacent to the current $\mathcal{A}$-grid-cell that is either not direct-early or not harvest-early and for each remaining, legal $\mathcal{B}$-target-grid-cell (i_B, j_B) adjacent to the current $\mathcal{B}$-grid-cell that is either not direct-early or not harvest-early:
  - if $\mathcal{A}$-target-grid-cell is not direct-early and $\mathcal{B}$-target-grid-cell is not direct-early: $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_D (i_A, j_A)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_D (i_B, j_B)$.

- if $\mathcal{A}$-target-grid-cell is not direct-only and not harvest-early and $\mathcal{B}$-target-grid-cell is not direct-early:
  $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_H (i_A, j_A)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_D (i_B, j_B)$.
- if $\mathcal{A}$-target-grid-cell is not direct-early and $\mathcal{B}$-target-grid-cell is not direct only and not harvest-early:
  $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_D (i_A, j_A)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_H (i_B, j_B)$.
- if both target-grid-cells are not direct-only and not harvest-early:
  $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_H (i_A, j_A)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_H (i_B, j_B)$.

8.3.2.3 Searching the Game Tree

The objective is to determine the sequence of grid-cells of maximum harvesting potential in a MAXIMIN sense. The evaluation, $hr(t)$, of a terminal game tree node t is:

- If the current grid-cells of the players are the same or adjacent, then determine a $\mathcal{ABA}$-path (see Section 3.3.4) on the current $\mathcal{A}$-grid-cell and, if this has harvesting-rate at least hr_{thresh}, then append it to P_A. Then $hr(t) = \frac{v(P_A)}{t_A}$.
- Otherwise determine an $\mathcal{A}$-intra-cell-open-harvest-path on the current $\mathcal{A}$-grid-cell and, if this has harvesting-rate of at least hr_{thresh}, then append it to P_A. Then $hr(t) = \frac{v(P_A)}{t_A}$.

The evaluation $hr(i)$ of a non-terminal game tree node i is the either:

- The MAXIMIN evaluation of the non-early child nodes if $\nexists$ early target-grid-cells.
- The maximum of the $\mathcal{A}$-early target-grid-cells and the MAXIMIN evaluation of the non-early child nodes if $\exists$ $\mathcal{A}$-early target-grid-cells.
- The minimum of the $\mathcal{B}$-early target-grid-cells and the MAXIMIN evaluation of the non-early child nodes if $\exists$ $\mathcal{A}$-early target-grid-cells.

8.3.2.4 Example of the GRID-DT Game Tree Generation

Figure 8.4 illustrates a snapshot of a GRID-DT search scenario. Player $\mathcal{A}$ starts in grid-cell $(1,3)$ and player $\mathcal{B}$ starts in grid-cell $(3,1)$. Player $\mathcal{A}$'s adjacent, remaining, legal grid-cells are $\{(1,2),(2,3)\}$ and player $\mathcal{B}$'s are $\{(2,1),(3,2)\}$. Suppose $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_D (1,2)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_H (3,2)$, illustrated by the solid arrows emanating from the player locations. The dotted arrows correspond the possible other branches from that location. Now player $\mathcal{A}$'s adjacent, remaining, legal grid-cells are $\{(1,1),(2,2)\}$ and player $\mathcal{B}$'s are $\{(2,2),(3,3)\}$. Suppose $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_H (1,1)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_H (2,2)$, again illustrated by the solid arrows and the dotted arrows correspond to the alternative possible branches. Now player $\mathcal{A}$'s adjacent, remaining, legal grid-cell is $(2,1)$ and player $\mathcal{B}$'s are $\{(2,1),(2,3)\}$ but not $(1,2)$ since has been visited by player $\mathcal{A}$. Suppose $\mathcal{A}\rightarrow\mathbb{G}_A \oplus_D (2,1)$ and $\mathcal{B}\rightarrow\mathbb{G}_B \oplus_D (2,1)$, again illustrated by the solid arrows and the dotted arrow corresponds to the alternative possible branch. Now the players are to conflict on grid-cell $(2,1)$ so we fathom the search using a $\mathcal{ABA}$-path on the shaded grid-cell.

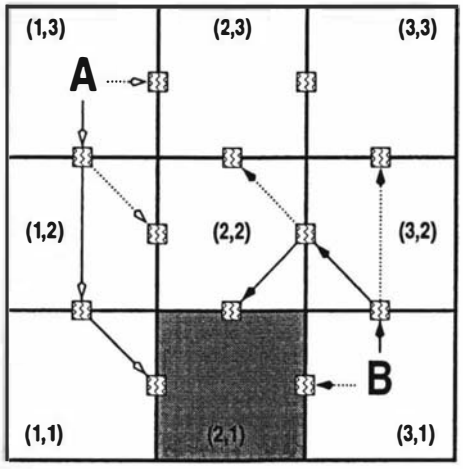


Figure 8.4: Example scenario of a GRID-DT search.

8.3.3 Grid Engine: GRID-PATH

The grid engine GRID-PATH is structured around a decision tree, since we plan independently of the opponent. To be able to apply GRID-PATH to finer grid-structures than is possible with GRID-DT, we wish to reduce the breadth of the decision tree as much as possible. Since we do not need to compare the arrival times at a grid-cell entry-location with the corresponding equivalent of the opponent, we can manage multiple prize-planning-paths to evaluate a single cluster-planning-path and, at each decision node, we need branch at most once to each adjacent grid-cell rather than at most once each for intra-cell-harvest and intra-cell-direct possibilities.

8.3.3.1 Definitions

A grid-planning-path, $\mathbb{G}_A$, associated with player $\mathcal{A}$, consists of a sequence of grid-cells and a projected location A. Three prize-planning paths are associated with $\mathbb{G}_A$:

Last-direct. Subpath such that previous intra-grid-cell traversal was intra-cell-direct.

Last-harvest. Subpath such that previous intra-grid-cell traversal was intra-cell-harvest.

All-direct. Subpath such that all previous intra-grid-cell traversals were all intra-cell-direct.

A general decision tree node j is defined by the following three attributes.

- A grid-planning path, $\mathbb{G}_A$.
- The current grid-cell and entry-location.
- The value and length of the *last-direct*, *last-harvest* and *all-direct* prize-planning-paths.

The decision tree root node is defined by: $\mathbb{G}_A = \emptyset$; the entry location is player $\mathcal{A}$'s current location; the current grid-cell is the grid-cell in which player $\mathcal{A}$ is currently located; all legal grid-cells are available; and the *last-direct*, *last-harvest* and *all-direct* paths are empty.

Consider a remaining, legal grid-cell (i, j) adjacent to the current $\mathcal{A}$-grid-cell which player $\mathcal{A}$ is about to enter. The operation $\mathcal{A}{\rightarrow}\mathbb{G}_A \oplus_P (x, y)$ defines a child decision node as follows.

- The centre of the common edge of the two grid-cells as the entry-location to grid-cell (i, j) and hence the exit-location of the current $\mathcal{A}$-grid-cell.
- Determine the intra-cell-direct path and intra-cell-harvest path (if not direct-only) and update the last-direct, last-harvest and all-direct planning-paths as follows.
  - The new last-direct is the intra-cell-direct appended to either the last-direct, last-harvest or all-direct so that the overall harvesting rate is maximized.
  - If direct-only then new last-harvest is the same as new last-direct; otherwise, new last-harvest is the intra-cell-harvest appended to either the last-direct, last-harvest or all-direct so that the overall harvesting rate is maximized.
  - The new all-direct is the intra-cell-direct appended to all-direct.
  - The new entry-location is the current exit-location.

8.3.3.2 Decision Tree Generation

A decision tree node is **terminal** if player $\mathcal{A}$ has no remaining, legal, adjacent grid-cells, or the decision tree has reached a pre-specified maximum depth. The maximum decision tree depth acts as a planning-horizon since we always consider the adjacent, legal, remaining grid-cells to branch to even if they give a worse harvesting-rate. We now define how to generate the game tree by showing how to generate the children of any non-terminal game tree node j.

Expansion rule (D1).

- For each remaining, legal $\mathcal{A}$-target-grid-cell (i_A, j_A) adjacent to the current $\mathcal{A}$-grid-cell: $\mathcal{A}{\rightarrow}\mathbb{G}_A \oplus_P (i_A, j_A)$.

8.3.3.3 Searching the Decision Tree

The objective is to determine the sequence of grid-cells of maximum harvesting potential.

For a terminal game tree node t, the evaluation $hr(t)$ is determined as follows. Construct the intra-cell open-harvest path on the current $\mathcal{A}$-grid-cell. If this has harvesting-rate of at least hr_{thresh}, then $hr(t)$ is a harvesting-rate of the best path formed by appending the intra-cell-open-harvest path to the current last-harvest, last-direct or all-direct path. Also, record which of last-harvest, last-direct or all-direct was selected.

Evaluate a non-terminal decision node j as $hr(j)$ by determining the remaining, adjacent, legal target-grid-cell of maximum evaluation and assuming its value. For that selected target-grid-cell determine which of last-harvest, last-direct or all-direct is implied by the associated path-type recorded in evaluating the target-grid-cell, and record this with the current $\mathcal{A}$-grid-cell.

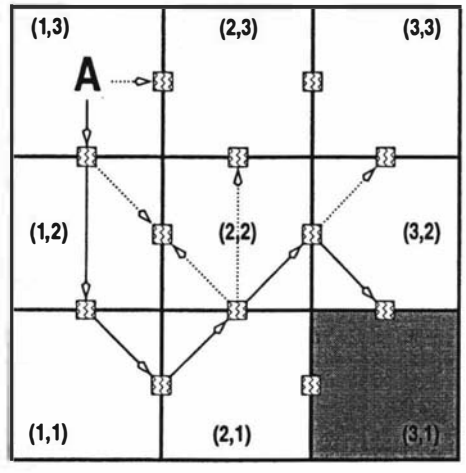


Figure 8.5: Example scenario of a GRID-PATH search.

8.3.3.4 Example of the GRID-PATH Decision Tree Generation

Figure 8.5 illustrates a snapshot of a GRID-PATH search scenario. Player $\mathcal{A}$ starts in grid-cell $(1,3)$ with adjacent grid-cells $\{(1,2),(2,3)\}$. Suppose $\mathcal{A}{\rightarrow}\mathbb{G}_A \oplus_P (1,2)$ illustrated by the solid arrows eminating from the player $\mathcal{A}$. The dotted arrow corresponds to the other possible target-grid-cell. Now player $\mathcal{A}$'s adjacent, remaining grid-cells are $\{(1,1),(2,2)\}$. Suppose $\mathcal{A}{\rightarrow}\mathbb{G}_A \oplus_P (1,1)$ and so on until player $\mathcal{A}$ has arrived at the boundary of $(3,2)$ and $(3,1)$ where there are no remaining, adjacent target-grid-cells.

8.4 Tactical Grid Planning

Suppose GRID-DT or GRID-PATH has provided a strategic plan, i.e., a target-grid-cell, a current-cell-type, a secondary-target-grid-cell and a target-cell-type for player $\mathcal{A}$. We now present methods to determine tactics which implement a strategic plan.

Let m be the Manhattan number of grid-cells between the players' current grid-cells, e.g., $m = 4$ in Figure 8.6. If the players are located in the same grid-cell then $m = 0$. If the players are in adjacent grid-cells then $m = 1$. We may apply a different tactical method depending on the value of m.

Case $m \geq 2$. Construct a *harvest-path* on the current $\mathcal{A}$-grid-cell and $\mathcal{A}$-target-grid-cell.

Case $m = 1$. Construct an $\mathcal{ABA}$*-path* on the current $\mathcal{A}$-grid-cell and $\mathcal{A}$-target-grid-cell (but see below) if there are sufficiently few prizes in these grid-cells; otherwise construct a *harvest-path*.

Case $m = 0$. Apply PRIZE-DT on the current $\mathcal{A}$-grid-cell and $\mathcal{A}$-target-grid-cell if there are sufficiently few prizes in these grid-cells; otherwise construct an $\mathcal{ABA}$*-path* or *harvest-path*.

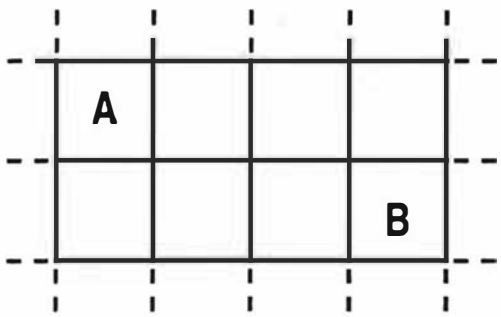


Figure 8.6: Manhattan grid-cell distance: $m = 4$.

8.4.1 Constructing a Harvest Path

We wish to construct a *harvest-path* P_A. If no $\mathcal{A}$-target-grid-cell is specified, then let P_A be the intra-cell open-harvest-path on the current grid-cell. Hence, suppose that a $\mathcal{A}$-target-grid-cell is specified.

Let P_A be an intra-cell harvest-path or intra-cell direct-path, as specified by current-cell-type, through the current grid-cell to the exit-location defined by the target-grid-cell.

- If no secondary-grid-cell is specified, then append to P_A an intra-cell open-harvest-path on the target-grid-cell.
- Otherwise, append to P_A an intra-cell harvest-path or intra-cell direct-path, as specified by target-cell-type, through the target-grid-cell to the exit-location defined by the secondary-grid-cell.

Remove the entry/exit-location between the grid-cells. Apply a local search heuristic to improve the harvesting rate of P_A such that the length of P_A cannot be increased above its original length (before the entry/exit-location between the grid-cells was removed) and the value of P_A cannot be decreased below its original value. Note that there is no precedence constraint between the prizes in the two grid-cells.

8.4.2 Constructing an $\mathcal{ABA}$-path

We only construct an open or closed *$\mathcal{ABA}$-path* (see Section 3.3.4) when the opponent is in the same grid-cell or in an adjacent grid-cell. The four cases are illustrated in Figure 8.7, in which the arrow indicates the $\mathcal{A}$-target-grid-cell.

In determining which prizes to consider in the construction of an $\mathcal{ABA}$-path, at least two points are valid. One argument is that player $\mathcal{A}$ is conservative and hence, from its perspective, player $\mathcal{B}$ would attempt to intercept directly if possible. Hence, in cases (ii) and (iii) of Figure 8.7, player $\mathcal{B}$ would not visit any prizes from the current $\mathcal{B}$-grid-cell. A second argument is that player $\mathcal{B}$ could increase its overall harvesting rate by collecting prizes from the current $\mathcal{B}$-grid-cell on route to the current $\mathcal{A}$-grid-cell or $\mathcal{A}$-target-grid-cell. Since this second point is more general, we

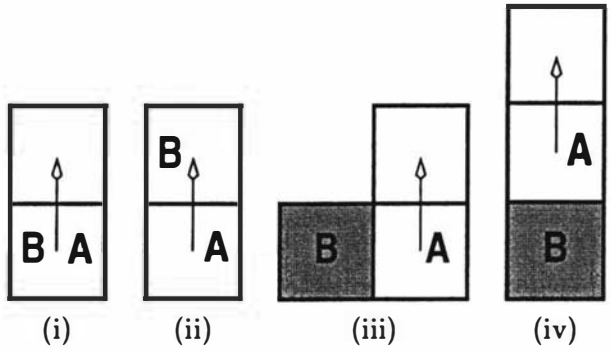


Figure 8.7: Four types of adjacency considerations

adapt the $\mathcal{ABA}$-PATH subproblem of Section 3.3.4 to incorporate prizes which are not available to player $\mathcal{A}$. These $\mathcal{B}$-only prizes correspond to the shaded grid-cells in Figure 8.7.

If the $\mathcal{A}$-target-grid-cell is specified, then a closed $\mathcal{ABA}$-path is required, i.e., the exit-location from the $\mathcal{A}$-target-grid-cell (if a $\mathcal{A}$-secondary-target-grid-cell is specified) or the exit-location from the current $\mathcal{A}$-grid-cell (otherwise) is the fixed destination for $\mathcal{ABA}$-path.

8.4.3 Applying PRIZE-DT

We apply PRIZE-DT only when the players are located in the same current grid-cell, and hence, there is the potential for local conflict, and there are sufficiently few prizes in the current grid-cell and the $\mathcal{A}$-target-grid-cell. The assumption is that this local conflict will overflow into the $\mathcal{A}$-target-grid-cell; we do not consider any $\mathcal{B}$-target-grid-cell. If there is no specified $\mathcal{A}$-target-grid-cell, then we apply PRIZE-DT on the current grid-cell only.

8.5 Dynamic Monitoring: Grid-DMS

A Small DMS (Dynamic Monitoring System) for small problems was designed in Section 6.4 and a Medium DMS for medium problems was designed in Section 7.6. In this section we design a Grid DMS based upon the grid constructions, grid engines and tactical subpath methods presented thus far.

8.5.1 Outline of Grid-DMS

Figure 8.8 illustrates the relationship between the monitors and frames in a Grid DMS. At the *global-frame* we apply a GRID-MONITOR, which constructs a grid-structure and a *prize-frame* consisting of an $\mathcal{A}$-target-grid-cell, an $\mathcal{A}$-current-cell-type, an $\mathcal{A}$-secondary-target-grid-cell, and an $\mathcal{A}$-target-cell-type. The GRID-MONITOR makes no use of any equivalent of a target-set, since it is not responsive to the observation of the opponent, but only to the opponent's current location and strategic possibilities. At the *prize-frame* we apply a PRIZE-MONITOR to determine a *step-frame* which either consists of a prize-ts and a target-prize or target-pair or consists of a prize

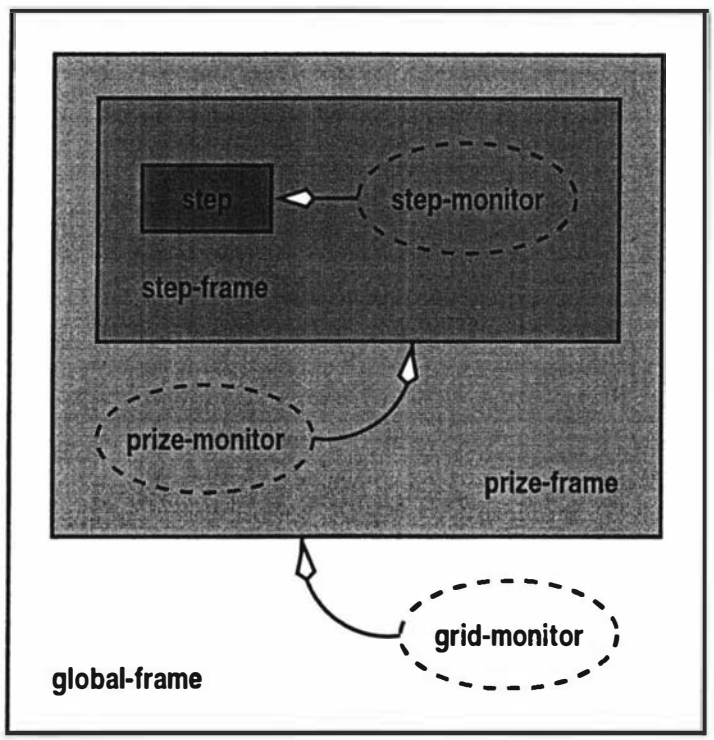


Figure 8.8: Monitors and Frames for a Grid-DMS

planning-path. At the *step-frame* we apply a STEP-MONITOR which determines a *step*.

8.5.2 Grid-DMS Specification

Algorithms 8.1–8.4 present a simple Grid-DMS. We do not wish to activate the GRID-MONITOR unnecessarily since the grid engine is assumed to be the most computationally expensive component of the grid DMS.

To implement the dynamism of the grid-structure, without updating it too often, we wait until a player moves across a grid-line. Only then is the grid-structure rebuilt, GRID-DT or GRID-PATH applied, and the type of tactical subpath method revised. Because of this we recommend a *single player dynamic grid* for GRID-PATH and *two player dynamic grid* for GRID-DT.

We also make use of the same PRIZE-TS building, refinement and checking to construct a *target-set-$\mathcal{ABA}$-path*, as was used for the Small-DMS.

Let Q be the set of remaining prizes. When using a *harvest-path* this defines the step-frame with no prize-ts. The *harvest-path* is not necessarily updated when a prize is claimed and, hence, the target-prize is the first available prize on the *harvest-path*. It is possible that a prize-frame consists of a sequence of two intra-cell-direct paths neither of which contain any prizes, in which case we go directly towards the exit-location from the $\mathcal{A}$-target-grid-cell.

Algorithm 8.1 monitor GRIDDMS-GRID-MONITOR

// Build new grid-structure.
Construct static, single player dynamic or two player dynamic grid-structure.

// Build prize-frame.
Apply GRID-DT or GRID-PATH.

end

Algorithm 8.2 monitor GRIDDMS-PRIZE-REFINE

// Refine current prize-ts.
REFINE-PRIZETS
if *prize-ts* $= \emptyset$ **then**
    // Build new step-frame.
    GRIDDMS-PRIZE-MONITOR
else if *change made to prize-ts* **then**
    // Update tactical response.
    Determine target-prize or target-pair.
else
    // No changes made to step-frame.
end

end

Algorithm 8.3 monitor GRIDDMS-PRIZE-MONITOR

```
    // Check if prize-frame exists and is still valid.
    if ∄ prize-frame or player moved outside current grid cell then
        // Build new prize-frame.
        GRIDDMS-GRID-MONITOR
    end

    // Build new step-frame.
    Delete prize-ts.
    Delete harvest-path.
    m ← Manhattan grid-cell distance between players.
    if (m = 0) then
        if ∄ step-frame or either prize just claimed then
            Build PRIZE-DT based prize-mts.
            Select prize-ts from prize-mts.
        else
            Build generic prize-ts.
        end
        Determine target-prize or target-pair.
    else if m=1 then
        Build generic prize-ts.
        Construct target-set-ABA-path. Select the first prize on A-path as target-prize.
    else
        Construct harvest-path.
        Select the first prize on harvest-path as target-prize.
    end

end
```

Algorithm 8.4 monitor GRIDDMS-STEP-MONITOR

```
// Check if step-frame exists and is still valid.
if ∄ step-frame or player moved outside current grid cell then
    GRIDDMS-PRIZE-MONITOR // Build new step-frame
else if prize just claimed then
    if ∄ harvest-path or harvest-path ∩Q ≠ ∅ then
        GRIDDMS-PRIZE-MONITOR // Build new step-frame
    end
else
    CHECK-PRIZETS
    if prize-ts invalid then
        GRIDDMS-PRIZE-MONITOR // Build new step-frame.
    else
        GRIDDMS-PRIZE-REFINE // Refine current step-frame.
    end
end

// Determine step.
if ∃ harvest-path then
    target ← first remaining prize on harvest-path.
else if ∃ target-prize then
    target ← target-prize.
else
    // Target target-prize-pair {x1, x2}.
    if (|prize-ts| = 1) then
        if (prize-ts ⊂ {x1, x2}) then
            TWO-PRIZE-BIAS
        else
            THREE-PRIZE-WINDOW
        end
    else
        COMPROMISE-WINDOW.
    end
end

end
```

Coda

▼ Summary

In this chapter we have considered an approach to strategic planning for large problems and we have designed of Grid-DMS which implements this approach.

▼ Link

This is the end of Part II in which we have designed a number of strategies which implement the SPA/DMS of Chapter 4 for different problems sizes. In Part III we consider the computational testing of these strategies, together with those from Chapter 3, and the design of problem instances.

8.A Appendix to Chapter 8

8.A.1 Constructing a Grid-Structure

Further to Section 8.2.1 we present details of how to construct the three rectilinear grid-structures. Suppose we wish to create a grid of size $n_y \times n_x$. Let

$$\begin{aligned} x_{min} &= \min_{j \in V} x_j & x_{MIN} &= \min\{x_{min}, x_A, x_B\} \\ x_{max} &= \max_{j \in V} x_j & x_{MAX} &= \max\{x_{max}, x_A, x_B\} \\ y_{min} &= \min_{j \in V} y_j & y_{MIN} &= \max\{y_{min}, y_A, y_B\} \\ y_{max} &= \max_{j \in V} y_j & y_{MAX} &= \max\{y_{max}, y_A, y_B\} \end{aligned}$$

8.A.1.1 Static Grid

Define

$$\delta x = \frac{x_{MAX} - x_{MIN}}{n_x}$$
$$\delta y = \frac{y_{MAX} - y_{MIN}}{n_y}$$

so that the grid-lines are at

$$x_{MIN} + i\delta x \ \forall i \in \{0, \ldots, n_x\}$$
$$y_{MIN} + i\delta y \ \forall i \in \{0, \ldots, n_y\}$$

Figure 8.9 illustrates the construction of a static grid considering either the x-coordinate or the y-coordinate independently.

8.A.1.2 Single Player Dynamic Grid

Consider the x-coordinate where n_x grid-cells are required such that player $\mathcal{A}$ is located at the centre of a grid-cell and grid-cells are as small as possible. We wish to determine the minimum grid-cell width, δx, satisfying

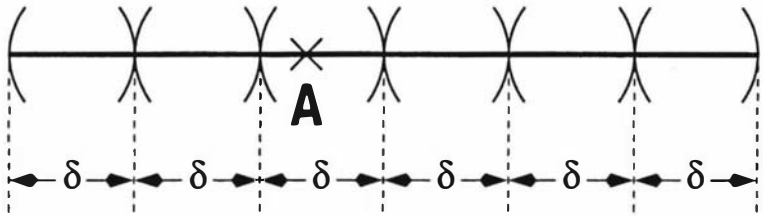


Figure 8.9: Construction of Static Grid

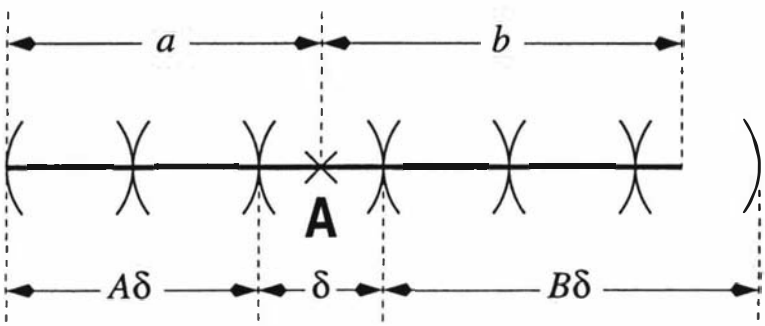


Figure 8.10: Construction of Single Player Dynamic Grid

$$\left\lceil \frac{\max\{0, \max\{x_{max}, x_B\} - x_A\}}{\delta x} - \tfrac{1}{2} \right\rceil + \left\lceil \frac{\max\{0, x_A - \min\{x_{min}, x_B\}\}}{\delta x} - \tfrac{1}{2} \right\rceil = n_x - 1. \tag{8.1}$$

Then

$$\begin{aligned} \delta x &= \frac{x_A - x_{MIN}}{n_x - \frac{1}{2}} \quad \text{if } x_A \geq \max\{x_{max}, x_B\} \\ \delta x &= \frac{x_A - x_{MIN}}{n_x - \frac{1}{2}} \quad \text{if } x_A \leq \min\{x_{min}, x_B\} \end{aligned}$$

Otherwise suppose $\min\{x_{min}, x_B\} < x_A < \max\{x_{max}, x_B\}$ and let $a = x_A - \min\{x_{min}, x_B\}$ and $b = \max\{x_{max}, x_B\} - x_A$.

We now present an algorithm for finding the minimum δx which satisfies (8.1). In Figure 8.10 we approximate the ratio $a : b$ by $A + \frac{1}{2} : B + \frac{1}{2}$ such that $A + B + 1 = n_x$ and A and B are non-negative integers. To cover the prizes requires that $(A + \frac{1}{2})\delta x \geq a$ and $(B + \frac{1}{2})\delta x \geq b$. Hence, we require a non-negative integer A to minimize

$$\delta x = \max\left\{ \frac{a}{A + \frac{1}{2}}, \frac{b}{n_x - A - \frac{1}{2}} \right\}. \tag{8.2}$$

Since $\frac{a}{A+\frac{1}{2}}$ is decreasing with A and $\frac{b}{n_x - A - \frac{1}{2}}$ is increasing with A we conclude that $A \in \{\lfloor z \rfloor, \lceil z \rceil\}$ where z satisfies

$$\frac{a}{z + \frac{1}{2}} = \frac{b}{n_x - z - \frac{1}{2}}$$

i.e.,

$$z = \frac{a n_x}{a + b} - \tfrac{1}{2}. \tag{8.3}$$

If $z \leq 0$ then let $A = 0$ and $\delta x = \max\{2a, \frac{b}{n_x - \frac{1}{2}}\}$. Otherwise we choose $A \in \{\lfloor z \rfloor, \lceil z \rceil\}$ which minimizes (8.2). Since $b > 0$,

$$\lceil z \rceil \leq \left\lfloor \frac{a}{a+b} n_x \right\rfloor \leq n_x - 1$$

and hence $B = n_x - 1 - A \geq 0$ as required.

Finally define the grid-lines at

$$x_A + (i - \tfrac{1}{2})\delta x \ \forall i \in \{-A, \ldots, B+1\}.$$

Note. It is reasonable that a good approximation of $a : b$ by $A + \frac{1}{2} : B + \frac{1}{2}$ should minimize

$$\begin{aligned} & \left| b(A + \tfrac{1}{2}) - a(B + \tfrac{1}{2}) \right| \\ = \ & \left| (a+b)(A + \tfrac{1}{2}) - a n_x \right| \end{aligned} \tag{8.4}$$

whose zero, z, also satisfies (8.3). However, the $A \in \{\lfloor z \rfloor, \lceil z \rceil\}$ that minimizes (8.4) does not necessarily minimize (8.2).

Consider the y-coordinate similarly, where we require n_y grid-cells. If square grid-cells are desired, then take $\max\{\delta x, \delta y\}$ as the grid-cell width and height.

8.A.1.3 Two Player Dynamic Grid

Consider the x-coordinate where n_x grid-cells are required. Let δx be the width of a grid-cell. We require that player $\mathcal{A}$ and player $\mathcal{B}$ are each located within $\frac{1}{k}\delta x$ of the centre of a grid-cell and that the grid-cells are as small as possible; the parameter $k > 2$ determines the required "nearness" to the centre of a grid-cell. If $x_A = x_B$ then we can determine δx as in the single player dynamic grid. Hence suppose, without loss of generality, that $x_A < x_B$.

Let

$$\begin{aligned} a &= x_A - x_{MIN} \\ b &= x_B - x_A \\ c &= x_{MAX} - x_B \end{aligned}$$

In Figure 8.11 we approximate the ratio $a : b : c$ by $A + \frac{1}{2} : B + 1 : C + \frac{1}{2}$, such that $A + B + C = n_x - 2$ and A and C are non-negative integers and B is an integer ≥ -1. Note that $B = 0$ implies that the players will be in adjacent grid-columns (along the x-coordinate) and $B = -1$ implies that the players will be in the same grid-column. All (A, B, C) triples can be enumerated.

Given (A, B, C) we must determine δx. Introduce "slack variables" s_1 and s_2 to position the grid-lines, as in Figure 8.12, such that

$$\left(B + 1 + \tfrac{2}{k}\right) \delta x = s_1 + b + s_2 \tag{8.5}$$

and

$$0 \leq s_1 \leq \tfrac{2}{k}\delta x \tag{8.6}$$

$$0 \leq s_2 \leq \tfrac{2}{k}\delta x \tag{8.7}$$

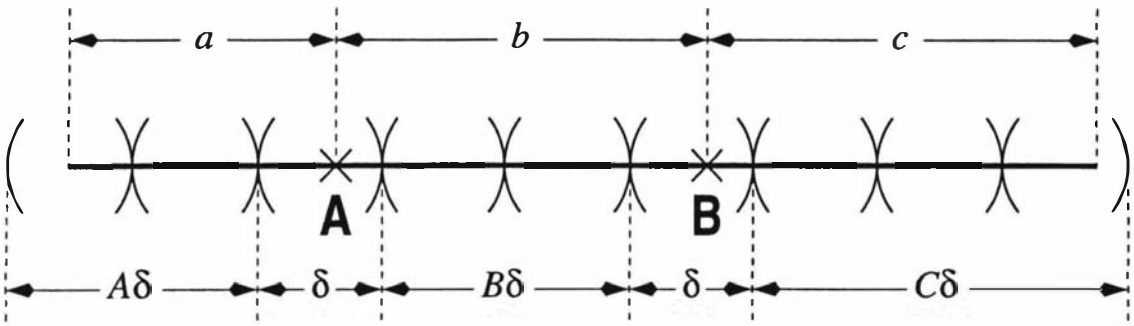


Figure 8.11: Construction of Two Player Dynamic Grid

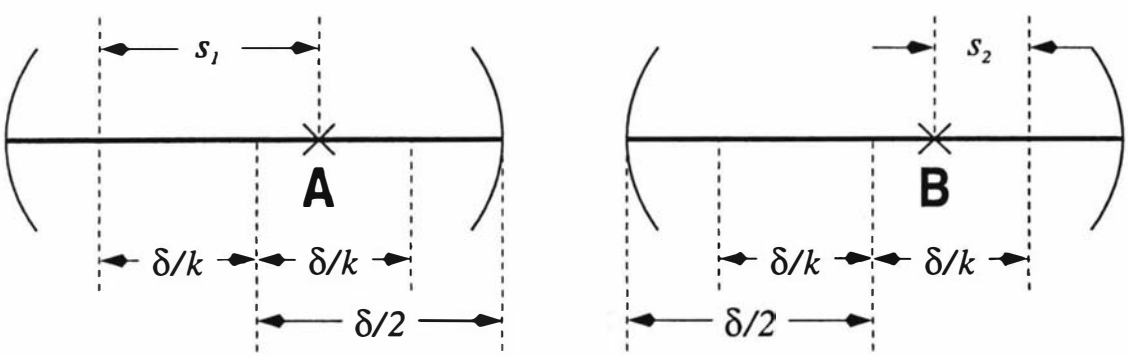


Figure 8.12: Positioning Players for Two Player Dynamic Grid

Finally we need to cover all the prizes.

$$(A + \tfrac{1}{2} - \tfrac{1}{k})\delta x + s_1 \geq a \tag{8.8}$$
$$(C + \tfrac{1}{2} - \tfrac{1}{k})\delta x + s_2 \geq c \tag{8.9}$$

Hence we have the following Linear Program:

(LP)

$$\min \delta x$$

subject to (8.5)-(8.9) in which the decision variables are $\{\delta x, s_1, s_2\}$.

The LP may be infeasible since not every (A, B, C) can simultaneously satisfy the player positioning constraints and the prize covering constraints.

Over all (A, B, C) such that the LP is feasible we select an (A, B, C) that minimizes δx. Note that an LP corresponding to $B = -1$ must be feasible since $s_1 = s_2$ and a suitably large δx satisfies the constraints. Hence a feasible (A, B, C) is available.

Finally, define the grid-lines at

$$x_A - s_1 + (i - A - \tfrac{1}{2} + \tfrac{1}{k})\delta x \ \forall i \in \{0, \ldots, n_x\}.$$

Consider the y-coordinate similarly, where we require n_y grid-cells. Square grid-cells would be difficult to construct since we must satisfy the requirement that each player is located within $\frac{1}{k}\delta$ of the centre of a grid-cell in each coordinate.

Part III

Computational Evaluation

Overview of Part III
Computational Evaluation

There are two major CPCP research questions which can only be meaningfully addressed via computational experimentation.

Understanding Problem Instances. What makes a problem instance difficult, and how can we design difficult test problem instances? We need to define a number of problem classes, and determine which classes are seemingly more difficult, by generating a large number of random problem instances from each class. Then we wish to be able to design a number of bad-case problem instances from each of the most difficult problem classes. However, we do not attempt worst-case analysis as this would require strong assumptions about the players selections.

Effectiveness of Strategies. What does a strategy need to address to be successful and which strategies or strategy paradigms are effective on various problem classes? We need some understanding of how effective each strategy is, over the range of problem sizes for which it is computationally tractable. We identify which are the most promising strategies in terms of worst performance, expected performance, and their nemeses. Analysis of which strategies are most suited to a given class of problems instance is a prerequisite to the development of *meta-strategies* for the *active learning monitors* described in Section 4.2.2. A meta-strategy would first classify a problem instance into one of a number of learned problem classes and assess what strategy (and parameters) to initially apply.

Computational experimentation with Vehicle Routing and Scheduling Problems usually compare the relative quality of solution, with respect to computational effort, of different heuristics in solving specific classes of problems. However, the effectiveness of a strategy for the CPCP cannot be evaluated in isolation from other strategies, since a strategy can only be evaluated against another strategy, one for each player.

We have already seen the tiny tournament of Section 5.5. In general, a tournament involves a set of strategies and a set of problem instances. Each strategy plays off against every other strategy on each of the problem instances in turn. There are, however, two principal difficulties in evaluating a strategy:

(i). How to weight the performance of a particular strategy against a number of opposing strategies when some strategies may be more difficult to play against than others.

(ii). How to weight the performance between two strategies over a number of problem instances when some problem instances may be more difficult than others.

Hence, the effectiveness of strategies and the difficulty of problem instances are not independent investigations. Problem instances are required to evaluate the performance of strategies and strategies are required to evaluate the difficulty of problem instances. To measure a strategies success we must consider both robustness (worst-case performance) and expected performance (average-case performance).

Hooker [105] argues that initial heuristics determine the benchmark problem set (since these are the problems it does well on) and the benchmark problem set determines the future heuristics designed for the problem (since they must perform well on the benchmark problems to be considered good).

To address these issues we adopt the following four step approach:

Step I. Specification of general problem instance classes (Section 9.1).

Step II. Preliminary tournaments (Section 10.1), between a range of strategies, on the general problem instance classes. Evaluate which strategies are most robust on each problem class by worst-case performance on average-case problems.

Step III. Prediction of the expected value of a problem instance for each player (Section 9.2) and sensitivity analysis of problem instances (Section 9.3), for the purpose of designing bad-case problem instances (Section 9.4).

Step IV. Final tournaments between the most robust strategies of the preliminary tournaments on a set of bad-case problem instances from each of the most difficult problem classes (Section 10.2). Evaluation is on the basis of expected, average-case performance on bad-case problem instances.

Chapter 9 considers the design of challenging test problems and Chapter 10 considers the design and execution of the computational tournaments between the proposed strategies. Finally, Chapter 11 draws some overall conclusions and formulates a programme of future research.

CHAPTER 9

Problem Design

When you can measure what you are speaking about, and express it in numbers, you know something about it.

— LORD KELVIN

Comparison of strategies requires problem instances on which to simulate two strategies playing against each other. Since strategies are designed for different problem instance characteristics, in this chapter we define three general classes of problem instances and draw on sensitivity analysis to design more challenging problem instances.

9.0 Introduction

A computational tournament requires the provision of a set of strategies to play in the tournament and a set of problem instances on which to conduct the tournament. In this chapter we develop a "*construction* and *improvement*" approach to generating problem instances.

Section 9.1 defines three simple problem instance classes based on individual prize values and locations, individual cluster values and locations, and prize value density, respectively. Although these problem instance classes are not restricted in size, i.e., number of prizes, they do correspond to the natural structures considered in Chapter 6 (prizes), Chapter 7 (clusters), and Chapter 8 (prize value density). The problem instances generated may be considered *average-case* since each is "constructed" from a general random distribution.

To define *bad-case* problem instances requires some numerical definition for evaluating the "badness" or "difficulty" of a problem instance. To this end, Section 9.2 compares a number of

static and dynamic predictors of the expected value of a problem instance to each player and, Section 9.3 graphically analyses the sensitivity of these predictors to small changes in a prize value, prize location, initial player location, or the overall deadline. These considerations enable Section 9.4 to define a *badness objective function* in terms of the sensitivity of the difference between predictors, and to propose a local search heuristic for generating bad-case problem instances from average-case problem instances.

9.1 Classes of Problem Instances

A **problem instance** is defined by the number of prizes, the location of each prize, the value of each prize, the initial location of each player, the overall deadline, and the step size, Δ. Four distinct problem instance classes have already been *implied* in this thesis.

Tiny Class. Chapter 5 analysed the case in which there are only two prizes. Section 5.5.3.2 further subdivided the **tiny class** (T-class) of dynamic two-prize problems into seven subclasses for the tiny tournament.

Prize Class. Chapter 6 developed tactical engines for evaluating contingent sequences of prizes. The **prize class** (P-class) (Section 9.1.1) focuses on the layout and value of individual prizes, generated by construction, or drawn from some probability distribution, or through a combination of these.

Cluster Class. Chapter 7 designed strategic engines for evaluating contingent sequences of clusters. The **cluster class** (C-class) (Section 9.1.2) focuses on the layout and value of individual clusters.

Density Class. Chapter 8 designed grid engines for strategic planning on problems with a large number of prizes with respect to spatial prize value density, without concern for individual prizes. The **density class** (D-class) (Section 9.1.3) focuses on the composition of prize value density features.

The natural structure of these problem classes ranges from explicit one-step constraint satisfaction through to significant concentrations of prizes. P-class problems store value at discrete locations; the result of tactical manoeuvring is to claim a prize and there is direct conflict over every prize. C-class problems store value in clusters; strategic inter-cluster manoeuvring is generally distinct from tactical intra-cluster manoeuvring. D-class problems distribute value over an area; strategic manoeuvring is seamlessly incorporated with harvesting of prizes, and harvesting in the most promising region is more important than considering the opponent as conflict only occurs at a very local scale.

We now define the prize, cluster, and density classes of problem instances in detail, with examples.

9.1.1 Prize Class of Problem Instances

The philosophy of the prize class (P-class) of problem instances is the layout and value of individual prizes. The model we use to define the class is a sequence of five components:

(1) number of prizes $\longrightarrow$ *(2) placement of prizes*
$\hookrightarrow$ *(3) value of prizes* $\longrightarrow$ *(4) placement of players*
$\hookrightarrow$ *(5) overall deadline*

Although this sequence is not strict, it serves as a useful guide by enumerating the subclasses and allowing later subclasses to functionally depend upon earlier subclasses. Each of these components is defined in more detail in the subsections which follow.

