An Investigation of the Trading Agent Competition

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Science

In

Computer Science

at Massey University, Albany

New Zealand

Tong Liu

2005

Preface

This thesis is for my Master of Science study at the Department of Information and Mathematic Science, Massey University. I have faced many challenges to finish it.

Conquering the difficulties gave me great pleasure, for it pushed back boundary-lines and promoted my personal growth.

In the first place I would like to thank Dr. Chris Messom for his support in starting this research. He showed more patience and help with me than I could have ever wished for.

I want to thank Professor Robert McKibbin for encouraging me to finish my thesis.

Tong Liu

2005

Abstract

The Internet has swept over the whole world. It is influencing almost every aspect of society. The blooming of electronic commerce on the back of the Internet further increases globalisation and free trade. However, the Internet will never reach its full potential as a new electronic media or marketplace unless agents are developed. The trading Agent Competition (TAC), which simulates online auctions, was designed to create a standard problem in the complex domain of electronic marketplaces and to inspire researchers from all over the world to develop distinctive software agents to a common exercise. In this thesis, a detailed study of intelligent software agents and a comprehensive investigation of the Trading Agent Competition will be presented. The design of the RiskerWise agent and a fuzzy logic system predicting the bid increase of the hotel auction in the TAC game will be discussed in detail.

Table of Contents:

List of Tables:

List of Figures:

Chapter 1: Introduction

In this chapter, the motivation behind this research will be introduced, the definitions of relative terms will be provided and the literature review of TAC game and TAC agents will be given. The overview of following chapters will be listed in the end of this chapter.

1.1 Background

Because of division of labour, most of people couldn't be self-sufficient anymore. In order to survive, people need to exchange goods and/or services. A market is a mechanism which allows people to trade. The traditional market is where traders set up stalls and buyers look around the merchandise. Extending the concept of the traditional market, the modern shopping malls, shopping centres or shopping arcades are built. It is a building or set of buildings that contain many stores/shops, which is easy for people to walk from store to store.

With the terrific developments of communication and information technologies, the Internet is experiencing an exponential growth. In 2000, there were 304 million people having Internet access and ten million domain names were registered (Anderberg, 2003). The Internet Society data shows that there were approx 285,139,107 host computers on the Internet (Internet Domain Survey, 2004) and there were 46,067,743 web sites by the year 2003 (Zakon, 2004).

The Millions of people are assembled by the Internet. Internet has become a world-wide medium for collaboration and interaction between individuals without regard for geographic location. Markets do not have to always locate in a physical space. People can exchange goods and/or services on the Internet. The physical limitations of traditional auctions such as time, space and presence have disappeared.

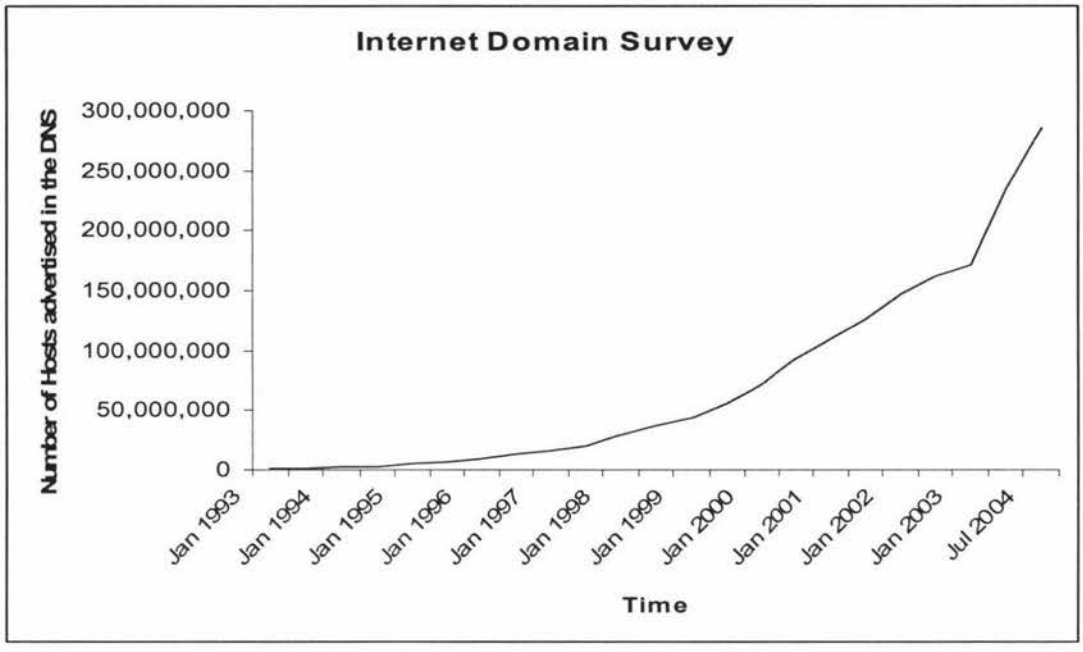


Figure 1: Number of Hosts advertised in the DNS (Internet Domain Survey, 2004)

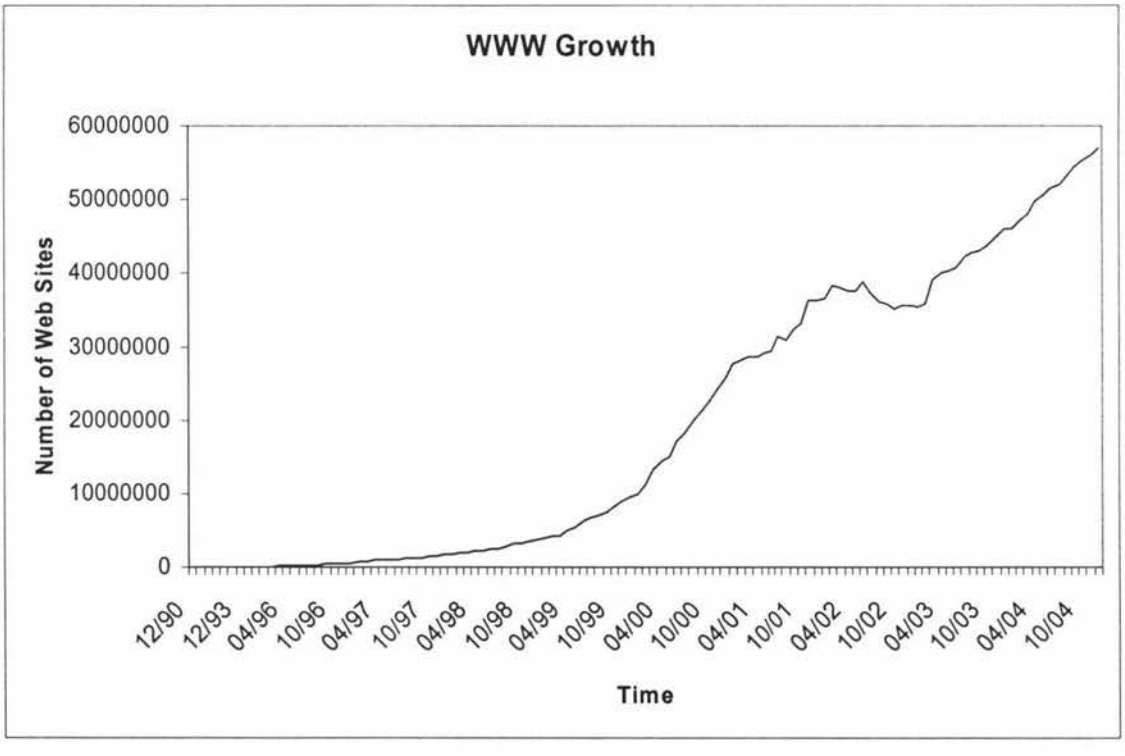


Figure 2: WWW Growth (Zakon, 2004)

The development and success of electronic commerce has dramatically increased the opportunities for automated trading agents because searching, accessing, filtering and integrating information is hard for a person or current computer systems especially when decisions are based on a massive amount of factors, using complex strategies. One

of the few studies comparing human and computer traders did not reflect very favourably on the humans (Das et al, 2001). Compared to human negotiation, automated trading agents can be faster, cheapter, more convenient (He 2004). Automated trading agents have the advantages of being able to work continuously and repetitively without losing concentration. Automated trading agents can also remove the human sensibilities that are often associated with bargaining (He 2004).

Once the agent is realized, many of the obstacles that currently limit how people use computers will disappear. It will make users' lives easier. People would like to delegate more functions to the agent (Negroponte, 1997).

There are many different trading methods. Auction is one of them. An auction is the process of buying and selling things by offering them up for bid, taking bids, and then selling the item. Internet auctions have become very popular. In the Internet Auction List there were more than 2500 auction company listings in 2003 (He 2004). eBay, an on-line auction, has 100 million registered users around the world (eBay, 2004).

In order to understand the effectiveness of agent strategies, the possible influences of automated traders to electronic markets and also to stimulate research in trading agents with an emphasis on developing a successful strategy for maximizing profits in a constrained environment, the Trading Agent Competition (TAC) game was designed (Strother 2000).

The TAC is a game simulating an electronic auction market. It was proposed by Wellman and Wurman. The first competition was held in July 2000 in Boston (Stone and Greenwald, 2001). TAC attracted 18 participants from six countries. Based on the success of the first event, the second competition was held in October 2001 in Tampa. The third TACs (Wellman et al., 2002; Greenwald, 2003), which were held in the following year had minor modifications. The fourth competition introduced new supply chain management research subject (Raghu et al., 2002).

To play TAC, software agents need to be designed. The goal of an agent is to satisfy its client. The agent will play the role of a travel agent with the goal of putting travel packages together for its clients. Each agent has eight clients who would like to take a

trip and also have their preferences for various aspects of the trip. The travel packages include airline tickets, reserve hotel rooms, and entertainment events. All these items are traded in different kind of on-line auctions.

The objective of this thesis is to investigate the TAC game and TAC agents. A trading agent is designed and implemented. A fuzzy logic module used to predict the price is also designed.

1.2 Definitions

1.2.1 Software agent

Before the agent definitions are given, one needs to be aware that there is not only one definition of agents. There are some widely accepted concepts characterizing agent systems and the definitions of agents with their own significances. They are described as follows:

- "The agent is an autonomous, self-contained, reactive, pro-active computer system with central locus of control that is able to communicate with other agents by an Agent Communication Language" (Wooldridge and Jeannins, 1994).
- Agent-Oriented Programming - An approach to building agents with mentality such as beliefs, desire and intentions (Franklin, 1996).
- An autonomous agent is a system which situated and is a part of the environment that senses that environment and acts in it to pursue its own agenda and to influence the future (Worldridge and Jeannins, 1995).
- An agent is an entity or object, which is able to execute symbolic external tasks, and reacts autonomously on the changes of its environment (Gadomski 1998).
- A Software agent is a computer program which functions as a user's personal assistant by performing tasks autonomously or semi autonomously. It is more than a passive task receiver and execution program (Harmon, 1995).

- The general functional definitions of software agent and intelligent agent are given below:

    A. "A software agent is a functional software module that is able to execute some predefined class of external tasks and has autonomy during these task realizations. It reacts on the predefined states of its own environment according to acquired information, its own built-in preferences and knowledge" (Gadomski 1998).
    B. "An intelligent agent is an agent with capability to change and evaluate its own preferences and knowledge" (Gadomski 1998). For example, it can learn or change goals if the original objectives are not reachable.

Researchers have offered a variety of agent definitions. The next section lists some of these definitions.

Virdhagriswaran, a researcher of MuBot Agent, an acronym for "Mobile Unstructured Business Object", defined that the agent has the ability for autonomous execution and domain oriented reasoning (Virdhagriswaran n.d.).

Russell and Norvig, researchers of the AIMA (Artificial Intelligence: a Modern Approach) Agent, stated that: "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors" (Russell and Norvig 1995).

Pattie Maes, researcher from MIT's Media Lab, described that Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed" (Maes 1995).

Smith, Cypher and Spohrer, researchers of the KidSim Agent defined an agent as a persistent software entity dedicated to a specific purpose. (Smith, Cypher and Spohrer 1994)

Hayes-Roth thought that the intelligent agents need continuously perform three functions: "perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions" (Hayes-Roth 1996).

Wooldridge and Jennings defined an agent as a hardware or (more usually) software-based computer system that has the following properties:

- Autonomy: an agent makes its own decisions about its actions and state rather than being influenced by others (Wooldridge and Jennings 1995)
- Social ability: an agent communicates with other agents or humans (Wooldridge and Jennings 1995)
- Reactivity: an agent perceives its environment and responds to it. It may make changes due to the environment (Wooldridge and Jennings 1995)
- Pro-activeness: an agent takes the initiative instead of only responding to its environment (Wooldridge and Jennings 1995)

Michael Coen, the researcher of the SodaBot Agent, defined: "Software agents are programs that engage in dialogs and negotiate and coordinate transfer of information" (Coen, 1995).

Brustoloni claimed that "Autonomous agents are systems capable of autonomous, purposeful action in the real world" (Brustoloni 1991).

Having these definitions given above, it is clear that there is no general definition for an agent. It could simply be described as a piece of software assisting users in the computers and computer networks.

1.2.2 General properties of an agent

The following sets are some agent attributes and properties. The list below is not complete and exhaustive.

- Reactivity: An agent has the ability to sense and act according to its environment.

- Knowledgeable: An agent has the ability to reason its goals; acquire knowledge and information from its environment.

- Inferential capability: An agent has the ability to make decision based on knowledge and information already have. It may choose best methods from itself, users, or other agents.

- Autonomous: An agent has the ability to independently act for its users. It is proactive not reactive.

- Adaptable: An agent has the ability to change its behaviour by learning, user preferences, or new capabilities.

- Collaborative: An agent has the ability to communicate, co-operate or collaborate with other agents in multi-agent societies.

- Communication ability: An agent has the ability to communicate with humans and other agents with suitable language.

- Mobile: An agent has the ability to move from one executing environment to another and continuing execution in a new environment.

- Persistent: An agent has the ability to keep its identity, knowledge and state over a long period of time even system failures.

- Personality: An agent has the human characters such as emotion, humour, etc.

1.2.3 Auction types

The following is a list of some basic types of auction:

- One-sided: a single seller accepts bids from multiple buyers or a single buyer accepts bids from multiple sellers.
- Two-sided or double auctions: multiple buyers and sellers to bid to trade goods.
- Continuous double auction (CDA): buyers and sellers match immediately on compatible bids.
- Sealed bid - No bids are visible to other bidders before auction closes. It clears only once and does not generate and publish price quotes.
- Open-outcry - Bids are made public at time of bidding.
- English (ascending) - Start at low price, increase amount until no further bidders, item goes to last bidder
- Dutch (descending) - Start at high price, decrease price til price accepted, e.g. tulips.
- First price - Pay amount of bid
- Second price - Pay amount of next highest bid
- Uniform-price auction: All the successful bidders pay the same price, which is decided by the auction.
- Common/Objective value - Item has identical value to all bidders, but each bidder has imprecise estimate.

- Private/Subjective Value - Each bidder knows and places different values on item according to the bidder's own information, but the bidder's value is private information to the bidder themselves.

Auctions can use any combination of the above types as long as they make sense. From the sellers' (auctioneer's) point of view, a good auction design gives the highest return to the seller.

1.3 Literature review

The TAC game and the strategies used in the TAC game will be introduced.

1.3.1 General information of TAC game

In each TAC game, eight trading agents compete for travel goods, with each agent representing eight clients. One customer can only have one agent. The duration of each game is 12 minutes.

Travel packages consist of the following (Game Overview, n.d.):

1. A round-trip flight,
2. A hotel reservation, and
3. Tickets to some of the following entertainment events
    - Alligator wrestling
    - Amusement park
    - Museum

Each customer will be given a set of preference for wishing to purchase travel arrangements. Preferences include the desired travel days; bonus for hotel quality; values for entertainment events.

A client's preference is characterized by

- Ideal arrival and departure dates.
- Bonus value for staying in the better hotel
- Bonus values for each of the three types of entertainment events

Agents must participate in auctions to try to acquire necessary resources. TAC has three different auction types bind together. Different auctions have different rules for matching the bids and recording the transactions. This increases the complexity of the TAC auction compare the real on-line auction. The three different kinds of auction in the TAC game are flight, hotel and entertainment auction. The flight auction is a continuously clearing one-sided auction with the changing price. The entertainment ticket auction is a standard continuous double auction. The most interesting auction is the hotel auction, which is a 16th price English ascending auction.

All of the auctions follow the high-level protocol below:

- Agents submit bids to the TAC server.
- The TAC server updates its price quote, publishing the current going prices.

Accepting bids, updating and publishing bids are most common tasks for an auction site server.

1.3.2 TAC auction rules

Different types of goods (flight tickets, hotel rooms and entertainment tickets) are traded at separate auctions with different rules. Agents can only buy air tickets and accommodation. Agents can buy and sell entertainment tickets. The following sections introduce the goods and auctions in the TAC market.