9.1.1.1 Number of Prizes

Let n be the specified number of prizes. Alternatively, a range of numbers of prizes could be specified, in which case let n be one number drawn from this range. Although the tactical engines of Chapter 6 assume a small problem instance, it is intended that there be no restriction on the number of prizes in the P-class.

9.1.1.2 Placement of Prizes

Test problems for the Euclidean TSP (customer locations only) are generally drawn from probability density functions over the subregion of the Euclidean plane (Bentley [16]). Since the prize location component of a CPCP is equivalent to a Euclidean TSP, we follow the same approach by defining four main types of prize location subclasses:

(c) Constructions.

(l) Random subsets of lattice-like locations.

(p) Small random perturbations of construction or lattice problems.

(r),(h) General probability distributions.

Table 9.1 expands these main types by defining a number of prize location distributions and constructions in which n is the number of prizes, $N(\mu, \sigma)$ represents a normal distribution with mean μ and standard deviation σ, and $U[a, b]$ represents a uniform distribution on the interval $[a, b]$. The following notes define some of these ideas more precisely.

Perturbations. Each prize location of a construction or lattice problem is perturbed by a bearing $\theta \sim U[0, 2\pi]$ and a distance $r \sim |N(0, 0.03)|$. This defines the P-class prize location subclasses **(pl)**, **(pw)**, **(pg)**, **(pd)**, **(ps)**, and **(pk)**.

Lattice. The density factor, ρ, is the probability that each vertex of a rectangular lattice contains a prize. A dense lattice ($\rho \approx 1$) would have a prize at almost every lattice point and a sparse lattice ($\rho \approx 0$) would have a prize at very few lattice points. We want an $x \times y$ lattice for which $\frac{x}{y} \approx \kappa$ and $xy \approx \frac{n}{\rho}$. Hence, let

$$y = \left\lceil \sqrt{\frac{n}{\rho\kappa}} \right\rceil \text{ and } x = \left\lceil \frac{n}{\rho y} \right\rceil .$$

Table 9.1: P-Class: Prize Location Component Subclasses

Identifier	Name	Subclass Description
(cl)	*line*	Evenly distributed along a line segment.
(cg)	*ghostbuster*	Evenly distributed on the circumference of a circle with an additional line through its centre.
(cw)	*wheel*	Evenly distributed on the circumference of a circle.
(ld)	*dense lattice*	Uniform random subset of a lattice of density $\rho \sim U[0.7, 1]$ and scale factor κ.
(ls)	*sparse lattice*	Uniform random subset of a lattice of density $\rho \sim U[0.1, 0.4]$ and scale factor κ.
(lk)	*spokes*	Uniform random subset of points on k spokes of density $\rho \sim U[0.1, 0.9]$, each spoke bearing separated by $\frac{2\pi}{k}$.
(ra)	*annulus*	Uniform random distribution on the area between two concentric circles.
(rc)	*circle*	Uniform random distribution on the circumference of a circle.
(rn)	*normal*	Each dimension independent $\sim N(\frac{1}{2}, \frac{1}{6})$.
(rq)	*square*	Uniform random distribution on the unit square.
(rr)	*rectangle*	Uniform random distribution on a rectangle of scale κ.
(rs)	*stringy*	Distributed along a wandering path.
(hs)	*hotspot*	Locations with polar coordinates $\theta \sim U[0, 2\pi]$ and $r \sim 1 - \sqrt{1 - U[0, 1]}$.
(hr)	*hotring*	Locations with polar coordinates $\theta \sim U[0, 2\pi]$ and $r^2 \sim N(0.13, 0.04)$.

Spokes. We generate k spokes of a wheel excluding the perimeter, each spoke containing $\lceil \frac{n}{\rho k} \rceil$ vertices. The density factor, ρ, is the probability that each such vertex contains a prize.

Stringy. A **stringy cluster** (see Section 7.1.1.4) is generated with an evenly spaced x-coordinate. The y-coordinate is composed of a uniformly distributed component (of magnitude at most $\frac{1}{5}$ of that of the length of the x-coordinate) plus a sinusoidal trend of low amplitude and random period.

9.1.1.3 Value of Prizes

We consider three main types of prize value allocation, assuming that the prize locations have already been determined. These are characterised by whether the prizes are dependent upon the prize or player locations.

Prize values independent of prize locations.

- **(i)** Identical value.
- **(u)** Uniform random values.
- **(n)** Normally distributed values.
- **(s)** Stratified values: randomly partition the prizes into two subsets, the larger subset having prize values $\sim N(4,1)$ and the smaller subset having prize values $\sim N(20,3)$.

Prize value dependent upon prize location.

- **(e)** Value function of proximity to some central location: generate prize values such that $v_j \leftarrow e^{-\mu d_{Cj}}$, where d_{Cj} is the distance from the central location to prize $j \in V$, and μ is a parameter.
- **(b)** Value function of proximity to random boundary location.
- **(c)** Value function of proximity to the perimeter of a central generating circle.
- **(q)** Value function of proximity to the perimeter of a central generating square.

Prize value dependent upon player locations.

- **(p)** Value function of proximity to the initial player locations, such that prizes closer to a player location are more valuable than those farther away.

9.1.1.4 Placement of Players

Players are located with respect to the prizes and each other. For one player, the following subclasses are defined:

- **(c)** Centroid of prize-value-weighted prize locations.
- **(b)** Location on the convex hull of the prize locations.

(m) Mid-range: determine the centroid of the prize locations (not weighted by prize value), divide the prize region into a 2×2 grid through the centroid, randomly select one of the grid-cells and determine its centroid.

(l) Location collinear with prizes if prizes are collinear, otherwise located on the principal component axis.

(s) Split the prizes between the players such that they are located at the weighted centroid of guaranteed prizes.

(i) Players have coincident locations.

Considering the two player in combination, the valid pairs of these subclasses are: **(ci)**, **(cb)**, **(cm)**, **(bi)**, **(bb)**, **(bm)**, **(mi)**, **(mm)**, **(li)**, **(ll)**, and **(ss)**.

9.1.1.5 Overall Deadline

The final component is the specification of the overall deadline, λ. Let the **cooperative time**, $\omega(\mathcal{A}, \mathcal{B})$, be the minimum time for the two players to collect *all* the prizes, given their initial locations. Two subclasses are considered:

(i) Infinite: $\lambda = \infty$.

(r) Restrictive: $\lambda \sim \omega(\mathcal{A}, \mathcal{B}) N(1, \frac{1}{4})$ such that $\lambda \geq \max\{\min_{j \in V} d_{Aj}, \min_{j \in V} d_{Bj}\}$.

9.1.1.6 Descriptor for Prize Problem Instance Class

A seven field descriptor is used to specify a particular subclass of the prize class of problem instances:

$$P : n\ell\ell v a b \lambda$$

where 'n' is the number of prizes, '$\ell\ell$' is the two-field prize location subclass, 'v' is the prize value subclass, 'ab' is the two-field player location subclass, and 'λ' is the overall deadline subclass. There are $N \times 20 \times 9 \times 11 \times 2 = 3960N$ prize problem instance subclasses, where N is the cardinality of the range of number of prizes considered.

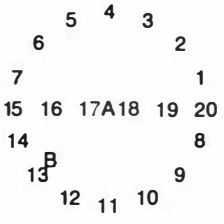
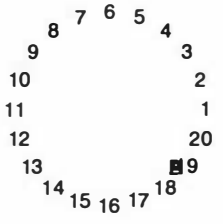
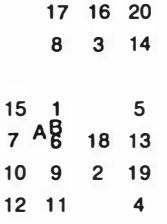
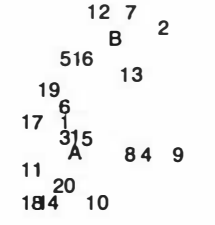
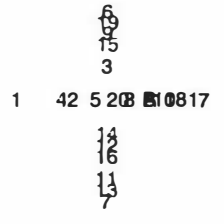
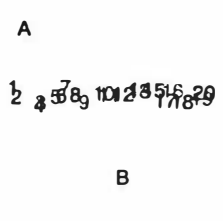
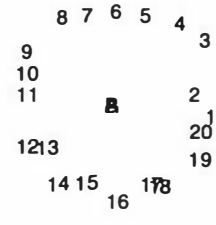
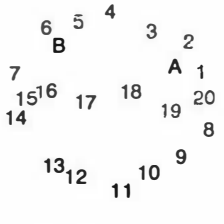
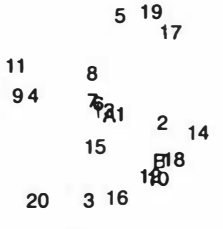
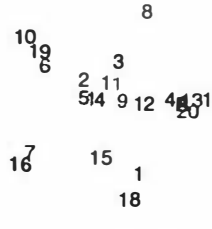
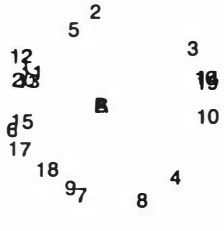
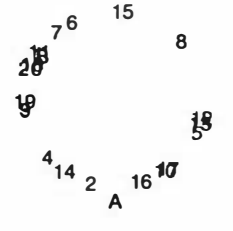
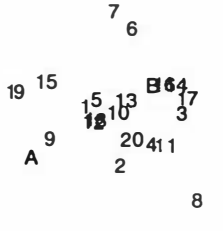
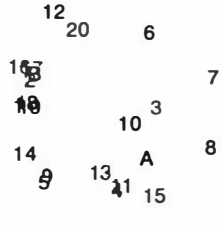
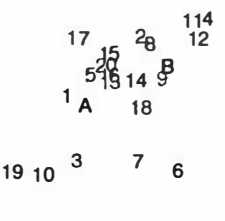
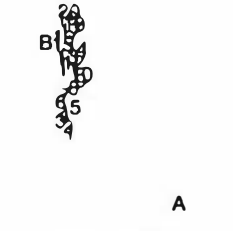
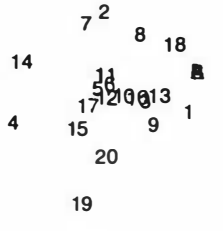
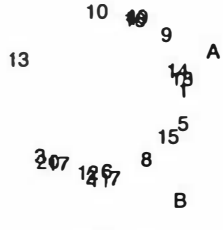
Table 9.2 gives some examples of P-class problem instances with 20 prizes. These examples show only the prize location and player location components—one of each prize location component subclass and random player location component subclass. A '–' is used as a wildcard to denote that a particular component is left unspecified.

9.1.2 Cluster Class of Problem Instances

The philosophy of the cluster class (C-class) of problem instances is the layout and value of clusters. A sequence of six inter-cluster and intra-cluster components defines the class:

$$\textit{(1) number of clusters}$$
$$\hookrightarrow \textit{(2) placement and value of clusters} \longrightarrow \textit{(3) size and scale of clusters}$$
$$\hookrightarrow \textit{(4) placement and value of prizes within clusters}$$

Table 9.2: Examples of P-Class Problem Instances

 P:20clucb–	 P:20cgccm–	 P:20cwqmi–	 P:20ldill–
 P:20lsnss–	 P:20lksmi–	 P:20plubb–	 P:20pwqci–
 P:20pgpmm–	 P:20pdbli–	 P:20psicm–	 P:20pkuli–
 P:20racci–	 P:20rcnbm–	 P:20rnebm–	 P:20rqsmm–
 P:20rrpss–	 P:20rsbll–	 P:20hsbbi–	 P:20hrebb–

↪ (5) placement of players ⟶ (6) overall deadline

9.1.2.1 Number of Clusters

Let m be the specified number of clusters and let n be the specified number of prizes. Alternatively, the number of clusters may be specified as a range. Note that m forms part of the subclass specification but that n does not.

9.1.2.2 Placement and Value of Clusters

Cluster problems can be viewed as "fractured" small problems in which a single prize has its value divided amongst a small number of prizes in the vicinity. The specific approach taken here is to construct P-class prize locations and prize values and use these as the cluster centres and cluster values, respectively. Hence the set of C-class cluster location subclasses is the same as for the P-class prize locations subclasses, i.e., {cl, cg, cw, ld, ls, lk, ra, rc, rn, rq, rr, rs, hs, hr, pl, pw, pg, pd, ps, pk}, and the set of C-class cluster value subclasses is the same as for the P-class prize value subclasses, i.e., {i, u, n, s, e, b, c, q, p}.

9.1.2.3 Size and Scale of Clusters

Clusters, however, are not point locations. The size of a cluster is the cardinality of the set of prizes which are members of that cluster. The scale of a cluster is the spatial extent of the cluster relative to the spatial extent of the prize set as a whole.

Size. The following two subclasses are applied:

- **(r)** The clusters have random sizes as defined in Section 7.A.1.3.
- **(e)** Each cluster has approximately the same number of prizes, i.e., either $\lceil \frac{n}{m} \rceil$ or $\lfloor \frac{n}{m} \rfloor$.

Scale. Let A be the specified prize region area. We consider loose clusters, i.e., those which cover a relatively large area, and tight clusters, i.e., those which cover a relatively small area.

- **(l)** Loose clusters: large area $a \sim \frac{A}{m} U[\frac{1}{2}, \frac{3}{4}]$.
- **(t)** Tight clusters: small area $a \sim \frac{A}{m} U[0, \frac{1}{4}]$.
- **(m)** Mixture of loose and tight clusters $a \sim \frac{A}{m} U[0, \frac{3}{4}]$.

9.1.2.4 Placement and Value of Prizes

The placement, value, size, and scale of clusters are sufficient to define the inter-cluster components of the C-class. We now need to select the prize layout and prize value within each cluster. We do not subclassify on placement and value of prizes but rather randomly select one of the P-class prize location subclasses, and one of the P-class prize value subclasses, for each cluster independently. The cluster of prizes is then scaled in terms of prize values, scaled in terms of cluster area, randomly rotated, and then its prize-weighted centroid is positioned on the prescribed cluster centre location.

9.1.2.5 Placement of Players

For each player, we independently determine whether that player is to be located on the boundary of the prize region or centrally, and then, locally, whether that player is to be located central within a cluster, on the boundary of a cluster, or between clusters. The options for player $\mathcal{A}$ are:

(a) Globally central, within a cluster.

(b) Globally central, on the boundary of a cluster.

(c) Globally central, between clusters.

(d) Global boundary, within a cluster.

(e) Global boundary, on the boundary of a cluster.

(f) Global boundary, between clusters.

The options for player $\mathcal{B}$ are the same except that, if player $\mathcal{A}$ is a member of subclass (a), (b), (d), or (e), then player $\mathcal{B}$ can be assigned to one of the following three subclasses.

(i) Coincident with player $\mathcal{A}$.

(w) Within the same cluster as player $\mathcal{A}$.

(h) On the boundary of the same cluster as player $\mathcal{A}$.

9.1.2.6 Overall Deadline

As in the P-class, two subclasses are considered:

(i) Infinite: $\lambda = \infty$.

(r) Restrictive: $\lambda \sim \omega(\mathcal{A}, \mathcal{B})N(1, \frac{1}{4})$, so that λ is sufficient to allow each player to visit all the prizes from at least one cluster.

9.1.2.7 Descriptor for Cluster Problem Instance Class

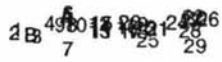
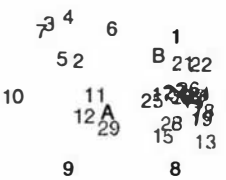
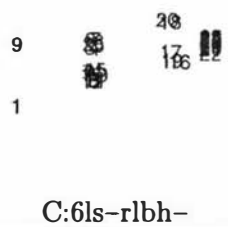
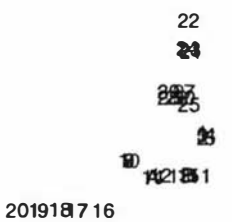
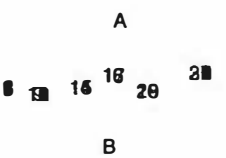
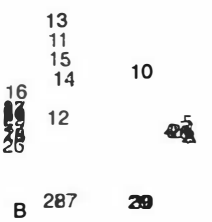
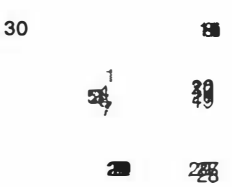
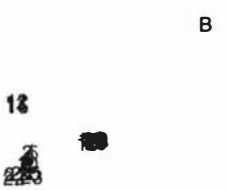
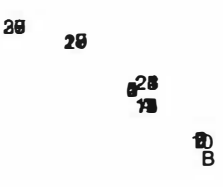
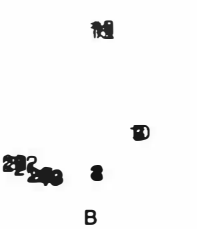
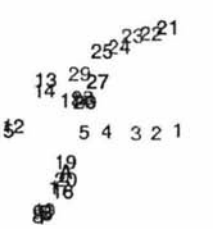
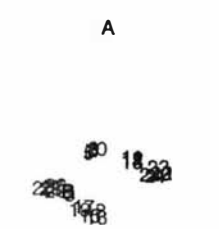
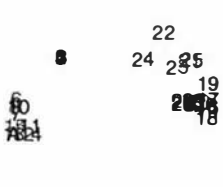
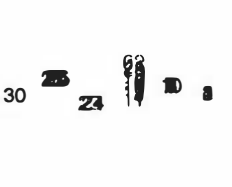
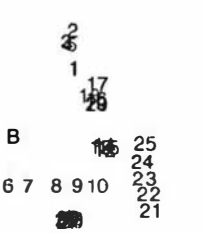
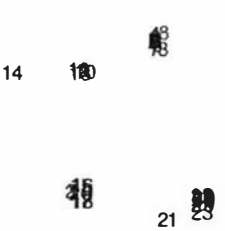
A nine field descriptor is used to specify a particular subclass of the cluster class of problem instances:

$$C : m\ell\ell v s \kappa a b \lambda$$

where 'm' is the number of clusters, '$\ell\ell$' is the cluster location subclass, 'v' is the cluster value subclass, 's' is the cluster size subclass, 'κ' is the cluster scale subclass, 'ab' is the player location subclass, and 'λ' is the overall deadline subclass. There are $M \times 20 \times 9 \times 2 \times 3 \times 6 \times 9 \times 2 = 116640M$ cluster problem instance subclasses, where M is the cardinality of the range of number of clusters considered.

Table 9.3 gives some examples of C-class problem instances. Each example gives a different cluster location subclass but cluster size, cluster scale, and player locations are randomly selected from those subclass components. Note that the values of prizes and clusters are not shown in these examples.

Table 9.3: Examples of C-Class Problem Instances

 C:6cl–etbw–	 C:6cg–rlae–	 C:6cw–rtdf–	 C:6ld–rmbb–
 C:6ls–rlbh–	 C:6lk–eldh–	 C:6pl–rtcc–	 C:6pw–rmde–
 C:6pg–rtbd–	 C:6pd–rtei–	 C:6ps–etff–	 C:6pk–emba–
 C:6ra–emaf–	 C:6rc–etdc–	 C:6rn–elbe–	 C:6rq–rmab–
 C:6rr–emeb–	 C:6rs–rmeh–	 C:6hs–elec–	 C:6hr–rldi–

9.1.3 Density Class of Problem Instances

The philosophy of the density class (D-class) of problem instances is based upon the spatial distribution of prize value density. The design of the D-class follows a different pattern to that established by the P-class and C-class. Firstly, we define a spatial probabilistic model for specifying prize locations and values. Secondly, we define four prize value density features, the building blocks for constructing the overall prize value density. Thirdly, we show how these features are superimposed and how the probabilities are translated into prize locations and values. Finally, we specify a sequence of six components which define the class in the pattern similar to P-class and C-class.

9.1.3.1 The Model

The model for constructing the prize locations and values is based upon a regular lattice of candidate prize locations. Two surfaces are constructed over these lattice points; the lattice is used to ensure a discrete coverage over the prize region, since these surfaces are continuous. The first is the probability of selecting that point as a prize location. The second is the relative value of the prize at that lattice point should the lattice point be selected as a prize. The convolution of these two surfaces is termed a **prize value density function**.

9.1.3.2 Prize Value Density Features

We define four prize value density features:

Background Feature. Low location selection probability and moderate value.

Hot Features. High location selection probability and high value.

Cold Features. High location selection probability and low value.

Black Features. Zero location selection probability.

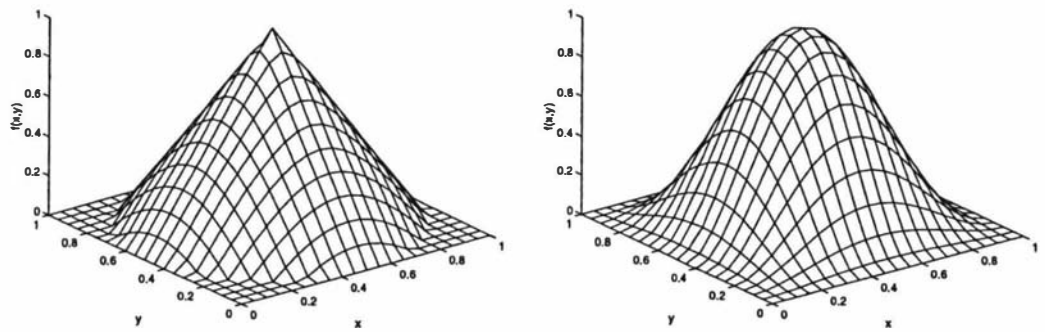
The features are defined in more detail below.

▼ Background Feature

The **background feature** is simply a uniform, low selection probability and relatively moderate, constant value over the whole lattice. This provides a relatively sparse blanket of background "noise" prizes. Thus, if there are no further prize value density features, the background would be equivalent to the lattice prize location subclass of P-class.

▼ Hot Features

Hot features have a high probability of selection associated with each member lattice point and a relatively high value. Thus a hot feature densely concentrates prize value with a number of high valued prizes in a small area. Three types of hot features are used: hot-spots, hot-rings, and hot-ridges.



(a) Cone ($a = 1$) (b) Quartic ($a = 1$)

Figure 9.1: Hotspot Prize Density Function $f(x, y)$

Hot-Spot. A **hotspot** is a concentration of prize value about a point which is approximately radially symmetric. Two function-based models are considered in Figure 9.1: a cone and a quartic.

Cone. For $(x, y) \in [0, 1] \times [0, 1]$ let $f(x, y) = \min\{1, \max\{0, a(1 - 2\sqrt{(x - \frac{1}{2})^2 + (y - \frac{1}{2})^2})\}\}$ where a is chosen such that $\int_0^1 \int_0^1 f(x, y)dxdy = 1$. From the cone probability density we can sample the bearing $\theta \sim U[0, 1]$ and the radius $r \sim 1 - \sqrt{1 - U[0, 1]}$. This is illustrated in Figure 9.1(a).

Quartic. For $(x, y) \in [0, 1] \times [0, 1]$ let $f(x, y) = (16a)^2(x^4 - 2x^3 + x^2)(y^4 - 2y^3 + y^2)$ where a is chosen such that $\int_0^1 \int_0^1 f(x, y)dxdy = 1$. This has zero derivatives at the maximum and at the boundary. This is illustrated in Figure 9.1(b).

These functions serve as both the selection probability and the prize value. A single parameter corresponds to either the maximum probability of selection (the **feature density**) or the maximum relative prize value (the **feature strength**). This parameter scales the $f(x, y)$. These two models may be used interchangeably; specifically, we select one model at random whenever a hot feature is required. This provides a mixture of sharp and smooth hot-spots.

Hot-Ring. A **hot-ring** is a concentration of prize value about the circumference of a circle of radius R. The selection probability and value are functions of the distance from the circumference of the defining circle. We adapt the cone and quartic hot-spot definitions for a lattice point at a distance r from the defining circle circumference.

Cone. Let $f(r) = \min\{1, \max\{0, a(1 - 2r)\}\}$.

Quartic. Let $f(r) = 4a(r^4 - 2r^3 + r^2)$.

Thus the parameters of a hot-ring are the defining radius, R, and the density, or strength, a.

Hot-Ridge. A **hot-ridge** is a concentration of prize value about a path, similar to a stringy cluster but without prize values (see Section 7.1.1.4). A hot-ridge is generated by considering the

distance, r, of each lattice point from the defining path and applying the same function, $f(r)$, as in a hot-ring.

▼ Cold Features

Cold features have a high probability of selection associated with each member lattice point and a relatively low value. Thus a cold feature densely concentrates prize location in a small area, but *not* prize value, since each prize is of low value. Three types of cold features are used, analogous to the the corresponding hot features: cold-spots, cold-rings and cold-valleys.

Cold-Spot. A **cold-spot** is a concentration of low prize values about a point. The selection probability is the same as for a hot-spot. However, the value is defined as the negative of that for the corresponding hot-spot. This is because when the features are superimposed (see Section 9.1.3.3 later) prize values are additive; thus, the negative value of the cold-spot is added to the already moderate value of the background to constitute a low value. Consequently, cold-spot values must be lower in magnitude than the background.

Cold-Ring. A **cold-ring** is a concentration of low prize values about the circumference of a circle. The selection probability is the same as for a hot-ring and the value is the negative of the value for the corresponding hot-ring.

Cold-Valley. A **cold-valley** is a concentration of low prize values about a path. Again, the selection probability is the same as for a hot-ridge and the value is the negative of the value for the corresponding hot-ridge.

▼ Black Features

Black-Hole. The only black feature considered is the **black hole** which annihilates all candidate prizes within a circle. When a black hole overlays another feature, all the prizes in that subregion are removed. This is achieved by giving the lattice points within the defining circle a selection probability of negative infinity.

9.1.3.3 Superposition of Features

Selection probability and value of background, hot, cold, and black features at lattice points are additive, i.e., the overall prize value density is defined by the selection probability and value surfaces which sum the contributions from each feature. The expected number of prizes is simply the sum of the individual lattice point selection probabilities. Hence, these selection probabilities are scaled such that the expected number of prizes is set to the required number of prizes. To construct the actual prize locations we randomly determine whether each lattice point will contain a prize according to its selection probability. Those lattice points that are selected take the corresponding value at that lattice point. Following this, each prize location is given a small, random perturbation so that the prizes do not lie precisely on a lattice (this is basically only cosmetic).

9.1.3.4 The Component Subclasses

It remains to specify the sequence of components which defines the D-class:

(1) number of each feature
↪ *(2) feature strength and scale*
↪ *(3) placement of players*

▼ Number of Each Feature

This component of the D-class is composed of the number of hot features, the number of cold features, and the number of black holes. We *do not* specify the number of hot-spots, hot-rings, hot-ridges, cold-spots, cold-rings, or cold-valleys. Rather we take the required number of hot features and randomly choose which types of hot features will be used. This is so that we do not produce too many subclasses of problem instances. Similarly, we take the required number of cold features and randomly choose which types of cold features will be used. The chosen features are then centred at random locations in the prize region with random orientations.

▼ Scale and Strength of Features

The *scale* of a feature is its spatial extent relative to the spatial extent of the whole prize region. Recall that the *strength* of a feature is the maximum value of the feature.

Scale.

- **(A)** Large area
- **(a)** Small area
- **(m)** Mixture of large and small areas

Strength.

- **(E)** High value per unit area.
- **(e)** Low value per unit area.
- **(m)** Mixture of large and small strengths.

Note that the maximum selection probability (feature density) is fixed for the background, hot, and cold features.

▼ Placement of Players

The critical subclasses of player placement are whether the players are in local conflict or not. The former can be modelled by P-class player location subclasses **(ci)** (with a small displacement), and the latter by subclass **(mm)**. We give these subclasses the labels **(c)** and **(m)**, respectively, for the D-class.

9.1.3.5 Descriptor for Density Problem Instance Class

A six field descriptor is used to specify a particular subclass of the density class of problems instances:

$$D : hcbssa$$

where '*hcb*' are, respectively, the number of hot, cold and black features; '*ss*' is the feature scale and strength subclass; and '*a*' is the player location subclass. Note that there is no overall deadline subclass. There are $HCB \times 3 \times 3 \times 2 = 18HCB$ density problem instance subclasses, where H, C and B are, respectively, the maximum number of hot, cold and black features considered.

Table 9.4 gives some examples of D-class problem instances for zero, one, or two of each feature type. Prizes which contribute to a hot feature are shown with a '*', prizes which contribute to a cold feature are shown with an 'o' and the remaining background prizes are show with a '·'.

9.2 Prediction of the Expected Value of the Game

A strategy can be evaluated in terms of *efficiency*, *accuracy*, and *effectiveness*. The goal of this section is to deal with *accuracy* of estimates of the value of a future game position by comparing the robustness of predictors of the value of the game to the relative initial player location.

9.2.1 Definitions

Let $\mathbb{S}$ be a set of player strategies. For $\mathsf{a}, \mathsf{b} \in \mathbb{S}$, let $v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ denote the total prize value claimed by player $\mathcal{A}$ in a simulation on problem instance $\wp$ in which player $\mathcal{A}$ adopts strategy a and player $\mathcal{B}$ adopts strategy b.

Definition 9.2.1

- Let $\mathbb{S}_\infty$ be the infinite set of all possible player strategies. The **expected value of the game** $\wp$ (or the **expected value of problem instance** $\wp$), $v^*_A(\wp)$, is defined as the *Nash equilibrium* value (in mixed strategies) to player $\mathcal{A}$ of the infinite two player game in which both players' pure strategies are $\mathbb{S}_\infty$ and the payoff to player $\mathcal{A}$, when player $\mathcal{A}$ selects the pure strategy $\mathsf{a} \in \mathbb{S}_\infty$ and player $\mathcal{B}$ selects the pure strategy $\mathsf{b} \in \mathbb{S}_\infty$, is $v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$, if such a Nash equilibrium exists.

- Let $v^\sharp_A(\wp)$ be defined in terms of the supremum (least upper bound) and infimum (greatest lower bound) by Equation (9.1).

$$v^\sharp_A(\wp) = \sup_{\mathsf{a} \in \mathbb{S}_\infty} \left\{ \inf_{\mathsf{b} \in \mathbb{S}_\infty} v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \right\} \tag{9.1}$$

  Although $v^*_A(\wp)$ may not exist, $v^\sharp_A(\wp)$ always exists, since the finite number of prizes implies that there are only a finite number of possible values for $v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$.

- Let $\mathbb{S}$ be a given finite set of player strategies. The **computational maximin value of the game** $\wp$ (or the **computational maximin value of the problem instance** $\wp$),

Table 9.4: Examples of D-Class Problem Instances

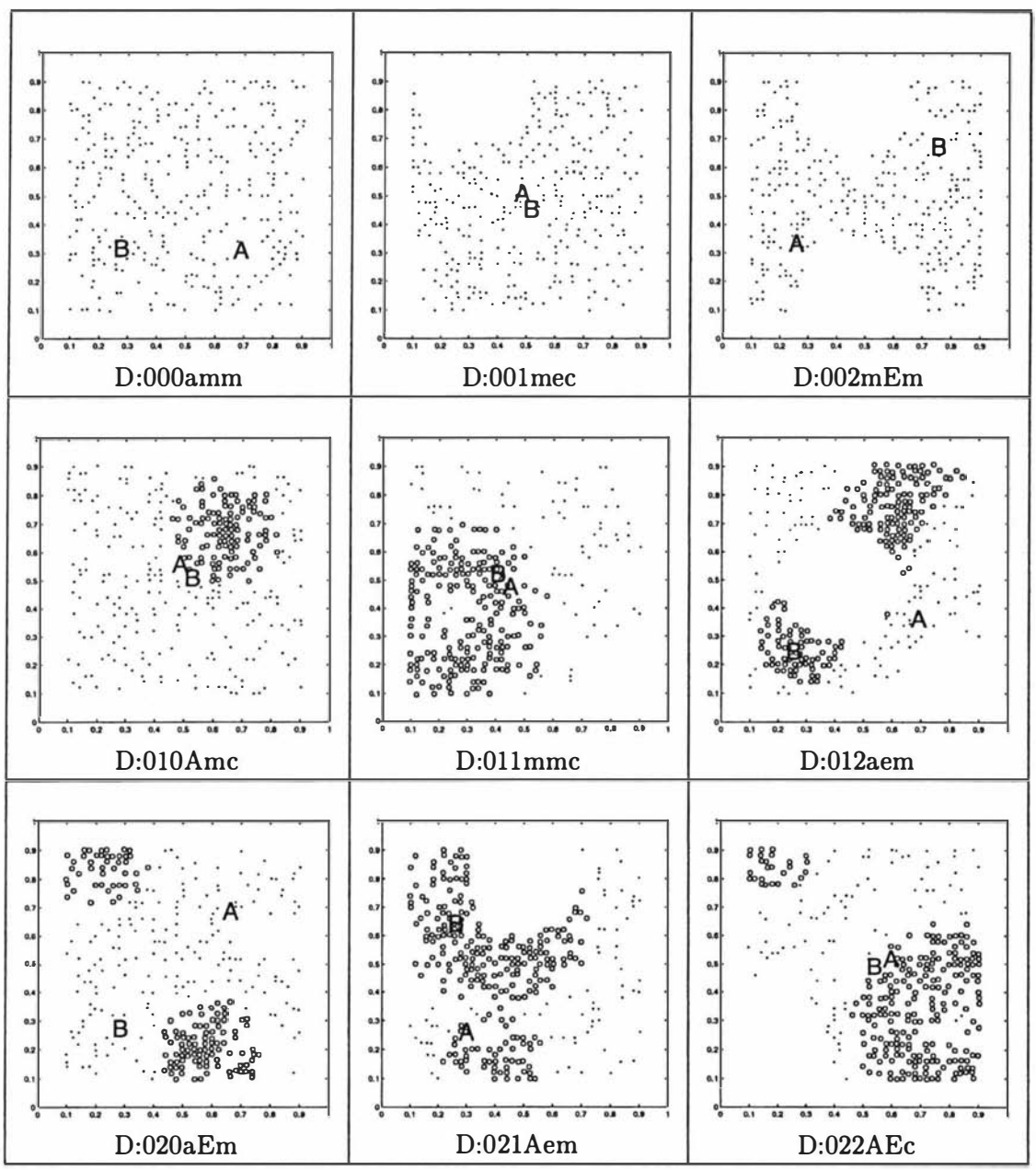


Table 9.4: Examples of D-Class Problem Instances (continued)

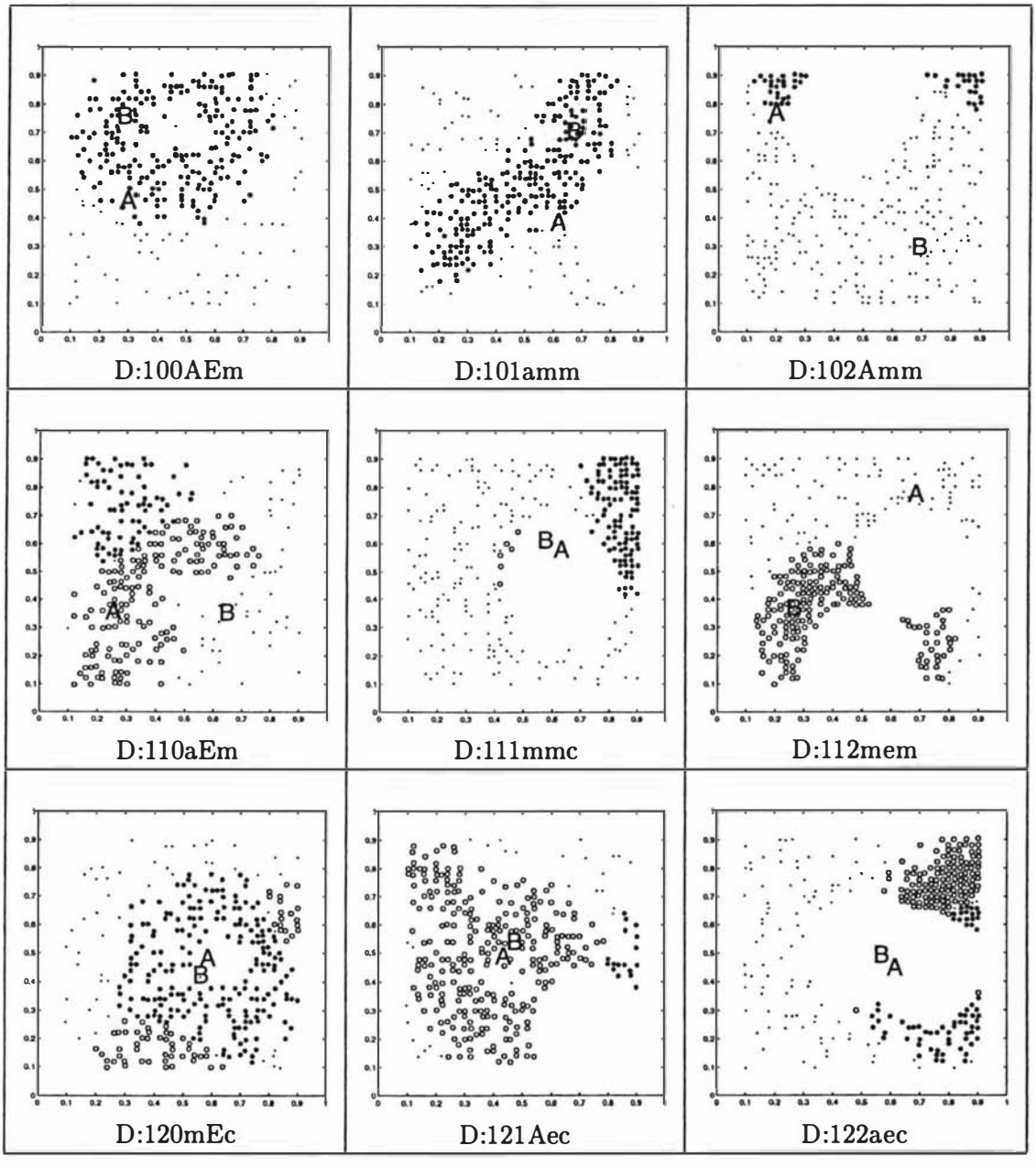


Table 9.4: Examples of D-Class Problem Instances (continued)

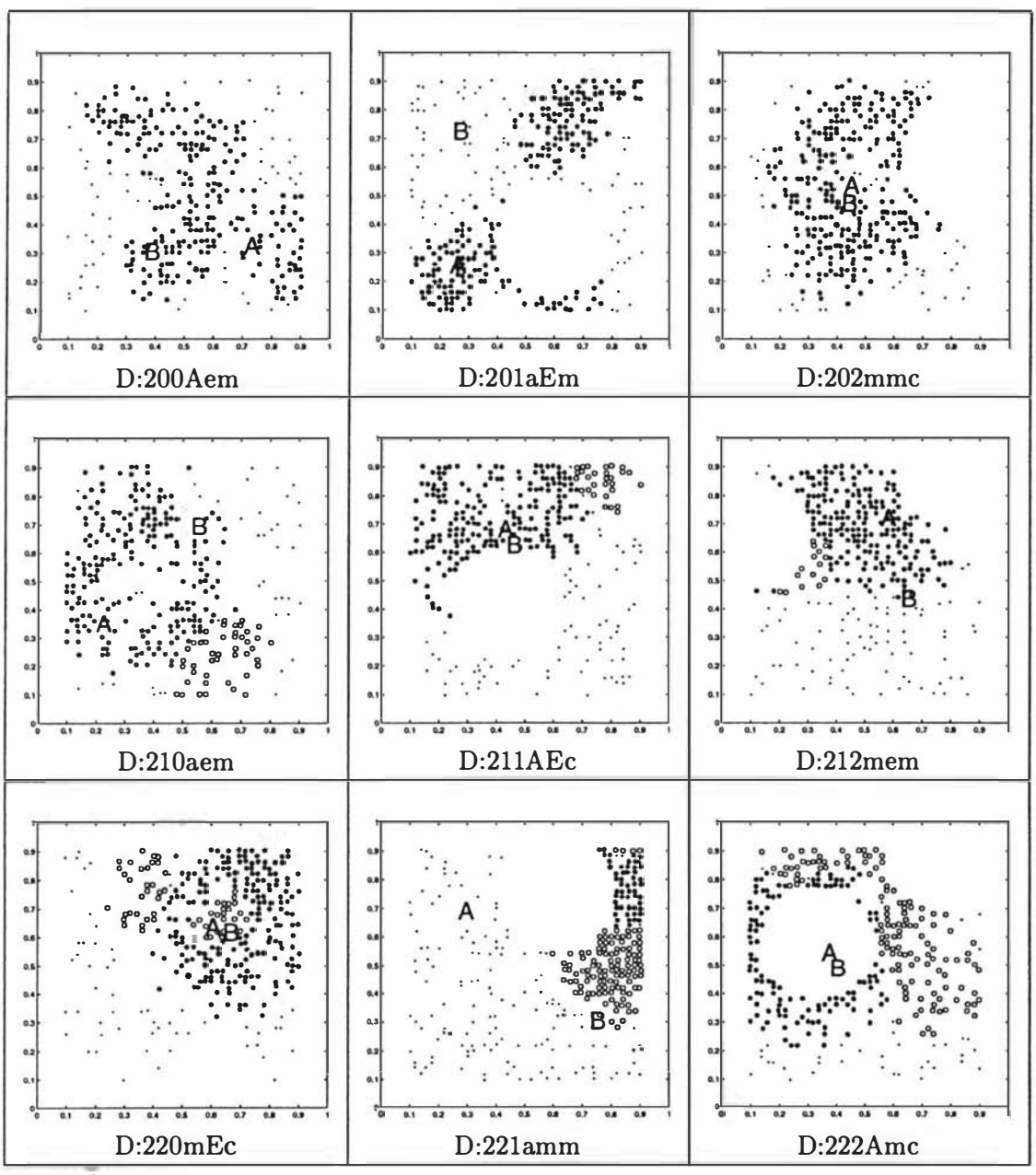


Table 9.5: Static Predictors of the Expected Value of the Game

Identifier	Predictor Name	Strategy	Tournament
P1	PRIZE-GUARANTEE	§3.3.1	§10.1.1
P2	PRIZE-DT-($\kappa = 0$)	§6.3	§10.1.1
P3	SINGLE-FAMILY-PRIZE-GUARANTEE	§7.2.2.4	§10.1.2
P4	SINGLE-FAMILY-PRIZE-DT-($\kappa = 0$)	§7.2.2.6	§10.1.2
P5	FAMILY-PRIZE-GUARANTEE	§7.2.2.4	§10.1.3
P6	FAMILY-PRIZE-DT-($\kappa = 0$)	§7.2.2.6	§10.1.3
P7	CLUSTER-GUARANTEE//PRIZE-DT-($\kappa = 0$)	§7.3.3	§10.1.4
P8	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-($\kappa = 0$)	§7.5	§10.1.4

$v_A^\diamond(\wp)$, with respect to the given $\mathbb{S}$, is defined by Equation (9.2).

$$v_A^\diamond(\wp) = \max_{a \in \mathbb{S}} \left\{ \min_{b \in \mathbb{S}} v_A(\wp; \mathcal{A} \sim a, \mathcal{B} \sim b) \right\} \tag{9.2}$$

- $v_B^*(\wp)$, $v_B^\natural(\wp)$ and $v_B^\diamond(\wp)$ are defined similarly, with respect to player $\mathcal{B}$.