1.3.2.1 Flight auctions

1.3.2.1.1 Flight tickets

There is only one auction for each day and direction (arrival or departure). All the flight tickets are sold by the TAC server. There are two auctions in day 2, 3, 4. There is only one in flight auction in day 1 and out flight auction in day 5. (There will be no in flights on the last day, nor out flights on the first day.) There are 8 auctions in total.

Day 1	**Day 2**	**Day 3**	**Day 4**	**Day 5**
in flight	in flight **out flight**	in flight **out flight**	in flight **out flight**	 **out flight**

Table 1: The available flight tickets

The TAC server offers an infinite supply of flight tickets. The details of the TAC flight price are described as follows. The TAC server sells the flight tickets based on a stochastic function. The method used to update flight prices is a random walk method.

$$X(t)=10+(t/720)*(x-10)$$

where t is the number of seconds since the game starts and x is a random variable chosen from a uniform distribution on [10, 90] for each flight separately (Game Overview, n.d.).

In order to find the relationship between flight price and game time, an experiment was designed. After 100,000 experiments, it is found that the flight ticket prices increase linear over the time, and its difference is fairly large.

Figure 3 is based on the result of the 100,000 experiments. The price starts from $10. The C++ program designed for the experiment is in Appendix 1.

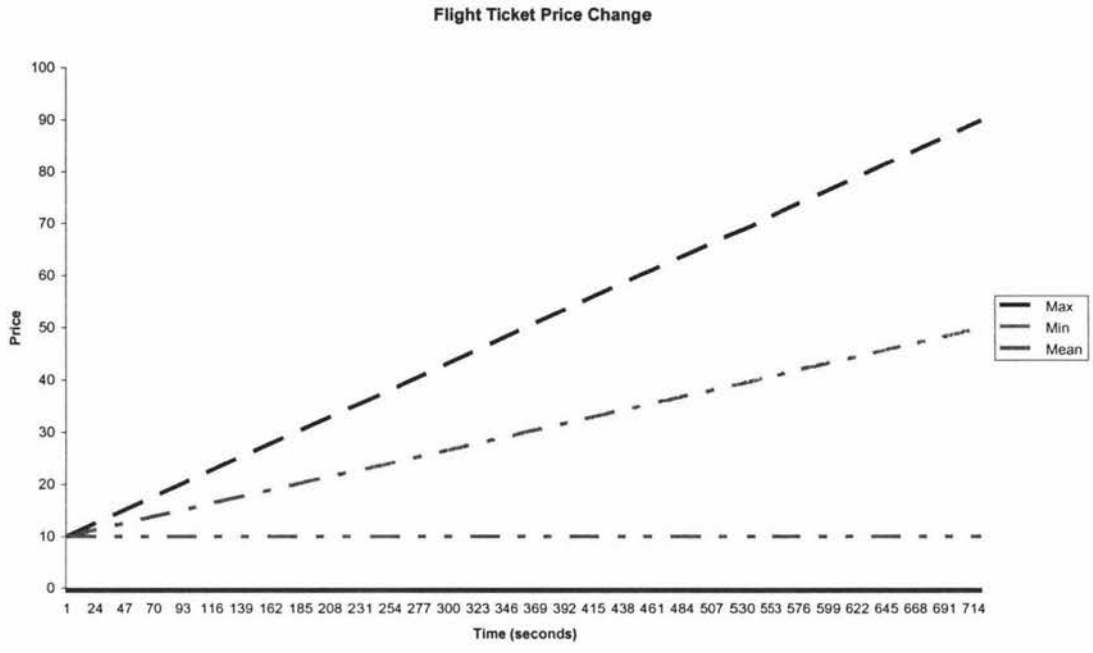


Figure 3: The flight prices are biased to drift upwards.

1.3.2.1.2 Flights auction

Flight auctions are continuously clearing one-sided auctions (TAC Server is a single seller, who accepts bids from multiple buyers), and clear continually (Once the action matches buyers and sellers, the transaction complete).

Agents can only buy the flight tickets but cannot sell or exchange their flight tickets. TAC server only accepts bids for buying flight tickets from agents.

If the price of buy bid point is equal to or higher than the current ask price, it will be matched immediately at the ask price. In other words, if the price of agent's buy bid is higher or equal to the TAC seller's sell price, the agent gets the ticket immediately and the agent has to pay for that.

If the price of buy bid point is less than the current ask price, it cannot be matched. The bid remains in the TAC auction as a standing bid. A standing buy bid remains in the TAC auction unless it can be matched by a sell bid which price drops to the same as or below the price of the standing buy bid. Because the flight price increases over the time, the chance for the standing buy bid to be matched is not good.

For example, TAC seller submits a sell bid of ((-50 450)) while an agent submits a bid of ((5 580) (6 385)). There are five units at $450 each would be matched. Since the whole bid could not match, the remaining part, ((6 385)), would remain in the auction.

1.3.2.2 Hotel auctions

1.3.2.2.1 Hotel rooms

There are only two hotels: one is Tampa Towers and another one is Shoreline Shanties. Clients must stay at least one night at one of the hotels. Tampa Towers hotel costs more compare with the Shoreline Shanties hotel. There are 16 rooms available per hotel per night. A client cannot change the hotels during the trip.

1.3.2.2.2 Hotel auction

Hotel auctions are Standard English ascending, multi-unit and sixteenth-price auctions (price increase, bidders pay amount of 16th highest bid), except that they all close at randomly determined times (Game Overview, n.d.). All of the bids whose price is higher than 16th price pay for the 16th price. Only the TAC servers can sell hotel rooms. There is no minimum bid price for either type of hotel. The TAC server sells 64 hotel rooms in total.

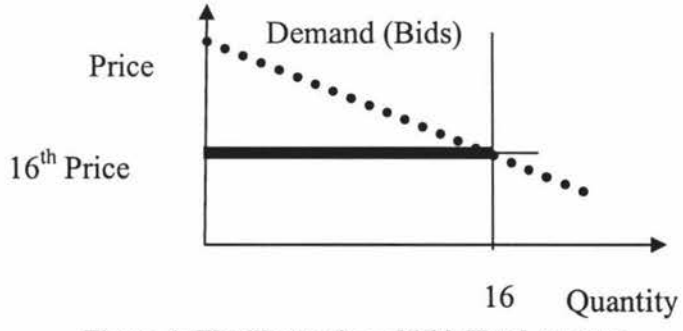


Figure 4: The illustration of 16th Hotel Auction.

Since clients only need hotels from the first day of their arrival and through the last day before their departure, there are no hotels available on the last day. There are 8 hotel auctions.

Day 1	Day 2	Day 3	Day 4	Day 5
TT Auction	TT Auction	TT Auction	TT Auction	
SS Auction	SS Auction	SS Auction	SS Auction	

Table 2: The hotel room auctions

Hotel auctions all close at randomly determined times. Specifically, one randomly chosen hotel auction will be closed at four minutes after the game starts. Then one randomly chosen hotel auction will be closed each one minute thereafter until 11 minutes when the last hotel auction is closed. The TAC server matches and clears hotel auction bids only once on the minute when it is closed (Game Overview, n.d.).

The agents don't know in advance when and which hotel auction will be closed and what the price of the hotel. TAC server only generates price quotes once per minute, on the minute when the hotel auction is closed. In most of the real on-line auctions, humans instead of software agents make decision. The uncertainty of the hotel auctions increases the difficulties for making decision. The type of TAC hotel auction could be adopted in the future as the software agent technology develops.

Agents can only submit buy bids instead of sell bids. The TAC seller submits sell bids to provide 16 rooms of two hotel types on each day for a minimum price above $0. The price quote is calculated as the 16^{th} highest price between all buy and sell bid units. Second-price auction is more common than 16^{th} auction. There is no big difference. To encourage bidders bid high is the same motivation behind 16^{th} auction and Second-price auction. The optimistic bidders would hope that the 16^{th} price or 2^{th} price will lower than the price their bid. Maskin and Riley (1999) show that "strong" buyers prefer the second-price auction.

Any new buy bid must satisfy the following conditions to be admitted to the auction: ASK be the current ask quote (16th highest price).

- It must offer to buy at least one room at a price greater than 16th highest price. It is not practical for a bid buying nothing. If the price is less than 16th highest price, the bid couldn't win anyway.

- If the agent has already submitted its bids to the hotel auction. It can not withdraw its bid. In the real on-line auction, withdrawing bid is also not encouraged. For example, it may be allowed to retract (cancel) bid in some cases if the retraction meets the requirements of eBay strict bid retraction policy.

- If the agent has a current buy bid, which could get m rooms in the current state, then the new bid must offer to buy at least m rooms a price greater than 16th highest price. This policy also encourage bidder bid high, which benefits the auctioneer.

When the TAC server clears and closes the hotel auction, the 16th highest price buy bids will be matched and the agents will pay their ask price for the hotel rooms.

For example, if the following hotel auction bids were submitted to TAC server:

If the following hotel auction bids were submitted to TAC server:

- Sell bid: ((-16 0)),
- Agent 1: ((3 3) (5 5) (9 8))
- Agent 2: ((4 1))

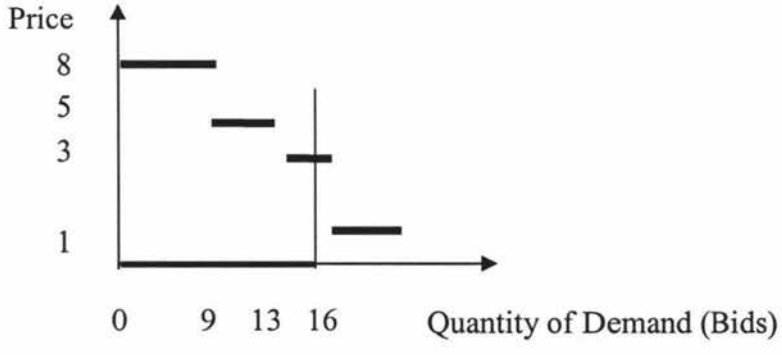


Figure 5: Example of Hotel Auction

In this example, if the TAC closed the hotel auction on the minute, Agent 1 would get 9 rooms at price 8, 5 rooms at price 5 and 2 rooms at price 3. Agent 1 asked for 17 rooms. It only got 16 rooms and Agent 2 did not get any rooms.

1.3.2.3 Entertainment ticket auctions

1.3.2.3.1 Entertainment tickets

All the travel agents receive allocation of entertainment tickets at the beginning of the game.

There are 8 entertainment tickets available for each entertainment type on each day in total. Each agent gets 12 entertainment tickets split as follows:

- On day 1 or day 4: One package of four of a specific entertainment type and one package of two of another different type.
- On day 2 or day 3: One package of four of a specific entertainment type and one package of two of another different type.

1.3.2.3.2 Entertainment auction

Agents buy or sell entertainment tickets through a continuous double auction (CDA), which all the agents can be buyers or sellers to bid to trade goods. Buyers and sellers match immediately for the compatible bids. There is one auction for each entertainment event on each day. Entertainment tickets are bought and sold on TAC auctions at prices the agents decide to bid.

- Tickets owned from the initial allocation are free.
- The score for selling entertainment tickets can be positive or negative because it is equal to the amount earned from selling entertainment tickets deduct the amount spent buying entertainment tickets.

Same as the hotel rooms, clients cannot use entertainment tickets on the day of departure (day 5).

Day 1	Day 2	Day 3	Day 4	Day 5
Alligator Wrestling	Alligator Wrestling	Alligator Wrestling	Alligator Wrestling	
Amusement Park	Amusement Park	Amusement Park	Amusement Park	
Museum	Museum	Museum	Museum	

Table 3: The available entertainment tickets auctions

Agents can submit bids with buy and/or sell points as long as a bid does not sell to itself. If the sell bid points prices are the same or below the price of the buy bid, the buy bid points will immediately match the lowest price sell bid points. Auctions clear continuously once the bids match. If the buy bid points prices are the same or above the price of the sell bid, the sell bid points will immediately match the highest price buy bid points. A bid point which does not completely match remains in the entertainment tickets auction.

Once new bids are submitted, price quotes are published immediately. The price quote includes the bid price and ask price. The price of the highest standing buy point will be the bid price. The price of the lowest standing sell point will be the ask price.

For example, if the standing bids in an entertainment ticket auction were:

- ((-2 101))
- ((-3 92) (-1 53))
- ((-5 63))
- ((2 44) (4 27))
- ((1 36))

Figure 6: Example of Entertainment Auction

- The bid price would become $44.
- The ask price would become $53.

1.3.3 TAC bid format and protocols

A bid represents an agent's willingness to sell and buy the goods in the auctions. A bid contains a bid string, which consists of a list of bid points in the following form:

"$((q_1\ p_1)\ (q_2\ p_2)\ (q_3\ p_3)\ \ldots\ (q_i\ p_j))$"

where q_i is a quantity of the goods an agent wants to buy or sell

p_i is a price of the goods an agent wants to buy or sell

If $q_i > 0$, it means that the agent wants to buy q_i amount of the good at the auction for less than or equal to p_i price per unit of that goods.

If $q_i < 0$, it means that the agent wants to sell q_i amount of the good at the auction for greater than or equal to p_j price per unit of the good.

The prices should always be nonnegative. None of the agents would like to sell goods and pay the money. If prices were negative, agents could buy goods and get money.

It is not acceptable in TAC for an agent submitting a bid to sell goods to itself. For example if an agent placed the bid "((-2 5) (2 10))", it could likely sell 2 units to itself at a price between $5 and $10. Agents sell goods to themselves would not get any benefit. It is waste of resources and time. It is sensible for an agent want to sell goods to itself. This TAC rule just prevents the agent make careless mistakes.

If a bid is matched at the TAC server, the bid string will change after the match. For example, if an agent submits the bid "((-3 5) (-4 30))" and sells two units of the good for $5, then the bid string becomes "((-1 5) (-4 30))". The remains of bid string stay in the server waiting to be match.

1.3.4 Final score of an TAC agent

The TAC server computes and reports each agent's optimal allocation. It calculates the score for all the agents at the end of each TAC game.

The final score of an agent is composed of:

+ The value of the allocation of the goods to clients,
- The penalty for changing clients' preference,
- The cost of buying flight tickets, hotel rooms and entertainment tickets,
- The penalty for negative entertainment balances

1.3.5 Characteristics of TAC game

- First, there are contests between agents. For exmple, the hotel rooms are finite and the price is unpredictable and the order of auctions closing was unknown and unpredictable).

- Second, there are interdependencies (He and Jennings, 2004). These are relations between different types of auctions (e.g. flight tickets will be wasted if the same hotel rooms are not available from arrival day to the day before departure); between different dates in the same type of auction (e.g. customers must stay in the same hotel during their trip. Customers cannot get extra utility for attending the same type of entertainment more than once during their trip); between the same kind counterpart auctions in the same day (e.g. if the price of good hotel is high, the customer can change to cheap hotel at the same day).

- Third, the bidding involves uncertainty (He and Jennings, 2004). For example, flight ticket prices start and change randomly; one randomly chosen hotel room auction closes from the 4^{th} to 11^{th} minutes after game starts; the customers' preferences are assigned randomly and the way players bid for their hotel rooms is unpredictable if they are new players.

- Fourth, a trade-off exists (He and Jennings, 2004). For instance, the prices of flight tickets in flight ticket auctions rise over the time as shown in the figure 3. But, if the agent buys cheap flight tickets very early, it might not be able to buy the necessary hotel rooms. This leads to some invalid travel packages. The flight tickets might be wasted. Hence, a trade-off exists between buying flights tickets earlier at lower prices and buying them later at higher prices to make sure they match with the hotel rooms that have been bought.

1.3.6 Strategies of TAC agents

The high-level bidding decisions of most previous games had the following structure:

- Analyse the game environment and history data
- Decide at what time to bid
- Predict the prices
- Decide on what goods to bid for

1.3.6.1 Analysis of competition environments

The success of an agent not only depends on its own strategies, but also depends on the strategies of the other competitors. The best solution is relative to other players' strategy (Vetsikas and Selman, 2003). Peter Stone stated:

> The success of agent strategies depends a great deal on the strategies of the other competition (Stone, 2002).

In both TAC-00 and TAC-01, the competitors learned about each other's strategies and made many adjustments. In TAC-00, only 14% of the agents were using a particularly effective (in isolation) high-bidding strategy during the qualifying round; by the finals 58% of the agents were using this strategy (Stone, 2002).

Before each game, ATTac downloads a published list of agents from the TAC website to identify known high-bidders. If there are more than two, it factors the information into bidding strategy.

The agent SouthamptonTAC designed by He, Minghua and Jennings, Nick observed the TAC game market environment and categorized the TAC game market environment into three kinds of environments according to the different risk attitudes of other agents and decided different strategies correspond to different environments (He and Jennings, 2004).