□

9.2.2 Computational Evaluation of Static Predictors

The tactical engines of Chapter 6 and the strategic engines of Chapter 7 each estimate the expected return from a representative, projected game position, i.e., the value of a problem instance; these are *static* predictors of the game value. Playing one strategy off against another on the same problem instance also provides an estimate of the value of the problem instance; this is a *dynamic* predictor of the game value.

Computational evaluation of predictors on a problem instance, $\wp$, involves comparison of the value of the problem instance estimated by each predictor against the computational maximin value of the problem instance, $v_A^\diamond(\wp)$, with respect to a set of strategies $\mathbb{S}$. However, a single problem instance is insufficient to evaluate a predictor—a set $\mathbb{P}$ of problem instances is required. Because of the requirement to determine $v_A^\diamond(\wp)$ for every $\wp \in \mathbb{P}$, the computational evaluation of a number of static predictors dovetails with the preliminary computational tournaments of Section 10.1, although there is no dependence of the preliminary computational tournaments on these computational evaluations of static predictors.

Table 9.5 gives a number of static predictors of the expected value of the game. Two predictors are chosen for each preliminary computational tournament from Sections 10.1.1–10.1.4. Figure 9.2 presents the corresponding results for the static predictors. The root mean square error between each predictor and $v_A^\diamond$ was calculated with respect to $\mathbb{S}$. Prize values for every problem instance are standardised to total 10000.

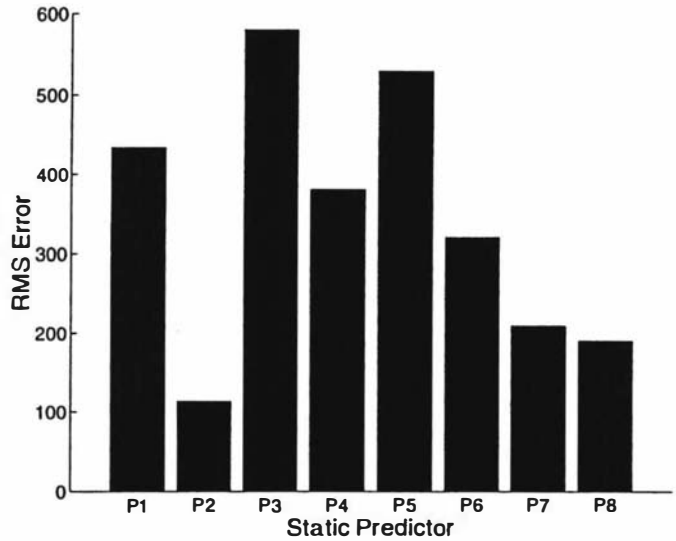


Figure 9.2: Results from Evaluation of Static Predictors

9.2.3 Conclusions

This section has introduced the concepts of static and dynamic predictors of the value of the game or, equivalently, the value of a problem instance. The aim of the computational evaluation is to indicate the relative accuracy of such predictors rather than to rigorously test, refine, or determine regression equations.

We can conclude that the estimates of the tactical and strategic engines are reasonably rough but are, in general, indicative. Certainly the the '-DT' based predictors are more accurate than the '-GUARANTEE' based predictors. However, much work remains in the evaluation of the predictive capabilities of these methods and the consequences of the decisions made by the corresponding DMS strategies.

9.3 Sensitivity Analysis

Sensitivity analysis is concerned with how small changes made to a problem instance affect its solution or outcome. This is useful in a diverse range of problems. Winston [207] introduces sensitivity analysis for linear programming. Golden and Stewart [90] undertake an empirical analysis of various TSP heuristics. Johnson and Papadimitriou [111] give performance guarantees for TSP heuristics. Greenberg [93] presents a recent annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. Sensitivity for optimization problems is an established, analytical practice, but for a new problem, a visualisation approach may give more rapid, preliminary insight.

In this section we follow the work of Jones [115, 116, 117, 118, 119, 120] in visualisation of spatial sensitivity analyses. We have already seen sensitivity diagrams for the two prize problem in Section 5.1.5, in which Figure 5.5 showed the sensitivity with respect to the location of one player,

and Figure 5.6 showed the sensitivity with respect to the location of one prize. Jones advocates the visual representation of sensitivity analysis in optimization problems for the purpose of studying the problem structure.

Jones [116, 117] considers the use of graphical depiction of sensitivity in general optimization problems, and solution methods, as an aid to understanding the structure of such problems. For problems in the Euclidean plane, plotting estimated value, actual value, or target prize over the plane, with respect to one of the spatial elements of the problem instance, gives a spatial sensitivity plot. Jones [118, 119, 120] looks specifically at spatial sensitivity analysis of heuristics for the Euclidean TSP and concludes that solution methods which produce "spatially contiguous" sensitivity plots offer closer to optimal solution quality than highly "fractured" sensitivity plots. In this way, the spatial sensitivity analysis offers some insight into the structure of problem instances and the ability of heuristics to produce good quality solutions.

9.3.1 Sensitivity to Static Evaluation

Two types of sensitivity plots are considered: sensitivity to a player location and sensitivity to a prize location.

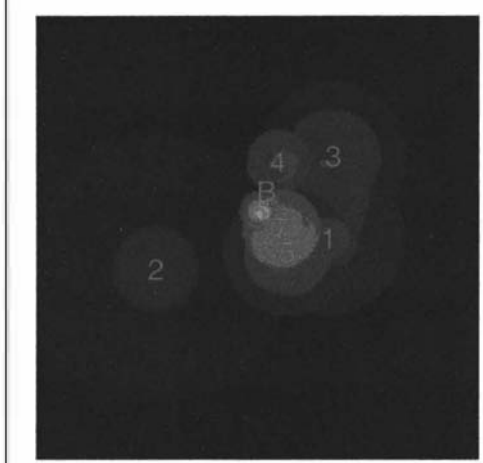
Figure 9.6 illustrates sensitivity of two *static* predictors to the initial location of a player. Figure 9.6(a) shows the sensitivity of the PRIZE-DT-($\kappa = 0$) predicted value for player $\mathcal{A}$, to the location of player $\mathcal{A}$, with fixed player $\mathcal{B}$ location. The fixed prize and player locations are shown in red, whilst the predicted value ranges from zero (black) to 10000 (white). Similarly, Figure 9.6(b) shows the sensitivity of the PRIZE-DT-($\kappa = 0$) predicted value for player $\mathcal{B}$, to the location of player $\mathcal{B}$, with fixed player $\mathcal{A}$ location; Figure 9.6(c) shows the sensitivity of the PRIZE-GUARANTEE predicted value for player $\mathcal{A}$, to the location of player $\mathcal{A}$, with fixed player $\mathcal{B}$ location; and Figure 9.6(d) shows the sensitivity of the PRIZE-GUARANTEE predicted value for player $\mathcal{B}$, to the location of player $\mathcal{B}$, with fixed player $\mathcal{A}$ location. These sensitivity plots show contiguous regions of value, indicating that these static predictors are robust with respect to spatial sensitivity.

Figure 9.7 illustrates sensitivity of these static predictors to the location of prize 1. Interestingly, for a given player, these plots are nearly identical. In comparison with the sensitivity to a player location, these sensitivity plots show fewer regions, but a similarly contiguous.

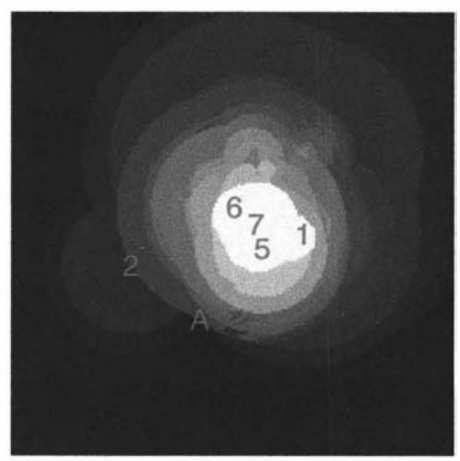
9.3.2 Sensitivity to Dynamic Evaluation

Figure 9.8 illustrates the sensitivity of two *dynamic* evaluators to the initial location of a player. Figure 9.8(a) shows the sensitivity of the result to player $\mathcal{A}$ of a simulation battle where both players play PRIZE-DT-($\kappa = 0$), to the location of player $\mathcal{A}$, with fixed player $\mathcal{B}$ location. Similarly, Figure 9.8(b) shows the sensitivity of the result to player $\mathcal{B}$ of a simulation battle where both players play PRIZE-DT-($\kappa = 0$), to the location of player $\mathcal{B}$, with fixed player $\mathcal{A}$ location. Figure 9.8(c) shows the sensitivity of the result to player $\mathcal{A}$ of a simulation battle where both players play NEAREST-NEIGHBOUR, to the location of player $\mathcal{A}$, with fixed player $\mathcal{B}$ location, and Figure 9.8(d) shows the sensitivity of the result to player $\mathcal{B}$ of a simulation battle where both players

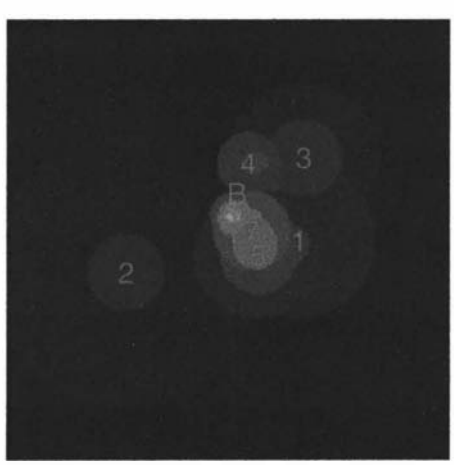
Table 9.6: Static Sensitivity to Player Location



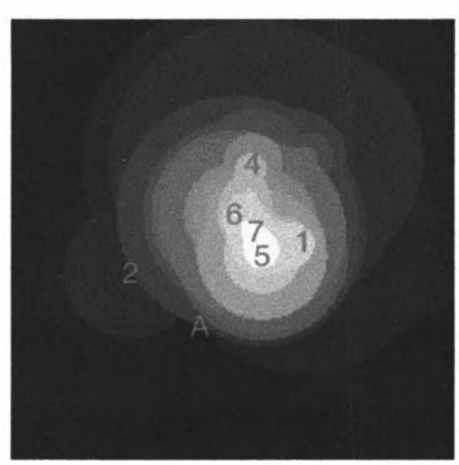
(a) $\mathcal{A}$-PRIZE-DT-value to $\mathcal{A}$-location



(b) $\mathcal{B}$-PRIZE-DT-value to $\mathcal{B}$-location

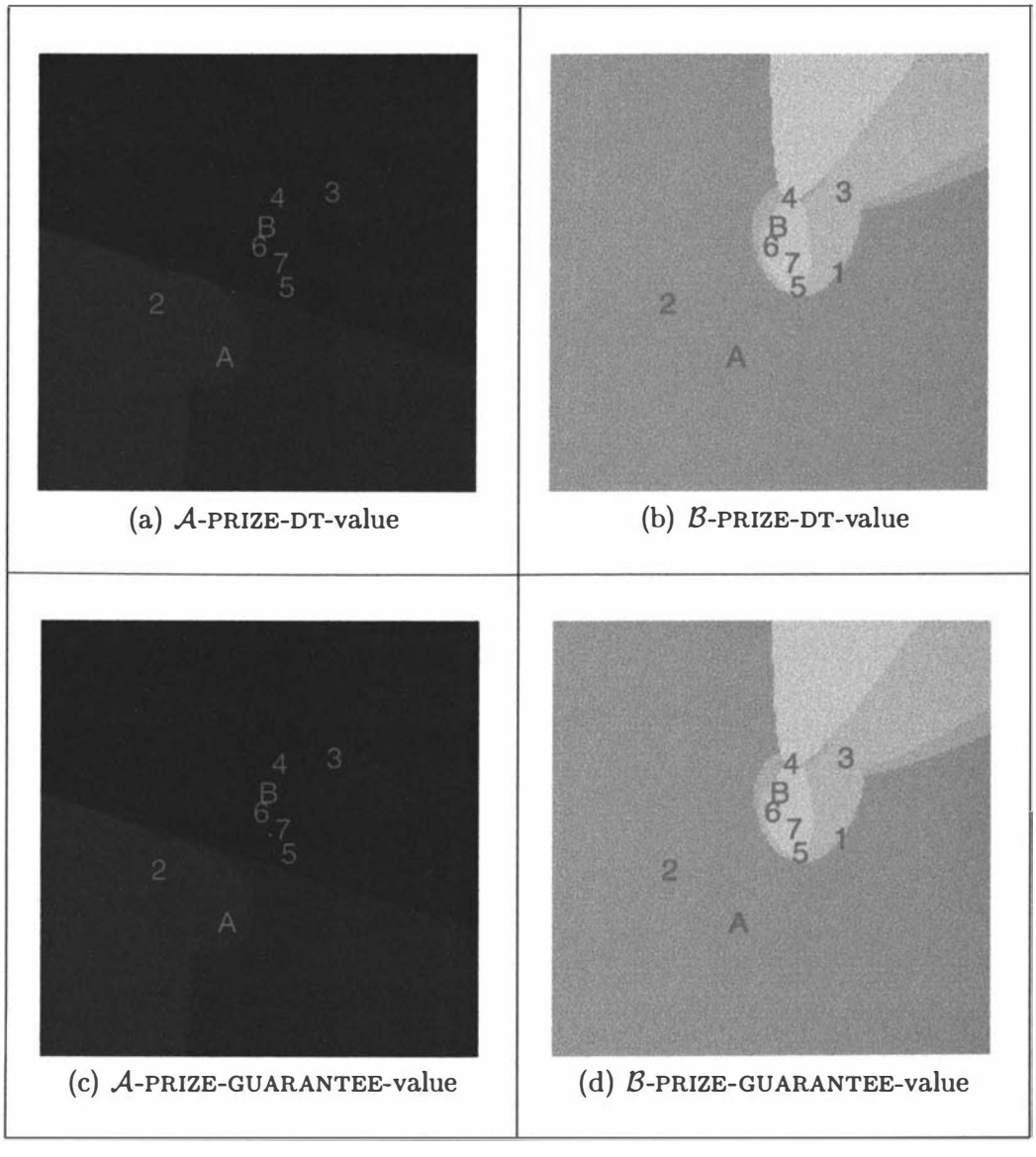


(c) $\mathcal{A}$-PRIZE-GUARANTEE-value to $\mathcal{A}$-location



(d) $\mathcal{B}$-PRIZE-GUARANTEE-value to $\mathcal{B}$-location

Table 9.7: Static Sensitivity to Prize 1 Location



(a) $\mathcal{A}$-PRIZE-DT-value

(b) $\mathcal{B}$-PRIZE-DT-value

(c) $\mathcal{A}$-PRIZE-GUARANTEE-value

(d) $\mathcal{B}$-PRIZE-GUARANTEE-value

play NEAREST-NEIGHBOUR, to the location of player $\mathcal{B}$, with fixed player $\mathcal{A}$ location. Similarly, Figure 9.9 illustrates the sensitivity of these dynamic evaluators to the location of prize 1.

The NEAREST-NEIGHBOUR dynamic evaluator is much more chaotic, implying that NEAREST-NEIGHBOUR is not a very sound strategy. In comparison with the PRIZE-DT static predictors, the PRIZE-DT dynamic evaluator shows a much more fractured plot. This indicates that value predicted by the static predictor is not always realised, either because of bad tactics or because the opponent is less predictable than expected.

9.3.3 Discussion

The goal of spatial sensitivity analysis for the CPCP is to illustrate the sensitivity of the CPCP to changes in a problem instance as a stepping stone to applying the idea to construction of problem instances that are challenging. If small changes in a problem instance produce significant variation in a good static predictor then the decision problem of the player for that problem instance becomes difficult.

It is also possible to study the sensitivity of value or target with respect to prize value or the overall deadline. Because these are both scalar, the sensitivity diagram will be a line graph rather than an image.

9.4 Bad-Case Problem Instance Design

The problem instance classes defined in Section 9.1 can be described as *average-case* problems since there is no prior reason to expect any one problem instance to be any more difficult that any other.

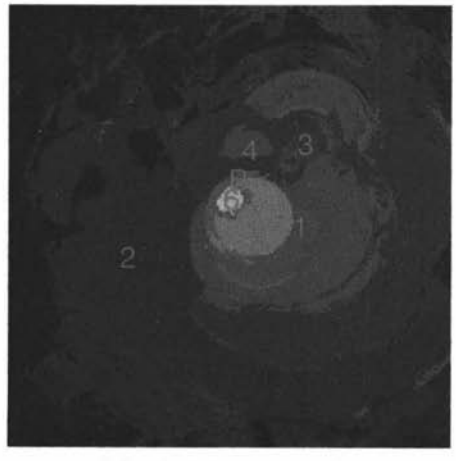
For combinatorial optimization problems with a maximization objective, the **worst-case performance** of a heuristic, H, is defined by

$$\rho_H = \inf_{\wp \in \mathbb{P}} \frac{v^H(\wp)}{v^*(\wp)}$$

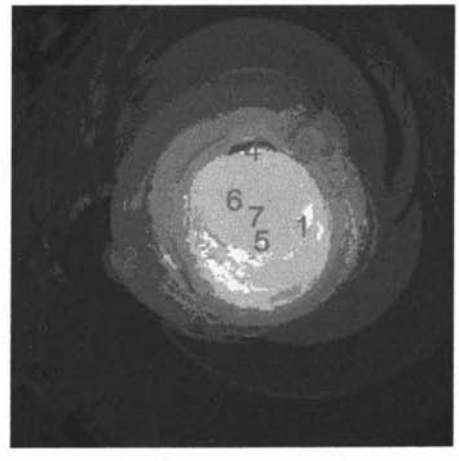
where $\mathbb{P}$ is the set of all possible instances of the given combinatorial optimization problem, and, for some specific problem instance $\wp \in \mathbb{P}$, $v^*(\wp)$ is the optimal value of that problem instance and $v^H(\wp)$ is the value determined by the heuristic H on that problem instance (see, e.g., Haimovich, Rinnooy Kan and Stougie [94] for examples involving the TSP and VRSP). A concrete value for ρ_H is usually proven by *construction* of an infinite family of problem instances with some parameter such that, in the limit of that parameter, the ratio $\frac{v^H(\wp)}{v^*(\wp)}$ tends to the value from above.

We wish to determine a set of **bad-case** problem instances that are sufficiently challenging to distinguish between the most robust strategies in terms of expected effectiveness. Hence a bad-case problem instance is "difficult" with respect to strategic and tactical decision making. In this way, bad-case analysis provides a "halfway house" between average-case and worst-case analysis. To this end we firstly discuss the characteristics of a problem instance that would make it difficult and, secondly, develop a quantitative measure of problem difficulty drawing upon the two previous sections.

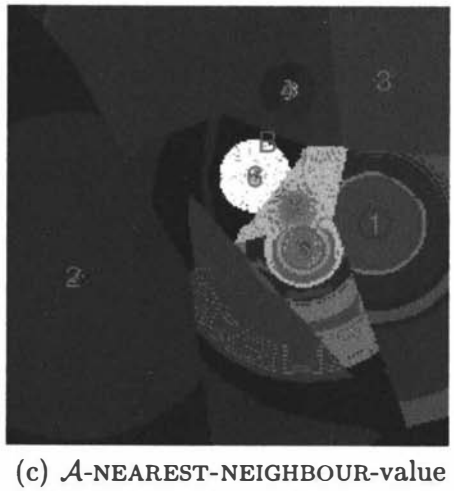
Table 9.8: Dynamic Sensitivity to Player Location



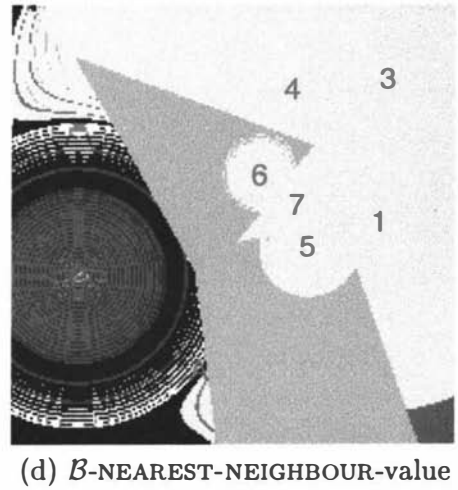
(a) $\mathcal{A}$-PRIZE-DT-value to $\mathcal{A}$-location



(b) $\mathcal{B}$-PRIZE-DT-value to $\mathcal{B}$-location



(c) $\mathcal{A}$-NEAREST-NEIGHBOUR-value to $\mathcal{A}$-location



(d) $\mathcal{B}$-NEAREST-NEIGHBOUR-value to $\mathcal{B}$-location

9.4.1 Bad-Case Problem Characteristics

A *knife-edge* decision is one upon which the result of the game essentially hinges, i.e., it is critical that the correct call be made if the maximum benefits are to be realised. In some cases a knife-edge decision may involve a game scenario that is impossible to resolve in pure one-step strategies. If the initial decision problem of at least one of the players is a difficult knife-edge decision then the problem instance itself could be described as difficult.

A problem is strategically difficult when:

- Experimentation shows that payoff estimated by tactical or strategic analysis is not realised (predictability).
- Small variations in the problem instances produce significant changes in the tactics required to realise the expected payoff (sensitivity).

Two possible paradigms for building bad-case problems are *structural templates* and *iterative knife-edge search*. The former is a more thorough approach in which building blocks of combinations of a small number of prizes are recursively assembled according to templates or patterns such that a sequence (or tree) of difficult knife-edge decisions are embedded in the problem design. The latter exploits the predictability and sensitivity of the problem instance by perturbing the initial player locations with respect to some surrogate objective for difficulty. The iterative knife-edge search is developed further in the following section. This produces problem instances with the *initial* tactical decision being difficult even though later decisions may be easy. In justification of investigating the latter approach, we note that a good understanding of what constitutes a difficult knife-edge decision is a prerequisite to being able to embed a sequence of such decision problems in a problem design.

9.4.2 Knife-Edge Search for Bad-Case Problems

By way of contrast, we expect that an "*easy problem instance*" would correspond to one for which it is easy to predict its value, i.e., the better predictors of Section 9.2 would predict approximately the same value. Hence one way to define a *difficult problem instance* is as a problem instance for which there is a high variability between predictors of the value of that problem instance, i.e., a difficult problem instance is one which is hard to predict accurately and hence hard to make good strategic and tactical decisions.

To define a quantitative measure of problem instance difficulty, choose two (static or dynamic) predictors, v' and v'', from those considered in Section 9.2. Define the difficulty, $\xi(\wp)$, of problem instance $\wp$ by Equation (9.3).

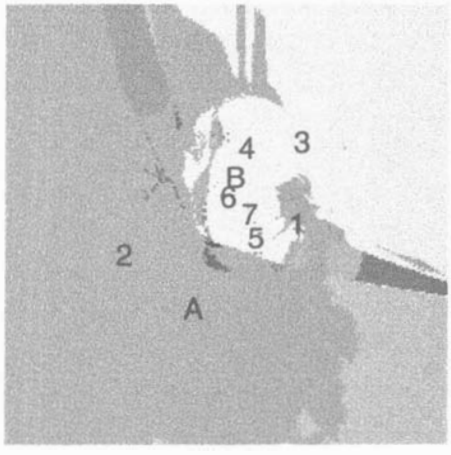
$$\xi(\wp) = \frac{(|v'_A(\wp) - v''_A(\wp)| + |v'_B(\wp) - v''_B(\wp)|)}{2\Omega(\wp)} \tag{9.3}$$

Recall that $\Omega(\wp)$ is the cooperative value, and note that $0 \leq \xi(\wp) \leq 1$. A problem instance $\wp$ is, therefore, bad-case if $\xi(\wp)$ is significantly nonzero.

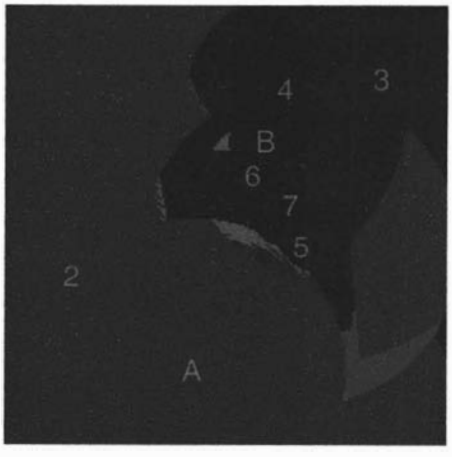
Bad-case problem instances may be found by "improving" the difficulty of an existing constructed problem instance by perturbing the initial player locations, prize locations, prize values,

Table 9.9: Dynamic Sensitivity to Prize 1 Location

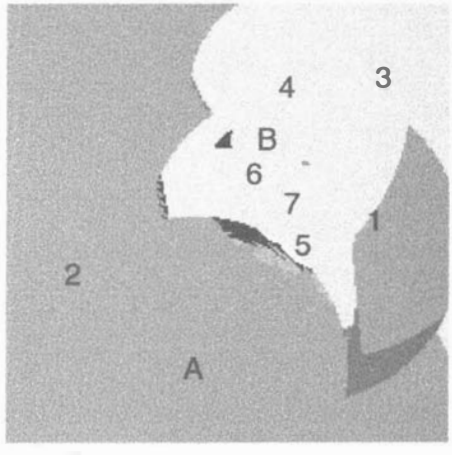
(a) $\mathcal{A}$-PRIZE-DT-value



(b) $\mathcal{B}$-PRIZE-DT-value



(c) $\mathcal{A}$-NEAREST-NEIGHBOUR-value



(d) $\mathcal{B}$-NEAREST-NEIGHBOUR-value

or overall deadline. We propose that bad-case problem instances can be found by searching a grid of possible initial player locations to find the combination of player $\mathcal{A}$ and player $\mathcal{B}$ initial locations which maximize $\xi(\wp)$, but fixing the prize locations, prize values, and the overall deadline. Local search is not appropriate since the discrete nature of $\xi(\wp)$ leads to spatial plateaux of constant values, with respect to perturbations in one spatial variable, which (non-metaheuristic) local search often finds difficult to search successfully.

Table 9.10 gives an example of *before* (Table 9.10(a)) and *after* (Table 9.10(b)) improvement with respect to PRIZE-DT-($\kappa = 0$) (υ') and PRIZE-GUARANTEE (υ''). The *before* problem instance is the same as that used in the sensitivity plots in Section 9.3. Table 9.10(c) shows the *before* sensitivity of $\xi(\wp)$ to the location of player $\mathcal{A}$ and Table 9.10(d) shows the *after* sensitivity of $\xi(\wp)$ to the location of player $\mathcal{A}$. Comparing Table 9.10(e) with Table 9.6(a) and Table 9.10(f) with Table 9.6(c) shows the corresponding differences between the PRIZE-DT and PRIZE-GUARANTEE sensitivity analyses *before* and *after*.

9.4.3 Conclusion

Applying static prediction and sensitivity analysis provides a means for generating problem instances that, we expect, are more challenging to strategies than problem instances simply drawn from a probability distribution. We cannot claim that this reflects any inherent difficulty in the problem instances, only that the static predictors we have developed in the form of tactical and strategic engines should find these more challenging.

Coda

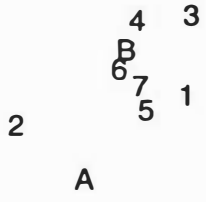
▼ Summary

We have developed methods for constructing various problem instance classes, evaluated methods for estimating the expected value of a given problem instance to each player, considered the sensitivity of problem instances to player location, prize location, prize value, and overall deadline, and, finally, developed a local search improvement method to generate bad-case problem instances.

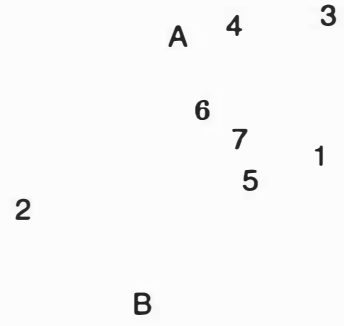
▼ Link

In the next chapter we turn to the computational tournaments to determine the best strategies and the most difficult problem instance classes.

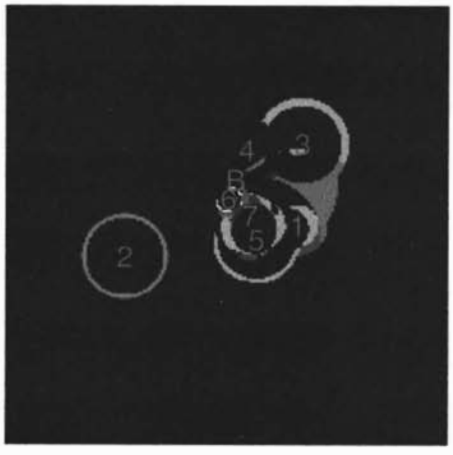
Table 9.10: Example of a Bad-Case Problem Instance



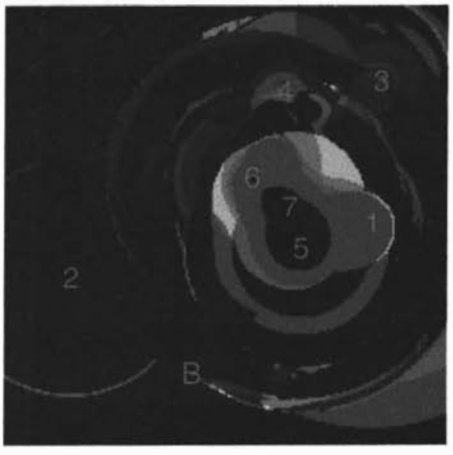
(a) Problem *Before*



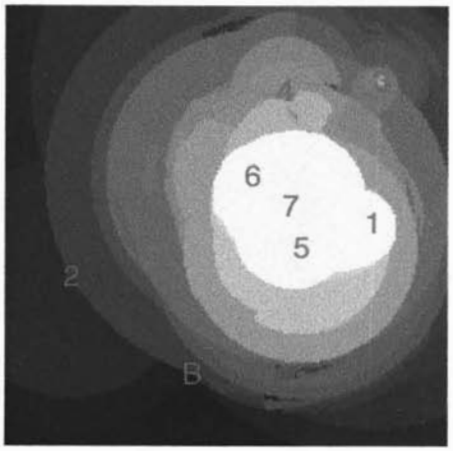
(b) Problem *After*



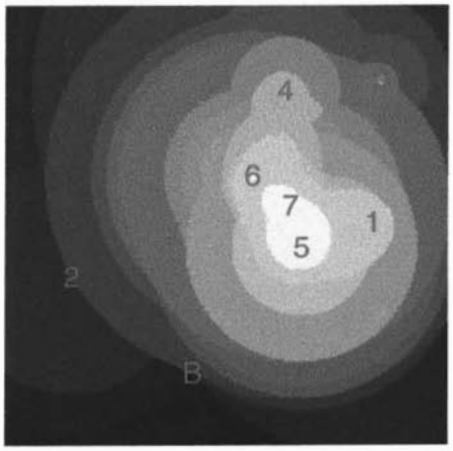
(c) *Before* static difficulty $\xi(\wp)$ sensitivity to $\mathcal{A}$-location



(d) *After* static difficulty $\xi(\wp)$ sensitivity to $\mathcal{A}$-location



(e) *After* static $\mathcal{A}$-PRIZE-DT-value sensitivity to $\mathcal{A}$-location



(f) *After* static $\mathcal{A}$-PRIZE-GUARANTEE-value sensitivity to $\mathcal{A}$-location

CHAPTER 10

Computational Tournaments

It is common sense to take a method and try it. If it fails, admit it frankly and try another. But above all, try something.

— FRANKLIN D. ROOSEVELT

Computational tournaments evaluate and compare the effectiveness of strategies and the difficulty of problem instances. In a series of simulations, each strategy plays off against every other strategy on a set of problem instances.

10.0 Introduction

Computational experiments with a new heuristic for an existing combinatorial optimization problem have traditionally involved the (favourable) comparison of the new heuristic against the best of the existing heuristics on some accepted "benchmark" set of problem instances. Recently, however, Barr et al. [13], Condon et al. [41], Hooker [105], and McGeoch [154] (amongst others) have strongly advocated the need for higher standards in the design and reporting on computational experiments with heuristic methods. In particular, a computational experiment must be unbiased, achieve the experimental goals, clearly demonstrate the performance of the tested heuristic, uncover reasons for performance, have justifiable rationale, generate supportable conclusions, and be reproducible (Barr et al. [13]). For combinatorial optimization problems, each heuristic may be evaluated separately from other heuristics. When dealing with computational evaluation of strategies for the CPCP, we attempt, where possible, to comply to these accepted standards but also consider the additional computational difficulties that are introduced by game-based considerations.

The computational experiments in this chapter all conform to a common structure: the computational tournament. Since there is no previous work, nor established benchmarks, with which to compare the results, the underlying goal of this chapter is to gather sufficient computational experience in order to make informed conjectures regarding the relative effectiveness of the strategies considered. Additionally, the computational effort required to evaluate the strategies is large even for small problems since the evaluation of each strategy involves many steps, many opponents, and many problem instances. For these reasons, the experimental aims have necessarily been modest in their scope. In the remainder of this introduction we consider the experimental aims, the common experimental design and selection of performance measures.

10.0.1 Experimental Aims

The experimental aims are two-fold:

- The first aim is to evaluate and compare strategy paradigms in terms of performance quality and required computational effort. We make no prior assumptions about the effectiveness of strategies. The strategies which are evaluated are *representatives* of the strategy paradigms and, as such, we need not exert unnecessary effort in finding the best parameters for the strategies.

- The second aim is to evaluate and compare problem instances with respect to difficulty. In particular, we wish to evaluate the problem instance classes of Section 9.1 (prize, cluster, and density) by component, e.g., for the prize class of problem instances, the four components are placement of prizes, value of prizes, placement of players, and overall deadline.

Recall from the Overview to Part III that Step II (preliminary tournaments) and Step IV (final tournaments) have different aims.

- For the preliminary tournaments, no prior assumptions are made about the effectiveness of opposing strategies, so all opponents must be treated equally. We cannot average the performance of a strategy against its opponents since we do not know if the opposing strategies are equally representative of "hard" opponents or "easy" opponents. It is more appropriate to initially evaluate robustness, i.e., worst performance against a set of opposing strategies, as a measure indicative of the performance of a strategy and to determine which strategies are "complex" and which are "naive". The aim of the preliminary tournaments is, therefore, to evaluate and compare the robustness of strategy paradigms and problem instance components.

- For the final tournaments, we restrict the set of strategies to the most robust of those in the preliminary tournaments and assume, for the purpose of evaluation, that they are all equally "complex". Therefore, we can take the average performance of a strategy against its opponents as the effectiveness of that strategy, and the aim of the final tournaments is to evaluate and compare the effectiveness of strategy paradigms and problem instance components.

10.0.2 Experimental Design

Computational experiments which compare the performance of strategies take the form of a *computational tournament* in which a number of strategies play off against one another on a set of problem instances. A **simulation battle** is a single simulated play of the CPCP on a particular problem instance with a particular pair of strategies, one for each player. This constitutes a (simulated) dynamic evaluation of the chosen strategies on the given problem instance.

The strategies proposed in Chapters 3–8 were each designed for a specific class of problem instance: the strategies of Chapter 5 cannot be applied to problem instances consisting of more than two prizes, the strategies of Chapter 6 become intractable for problem instances of (approximately) more than ten prizes, and it would certainly be inappropriate to apply the strategies of Chapter 8 to problem instances with only a few prizes. In summary, not all strategies are comparable in terms of effectiveness: we must compare strategies on problem instances of the order of the size for which they were originally designed, or, more simply, the computational tournaments must compare "apples with apples."

We consider *five independent tournaments* which feature different participating strategies and different sizes and classes of problem instances. Each tournament is motivated by a "defining strategy" which we use to bound the size of the problem instances for that tournament. Three tournaments correspond to the small, medium, and large problems of Chapters 6–8 respectively.

Small Tournament (S): A comparison of strategies on *small problems*, defined as those problem instances where the size is such that PRIZE-DT is computationally tractable, i.e., up to approximately ten prizes.

Medium Tournament (M): A comparison of strategies on *medium problems*, defined as those problem instances where the size is such that FAMILY-PRIZE-DT is computationally tractable, i.e., up to approximately six clusters of up to approximately five prizes each.

Large Tournament (L): A comparison of strategies on *large problems*, defined as those problem instances where the size is such that GRID-DT is computationally tractable, i.e., up to approximately one hundred prizes.

Two additional transitional tournaments are also designed to respectively compare the 'FAMILY-' strategies against the 'CLUSTER-' strategies and compare the 'CLUSTER-' strategies against the 'GRID-' strategies.

Small-Medium Tournament (SM): A comparison of strategies on **small-medium problems**, defined as those problem instances where the size is such that (SINGLE-FAMILY) FAMILY-PRIZE-DT is computationally tractable, i.e., up to approximately twenty prizes for anything up to ten clusters.

Medium-Large Tournament (ML): A comparison of strategies on **medium-large problems**, defined as those problem instances where is size is such that CLUSTER-DT is computationally tractable, i.e., up to approximately three families and eight clusters.

Table 10.1: Participating '-DT' Strategies in the Tournaments

Name	S	SM	M	ML	L
PRIZE-PARANOID	♮	–	–	–	–
ORIGINAL-DT	♮	–	–	–	–
PRIZE-DT	[♮]	–	–	–	–
FAMILY-PRIZE-PARANOID	∂	∂	♮	–	–
FAMILY-ORIGINAL-DT	∂	∂	♮	–	–
FAMILY-PRIZE-DT	∂	[∂]	[♮]	–	–
CLUSTER-PARANOID//PRIZE-PARANOID	–	∂	♮	♮	–
CLUSTER-PARANOID//PRIZE-DT	–	∂	♮	♮	–
CLUSTER-DT//PRIZE-PARANOID	–	∂	♮	♮	–
CLUSTER-DT//PRIZE-DT	–	∂	♮	[♮]	–
GRID-PATH//HARVEST-PATH	–	–	–	♮	♮
GRID-PATH//$\mathcal{ABA}$-PATH	–	–	–	♮	♮
GRID-PATH//PRIZE-DT	–	–	–	♮	–
GRID-DT//HARVEST-PATH	–	–	–	♮	♮
GRID-DT//$\mathcal{ABA}$-PATH	–	–	–	♮	[♮]
GRID-DT//PRIZE-DT	–	–	–	♮	–

KEY	
∂	Partial (single family only)
♮	Natural (unrestricted)
[ ]	Defining strategy for tournament

These five tournaments are delineated to compare subsets of the strategies in Table 10.1 on suitably sized problems; the strategies with their entry enclosed in square brackets define the maximum size of problem instance for each tournament, those marked '∂' represent the restriction of the family-cluster structure to a single family (where this is appropriate), and the remaining strategies are marked '♮'.

To complete the design of each of these computational tournaments we must specify which *strategies* are to participate and which sizes and classes of *problem instances* the players will compete over. This is left, individually, to Sections 10.1.1–10.1.5 and 10.2.1–10.2.5. With respect to reproducibility of these computational tournaments, the complete CRiKET system is available for comparison or extension (see Appendix A).

10.0.3 Performance Measures

In this experimental design, the independent variables (or factors) are the individual strategies and the components of the problem instance classes. The dependent variables are the *strategy effectiveness*, *computational effort*, and *problem instance difficulty*, as described below.

10.0.3.1 Strategy Effectiveness

We can evaluate $v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ and $v_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ by dynamic simulation battle. However, one player may be able to accumulate more value than the other simply because of the inequity of the starting locations. A solution to this problem is to repeat the simulation battle with the initial player location reversed and let the score be the sum of the original and reversed battle results. We expect that strategies of approximately equivalent effectiveness will claim approximately the same total in prizes from the two simulation battles. We also expect that if one strategy dominates another then it will do so, on average, from both orientations of the initial player locations.

A further problem is the comparability of a score on one problem instance with the score on another. Benchmarking the score against the expected value of the problem instance would be ideal, but we must settle for the computational maximin value of the problem instance. In addition, the total prize pool for each problem instance is standardised to 10000. Hence the resulting scores, $\varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ for player $\mathcal{A}$ and $\varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$ for player $\mathcal{B}$, for the simulation battle between player $\mathcal{A}$ adopting strategy a and player $\mathcal{B}$ adopting strategy b on problem instance $\wp$, are defined by Equations (10.1)–(10.2).

$$\begin{aligned} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) &= v_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) - v_A^\diamond(\wp) \quad + \\ &\quad\; v_B(\wp; \mathcal{A} \sim \mathsf{b}, \mathcal{B} \sim \mathsf{a}) - v_B^\diamond(\wp) \qquad (10.1) \\ \varkappa_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) &= v_B(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) - v_B^\diamond(\wp) \quad + \\ &\quad\; v_A(\wp; \mathcal{A} \sim \mathsf{b}, \mathcal{B} \sim \mathsf{a}) - v_A^\diamond(\wp) \qquad (10.2) \end{aligned}$$

We must then determine the effectiveness of an individual player strategy. This can only be evaluated with respect to a set of opposing strategies, $\mathbb{S}$, and a set of problem instances, $\mathbb{P}$. However, the evaluation is different for the preliminary and final tournaments for the following reasons:

- For the preliminary tournaments, we wish to evaluate the robustness of a strategy, i.e., determine the worst-case result against an average-case set of problem instances and opponent strategies. Hence, we define the effectiveness, $\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S})$, of strategy a by Equation (10.3), the 'MIN-MIN' evaluation.

$$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S}) = \min_{\mathsf{b} \in \mathbb{S}} \min_{\wp \in \mathbb{P}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \qquad (10.3)$$

  The 'MIN-MEAN' evaluation of Equation (10.4) is also calculated but does not define effectiveness.

$$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S}) = \min_{\mathsf{b} \in \mathbb{S}} \frac{1}{|\mathbb{P}|} \sum_{\wp \in \mathbb{P}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \qquad (10.4)$$

- For the final tournaments, we wish to evaluate the expected performance of a strategy, i.e., determine the average-case result against a bad-case set of problem instances and opponent strategies. Hence, we define the effectiveness of strategy a by the alternative Equation (10.5), the 'MEAN-MEAN' evaluation.

$$\varkappa_A(\mathbb{P}; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathbb{S}) = \frac{1}{|\mathbb{S}|} \sum_{\mathsf{b} \in \mathbb{S}} \frac{1}{|\mathbb{P}|} \sum_{\wp \in \mathbb{P}} \varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b}) \qquad (10.5)$$

Strategy effectiveness is one performance measure; the other considered for the computational tournaments is computational effort.

Consider Equation (10.6) as a score, $\varkappa^{\diamond}(\wp)$, for problem instance $\wp$.

$$\varkappa^{\diamond}(\wp) = \max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} \varkappa_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) = \max_{\mathsf{b}\in\mathbb{S}} \min_{\mathsf{a}\in\mathbb{S}} \varkappa_B(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) \tag{10.6}$$

Moreover,

$$\begin{aligned}
&\max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} \varkappa_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) \\
&\quad = \max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} (\upsilon_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) + \upsilon_B(\wp; \mathcal{A}\sim\mathsf{b}, \mathcal{B}\sim\mathsf{a})) \quad - \\
&\qquad (\upsilon_A^{\diamond}(\wp) + \upsilon_B^{\diamond}(\wp)) .
\end{aligned}$$

That is, $\varkappa^{\diamond}(\wp)$ is the difference between the *maximin of the sum* of the total prize value from each orientation of the initial player locations and the *sum of the maximin* of the total prize value from each orientation separately. If an *optimal* (Nash equilibrium) strategy were included in $\mathbb{S}$, we would expect that

$$\max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} (\upsilon_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) + \upsilon_B(\wp; \mathcal{A}\sim\mathsf{b}, \mathcal{B}\sim\mathsf{a})) = \upsilon_A^{\diamond}(\wp) + \upsilon_B^{\diamond}(\wp),$$

and, thus,

$$\max_{\mathsf{a}\in\mathbb{S}} \min_{\mathsf{b}\in\mathbb{S}} \varkappa_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b}) = 0.$$

This implies that $\varkappa_A$ is not biased with respect to problem instance.

10.0.3.2 Computational Effort

Computational effort is measured in terms of the total number of steps in a simulation battle, $\Xi(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b})$, the total processor time consumed by player $\mathcal{A}$'s strategy, $\tau_A(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b})$, and the total processor time consumed by player $\mathcal{B}$'s strategy, $\tau_B(\wp; \mathcal{A}\sim\mathsf{a}, \mathcal{B}\sim\mathsf{b})$. Initialisation of player strategies is not included in the processor time since a superset of initialisation procedures, common to many strategies, is performed, even if not required.

All reported computations were performed on a 200MHz SUN Ultra 2 UPA (Enterprise 2200) running Solaris 2.5.1 at approximately 7.72 SPECint95 and 11.4 SPECfp95 with 256M memory. All strategy implementations are in C/C++ compiled with GNU g++ version 2.7.2.1. Time is measured using the `gethrvtime` system call.

10.0.3.3 Problem Instance Difficulty

A problem instance may also be evaluated with respect to the simulation battles on that problem instance. Equation (9.3), on page 393, defines the **difficulty**, $\xi(\wp)$, of a problem instance $\wp$. Let $\upsilon' \leftarrow \upsilon^{\text{PRIZE-DT}}$ and $\upsilon'' \leftarrow \upsilon^{\diamond}$. Then Equation (10.7) gives the measure we will use for the difficulty of a problem instance derived from a computational tournament.

$$\xi(\wp) = \frac{(|\upsilon_A^{\text{PRIZE-DT}}(\wp) - \upsilon_A^{\diamond}(\wp)| + |\upsilon_B^{\text{PRIZE-DT}}(\wp) - \upsilon_B^{\diamond}(\wp)|)}{2\Omega(\wp)} \tag{10.7}$$

This measure is used for the small ($v' \leftarrow v^{\text{PRIZE-DT}}$), small-medium ($v' \leftarrow v^{\text{SINGLE-FAMILY-PRIZE-DT}}$), and medium ($v' \leftarrow v^{\text{FAMILY-PRIZE-DT}}$) tournaments with the parameter $\kappa = 0$. Equation (10.7) encapsulates some of the concept of difficulty, but it remains to be determined, computationally, whether it is a useful discriminator between difficult and easy problem instances.

A different approach is taken for the medium-large and large tournaments. The 'GRID-' strategic engines do not directly estimate the value of the problem instance, and the 'CLUSTER-' strategic engines provide very conservative estimates of the value of the problem instance due the the family-no-return, cluster-no-return requirements and the very loose clustering. The idea is to attempt to identify problems for which it is difficult to plan effective harvesting of a sequence of prize value concentration features in the medium-term, relative to a basic strategy which does no medium-term planning but efficiently sequences prizes in the near-term. Hence, we define the difficulty in terms of the absolute difference between the DYNAMIC predictors: GRID-PATH//HARVEST-PATH//STATIC versus itself and GRID-PATH//HARVEST-PATH//STATIC versus HORIZON-OP.

Finally, for a problem class $\mathbb{P}$, let the difficulty score of the class, $\bar{\xi}(\mathbb{P})$, be the average difficulty score of the problem instances in the class.

10.1 Preliminary Computational Tournaments

The experimental aims of the **Preliminary Computational Tournaments** are to evaluate and compare the robustness of strategy paradigms and the difficulty of problem instance components. These aims are addressed using the experimental design and performance measures outlined in the previous section. This constitutes Step II of our four step approach described in the overview of Part III. Step I provided the problem instances class components (see Section 9.1) and the strategies compared are those from Chapter 3 and Part II. The following five preliminary tournaments (small, small-medium, medium, medium-large, and large) each further refine the aim and specify the participating strategies and problem instances to complete the experimental design for that tournament.

10.1.1 Small Preliminary Tournament

The aim of the small preliminary tournament is to compare the robustness of, and computational effort required by, the strategies designed in Chapters 3 and 6, and to compare the difficulty of contributing components of problem instance classes. The problem instance size must be suitable with respect to computational tractability of the strategies. The defining strategy from Table 10.1 is PRIZE-DT. The tactical engine PRIZE-DT remains tractable up to approximately ten prizes, but a tactical engine is generally invoked several times throughout a simulation battle.

10.1.1.1 Participating Strategies

The participants in the small preliminary tournament are listed in Table 10.2. There are four distinct groups of strategies: the basic strategies (**G1–G12**), the local search based strategies (**S1–S6**), the non-cluster tactical engine based strategies (**S7–S14**), and the cluster tactical engine

based strategies (**S15–S21**). The basic strategies form a benchmark across all the preliminary tournaments since they are not constrained by computational tractability considerations. If each strategy plays against every other strategy, from both initial player locations, there is a total of $33^2 = 1089$ simulation battles per problem instance.

For the tactical engine based strategies, ORIGINAL-DT and PRIZE-DT, three representative versions of each are included, based on the evaluator for each game tree node. For ORIGINAL-DT, these are MINIMAX, MAXIMIN and MINIMAXIMIN. Recall that for PRIZE-DT, the parameter κ determines which of GENERALIZED-MINIMAX or GENERALIZED-MAXIMIN is employed: when $\kappa \ll 0$, MINIMAX is always used; when $\kappa \gg 0$, MAXIMIN is always used; and when $\kappa = 0$, MINIMAX is used if $t_A \geq t_B$, and MAXIMIN is used if $t_A < t_B$.

A family-cluster structure must be specified for 'SINGLE-FAMILY-' strategies. Since one cluster, or all singleton clusters, reduces the 'SINGLE-FAMILY-' strategies to their equivalent non-cluster strategies, we standardise the clustering component of these strategies to three clusters, predetermined according to the improved prize-value-weighted-Ward method of Section 7.1.1.2. Additionally, the ANY–ALL–PCTSP requirement is set of 'ANY' since this is most suitable for small problems.

Monitor parameters determine the conservativeness of response to prediction or observation of the opponent's actions, and influence the frequency of invoking a tactical engine. Therefore, the monitor parameters have been set conservatively prior to the tournament.

10.1.1.2 Problem Instances

We are interested in small problems with little significant, natural, clustered structure. All problem instances considered here are from the P-class (see Section 9.1.1) for which the defining components are: number of prizes, prize location subclass, prize value subclass, player location subclass, and overall deadline subclass. Prize and player locations are scaled to fit within the unit square and the prize values are scaled such that $\sum_{j \in V} v_j = 10000$.

The overwhelming constraint on the number of prizes we can consider is the computational tractability of the tactical engines. A ORIGINAL-DT game tree is generally deeper than a PRIZE-DT game tree on the same number of prizes, but it is usually less broad. Although these tactical engines can implicitly search game trees up to approximately ten prizes in reasonable computing time, the number of battles required to be simulated makes ten prizes prohibitive. Hence we adopt exactly seven prizes: challenging enough tactically but not too computationally burdensome.

The original intention was to study at least ten instances of *every* combination of P-class component subclasses, but this proved to be too computationally ambitious. This is due solely to the immense computational requirements of a game-based tournament and the number of steps required per simulation battle. Hence we consider the prize location, prize value, player location and overall deadline components as independent factors. One hundred problem instances were generated for each subclass of the prize location component, a total of 2000 problem instances. These were distributed such that there are at least one hundred of each subclass of the prize value, player location and overall deadline components. Since there are $9 \times 11 \times 2 = 198$ combinations of the prize value, player location and overall deadline subclasses, only 100 problem instances for

Table 10.2: Small Preliminary Tournament: Participating Strategies

Identifier	Strategy Name	Section	Type
G1	CONSISTENT RANDOM PRIZE	§3.1.2	**S**
G2	NEAREST NEIGHBOUR	§3.2.1.1	**D**
G3	GREEDY	§3.2.1.3	**D**
G4	SUBGRAVITY GREEDY	§3.2.1.3	**D**
G5	GREEDY EN ROUTE INSERTION	§3.2.2.2	**D**
G6	GREEDY PAIR	§3.2.2.3	**D**
G7	GUARANTEED NEAREST NEIGHBOUR	§3.2.3.1	**D**
G8	GUARANTEED GREEDY	§3.2.3.1	**D**
G9	NEAREST NEIGHBOUR AVOIDANCE	§3.2.3.1	**D**
G10	GREEDY AVOIDANCE	§3.2.3.1	**D**
G11	TARGET-SET NEAREST NEIGHBOUR AVOIDANCE	§3.2.3.2	**T**
G12	TARGET-SET GREEDY AVOIDANCE	§3.2.3.2	**T**
S1	GUARANTEE-SUBPATH	§3.3.1	**L**
S2	HORIZON-OP	§3.3.2	**L**
S3	HARVEST PATH	§3.3.3	**L**
S4	$\mathcal{ABA}$-PATH	§3.3.4	**L**
S5	TARGET-SET GUARANTEE-SUBPATH	§3.3.1	**LT**
S6	TARGET-SET $\mathcal{ABA}$-PATH	§3.3.4	**LT**
S7	PRIZE-GUARANTEE	§3.3.1	**D**
S8	PRIZE-PARANOID	§6.1	**M**
S9	ORIGINAL-DT-(MINIMAX)	§6.2	**M**
S10	ORIGINAL-DT-(MAXIMIN)	§6.2	**M**
S11	ORIGINAL-DT-(MINIMAXIMIN)	§6.2	**M**
S12	PRIZE-DT-(MINIMAX)	§6.3	**M**
S13	PRIZE-DT-($\kappa = 0$)	§6.3	**M**
S14	PRIZE-DT-(MAXIMIN)	§6.3	**M**
S15	SINGLE-FAMILY-PRIZE-GUARANTEE	§7.2.2.4	**D**
S16	SINGLE-FAMILY-ORIGINAL-DT-(MINIMAX)	§7.2.2.5	**M**
S17	SINGLE-FAMILY-ORIGINAL-DT-(MAXIMIN)	§7.2.2.5	**M**
S18	SINGLE-FAMILY-ORIGINAL-DT-(MINIMAXIMIN)	§7.2.2.5	**M**
S19	SINGLE-FAMILY-PRIZE-DT-(MINIMAX)	§7.2.2.6	**M**
S20	SINGLE-FAMILY-PRIZE-DT-($\kappa = 0$)	§7.2.2.6	**M**
S21	SINGLE-FAMILY-PRIZE-DT-(MAXIMIN)	§7.2.2.6	**M**

KEY	
D	Deterministic
S	Stochastic
L	Local Search
T	Target-Set
M	Monitored

each subclass cannot provide statistically rigorous results, but should be sufficient to give some indicative results.

10.1.1.3 Results and Analysis

Having completed the design of the small preliminary tournament we can now present the tournament results. The tournament comprised 2000 problem instances and 33 strategies for a total of $2000 \times 33^2 = 2,178,000$ simulation battles.

Let $\{\mathbb{P}_i\}$ be the subclasses of $\mathbb{P}$ and let $|\mathbb{P}| = \sum_i |\mathbb{P}_i|$. A computational tournament is conducted on one subclass, $\mathbb{P}_i$, at a time and the results aggregated over all the subclasses. Specifically, the results from a computational tournament on a particular subclass set of problem instances, $\mathbb{P}_i$, with strategies, $\mathbb{S}$, are divided into *performance* and *computational effort*. In the following, '$\varkappa$' stands for $\varkappa_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$, '$\tau$' stands for $\tau_A(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$, and '$\Xi$' stands for $\Xi(\wp; \mathcal{A} \sim \mathsf{a}, \mathcal{B} \sim \mathsf{b})$.

Performance Results. For each pair of strategies $\mathsf{a}, \mathsf{b} \in \mathbb{S}$:

- $\min_{\wp \in \mathbb{P}_i} \varkappa$
- $\sum_{\wp \in \mathbb{P}_i} \varkappa$
- $\sum_{\wp \in \mathbb{P}_i} \varkappa^2$

Computational Effort Results. For each strategy $\mathsf{a} \in \mathbb{S}$:

- $\sum_{\wp \in \mathbb{P}_i} \sum_{\mathsf{b} \in \mathbb{S}} \left(\frac{\tau}{\Xi}\right)$
- $\sum_{\wp \in \mathbb{P}_i} \sum_{\mathsf{b} \in \mathbb{S}} \left(\frac{\tau}{\Xi}\right)^2$

That is, for each subclass, it is sufficient to produce these specific numerical results. In the following, these *per-subclass* results are aggregated and summarised by strategy and by component subclass.

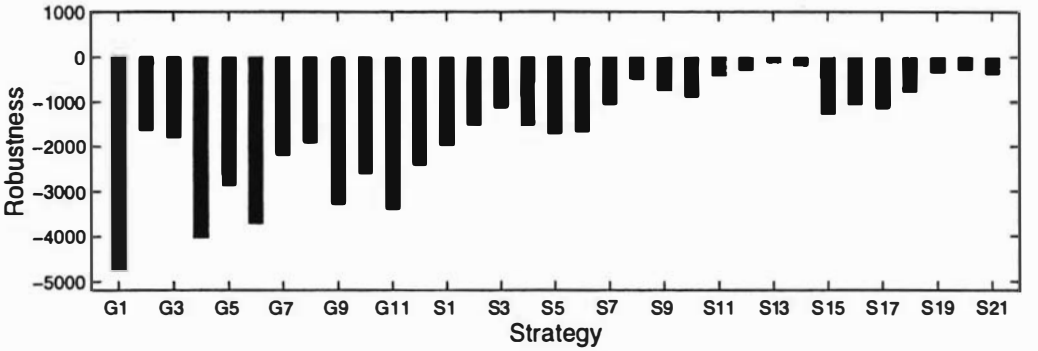
▼ Robustness and Computational Effort of Strategies

Figure 10.1 compares the overall robustness and computational effort of strategies from the small preliminary tournament.

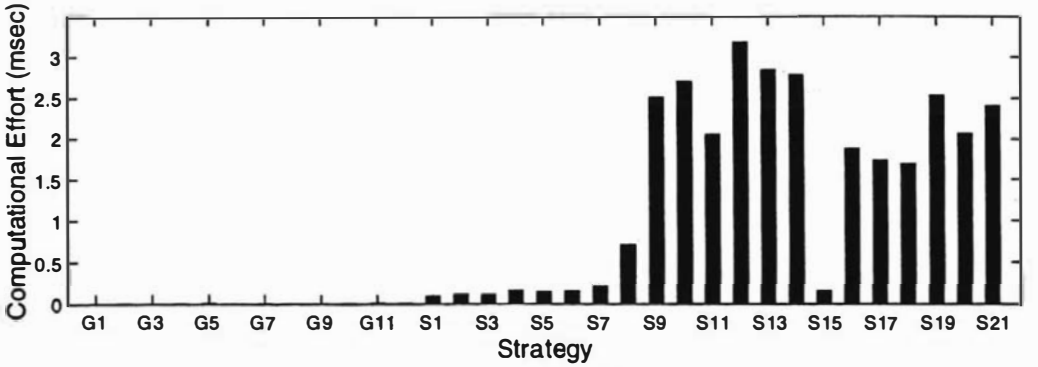
Firstly, Figure 10.1(a) shows the *overall* 'MIN-MIN' *robustness* of each strategy $\mathsf{a} \in \mathbb{S}$, given by Equation (10.8) in which the square brackets indicate that portion of the equation which is a specific, numerical result from a computational tournament on a single subclass $\mathbb{P}_i$.

$$\text{robustness}(\mathsf{a}) = \min_{\mathsf{b} \in \mathbb{S}} \min_{\mathbb{P}_i} \left[\min_{\wp \in \mathbb{P}_i} \varkappa \right] \tag{10.8}$$

The overall tradeoff between computational effort and robustness of strategies stands out, with the strategies requiring more effort generally achieving greater robustness. While the basic and local search based strategies (originally modified from VRSP and TSSP heuristics) are computationally inexpensive, the tactical engine based strategies designed specifically for the CPCP perform much more robustly but with a corresponding increase in computational expense.



(a) Strategy Robustness



(a) Computational Effort

Figure 10.1: Small Preliminary Tournament: Results by Strategy

The three versions of PRIZE-DT were the most robust, followed by the SINGLE-FAMILY-PRIZE-DT strategies and ORIGINAL-DT-(MINIMAXIMIN). Surprisingly, PRIZE-PARANOID also did well, but the MINIMAX and MAXIMIN versions of ORIGINAL-DT strategies were not very robust. Additionally, the two target-set versions of local search based strategies, TARGET-SET GUARANTEE-SUBPATH and TARGET-SET $\mathcal{AB}\mathcal{A}$-PATH, were disappointing. Overall, the tactical engine based strategies performed well with respect to robustness, except for ORIGINAL-DT and PRIZE-GUARANTEE, but the basic and local search based strategies performed poorly. This indicates that in the worst case, a tactical engine based strategy should give a better performance than a strategy not based on a tactical engine.

Secondly, Figure 10.1(b) shows the *computational effort*, $\frac{\tau}{\Xi}$, of each strategy $\mathsf{a} \in \mathbb{S}$ as $\text{mean}(\frac{\tau}{\Xi}|\mathsf{a})$, given by Equation (10.9).

$$\text{mean}(\tfrac{\tau}{\Xi}|\mathsf{a}) = \frac{1}{|\mathbb{S}||\mathbb{P}|} \sum_{\mathbb{P}_i} \left[\sum_{p \in \mathbb{P}_i} \sum_{\mathsf{b} \in \mathbb{S}} \left(\frac{\tau}{\Xi} \right) \right] \tag{10.9}$$

As expected, a pattern with respect to computational effort is evident: the basic strategies are computationally cheap, and the tactical engine based strategies are more than an order of magnitude more computationally expensive than the local search based strategies.

Note that $\frac{\tau}{\Xi}$ is the computational effort per step averaged over a simulation battle and hence $\text{std}(\frac{\tau}{\Xi}|\mathsf{a})$ measures the variation in this average computational effort rather than the variation in the actual computational effort in one step. Thus we cannot determine from these results what proportion of *steps* require little computational effort and what proportion require much computational effort. However, this would be an important investigation for future work.

▼ Summary of Results by Strategy

Table 10.3 presents a more extensive summary of the performance of each strategy, $\mathsf{a} \in \mathbb{S}$, in three sections: *worst subclass*, *nemesis*, and *overall*.

Worst Subclass. For each strategy we determine the worst subclass on which that strategy played, according to three criteria:

- MIN-MIN: $\min_{\mathbb{P}_i} \min_{\mathsf{b} \in \mathbb{S}} \left[\min_{p \in \mathbb{P}_i} \varkappa \right]$
  This is the worst performance of strategy $\mathsf{a} \in \mathbb{S}$ on any problem instance against any opponent, i.e., the *robustness* of the strategy as defined by Equation (10.8).
- MIN-MEAN: $\min_{\mathbb{P}_i} \min_{\mathsf{b} \in \mathbb{S}} \frac{1}{|\mathbb{P}_i|} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$
  This is the worst average performance of strategy $\mathsf{a} \in \mathbb{S}$ on a subclass against a single opponent, averaged over all problem instances in that subclass.
- MEAN-MEAN: $\min_{\mathbb{P}_i} \frac{1}{|\mathbb{S}||\mathbb{P}_i|} \sum_{\mathsf{b} \in \mathbb{S}} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$
  This is the worst average performance of strategy $\mathsf{a} \in \mathbb{S}$ on a subclass, averaged over all opposing strategies and all problem instances in that subclass.

Table 10.3: Small Preliminary Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
G1	-1283	-1177	-4715	{G6}	-1557	{G11}	-471	357
G2	-1059	-946	-1619	{S10}	-568	{S2}	-188	353
G3	-504	-504	-1797	{G3}	-721	{S10}	-511	377
G4	-365	-312	-4023	{S4}	-1865	{G11}	343	368
G5	-591	-566	-2866	{G11}	-263	{G4}	-255	329
G6	-771	-626	-3708	{S7}	-948	{S10}	315	303
G7	-535	-570	-2186	{S13}	-1215	{S7}	-1170	381
G8	-343	-233	-1901	{S2}	432	{G2}	489	314
G9	-1207	-1117	-3273	{G2}	-1064	{G4}	-415	351
G10	-1329	-1329	-2585	{S1}	-1596	{G5}	-1329	426
G11	-1011	-963	-3379	{G11}	-657	{G4}	-636	405
G12	-104	-116	-2403	{G11}	-331	{G3}	-276	341

Table 10.3: Small Preliminary Tournament: Results by Strategy (continued)

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
S1	-830	-669	-1956	{S4}	-679	{S6}	433	339
S2	-146	-120	-1492	{G6}	-131	{G6}	68	332
S3	-477	-499	-1121	{S5}	-690	{G3}	-671	360
S4	-908	-940	-1507	{G11}	-1162	{G5}	-1154	429
S5	-557	-465	-1692	{G6}	-631	{G8}	412	384
S6	-364	-370	-1654	{S4}	-955	{G12}	-894	411
S7	-30	-47	-1029	{G6}	-244	{S9}	-177	337
S8	95	97	-479	{G2}	40	{G6}	111	360
S9	-248	-182	-727	{S5}	-35	{S10}	293	321
S10	-120	-116	-872	{S10}	-89	{S9}	90	348
S11	-39	-18	-409	{S14}	-232	{S9}	215	376
S12	154	165	-274	{S10}	147	{G1}	257	351
S13	209	207	-109	{S9}	130	{G3}	130	304
S14	-51	21	-178	{S21}	404	{G1}	493	302
S15	-618	-530	-1237	{S7}	-588	{G4}	298	371
S16	-464	-427	-1030	{S15}	-315	{S10}	-113	300
S17	-849	-782	-1135	{S18}	-343	{S9}	-338	345
S18	-373	-300	-753	{S10}	-120	{S9}	190	379
S19	92	96	-327	{S16}	115	{G1}	204	369
S20	-57	-57	-279	{S14}	-133	{S1}	-47	315
S21	-115	-70	-362	{S10}	100	{S10}	225	309

Nemesis. The nemesis of a strategy is the strategy which most effectively opposes it over the tournament.

- MIN-MIN: $\min_{b \in \mathbb{S}} \min_{\mathbb{P}_i} \left[\min_{p \in \mathbb{P}_i} \varkappa \right]$

  This is the worst performance of strategy $a \in \mathbb{S}$ against any opponent on any problem instance. Since the MIN-MIN criterion of worst-subclass and nemesis are identical, Table 10.3 only includes the nemesis-MIN-MIN column.

- MIN-MEAN: $\min_{b \in \mathbb{S}} \frac{1}{|\mathbb{P}|} \sum_{\mathbb{P}_i} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$

  This is the worst average performance of strategy $a \in \mathbb{S}$ against a single opponent, averaged over all problem instances.

Overall. For each strategy we determine the overall expected effectiveness of that strategy. This is primary performance measure for a final tournament; it is not required for a preliminary tournament but is useful for comparison.

- MEAN-MEAN: $\frac{1}{|\mathbb{S}||\mathbb{P}|} \sum_{b \in \mathbb{S}} \sum_{\mathbb{P}_i} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$

  This is the average performance of strategy $a \in \mathbb{S}$, averaged over all opposing strategies and all problem instances.

- STD: $\sqrt{\frac{1}{|\mathbb{S}||\mathbb{P}|-1} \left(\sum_{b \in \mathbb{S}} \sum_{\mathbb{P}_i} \left[\sum_{p \in \mathbb{P}_i} \varkappa^2 \right] - \frac{1}{|\mathbb{S}||\mathbb{P}|} \left(\sum_{b \in \mathbb{S}} \sum_{\mathbb{P}_i} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right] \right)^2 \right)}$

  This measures the standard deviation in performance of strategy $a \in \mathbb{S}$ over all opposing strategies and all problem instances.

It is interesting to note that the MIN-MIN nemesis for the basic and local search based strategies were, generally, basic and local search based strategies, and the MIN-MIN nemesis for the tactical engine based strategies were, generally, also tactical engine based strategies. The pattern of robustness shown by the nemesis MIN-MIN column is generally also reflected in the two MIN-MEAN columns (worst subclass and nemesis). However, as expected, the two MEAN-MEAN columns (worst subclass and overall) do not follow this same pattern. This indicates that although the problem instances are not much different in difficulty, the strategies are. The overall STD for each strategy is also reasonably consistent, suggesting that the variation of performances of a strategy, over all subclasses and all opponents, is similar.

▼ Summary of Results by Problem Subclass

Table 10.4 presents a summary of the results for each problem subclass in two sections: best performance by a strategy on that problem subclass, and difficulty of that problem subclass.

Table 10.4: Small Preliminary Tournament: Results by Prize Subclass

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
P:7ls----	0	{S14}	815	{S14}	1261	{G11}	0.188	0.024
P:7pk----	0	{S14}	889	{S12}	1266	{S5}	0.188	0.028
P:7rq----	0	{S13}	599	{S13}	1061	{S5}	0.181	0.022
P:7cw----	0	{S14}	566	{S14}	1060	{S1}	0.181	0.034
P:7ps----	0	{S12}	519	{S13}	729	{G12}	0.179	0.027
P:7cg----	0	{S14}	586	{S13}	792	{G10}	0.177	0.031
P:7cl----	0	{S14}	610	{S13}	841	{S5}	0.175	0.021
P:7rs----	0	{S14}	539	{S14}	739	{G12}	0.174	0.028
P:7hr----	0	{S13}	1083	{S12}	1361	{G12}	0.172	0.019
P:7rc----	0	{S12}	962	{S12}	1185	{S2}	0.169	0.021
P:7pw----	0	{S14}	520	{S14}	799	{S4}	0.168	0.022
P:7hs----	0	{S14}	417	{S14}	794	{G10}	0.167	0.028
P:7pl----	0	{S12}	595	{S13}	854	{G2}	0.167	0.021
P:7ra----	0	{S12}	825	{S13}	1090	{G5}	0.167	0.024
P:7rr----	0	{S14}	605	{S12}	784	{G7}	0.165	0.023
P:7rn----	0	{S12}	582	{S12}	975	{G10}	0.164	0.025
P:7pd----	0	{S12}	840	{S14}	975	{G9}	0.163	0.014
P:7pg----	0	{S13}	889	{S13}	1164	{G10}	0.161	0.020
P:7lk----	0	{S13}	1078	{S12}	1180	{G11}	0.159	0.021
P:7ld----	0	{S13}	1081	{S12}	1195	{S1}	0.146	0.033
P:7--i---	0	{S13}	687	{S13}	1073	{S6}	0.313	0.040
P:7--n---	0	{S12}	735	{S12}	899	{S4}	0.302	0.029
P:7--u---	0	{S13}	701	{S14}	936	{G3}	0.228	0.045
P:7--c---	0	{S13}	864	{S14}	1091	{G11}	0.199	0.034
P:7--q---	0	{S14}	897	{S12}	1089	{G4}	0.192	0.032
P:7--b---	0	{S12}	833	{S14}	1196	{S3}	0.164	0.025
P:7--p---	0	{S13}	956	{S14}	1259	{S6}	0.158	0.035
P:7--s---	0	{S12}	625	{S12}	788	{G12}	0.146	0.033
P:7--e---	0	{S12}	799	{S12}	1260	{G11}	0.146	0.026
P:7---mi-	0	{S14}	982	{S13}	1219	{G11}	0.301	0.039
P:7---li-	0	{S13}	725	{S12}	1051	{S2}	0.277	0.027
P:7---bi-	0	{S12}	548	{S12}	872	{G9}	0.274	0.038
P:7---ci-	0	{S14}	1047	{S12}	1227	{G3}	0.273	0.044
P:7---bm-	0	{S12}	945	{S13}	1219	{S6}	0.230	0.032
P:7---mm-	0	{S12}	718	{S13}	1027	{G3}	0.219	0.028
P:7---ll-	0	{S12}	759	{S14}	935	{G6}	0.217	0.033
P:7---cm-	0	{S13}	602	{S14}	800	{G3}	0.208	0.034
P:7---ss-	0	{S12}	824	{S12}	992	{G12}	0.172	0.017
P:7---cb-	0	{S14}	1025	{S13}	1292	{G12}	0.171	0.018
P:7---bb-	0	{S14}	905	{S14}	1241	{S1}	0.144	0.015
P:7-----r	0	{S12}	669	{S12}	1062	{S5}	0.180	0.027
P:7-----i	0	{S14}	900	{S13}	1275	{G4}	0.166	0.030

Best Strategy. For each problem subclass, $\mathbb{P}_i$, we determine the best performing strategy on that subclass, according to three criteria:

- MIN-MIN: $\max_{a \in \mathbb{S}} \min_{b \in \mathbb{S}} \left[\min_{p \in \mathbb{P}_i} \varkappa \right]$
- MIN-MEAN: $\max_{a \in \mathbb{S}} \min_{b \in \mathbb{S}} \frac{1}{|\mathbb{P}_i|} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$
- MEAN-MEAN: $\max_{a \in \mathbb{S}} \frac{1}{|\mathbb{S}|} \sum_{b \in \mathbb{S}} \frac{1}{|\mathbb{P}_i|} \left[\sum_{p \in \mathbb{P}_i} \varkappa \right]$

These characterise the difficulty of the problem subclass by the best performing strategy in terms of MAXIMIN over worst performance and average performance, and MAX over all performances.

Difficulty. Recall that the difficulty, $\xi(\wp)$, of a problem instance was defined in Equation (10.7). The mean (MEAN) and standard deviation (STD) of $\xi(\wp)$ over all problem instances $\wp \in \mathbb{P}_i$ are given.

Table 10.4 is divided into four parts corresponding to the four components of the P-class of problem instances: prize location, prize value, player location and overall deadline. The subclasses within each part are sorted according to MEAN difficulty (greater is more difficult). There is little distinction between the different prize location subclasses and the overall deadline subclasses. However, the prize value subclasses appear to fall into three clusters, {i,n}, {u,c,q}, and {b,p,s,e}, in decreasing order of difficulty, over a much wider range of difficulty values than the prize location subclasses. This indicates that problem instances with a small number of high value prizes are easier than those with similar prize values. The player location subclasses also appear to fall into three clusters, {mi,li,bi,ci}, {bm,mm,ll,cm}, and {ss,cb,bb}. This indicates that problem instances with separated initial player locations are easier than those with initial player locations close together.

Note that MIN-MIN is always zero, indicating that there is a problem instance in every subclass such that the corresponding best MIN-MIN strategy defines both $\upsilon_A^\diamond(\wp)$ and $\upsilon_B^\diamond(\wp)$. In addition, the best MIN-MIN strategy and best MIN-MEAN strategy are consistently selected from one of the PRIZE-DT strategies, indicating that these are most robust strategies overall. This appears to also justify the choice of PRIZE-DT in the definition of $\xi(\wp)$. The performance of the best MIN-MEAN strategy and best MEAN-MEAN strategy do not appear to follow the same pattern as the MEAN difficulty, which suggests that average difficulty is not necessarily related to average performance over a subclass of problem instances.

10.1.1.4 Conclusions from the Small Preliminary Tournament

The lack of robustness of ORIGINAL-DT was unexpected. This may suggest that the target-prize determined by a *probe* is confusing to the player which does not reach that target-prize. A greater understanding of the tactics determined by the tactical engines of Chapter 6 is an important topic of future investigation.

In conclusion, the strategies to be considered for the small final tournament (Section 10.2.1) are: **S8**, **S11**, **S12**, **S13**, **S14**, **S19**, **S20**, and **S21**. We will not consider the prize location or overall deadline components in the small final tournament, since the results with respect to these components were inconclusive. Rather, we will concentrate on the {i,n,u,c,q} subclasses of the prize value component and the {mi,li,bi,ci} subclasses of the player location component.

10.1.2 Small-Medium Preliminary Tournament

The aim of the small-medium preliminary tournament is to compare the robustness of, and computational effort required by, the DMS strategies designed in Chapter 7 (both tactical engine based and strategic engine based) on clustered problem instances, and to compare the difficulty of contributing components of the problem instance classes. The focus is on cluster-based strategies restricted to a single family. The defining strategy from Table 10.1 is SINGLE-FAMILY-PRIZE-DT, which is certainly capable of implicitly searching game trees on up to twenty prizes.

10.1.2.1 Participating Strategies

The participants in the small-medium preliminary tournament are those listed in Table 10.5 plus strategies **G1**–**G12** from Table 10.2.

The notation for DMS strategies involving a strategic engine at the *cluster-frame* and a tactical engine at the *prize-frame* is to separate strategic engine specification from the tactical engine specification by '//'. The prefix '(SF)' indicates that the strategy (actually the *global-frame* of the strategy) is restricted to a single family.

Each cluster-based strategy is restricted to a single family and to four clusters determined using the prize-value-weighted-Ward method with improvements. The ANY–ALL–PCTSP requirement is set to 'PCTSP' for each cluster and the required value on cluster $[ci]$, $\eta_{[ci]}$, is set to $\frac{1}{2}v([ci])$, i.e., half the total value of the cluster.

10.1.2.2 Problem Instances

We are interested in clustered problems, so all problem instances considered here are from the C-class (see Section 9.1.2) for which the defining components are: number of clusters, cluster location subclass, cluster value subclass, cluster size subclass, cluster scale subclass, player location subclass, and overall deadline subclass. The number of clusters is fixed at four, and the number of prizes is fixed at twenty. This is sufficiently challenging tactically for SINGLE-FAMILY-PRIZE-DT but not too computationally intensive. Only twenty problem instances were generated for each subclass because we have approximately twice as many subclasses as in the small preliminary tournament and the expected duration of each battle simulation is also double since there are approximately three times the number of prizes within the same area. The 20 problem instances for each of the 48 player location subclasses were redistributed amongst the other component subclasses, so that there is a total of $20 \times 48 = 960$ distinct problem instances, as opposed to the 2000 distinct P-class problem instances in the small preliminary tournament.

Table 10.5: Small-Medium Preliminary Tournament: Participating Strategies

Identifier	Strategy Name	Section	Type
SM1	SINGLE-FAMILY-PRIZE-GUARANTEE	§7.2.2.4	**D**
SM2	SINGLE-FAMILY-ORIGINAL-DT-(MINIMAX)	§7.2.2.5	**M**
SM3	SINGLE-FAMILY-ORIGINAL-DT-(MAXIMIN)	§7.2.2.5	**M**
SM4	SINGLE-FAMILY-ORIGINAL-DT-(MINIMAXIMIN)	§7.2.2.5	**M**
SM5	SINGLE-FAMILY-PRIZE-DT-(MINIMAX)	§7.2.2.6	**M**
SM6	SINGLE-FAMILY-PRIZE-DT-($\kappa = 0$)	§7.2.2.6	**M**
SM7	SINGLE-FAMILY-PRIZE-DT-(MAXIMIN)	§7.2.2.6	**M**
SM8	(SF)-CLUSTER-GUARANTEE//PRIZE-GUARANTEE	§7.3.3	**D**
SM9	(SF)-CLUSTER-GUARANTEE//PRIZE-DT-(MINIMAX)	§7.3.3	**D**
SM10	(SF)-CLUSTER-GUARANTEE//PRIZE-DT-($\kappa = 0$)	§7.3.3	**D**
SM11	(SF)-CLUSTER-GUARANTEE//PRIZE-DT-(MAXIMIN)	§7.3.3	**D**
SM12	(SF)-CLUSTER-PARANOID//PRIZE-GUARANTEE	§7.4	**M**
SM13	(SF)-CLUSTER-PARANOID//PRIZE-DT-(MINIMAX)	§7.4	**M**
SM14	(SF)-CLUSTER-PARANOID//PRIZE-DT-($\kappa = 0$)	§7.4	**M**
SM15	(SF)-CLUSTER-PARANOID//PRIZE-DT-(MAXIMIN)	§7.4	**M**
SM16	(SF)-CLUSTER-DT-(MINIMAX)//PRIZE-GUARANTEE	§7.5	**M**
SM17	(SF)-CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MINIMAX)	§7.5	**M**
SM18	(SF)-CLUSTER-DT-(MINIMAX)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
SM19	(SF)-CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
SM20	(SF)-CLUSTER-DT-($\kappa = 0$)//PRIZE-GUARANTEE	§7.5	**M**
SM21	(SF)-CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MINIMAX)	§7.5	**M**
SM22	(SF)-CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
SM23	(SF)-CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
SM24	(SF)-CLUSTER-DT-(MAXIMIN)//PRIZE-GUARANTEE	§7.5	**M**
SM25	(SF)-CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MINIMAX)	§7.5	**M**
SM26	(SF)-CLUSTER-DT-(MAXIMIN)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
SM27	(SF)-CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MAXIMIN)	§7.5	**M**

10.1.2.3 Results and Analysis

We can now present the results from the small-medium preliminary tournament that comprised 960 problem instances and 39 strategies, for a total of $960 \times 39^2 = 1,460,160$ simulation battles. These results follow the same format as for the small preliminary tournament (see Section 10.1.1).

▼ Robustness and Computational Effort of Strategies

Figure 10.2 compares the overall robustness and computational effort by strategy, following the same specification as Figure 10.1. Figure 10.2(a) shows the *overall* 'MIN-MIN' *robustness* of each strategy according to Equation (10.8). Figure 10.2(b) shows the *computational effort*, mean($\frac{\tau}{\equiv}$), of each strategy according to Equation (10.9).

The PRIZE-DT strategies (**SM5–SM7**) and the CLUSTER-DT-($\kappa = 0$) strategies (**SM20–SM23**) appear to be the most robust of these strategies, although the PRIZE-DT strategies are more robust. The 'PRIZE-' methods are generally less computationally expensive than the 'CLUSTER-' methods, although the difference is not great, since a strategic engine is invoked reasonably infrequently and thus its computational demand is averaged over a relatively large number of steps.

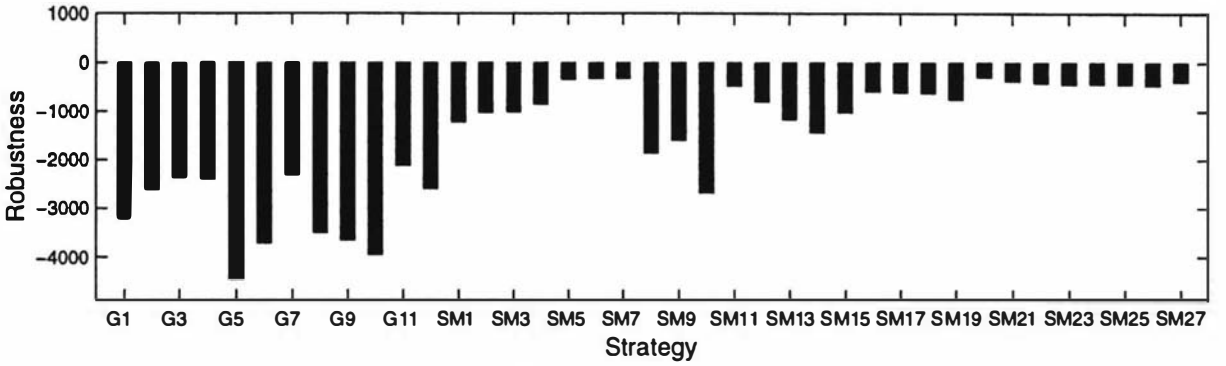
▼ Summary of Results by Strategy

Table 10.6 presents a more extensive summary of the performance of each strategy, following the same specification as Table 10.3 from the small preliminary tournament. We observe that the MIN-MIN nemesis for the tactical and strategic engine based strategies are, generally, tactical engine based strategies. Note that the overall MEAN-MEAN does not follow the same pattern as the nemesis MIN-MIN, but is similar to nemesis MIN-MIN.