- Non-competitive environment: there is no price war in the hotel room auctions. Agents can easily obtain the hotel rooms at very low prices. In this environment, the agent doesn't change the travel plan for each client. It buys almost all the flight tickets at the beginning of the game and all the rest of the tickets at the end of 4th minute (He and Jennings, 2004).

- Semi-competitive environment: the competition of most hotel room auctions is reasonable. The hotel room prices are moderate e.g. the price of a good hotel room is 120 and the price of a cheap hotel room is 60. In this environment, the agent predicts the closing prices of the hotels and changes the travel plans for its

clients if new plans could make the clients' utilities higher enough. It buys almost all the flight tickets at the beginning of the game and all the rest tickets at the end of 4th minute (He and Jennings, 2004).

- Competitive environment: there are price wars in the hotel room auctions. The prices of some hotel rooms are very high, e.g. the price of a good hotel room is 300. In this environment, the agent uses the fuzzy reasoning methods to predict the hotel room closing prices and bid adaptively. It buys most flight tickets at the beginning of the game and the rest of the flight tickets based on the flight category (He and Jennings, 2004).

The environments are decided by the past games history before a game starts. In the TAC semi-final, general seeding round data was used to predict the environment because there was not enough past data.

After the 4th minutes of the game start, one of the hotel room auctions will be closed. The agent then can use the current hotel prices to decide if it needs to change its strategy. For example, the agent may change its strategy from Non-competitive environment to Competitive environment.

1.3.6.2 Time to bid flight tickets: early bird and deliberate buyer

The flight price in general is going up all the time as shown in figure 3. There is a dilemma, which is bidding *early* could get the cheap tickets but may waste the tickets (travel plan changes); or bidding *late* the agent can pay a high price for the tickets after the hotel auctions (accommodation secured not to waste the tickets).

Bidding for all the flight tickets late is not very wise strategy (the price difference may be over $650). Vetsikas and Selman did experiment on the performance of different bidding times:

- Late Bidder: Buy at the beginning only tickets that are "certain" to be used
- Early Bidder: Buy all tickets at the beginning

High aggressive bidder bids for all rooms progressively closer to the marginal utility. Medium bidders bid for critical rooms close to marginal utility and the rest of the rooms an increment above the current price.

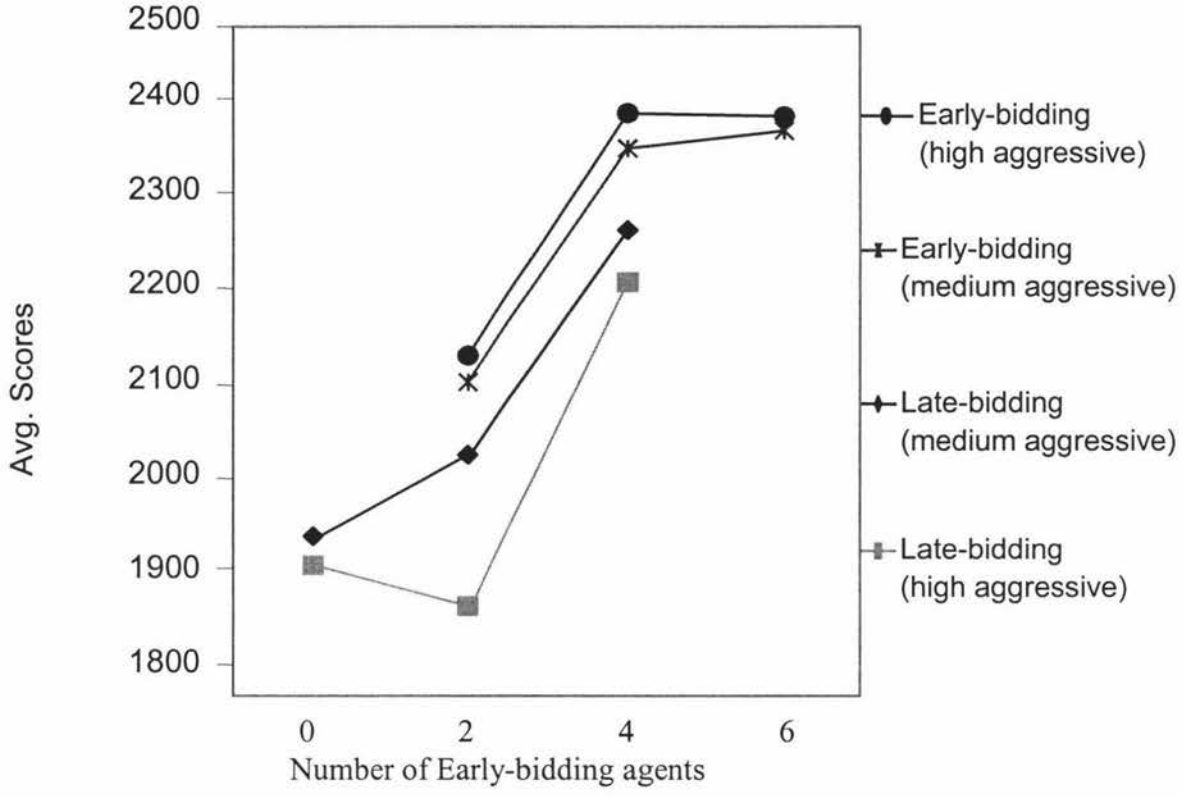


Figure 7: The performance of agents with four different strategies against different number of early-bidding agents (Vetsikas and Selman, 2003)

Figure 7 shows that Early-bidding is better than late bidding.

In TAC'01, there are two main candidate heuristics – early bird and deliberate buyer (they are also called "open-loop" and "closed-loop" by Stone et al. 2003 respectively). A trading agent using the early bird heuristic makes decisions at the very beginning of game and does not change them. This early bird heuristic was recognized as the reason to the success of LivingAgent (Fritschi and Dorer, 2002). LivingAgent was the winner of TAC'01. The early bird heuristic is based on perfect prediction assumption, which means that an agent could accurately predict the clearing prices for the auctions at the beginning of a game. This assumption is supposed to guarantee the optimality of static resource allocation (Ding *et al.* 2003).

The open-loop strategy has the advantage of buying a minimal set of goods. That is, it never buys more than it can use. On the other hand, it is susceptible to unexpected prices in that it can get stuck paying arbitrarily high prices for the hotel rooms it has decided to buy. In particular, if all eight agents are open loop and place very high bids for the goods they want, many of the prices will skyrocket, eliminating any potential profit. Thus, a set of open-loop agents would tend to get negative scores (Stone, Schapire, Littman, and McAllester, 2003).

The agent like ATTac used a deliberate buyer bidding decision based on a cost-benefit analysis: ATTac analyses the costs of postponing bids on auctions, if the cost exceeds the benefit of winning that good under multiple scenarios, then decide to bid or not.

Theoretically, the deliberate buyer should have a better performance compared to the early bird. But practically, its performance is very sensitive to its implementation. The difference of hotel room auction clearing prices and the ensemble of game participants also affect the performance of agents (Stone *et al.* 2002). Some experiments on open-loop vs. closed-loop (deliberate buyer and early bird) strategies were completed. The results are shown in the table 4.

Agent	Score	Utility
EarlyBidder	2869 ± 69	10079 ± 55
ATTac-2001(2)	2614 ± 38	9671 ± 32
ATTac-2001(3)	2570 ± 39	9641 ± 32
ATTac-2001(4)	2494 ± 68	9613 ± 55

Table 4: The results of one EarlyBidder against three different versions of deliberate buyer over 197 games (Stone, *et al*, 2003)

According to Stone, Schapire, Littman, and McAllester, the results in the table 4 suggest that the variation of the closing prices is the major decisive factor between the effectiveness of the open-loop and closed-loop strategies. They think that the closed-loop strategy could do better in large price variance situation, while the open-loop strategy should do better in the small price variance situation (Stone, *et al*, 2003).

In TAC'02, the winners used both heuristics. "the most successful agents were primarily heuristic-based and domain-specific" (Greenwald, 2003). The originally NP-complete optimisation problem became more tractable when the domain-specific heuristics are used.

- Compose travel plans earlier: buy most flight tickets earlier but postpone purchasing "risky" flight tickets to allow change resource allocation later (e.g. ATTAC (Stone *et al.*, 2002) and Whitebear (Vetsikas and Selman, 2003));

- Change among different heuristics strategies according to the prediction of competitiveness environment of the TAC game (e.g. SouthamptonTAC (He and Jennings, 2004));

- Use early bird heuristic in the hotel and flight auctions, and bidder heuristic in the entertainment auctions (e.g. UMBCTAC (Ding *et al.*, 2002)).

The success of these strategies is based not only by the ability of predicting accurately, but also by the ability to avoid and handle risk, especially avoid buying hotel rooms at a very high price.

Vetsikas and Selman (2003) did experiment on six different bidding strategies: high aggressive late Bidder; median aggressive late Bidder; high aggressive early Bidder; median aggressive early Bidder; high aggressive strategic Bidder; median aggressive strategic Bidder.

Bidding Strategies for Hotels:

- Low aggressiveness: Bids higher than the current ask price by an increment.

- High aggressiveness: Bids for all rooms progressively closer to the marginal utility.

- Medium aggressiveness: Combines two previous strategies

- ✓ For critical rooms (rooms with high marginal utility) the bid is close to the marginal utility

- ✓ For all other rooms it bids an increment above the current price (the increment increases as time passes)

Bidding time for Plane Tickets

- ➢ Late Bidder: (boundary str.): Buy at the beginning only tickets that are "certain" to be used

- ➢ Early Bidder: (boundary str.): Buy all tickets at the beginning

- ➢ Strategic Bidder: (intermediate str.)

  - ✓ Uses "Strategic Demand Reduction"
  - ✓ Buy all tickets at the beginning, except the ones that are "highly likely not to be used"

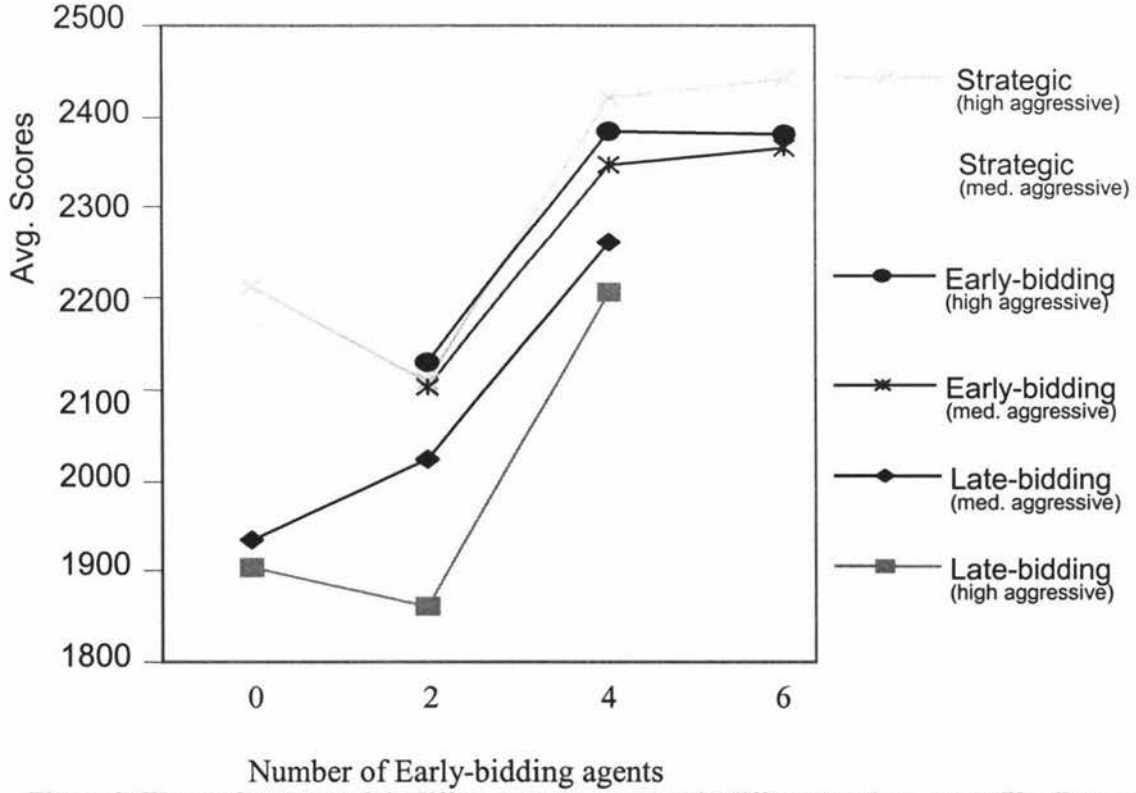


Figure 8: The performance of six different strategy agents in different environments (Vetsikas and Selman, 2003).

Figure 8 shows that the strategically bidding agents perform best overall.

1.3.6.3 Analysis of the hotel combinations

The success of an agent not only depends on its own strategies, but also depends on the preferences assigned by the server. There are 20 possible hotel combinations for a customer. Table 5 lists the 20 possible travel schedules.

Number	AD	DD	Hotel		Number	AD	DD	Hotel
1	1	2	SS		11	1	2	TT
2	2	3	SS		12	2	3	TT
3	3	4	SS		13	3	4	TT
4	4	5	SS		14	4	5	TT
5	1	3	SS		15	1	3	TT
6	2	4	SS		16	2	4	TT
7	3	5	SS		17	3	5	TT
8	1	4	SS		18	1	4	TT
9	2	5	SS		19	2	5	TT
10	1	5	SS		20	1	5	TT

Table 5: A customer's possible travel schedules

AD means Arrival Day, DD represents Departure Day. The number in AD, DD column corresponds to a weekday, e.g. 1 means Day 1. In the hotel column, TT means Tampa Towers and SS means Shoreline Shanties.

Researchers from University of Maryland computed the estimated price for each hotel combination. Hotel price was based on 1000 games in the 2002 seeding round. The mean is denoted by a solid line, and the median is denoted by a circle (Ding, *et al.* 2003).

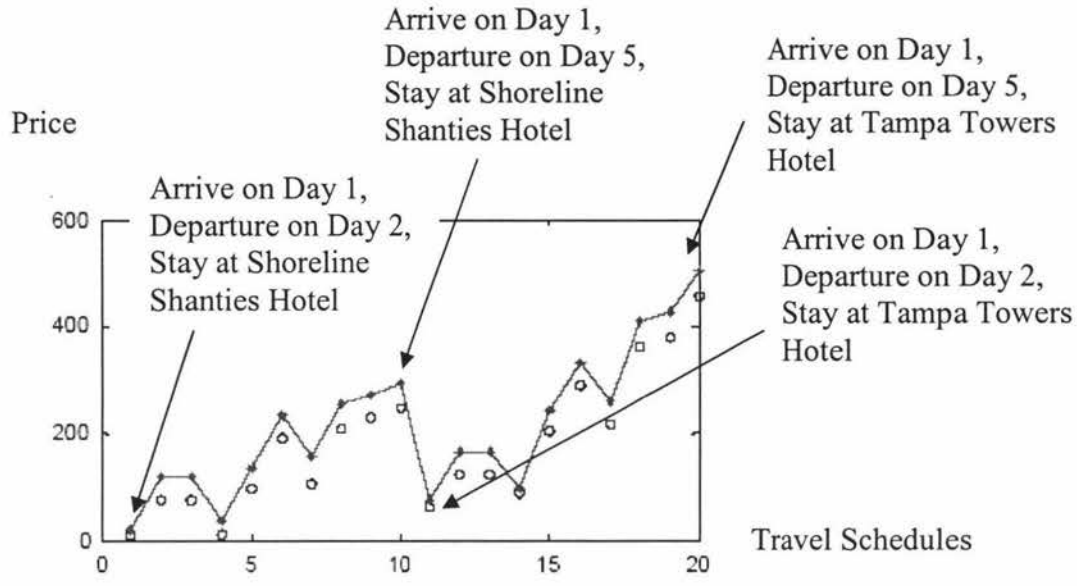


Figure 9: The estimated price for each hotel combination (Ding, *et al.* 2003)

For the same arrival day and departure day, it costs more for the agents to satisfy the clients who want to stay at Tampa Towers hotel than to satisfy the clients who want to stay at Shoreline Shanties hotel.

For both Shoreline Shanties hotel and Tampa Towers hotel, it costs less for the agents to satisfy the clients whose arrival day is day 1 and departure day is day 2 than to satisfy the clients whose arrival day is day 1 and departure day is day 5.

They found that (Ding, *et al.* 2003):

- Shorter hotel combinations cost less
- Short trips will always have better performance
- Shoreline Shanties hotels cost less

1.3.6.4 Price prediction

Hotel auctions are important in securing feasible travel packages and the most contested items during the TAC competition. Because of the random nature of the customers'

preferences and the way other agents deal with their hotel bidding, there are risks and uncertainty associated with the hotel auctions.
However, general price trends cannot be captured completely because they depend on the identity of the participating agents. The uncertainty of hotel price significantly influence the relative cost of assembling trips for clients.