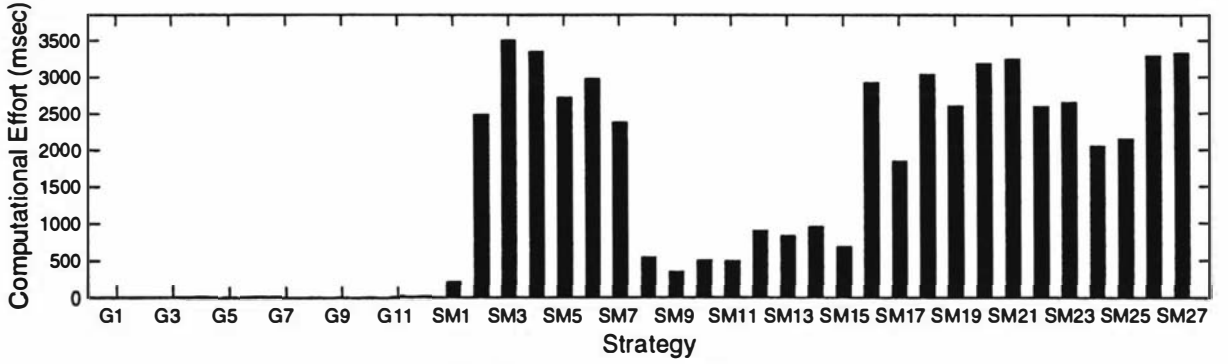
▼ Summary of Results by Problem Subclass

Table 10.7 presents a summary of the best performance by a strategy on each problem subclass, and the difficulty of each problem subclass, following the same specification as Table 10.4. The exception is that the difficulty, $\xi(\wp)$, is defined in terms of SINGLE-FAMILY-PRIZE-DT-($\kappa = 0$) (instead of PRIZE-DT-($\kappa = 0$)) since this is the defining strategy for this tournament, c.f., Equation (10.7). The table is divided into five parts, corresponding to five components of the C-class of problem instances: cluster location, cluster value, cluster size and scale, overall deadline, and player location. The subclasses within each part are sorted according to MEAN difficulty (greater is more difficult).

There is little distinction between the cluster location subclasses, but this is consistent with the similar lack of distinction between the prize location subclasses of the small preliminary tournament. Similarly, the overall deadline subclasses are not observably different in terms of any of the criteria. However, the cluster value subclasses appear to fall into the same three groups as the prize value subclasses in the small preliminary tournament. This is not surprising since the prize location and prize value subclasses of P-class are used directly as the cluster location and cluster value subclasses of C-class, and the clusters were originally designed as "fractured prizes".



(a) Strategy Robustness



(b) Computational Effort

Figure 10.2: Small-Medium Preliminary Tournament: Results by Strategy

Table 10.6: Small-Medium Preliminary Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
G1	-110	-85	-3231	{SM2}	120	{G11}	363	316
G2	-843	-839	-2631	{G8}	-808	{G11}	-773	438
G3	-165	-121	-2378	{SM14}	139	{SM3}	214	320
G4	-947	-974	-2424	{SM18}	-1572	{SM2}	-1498	529
G5	-910	-911	-4452	{SM7}	-1713	{SM8}	-924	323
G6	-783	-637	-3708	{G9}	-142	{G11}	290	357
G7	-962	-921	-2328	{SM8}	-868	{G10}	339	301
G8	-722	-663	-3493	{G6}	-187	{G9}	61	371
G9	-614	-587	-3647	{SM18}	-752	{SM7}	-400	305
G10	-1338	-1213	-3948	{SM19}	-1068	{SM8}	-321	344
G11	-62	-81	-2115	{G6}	-615	{SM7}	-586	324
G12	-268	-329	-2596	{G9}	-792	{SM8}	-731	395

Table 10.6: Small-Medium Preliminary Tournament: Results by Strategy (continued)

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
SM1	-231	-266	-1218	{SM18}	-641	{SM7}	-628	408
SM2	-398	-382	-1023	{SM27}	-708	{G10}	-241	323
SM3	-337	-281	-1009	{SM18}	-324	{G11}	303	327
SM4	-129	-124	-854	{SM20}	-9	{SM19}	18	362
SM5	-49	-38	-340	{SM19}	217	{SM3}	392	360
SM6	193	193	-325	{SM22}	210	{SM4}	213	305
SM7	-248	-244	-324	{SM19}	88	{SM2}	98	373
SM8	-812	-798	-1861	{SM18}	-865	{G11}	-592	304
SM9	-951	-842	-1594	{SM19}	96	{G11}	408	319
SM10	-606	-660	-2676	{SM6}	-1108	{SM2}	-1027	388
SM11	-167	-160	-474	{SM17}	-17	{SM19}	123	307
SM12	-355	-256	-806	{SM5}	388	{SM3}	439	320
SM13	-599	-579	-1161	{SM18}	-309	{SM7}	-236	303
SM14	-722	-713	-1428	{SM25}	-867	{SM13}	-648	336
SM15	-253	-185	-1014	{SM5}	-2	{G10}	385	357
SM16	-301	-273	-584	{SM7}	-10	{SM7}	18	333
SM17	-175	-166	-608	{SM24}	-231	{SM8}	-90	357
SM18	25	32	-625	{SM25}	-26	{SM8}	416	348
SM19	-369	-346	-754	{SM24}	-190	{SM7}	-189	311
SM20	-112	-52	-299	{SM19}	96	{SM3}	391	373
SM21	-7	27	-375	{SM19}	184	{SM4}	335	348
SM22	-233	-222	-420	{SM19}	-151	{SM19}	-23	325
SM23	-311	-233	-444	{SM25}	247	{G9}	262	332
SM24	-143	-97	-436	{SM23}	-107	{SM19}	186	310
SM25	-11	-2	-440	{SM17}	-42	{SM4}	186	372
SM26	-198	-192	-466	{SM19}	174	{SM2}	193	301
SM27	34	43	-393	{SM19}	105	{SM2}	255	350

Table 10.7: Small-Medium Preliminary Tournament: Results by Cluster Subclass

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
C:4pl------	0	{SM19}	763	{SM6}	953	{G8}	0.220	0.066
C:4ld------	**0**	{SM19}	721	{SM6}	961	{G8}	0.215	0.054
C:4hs------	**0**	{SM17}	408	{SM25}	656	{G3}	0.213	0.038
C:4rc------	0	{SM26}	655	{SM25}	831	{G4}	0.211	0.043
C:4pk------	0	{SM7}	516	{SM6}	741	{G12}	0.210	0.051
C:4lk------	**0**	{SM19}	366	{SM6}	751	{G5}	0.207	0.048
C:4rn------	0	{SM25}	631	{SM5}	964	{G9}	0.207	0.062
C:4rr------	**0**	{SM23}	683	{SM5}	998	{G2}	0.206	0.053
C:4rq------	0	{SM19}	782	{SM16}	1047	{G4}	0.202	0.041
C:4hr------	0	{SM22}	706	{SM20}	815	{G8}	0.201	0.061
C:4cw------	0	{SM19}	690	{SM19}	913	{G4}	0.198	0.058
C:4ps------	**0**	{SM21}	558	{SM22}	743	{G12}	0.197	0.034
C:4pg------	272	{SM17}	699	{SM22}	948	{G7}	0.197	0.046
C:4rs------	0	{SM7}	847	{SM21}	863	{G3}	0.196	0.044
C:4ls------	231	{SM18}	693	{SM26}	904	{G8}	0.195	0.056
C:4pd------	0	{SM20}	501	{SM20}	614	{G4}	0.194	0.039
C:4pc------	182	{SM22}	632	{SM18}	914	{SM9}	0.191	0.064
C:4cl------	**0**	{SM26}	600	{SM6}	878	{SM11}	0.186	0.056
C:4cg------	0	{SM25}	339	{SM6}	465	{SM15}	0.182	0.043
C:4ra------	0	{SM5}	610	{SM6}	685	{SM14}	0.181	0.041
C:4--i-----	0	{SM5}	581	{SM23}	917	{G6}	0.384	0.077
C:4--n-----	0	{SM16}	562	{SM19}	866	{G10}	0.317	0.061
C:4--u-----	0	{SM27}	540	{SM27}	855	{G12}	0.238	0.086
C:4--c-----	0	{SM27}	715	{SM19}	976	{SM14}	0.230	0.053
C:4--q-----	232	{SM17}	666	{SM24}	852	{G7}	0.216	0.067
C:4--e-----	0	{SM5}	411	{SM5}	558	{G5}	0.186	0.056
C:4--p-----	0	{SM6}	480	{SM20}	936	{SM10}	0.173	0.063
C:4--b-----	**0**	{SM19}	537	{SM16}	818	{G9}	0.161	0.023
C:4--s-----	0	{SM21}	423	{SM23}	706	{G4}	0.161	0.054
C:4---rl---	0	{SM21}	450	{SM19}	840	{G9}	0.255	0.065
C:4---et---	0	{SM16}	427	{SM27}	646	{G2}	0.240	0.047
C:4---el---	0	{SM5}	397	{SM27}	656	{G6}	0.232	0.045
C:4---rm---	281	{SM26}	690	{SM24}	1067	{G6}	0.194	0.034
C:4---rt---	0	{SM7}	790	{SM6}	952	{G12}	0.186	0.051
C:4---em---	0	{SM27}	474	{SM24}	662	{G11}	0.159	0.055
C:4-------i	158	{SM18}	595	{SM17}	1028	{SM12}	0.199	0.065
C:4-------r	134	{SM20}	691	{SM6}	980	{SM15}	0.197	0.064

Table 10.7: Small-Medium Preliminary Tournament: Results by Cluster Subclass (continued)

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
C:4-----ai-	235	{SM17}	608	{SM22}	912	{SM15}	0.307	0.088
C:4-----eh-	0	{SM7}	636	{SM7}	741	{G8}	0.305	0.020
C:4-----aw-	0	{SM25}	528	{SM20}	810	{G3}	0.301	0.072
C:4-----bh-	0	{SM24}	433	{SM16}	718	{SM12}	0.301	0.086
C:4-----bb-	0	{SM23}	353	{SM23}	527	{SM14}	0.296	0.080
C:4-----bw-	0	{SM19}	729	{SM20}	921	{G11}	0.295	0.072
C:4-----ei-	0	{SM25}	721	{SM21}	833	{G6}	0.294	0.025
C:4-----ba-	0	{SM25}	424	{SM27}	665	{G6}	0.290	0.086
C:4-----dw-	0	{SM25}	776	{SM27}	1053	{SM11}	0.290	0.088
C:4-----cc-	0	{SM27}	785	{SM16}	1129	{G3}	0.283	0.068
C:4-----ac-	0	{SM16}	437	{SM6}	629	{G4}	0.281	0.066
C:4-----aa-	195	{SM23}	538	{SM17}	723	{SM15}	0.278	0.088
C:4-----dh-	0	{SM20}	419	{SM22}	669	{SM9}	0.270	0.075
C:4-----bi-	0	{SM6}	504	{SM16}	886	{G4}	0.269	0.075
C:4-----bc-	0	{SM5}	477	{SM25}	786	{G7}	0.267	0.065
C:4-----ca-	0	{SM20}	481	{SM16}	799	{G5}	0.266	0.087
C:4-----di-	168	{SM6}	691	{SM5}	981	{G3}	0.265	0.077
C:4-----ab-	0	{SM6}	635	{SM16}	911	{G8}	0.264	0.089
C:4-----ah-	212	{SM6}	683	{SM18}	961	{SM8}	0.260	0.072
C:4-----cb-	0	{SM26}	682	{SM5}	916	{G4}	0.249	0.088
C:4-----ew-	0	{SM16}	742	{SM7}	1206	{G7}	0.248	0.023
C:4-----dc-	0	{SM22}	693	{SM18}	984	{SM12}	0.173	0.016
C:4-----df-	0	{SM20}	628	{SM25}	1044	{SM14}	0.171	0.063
C:4-----ce-	0	{SM18}	371	{SM20}	746	{SM8}	0.170	0.025
C:4-----da-	0	{SM23}	452	{SM21}	596	{SM15}	0.168	0.017
C:4-----ff-	0	{SM18}	553	{SM24}	785	{SM15}	0.166	0.054
C:4-----cf-	0	{SM16}	572	{SM23}	758	{G3}	0.166	0.021
C:4-----de-	0	{SM19}	883	{SM17}	1086	{SM10}	0.163	0.057
C:4-----eb-	0	{SM22}	697	{SM23}	970	{SM10}	0.160	0.017
C:4-----db-	0	{SM23}	654	{SM21}	846	{G4}	0.158	0.019
C:4-----ee-	0	{SM6}	371	{SM6}	544	{G8}	0.158	0.022
C:4-----fb-	0	{SM25}	593	{SM23}	824	{G5}	0.156	0.021
C:4-----ea-	292	{SM18}	423	{SM17}	821	{G7}	0.153	0.019
C:4-----fd-	257	{SM26}	600	{SM7}	869	{G6}	0.153	0.059
C:4-----dd-	0	{SM19}	845	{SM27}	1224	{G3}	0.150	0.064
C:4-----cd-	0	{SM17}	550	{SM5}	602	{G10}	0.149	0.018
C:4-----bf-	252	{SM21}	695	{SM25}	967	{G6}	0.147	0.019
C:4-----fc-	346	{SM6}	788	{SM22}	1006	{SM15}	0.144	0.023
C:4-----bd-	0	{SM7}	474	{SM21}	714	{SM15}	0.141	0.021
C:4-----fa-	0	{SM26}	732	{SM26}	991	{SM8}	0.135	0.021
C:4-----ed-	215	{SM21}	693	{SM25}	998	{SM14}	0.135	0.018
C:4-----ad-	0	{SM5}	456	{SM26}	823	{SM8}	0.127	0.021
C:4-----ae-	0	{SM16}	762	{SM25}	1212	{G7}	0.127	0.022
C:4-----ef-	355	{SM5}	377	{SM23}	581	{G12}	0.121	0.020
C:4-----be-	0	{SM17}	359	{SM26}	648	{SM9}	0.121	0.024
C:4-----ec-	233	{SM26}	421	{SM23}	589	{G12}	0.115	0.015
C:4-----fe-	252	{SM17}	487	{SM6}	745	{G6}	0.114	0.062
C:4-----af-	0	{SM24}	478	{SM16}	817	{G5}	0.106	0.019

It also indicates that the cluster value as a whole is more significant, in terms of robustness, than the distribution of prizes within the cluster. The player location subclasses (defined differently that in the small preliminary tournament) appear to fall into two basic groups. Firstly, those player location subclasses which involve any of {h,i,w} tend to be quite difficult; these subclasses correspond to player locations within the same cluster. Also difficult are those player location subclasses which involve any two of {a,b,c}; these subclasses correspond to player locations which are both central to the prize region. Secondly, the remaining subclasses appear to easier; a tenable explanation is that these player locations are, on average, not close together.

10.1.2.4 Conclusions from the Small-Medium Preliminary Tournament

In conclusion, the strategies to be considered for the small-medium final tournament (Section 10.2.2) are: **SM5**, **SM6**, **SM7**, **SM20**, **SM21**, **SM22**, and **SM23**. Additionally, we will only consider the cluster value and player location components of the C-class of problem instances. In particular we will concentrate on the {i,n,u,c,q} subclasses of the cluster value component, and subclasses of the the player location component which involve any of {h,i,w} or any pair from {a,b,c}.

10.1.3 Medium Preliminary Tournament

The aim of the medium preliminary tournament is to compare the robustness of, and computational effort required by, the DMS strategies designed in Chapter 7 (both tactical-engine-based and strategic-engine-based) on clustered problem instances, and to compare the difficulty of contributing components of the problem instance classes. The focus is on family-cluster-based strategies exploiting a full family-cluster structure, i.e., there is no restriction of the number of families to a single family. The defining strategy from Table 10.1 is FAMILY-PRIZE-DT, which is capable of implicitly searching game trees up to 40 prizes as long as there are sufficiently many clusters, each of which are not too large, i.e., FAMILY-PRIZE-DT is intractable when there are few large clusters or many small clusters.

10.1.3.1 Participating Strategies

The participants in the medium preliminary tournament are those listed in Table 10.8 plus strategies **G1**–**G12** from Table 10.2.

To ensure that the 'FAMILY-' strategies remain computationally feasible, we require sufficient cluster such that there are not too many prizes per cluster and not too many clusters per family. However, we also wish to have sufficiently many prizes so that there the possibility of some intra-cluster interaction between the players. In constructing the problem instances, we will fix the number of prizes at 40. Hence, each cluster-based strategy is standardised to a common family-cluster structure consisting of nine clusters and three families. The clustering method is prize-value-weighted-Ward with improvement and the family method is single linkage agglomerative clustering, i.e., nearest neighbouring cluster.

The ANY–ALL–PCTSP requirement is set to 'ALL' for each cluster since, with the overall focus

Table 10.8: Medium Preliminary Tournament: Participating Strategies

Identifier	Strategy Name	Section	Type
M1	FAMILY-PRIZE-GUARANTEE	§7.2.2.4	**D**
M2	FAMILY-ORIGINAL-DT-(MINIMAX)	§7.2.2.5	**M**
M3	FAMILY-ORIGINAL-DT-(MAXIMIN)	§7.2.2.5	**M**
M4	FAMILY-ORIGINAL-DT-(MINIMAXIMIN)	§7.2.2.5	**M**
M5	FAMILY-PRIZE-DT-(MINIMAX)	§7.2.2.6	**M**
M6	FAMILY-PRIZE-DT-($\kappa = 0$)	§7.2.2.6	**M**
M7	FAMILY-PRIZE-DT-(MAXIMIN)	§7.2.2.6	**M**
M8	CLUSTER-GUARANTEE//PRIZE-GUARANTEE	§7.3.3	**D**
M9	CLUSTER-GUARANTEE//PRIZE-DT-(MINIMAX)	§7.3.3	**D**
M10	CLUSTER-GUARANTEE//PRIZE-DT-($\kappa = 0$)	§7.3.3	**D**
M11	CLUSTER-GUARANTEE//PRIZE-DT-(MAXIMIN)	§7.3.3	**D**
M12	CLUSTER-PARANOID//PRIZE-GUARANTEE	§7.4	**M**
M13	CLUSTER-PARANOID//PRIZE-DT-(MINIMAX)	§7.4	**M**
M14	CLUSTER-PARANOID//PRIZE-DT-($\kappa = 0$)	§7.4	**M**
M15	CLUSTER-PARANOID//PRIZE-DT-(MAXIMIN)	§7.4	**M**
M16	CLUSTER-DT-(MINIMAX)//PRIZE-GUARANTEE	§7.5	**M**
M17	CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MINIMAX)	§7.5	**M**
M18	CLUSTER-DT-(MINIMAX)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
M19	CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
M20	CLUSTER-DT-($\kappa = 0$)//PRIZE-GUARANTEE	§7.5	**M**
M21	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MINIMAX)	§7.5	**M**
M22	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
M23	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
M24	CLUSTER-DT-(MAXIMIN)//PRIZE-GUARANTEE	§7.5	**M**
M25	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MINIMAX)	§7.5	**M**
M26	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
M27	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MAXIMIN)	§7.5	**M**

on family-clustered problems; we can simplify the intra-cluster planning for both the 'FAMILY-' and 'CLUSTER-' strategies by assuming that all prizes within a cluster are to claimed contiguously.

10.1.3.2 Problem Instances

We are interested in clustered problems, so all problem instances considered here are from the C-class (see Section 9.1.2) for which the defining components are: number of clusters, cluster location subclass, cluster value subclass, cluster size subclass, cluster scale subclass, player location subclass, and overall deadline subclass. Note that there is no specification of the number of families in the construction of the problem subclasses. Rather, the number of families is standardised between the strategies so that they can be more equitabily compared.

For reasons given above, the number of clusters is fixed at nine, and the number of prizes fixed at 40. As in the small-medium preliminary tournament, only 20 problem instances were generated for each subclass. The 20 problem instances for each of the player location subclasses were redistributed between the other subclasses.

10.1.3.3 Results and Analysis

The results from the medium preliminary tournament follow the same format as for the small preliminary tournament (see Section 10.1.1).

▼ Robustness and Computational Effort of Strategies

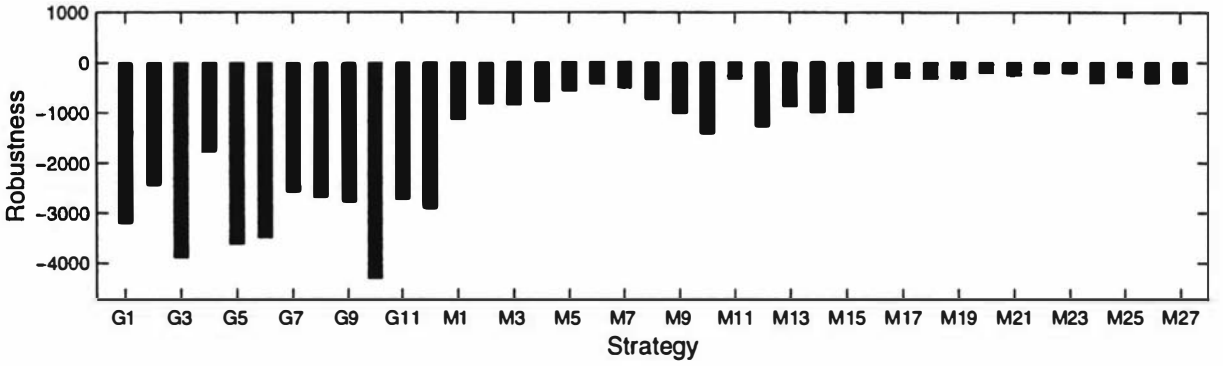
Figure 10.3 compares the overall robustness and computational effort by strategy, following the same specification as Figure 10.1. Figure 10.3(a) shows the *overall* 'MIN-MIN' *robustness* of each strategy and Figure 10.3(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$). As in the small-medium preliminary tournament, the PRIZE-DT strategies (**M5–M7**) and CLUSTER-DT-($\kappa = 0$) strategies (**M20–M23**) are the most robust and among the most expensive. However, in the exact reverse of the small-medium preliminary tournament, the PRIZE-DT strategies are more computationally expensive than the CLUSTER-DT-($\kappa = 0$) strategies, and the PRIZE-DT strategies are not as robust as the CLUSTER-DT-($\kappa = 0$) strategies.

▼ Summary of Results by Strategy

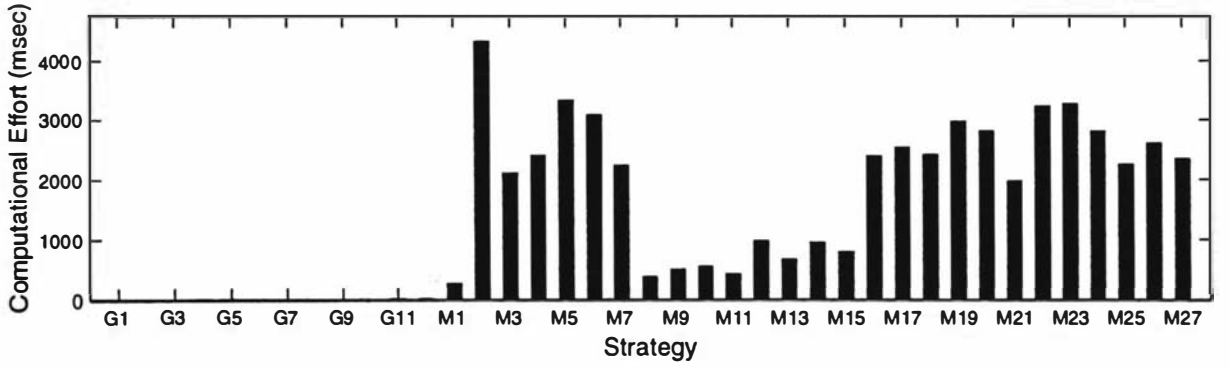
Table 10.9 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.3. Although the worst subclass columns appear to follow a similar pattern, the nemesis MIN-MIN, nemesis MIN-MEAN and overall MEAN-MEAN are all different. This supports the notion that strategies which perform effectively over the range of opponent strategies are not necessarily the most robust in the worst case.

▼ Summary of Results by Problem Subclass

Table 10.10 presents a summary of the best performance by a strategy on each problem subclass, and the difficulty of each problem subclass, following the same specification as Table 10.4. The



(a) Strategy Robustness



(b) Computational Effort

Figure 10.3: Medium Preliminary Tournament: Results by Strategy

Table 10.9: Medium Preliminary Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
G1	-1290	-1275	-3205	**{G4}**	-984	**{M3}**	-980	333
G2	-294	-336	-2448	**{G5}**	-1268	**{M26}**	-982	448
G3	-19	8	-3880	**{G7}**	69	**{M6}**	181	370
G4	-634	-627	-1775	**{G4}**	-79	**{M27}**	-69	360
G5	-667	-594	-3602	**{M20}**	77	**{M12}**	453	386
G6	-642	-628	-3479	**{M10}**	-1385	**{M12}**	-517	359
G7	-983	-926	-2580	**{G8}**	-1196	**{M27}**	-493	317
G8	-941	-958	-2680	**{G6}**	-1486	**{M27}**	-1096	434
G9	-632	-492	-2784	**{M3}**	467	**{M9}**	467	359
G10	-713	-713	-4302	**{G11}**	-1333	**{M12}**	-713	321
G11	-970	-992	-2725	**{M5}**	-1587	**{M20}**	-1371	367
G12	-1093	-1051	-2920	**{G10}**	-515	**{M12}**	91	321

Table 10.9: Medium Preliminary Tournament: Results by Strategy (continued)

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
M1	-200	-182	-1130	{M5}	-445	{M12}	358	313
M2	-212	-199	-819	{M20}	-208	{G7}	13	344
M3	-276	-187	-852	{M22}	383	{M6}	436	302
M4	-215	-186	-774	{M23}	-158	{M20}	174	363
M5	-136	-132	-566	{M23}	-5	{M9}	16	308
M6	-310	-271	-430	{M24}	374	{M20}	477	351
M7	-134	-123	-511	{M5}	-343	{G7}	95	343
M8	-432	-333	-728	{M21}	-3	{G7}	473	394
M9	-341	-248	-1013	{M27}	-529	{G7}	482	379
M10	-821	-748	-1407	{M20}	-854	{M26}	-119	327
M11	162	185	-323	{M19}	-46	{G7}	482	372
M12	-931	-922	-1267	{M27}	-15	{M27}	398	373
M13	-237	-234	-872	{M20}	-249	{M12}	-40	334
M14	-277	-277	-1002	{M20}	-397	{M27}	-276	342
M15	-374	-360	-989	{M22}	-636	{M20}	385	316
M16	-26	-7	-493	{M25}	295	{M26}	379	344
M17	-40	-1	-295	{M26}	180	{M27}	276	323
M18	-11	14	-322	{M19}	-160	{G6}	331	370
M19	43	46	-324	{M21}	-90	{G10}	181	309
M20	138	136	-206	{M26}	37	{M3}	37	382
M21	133	135	-263	{M6}	267	{G7}	275	309
M22	-1	36	-215	{M5}	52	{M20}	295	316
M23	159	163	-209	{M19}	178	{M3}	207	328
M24	-197	-187	-407	{M17}	-60	{G10}	186	383
M25	-193	-159	-296	{M19}	-113	{M27}	101	307
M26	-168	-124	-406	{M27}	-78	{M3}	396	322
M27	-34	4	-409	{M22}	-110	{M27}	381	370

exception is that the difficulty, $\xi(\wp)$, is defined in terms of FAMILY-PRIZE-DT-($\kappa = 0$) (instead of PRIZE-DT-($\kappa = 0$)) since this is the defining strategy for this tournament, c.f., Equation (10.7).

Again, there is little distinction between the cluster location subclasses and overall deadline subclasses. As in the small-medium preliminary tournament, the {i,n,u,c,q} subclasses of the cluster value component were observably more difficult than the other subclasses and the player location subclasses which always place the players in the same cluster are invariably difficult.

10.1.3.4 Conclusions from the Medium Preliminary Tournament

The observations are similar between the results from the medium preliminary tournament, involving strategies applying a family-cluster structure, and the results from the small-medium preliminary tournament, involving strategies applying a single-family clustering. This implies that scaling up the DMS based strategies from single-family to unrestricted families is sufficient to be able to solve problems with double the number of prizes *with approximately the same degree of robustness.* This supports the notion that the family-cluster structure prunes the strategic or tactical game tree so that the important strategic decisions are made at a higher *frame* than the equivalent non-family engine. Additionally, FAMILY-PRIZE-DT remains tractable even for 40 prizes because of the family-no-return, cluster-no-return and the 'ALL' requirement which serve to massively prune the equivalent PRIZE-DT game tree. Even so, FAMILY-PRIZE-DT is not as robust as CLUSTER-DT, which indicates that CLUSTER-DT is able to make better strategic decisions in the medium-term, in the worst case.

In conclusion, the strategies to be considered for the medium final tournament (Section 10.2.3) are: **M5**, **M6**, **M7**, **M20**, **M21**, **M22**, and **M23**. As in the conclusions to the small-medium preliminary tournament, we will only consider further the {i,n,u,c,q} subclasses of the cluster value component, and subclasses of the the player location component which involve any of {h,i,w} or any pair from {a,b,c}.

10.1.4 Medium-Large Preliminary Tournament

The aim of the medium-large preliminary tournament is to compare the robustness of, and computational effort required by, the DMS 'CLUSTER-' strategies designed in Chapter 7 and the DMS 'GRID-' strategies designed in Chapter 8, and to compare the difficulty of contributing components of the problem instance classes. The focus is on problem instances which are not explicitly clustered and hence test the ability of the 'CLUSTER-' strategies to impose a family-cluster structure rather than map onto a natural prize structure. The defining strategy from Table 10.1 is CLUSTER-DT which is capable of solving reasonably large problems if there is a good balance between the number of families, number of clusters per family, and number of prizes per cluster.

10.1.4.1 Participating Strategies

The participants in the medium-large preliminary tournament are those listed in Table 10.11 plus strategies **G1**–**G12** from Table 10.2. Although there are 50 strategies competing in this tournament, the strategic engines (both cluster based and grid based) are invoked infrequently.

Table 10.10: Medium Preliminary Tournament: Results by Cluster Subclass

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
C:9ra------	0	{M27}	796	{M18}	1161	{M13}	0.175	0.029
C:9rr------	203	{M24}	960	{M24}	1306	{G7}	0.175	0.050
C:9hr------	0	{M23}	986	{M18}	1420	{M8}	0.173	0.096
C:9pd------	0	{M7}	789	{M26}	954	{M8}	0.169	0.052
C:9pg------	308	{M21}	1074	{M6}	1413	{G12}	0.167	0.043
C:9pl------	0	{M20}	925	{M17}	1383	{G11}	0.165	0.036
C:9cg------	0	{M16}	527	{M20}	653	{G8}	0.164	0.076
C:9ls------	0	{M20}	800	{M6}	1114	{M9}	0.164	0.044
C:9ps------	0	{M7}	672	{M23}	889	{G2}	0.164	0.046
C:9hs------	0	{M17}	848	{M25}	1303	{M10}	0.163	0.038
C:9rc------	335	{M23}	744	{M24}	931	{G11}	0.163	0.060
C:9cw------	0	{M5}	545	{M16}	1318	{M15}	0.162	0.037
C:9pk------	0	{M24}	913	{M25}	1229	{M13}	0.162	0.056
C:9rn------	0	{M20}	860	{M17}	1135	{M11}	0.162	0.065
C:9ld------	0	{M6}	587	{M7}	1485	{M13}	0.161	0.091
C:9pc------	147	{M5}	787	{M19}	1244	{M13}	0.160	0.087
C:9rq------	290	{M22}	371	{M24}	762	{M13}	0.159	0.028
C:9rs------	0	{M21}	905	{M20}	1209	{M8}	0.155	0.045
C:9lk------	291	{M7}	845	{M7}	1221	{G3}	0.153	0.003
C:9cl------	0	{M7}	1010	{M16}	1490	{G5}	0.150	0.064
C:9--n-----	0	{M23}	772	{M27}	1127	{M11}	0.404	0.066
C:9--i-----	0	{M22}	632	{M18}	998	{M10}	0.401	0.090
C:9--u-----	0	{M27}	717	{M27}	1046	{G11}	0.317	0.087
C:9--q-----	0	{M27}	869	{M27}	1415	{M10}	0.294	0.054
C:9--c-----	0	{M16}	854	{M25}	1137	{G7}	0.245	0.053
C:9--p-----	230	{M16}	744	{M19}	1057	{G11}	0.224	0.054
C:9--e-----	0	{M22}	820	{M21}	1395	{M13}	0.203	0.067
C:9--s-----	0	{M17}	957	{M24}	1308	{M8}	0.195	0.068
C:9--b-----	0	{M24}	928	{M5}	1240	{M9}	0.165	0.019
C:9---el---	0	{M19}	821	{M22}	1058	{M10}	0.282	0.065
C:9---et---	0	{M22}	342	{M5}	619	{M13}	0.259	0.075
C:9---rl---	0	{M26}	982	{M21}	1479	{M15}	0.245	0.045
C:9---rm---	293	{M24}	782	{M5}	1096	{G6}	0.240	0.059
C:9---em---	265	{M18}	705	{M18}	977	{M14}	0.237	0.007
C:9---rt---	0	{M18}	689	{M25}	878	{G11}	0.228	0.036
C:9-------r	0	{M27}	752	{M21}	1044	{G9}	0.168	0.038
C:9-------i	0	{M25}	648	{M6}	1118	{G11}	0.151	0.009

Table 10.10: Medium Preliminary Tournament: Results by Cluster Subclass (continued)

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
C:9-----bh-	0	{M20}	1251	{M19}	1475	{M11}	0.347	0.086
C:9-----dh-	0	{M23}	859	{M6}	1186	{M9}	0.345	0.090
C:9-----eh-	0	{M22}	891	{M24}	1313	{G8}	0.342	0.020
C:9-----ai-	0	{M6}	755	{M7}	917	{G12}	0.341	0.076
C:9-----aw-	0	{M5}	766	{M18}	1209	{G4}	0.341	0.076
C:9-----ab-	0	{M25}	786	{M18}	1029	{G4}	0.336	0.073
C:9-----ew-	0	{M7}	771	{M20}	1132	{G8}	0.334	0.016
C:9-----bc-	0	{M23}	620	{M26}	1010	{M9}	0.328	0.055
C:9-----bb-	0	{M20}	504	{M23}	1058	{G8}	0.328	0.072
C:9-----bw-	0	{M18}	876	{M22}	1385	{G12}	0.327	0.071
C:9-----cb-	254	{M7}	1048	{M17}	1275	{G4}	0.324	0.071
C:9-----di-	0	{M5}	958	{M16}	1169	{M9}	0.323	0.075
C:9-----aa-	240	{M25}	838	{M24}	1326	{M15}	0.322	0.089
C:9-----ca-	0	{M18}	578	{M25}	1029	{G7}	0.318	0.086
C:9-----cc-	285	{M19}	502	{M23}	735	{G8}	0.313	0.056
C:9-----ei-	0	{M22}	886	{M19}	1135	{M8}	0.312	0.018
C:9-----ah-	0	{M5}	1077	{M17}	1241	{M8}	0.305	0.080
C:9-----dw-	0	{M7}	1266	{M17}	1600	{G2}	0.296	0.088
C:9-----bi-	0	{M26}	926	{M21}	1224	{M15}	0.294	0.073
C:9-----ac-	0	{M27}	803	{M23}	1297	{M8}	0.290	0.057
C:9-----ba-	0	{M6}	572	{M7}	1142	{G7}	0.284	0.077
C:9-----fa-	0	{M5}	773	{M25}	1112	{G9}	0.244	0.024
C:9-----be-	0	{M27}	936	{M26}	1385	{M10}	0.241	0.017
C:9-----bf-	0	{M26}	818	{M25}	1069	{M13}	0.240	0.020
C:9-----af-	0	{M7}	953	{M21}	1408	{G7}	0.238	0.024
C:9-----fc-	0	{M22}	1105	{M23}	1386	{M9}	0.238	0.023
C:9-----ea-	0	{M20}	1094	{M5}	1601	{M13}	0.237	0.021
C:9-----de-	0	{M26}	1012	{M24}	1084	{G4}	0.237	0.052
C:9-----df-	0	{M6}	941	{M16}	1238	{M10}	0.236	0.062
C:9-----bd-	0	{M25}	617	{M20}	1267	{G9}	0.233	0.016
C:9-----cd-	0	{M27}	1624	{M24}	1655	{G12}	0.231	0.018
C:9-----db-	0	{M25}	1121	{M25}	1328	{G5}	0.229	0.015
C:9-----cf-	0	{M25}	845	{M19}	1314	{G12}	0.228	0.015
C:9-----ec-	305	{M24}	956	{M26}	1412	{G9}	0.223	0.019
C:9-----ad-	143	{M18}	746	{M6}	1054	{G10}	0.216	0.022
C:9-----fb-	318	{M18}	1209	{M7}	1442	{G6}	0.212	0.023
C:9-----dd-	0	{M16}	796	{M27}	1087	{G3}	0.212	0.055
C:9-----ef-	333	{M20}	669	{M19}	975	{G7}	0.212	0.018
C:9-----ae-	289	{M27}	641	{M6}	897	{G4}	0.211	0.015
C:9-----ed-	0	{M7}	654	{M26}	946	{M12}	0.205	0.023
C:9-----da-	0	{M6}	929	{M6}	1438	{G4}	0.200	0.019
C:9-----eb-	344	{M22}	1240	{M26}	1666	{G9}	0.199	0.019
C:9-----ee-	0	{M19}	462	{M21}	748	{G5}	0.197	0.025
C:9-----ff-	0	{M20}	1304	{M23}	1508	{G6}	0.193	0.065
C:9-----fe-	0	{M23}	1194	{M23}	1270	{G7}	0.193	0.056
C:9-----fd-	427	{M6}	667	{M27}	1123	{G3}	0.189	0.051
C:9-----dc-	0	{M6}	563	{M27}	893	{M8}	0.185	0.024
C:9-----ce-	0	{M19}	764	{M27}	1102	{M14}	0.179	0.019

Table 10.11: Medium-Large Tournament: Participating Strategies

Identifier	Strategy Name	Section	Type
ML1	CLUSTER-GUARANTEE//PRIZE-GUARANTEE	§7.3.3	**D**
ML2	CLUSTER-GUARANTEE//PRIZE-DT-(MINIMAX)	§7.3.3	**D**
ML3	CLUSTER-GUARANTEE//PRIZE-DT-($\kappa = 0$)	§7.3.3	**D**
ML4	CLUSTER-GUARANTEE//PRIZE-DT-(MAXIMIN)	§7.3.3	**D**
ML5	CLUSTER-PARANOID//PRIZE-GUARANTEE	§7.4	**M**
ML6	CLUSTER-PARANOID//PRIZE-DT-(MINIMAX)	§7.4	**M**
ML7	CLUSTER-PARANOID//PRIZE-DT-($\kappa = 0$)	§7.4	**M**
ML8	CLUSTER-PARANOID//PRIZE-DT-(MAXIMIN)	§7.4	**M**
ML9	CLUSTER-DT-(MINIMAX)//PRIZE-GUARANTEE	§7.5	**M**
ML10	CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MINIMAX)	§7.5	**M**
ML11	CLUSTER-DT-(MINIMAX)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
ML12	CLUSTER-DT-(MINIMAX)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
ML13	CLUSTER-DT-($\kappa = 0$)//PRIZE-GUARANTEE	§7.5	**M**
ML14	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MINIMAX)	§7.5	**M**
ML15	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
ML16	CLUSTER-DT-($\kappa = 0$)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
ML17	CLUSTER-DT-(MAXIMIN)//PRIZE-GUARANTEE	§7.5	**M**
ML18	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MINIMAX)	§7.5	**M**
ML19	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-($\kappa = 0$)	§7.5	**M**
ML20	CLUSTER-DT-(MAXIMIN)//PRIZE-DT-(MAXIMIN)	§7.5	**M**
ML21	GRID-DT//PRIZE-DT//STATIC	§8.3.2	**M**
ML22	GRID-DT//PRIZE-DT//1-DYNAMIC	§8.3.2	**M**
ML23	GRID-DT//PRIZE-DT//2-DYNAMIC	§8.3.2	**M**
ML24	GRID-DT//$\mathcal{ABA}$-PATH//STATIC	§8.3.2	**M**
ML25	GRID-DT//$\mathcal{ABA}$-PATH//1-DYNAMIC	§8.3.2	**M**
ML26	GRID-DT//$\mathcal{ABA}$-PATH//2-DYNAMIC	§8.3.2	**M**
ML27	GRID-DT//HARVEST-PATH//STATIC	§8.3.2	**D**
ML28	GRID-DT//HARVEST-PATH//1-DYNAMIC	§8.3.2	**D**
ML29	GRID-DT//HARVEST-PATH//2-DYNAMIC	§8.3.2	**D**
ML30	GRID-PATH//PRIZE-DT//STATIC	§8.3.3	**M**
ML31	GRID-PATH//PRIZE-DT//1-DYNAMIC	§8.3.3	**M**
ML32	GRID-PATH//PRIZE-DT//2-DYNAMIC	§8.3.3	**M**
ML33	GRID-PATH//$\mathcal{ABA}$-PATH//STATIC	§8.3.3	**M**
ML34	GRID-PATH//$\mathcal{ABA}$-PATH//1-DYNAMIC	§8.3.3	**M**
ML35	GRID-PATH//$\mathcal{ABA}$-PATH//2-DYNAMIC	§8.3.3	**M**
ML36	GRID-PATH//HARVEST-PATH//STATIC	§8.3.3	**D**
ML37	GRID-PATH//HARVEST-PATH//1-DYNAMIC	§8.3.3	**D**
ML38	GRID-PATH//HARVEST-PATH//2-DYNAMIC	§8.3.3	**D**

For each of the CLUSTER-GUARANTEE, CLUSTER-PARANOID, and CLUSTER-DT strategic engines, we employ PRIZE-GUARANTEE and PRIZE-DT as the underlying tactical engine. The ANY-ALL-PCTSP requirement for these 'CLUSTER-' strategies is set to 'PCTSP' for each cluster since each cluster is likely to be reasonably loose, i.e., the intra-cluster distances are approximately the same as the inter-cluster distances. The required value on cluster $[ci]$, $\eta_{[ci]}$, is set at $\frac{1}{2}v([ci])$, which ensures that players will dominate clusters as cluster-leads in the 'CLUSTER-' strategies.

The notation for DMS strategies involving a grid-engine at the GLOBAL-FRAME is to list the name of the grid-engine (either GRID-DT or GRID-PATH), the separator '//', the *minimal* tactical grid planning method (one of PRIZE-DT, $\mathcal{ABA}$-PATH, or HARVEST-PATH), the separator '//', and finally the grid-type (one of STATIC, 1-DYNAMIC, or 2-DYNAMIC). Recall from Section 8.4 that the Manhattan number of grid-cells between the players' current grid-cells, m, determines which tactical planning method is applied at the *prize-frame*. The following matrix defines the condition under which each tactical planning method is invoked.

Minimal Tactical Planning Method	Tactical Planning Method Invoked		
	PRIZE-DT	$\mathcal{ABA}$-PATH	HARVEST-PATH
PRIZE-DT	$m = 0$	$m = 1$	$m \geq 2$
$\mathcal{ABA}$-PATH	*never*	$0 \leq m \leq 1$	$m \geq 2$
HARVEST-PATH	*never*	*never*	$m \geq 0$

Each 'GRID-' strategy is constrained to a 4×4 grid structure. Similarly, each 'CLUSTER-' strategy is standardised to a common family-cluster structure consisting of four families of four clusters each. The clustering method employed is to use a 2×2 static grid to define an initial division of the prizes into (at most) four non-empty families. Within each family, the prizes are further subdivided into (at most) four non-empty clusters using a 2×2 grid on that family. Membership of a cluster to a family is now fixed. Finally, prize-value-weighted cluster improvement is applied.

10.1.4.2 Problem Instances

We are interested in problem which are not explicitly structured but have some concentration, or dearth, of prize value which may be used by a cluster based strategy to divide the prizes. All problem instances considered here are from the D-class (see Section 9.1.3) for which the defining components are: number of hot, cold and black features; feature scale and strength subclass; and player location subclass. Effectively, the D-class problem instances are able to be clustered since the hot and cold features will not contain too many prizes.

The number of prizes is constrained to approximately 60 so that there will not be too many prizes per cluster, but, on the other hand, there will be sufficiently many prizes per grid-cell that the intra-grid-cell HARVEST-PATH and DIRECT-PATH may be different. The number of hot, cold and black features range from zero to two to construct the subclasses of the number of features component. Once again, only 20 problem instances per subclass are possible, except that 100 problem instances were generated for each of the two player location subclasses.

10.1.4.3 Results and Analysis

The results from the medium-large preliminary tournament follow the same format as for the small preliminary tournament (see Section 10.1.1).

▼ Robustness and Computational Effort of Strategies

Figure 10.4 compares the overall robustness and computational effort by strategy, following the same specification as Figure 10.1. Figure 10.4(a) shows the *overall* 'MIN-MIN' *robustness* of each strategy, and Figure 10.4(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$). The most computationally expensive strategies are the CLUSTER-DT strategies. In comparison, the 'GRID-' strategies are relatively inexpensive. However, the CLUSTER-DT-($\kappa = 0$) variations and the GRID-PATH variations are both relatively robust.

▼ Summary of Results by Strategy

Table 10.12 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.3. An important observation is that the 'CLUSTER-' strategies are absent from the list of nemesis strategies, indicating that the 'GRID-' strategies are generally more difficult to play against. A surprise result is that CLUSTER-GUARANTEE contributes two robust performances (**ML2** and **ML3**) but **ML1** and **ML4** perform very poorly and, hence, we do not consider CLUSTER-GUARANTEE as an overall robust strategy paradigm.

▼ Summary of Results by Problem Subclass

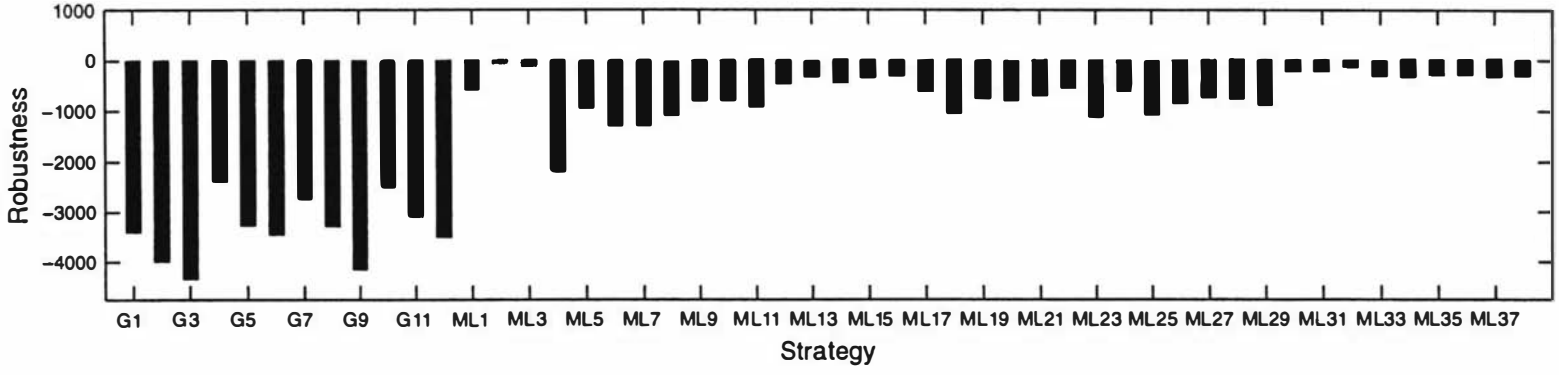
Table 10.13 presents a summary of the best performance by a strategy on each problem subclass, and the difficulty of each problem subclass, following the same specification as Table 10.4. Recall from Section 10.0.3.3 that $\xi(\wp)$ is defined in terms of the dynamic predictors GRID-PATH//HARVEST-PATH//STATIC and HORIZON-OP for the medium-large and large tournaments.

The MEAN difficulty of the number of prize value density features subclass indicates that problem instances with more hot features are generally more difficult and, with a fixed number of hot features, problem instances with more cold or black features are also more difficult. However, the feature scale and strength subclasses and the player location subclasses offer no noticeable trend. Additionally, the GRID-PATH strategies are usually evident as the best MIN-MIN strategy, which implies that these GRID-PATH strategies are able to play well against a range of strategies over a range of problem instances.

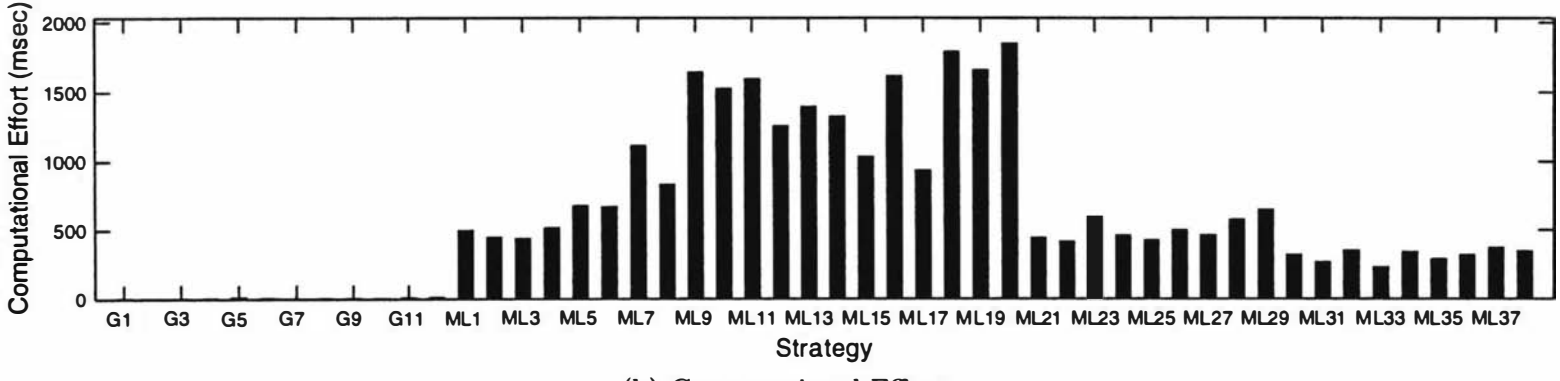
10.1.4.4 Conclusions from the Medium-Large Preliminary Tournament

The dominance of GRID-PATH over GRID-DT and CLUSTER-DT implies that efficient medium-term planning is more important to larger problems than medium-term competition.

In conclusion, the strategies that will be considered for the medium-large final tournament (Section 10.2.4) are: **ML13**, **ML14**, **ML15**, **ML16**, **ML30**, **ML33**, and **ML36**. For the problem instances, we will concentrate on the the number of features, and in particular, we will increase



(a) Strategy Robustness



(b) Computational Effort

Figure 10.4: Medium-Large Preliminary Tournament: Results by Strategy

Table 10.12: Medium-Large Preliminary Tournament: Results by Strategy

Strategy	Worst Subclass		Nemesis				Overall	
$a \in \mathbb{S}$	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
G1	-838	-767	-3385	{G3}	-344	{ML33}	420	366
G2	-622	-640	-3988	{ML30}	-1043	{ML27}	-993	416
G3	-810	-730	-4325	{ML29}	-940	{ML37}	140	321
G4	-734	-719	-2393	{G1}	-447	{ML28}	-235	315
G5	-350	-347	-3255	{G1}	-302	{G10}	415	371
G6	-601	-550	-3433	{G9}	325	{ML36}	363	315
G7	-384	-390	-2741	{ML31}	-652	{ML36}	-471	307
G8	-763	-760	-3266	{ML23}	-1452	{ML27}	-437	319
G9	-873	-752	-4144	{ML24}	20	{ML28}	384	332
G10	-285	-293	-2508	{ML27}	-1472	{ML33}	-1467	427
G11	-1023	-1013	-3094	{G5}	-935	{G4}	-734	350
G12	-84	-82	-3486	{ML21}	-1370	{G10}	132	340
ML1	68	67	-577	{ML27}	-8	{ML38}	57	347
ML2	52	53	-52	{ML30}	81	{ML38}	81	344
ML3	310	308	-114	{ML31}	-36	{ML36}	-34	367
ML4	-376	-467	-2197	{ML26}	-1180	{G6}	-1173	471
ML5	-164	-206	-933	{ML24}	-651	{ML38}	-585	307
ML6	-1012	-982	-1318	{ML33}	-668	{ML24}	-537	324
ML7	-614	-611	-1279	{ML27}	-544	{ML33}	-319	350
ML8	-265	-294	-1062	{ML31}	-612	{ML27}	-540	326

Table 10.12: Medium-Large Preliminary Tournament: Results by Strategy (continued)

Strategy	Worst Subclass		Nemesis				Overall	
$a \in \mathbb{S}$	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
ML9	-399	-379	-808	{ML34}	-368	{G6}	-224	333
ML10	-304	-291	-795	{ML27}	-176	{ML38}	-176	358
ML11	-289	-240	-945	{ML33}	-119	{G4}	103	333
ML12	-133	-131	-465	{ML24}	-181	{ML34}	404	358
ML13	-164	-132	-317	{ML33}	168	{ML28}	171	347
ML14	-243	-159	-459	{ML31}	426	{ML31}	465	356
ML15	-111	-81	-343	{ML29}	112	{ML33}	122	354
ML16	-118	-111	-320	{ML24}	62	{ML34}	81	344
ML17	-525	-516	-613	{ML27}	31	{ML38}	33	337
ML18	-447	-420	-1043	{ML23}	-182	{ML38}	-137	358
ML19	-193	-152	-742	{ML26}	150	{ML34}	150	339
ML20	-601	-515	-780	{ML31}	63	{ML38}	91	340
ML21	-201	-202	-700	{ML31}	-217	{G10}	-217	306
ML22	-235	-200	-548	{ML22}	63	{ML34}	194	362
ML23	-544	-546	-1118	{ML33}	-623	{ML33}	-560	321
ML24	-64	-60	-607	{ML32}	-59	{ML27}	211	321
ML25	-490	-484	-1086	{ML35}	-444	{ML37}	-444	309
ML26	-148	-151	-849	{ML21}	-278	{G9}	-229	316
ML27	-469	-435	-733	{ML38}	-107	{ML38}	-36	377
ML28	-204	-144	-785	{ML33}	-308	{ML28}	228	370
ML29	-233	-211	-880	{ML26}	-45	{ML31}	-45	345
ML30	59	75	-219	{ML30}	167	{ML33}	245	312
ML31	-33	-31	-225	{ML36}	-4	{ML26}	409	366
ML32	86	88	-132	{ML24}	141	{ML26}	153	326
ML33	90	97	-305	{ML32}	-95	{G4}	162	366
ML34	-59	-64	-339	{ML33}	-121	{ML36}	-110	314
ML35	-205	-165	-291	{ML31}	-221	{ML36}	250	358
ML36	11	6	-299	{ML27}	-118	{G9}	-50	369
ML37	111	109	-351	{ML27}	83	{ML38}	83	334
ML38	-109	-68	-292	{ML37}	343	{ML38}	435	365

Table 10.13: Medium-Large Preliminary Tournament: Results by Cluster Subclass

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
D:211---	198	{ML35}	424	{ML29}	398	{ML7}	0.263	0.105
D:212---	0	{ML37}	84	{G6}	200	{ML15}	0.262	0.117
D:221---	0	{G1}	437	{ML33}	662	{ML1}	0.250	0.114
D:222---	0	{ML35}	91	{ML33}	200	{ML18}	0.237	0.108
D:220---	212	{ML33}	512	{G8}	1057	{ML5}	0.222	0.107
D:210---	258	{ML35}	456	{G5}	816	{G11}	0.211	0.100
D:202---	199	{ML34}	419	{G12}	713	{ML12}	0.201	0.091
D:111---	212	{ML30}	394	{ML29}	502	{G2}	0.193	0.012
D:121---	0	{ML35}	682	{ML22}	813	{G4}	0.190	0.017
D:201---	147	{ML33}	655	{ML35}	898	{ML32}	0.188	0.089
D:122---	209	{ML35}	226	{G7}	409	{ML6}	0.185	0.017
D:120---	250	{G10}	1371	{ML32}	1551	{ML14}	0.183	0.019
D:200---	0	{ML36}	156	{ML30}	854	{ML38}	0.177	0.085
D:100---	236	{ML36}	482	{ML27}	725	{ML30}	0.173	0.050
D:112---	204	{ML32}	624	{G4}	1257	{G8}	0.168	0.020
D:011---	0	{G1}	51	{ML33}	200	{G10}	0.168	0.106
D:110---	193	{ML30}	319	{ML21}	393	{ML37}	0.165	0.014
D:101---	0	{ML32}	507	{ML36}	781	{ML29}	0.160	0.054
D:021---	262	{ML37}	832	{ML26}	1104	{ML33}	0.155	0.082
D:020---	0	{ML36}	498	{ML35}	976	{ML16}	0.154	0.089
D:012---	190	{ML37}	783	{G8}	1043	{ML36}	0.152	0.092
D:001---	0	{G1}	639	{ML26}	1100	{G10}	0.144	0.075
D:102---	229	{ML36}	377	{ML21}	429	{ML9}	0.140	0.060
D:022---	0	{G2}	223	{ML24}	366	{ML37}	0.138	0.085
D:010---	176	{ML30}	396	{G5}	637	{G9}	0.131	0.104
D:000---	0	{G10}	361	{ML24}	542	{ML10}	0.122	0.077
D:002---	141	{ML32}	262	{ML21}	341	{ML22}	0.108	0.093
D:---ae-	185	{G10}	392	{ML36}	872	{G11}	0.178	0.091
D:---mE-	219	{ML38}	339	{G8}	419	{G7}	0.174	0.035
D:---aE-	213	{G10}	580	{ML30}	734	{ML25}	0.159	0.095
D:---am-	233	{ML33}	645	{ML33}	940	{ML1}	0.151	0.060
D:---Am-	205	{G10}	592	{ML28}	1009	{ML13}	0.150	0.091
D:---AE-	0	{ML30}	234	{ML28}	631	{ML34}	0.148	0.077
D:---me-	211	{G2}	517	{G10}	787	{G9}	0.144	0.057
D:---Ae-	0	{G1}	143	{ML24}	451	{G12}	0.132	0.059
D:---mm-	160	{ML34}	596	{ML26}	706	{G5}	0.131	0.058
D:-----m	0	{ML36}	1724	{ML21}	1456	{ML31}	0.142	0.096
D:-----c	214	{ML36}	793	{G4}	1008	{G9}	0.139	0.078

Table 10.14: Large Preliminary Tournament: Participating Strategies

Identifier	Strategy Name	Section	Type
L1	GRID-DT//$\mathcal{ABA}$-PATH//STATIC	§8.3.2	**M**
L2	GRID-DT//$\mathcal{ABA}$-PATH//1-DYNAMIC	§8.3.2	**M**
L3	GRID-DT//$\mathcal{ABA}$-PATH//2-DYNAMIC	§8.3.2	**M**
L4	GRID-DT//HARVEST-PATH//STATIC	§8.3.2	**D**
L5	GRID-DT//HARVEST-PATH//1-DYNAMIC	§8.3.2	**D**
L6	GRID-DT//HARVEST-PATH//2-DYNAMIC	§8.3.2	**D**
L7	GRID-PATH//$\mathcal{ABA}$-PATH//STATIC	§8.3.3	**M**
L8	GRID-PATH//$\mathcal{ABA}$-PATH//1-DYNAMIC	§8.3.3	**M**
L9	GRID-PATH//$\mathcal{ABA}$-PATH//2-DYNAMIC	§8.3.3	**M**
L10	GRID-PATH//HARVEST-PATH//STATIC	§8.3.3	**D**
L11	GRID-PATH//HARVEST-PATH//1-DYNAMIC	§8.3.3	**D**
L12	GRID-PATH//HARVEST-PATH//2-DYNAMIC	§8.3.3	**D**

the number of hot features considered, so that there are either two or three hot features, with the same ranges of cold and black features.

10.1.5 Large Preliminary Tournament

The aim of the large preliminary tournament is to compare the robustness of, and computational effort required by, the DMS 'GRID-' strategies designed in Chapter 8, and to compare the difficulty of contributing components of the problem instance classes. The focus is on problem instances with a large number of prizes but with some variation in prize value density across the prize region. The central question is whether the medium-term planning of the 'GRID-' strategies makes any notable difference in performance over the myopic basic strategies. The defining strategy from Table 10.1 is GRID-DT//$\mathcal{ABA}$-PATH.

10.1.5.1 Participating Strategies

The participants in the large tournament are those listed in Table 10.14 plus strategies **G1–G12** from Table 10.2. Note that PRIZE-DT is not included as a tactical planning method because it is too computationally expensive on large problems. This leaves 24 strategies. However, the expected duration or each simulation battle is longer than in any other tournament. The parameter of note is the grid-size. which is standardised to 4×4 .

10.1.5.2 Problem Instances

We are interested in problems with explicit variation of prize value density across the prize region, so all problem instances considered here are from the D-class (see Section 9.1.3) for which the defining components are: number of hot, cold and black features; feature scale and strength subclass; and player location subclass.

The number of prizes is constrained to approximately 150 prizes, which is sufficient to test the medium-term planning ability of the 'GRID-' strategies, but not too many so that the expected duration of each simulation battle becomes prohibitive. We consider between zero and two of each prize value density feature as the subclasses of the number of features component. Again, only 20 of each subclass were able to be constructed due to the large computational effort required to execute the tournament.

10.1.5.3 Results and Analysis

The results from the large preliminary tournament follow the same format as for the medium-large preliminary tournament (see Section 10.1.4).

▼ Robustness and Computational Effort of Strategies

Figure 10.5 compares the overall robustness and computational effort by strategy, following the same specification as Figure 10.1. Figure 10.5(a) shows the *overall* 'MIN-MIN' *robustness* of each strategy and Figure 10.5(b) shows the *computational effort*, mean($\frac{\tau}{\overline{\tau}}$). The GRID-PATH strategies are more robust and slightly less computationally expensive than the GRID-DT. The most significant result is that the basic strategies are approximately as poorly robust, which indicates that the 'GRID-' paradigms provide important strategic guidance with respect to harvesting in significant subregions of the prize region. This result supports the original design of the harvesting based strategies for large problems discussed in Chapter 8.

▼ Summary of Results by Strategy

Table 10.15 presents a more extensive summary of the performance of each strategy, $\mathsf{a} \in \mathbb{S}$, following the same specification as Table 10.3. Again, the results demonstrate that overall MEAN-MEAN effectiveness is not necessarily related to MIN-MIN robustness.

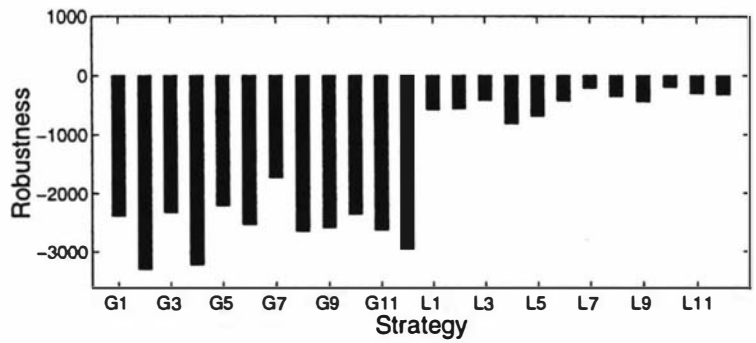
▼ Summary of Results by Problem Subclass

Table 10.16 presents a summary of the best performance by a strategy on each problem subclass, and the difficulty of each problem subclass, following the same specification as Table 10.4.

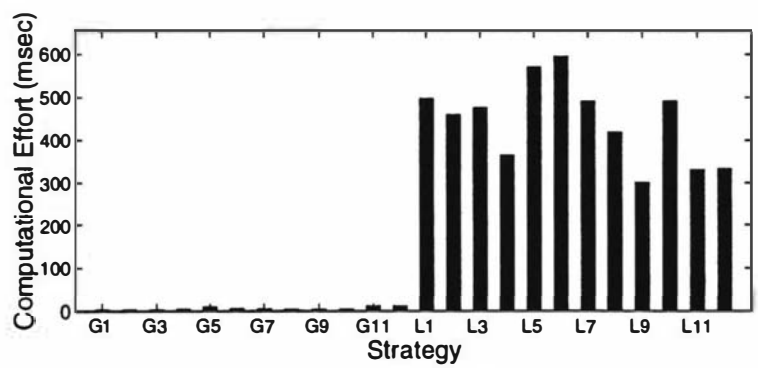
Similar to the medium-large tournament, problem instances with a greater number of hot features are generally more difficult than problem instances with few hot features. Additionally, the GRID-PATH strategies tend to dominate as the best MIN-MEAN strategy, implying an ability to play well on all problem types and against all opponents.

10.1.5.4 Conclusions from the Large Preliminary Tournament

In conclusion, the strategies to be considered for the large final tournament (Section 10.2.5) are **L7–L12**. For the problem instances, as in the conclusions to the medium-large preliminary tournament, we will concentrate on the the number of features, and in particular, we will increase the number of hot features considered, so that there are either two or three hot features, with the same ranges of cold and black features.



(a) Strategy Robustness



(b) Computational Effort

Figure 10.5: Large Preliminary Tournament: Results by Strategy

Table 10.15: Large Preliminary Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
G1	-710	-596	-2382	{L5}	15	{G12}	237	309
G2	-299	-385	-3303	{L7}	-1047	{L6}	-924	327
G3	-1363	-1307	-2326	{L1}	-189	{G12}	-138	328
G4	-58	-59	-3231	{L7}	-157	{L5}	-66	352
G5	-763	-712	-2212	{L4}	153	{L6}	492	399
G6	-1132	-964	-2536	{L6}	-581	{L6}	137	332
G7	-205	-221	-1733	{L10}	-969	{G12}	-673	339
G8	-456	-574	-2650	{L11}	-1393	{L3}	-1346	497
G9	-449	-364	-2591	{L12}	235	{L4}	305	360
G10	-1119	-1091	-2366	{L5}	-59	{L2}	-57	330
G11	-603	-521	-2629	{L3}	465	{L4}	476	357
G12	-6	-5	-2947	{L4}	-741	{L4}	25	369
L1	-88	-60	-583	{L7}	121	{L1}	369	348
L2	-163	-156	-564	{L7}	-26	{L1}	75	337
L3	63	75	-425	{L12}	-63	{L5}	185	360
L4	-81	-46	-819	{L12}	-449	{L1}	387	354
L5	-322	-313	-696	{L8}	-343	{L4}	-204	324
L6	66	66	-434	{L5}	-25	{G5}	451	378
L7	153	184	-216	{L8}	-42	{L4}	438	364
L8	-133	-106	-354	{L8}	-57	{G5}	81	347
L9	-38	-56	-447	{L6}	-305	{L1}	-195	355
L10	76	87	-195	{L4}	98	{L2}	327	373
L11	-140	-67	-310	{L7}	116	{L4}	397	320
L12	45	103	-327	{L12}	427	{L5}	460	335

Table 10.16: Large Preliminary Tournament: Results by Density Subclass

Subclass ($\mathbb{P}_i$)	Best Strategy						Difficulty	
	MIN-MIN		MIN-MEAN		MEAN-MEAN		MEAN	STD
D:202---	0	{L8}	333	{L12}	503	{L9}	0.232	0.064
D:211---	239	{L11}	523	{L8}	1097	{L5}	0.224	0.073
D:200---	198	{G2}	620	{L7}	1188	{G9}	0.223	0.073
D:110---	237	{L7}	567	{L10}	1019	{L8}	0.200	0.042
D:012---	210	{L8}	491	{L9}	1356	{L5}	0.199	0.068
D:222---	248	{L11}	855	{L12}	1100	{L1}	0.197	0.072
D:201---	215	{L7}	678	{L10}	1102	{L10}	0.193	0.003
D:122---	216	{L11}	632	{L11}	932	{G4}	0.190	0.007
D:212---	244	{L12}	615	{L12}	1213	{G11}	0.185	0.035
D:022---	245	{G3}	562	{L11}	962	{L1}	0.179	0.087
D:121---	187	{G4}	309	{L8}	525	{G7}	0.170	0.030
D:221---	0	{G2}	594	{L12}	1327	{G10}	0.169	0.076
D:220---	212	{L8}	550	{L12}	972	{G3}	0.168	0.037
D:111---	0	{L8}	399	{L9}	860	{G2}	0.159	0.066
D:101---	134	{L9}	291	{L8}	364	{L10}	0.157	0.033
D:210---	0	{L9}	542	{L7}	737	{G6}	0.152	0.063
D:100---	0	{G3}	390	{L11}	1167	{G5}	0.151	0.082
D:021---	210	{G2}	366	{L12}	757	{L1}	0.137	0.098
D:112---	174	{L8}	245	{L8}	374	{G4}	0.130	0.029
D:120---	205	{L9}	873	{L9}	1109	{G8}	0.120	0.034
D:000---	0	{G4}	332	{L11}	549	{L7}	0.107	0.010
D:102---	0	{L9}	216	{L7}	243	{G9}	0.106	0.048
D:010---	244	{L10}	201	{L7}	990	{L1}	0.079	0.052
D:020---	255	{L9}	483	{L10}	792	{G4}	0.072	0.055
D:001---	218	{G2}	1073	{L11}	1550	{L2}	0.064	0.063
D:002---	0	{G2}	538	{L11}	740	{G7}	0.058	0.044
D:011---	253	{L11}	534	{L11}	567	{L10}	0.011	0.066
D:---mE-	0	{G4}	529	{L7}	712	{L11}	0.174	0.063
D:---am-	282	{G2}	643	{L10}	746	{L12}	0.165	0.053
D:---mm-	258	{G3}	690	{L11}	1240	{G11}	0.156	0.025
D:---Am-	180	{L7}	262	{L10}	861	{L12}	0.150	0.042
D:---me-	0	{L7}	549	{L9}	1170	{L4}	0.148	0.077
D:---aE-	0	{L11}	539	{L9}	1034	{L1}	0.139	0.026
D:---Ae-	187	{L7}	312	{L9}	481	{L4}	0.129	0.049
D:---ae-	0	{L7}	789	{L12}	1146	{L11}	0.122	0.064
D:---AE-	158	{L12}	537	{L8}	948	{L1}	0.111	0.052
D:-----c	0	{G2}	668	{L12}	772	{L9}	0.170	0.055
D:-----m	0	{L12}	37	{L8}	490	{L8}	0.142	0.026

10.1.6 Conclusions from Preliminary Computational Tournaments

We have frequently observed that those strategies which perform well with respect to effectiveness (overall MEAN-MEAN) are generally different from those strategies that are most robust. This justifies the need, expressed in the overview of Part III, for evaluating effectiveness against a benchmark set of robust strategies and challenging problem instances. We have also consistently observed that the prize/cluster/feature value distribution and player locations are the most important factors related to problem instance difficulty, and, surprisingly, that prize/cluster/feature location appear not to be.

It has also been realised that the computational effort required to execute these computational tournaments has been very large in comparison with traditional computational comparison of VRSP heuristics. This has necessarily restricted the number of problem instances generated and the range of strategies and strategy parameters tested. However, this initial computational experience indicates roughly which problem subclasses are the most difficult and which strategies are the most robust.

To conclude, the preliminary computational tournaments have achieved a broad comparison of strategies and problem subclasses. This should be used to justify and guide future computational experiments with more specific aims. We have assembled the strategies and problem instance subclasses to use in the final computational tournaments which follow. The focus now shifts from *robustness* of a strategy to evaluating its average *effectiveness* with respect to a benchmark set of strategies and problem instances.

10.2 Final Computational Tournaments

The experimental aim of the **Final Computational Tournaments** is to evaluate and compare the effectiveness of strategy paradigms against a benchmark set of robust strategies and bad-case problem instances. This aim is addressed using the experimental design and performance measures outlined in Section 10.0. The final computational tournaments constitute Step IV of our four step approach described in the overview of Part III.

The following five final tournaments (small, small-medium, medium, medium-large, and large) take the recommended strategies and problem instance classes from the corresponding five preliminary tournaments. However, the problem instances for the small, small-medium, and medium final tournaments are further manipulated into bad-case problems (see Section 9.4) in accordance with Step III of our four step approach. For the medium-large and large final tournaments, the problem instances are drawn from the most difficult subclasses as recommended by the corresponding preliminary tournaments.

10.2.1 Small Final Tournament

The aim of the small final tournament is to compare the effectiveness of, and computational effort required by, a benchmark set of the most robust strategies from the small preliminary tournament (Section 10.1.1) against a set of challenging, bad-case problem instances.

10.2.1.1 Participating Strategies

The participants in the small final tournament are those recommended in the conclusion to the small preliminary tournament as the most robust: **S8**, **S11**, **S12**, **S13**, **S14**, **S19**, **S20**, and **S21** from Table 10.2.

10.2.1.2 Problem Instances

The set of 7-prize problem instances are initially generated from the subclasses of P-class recommended in the conclusion to the small preliminary tournament as the most difficult: the {i,n,u,c,q} subclasses of the prize value component and the {mi,li,bi,ci} subclasses of the player location component. These problem instances are then "improved" with respect to difficulty, i.e., made more difficult, by the knife-edge search of Section 9.4.2. The two static predictors used in the objective function are PRIZE-GUARANTEE and PRIZE-DT-($\kappa = 0$). One hundred such bad-case problem instances were generated for each of the nine problem instance subclasses.

10.2.1.3 Results and Analysis

The results from the small final tournament follow basically the same format as for the small preliminary tournament (see Section 10.1.1), except that the effectiveness of a strategy, as defined by Equation (10.5), replaces the robustness of a strategy as our central measure of strategy performance.

▼ Effectiveness and Computational Effort of Strategies

Figure 10.6 compares the overall effectiveness and computational effort by strategy.

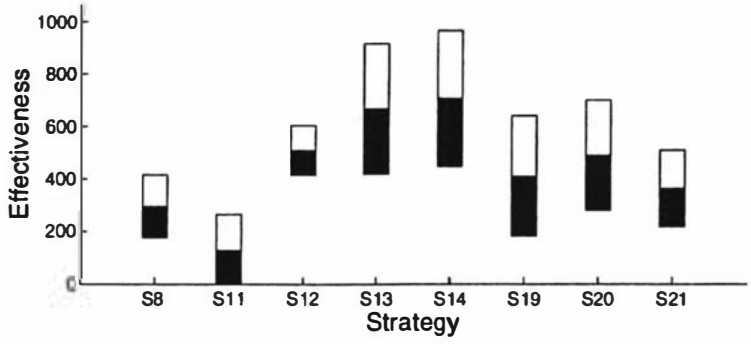
Firstly, Figure 10.6(a) shows the *overall* 'MEAN-MEAN' *effectiveness* of each strategy, $a \in \mathbb{S}$, as $\text{mean}(\varkappa|a) \pm \text{std}(\varkappa|a)$, given by Equations (10.10)–(10.11). The middle of the bar is the mean and the top and bottom of the bar are one standard deviation either side of the mean.

$$\text{effectiveness}(a) = \text{mean}(\varkappa|a) = \frac{1}{|\mathbb{S}|}\sum_{b\in\mathbb{S}}\frac{1}{|\mathbb{P}|}\sum_{\mathbb{P}_i}\left[\sum_{p\in\mathbb{P}_i}\varkappa\right] \tag{10.10}$$

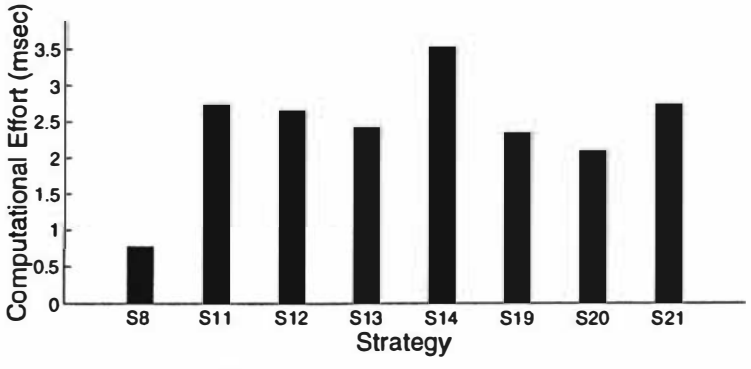
$$\text{std}(\varkappa|a) = \sqrt{\frac{1}{|\mathbb{S}||\mathbb{P}|-1}\left(\sum_{b\in\mathbb{S}}\sum_{\mathbb{P}_i}\left[\sum_{p\in\mathbb{P}_i}\varkappa^2\right] - \frac{1}{|\mathbb{S}||\mathbb{P}|}\left(\sum_{b\in\mathbb{S}}\sum_{\mathbb{P}_i}\left[\sum_{p\in\mathbb{P}_i}\varkappa\right]\right)^2\right)} \tag{10.11}$$

Secondly, Figure 10.6(b) shows the *computational effort*, $\frac{\tau}{\Xi}$, of each strategy $a \in \mathbb{S}$ as $\text{mean}(\frac{\tau}{\Xi})$, according to Equation (10.9).

PRIZE-PARANOID (**S8**) is computationally the least expensive, with the other strategies requiring approximately the same, greater, computational effort. The most effective strategy is PRIZE-DT and the least effective strategy is ORIGINAL-DT.



(a) Strategy Effectiveness



(b) Computational Effort

Figure 10.6: Small Final Tournament: Results by Strategy

▼ Summary of Results by Strategy

Table 10.17 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.3. It is noteworthy that, for this tournament, the nemesis of all the strategies is SINGLE-FAMILY-PRIZE-DT rather than PRIZE-DT. However, PRIZE-DT performed well with respect to both robustness and nemesis MIN-MEAN.

10.2.2 Small-Medium Final Tournament

The aim of the small-medium final tournament is to compare the effectiveness of, and computational effort required by, a benchmark set of the most robust strategies from the small-medium preliminary tournament (Section 10.1.2) against a set of challenging, bad-case problem instances.

10.2.2.1 Participating Strategies

The participants in the small-medium final tournament are those recommended in the conclusion to the small-medium preliminary tournament as being the most robust: **SM5**, **SM6**, **SM7**, **SM20**, **SM21**, **SM22**, and **SM23** from Table 10.5.

Table 10.17: Small Final Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
S8	-214	-169	-401	{S20}	195	{S8}	296	120
S11	66	66	-410	{S20}	-62	{S21}	128	137
S12	-106	-78	-271	{S20}	62	{S19}	509	94
S13	19	42	-83	{S20}	177	{S14}	668	249
S14	8	61	-210	{S20}	428	{S14}	707	259
S19	-183	-134	-374	{S20}	-163	{S21}	411	229
S20	13	46	-303	{S20}	179	{S19}	489	211
S21	-88	-54	-437	{S21}	-99	{S12}	363	146

10.2.2.2 Problem Instances

The set of 20-prize problem instances are initially generated from the subclasses of C-class recommended in the conclusion to the small-medium preliminary tournament as the most difficult: the {i,n,u,c,q} subclasses of the cluster value component, and subclasses of the the player location component which involve any of {h,i,w} or any pair from {a,b,c}. These problem instances are made more difficult by the knife-edge search of Section 9.4.2. The two static predictors used in the objective function are SINGLE-FAMILY-PRIZE-GUARANTEE and SINGLE-FAMILY-PRIZE-DT-($\kappa = 0$), with a common, four-cluster, family-cluster structure determined using the prize-value-weighted-Ward method with improvement and ANY–ALL–PCTSP requirement set to 'PCTSP' with $\eta_{[ci]} \leftarrow \frac{1}{2}v([ci])$. One hundred such bad-case problem instances were generated for each of the problem instance subclasses.

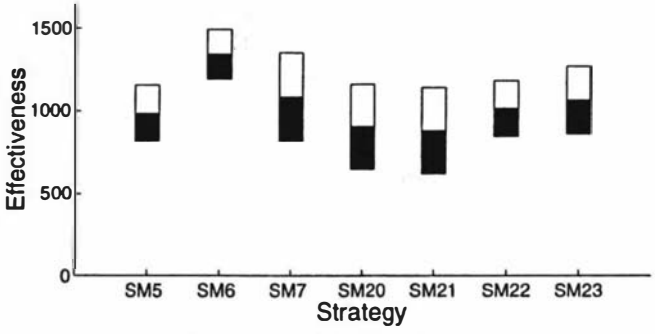
10.2.2.3 Results and Analysis

The results from the small-medium final tournament follow the same format as for the small final tournament (see Section 10.2.1).

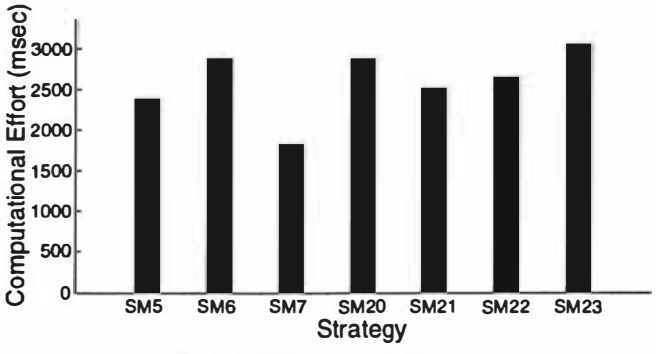
▼ Effectiveness and Computational Effort of Strategies

Figure 10.7 compares the overall effectiveness and computational effort by strategy, following the same specification as Figure 10.6. Figure 10.7(a) shows the *overall* 'MEAN-MEAN' *effectiveness* of each strategy, $a \in \mathbb{S}$, as mean($\varkappa|a$) $\pm$ std($\varkappa|a$), according to Equations (10.10)–(10.11). Figure 10.7(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$).

The SINGLE-FAMILY-PRIZE-DT strategies are generally more effective than than the (SF)-CLUSTER-DT strategies and, in particular, the SINGLE-FAMILY-PRIZE-DT is most effective with $\kappa = 0$. All the strategies required approximately the same amount of computational effort.



(a) Strategy Effectiveness



(b) Computational Effort

Figure 10.7: Small-Medium Final Tournament: Results by Strategy

▼ Summary of Results by Strategy

Table 10.18 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.17. Notably, the MIN-MIN nemesis is usually one of the SINGLE-FAMILY-PRIZE-DT strategies.

10.2.3 Medium Final Tournament

The aim of the medium final tournament is to compare the effectiveness of, and computational effort required by, a benchmark set of the most robust strategies from the medium preliminary tournament (Section 10.1.3) against a set of challenging, bad-case problem instances.

10.2.3.1 Participating Strategies

The participants in the medium final tournament are those recommended in the conclusion to the medium preliminary tournament as the most robust: **M5**, **M6**, **M7**, **M20**, **M21**, **M22**, and **M23** from Table 10.8.

Table 10.18: Small-Medium Final Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
SM5	80	107	-332	{SM5}	-180	{SM23}	985	169
SM6	-199	-33	-300	{SM21}	163	{SM7}	1342	151
SM7	-205	-98	-455	{SM5}	-34	{SM23}	1084	266
SM20	-256	-127	-365	{SM6}	66	{SM23}	905	257
SM21	-31	-4	-452	{SM5}	-295	{SM21}	880	261
SM22	22	150	-394	{SM5}	-113	{SM20}	1013	168
SM23	-274	-101	-492	{SM6}	-9	{SM22}	1065	204

10.2.3.2 Problem Instances

The set of 40-prize problem instances are initially generated from the subclasses of C-class recommended in the conclusion to the medium preliminary tournament as the most difficult: the {i,n,u,c,q} subclasses of the cluster value component, and subclasses of the the player location component which involve any of {h,i,w} or any pair from {a,b,c}. These problem instances are then made more difficult by the knife-edge search of Section 9.4.2. The two static predictors used in the objective function are FAMILY-PRIZE-GUARANTEE and FAMILY-PRIZE-DT-($\kappa = 0$), with a common, three-family, nine-cluster, family-cluster structure determined as described in Section 10.1.3.1 and ANY–ALL–PCTSP requirement set to 'ALL'. Fifty such bad-case problem instances were generated for each of the nine problem instance subclasses.