There are several approaches predicting hotel clearing price (Wellman *et al.* 2004):

- Use the current price quote
- Adjust the current price quote by the difference between clearing price and the price at current time
- Predict by fitting a curve to the price points seen in the current game.
- Predict based on closing price data for that hotel in the past games or also used extrapolation from current prices.
- Similar approach as above, but condition on hotel closing time, awareness that the closing sequence will influence relative prices.
- Similar approach as above, but condition on full ordering of hotel closings, or what hotels are open or close at a particular point.
- Learn a mapping of the features of the current game to closing prices based on historic data.
- Use Fuzzy logic rules based on observation about associations between abstract features.
- Use a moving average technique

The team from the Artificial Intelligence laboratory, University of Michigan surveyed the price prediction approaches employed in TAC-02 game. Based on the survey responses and other strategies used in previous game, prediction techniques used in the TAC includes (Wellman, et al, 2004):

- Historical Averaging: Agent harami; agent UMBCTAC; Agent SouthamptonTAC (He & Jennings, 2004); Agent ROXYBOT (Greenwald, 2002); Agent cuhk (Wellman *et al.*, 2002)

- Machine Learning: Agent ATTac (Stone et al., 2003), Agent kavayaH (Putchala et al., 2002)

- Competitive Analysis: Agent Walverine (Cheng et al., 2005) (Greenwald, 2003)

- Fuzzy reasoning techniques: Agent SouthamptonTAC (He and Jennings, 2004)

- A Partially Observable Markov Decision Process Approach: Agent TOMAHACK (Braziunas, *et al,* 2002)

- Branch-and-Bound Optimization: Agent SICS (Boyan and Greenwald 2001)

1.3.6.5 What price to bid?

In the first price auction, most of the buyers would like to bid low or the actual price because the winner pays her/his winning bid price. But in the second-price auction, the buyer may take the risk to bid high because the winner only pays the second highest price. What is the best strategy: to bid high or low or actual price in the second-price auction? Paup Klemperer states:

> A little reflection shows that in a second-price sealed-bid private-values auction it is optimal for a player to bid her/his true value, whatever other players do. In other words "truth telling" is a dominant strategy equilibrium (and so also a

Nash equilibrium), so here, too, the person with the highest value will win at a price equal to the value of the second-highest bidder (Klemperer, 1999).

This is also applied to the TAC auctions. Each bidder places their own private values on the TAC items such as hotel rooms. The hotel auction is a 16th English ascending auction, which means the agent with the highest value will win at a price equal to the value of the 16th highest bidder.

Assume that Bidder1 and Bidder2 have the following price arrangement:

- Bidder1 high price: $ 600
- Bidder1 actual price: $ 200
- Bidder1 low price: $ 100
- Bidder2 high price: $ 500
- Bidder2 actual price: $ 150
- Bidder2 low price: $ 50

Assume that Bidder2 holds the 16th highest price in the 16th price auction. Bidder1 enters the auction with different prices. Table 6 shows a simple example of how the equilibrium occurs. The description is from the Bidder1's point of view.

Bidder1 / **Bidder2**	**Bid high Price (b1) $600**	**Bid Actual Price (b1) $200**	**Bid Low Price (b1) $100**
Bid high Price (b2) $500	Win but lose. Have to pay high price $500. (Unhappy)	**Happy** to give up	Lose but **happy**
Bid Actual Price (b2) $150	Win with profit $50 **(Happy)**	Win with profit $50 **(Happy)**	Lose (unhappy)
Bid Low Price (b2) $50	Win with profit $550 **(Happy)**	**Win with profit $150 (Happy)**	Win with profit $150 **(Happy)**

Table 6: The bidder price matrix 1

This example shows that if the bidder1 bids the true value, it never loses. It only pays her/his acceptable price or makes profit when she wins.

This time, assume that Bidder1 holds the 16th highest price in the 16th price auction. Bidder2 enter the auction with different prices. The description is from the Bidder2's point of view.

Bidder1 / Bidder2	**Bid high Price (b1) $600**	**Bid Actual Price (b1) $200**	**Bid Low Price (b1) $100**
Bid high Price (b2) $500	Lose but **happy**. Don't have to pay high price $600.	Win but lose $50 (Unhappy)	Win with $50 profit **happy**
Bid Actual Price (b2) $150	Lose but **happy**. Don't have to pay high price $600.	Lose but **happy**. Don't have to pay price $200.	Win with $50 profit **happy**
Bid Low Price (b2) $50	Lose but **happy**. Don't have to pay high price $600.	Lose but **happy**. Don't have to pay price $200.	Lose but unhappy. Could have $50 profit.

Table 7: The bidder price matrix 2

This example shows that if the bidder2 bids the true value, it never loses. It only pays her/his acceptable price or makes profit when she wins.
Vetsikas and Selman talked about the bid price dilemma in their presentation (Vetsikas and Selman, 2003):

- If not aggressive, could get outbid and lose rooms needed.
- If too aggressive, prices will skyrocket and the agent's score will be reduced.

Peter R. Wurman also pointed out that:

> "Truth telling is a dominant strategy because it is optimal regardless of the other agents' strategies" (Wurman *et al.*, 1998).

If all the agents use the same strategy, for example all bids low price or all bids true value. Which strategy is better? Wellman and other researchers did experiments on those two different bid strategies: shading (an agent bids at a lower price than its marginal values), non-shading (agents bid true marginal values). They found that if the agents bid their true values this would improve social welfare, but sacrifice individual

profits. The average client-adjusted payoffs for all shading, all non-shading are 3339 and 3155, respectively. The corresponding market efficiencies are 88.5% and 89.4% (Wellman, *et al.*, 2003).

If the agent knows the other agents' strategies, there is another tactic that could be applied, which is "against the tide". There is an interesting observation of going "against the tide". Vetsikas and Selman stated:

> In general an agent wins when the agent is going against the tide. i.g. being aggressive when most other agents are not (Vetsikas and Selman, 2003).

Experiments made by ATTac show that: When one ATTac played with seven Early Bidders, ATTac is against the tide. ATTac won.

Agent	Score	Utility
ATTac	2431 ± 464	8909 ± 264
EarlyBidder	-4880 ± 337	9870 ± 34

Table 8: One ATTac played with seven Early Bidders (Stone, *et al.*, 2002).

When seven ATTac played with one Early Bidder, Early Bidder is against the tide. Early Bidder won.

Agent	Score	Utility
ATTac	2578 ± 25	9650 ± 21
EarlyBidder	2869 ± 69	10079 ± 55

Table 9: Seven ATTac played with one Early Bidder (Stone, *et al.*, 2002).

If in the unknown situation, the best strategy is to bid at true valuation.

1.3.7.6 Completion problem

Given the current holdings, and given (expected) prices, what goods should be chosen to buy or sell at these prices?

There are two general approaches in TAC-01:

- Global optimisation: Agents like whitebear solved the completion problem using global optimisation techniques used by TAC-00 agents, including integer linear programming (Stone et al. 2001) and heuristic search (Greenwald & Boyan 2001).

- Local optimisation: Agents like TacsMan constructed travel packages by optimising utility client-by-client (Porter, *et al,* n.d.).

Local optimisation completion strategy is a kind of greedy strategy for allocation. It is computationally feasible to quickly determine the maximum utility achievable by one client given a set of purchased goods, move on to another client with the remaining goods, etc. However, this strategy can lead to sub-optimal solutions.

A different approach is a heuristic approach that implements the greedy strategy over a number of random client orderings and chooses the most profitable resulting allocation. Empirically, the resulting allocation is often optimal. ATTac chose 100 random client orderings to implement the heuristic approach. In a set of seven games from just before the tournament, ATTac's greedy allocator was run approximately 600 times and produced allocations that averaged 99.5% of the optimal value (Stone, *et al.*, 2001).

1.3.6.6.1 Linear Programming Approach

In the competition, Agent ATTac implemented integer linear programming approach, an allocation strategy to find the optimal allocation of goods (Stone, *et al.*, 2001). The integer linear programming approach works by defining a set of variables, constraints on these variables, and an objective function. Agent ATTac defined 272 variables and 188 constraints (Stone, *et al.*, 2001).

1.3.6.6.2 A Genetic Algorithm-Based Optimisation Technique

Agent PaininNEC used a combination of heuristics, including a genetic algorithm-based optimization technique to find the optimal requirements of hotel rooms (Greenwald, 2003). Genetic Algorithms are useful and efficient when the search space is large, complex or poorly understood, the domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space. The advantages of the GA approach are intrinsically parallel; able to manipulate many parameters simultaneously and handle arbitrary kinds of constraints and objectives. Their major disadvantage is that they are relatively slow, being very computationally intensive compared to other methods, such as random optimization.

1.3.6.6.3 Generate domain-specific heuristics approach

Agent WHITEBEAR generates domain-specific heuristics and experiment with these heuristics to determine which are the most effective. Sometimes, the most effective heuristic is a combination of two or more heuristics (Vetsikas and Selman 2002).

1.4 Summary

In this chapter, the motivation behind this research was explained. The definitions of relative terms including softare agent were provided and the literature review of TAC game and TAC agents were given.

The purpose of this study is to investigate the TAC game and the TAC agents. As shown in the literature review, there has no article evaluating the TAC game from auctioneer's point of view. There is no article listing the changes of TAC game. There is no article describe about how to play against 7 aggressive open loop agents in TAC game. This thesis will address these issues. A fuzzy logic model for predicting the TAC hotel price change will be designed. A short overview of following sections in the thesis is presented below.

Chapter 2: Is TAC game a good design?

In this chapter, the design of the TAC game will be evaluated from an auctioneer's point of view. The changes of TAC game will also be listed.

Chapter 3: The RiskerWise agent

In this chapter, the design and performance of the RiskerWise agent playing against seven open-loop EarlyBidder DummyAgents will be presented in detail. And the relationships among score, utility and cost will be discussed.

Chapter 4: Using Fuzzy Logic to Predict the Hotel Price Increase

In this chapter, basic concepts of Fuzzy Logic are introduced and the design of fuzzy logic system for predicting hotel auction price change in TAC is explained.

Chapter 5: Conclusion and future development

This Chapter gives conclusions of the thesis and future development of the Fuzzy Logic hotel price change prediction System.

Chapter 2: Is TAC game a good design?

In this chapter, the changes of TAC game will be listed and the design of the TAC game will be evaluated from an auctioneer's point of view.

2.1 Changes of the TAC game

The TAC game started in 2000. There have been some changes since the first game. The main idea and overall concept of the current TAC Game remain the same as the previous ones. But there were noticeable changes.

- Change in TAC game time: The time of the TAC game decreased from 15 minutes to 12.

- Change in TAC game allocation: TAC server does the allocation for all the agents after the game finishes at the current game. In previous game, each agent needs to present final allocation of own goods for clients to a server at the very end of the game.

- Changes in flight auctions: In previous games airline prices periodically increase or decrease by a random amount chosen from the set {– 10, –9, ..., 9, 10} with equal probability the expected change in price for each airline auction is 0. And the maximum price that flight auctions might have reached in previous games for one ticket is 600USD and in current games it could go up to 800USD.

- Changes in hotel auctions: In previous games hotel auctions used to close only in the end of each game's instance, although the auction is subject to close early after random periods of inactivity. That means all the agents didn't know if they would hold needed tickets or not almost till the end of the game, contrary to the current game, when each auction closes randomly every next minute starting from the 4th and after 11 minutes all agents have a clear view of all their holdings.

- Changes in entertainment auctions: In previous games there were different endowments of the entertainment tickets. In contrary to current games when each agent receives 12 tickets arranged in bundles, in the previous games agents might receive an initial endowment of tickets of each event on each night: zero with probability of ¼, one with probability of ½ and two with probability ¼.

- Changes of server: The previous two trading agent competitions, TAC-00 and TAC-01, were played on the Michigan University Internet AuctionBot platform, a configurable auction server available on the Internet. Since 2002, the Trading Agent Competition game and auction servers have been running at Swedish Institute of Computer Science (SICS) in Kista, Sweden. SICS developed an open-source TAC server software, which is written in Java and SICStus Prolog. TAC server consists of Game Server and Information Server. The Game Server runs the actual games and communicates with trading agents. The Information Server manages participants, schedules games, and collects and distributes game information and statistics via the Web and TAC game monitoring applets.

- Changes of hotel and entertainment names: In TAC – 2000, there are two types of hotel rooms – the Boston Grand Hotel (BGH) and Le Fleabag Inn (LFI). The Boston Grand Hotel (BGH) is a better hotel. Since 2001, the hotel names changed to Tampa Towers (TT) and Shoreline Shanties (SS). Tampa Towers is a nicer place to stay and cost more.

- Changes of entertainments' names: In TAC – 2000, there are three types of entertainment: Symphony, theatre, and baseball. Since 2001, the entertainment events changed to: Alligator wrestling, Amusement park and Museum.

2.2 TAC prevents last-minute bidding collusion

From the auctioneer's point of view, a good auction design gives the highest return to the seller. Auctions have characteristics like correlation and affiliation. The collusive agreement between the bidders prevents the auctioneer making the biggest profit. For the auctioneer, a crucial concern about auctions in practice is the ability of bidders to collude (Dixit and Skeath, 1999). The following sections explain how TAC prevents colluding.

Peter Wurman pointed out "There is an interesting collusion phenomenon in the on-line auction, which is that bidders tend to wait until the end of the auction and place a bid high enough to win" (Wurman, 2000). In the first TAC game, TAC'00, most of the agents delayed most of their purchases. The high-level bidding strategy of ATTac-2000, winner of 2000, is based on the premise that it is best to delay "committing" to the current G* (The most profitable allocation of goods given current holdings and prices) for as long as possible. The figure 10 shown below visualizes activity from the second game of the TAC 2000 finals. Rectangular towers are used to represent each bid made during the game.

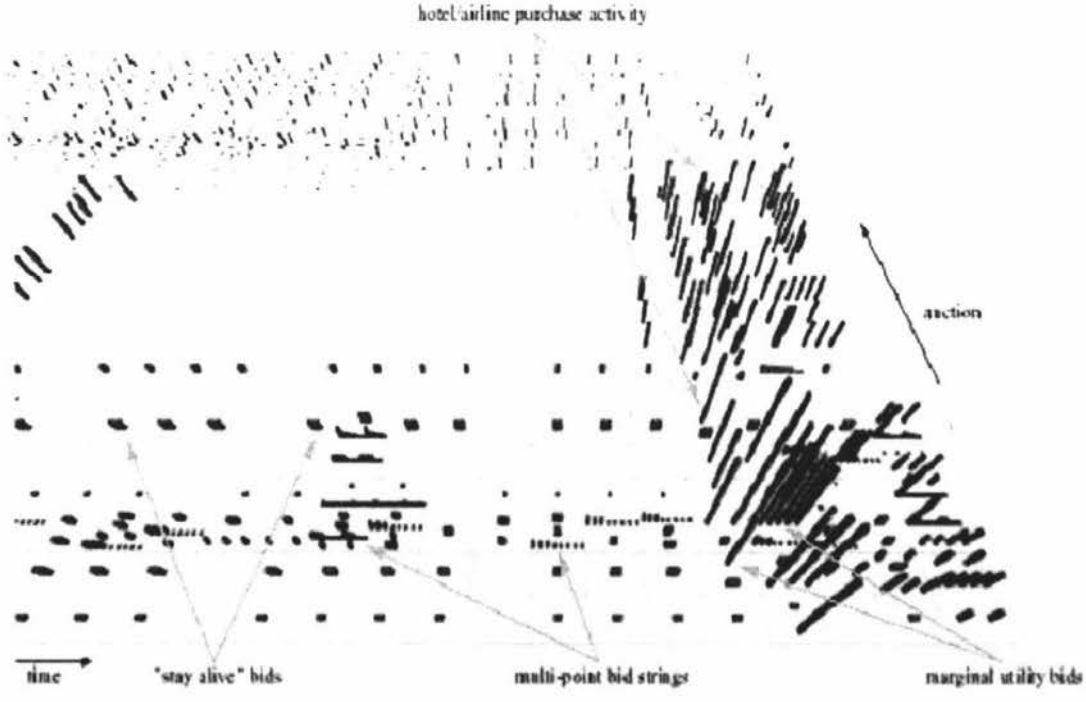


Figure 10: Visualizes bidding activity of TAC 2000 (Healey and Wurman, 2001)

Some observations about the TAC 2000 game play are: Most agents placed bids for hotel rooms and airline tickets at or near the end of an auction (Healey and Wurman, 2001).