10.2.3.3 Results and Analysis

The results from the medium final tournament follow the same format as for the small final tournament (see Section 10.2.1).

▼ Effectiveness and Computational Effort of Strategies

Figure 10.8 compares the overall effectiveness and computational effort by strategy, following the same specification as Figure 10.6. Figure 10.8(a) shows the *overall* 'MEAN-MEAN' *effectiveness* of each strategy, and Figure 10.8(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$).

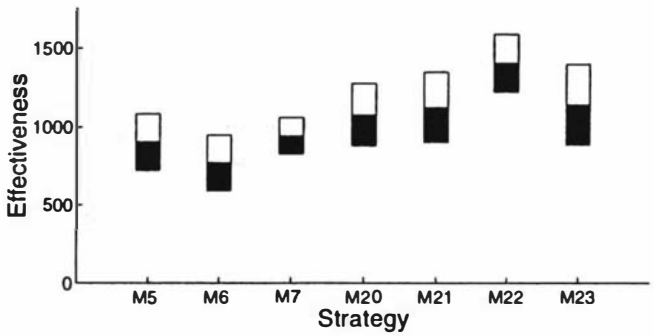
We observe that the CLUSTER-DT strategies are more effective than the FAMILY-PRIZE-DT strategies, and that the best of the CLUSTER-DT strategies corresponds to the tactical engine PRIZE-DT-($\kappa = 0$). However, the FAMILY-PRIZE-DT strategies are notably more computationally expensive, which is understandable since the FAMILY-PRIZE-DT tactical engines is approaching the limit of its ability to search the corresponding game-tree. In contrast, the CLUSTER-DT strategic game tree is reasonably small, as is each of the tactical game trees associated with the specific cluster targeting scenarios.

▼ Summary of Results by Strategy

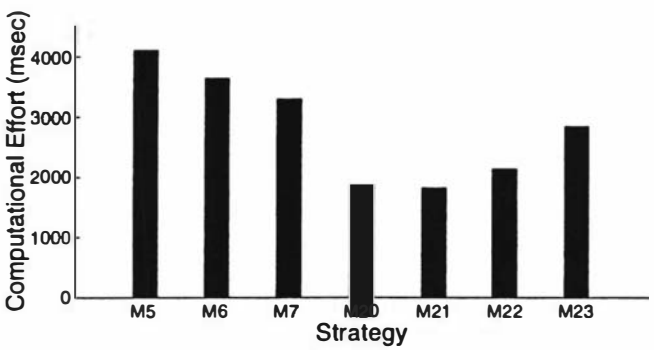
Table 10.19 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.17. Although the CLUSTER-DT strategies are more robust, we observe that the MIN-MIN nemesis is generally FAMILY-PRIZE-DT.

10.2.4 Medium-Large Final Tournament

The aim of the medium-large final tournament is to compare the effectiveness of, and computational effort required by, a benchmark set of the most robust strategies from the medium-large preliminary tournament (Section 10.1.4) against a set of challenging problem instances.



(a) Strategy Effectiveness



(b) Computational Effort

Figure 10.8: Medium Final Tournament: Results by Strategy

10.2.4.1 Participating Strategies

The participants in the medium-large final tournament are those recommended in the conclusion to the medium preliminary tournament as the most robust: **ML13**, **ML14**, **ML15**, **ML16**, **ML30**, **ML33**, and **ML36** from Table 10.11.

10.2.4.2 Problem Instances

The set of 60-prize problems are generated from the subclasses of D-class recommended in the conclusion to the medium preliminary tournament as the most difficult: the subclasses of the number of features component with the number of hot features ranging from two to three and the number of cold and black features ranging from zero to two. Fifty such problem instances were generated from each subclass. Knife-edge search is not applied to these problem instances, since we have no STATIC predictor capable of estimating the value of the game for problem instances of this size.

10.2.4.3 Results and Analysis

The results from the medium-large final tournament follow the same format as for the small final tournament (see Section 10.2.1).

Table 10.19: Medium Final Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
M5	-286	-224	-653	{M5}	-222	{M20}	904	181
M6	82	124	-456	{M5}	-76	{M6}	769	178
M7	-286	-267	-507	{M6}	-202	{M20}	943	118
M20	170	296	-174	{M6}	21	{M6}	1078	202
M21	107	191	-189	{M21}	-122	{M21}	1127	226
M22	-4	127	-194	{M6}	226	{M20}	1411	182
M23	187	234	-160	{M6}	-67	{M20}	1146	259

▼ Effectiveness and Computational Effort of Strategies

Figure 10.9 compares the overall effectiveness and computational effort by strategy, following the same specification as Figure 10.6. Figure 10.9(a) shows the *overall* 'MEAN-MEAN' *effectiveness* of each strategy, and Figure 10.9(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$).

We observe that the GRID-PATH strategies are both more effective and less computationally expensive than the CLUSTER-DT strategies. This implies that for the size and structure of problem instances considered in this tournament, GRID-PATH is the most appropriate strategy paradigm. There is little discernible difference, however, between the the tactical planning methods which serve GRID-PATH.

▼ Summary of Results by Strategy

Table 10.20 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.17. The MIN-MIN nemesis of each strategy is always GRID-PATH and GRID-PATH is more robust than CLUSTER-DT. These results indicate that GRID-PATH dominates CLUSTER-DT on a significant proportion of their battle simulations.

10.2.5 Large Final Tournament

The aim of the large final tournament is to compare the effectiveness of, and computational effort required by, a benchmark set of the most robust strategies from the large preliminary tournament (Section 10.1.5) against a set of challenging problem instances.

10.2.5.1 Participating Strategies

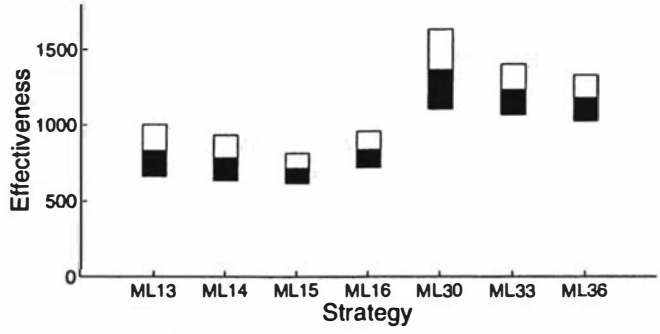
The participants in the medium-large final tournament are those recommended in the conclusion to the medium preliminary tournament as the most robust: **L7–L12** from Table 10.14.

10.2.5.2 Problem Instances

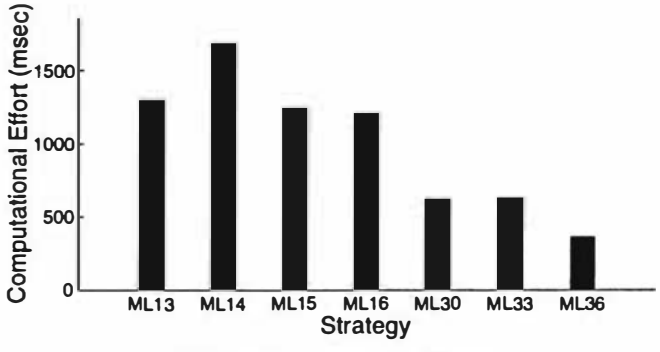
The set of 150-prize problems are generated from the subclasses of D-class recommended in the conclusion to the medium preliminary tournament as the most difficult: the subclasses of the

Table 10.20: Medium-Large Final Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
ML13	-223	-91	-448	**{ML36}**	14	**{ML30}**	834	168
ML14	-135	-35	-548	**{ML36}**	-19	**{ML13}**	786	148
ML15	27	123	-494	**{ML33}**	-88	**{ML15}**	718	99
ML16	-383	-321	-915	**{ML30}**	-13	**{ML33}**	841	118
ML30	69	158	-190	**{ML36}**	-42	**{ML13}**	1366	262
ML33	0	136	-209	**{ML30}**	281	**{ML14}**	1234	169
ML36	-64	-38	-195	**{ML36}**	394	**{ML15}**	1179	154



(a) Strategy Effectiveness



(b) Computational Effort

Figure 10.9: Medium-Large Final Tournament: Results by Strategy

number of features component with the number of hot features ranging from two to three and the number of cold and black features ranging from zero to two. Fifty such problem instances were generated from each subclass. Knife-edge search is not applied to these problem instances.

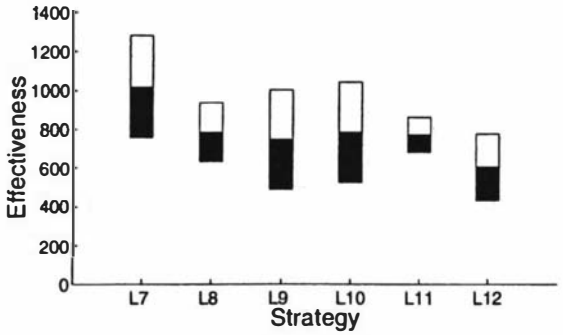
10.2.5.3 Results and Analysis

The results from the large final tournament follow the same format as for the small final tournament (see Section 10.2.1).

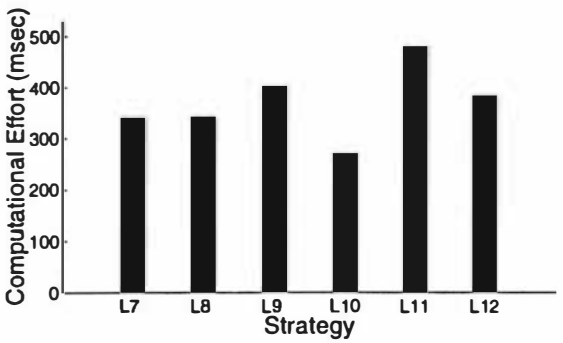
▼ Effectiveness and Computational Effort of Strategies

Figure 10.10 compares the overall effectiveness and computational effort by strategy, following the same specification as Figure 10.6. Figure 10.10(a) shows the *overall* 'MEAN-MEAN' *effectiveness* of each strategy and Figure 10.10(b) shows the *computational effort*, mean($\frac{\tau}{\Xi}$).

It appears that the $\mathcal{ABA}$-PATH tactical planning method is more effective than the HARVEST-PATH tactical planning method. This is understandable as $\mathcal{ABA}$-PATH is able to plan better under near-term tactical conflict than HARVEST-PATH, but $\mathcal{ABA}$-PATH provides essentially the same harvesting path as HARVEST-PATH when there is no tactical conflict in the near-term. There is little difference in computational effort between these strategies.



(a) Strategy Effectiveness



(b) Computational Effort

Figure 10.10: Large Final Tournament: Results by Strategy

▼ Summary of Results by Strategy

Table 10.21 presents a more extensive summary of the performance of each strategy following the same specification as Table 10.17. The MIN-MIN nemesis of each strategy is generally the $\mathcal{ABA}$-PATH version of GRID-PATH, indicating that this is the more difficult of the two tactical planning methods for another strategy to play against.

10.2.6 Conclusions from Final Computational Tournaments

For the problem instances (P-class and C-class) considered in the small, small-medium, and medium tournaments, the best performing strategies (both robust and effective) were PRIZE-DT, SINGLE-FAMILY-PRIZE-DT, and CLUSTER-DT respectively. Of particular note is that, in each case, $\kappa = 0$ performed better than either MINIMAX or MAXIMIN, and in the case of CLUSTER-DT, the PRIZE-DT-($\kappa = 0$) tactical engine was the best servant. This indicates that there is some combination of the number of prizes, and the natural structure in terms of clusters, which marks a transition from the PRIZE-DT paradigm of tactical planning to the CLUSTER-DT paradigm of strategic planning as the most effective paradigm. This supports the original design notion of the SPA/DMS in that the decision frames should map onto whatever natural, hierarchical structure is evident in a a particular problem instances by selecting an appropriate (dynamic) family-cluster

Table 10.21: Large Final Tournament: Results by Strategy

Strategy $a \in \mathbb{S}$	Worst Subclass		Nemesis				Overall	
	MIN-MEAN	MEAN-MEAN	MIN-MIN		MIN-MEAN		MEAN-MEAN	STD
L7	-181	-83	-713	**{L9}**	46	**{L7}**	1020	262
L8	-299	-153	-868	**{L7}**	-280	**{L10}**	785	153
L9	1	33	-321	**{L8}**	-54	**{L11}**	747	258
L10	-48	29	-603	**{L7}**	34	**{L11}**	786	260
L11	-47	41	-312	**{L8}**	135	**{L8}**	772	91
L12	-85	-5	-580	**{L12}**	-437	**{L7}**	605	171

structure and combination of *cluster-* and *prize-frames*.

For the problem instances (D-class) considered in the medium-large and large tournaments in which there were at least 60 prizes and little significant clustering, the GRID-PATH paradigms was found to be both computationally relatively inexpensive and the most effective. This indicates that, for large problem instances, harvesting in the right place is more significant than engaging the opponent.

Overall, the results from these computational tournaments have supported the notion that, for "discrete" problem instances in which it is possible to contingently plan, the PRIZE-DT and CLUSTER-DT DMS strategies are most effective but also expensive, whereas for "continuous" problem instances in which it is not possible to contingently plan in the medium- to long-term, the efficient sequencing of concentrations of prize value, without need for cognizance of the opponent, is both effective and inexpensive. In terms of the balance of strategy versus routing considerations, the small problem instances involve all strategy, the large problem instances involve all routing, while the medium problem instances require a combination of both strategy and routing.

Although we have tested few permutations of parameters, or different modular components of the strategy paradigms, we have shown that it is possible to construct a benchmark set of both strategies and problem instances by a "boot-strapping" process involving the evaluation of strategy robustness and problem difficulty. This process is very computationally demanding in comparison with the testing of VRSP heuristics, which require only the construction of a benchmark set of problem instances.

Coda

▼ Summary

This chapter has compared the strategies designed throughout this thesis on a number of classes of problem instances. We have successfully constructed set of benchmark robust strategies and challenging problem instances against which strategies may be evaluated. By way of computationally demanding tournament play, we have observed that, in general, the strategy paradigms designed to implement the SPA/DMS are the most effective for the size and structure of problem instance for which they are designed.

▼ Link

It remains in Chapter 11 to draw some major conclusions regarding the research questions of Section 2.7, integrating some conclusions about the implications for the new field of competitive routing.

CHAPTER 11

Conclusions and Recommendations for Future Research

Whatever one man is capable of conceiving, other men will be able to achieve.

— JULES VERNE

This thesis introduced a new family of problems to the field of vehicle routing and scheduling. We have laid some basic foundations which, we hope, future researchers will find useful for further investigation and experimentation. In this chapter we review the major achievements of this thesis and draw some conclusions with respect to the design of strategies and foundations of the topic. We also evaluate progress towards a number of research objectives and propose a programme of future research towards these research objectives.

11.1 Conclusions: Foundations and Strategies

The field of vehicle routing and scheduling is very large, with a comprehensive published literature and extensive practical implementation as software systems for distribution management. Although competition routing may be treated as a subfield of vehicle routing and scheduling, the converse is more appropriate, i.e., we can think of the existing field of vehicle routing and scheduling as a subfield of competition routing since any VRSP can be modelled as a CRP with a single decision maker. Since all the feature-rich characteristics of a VRSP may also be found in a CRP, the field of competitive routing is at least as large and hence it is important to review what has been established by this thesis in this new field.

11.1.1 Review

Part I made four major contributions to the foundations of competition routing. Firstly, we developed a reference model for competition routing problems by extending the logical structure of a vehicle routing and scheduling problem to including multiple decision makers (Sections 2.1–2.5). Secondly, we proposed a simple problem definition as a core problem for computational study; although this is not the only core problem, it includes the essential characteristics which strategies for any CRP must address (Section 2.6). Thirdly, we evaluated (and rejected) the existing VRSP solution paradigm as an appropriate solution paradigm for the CPCP (Chapter 3). Finally, we proposed a strategic planning architecture — based on planning horizon, dynamic response to observation, and scenario analysis — as a solution paradigm for further development and investigation (Chapter 4).

Part II expanded upon the strategic planning architecture with the design of strategies balancing elements of game theory and efficient routing. The approach was broad in the sense that we employed a range of solution techniques concomitant with a range of problem sizes and structures: mathematically analytic strategies for two prize problems (Chapter 5), game theory based strategies for problems with few prizes (Chapter 6), a balance of inter-cluster and intra-cluster planning for problems with a natural cluster structure (Chapter 7), and efficient routing based strategies for problems with many prizes (Chapter 8). The underlying goal was to design an implementation of the strategic planning architecture suitable for a range of problem sizes, hence recognising and resolving many strategic and technical problems in the process. As a result, there remains the opportunity to explore the strategy paradigms for specific problem sizes in more detail.

Part III has compared the strategies proposed in Chapter 3 and Part II with respect to robustness, effectiveness and computational effort. The computational experience gained in Chapter 10 is a necessary prerequisite for assessing which strategy paradigms should be investigated further and to identify their strengths and weaknesses. Initial progress towards the dual goals of understanding problem instances and evaluating the effectiveness of strategies has been to develop the computational tournament design as the method of evaluation for both problems and strategies.

11.1.2 Strategies

Considerable effort in this thesis has gone into designing strategies for the various problem instance sizes and structures. For the two prize problem, we can classify the current state of the game with respect to the prize and pair constraints and, if necessary, with respect to the accessibility and median-accessibility and then apply the most successful strategy for that subclass as determined by the tiny tournament. From the computational tournaments of Chapter 10 we can conclude that the defining strategies from Table 10.1 performed that best overall, except for the large tournament in which GRID-PATH was more effective than any of the GRID-DT based strategies.

We can conclude that the family of '-DT' strategies is a tenable benchmark against which future strategies of strategy paradigms may be compared. However, the focus of the design of strategies has been on applying the '-DT' strategies to implement the frames of the SPA. Hence, whilst we have shown that these '-DT' strategies perform well against the other strategies propose,

we cannot conclude that this is the best possible strategic paradigm. Even though the game-tree based tactical and strategic engines are computationally expensive, we would expect, given the game nature of the CPCP, that all robust, effective strategies would exhibit a similar order of magnitude in computational effort.

11.1.3 Foundations

The foundations of competition routing are encapsulated in the research challenges of Section 1.3 and the research questions of Section 2.7. The latter revolved around two considerations: understanding of the CPCP, and the relationship between the VRSP, CRP and CPCP.

11.1.3.1 The CPCP

In reviewing the Competitive Prize Collection Problem, as defined by Definition 2.6.1, we note that each of the core components of the definition are as simple as possible.

Prize contestability. Since all prizes are contestable, at any point during a play of the game, a player may consider that it is always possible to claim any prize which is not yet claimed. Moreover, this is also true in developing future scenarios. In summary, contestability is the essential defining characteristic of a CRP and future contestability is what distinguishes tactical planning from path planning.

Known prizes. Because all prize details are known in advance, contingent planning is characterised by future location and prizes claimed rather than with respect to expected future arrival of additional prizes. Although players' strategies are dynamic, the CPCP is static. If, instead, prize locations, prize values or prize existence were stochastic, then a strategy would need to balance tactical planning with stochastic expectation.

Euclidean plane. The Euclidean plane greatly simplified the path planning component of strategies allowing for the feasibility of PRIZE-DT-like tactical engines. This is because we can concentrate on prize targets rather than on the explicit steps required to get to the target. If instead the CPCP were to be played on a graph, the next vertex on the graph may be more important than the next prize to be moved to on the shortest path. In this way, the Euclidean plane removes a layer of decision making so that we have been able to concentrate on tactical planning rather than path planning.

Private objectives. We have designed strategies which either have no need to make assumptions regarding the opponent's objective or assume that the opponent is rational and intelligent (see Section 2.4.1). Furthermore, the only objective considered was to selfishly maximize the prize value collected. Although these are useful initial assumptions, there is still scope in between these two extremes. Since objectives are private, we have not designed any strategies to counter a particular opponent objective, nor have we attempted to predict the "style" of tactics that an opponent is adopting, i.e., opponent modelling. This is both a simplification and an opportunity lost.

Perfect observation. This is a huge simplification. The only uncertainty is due to the future movements of the opponent, but any point in time is essentially a restart. All information for a decision is "timely" which is a significant simplification in terms of designing strategies. If information is not timely, then an added complication is that a player may wish to move to gather information, as opposed to tactical movements.

Comparing the CPCP to the TSP, we see that both are simply stated, deceptively difficult problems. The difficulty with the TSP is computational, i.e., to determine a good solution requires considerable computational effort. However, the main difficulty for the CPCP is predictive: although it is computationally expensive to propose and evaluate scenarios, we do not know if the evaluation of the initial commitment is accurate, nor do we know the variability in the possible result if that choice is selected. The CPCP is therefore fundamentally different from the TSP.

11.1.3.2 The Field

Competition routing draws from both the field of game theory and the field of vehicle routing and scheduling. The interaction of the two fields has resulted in a contribution of one field to the other and vice versa.

The application of game theory to (competitive) vehicle routing is through the tactical/strategic and strategic engines which propose and evaluate tactical/strategic scenarios. The multiple-stage game tree formulation is an approximation of the tactical/strategic planning problem faced by the player at that point in the play of the game. A large number of simplifying assumptions are made in order to fit the game tree model to the decision problem. However, the amount of computational effort required to solve a more detailed approximation of the decision problem would make it effectively intractable. In game theory, the game tree formulation *is itself* the game, the computational difficulty is in efficiently and implicitly searching the game tree. For tactical/strategic planning, a game-based model approximates another game-based model to provide comparative information for decision making, although the value estimated for the chosen scenario may not eventuate. In summary, game theory can be useful in vehicle routing for formulating approximate static decision problems.

Decision problems in dynamic vehicle routing are often modelled as static snapshots of the current state of knowledge. There is, however, a large degree of prediction and observation of stochastic and dynamic influences on the decision problem, and these are encapsulated in static estimates or expectations. In game theory the decision problem is static. Conversely the decision problem faced by a player in the CPCP is dynamic, and so we have applied the dynamic concepts of prediction and observation from dynamic vehicle routing to the game-based model of the tactical/strategic planning problem. The game-based model itself adapts within the setting of the overall game. In summary, dynamic prediction and observation employed in dynamic vehicle routing are also useful in refining approximations to decision problems within a game.

Although the CPCP is fundamentally different from the TSP, this *does not imply* that the CRP is fundamentally different from the VRSP. Strategies for CRPs that are strongly competitive must consider CPCP-like principles, but strategies for CRPs that are weakly competitive may be simple adaptations of corresponding VRSP heuristics. Additionally, other constraints or features

of a given CPCP may dominate the competitive component and render tactical decision making less important.

11.1.3.3 Progress Towards Research Challenges

The research challenges identified in Section 1.3 were described as follows:

I. To model the foundational components of a CRP.

II. To identify and analyse the fundamental principles which strategies must address—both routing and strategic components—and hence formulate and evaluate an effective and efficient solution paradigm for the class of CRPs.

In this section we attempt to critique the contributions of this thesis vis-à-vis these challenges.

Towards research challenge I, the RM-CRP of Chapter 2 structured the range of possible CRPs within descriptive model components. Further, Chapter 4 described the components of a solution architecture. This certainly comprises a tenable component model for the general CRP. Whether the resulting model captures the "foundational components" is yet to be confirmed. The difficulty is inherited from the field of vehicle routing: although the VRP itself is well defined, in practice it is often difficult to decouple the vehicle routing component of a problem from other components. Hence, until a variety of CRPs have been investigated, we cannot begin to assess whether the foundational components have been correctly formulated. Different models may be necessary depending upon the significance of decision maker interaction to the overall problem in relation to other elements of the problem. On the other hand, through the CPCP we can infer some understanding of the interaction of decision makers, and conclude that both game theoretic and efficient routing considerations are important in balance, and that the balance is primarily dictated by computational resource with respect to problem size and structure.

Research challenge II encapsulates the ambitious goal of determining *the* solution paradigm for the CRP. Nevertheless, we have made some significant initial steps towards this goal. Firstly, we found that the VRSP solution paradigm—typified by the simple heuristic strategies of Chapter 3—was deficient with respect to a number of basic strategic principles. Identification of this subset of fundamental principles led to the design of the SPA/DMS of Chapter 4. However, there is still significant room for additional principles. Secondly, we have compared some simple strategies with strategies which implement the SPA/DMS for different problem sizes. These computational evaluations indicate that the strategies designed to implement the SPA/DMS are generally more effective than the simple strategies, although there is a wide gulf between the strategies in terms of strategic sophistication and computational effort. However, the SPA/DMS has been useful in identifying and structuring the design of strategies which attempt to realise the strategic principles that were identified as deficient.

In summary, the research challenges remain, but we have made significant progress against which future research may be compared. We can also see how a future programme of research may build upon this work to gain a deeper understanding of strategies for the CPCP and a broader understanding of competition routing.

11.2 Recommendations for Future Research

There are two primary directions in which future research should be directed. The first is to design new strategies and strategy paradigms for the CPCP, and formulate computational experiments designed to understand the performance of components of new and existing strategies or strategy paradigms. The second is to expand the foundations of competition routing by investigating the "next tier" of core competition routing problems so that the important strategic principles can be extracted for the addition of some new problem feature, particularly in comparison with corresponding additions to VRPs.

11.2.1 Strategies

In order to obtain a deeper understanding of the CPCP, we propose that existing strategy paradigms be improved and experimentally characterised, and that new strategy paradigms be designed and developed. Both of these are also important for other CRPs since experience with strategies for the CPCP may assist development of strategies for other CRPs and hence understanding of those CRPs.

11.2.1.1 Characterisation of Strategies

Following on from the comparison of effectiveness and computational effort of strategies in Chapter 10, further computational experiments must endeavour to understand the interaction of components and parameters of individual strategies. For example, understanding of FAMILY-PRIZE-DT can be decomposed into the PRIZE-MONITOR (accuracy of predicting *prize-ts*, narrowing of *prize-ts* through observation), the tactical engine (tradeoff between granularity of clustering, game tree search depth, heuristic fathoming, game table evaluation assumptions), the accuracy with which the tactical engine is able to predict value under a variety of scenarios (against an unpredictable opponent), and the frequency with which the tactical engine is invoked. This would necessarily involve computational experiments which isolate the contribution of an individual component or parameter to the overall effectiveness of a strategy, rather than comparison of a wide range of individual strategies.

11.2.1.2 Improvement of Strategy Paradigms

The existing strategy paradigms are currently represented by prototype implementations which capture the essential idea but have not been intensely tuned or refined. The following improvements to the current strategy paradigms may be considered:

Heuristic Evaluation. This is the approach taken by computer strategies for playing chess, since the game tree of all possible chess board configurations is too large to search in reasonable time. Pearl [174] states that "having no practical way of evaluating the exact status of successor game positions, one may naturally resort to heuristic approximations." The tactical engines which currently employ full game tree searching may be adapted to use heuristic evaluation, and fathoming and game tree nodes of a given depth. Additionally, we

may focus tactical planning on the near-term by discounting the perceived value of prizes planned past the near-term.

Tactical Engine: HARVEST-DT. Tactical planning need not evaluate the actual objective function that we are attempting to optimize. For example, we may modify the evaluation mechanism of PRIZE-DT to replace the expected prize value by the expected harvesting rate up to some future planning horizon. In this way, the tactics focus not only on efficient harvesting in the near-term, but also on contingent planning for claiming future prizes. The resulting tactical engine could be called HARVEST-DT, to follow the nomenclature introduced in this thesis.

Active Monitoring. Holland [101] has briefly investigated active monitoring (see Section 4.2.2) in the context of an iterated CPCP. The iterated CPCP is a (possibility infinite) sequence of instances of the CPCP played sequentially by the same players. Players have complete recall of the history of each CPCP instance played thus far in the sequence. The players learn by recording the performance of each component strategy played thus far and dynamically tuning the selection probabilities for each component strategy from MAX VALUE, GUARANTEED MAX VALUE, GREEDY and GRID MAX VALUE. While tactical engines rely upon efficient game tree searching, we have already seen that the tactics implied are only as good as the ability to match the tactics to the observation of the opponent. Hence, research into active monitoring involves the development of methods that incorporate experience into the dynamic choice of strategy.

11.2.1.3 New Strategy Paradigms

While the PRIZE-DT strategy, and those derived from it, have been shown to be effective in comparison with other simple strategies, new strategy paradigms need to be designed and developed. These should involve both methods based on other game-tree approximations of the decision problem and on heuristic methods encapsulating alternative evaluations of the game state. The following paradigms may be useful.

Tactical Engine: EVENT-DT. For sufficiently small problems, another tactical approach may be feasible that extrapolates the rigorous approach to the two prize problem of Chapter 5. Strategies for two prize problems are anchored to the prize and pair constraints (5.1)–(5.4), whereas strategies for small problems are guided by $\Gamma_A(\wp)$, $\Gamma_B(\wp)$ and $\Omega(\wp)$, at game position $\wp$, and that game position is called *static* if $\Gamma_A(\wp) + \Gamma_B(\wp) = \Omega(\wp)$ and *dynamic* otherwise. In simulation theory, an **event** is defined as a perceived change in the state of the system. Suppose we define the state of a game position as $\langle \Omega(\wp), \Gamma_A(\wp), \Gamma_B(\wp) \rangle$. We adapt the idea of a *probe* to construct a new type of projected game position as follows. Consider the scenario in which both players select a target-prize. The players move towards their respective target-prizes until the state of the game changes, i.e., until at least one of Ω, Γ_A, and Γ_B changes value. We call this type of projection an **event probe**. In this way we may define a tactical engine akin to ORIGINAL-DT, called EVENT-DT. The main difficulty is ensuring that the game tree is finite, or that an α–β search of the game tree is

finite, since a prize is not necessarily claimed at each depth of the game tree. Even if this is not possible, it may be useful just to use event probes at the root node of ORIGINAL-DT or PRIZE-DT.

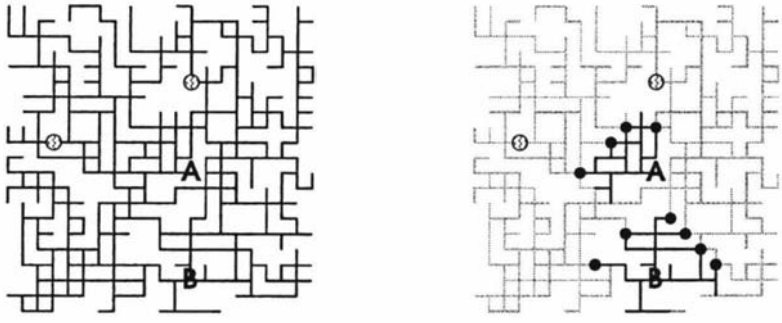
Covering. The Single Vehicle Routing Allocation Problem of Beasley and Nascimento [15] generalizes both the TSSP+1 and various TSP-like problems with a covering component (see Section 2.1). Three types of selection decisions are required: which prizes to visit, which prizes may be allocated to an on-route prize, and which prizes to leave isolated. This may be developed into a strategy paradigm for the CPCP. Covering is an implicit principle embedded in contingent planning. We wish to determine tactics such that a range of possible paths through a set of prizes is possible. Investigations could be in terms of strategy development for the CPCP—far-term planning involves covering subpaths rather than explicit subpaths—or by defining a covering variation of the CPCP (or competitive variation of the Covering Tour Problem) in which a prize may be claimed by visiting some location within a fixed distance of that prize rather than explicitly visiting the prize location.

11.2.2 Expanding the Foundations

The CPCP was conceived as the core version of the CRP. In this respect it was important to consider the CPCP before considering other possible CRPs. To build toward understanding CRPs in general, it will be necessary to formulate the "next tier" of CRPs which incorporate another atomic feature or change to the CPCP that is common to a range of possible CRPs. The goal would be to determine whether additional strategic principles are necessary in order to successfully "solve" the new problem and whether strategies for the CPCP may be adapted for the new problem. The following CRPs constitute an initial set of extensions to the CPCP, each of which adapts the CPCP with respect to some aspect of the address or decision maker component. Together these problems expand the foundations of competition routing, each making a nontrivial contribution to the balance of strategic planning.

11.2.2.1 Orienteering Event on Unknown Terrain

Chao, Golden and Wasil [35] describe the sport of Orienteering on which the Orienteering Problem is based (see Section 2.1.3). Orienteering is usually played in mountainous terrain in which visibility is restricted to "line of sight" only. Suppose that the CPCP is played on a terrain modelled by a digital terrain model and that the players do not know the location or value of a prize (nor whether the prize exists) until the player sights the prize. In addition, suppose that the complete terrain is initially unknown to the players and must be explored. We call such a problem the **Competitive Orienteering Event** (COE). Strategies for the COE must also address navigation in an unknown terrain (Mitchell [160] and Rao [184]), path planning with respect to obstacles (Rowe [188]) and terrain exploration such as planning *watchman paths* (Telcik[198]) which visit high points with good visibility of the terrain. Since the balance between navigation, exploration, and prize collection may be of interest in a military application, the COE is certainly an important core CRP.



(a) Two Prize Grid Network CPCP

(b) Valid Player Locations After Five Steps

Figure 11.1: A Grid Network CPCP

An additional characteristic for score orienteering events is the "motor rally" start in which players start at different times but they are out on the orienteering course for a fixed maximum time limit. Strategies must then consider a period, either at the start or end of the play, in which there is no competition.

11.2.2.2 Grid Networks

Grid networks have been used in Vehicle Routing and Scheduling Problems to model rectilinear city street networks including one way streets (Byrne [32]). The Grid Network CPCP is simply defined as the CPCP with the Euclidean plane replaced by a grid network. In particular, each player still has perfect observation of the prize locations and the opponent's location from any location on the grid network. The added strategic complication in this type of CRP is that players' movements are more restricted.

Consider the two prize problem on a grid network in Figure 11.1; the "sparse grid network" is from Frizzell [71, page 173]. The game position of Figure 11.1(a) is dynamic. The equivalent of a median location for player $\mathcal{B}$ may not necessarily lie along a shortest path between the two prizes. Suppose that the players move one grid unit at each step. Figure 11.1(b) shows the possible locations of the players after five steps. Some of these locations are dead ends (by inspection) and some take the players no closer to either prize. The non-dominated frontier is indicated as solid dots for each player.

The usual approach for solving the TSP or VRSP on a grid network is to determine the shortest path *distances* between cities (prizes); however, for the Grid Network CPCP we need to plan in terms of actual paths. Additionally, a path which has more branching locations may be preferred to a shortest path which has fewer branching locations. Hence a strategy must consider explicit path planning with respect to possible target prizes, rather than only shortest path distances.

If we were to restrict the ability of players to observe prizes or the opponent to line of sight along an edge in the grid, then the resulting problem would have much in common with the

unknown terrain problem and also with maze exploration problems in Artificial Intelligence.

11.2.2.3 Competitive Dial-A-Ride Problems

The **Many-To-Many Dial-A-Ride Problem** (DARP) involves both pickup and delivery of customers such that each customer is picked up at its source location before being dropped off at its destination location (Psaraftis [178]). Beale [14] proposed the **Task Routing Collection Problem** (TRCP) as a DARP version of the OP in which the prize value is received once the customer is delivered to its destination, and not all customers need be serviced.

A competitive version of the TRCP, the **Competitive Dial-A-Ride Problem** (CDARP) involves two players competing over the same set of customers. To enforce delivery of customers, each player is capacitated, i.e., there is a fixed maximum number of customers that have been picked up by a player that have not yet been dropped off. The CDARP is significant since the dial-a-ride problems usually occur in a dynamic environment where calls for service occur in real time, and this is a likely setting in which practical competitive routing problems may arise.

Strategies for small CDARPs with unit capacity are straightforward extensions of the strategies for small CPCPs since, whenever a player picks up a prize at its source location, that player is committed to the direct path to that prize's destination location. Although the strategies for small problems may not be significantly different, for larger problems the ratio of service time (the time between pickup and delivery of a customer) to vacant time (the time during which the player's vehicle is empty) determines the extent to which spatial proximity of prize source locations is significant and, therefore, the extent to which clustering is useful.

The **Many-To-One Dial-A-Ride Problem** restricts all destinations to a single location. The capacitated competitive version involves collecting prizes such that at most k prizes may be carried by a player at any one time and prizes may be deposited at some location which serves as a bank. An interesting prospect is the possibility that a player may put down a prize already collected in favour of a more valuable prize.

11.2.2.4 Active Cooperation

Passive cooperation occurs when the players unintentionally cooperate for mutual benefit when there is restrictive overall deadline. No communication takes place between the players. It is difficult to *observe* passive cooperation distinct from competition since they occur concurrently. If communication is permitted between players and contractual agreements are enforceable with severe penalties, then the players may agree that each should follow a prescribed path for a period of time. This is **active cooperation**, the deliberate contractual cooperation between the players for mutual benefit over a period of play. Active cooperation is only possible by extending the definition of the CPCP by incorporating an explicit mechanism for the players to make agreements. The addition strategic complications in this type of CRP are the needs to be able to assess the opportunity cost of noncooperation and to employ an effective bargaining strategy.

11.2.2.5 Real-Time CPCP

Séguin, Potvin, Gendreau, Crainic and Marcotte [192] consider the class of **Real Time Decision Problems** (RTDP) in which the objective is to provide responses of a required quality in a continuously evolving environment, within a prescribed time frame, using limited resources and information that is often incomplete or uncertain (see Section 2.3.3). In a **Real-Time CPCP** (RT-CPCP) the play of the game evolves in real time concurrent with the computation time necessary to make decisions. Balancing computational requirements for decision making with evolving movements in the game is what makes this type of CRP different from the CPCP.

When the response time is long, i.e., the computation time is much quicker than the evolution of the game, the decision problem resembles a strict computational budget for each move. However, when the response time is short, i.e., the computation time is of the same order as the evolution of the game, short-term and long-term planning must occur on the same time scale as the game is being played.

11.2.2.6 Multiple Players, Teams, and Dynamic Cooperation

The CPCP is defined with two players only. When more than two players are involved, the strategic and tactical interactions between the players are much more complex. A *fixed team* consists of several players who must cooperate perfectly, hence a fixed team corresponds to a single decision maker with a single team objective and multiple vehicles. A *dynamic coalition* consists of several players or teams who agree to cooperate for mutual benefit until one of them withdraws from the coalition. Upon entering the coalition, the players agree on how the prizes collected will be distributed over the life of the coalition. Each player or team corresponds to a decision maker who retains an individual objective.

The study of problems involving multiple players requires the dynamic application of non-cooperative game theory. Since more tactics are possible, especially in considering collusion with the opponents, the scarce computational resource implies that more heuristic strategies would be necessary rather than game-tree based strategies. The multi-player version of minimax, called *maxn* by Mutchler [164], is a conservative evaluator which assumes that all the opponents collude to create the worst possible scenario.

11.2.2.7 Two Fixed Teams and Two Prizes

Suppose there are two prizes and two fixed teams—$\{\mathcal{A}\}$ and $\{\mathcal{B},\mathcal{C}\}$—and constraints (11.1)–(11.4) hold.

$$
\begin{aligned}
d_{A1} &< \min\{d_{B1}, d_{C1}\} && (11.1)\\
d_{A2} &< \min\{d_{B2}, d_{C2}\} && (11.2)\\
d_{A1} + d_{12} &> \max\{d_{B2}, d_{C2}\} && (11.3)\\
d_{A2} + d_{12} &> \max\{d_{B1}, d_{C1}\} && (11.4)
\end{aligned}
$$

This is easily solved. The optimal strategy for team $\{\mathcal{B},\mathcal{C}\}$ is $\mathcal{B}\rightarrow 1$ and $\mathcal{C}\rightarrow 2$ (one of them will certainly claim a prize). The optimal strategy for player $\mathcal{A}$ is $\mathcal{A}\rightarrow\text{argmax}\{v_1, v_2\}$.

Suppose there are two prizes and two fixed teams—$\{\mathcal{A},\mathcal{C}\}$ and $\{\mathcal{B}\}$—and constraints (11.5)–(11.8) hold.

$$d_{A1} < d_{B1} < d_{C1} \tag{11.5}$$
$$d_{A2} < d_{B2} < d_{C2} \tag{11.6}$$
$$d_{A1} + d_{12} > d_{B2} \tag{11.7}$$
$$d_{A2} + d_{12} > d_{B1} \tag{11.8}$$

Now we must ask what effect player $\mathcal{C}$ has on the well-understood interaction between player $\mathcal{A}$ and player $\mathcal{B}$. This case may still be *dynamic* since player $\mathcal{B}$ may be able to establish a median feasible location from which player $\mathcal{C}$ *cannot* conspire with player $\mathcal{A}$ to deprive player $\mathcal{B}$ of a prize. Strategies for both teams would revolve around six slack variables. Even for two-prize three-player problems, there remain interesting problems which require strategic analysis.

11.2.2.8 Development of a Research Programme

The formulation and development of variations of the CPCP, comprising fixed team and dynamic cooperation between players, provide a substantial programme of future work. Figure 11.2 shows the dependencies between the following problems. The problems on the right hand side of the figure are those involving individual players, initially restricted so that no form of active cooperation with the opponent is possible (although incidental cooperation may occur) before active cooperation is studied involving dynamic (possibly temporary) coalitions between players. The parallel problems on the left hand side of the figure are those involving fixed inflexible teams, culminating in the **Multiple Fixed Team Dynamic Cooperative CPCP** (MFTDC-CPCP). The problems on each level can be developed independently but both problems on a level depend upon both problems from the level above. The *Two Fixed Team CPCP* (TFT-CPCP) extends the CPCP to two fixed teams and, as such, depends upon understanding both the TOP and the CPCP. The **Three Player Dynamic Cooperative CPCP** (TPDC-CPCP) depends upon both the **Multiple Player Non-Cooperative CPCP** (MPNC-CPCP)—since a player must be able to compare a completely independent campaign with that of forming a coalition with either or both of the opponents—and the Two Fixed Team CPCP—since a player must be able to evaluate the possibility of playing the entire game against the two opponents' cooperative campaign.

Coda

This thesis has introduced the field of competition routing, made contributions to problem modelling, solution architecture, strategy design and computational evaluation, and proposed a research programme for future development. We hope that this thesis has been of interest and that competition routing may become an active topic of future research.