This is rational. Al Roth and Axel Ockenfels from Harvard University believe bidders collude to keep prices down. Their hypothesis is that late bidding is a way for bidders to bypass the bidding wars, which allows them to get the item at a lower price (Ozmioz, 2002). Bidding high late and letting chance decide the winner is better for all the bidders than bidding high early and starting a bidding war. This is all well and good for the bidders (Ozmioz, 2002). Bidders pay less means that auctioneer gets less. This is not good design from an auctioneer's point of view.

In order to get more profit for the auctioneer, it is good to encourage bidders to bid earlier. TAC-01 decided to close one randomly chosen hotel auction at 4 mins time into the game and close one randomly chosen hotel auction each one minute thereafter. This changes agents' bidding strategies. Most of the agents submit bids before 4 mins into the game rather than waiting near the end of an auction. TAC prevented the last-minute bidding collusion phenomenon in the on-line auction.

2.3 TAC prevents low price bidding collusion

Demand, supply and price are three basic economic terms. Demand is the quantity the buyers want to have; supply is the quantity the sellers want to sell; and the price is either the selling or buying price. The relationship between demand, supply and price are shown in figure 11.

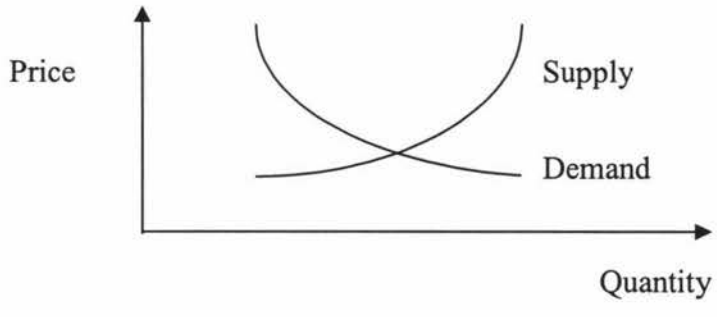


Figure 11: The demand, supply and price model

In double auctions, when supply couldn't satisfy demand, the price increases; when the price is increased, more sellers want to sell and the supply is increased. Once the supply exceeds demand, the sell price drops to attract more demand. In the one-side auction, where the supply is fixed, when the buyer's demand exceeds the supply, the sell price increases until the buyers quit.

Robinson makes the simple but important point that a collusive agreement may be easier to sustain in a second-price auction than in a first-price auction (Klemperer, 1999). Assuming, for simplicity, no problems in coming to agreement among all the TAC agents, or in sharing the rewards between them, and enough hotel rooms for all the agents, the optimal agreement in a TAC hotel room auction is all the agents bid zero, and no agnet has any incentive to cheat on this agreement. Eventually, all the agents pay zero for the hotel room. This could be implicitly Nash equilibria.

In the TAC game, client preferences are random. The randomised client preferences of arrival and departure days cause the random hotel rooms demands. The statistic data shows that the average number of rooms required for each agent on day 2 and day 3 is high. The hotel rooms are short in those two days. TAC breaks the Nash equilibria for bidding low price collusion.

2.4 Externalities in TAC

A bidder's estimate of value is affected not only by the bidder's own perceptions but also by the perceptions of other bidders if other bidders' perceptions are informative. Every agent has their own private value for the goods, but they are also concerned about "What others pay?" If the price is less than the agent's predicted value, the agent will maximize its utility by giving low bid prices too. Information is affiliated.

For example, in one of the hotel room auctions of TAC, an agent needs 15 rooms and the current ask price is $100. If this agent knows that there is one $150 bid and the other 15 bids are $100. Then this agent could bid for 15 rooms for $101, which is a guarantee that the 15 rooms are secure for this agent at the price of $101 at this point.

In a non-cooperative game, the utility payoff to one player only depends on the strategies taken by all the players, but not on the identity of the players that undertake certain actions or face certain outcomes. If this is the case, the game contains externalities (Laffont, 1987). Externalities between bidders tend to make collusion harder (Caillaud and Jehiel, 1998).

In the TAC game, only the server (auctioneer) publishes the ask price. In the TAC hotel room auction, the server publish the 16^{th} ask price of hotel room. The agents cannot communicate with others directly; they don't know other individual agents' needs and bids. For example, in one of the hotel room auctions of TAC, an agent needs 15 rooms and the current ask price is $100. The server publish the 16^{th} hotel room ask price $100. The only thing this agent knows is that the 16^{th} hotel room price. This agent may assume that 15 bids are $150 and one bid is $100. It might guarantee one room is secure for this agent at the price of $101 at this point. It wouldn't guarantee this agent get 15 rooms for $101. Beacuse this agent doesn't know the others' information, this agent may try bid high to secure the 15 rooms. For example, this agent may assume that 15 bids are $150 and one bid is $100. In this case, this agent needs to bid $151 to secure the 15 hotel rooms. This lower information penetrability in TAC tends to lead some of the agents to bid high. This is a good design from auctioneer's point of view.

2.5 TAC: an efficient market

If the prices prevailing in a market make it impossible to earn abnormal economic profits by trading in that market on some specified amount of information, it is an efficient market. "An efficient market is a market in which prices always "fully reflect" available information" (Fama, 1969).

If a price moves in a random walk, the value of the price in any period will be equal to the value of the price in the period before, plus or minus some random variable. That random variable is normally distributed with a constant variance and is entirely independent of the value of the price (Fama, 1969).

Since return distributions are assumed to be stationary through time, past returns are the best source of such information. However, that the sequence (or the order) of the past returns is of no consequence in assessing distributions of future returns (Fama, 1969).

Every agent in the TAC market has the same right to access the historical information and has the same right to access the information when the auction is going on. The price follows a random walk.

Pier Lica Lanzi and Alessandro Strada from Artificial Intelligence and Robotics Laboratory of Politecnico di Milano presented a statistic analysis on the data from the Second Trading Agent Competition (TAC01). The result shows that no agent performed significantly better than the others, from a statistical viewpoint. This suggests that in TAC-01 there is not a real "winner". TAC is an efficient market.

2.6 Suggestions for TAC

TAC is a fair market. All the agents cannot communicate to each other. They get an ask price from the server. If they want to sell entertainment tickets, they need to send sell bids to the server. The server is the interface of all the agents. Externalities are designed for the TAC environment. Does the externality work perfectly?

Assume, for simplicity, there is no problem in coming to agreement among two bidders. Is there any possibility that agents could communicate with each other? I did an experiment. I designed two agents, one called naughty and another one called sacrifice. The naughty sells its entertainment tickets at a high price and sacrifice buys the entertainment tickets at the high price in the game. The code of naughty is in Appendix 24. The code of sacrifice is in Appendix 25.

Agent naughty sells entertainment tickets for 18000.

```
private void sellEnt() {
...
    for (int day = 1; day <= 4; day++) {
```

```
    for (int type = 1; type <=3; type++) {
     int auction = agent.getAuctionFor(2, type, day);
     entOwn[day-1][type-1] = agent.getOwn(auction);
     if (entOwn[day-1][type-1] > 0) {
      Bid bid = new Bid(auction);
       bid.addBidPoint(-entOwn[day-1][type-1],18000);
          agent.submitBid(bid);
...
}
```

Sacrifice buys the entertainment tickets for 20000

```
  private void buyEnt(){
   for (int day = 1; day <= 4; day++) {
    for (int type = 1; type <=3; type++) {
    int auction = agent.getAuctionFor(2, type, day);
      Bid bid = new Bid(auction);
      bid.addBidPoint(2,20000);
      agent.submitBid(bid);
...
```

It ends up that Naughty got a very high score. This shows that it is possible to break the externality in the TAC game.

Even if there is no agreement among all the bidders; the bidding price would be affected if the bidder knows more about other bidders' perceptions. Wellman pointed out that the 12 agents that entered TAC-02 provides some natural collaboration, even though they were not designed to be collaborating agents (Wellman, et, al, 2003). One of reason is that the twelve agents played in 2001. They know each other's strategy. The information retrieved (owned) increasing the collaboration possibility.

If the players are not identified, the collusive agreement is not easy to reach. My suggestion is that TAC host the game with anonymous name of agents for public. TAC

server knows all the agents, but the agents don't know each other. If they don't know each other's strategy, the collaboration possibility will decrease.

2.7 Summary

This chapter investigated the TAC game from an auctioneer's point of view. The goal of an auctioneer is to make maximum profit. Collusion between bidders could hinder an auctioneer achieving this goal. Preventing collusion is one of the important aspects of TAC design. TAC closes randomly chosen hotel auction. This strategy prevents last-minute collusion. TAC doesn't provide enough hotel rooms especially in day 2 and day 3. This strategy prevents low-price collusion. Externalities between bidders in TAC tend to make collusion harder. If TAC hosts the game with anonymous name of agents for public, it makes collusion even harder for agents.

TAC design is a good design from an auctioneer's point of view. TAC is also an efficient market.

The more aggressive agents are in the TAC market, the worse the market. Those aggressive agents cause all the agents to suffer. In order to find the strategy for an agnet to play aginst seven open-loop (EarlyBidder) agents, the RiskerWise agent was designed. Chapter 3 will explain the design of the RiskerWise agent in details.

Chapter 3: The Riskerwise Agent

The open-loop EarlyBidder causes economy to suffer, which causes all the players to suffer. In order to find the strategy to play aginst seven open-loop (EarlyBidder) agents, the RiskerWise agent was designed. The RiskerWise agent played 58 games with seven aggressive open-loop (EarlyBidder) DummyAgents. RiskerWise agent achieved a score of 2691.3 ± 745.8. In the following sections, the client utility function and DummyAgent will be introduced. The details of how RiskerWise was designed and the performance of RiskerWise played against seven open-loop EarlyBidder DummyAgents will be evaluated. The relationship between the score, cost and utility will be analysed.

3.1 Client utility functions

The goal of an agent is to maximize the score. The client utility plays an important role for the score. This section explains the client utility function in details.

The preferences of a client are:

- A preferred arrival date. The preferred arrival dates for each client are chosen randomly.
- A preferred departure date. The preferred departure dates for each client are chosen randomly.
- A bonus value for the better hotel.
- A bonus value for each entertainment type.

A travel package is specified by the following factors:

- An actual arrival date
- An actual departure date
- Tampa Towers indicator (*TT?* in $\{0, 1\}$).

  According to the game rule, the clients have to stay at least one night. So the soonest departure day is one day after arrival. The clients don't need

hotel rooms for the day of their departure. The client can't change the Hotel during the trip. The hotel bonus is a bonus for the whole trip, not for each day.

- Entertainment ticket indicator (*AW?*, *AP?*, *MU?*, each in {0,1}).

  According to the game rule, the clients cannot get additional bonus for having more than one of the same types of entertainment ticket during their trip.

Clients receive zero utility for infeasible packages.

- According to the game rule, a travel package is *feasible* only if it has the same hotel rooms for every night between the arrival and departure dates. The client has to stay at least one day.
- According to the game rule, an entertainment package is *feasible* only if there is no more than one ticket per day, and all of the tickets are in the period when the client is in town. A client can only have one ticket for each entertainment type.

Before the functions are introduced, the abbreviations are explained:

TT: stands for the Tampa Towers Hotel
SS: stands for the Shoreline Shanties Hotel
HP: stands for the premium value for the better hotel
AW: stands for Alligator wrestling, one of the entertainment events
AP: stands for Amusement park, one of the entertainment events
MU: stands for Museum, one of the entertainment events
PA: stands for a single preferred arrival date
PD: stands for a single preferred departure date
AA: stands for an actual arrival date
AD: stands for an actual departure date
?: stands for the indicator. If the condition is true, it is 1, otherwise, it is 0.

TT?: If the agent bought Tampa Towers room, TT is 1, otherwise, TT is 0.
AW?, AP?, MU?: A ticket indicator for each event type. For example, if the agent bought in Alligator wrestling, AW is 1, otherwise, AW is 0.

A client's utility for a feasible travel package and a feasible entertainment package is given by the following formula (Game Overview, n.d.):

$$u = 1000 - \textit{travel penalty} + \textit{hotel bonus} + \textit{fun bonus}$$

where

$$\textit{travel penalty} = 100*(|AA - PA| + |AD - PD|)$$

$$\textit{hotel bonus} = TT? * HP$$

$$\textit{fun bonus} = AW? * AW + AP? * AP + MU? * MU$$

3.2 DummyAgents

The DummyAgents plays an open-loop (EarlyBidder) strategy. They are greedy. They bid following with their clients' preferences without concern about the cost. They buy flight tickets at the beginning of the game. They bid the good hotel for those clients who have more than 70 hotel bomus. They start their bids in $200. They keep increasing their bid if it is below the 16 highest bids, which tends to lead to overvalue the hotel auctions. This can make it difficult for any players to pay off in a game against several dummy agents.

3.3 Strategies of RiskerWise

The success of agent strategies depends a great deal on the strategies of the other competitors (Stone, 2002). The first task for the RiskerWise agent was to analyse the strategies of its competitors: seven DummyAgents. DummyAgents' strategies are open-

loop and aggressive. DummyAgents could pay arbitrarily high prices for the hotel rooms they have decided to buy. Many of the prices for hotel rooms are expected skyrocket.

The prices of flight tickets in flight ticket auctions rise over the time as shown in the figure 3. It shows that early-bidding is better thatn late bidding in figure 7. Because one of the hotel auctions will be cloased on 4^{th} minutes after the TAC game starts, it is wise to make decision before 4^{th} minutes after the TAC game starts. On the negative side, it might cost more. If the agent buys cheap flight tickets very early, it might not be able to buy the necessary hotel rooms. The flight tickets might be wasted. This leads to some invalid travel packages, which means the utility would be zero. But the agent has to pay for the flight cost. This results in the negative score for the agent. Because DummyAgents make their decision at the beginning of the game and don't change their strategies during the game, the RiskerWise agent could use current price to make its decision earlier to avoid pay more on the filght tickets.

Expecting the bad environment, the RiskerWise agent doesn't plan to simply following its clients' preferences. The RiskerWise agent uses search algorith to quickly find the highest client utilities, and then bids the goods according to the search results.

As discussed in literature review, truth telling is an optimal strategy. RiskerWise's bidding strategies for hotel are that bid higher than the current ask price by an increment and give up when the price of a particular auction exceeds its true value.

The flowchart of strategy for the hotel room auction is shown in Figure 12.

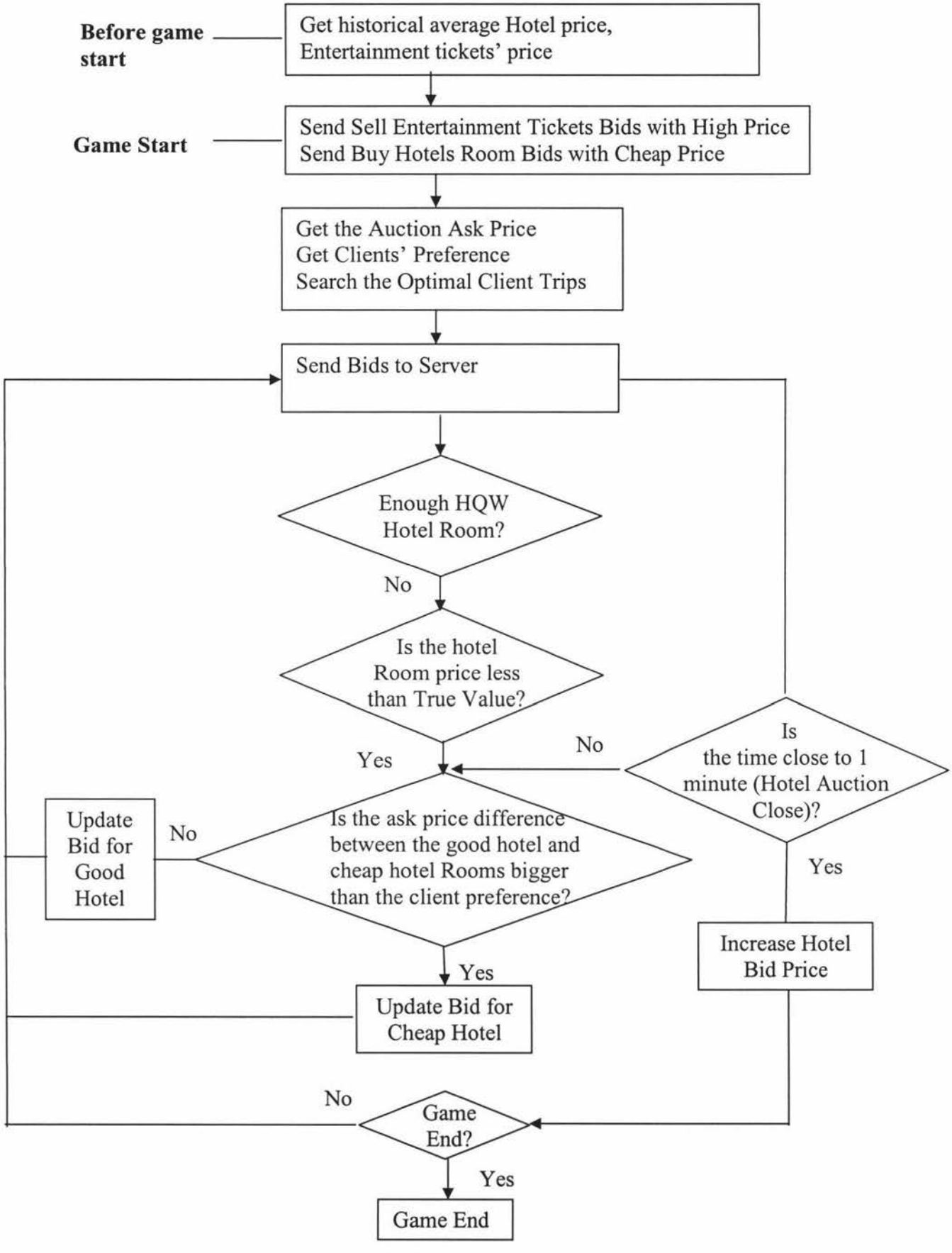


Figure 12: The flowchart of strategy for the hotel room auction

When the game starts, the RiskerWise agent sends bids to buy all the hotel rooms at a very cheap price ($1) and sell entertainment tickets according to the max client preference of the entertainment type. It then searches to find the optimal client trips, assuming hotel prices and entertainment tickets' price will be at historical averages. It then allocates the resources according to the new client optimal preferences, then sends the Bids immediately (In the Flight auctions, the agent placing a winning bid pay not their bid but the current ask price. Thus, RiskerWise bids above the current ask price ($1000) to ensure that these bids are successful). After 4 minutes, RiskerWise checks if it has enough hotel rooms. If not, RiskerWise checks both types of the hotel rooms on the allocated days. If it hasn't enough hotel rooms, it then checks the hotel ask price difference, if the hotel ask price difference is bigger than the good hotel preference, it then changes the bid to the cheap hotel; otherwise, it changes to the good hotel. When the game time is close to the one of the hotel room auction closing times (every min after 4 mins from game starts), the RiskerWise agent raises the bids for the required hotels at prices higher than the ask price to win the auction in case the competitors raise their price suddenly. The RiskerWise agent checks the desired tickets (if clients have a preference but don't have tickets, the agent needs to buy them), then finds if there are some not in the held tickets (allocated by the Auctioneer), then buys those, which are not in the held tickets and sells the not desired tickets at a high price. The price of buying entertainment tickets increases and the selling price of the entertainment tickets decrease while the time passes.

STRATEGY FOR HOTEL AUCTIONS

BEGINNING

1. Sends bids to buy all the hotel rooms at a very cheap price ($1).
2. Determine initial sets of clients' room choices by search
3. Bidding in all needed auctions
4. Update quote. RiskerWise checks if it has enough HQW hotel rooms in both types of hotel type. If not, RiskerWise then checks the hotel ask price difference,

if the hotel ask price difference between good hotel and cheap hotel is bigger than the good hotel preference, it then changes bid to the cheap hotel; otherwise, it changes to the good hotel.

Go to 4 point until it is close to the 4th minutes of the game

5. Check if RiskerWise has obtained all the needed hotel room in both types
6. If not - Calculate if it still can win needed hotel (HQW)
7. If not - RiskerWise then checks if the ask price exceeds its acceptable price, if not; it compares the hotel ask-price difference. If the hotel ask-price difference between good hotel and cheap hotel is bigger than the good hotel preference, it then increases the bids for the cheap hotel; otherwise, it increase the bids for good hotel.
8. When the game time is close to any one of the hotel room auction closing time, RiskerWise raises the bids for the required hotels at prices much higher than the ask price to win the auction in case the competitors raise their price suddenly.

Because a hotel auction may close and the agent will miss out on those rooms, as a rational agent, the RiskerWise agent submits its hotel bids before the end of the 4th minute.

Because the RiskerWise agent buys the air tickets very early, the cost of the air tickets will be treated as a sunk cost.

According to the game rule, hotel bonus values are chosen for each client uniformly in the range of $50 to $150. The flight prices are constrained to stay in the range between $150 and $800. Entertainment bonus values for every entertainment type are chosen uniformly in the range of $0 to $200.

Assuming it is about 700 for return tickets for a client, 80 for an average entertainment bonus and 90 for an average hotel bonus, the max acceptable bid price for the hotel room could be:

By using the utility formula, if the agent pays 470, the utility will be zero.

$1000 + 80 + 90 - x - 700 = 0$

$x = 470$

But if the RiskerWise agent couldn't buy the hotel room, the RiskerWise agent will lose 700 and can't satisfy the client needs. If the RiskerWise agent pays up to 1170, it will lose 700, but satisfy the client.

$1000 + 80 + 90 - 1170 - 700 = -700$

RiskerWise chooses the second option. But if the hotel price rises above 1170, RiskerWise gives up. RiskerWise doesn't want to lose more than 700.

STRATEGY FOR FLIGHT AUCTIONS

Buy all needed flights for the price wanted by the server as soon as the initial allocation is completed. RiskerWise bids above the current ask price ($1000) to ensure that these bids are successful because the agent placing a winning bid pay not their bid but the current ask price in the Flight auctions.

STRATEGIES FOR ENTERTAINMENT AUCTIONS

1. Find the allocated entertainment tickets by server
2. Calculate the max entertainment preference of clients
3. When the game starts, RiskerWise sends bids to sell entertainment tickets higher than the max client preference of the entertainment type.
4. Update quote. The price of buying entertainment tickets increase and the selling price of the entertainment tickets decrease while the time passes.
5. The RiskerWise checks the desired tickets (clients have a preference but don't have tickets, the agent needs to buy them), if there are desired tickets, send buy bids.
6. If there are undesired tickets (clients don't have a preference but have tickets), sell the holding tickets at a high price.

7. The price of buying entertainment tickets increase and the selling price of the entertainment tickets decrease while the time passes.

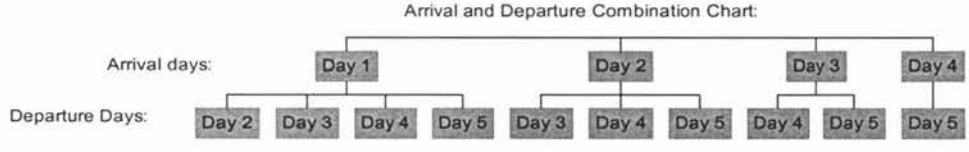
   Buy Bid: 30 + (GameTime * 100) / 720000;

   Sell Bid: 200 - (GameTime() * 120) / 720000;

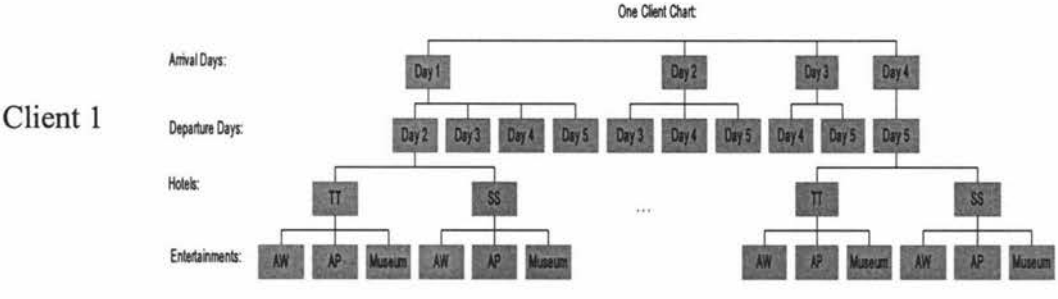
3.4 The Search algorithm

When the game starts, the reality of the market's demand is unpredictable. It is wise to use the historical average prices for the entertainment price. The RiskerWise gets all the client preferences, flight price and historical data, uses a search algorithm to calculate the highest client utilities, and then bids according to the new client optimal preferences.

The search tree is very large if the agent wants to find the global optimal for 8 clients. Let's have a look at the search elements. There are 10 possible arrival and departure days for each client. Clients could arrive from Day 1 to Day 4 and departure from Day 2 to Day 5.



There are 2 possible Hotels (the Tampa Towers and Shoreline Shanties) for each client. There are 3 possible Entertainments could be chosen for each client. The search tree looks like the graph below.



Client 1

Client 2

. . .

. . .

. . . .

. . . .

. . . .

Figure 13: The global optimal search tree

The search space is 167,961,600,000,000, which is (10 * 2 * 3) to the power of 8.

Given the big search space and the 120 MB memory of the computer used, the RiskerWise agent would not be able to finish the search in a short period of time. So the RiskerWise agent reduces the size of the state space by searching one client at a time. The search space reduces to (10 * 2 * 3) * 8 = 480, which is 1/349920000000 of the original search space.

The RiskerWise agent used Breadth-first search (BFS) as a search algorithm, which starts at the root node and explores all the neighbouring nodes. Then for each of those nearest nodes, explore their unexplored neighbour nodes, and so on until it finds the goal. Formally, BFS is an uninformed search method that aims to expand and examine all nodes of a tree systematically in search of a solution. It exhaustively searches the entire tree without considering the goal until it finds it.

- BFS is complete - it finds a goal-state if one exists.
- BFS is optimal - it needs the smallest number of steps to reach the goal.
- BFS has space complexity linear in the size of the tree/graph searched as it needs to store all expanded nodes in memory.
- BFS has time complexity linear in the size of the graph or tree searched.

The Riskerwise agent Pseudo-code Search Algorithm

For client 1 to 8

For Arrival day 1 to 4

For Departure day 2 to 5

Calculate the utility

Find max utility

Store the data

Repeat until search the entire tree

The code is shown below.

```
protected void search()
{
 ...
    for (countClient = 0; countClient<8; countClient++)
  {
    int i=countClient;
    int Arrival =agent.getClientPreference(i, TACAgent.ARRIVAL);
    int Departure = agent.getClientPreference(i, TACAgent.DEPARTURE);
    float Hotelbouns = agent.getClientPreference(i, TACAgent.HOTEL_VALUE);
    e1 = agent.getClientPreference(i, TACAgent.E1);
    e2 = agent.getClientPreference(i, TACAgent.E2);
    e3 = agent.getClientPreference(i, TACAgent.E3);
    float maxTemp = 0;
    for (Aday[i] = 1; Aday[i] <= 4; Aday[i] ++) {
      for (Dday[i] = Aday[i]+1; Dday[i] <=5; Dday[i] ++) {
        float temp = 1000+100*(Math.abs(Aday[i]-Arrival)+ Math.abs(Dday[i]-
                    Departure))-p[Aday[i]-1]-p[Dday[i]+2]+hotelScore(Aday[i],
                     Dday[i],Hotelbouns) + funbounds(Aday[i], Dday[i]);
       ...
}
```

3.5 Data structures used

After calculating all the scores, the RiskerWise agent needs to store the information and quick access the largest score. The data structure needs to have the characteristics such as quick to access the largest value.

A heap is a binary tree that is ordered from top to bottom, each node in a heap along every path from the root to a leaf, the nodes are arranged in a descending order, which states that the node's key is larger than (or equal to) the keys of its children. It is also called "max heap". The highest priority item is at the root. In other words, the largest value data item with the smallest key. The RiskerWise agent only need access the smallest key to get all the information of the max score. Siblings' nodes in a heap are not ordered in sequence, but this is not a problem since only the best individual is needed.

Heap has the characteristic the RiskerWise agent needed. The RiskerWise agent chose the Heap as its data structure. The heap is implemented as an array. The code of heap is in appendix 3.

3.6 Performance

The RiskerWise agent played 58 games continuously with 7 open-loop risk players (DummyAgents) from Game 25863 to 25920. It got most of the top scores of the 58 games and it didn't have any negative scores. The scores table of the 58 games played by the RiskerWise agent is in appendix 5. Figure 14 shows the statistics of the RiskerWise agent.

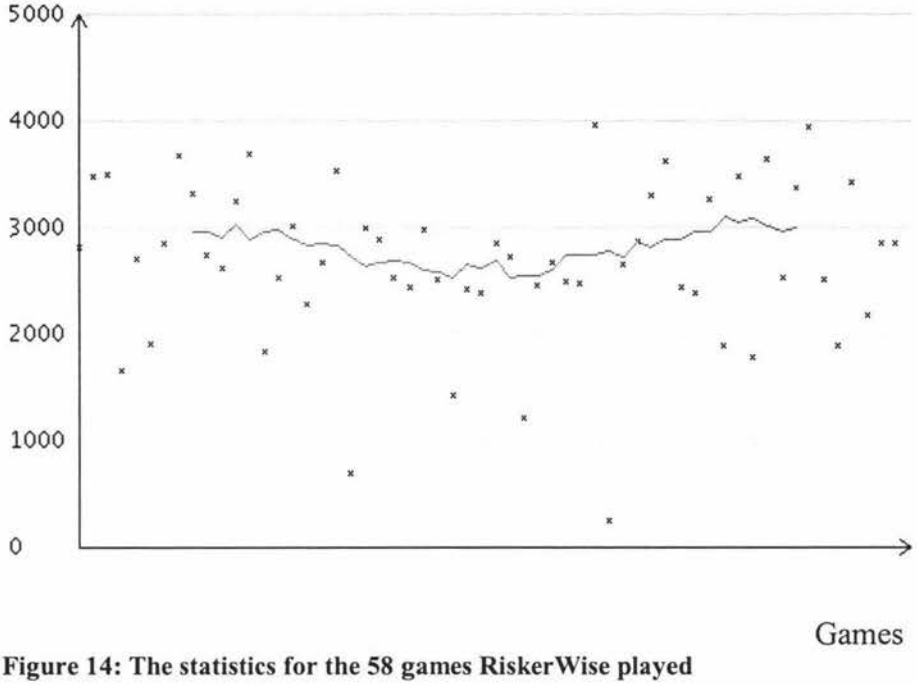


Figure 14: The statistics for the 58 games RiskerWise played

3.6.1 Descriptive Statistics of Score

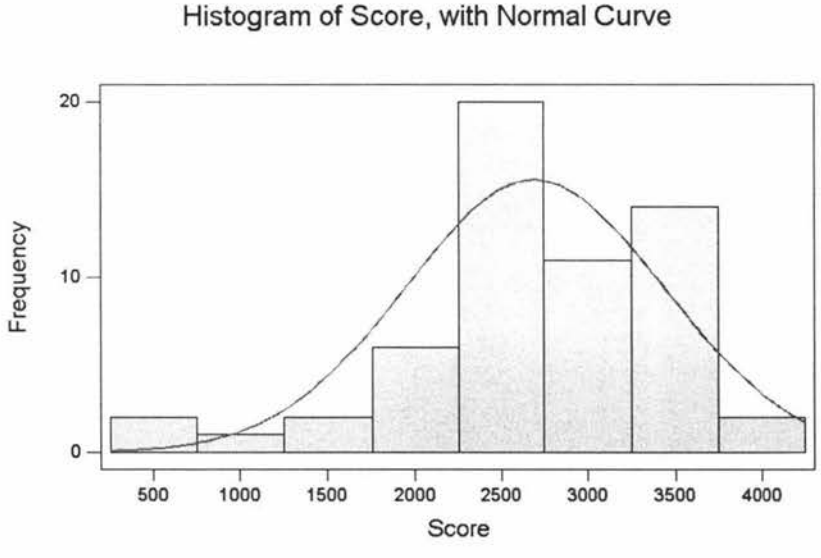


Figure 15: Histogram of Score, with Normal Curve

Variable	N	Mean	Median	TrMean	StDev
Score	58	2691.3	2704	2736	745.8
Variable	SE Mean	Minimum	Maximum	Q1	Q3
Score	97.9	257	3966	2434	3290

The median score is 2704. Neglecting the most extreme 5% of the data at each end and averaging the rest, the mean is 2736.

The standard deviation is 745.8. There are about 95% of the data will be within 1945.5 and 3437.1.

The accuracy of the mean 2691.3 is likely to be as estimate of the population mean is 97.9%.

The maximum score is 3966 while the minimum is 257.

The bottom 25% of data (at the top of the first quarter) is 2434 and the top 25% of data (at the top of the third quarter) is 3290.