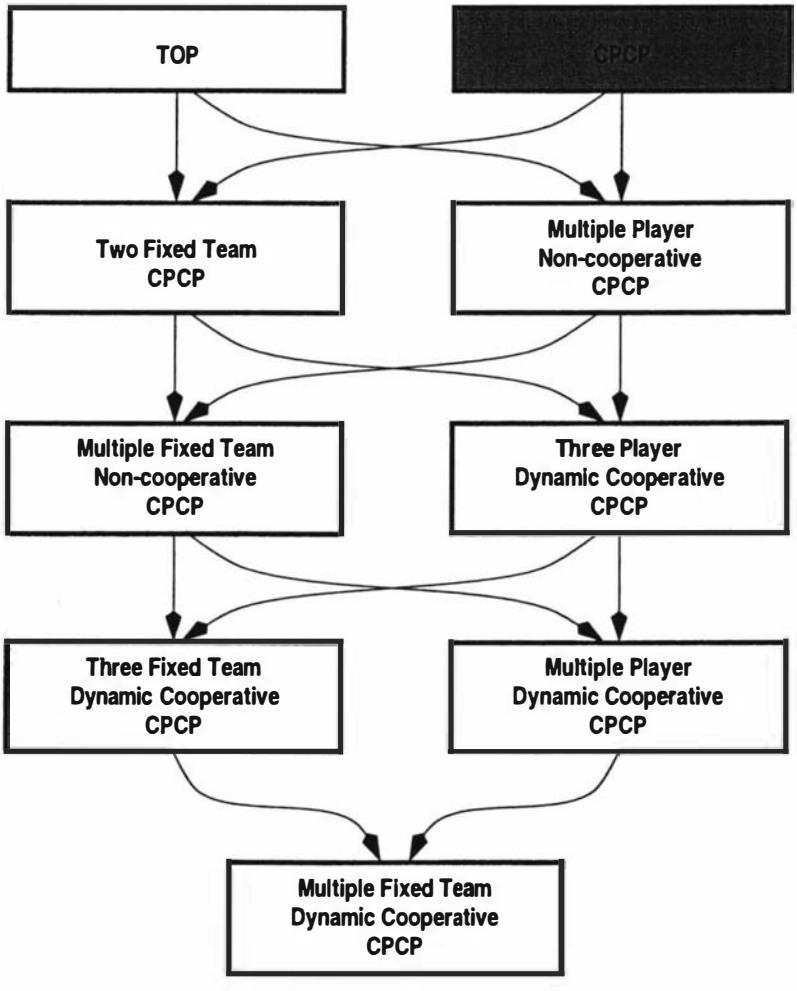


Figure 11.2: Dependencies between Team and Cooperative CPCPs

Appendices

APPENDIX A

Competition Routing Kernel Environment (CRiKET)

There is always one more bug.
— LUBARSKY'S LAW OF CYBERNETIC ENTOMOLOGY

The strategies discussed in this thesis have been implemented as a combination of 'C' and MATLAB to form a **Competition Routing Kernel Environment** (CRiKET). This appendix outlines the major components of the CRiKET system. The facilities in CRiKET form the kernel of implementations to investigate competitive routing. These would be easily expandable to other similar problems including three or more players and task routing (dial-a-ride) type problems.

A.1 CRiKET Tournament

CRiKET Tournament implements the computational tournaments of Chapter 10. It processes '`*.cb`' bulk format problem instance files, created by CRiKET Generator of Section A.4, consisting of multiple problem instances from a single problem subclass, and produces corresponding '`*.cbresult`' tournament results files, and '`*.cbvalue`' problem expected value files. These are then summarised by a set of MATLAB scripts to format the LaTeX result tables, and MATLAB barcharts, in Chapter 10.

A.2 CRiKET Simulator

The CRiKET Simulator is used to run a pair of strategies on a single problem instance.

There are two data formats used by the simulator:

- A '*.cp' file contains a single problem instance plus a (possibly empty) history of the player of the game. This is the file format also used by the CRiKET VIEWER. The advantage of this format is that we can easily save the game and any point and restart from that point if desired, thus can do periodic dumps of the game throughout its run.
- A '*.cs' file contains the parameters for a single strategy.

A '*.cp' single format problem instance file can be *extracted* from a '*.cb' bulk format problem instance file. Table A.1 gives a specification of the single format problem instance file and Table A.2 gives a specification of the bulk format problem instance file. Also, a '*.cs' strategy file consists of the strategy ID number, the number of parameters and a sequence of parameter values.

A.3 CRiKET Viewer

The CRiKET Viewer animates a '*.cp' single format problem instance file. This contains all the initial problem instance (overall time limit, player locations, prize locations and values and step size) and also the (partial) history of a play of the game (player locations and when and by whom each prize is claimed if claimed). This is primarily an interactive tool for visualising passages of play or saving particular game states. Figure A.1 is a snapshot of the main CRiKET viewer window.

Figure A.2 is a snapshot of the CRiKET viewer controls window which includes the pause, step, time direction (forward or reverse) and quit controls and scoreboard features including current simulation time and current total prize value claimed by each player.

The facilities offered by the CRiKET VIEWER include:

⟨load⟩ Load a .cp problem file. Specify a starting time or alternatively specific a starting prize (which is eventually claimed) and let the starting time be the time at which that prize is claimed. Similarly specify a stopping time or alternatively a stopping prize. The default starting time is time zero and the default stopping time is the earlier of the overall deadline λ and the time at which the last prize is claimed.

⟨pause/unpause⟩ Suspend or recommence animation.

⟨direction⟩ Change direction of time, playing forwards or backwards.

⟨speed⟩ Select the number of steps which are skipped between frames of the animation.

⟨replay⟩ Jump back to the start time if direction is forward or jump forward to the end time if direction is backward.

⟨jump⟩ Jump to a specific time or prize claim.

⟨save⟩ Save a '.cp' which is either a specific game state (which is either the state of the current paused time) or the pre-specified start–stop or start–current or current–stop.

Table A.1: Specification of Single Format Problem Instance File

step_size	*// problem step size*
ON_overall_deadline	*// 0=no deadline,1=deadline exists*
overall_deadline	*// problem overall deadline* λ
num_prizes	*// number of prizes*
suggest_num_clusters	*// suggested number of clusters*
suggest_num_families	*// suggested number of families*
for each *prize*	
prize_value	
prize_location_Xcoord	
prize_location_Ycoord	
prize_claim_who	*// 0=unclaimed,1=claimed by player* $\mathcal{A}$
	// 2=claimed by player $\mathcal{B}$*,3=shared*
prize_claim_time	*// time prize was claimed if claimed*
end	
num_previous_steps	*// number of previous historical iterations*
for each *previous iteration*	
history_playerA_location_Xcoord	
history_playerA_location_Ycoord	
history_playerB_location_Xcoord	
history_playerB_location_Ycoord	
history_playerA_target_prize	
history_playerB_target_prize	
end	
playerA_current_location_Xcoord	
playerA_current_location_Ycoord	
playerB_current_location_Xcoord	
playerB_current_location_Ycoord	

Table A.2: Specification of Bulk Format Problem Instance File

```
problem_class                 // 0=two prize,1=prize class
                              // 2=cluster class,3=density class
problem_subclass              // subclass within class
problem_numinstances          // number of problem instances
for each problem
   99999                      // problem separator
   step_size                  // problem step size
   ON_overall_deadline        // 0=no deadline,1=deadline exists
   overall_deadline           // problem overall deadline λ
   num_prizes                 // number of prizes
   suggest_num_clusters       // suggested number of clusters
   suggest_num_families       // suggested number of families
   for each prize
      prize_value
      prize_location_Xcoord
      prize_location_Ycoord
   end
   playerA_location_Xcoord
   playerA_location_Ycoord
   playerB_location_Xcoord
   playerB_location_Ycoord
end
```

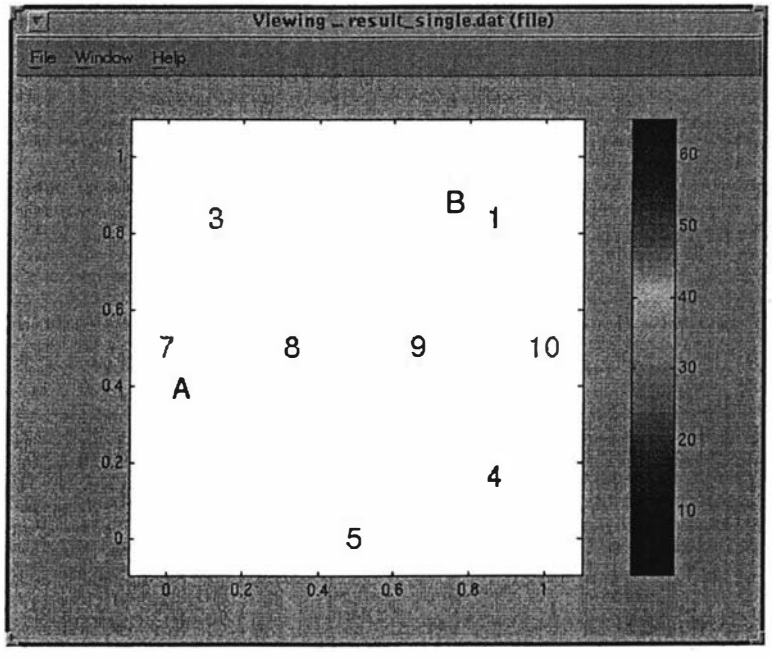


Figure A.1: CRiKET Viewer

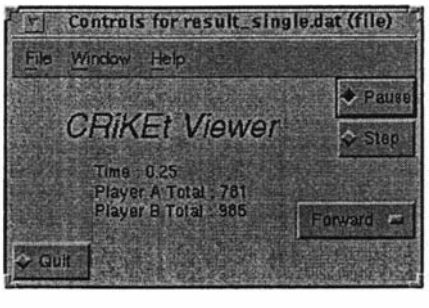


Figure A.2: CRiKET Viewer Controls

The importance of being able to save '.cp' files with a subset of the game history is that we can then run the CRiKET SIMULATOR from the stop state with the history subset standing as the entire history.

A.4 CRiKET Generator

CRiKET Generator constructs '*.cb' bulk format problem instance files. This consists of several MATLAB scripts for constructing problems in bulk, and interactive scripts for manipulating and *mosaicing* problem instances. The tiny problem instances of the tiny tournament are generated by a randomised search scheme.

A.5 CRiKET Sensitivity

The analyser is used to generate sensitivity plots of Chapter 9, and to implement the knife-edge search for bad-case problems of Section 9.4.2. The actual sensitivity figures are plotted in MATLAB.

Bibliography

[1] AARTS, E., AND LENSTRA, J. K., Eds. *Local Search in Combinatorial Optimization.* Wiley, 1997.

[2] ANDERSON, E. J. Mechanisms for local search. *European Journal of Operational Research 88* (1996), 139–151.

[3] ANDERSON, E. J., AND ARAMENDIA, M. A. The search game on a network with immobile hider. *Networks 20* (1990), 817–844.

[4] ANEJA, Y. P., AND PUNNEN, A. P. Efficient heuristics for the prize collecting TSP. Presented at APORS '97 conference, Melbourne, Australia, December 1997.

[5] AWERBUCH, B., AZAR, Y., BLUM, A., AND VEMPALA, S. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM Journal on Computing 28*, 1 (1998), 254–262.

[6] AWERBUCH, B., AZAR, Y., PLOTKIN, S., AND WAARTS, O. Competitive routing of virtual circuits with unknown duration. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (Arlington, Virginia, 23–25 Jan. 1994), pp. 321–327.

[7] AXELROD, R. *The Evolution of Cooperation.* Penguin, London, New York, 1990.

[8] AXELROD, R. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration.* Princeton University Press, Princeton, New Jersey, 1997.

[9] BAKER, E. K. An exact algorithm for the time-constrained traveling salesman problem. *Operations Research 31* (1983), 938–945.

[10] BALAS, E. The prize collecting traveling salesman problem. *Networks 19* (1989), 621–636.

[11] BALAS, E. The prize collecting traveling salesman problem: II. Polyhedral results. *Networks 25* (1995), 199–216.

[12] BALL, M. O. Allocation/routing: models and aslgorithms. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 199–221.

[13] BARR, R. S., GOLDEN, B. L., KELLY, J. P., RESENDE, M. G. C., AND STEWART, W. R. Designing and reporting on computational experiments with heuristics. *Journal of Heuristics 1* (1995), 9–32.

[14] BEALE, I. R. The task routing collection problem. Presented at APORS '97 conference, Melbourne, Australia, December 1997.

[15] BEASLEY, J. E., AND NASCIMENTO, E. M. The vehicle routing-allocation problem: a unifying framework. *T.O.P. 4*, 1 (1996), 65–86.

[16] BENTLEY, J. L. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on Computing 4*, 4 (1992), 387–411.

[17] BERTSIMAS, D., AND SIMCHI-LEVI, D. The new generation of vehicle routing research: Robust algorithms addressing uncertainty. *Operations Research 44* (1994), 286–304.

[18] BERTSIMAS, D. J., AND VAN RYZIN, G. A stochastic and dynamic vehicle routing problem in the euclidean plane. *Operations Research 39*, 4 (1991), 601–615.

[19] BHATTACHARYA, S., AND BAGCHI, A. A general framework for minimax search in game trees. *Information Processing Letters 52* (1994), 295–301.

[20] BIENSTOCK, D., GOEMANS, M. X., SIMCHI-LEVI, D., AND WILLIAMSON, D. A note on the prize collecting salesman problem. *Mathematical Programming 59* (1993), 413–420.

[21] BINMORE, K. G. *Fun and Games.* D. C. Heath and Company, Lexington, Mass., 1992.

[22] BODIN, L. D., GOLDEN, B. L., ASSAD, A. A., AND BALL, M. O. Routing and scheduling of vehicles and crews: the state of the art. *Computers and Operations Research 10* (1983), 63–211.

[23] BONDY, J. A., AND MURTY, U. S. R. *Graph Theory with Applications.* North Holland, New York, 1976.

[24] BOOKER, L. B., GOLDBERG, D. E., AND HOLLAND, J. H. Classifier systems and genetic algorithms. *Artificial Intelligence 40* (1989), 235–282.

[25] BOWERS, M., NOON, C. E., AND THOMAS, B. A parallel implementation of the TSSP+1 decomposition for the capacity-constrained vehicle routing problem. *Computers and Operations Research 23*, 7 (1996), 723–732.

[26] BOYD, J. A., CLARKE, J., GEMMELL, Z., AND MILLER, K. Modeling competition in vehicle routing. Unpublished BTech project, Massey University, 1997.

[27] BRANDENBURGER, A. M., AND NALEBUFF, B. J. *Co-opetition.* Doubleday, New York, 1996.

[28] BRIDEAU, R. J., AND CAVALIER, T. M. The maximum collection problem with time-dependent rewards. Unpblished working paper, Pennsylvania State University, 1994.

[29] BROOKSHEAR, J. G. *Computer Science: An Overview*, 3rd ed. Benjamin/Cummings, 1991.

[30] BUTT, S., AND RYAN, D. Using column generation to solve the multiple tour maximum collection problem. In *Proceedings of the 32nd Annual ORSNZ Conference* (1996), pp. 143–148.

[31] BUTT, S. E., AND CAVALIER, T. M. A heuristic for the multiple tour maximum collection problem. *Computers and Operations Research 21*, 1 (1994), 101–111.

[32] BYRNE, M. D. *Aspects of the Vehicle Routing Problem with Pickup and Delivery.* PhD thesis, Massey University, 1993.

[33] CAO, B. Search-hide games on trees. Tech. rep., Bericht S-9302, IASFOR (Institu für Angewandte Systemforschung und Operations Research), Universität der Bundeswehr München, 1993.

[34] CAO, B., AND VON STENGEL, B. Search-hide games on graphs. Tech. rep., Bericht S-9303, IASFOR (Institu für Angewandte Systemforschung und Operations Research), Universität der Bundeswehr München, 1993.

[35] CHAO, I.-M., GOLDEN, B. L., AND WASIL, E. A. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research 88* (1996), 475–489.

[36] CHAO, I.-M., GOLDEN, B. L., AND WASIL, E. A. The team orienteering problem. *European Journal of Operational Research 88* (1996), 464–474.

[37] CHECKLAND, P. *Systems Thinking, Systems Practice.* J. Wiley, 1981.

[38] CHIKRIJ, A. A., AND GLUSHKOV, V. M. Methods of group pursuit. In *Stochastic Optimization V* (1986), V. I. Arkin, A. Shiraev, and R. Wets, Eds., Lecture Notes in Control and Information Sciences, Springer-Verlag, pp. 632–640.

[39] CHISMAN, J. A. The clustered traveling salesman problem. *Computers and Operations Research 2* (1975), 115–119.

[40] CHRISTOFIDES, N., MINGOZZI, A., AND TOTH, P. The vehicle routing problem. In *Combinatorial Optimization*, N. Christofides, A. Mingozzi, P. Toth, and C. Sandi, Eds. Wiley, Chicester, 1979, pp. 315–338.

[41] CONDON, A., FICH, F., FREDERICKSON, G. N., GOLDBERG, A. V., JOHNSON, D. S., LOUI, M. C., MAHANEY, S., RAGHAVAN, P., SAVAGE, J., SELMAN, A., AND SHMOYS, D. B. Strategic directions for research in theory of computing. *SIGACT News 28*, 3 (1997), 75–93. `http://wocket.csl.uiuc.edu/~loui/complete.html`.

[42] CURIEL, I., PEDERZOLI, G., AND TIJS, S. Sequencing games. *European Journal of Operational Research 40* (1989), 344–351.

[43] CURRENT, J., AND MARSH, M. Multiobjective transportation network design and routing problems: taxonomy and annotation. *European Journal of Operational Research 65* (1993), 4–19.

[44] CURRENT, J., AND MIN, H. Multiobjective design of transportation networks: taxonomy and annotation. *European Journal of Operational Research 26* (1986), 187–201.

[45] CURRENT, J., PIRKUL, H., AND ROLLAND, E. Efficient algorithms for solving the shortest covering path problem. *Transportation Science 28*, 4 (1994), 317–327.

[46] CURRENT, J. R., AND SCHILLING, D. A. The covering salesman problem. *Transportation Science 23* (1989), 208–213.

[47] CURRENT, J. R., AND SCHILLING, D. A. The median tour and maximal covering tour problems: formulations and heuristics. *European Journal of Operational Research 73* (1994), 114–126.

[48] DANTZIG, G. B., AND RAMSER, J. H. The truck dispatching problem. *Management Science 6*, 1 (1959), 80–91.

[49] DAWES, R. W. Some pursuit–evasion problems on grids. *Information Processing Letters 43* (1992), 241–247.

[50] DE AMORIM, S. G., BARTHÉLEMY, J.-P., AND RIBEIRO, C. C. Clustering and clique partitioning: simulated annealing and tabu search approaches. *Journal of Classification 9* (1992), 17–41.

[51] DELL'AMICO, M., MAFFIOLI, F., AND VÄRBRAND, P. On prize-collecting tours and the asymmetric travelling salesman problem. *Int. Trans. Opl Res. 2* (1995), 297–308.

[52] DENARDO, E. V. *Dynamic Programming: Models and Applications.* Englewood Cliffs, New Jersey, 1982.

[53] DEO, N., AND PANG, C.-Y. Shortest-path algorithms: taxonomy and annotation. *Networks 14* (1984), 275–323.

[54] DESROCHERS, M., LENSTRA, J. K., AND SAVELSBERGH, M. W. P. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research 46* (1990), 322–332.

[55] DIABY, M., AND RAMESH, R. The distribution problem with carrier service: a dual based penalty approach. *ORSA Journal on Computing 7* (1995), 24–35.

[56] DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[57] DONALD, B. R. Computational robotics: the geometric theory of manipulation, planning and control. *Algorithmica 10* (1993), 91–101. Guest Editor's Foreward to Special Issue.

[58] DREYFUS, S. E. An appraisal of some shortest path algorithms. *Operations Research 17* (1969), 395–412.

[59] DROR, M., LAPORTE, G., AND TRUDEAU, P. Vehicle routing with stochastic demands: properties and solution frameworks. *Transportation Science 23* (1989), 166–176.

[60] DROR, M., AND POWELL, W. Stochastic and dynamic models in transportation. *Operations Research 41* (1993), 11–14. Preface to Special Issue on Stochastic and Dynamic Models in Transportation.

[61] DUECK, G. New optimization heuristics the great deluge algorithm and the record-to-record travel. *Journal of Computational Physics 104* (1993), 86–92.

[62] DUECK, G., AND SCHEUER, T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics 90* (1990), 161–175.

[63] EILON, S., WATSON-GANDY, C. D. T., AND CHRISTOFIDES, N. *Distribution Management: Mathematical Modelling and Practical Analysis.* Griffin, London, 1971.

[64] ERKUT, E., AND ZHANG, J. The maximum collection problem with time-dependent rewards. *Naval Research Logistics 43* (1996), 749–763.

[65] EVERITT, B. S. *Cluster Analysis*, 3rd ed. Halsted Press, New York, 1993.

[66] FEKETE, S., AND SCHMITT, M. Traveling salesmen in the age of competition. ZPR 97.266, Center for Parallel Computing, Universität zu Köln, 1997.

[67] FEO, T., AND RESENDE, M. Greedy randomized adaptive search procedures. *Journal of Global Optimization 16* (1995), 109–133.

[68] FISCHETTI, M., AND TOTH, P. An additive approach for the optimal solution of the prize-collecting travelling salesman problem. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 319–343.

[69] FLOYD, R. W. Algorithm 97: shortest path. *Communications of the Association for Computing Machinery 5* (1962), 345.

[70] FOULDS, L. R. *Combinatorial Optimization for Undergraduates.* Springer-Verlag, Berlin; New York, 1984.

[71] FRIZZELL, P. W. *Aspects of the Vehicle Routing and Scheduling Problem.* PhD thesis, Massey University, 1993.

[72] FUDENBERG, D., AND TIROLE, J. *Game Theory.* MIT Press, Cambridge, 1991.

[73] GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP Completeness.* Freeman, San Francisco, CA, 1979.

[74] GENDREAU, M., HERTZ, A., AND LAPORTE, G. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research 40* (1992), 1086–1094.

[75] GENDREAU, M., HERTZ, A., AND LAPORTE, G. A tabu search heuristic for the vehicle routing problem. *Management Science 40*, 10 (1994), 1276–1290.

[76] GENDREAU, M., LAPORTE, G., AND SÉGUIN, R. Stochastic vehicle routing. *European Journal of Operational Research 88* (1996), 3–12.

[77] GENDREAU, M., LAPORTE, G., AND SEMET, F. The covering tour problem. *Operations Research 45*, 4 (1997), 568–576.

[78] GENSCH, D. H. An industrial application of the traveling salesman's subtour problem. *AIIE Transactions 10* (1978), 362–370.

[79] GLOVER, F. Tabu search. Part I. *ORSA Journal on Computing 1*, 3 (1989), 190–206.

[80] GLOVER, F. Tabu search. Part II. *ORSA Journal on Computing 2*, 1 (1990), 4–32.

[81] GLOVER, F. Genetic algorithms and scatter search: unsuspected potentials. *Statistics and Computing 4* (1994), 131–140.

[82] GLOVER, F. W., AND LAGUNA, M. *Tabu Search.* Kluwer, Boston, 1997.

[83] GOEMANS, M. X., AND WILLIAMSON, D. A general approximation technique for constrained forest problems. *SIAM Journal on Computing 24* (1995), 296–317.

[84] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, Mass, 1989.

[85] GOLDEN, B., ASSAD, A., AND DAHL, R. Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems 7* (1984), 181–190.

[86] GOLDEN, B., LEVY, L., AND DAHL, R. Two generalizations of the traveling salesman problem. *Omega 9*, 4 (1981), 439–441.

[87] GOLDEN, B. L., AND ASSAD, A. A., Eds. *Vehicle Routing: Methods and Studies.* Elsevier Science, New York, 1988.

[88] GOLDEN, B. L., BODIN, L., DOYLE, T., AND STEWART, W. R. Approximate travelling salesman algorithms. *Operations Research 28* (1980), 694–711.

[89] GOLDEN, B. L., LEVY, L., AND VOHRA, R. The orienteering problem. *Naval Research Logistics 34* (1987), 307–318.

[90] GOLDEN, B. L., AND STEWART, W. R. Empirical analysis of heuristics. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Eds. Wiley, New York, 1985, pp. 207–249.

[91] GOLDEN, B. L., WANG, Q., AND LIU, L. A multifaceted heuristic for the orienteering problem. *Naval Research Logistics 35* (1988), 359–366.

[92] GÖTHE-LUNDGREN, M., JÖRNSTEN, K., AND VÄRBRAND, P. On the nucleolus of the basic vehicle routing game. *Mathematical Programming 72* (1996), 83–100.

[93] GREENBERG, H. J. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. CCM No. 108, Center for Computational Mathematics, Mathematics Department, University of Colorado at Denver, Denver, CO, 1997.

[94] HAIMOVICH, M., RINNOOY KAN, A. H. G., AND STOUGIE, L. Analysis of heuristics for vehicle routing problems. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 47–61.

[95] HAMACHER, A., AND MOLL, C. The euclidian traveling salesman selection problem. Report no 95-199, Zentrum für Paralleles Rechnen, Universität zu Köln, 1995.

[96] HAYES, M., AND NORMAN, J. M. Dynamic programming in orienteering: route choice and the siting of controls. *Journal of the Operational Research Society 35*, 9 (1984), 791–796.

[97] HENIG, M. I. The shortest path problem with two objective functions. *European Journal of Operational Research 25* (1985), 281–291.

[98] HILLMAN, A. P., ALEXANDERSON, G. L., AND GRASSL, R. M. *Discrete and Combinatorial Mathematics.* MacMillan, 1987.

[99] HIQUEBRAN, D. T., ALFA, A. S., SHAPIRO, J. A., AND GITTOES, D. H. A revised simulated annealing and cluster-first route-second algorithm applied to the vehicle routing problem. *Engineering Optimization 22* (1994), 77–107.

[100] HÖCK, B. K. An examination of heuristic algorithms for the travelling salesman problem. Master's thesis, University of Cape Town, 1988.

[101] HOLLAND, B. Learning in the competitive environment of the prize collecting TSP. Unpublished BSc(Hons) project, Massey University, 1997.

[102] HOLLAND, J. H. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery 9* (1962), 297–314.

[103] HOLLAND, J. H. A mathematical framework for studying learning in classifier systems. *Physica D 22* (1986), 307–317.

[104] HOLLAND, J. H., HOLYOAK, K. J., NISBETT, R. E., AND THAGARD, P. R. *Induction: Process of Inference, Learning and Discovery.* MIT Press, Cambridge, MA, 1986.

[105] HOOKER, J. N. Testing heuristics: we have it all wrong. *Journal of Heuristics 1* (1995), 33–42.

[106] IBM. ACM chess challenge: Kasparov vs Deep Blue. `http://www.chess.ibm.park.org`, 1996.

[107] IBM. Kasparov vs Deep Blue: The Rematch. `http://www.chess.ibm.com`, 1997.

[108] JAILLET, P. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research 36*, 6 (1988), 929–936.

[109] JAILLET, P. Analysis of probabilistic combinatorial optimization problems in euclidean spaces. *Mathematics of Operations Research 18*, 1 (1993), 51–70.

[110] JAILLET, P., AND ODONI, A. The probabilistic vehicle routing problem. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988.

[111] JOHNSON, D. S., AND PAPADIMITRIOU, C. H. Performance guarantees for heuristics. In *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Eds. Wiley, New York, 1985, pp. 145–180.

[112] JOHNSTON, M. R. A game tree search-based heuristic strategy for the competitive prize collection problem. In *Proceedings of the 32nd Annual ORSNZ Conference* (1996), Operational Research Society of New Zealand, pp. 137–142.

[113] JOHNSTON, M. R., AND GIFFIN, J. W. Modelling a competitive prize-collecting TSP. In *Proceedings of the 30th Annual ORSNZ and 45th Annual NZSA Conference* (1994), Operational Research Society of New Zealand, pp. 172–177.

[114] JOHNSTON, M. R., AND GIFFIN, J. W. Exploring competitive prize collection. In *Proceedings of the 31st Annual ORSNZ Conference* (1995), Operational Research Society of New Zealand, pp. 5–12.

[115] JONES, C. V. The stability of solutions to the euclidean traveling salesman problem. Unpublished working paper, Simon Fraser University, 1992.

[116] JONES, C. V. *Visualization and Optimization.* Kluwer, Boston, 1994.

[117] JONES, C. V. Visualization and optimization. *ORSA Journal on Computing 6*, 3 (1994), 221–257.

[118] JONES, C. V. Sensitivity analysis for the euclidean traveling salesman problem. `http://weber.u.washington.edu/~cvj/tsp/tspnew.html`, 1997.

[119] JONES, C. V. The stability of solutions to the euclidean traveling salesman problem. Part I: Experimental results. Unpublished working paper, University of Washington, 1997.

[120] JONES, C. V. The stability of solutions to the euclidean traveling salesman problem. Part II: Theoretical results. Unpublished working paper, University of Washington, 1997.

[121] JONGENS, K., AND VOLGENANT, T. The symmetric clustered traveling salesman problem. *European Journal of Operational Research 19* (1985), 68–75.

[122] KAHNEMAN, D., SLOVIC, P., AND TVERSKY, A. *Judgement Under Uncertainty: Heuristics and Biases.* Cambridge University Press, 1982.

[123] KAMBHAMPATI, S., AND DAVIS, L. S. Multiresolution path planning for mobile robots. *IEEE Journal of Robotics and Automation 2*, 3 (1986), 135–145.

[124] KANTOR, M. G., AND ROSENWEIN, M. B. The orienteering problem with time windows. *Journal of the Operational Research Society 43*, 6 (1992), 629–635.

[125] KATAOKA, S., AND MORITO, S. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan 31*, 4 (1988), 515–530.

[126] KELLER, C. P. Algorithms to solve the orienteering problem: a comparison. *European Journal of Operational Research 41* (1989), 224–231.

[127] KELLER, C. P., AND GOODCHILD, M. F. The multiobjective vending problem: a generalization of the travelling salesman problem. *Environment and Planning B: Planning and Design 15* (1988), 447–460.

[128] KINDERVATER, G., LENSTRA, J. K., AND SAVELSBERGH, M. Sequential and parallel local search for the time-constrained traveling salesman problem. *Discrete Applied Mathematics 42* (1993), 211–225.

[129] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by simulated annealing. *Science 220* (1983), 671–680.

[130] KIRKWOOD, C. W. Does operations research address strategy? *Operations Research 38*, 5 (1990), 747–751.

[131] KNUTH, D. E., AND MOORE, R. W. An analysis of alpha-beta pruning. *Artificial Intelligence 6* (1975), 293–326.

[132] KRANTZ, M. Deeper in thought. *Time 149*, 10 (March 10, 1997), 54–55.

[133] LAPORTE, G. Location-routing problems. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 163–197.

[134] LAPORTE, G. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research 59* (1992), 231–247.

[135] LAPORTE, G. The vehicle routing problem: an overview of exact and approximate algorithms. *European Journal of Operational Research 59* (1992), 345–358.

[136] LAPORTE, G., ASEF-VAZIRI, A., AND SRISKANDARAJAH, C. Some applications of the generalized travelling salesman problem. *Journal of the Operational Research Society 47* (1996), 1461–1467.

[137] LAPORTE, G., LOUVEAUX, F., AND MERCURE, H. A priori optimization of the probabilistic traveling salesman problem. *Operations Research 42* (1994), 543–549.

[138] LAPORTE, G., AND MARTELLO, S. The selective travelling salesman problem. *Discrete Applied Mathematics 26* (1990), 193–207.

[139] LAPORTE, G., AND NOBERT, Y. Generalized travelling salesman problem through n sets of nodes: an integer programming approach. *INFOR 21* (1983), 61–75.

[140] LAPORTE, G., POTVIN, J.-Y., AND QUILLERET, F. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics 2* (1996), 187–200.

[141] LAWLER, E. L. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science 18* (1972), 401–405.

[142] LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., AND SHMOYS, D. B., Eds. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, New York, 1985.

[143] LEIFER, A. C., AND ROSENWEIN, M. B. Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research 73* (1994), 517–523.

[144] LENSTRA, J. K., AND RINOOY KAN, A. H. G. On general routing problems. *Networks 6* (1976), 273–280.

[145] LIN, W., AND KERNIGHAN, B. W. An effective heuristic algorithm for the traveling salesman problem. *Operations Research 21* (1973), 498–516.

[146] LIPMAN, B. L. How to decide how to decide how to ...: modeling limited rationality. *Econometrica 59*, 4 (1991), 1105–1125.

[147] LOKIN, F. C. J. Procedures for travelling salesman problems with additional constraints. *European Journal of Operational Research 3* (1978), 135–141.

[148] LOZANO-PÉREZ, T. Spatial planning: a configuration space approach. *IEEE Transactions on Computers 32* (1983), 108–120.

[149] LUCENA, A. Time-dependent traveling salesman problem: the deliveryman case. *Networks 20* (1990), 753–763.

[150] LUND, K., MADSEN, O. B. G., AND RYGAARD, J. M. Vehicle routing problems with varying degrees of dynamism. Tech. Rep. IMM-REP-1996-1, Institute of Mathematical Modelling, Technical University of Denmark, 1996.

[151] MALANDRAKI, C., AND DASKIN, M. S. Time dependent vehicle routing problems: formulation, properties and heuristic algorithms. *Transportation Science 26* (1992), 185–200.

[152] MALANDRAKI, C., AND DASKIN, M. S. The maximum benefit chinese postman problem and the maximum benefit travelling salesman problem. *European Journal of Operational Research 65* (1993), 218–234.

[153] MALANDRAKI, C., AND DIAL, R. B. A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. *European Journal of Operational Research 90* (1996), 45–55.

[154] MCGEOCH, C. C. Towards an experimental method for algorithm simulation. *INFORMS Journal on Computing 8*, 1 (1996), 1–15.

[155] MERISTÖ, T. Not forecasts but multiple scenarios when coping with uncertainties in the competitive environment. *European Journal of Operational Research 38* (1989), 350–357.

[156] MILLAR, H. H. Planning fish scouting activity in industrial fishing. *Fisheries Research 25* (1996), 63–75.

[157] MILLAR, H. H., AND KIRAGU, M. A time-based formulation and upper bounding scheme for the selective travelling salesperson problem. *Journal of the Operational Research Society 48* (1997), 511–518.

[158] MILLER, D. L., AND PEKNY, J. F. Results from a parallel branch and bound algorithm for the asymmetric traveling salesman problem. *Operations Research Letters 8* (1989), 129–135.

[159] MILLER, D. L., AND PEKNY, J. F. Exact solution of large asymmetric traveling salesman problems. *Science 251* (1991), 754–761.

[160] MITCHELL, J. S. B. An algorithmic approach to some problems in terrain navigation. *Artificial Intelligence 37* (1988), 171–201.

[161] MITTENTHAL, J., AND NOON, C. E. An insert/delete heuristic for the travelling salesman subset-tour problem with one additional constraint. *Journal of the Operational Research Society 43*, 3 (1992), 277–283.

[162] MORTON, T. E., AND PENTICO, D. W. *Heuristic scheduling systems: with applications to production systems and project management.* New York : Wiley, 1993.

[163] MOSHEIOV, G. The travelling salesman problem with pick-up and delivery. *European Journal of Operational Research 79* (1994), 299–310.

[164] MUTCHLER, D. The multi-player version of minimax displays game-tree pathology. *Artificial Intelligence 64* (1993), 323–336.

[165] MYERSON, R. B. *Game Theory: Analysis of Conflict.* Harvard University Press, Cambridge, Massachusetts, 1991.

[166] NEWELL, A., AND SIMON, H. A. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, N.J., 1972.

[167] NIJENHUIS, A., AND WILF, H. S. *Combinatorial Algorithms For Computers and Calculators*, 2nd ed. Academic Press, New York, 1978.

[168] NOON, C. E., MITTENTHAL, J., AND PILLAI, R. A TSSP+1 decomposition strategy for the vehicle routing problem. *European Journal of Operational Research 79* (1994), 524–536.

[169] ORLOFF, C. S. A fundamental problem in vehicle routing. *Networks 4* (1974), 35–64.

[170] O'ROURKE, J. *Computational Geometry in C.* Cambridge University Press, Cambridge, 1994.

[171] OSBORNE, M. J., AND RUBINSTEIN, A. *A Course in Game Theory.* MIT Press, Cambridge, Mass., 1994.

[172] OWEN, G. *Game Theory*, 3rd ed. Academic Press, 1995.

[173] PAPADIMITRIOU, C. H., AND STEIGLITZ, K. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Englewood Cliffs, 1982.

[174] PEARL, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, Mass, 1984.

[175] PEKNY, J. F., AND MILLER, D. L. An exact parallel algorithm for the resource constrained traveling salesman problem with application to scheduling with an aggregate deadline. Edrc 05-44-89, Engineering Design Research Center, Carnegie Mellon University, 1989.

[176] POTTERS, J. A. M., CURIEL, I. J., AND TIJS, S. H. Traveling salesman games. *Mathematical Programming 53* (1992), 199–211.

[177] POTVIN, J.-Y., AND GUERTIN, F. The clustered traveling salesman problem: a genetic approach. In *Meta-heuristics : Theory and Applications*, I. H. Osman and J. P. Kelly, Eds. Kluwer Academic, 1996, ch. 37, pp. 619–631.

[178] PSARAFTIS, H. N. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science 14*, 2 (1980), 130–154.

[179] PSARAFTIS, H. N. Dynamic vehicle routing problems. In *Vehicle Routing: Methods and Studies*, B. L. Golden and A. A. Assad, Eds. Elsevier Science, New York, 1988, pp. 223–248.

[180] PSARAFTIS, H. N. Dynamic vehicle routing: status and prospects. *Annals of Operations Research 61* (1995), 143–164.

[181] PUNNEN, A. P., AND ANEJA, Y. P. Randomized local search algorithms. Tech. rep., TR-93-005, University of New Brunswick, 1993.

[182] RAMESH, R., AND BROWN, K. M. An efficient four-phase heuristic for the generalized orienteering problem. *Computers and Operations Research 18*, 2 (1991), 151–165.

[183] RAMESH, R., YOON, Y.-S., AND KARWAN, M. H. An optimal algorithm for the orienteering tour problem. *ORSA Journal on Computing 4*, 2 (1992), 155–165.

[184] RAO, N. S. V. On performance of paths planning algorithms in unknown terrains. *ORSA Journal on Computing 4*, 2 (1992), 218–224.

[185] REEVES, C. R., Ed. *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific, Oxford, 1993.

[186] REIJNIERSE, J. H., AND POTTERS, J. A. M. Search games with immobile hider. *International Journal of Game Theory 21* (1993), 385–394.

[187] ROSENKRANTZ, D. J., STEARNS, R. E., AND LEWIS II, P. M. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing 6* (1977), 563–581.

[188] ROWE, N. C. Roads, rivers, and obstacles: optimal two-dimensional path planning around linear features for a mobile agent. *International Journal of Robotics Research 9*, 6 (1990), 67–74.

[189] SAVELSBERGH, M. W. P. *Computer Aided Routing.* Centrum voor Wiskunde en Informatica (CWI) Tract, 1992.

[190] SAVELSBERGH, M. W. P., AND SOL, M. The general pickup and delivery problem. *Transportation Science 29* (1995), 17–29.

[191] SCHOEMAKER, P. J. H. Strategy, complexity and economic rent. *Management Science 36*, 10 (1990), 1178–1192.

[192] SÉGUIN, R., POTVIN, J.-Y., GENDREAU, M., CRAINIC, T. G., AND MARCOTTE, P. Real-time decision problems: an operational research perspective. *Journal of the Operational Research Society 48* (1997), 162–174.

[193] SELIM, S. Z., AND ALSULTAN, K. A simulated annealing algorithm for the clustering problem. *Pattern Recognition 24*, 10 (1991), 1003–1008.

[194] SHAH, S. comp.ai.games frequently asked questions. `http://intranet.ca/~sshah/cagfaq.html`, 1996.

[195] SIMON, H. A., AND SCHAEFFER, J. The game of chess. In *Handbook of Game Theory with Economic Applications*, R. J. Aumann and S. Hart, Eds. Elsevier Science, New York, 1992, pp. 1–17.

[196] SOKKAPPA, P. R. *The Cost-Constrained Traveling Salesman Problem.* PhD thesis, Lawrence Livermore National Laboratory, University of California, 1990.

[197] TAYLOR, F. An investigation into the score orienteering problem. Unpublished BSc(Hons) project, Massey University, 1992.

[198] TELCIK, T. M. Vehicle navigation using a triangulated irregular network: solving the cross-country problem. Unpublished BSc(Hons) thesis, Curtin University of Technology, Nov. 1992.

[199] THOMAS, L. C., AND WASHBURN, A. R. Dynamic search games. *Operations Research 39*, 3 (1991), 415–422.

[200] TSILIGIRIDES, T. Heuristic methods applied to orienteering. *Journal of the Operations Research Society 35*, 9 (1984), 797–809.

[201] TVERSKY, A., AND KAHNEMAN, D. The framing of decisions and the rationality of choice. *Science 211* (1981), 453–458.

[202] VOLGENANT, T., AND JONKER, R. On some generalizations of the travelling salesman problem. *J. Opl Res. Soc 38*, 11 (1987), 1073–1079.

[203] VON NEUMANN, J., AND MORGENSTERN, O. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, 1944.

[204] WANG, Q., AND PARLAR, M. Static game theory models and their applications in management science. *European Journal of Operational Research 42* (1989), 1–21.

[205] WANG, Q., SUN, X., GOLDEN, B. L., AND JIA, J. Using artificial neural networks to solve the orienteering problem. *Annals of Operations Research 61* (1995), 111–120.

[206] WILLIAMSON, D. P. *On the Design of Approximation Algorithms for a Class of Graph Problems.* PhD thesis, Massachusetts Institute of Technology, 1993.

[207] WINSTON, W. L. *Operations Research: Applications and Algorithms*, 3rd ed. Duxbury Press, UNKNOWN, 1994.

[208] YAKOWITZ, S. A statistical foundation for machine learning with application to go-moku. *Computes Math. Applic. 17*, 7 (1989), 1095–1102.

Index of Problems