3.6.2 Descriptive Statistics of Utility

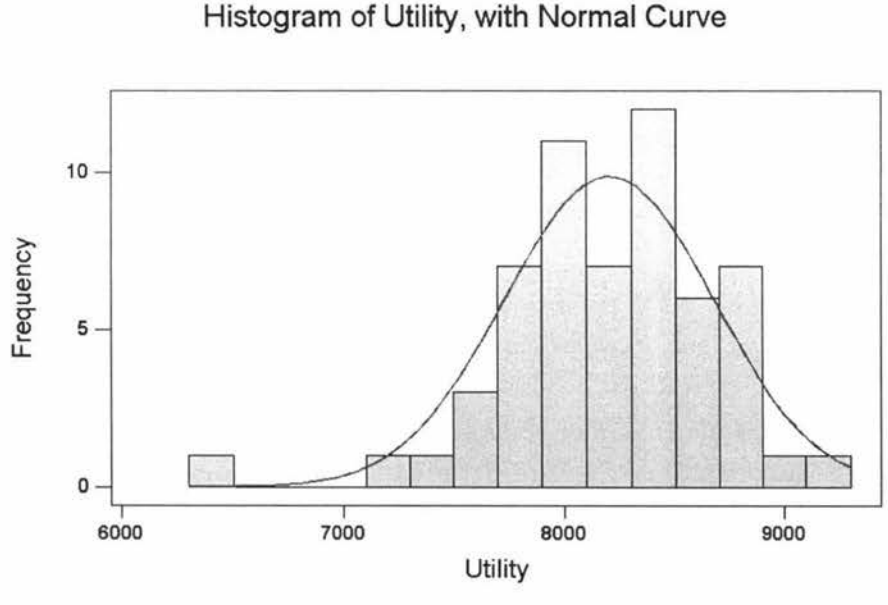


Figure 16: Histogram of Utility, with Normal Curve

Variable	N	Mean	Median	TrMean	StDev
Utility	58	8203.3	8228	8227	468.8
Variable	SE Mean	Minimum	Maximum	Q1	Q3
Utility	61.6	6400	9147	7936.8	8513.8

The median utility is 8228.0. Neglecting the most extreme 5% of the data at each end and averaging the rest, the mean is 8227.0.

The standard deviation of the utility is 61.6. The standard deviation of the utility is small, which means that the most of the utility values are similar. The allocation strategy is similar. About 95% of the data will be within 2 standard deviation of the mean 8203.3, which means that there are about 95% of the data will be within 8141.7 and 8264.9.

The accurate the mean 8203.3 is likely to be as an estimate of the population mean is 61.6%.

The maximum utility is 9147.0 while the minimum is 6400.0.
The bottom 25% of data (at the top of the first quarter) is 7936.8 and the top 25% of data (at the top of the third quarter) is 8513.8.

3.6.3 Descriptive Statistics of Cost

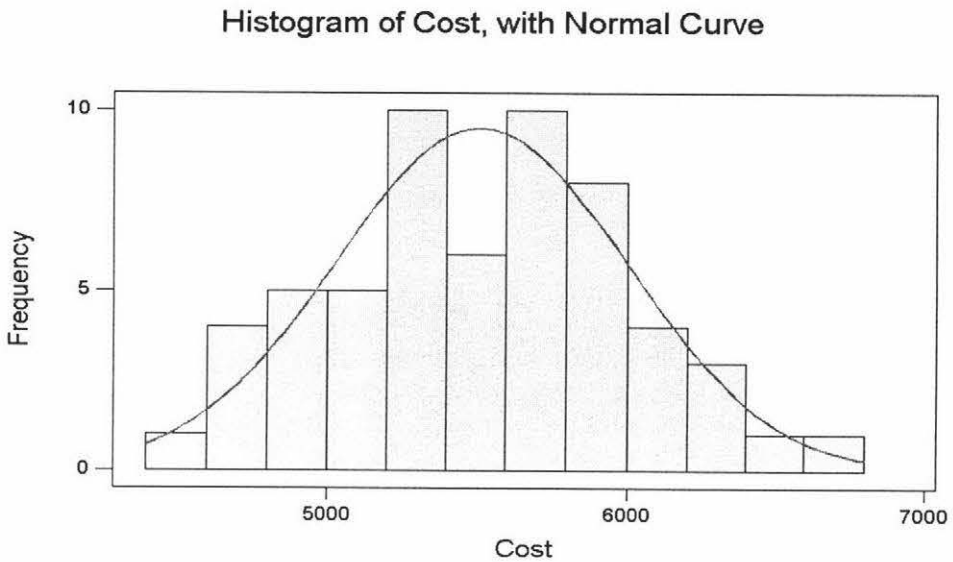


Figure 17: Histogram of Cost, with Normal Curve

Variable	N	Mean	Median	TrMean	StDev
Cost	58	5512	5524.5	5503.6	486.8
Variable	SE Mean	Minimum	Maximum	Q1	Q3
Cost	63.9	4590	6678	5185	5824.5

The median cost is 5524.5. Neglecting the most extreme 5% of the data at each end and average the rest, the mean of cost is 5503.6.

The standard deviation of cost is 486.8. There are about 95% of the data will be within 5448.1 and 5998.8.

The accuracy of the mean 5512.0 is likely to be as an estimate of the population mean is 63.9%.

The maximum cost is 6678.0 while the minimum is 4590.0.

The bottom 25% of data (at the top of the first quarter) is 5185.0 and the top 25% of data (at the top of the third quarter) is 5824.5.

3.6.4 Confidence Interval

The confidence interval give a lower bound and upper bound for the population mean based on the sample. The methodology assumes the sample is random, and will give correct bounds for μ on 99.9% of occasions (99.9% of random samples).

3.6.4.1 One-Sample T: Score

Variable	N	Mean	StDev	SE Mean	99.9% CI
Score	58	2691.3	745.8	97.9	(2351.6, 3031.1)

The 99.9% confidence interval of score is between 2351.6 and 3031.1, which means that there is a 99.9% chance that Riskerwise's score will be between 2351.6 and 3031.1.

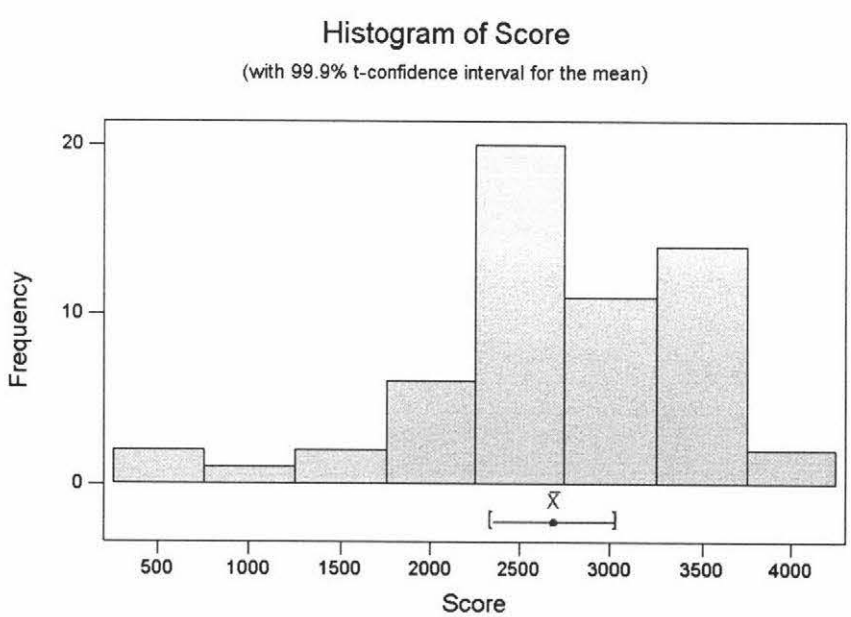


Figure 18: Histogram of Score (with 99.9% t-confidence interval for the mean)

3.6.4.2 One-Sample T: Utility

Variable	N	Mean	StDev	SE Mean	99.9% CI
Utility	38	8203.3	468.8	61.6	(7989.7, 8416.9)

The 99.9% confidence interval of utility is between 7989.7 and 8416.9, which means that there is a 99.9% of the chance that Riskerwise's utility will be between 7989.7 and 8416.9.

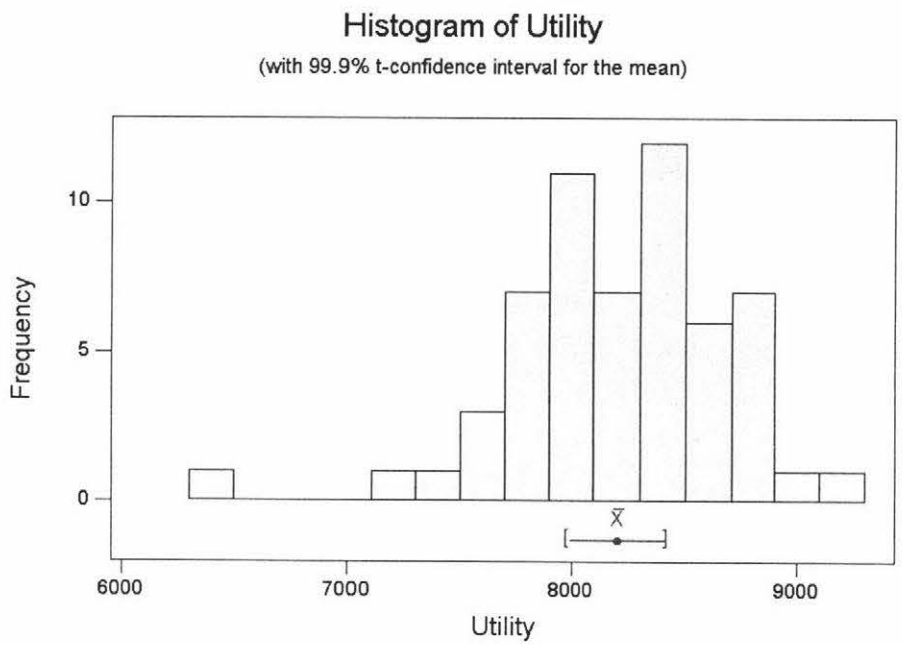


Figure 19: Histogram of Utility (with 99.9% t-confidence interval for the mean)

3.6.4.3 One-Sample T: Cost

Variable	N	Mean	StDev	SE Mean	99.9% CI
Cost	58	5512	486.8	63.9	(5290.2, 5733.8)

The 99.9% confidence interval of cost is between 5290.2 and 5733.8, which means that there is a 99.9% chance that Riskerwise's cost will be between 5290.2 and 5733.8.

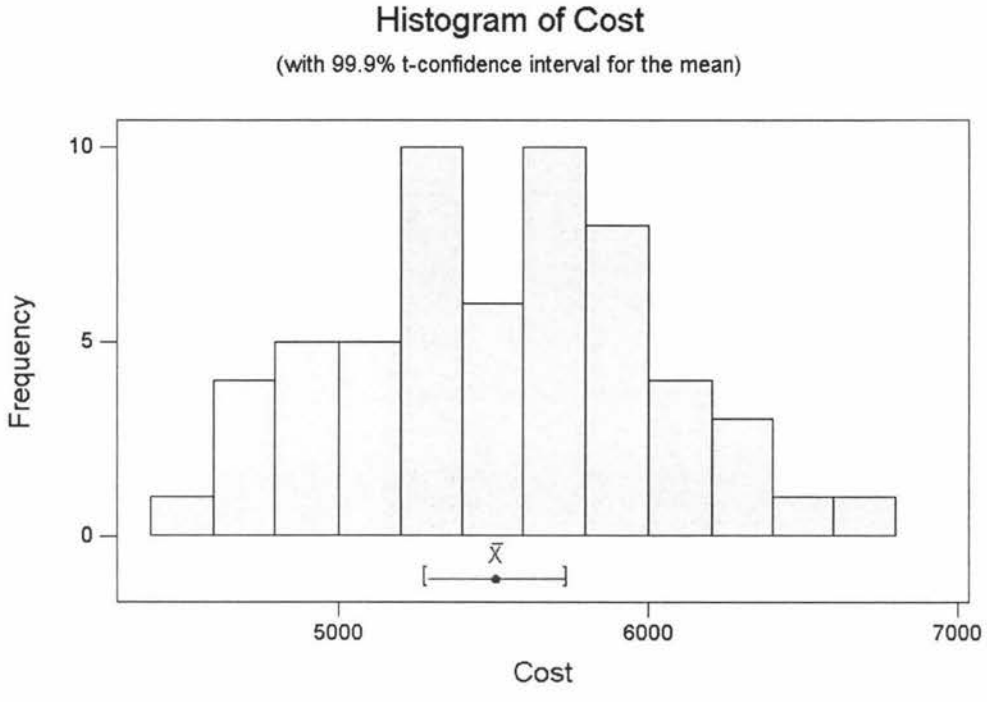


Figure 20: Histogram of Cost (with 99.9% t-confidence interval for the mean)

3.7 Two games the RiskerWise agent participated in

The RiskerWise agent played with seven aggressive DummyAgents. The more open-loop risk players, the worse the environment. Open-loop risk players make the price sky-high, which harms all the players. The RiskerWise agent avoided the most expensive hotel rooms (day 2 and day 3 hotel rooms are the most expensive ones). The RiskerWise agent bought most of hotel rooms on day 1 and day 4 for its clients. The RiskerWise agent arranged short travel for its client, for example, arrival on day 1 and departure on day 2 or arrival on day 4 and departure on day 5. It may not be very good for its clients' utilities, but the overall cost-savings are worthwhile.

3.7.1 Game 25899

The RiskerWise agent won the game with the best score. The table 10 shows that the market environment was very bad. The total cost for goods was 102168 in this market. Only The RiskerWise agent made profit, the seven competitors made a huge loss. Dummy-4 satisfied its customers' preferences most. It had the highest utility score. But it also had the biggest cost, which caused its failure. Because of the bad market environment, The RiskerWise agent had to change its clients' preferred travel dates, which suffered on the utility. But The RiskerWise agent had the least cost. It only paid a low price ($1) for the hotel room; it had the highest score in the end.

Agent	Utility	Cost	Score
RiskerWise	8765	4799.00	3966.00
Dummy-1	0	4888.00	-4888.00
Dummy-7	9783	15735.00	-5952.00
Dummy-6	9259	15773.00	-6514.00
Dummy-2	4797	11676.00	-6879.00
Dummy-5	9898	16968.00	-7070.00
Dummy-4	10031	17541.00	-7510.00
Dummy-3	5032	14788.00	-9756.00

Table 10: Game results at game 25899

Game results for the RiskerWise agent in game 25899 are in appendix 6.
Client Preferences for the RiskerWise agent in game 25899 is in appendix 7.
Endowments for the RiskerWise agent in game 25899 are in appendix 8.
Owned Goods for the RiskerWise agent in game 25899 are in appendix 9.
Transactions for the RiskerWise agent in game 25899 are in appendix 10.

3.7.2 Game 25900

The market environment was worse than in game 25899. The total cost for goods was 113892 in this market. It cost 11724 more on goods as compared with game 25899. It is again only the Riskerwise agent that made a profit, seven competitors made huge losses. Dummy-4 satisfied the customers most, it has highest utility score. The RiskerWise agent used the same strategy, which was shorten the trips and bought the hotel rooms on day 1 and day 4. But the RiskerWise agent lost 2365 on utility compared with its performance on game 25899, because the clients' preferences of RiskerWise have a longer stay period (16 days and 11 days in Day 2 and Day 3) compared with the RiskerWise agent in the game 25899 (12 days and 8 days in Day 2 and Day 3). The client preferences play an important role in the result of the competition.

Agent	Utility	Cost	Score
RiskerWise	6400	6143.00	257.00
dummy-4	9846	15392.00	-5546.00
dummy-1	2676	8296.00	-5620.00
dummy-5	9403	15244.00	-5841.00
dummy-6	9818	16211.00	-6393.00
dummy-2	7218	15473.00	-8255.00
dummy-3	9794	18175.00	-8381.00
dummy-7	8926	18958.00	-10032.00

Table 11: Game results at game 25900

Game results for the RiskerWise agent in game 25900 are in appendix 11.
Client Preferences for the RiskerWise agent in game 25900 is in appendix 12.
Endowments for the RiskerWise agent in game 25900 are in appendix 13.
Owned Goods for the RiskerWise agent in game 25900 is in appendix 14.
Transactions for RiskerWise in game 25900 are in appendix 15.

3.8: Relationships among score, utility and cost

The research hypothesis is that there are relationship between the Score and Utility. The idea of hypothesis testing is to establish truth by falsifying error. That is, because the samples are available, it is hard to positively prove that the variables have relationship (that would require doing a census of all the games). But if we hypothesise the opposite (viz that there is no relationship between the variables) and show that this leads to a very unlikely situation. Thus the null hypothesis H0 for this test is that the variables are independent. If the p-value < 0.01, the probability of such a discrepancy is small, the test is significant and the null hypothesis should be rejected.

To find out the relationship between the variables (score, utility and cost) is statistically significant or not, regression analysis is used. Linear Regression estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression is used to model the value of a dependent scale variable based on its linear relationship to one or more predictors.

3.8.1 Is the higher the utility, the higher the score?

Null hypothesis H0: Score and Utility don't have relationship.
Alternative hypothesis H1: The Score has correlation with Utility.

Figure 21 shows the scatter of score and utility with linear fit line.

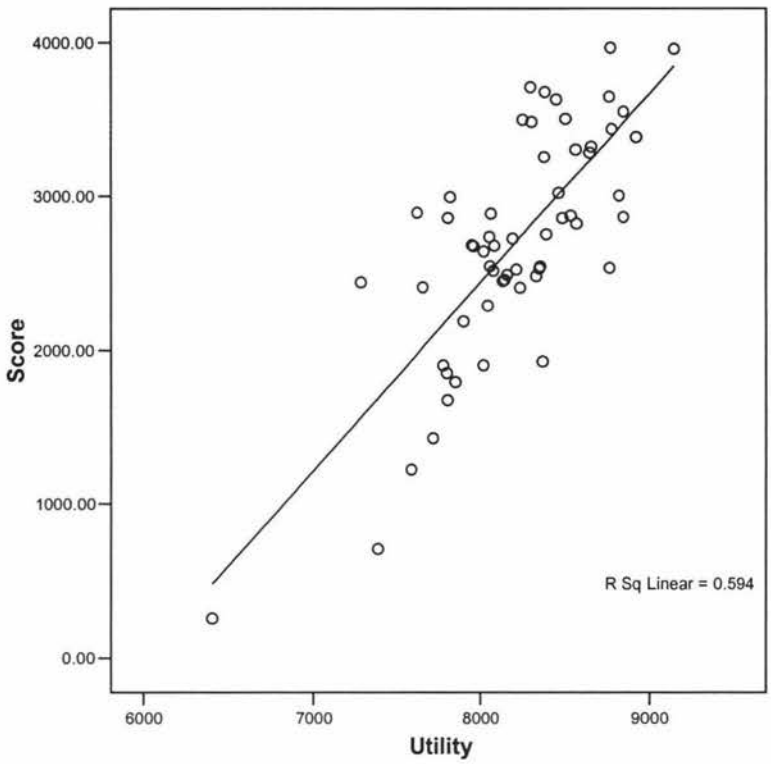


Figure 21: The graph of Scatter of Score Utility with the linear fit line

The resulting scatter plot appears to be suitable for linear regression. A scatter plot indicates that these variables are linearly related. These variables have a positive relationship. As the Utility of the Riskerwise agent increases, the Score increased. Further study the coefficients table, which shows the coefficients of the regression line.

Model		Unstandardized Coefficients		Standardized Coefficients	T	Sig.
		B	Std. Error	Beta		
1	(Constant)	-7366.859	1112.721		-6.621	.000
	Utility	1.226	.135	.771	9.054	.000

Table 12: Coefficients of Score and Utility. Score is a Dependent Variable.

The model is linear because increasing the value of the Utility by 1 unit increases the value of the Score by 1.23 units. Note that -7367 is the intercept, the model-predicted value of the dependent variable (Score) when the value of the predictor (Utility) is equal to 0.

The linear regression model assumes that there is a linear, or "straight line," relationship between the dependent variable and each predictor. This relationship is described in the following formula. The regression equation is

Score = - 7367 + 1.23 Utility

It states that the expected Score is equal to 1.226 * Utility – 7366.859. If the Utility is 9000, the predicted score would be 1.226 * 9000 – 7366.859 = 3667.141.

The ANOVA table tests the acceptability of the model from a statistical perspective.

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	18833717.212	1	18833717.212	81.971	.000(a)
	Residual	12866662.021	56	229761.822		
	Total	31700379.233	57			

Table 13: The ANOVA table for Score as a dependent variable and Utility as a predictor

The Regression row displays information about the variation accounted for by the model.

The significance value is less than 0.01, which means that the variation explained by the model is not due to chance. The Null Hypothesis is rejected. The Scores have correlation with Utility.

While the ANOVA table is a useful test of the model's ability to explain any variation in the dependent variable, it does not directly address the strength of that relationship. The model summary table reports the strength of the relationship between the model and the dependent variable.

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.771(a)	.594	.587	479.33477

Table 14: Model Summary for Score as a dependent variable and Utility as a predictor

R Square is the coefficient of determination, which is obtained from the squared value of the multiple correlation coefficient. It shows that 59.4% of the variation in score explained by utility in the model.

A residual is the difference between the observed values and model-predicted values of the dependent variable. The residual for a given Utility is the observed value of the error term for that Utility. A histogram or P-P plot of the residuals will be used to check the assumption of normality of the error term.

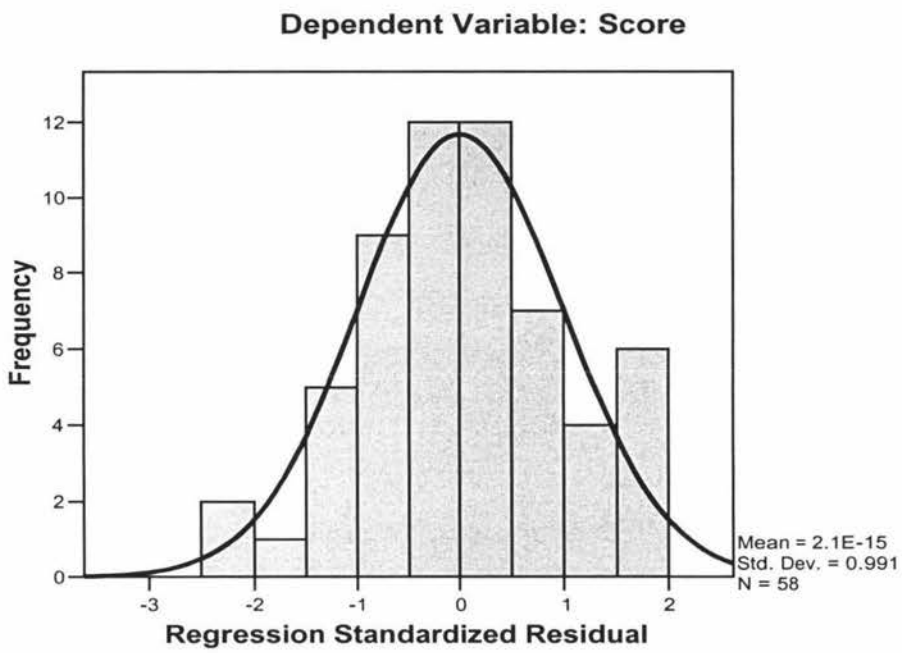


Figure 22: The histogram of the residuals

The shape of the histogram should approximately follow the shape of the normal curve. This histogram is acceptably close to the normal curve.

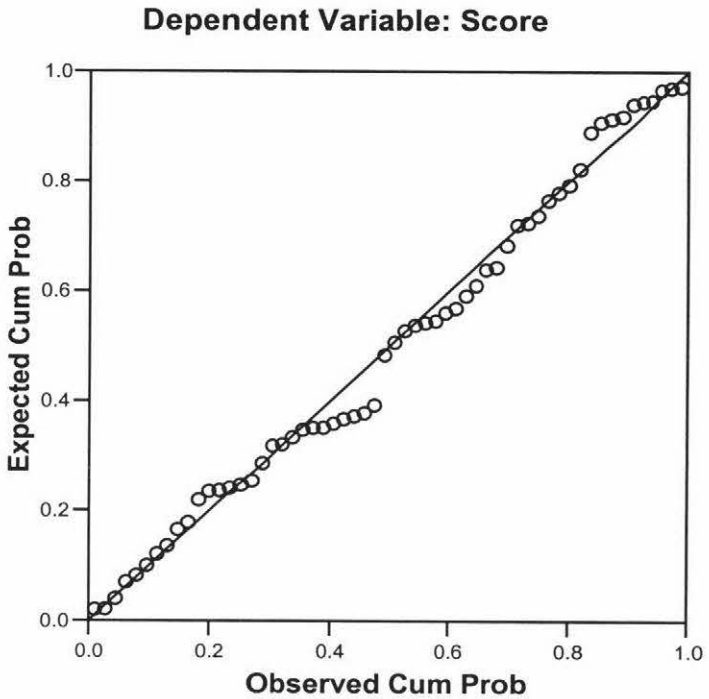


Figure 23: Normal P-P Plot of Regression Standardized Residual

The P-P plotted residuals followed the 45-degree line. The plot of residuals by the predicted values shows that the variance of the errors doesn't increase dramatically with changing the predicted Utility. There is good scatter.

Neither the histogram nor the P-P plot indicates that the normality assumption is violated. The Score has positive relationship with Utility. The higher Utility, the higher Score.

3.8.2 Is the lower the cost, the higher the score?

Null hypothesis H0: Score and Cost don't have relationship.
Alternative hypothesis H1: The Score has correlation with Cost.

The figure below shows the scatter of score and cost with the linear fit line.

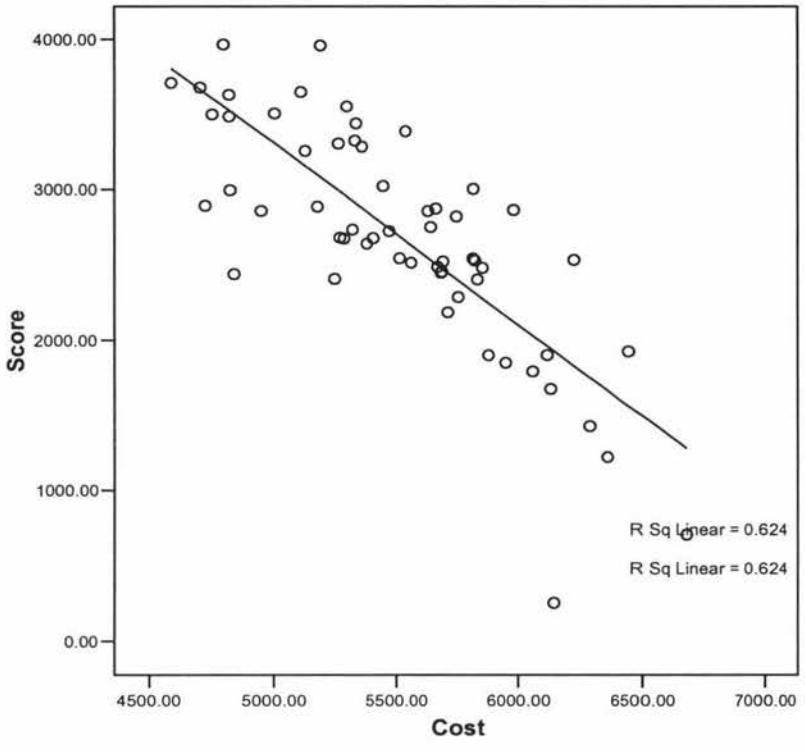


Figure 24: The graph of Scatter of Score Cost with the linear fit line

The resulting scatter plot appears to be suitable for linear regression. A scatter plot indicates that these variables are linearly related. These variables have a negative relationship. As the Cost of the Riskerwise agent increases, the Score deceased.

Further study the coefficients table, which shows the coefficients of the regression line.

Model		Unstandardized Coefficients		Standardized Coefficients	T	Sig.
		B	Std. Error	Beta		
1	(Constant)	9359.273	694.990		13.467	.000
	Cost	-1.210	.126	-.790	-9.631	.000

Table 15: Coefficients of Score and Cost. Score is a Dependent Variable.

The model is linear because increasing the value of the Cost by 1 unit decreases the value of the Score by 1.21 units.

The linear regression model assumes that there is a linear, or "straight line," relationship between the dependent variable and each predictor. This relationship is described in the following formula. The regression equation is

Score = 9359.273 - 1.21*Cost

It states that the expected Score is equal to 9359.273 - 1.21*Cost. The score decreases while the cost increases. If the Cost is 5000, the predicted score would be 9359.273 - 1.21*5000 = 3309.273.

The ANOVA table tests the acceptability of the model from a statistical perspective.

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	19766630.146	1	19766630.146	92.756	.000(a)
	Residual	11933749.087	56	213102.662		
	Total	31700379.233	57			

Table 16: The ANOVA table for Score as a dependent variable and Cost as a predictor

The regression and residual sums of squares indicates that more than half of the variation in score is explained by the Utility of the model.

The significance value is less than 0.01, which means that the variation explained by the model is not due to chance. Null Hypothesis is rejected. The Score have correlation with Cost.

The model summary table reports the strength of the relationship between the model and the dependent variable.

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.790(a)	.624	.617	461.63044

Table 17: Model Summary for Score as a dependent variable and Cost as a predictor

It shows that 62.4% of the variation Score explained by Cost in the model. The residual for a given Cost is the observed value of the error term for that Utility. A histogram or P-P plot of the residuals will help to check the assumption of normality of the error term.

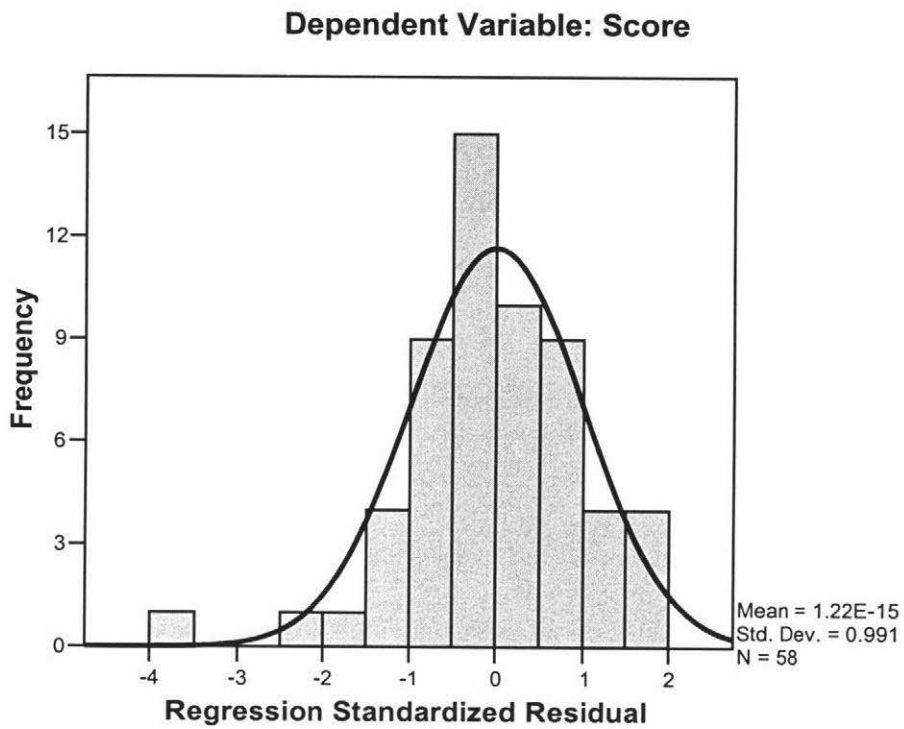


Figure 25: The histogram of the residuals

The shape of the histogram should approximately follow the shape of the normal curve. This histogram is acceptably close to the normal curve.

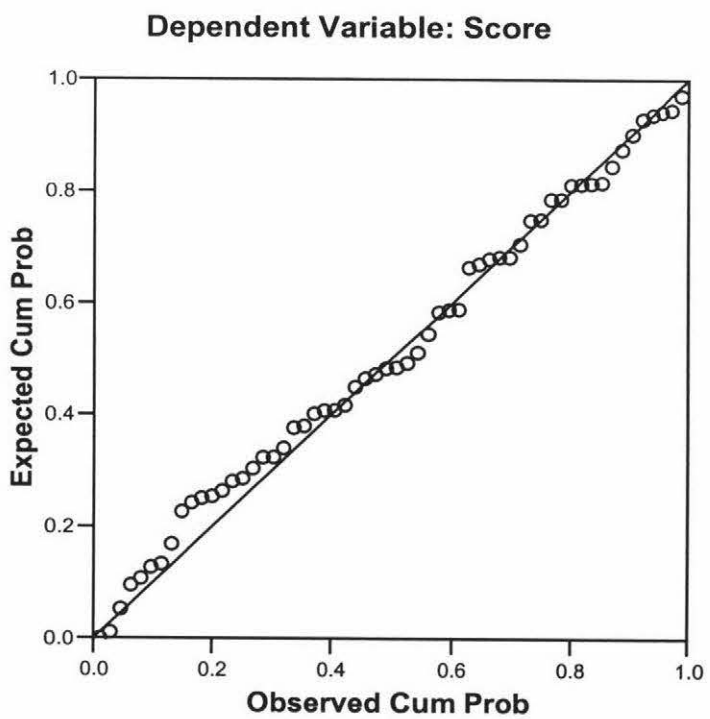


Figure 26: Normal P-P Plot of Regression Standardized Residual

The P-P plotted residuals followed the 45-degree line. The plot of residuals by the predicted values shows that the variance of the errors doesn't change dramatically with changing the predicted Cost. There is good scatter.

Neither the histogram nor the P-P plot indicates that the normality assumption is violated. The Score has negative relationship with Cost. The higher Cost, the lower Score.

3.9 Summary

The purpose of designing the RiskerWise agent is to find the strategy for agents playing against several open-loop greedy agents in the TAC game. This chapter explained the design of the RiskerWise agent in detail. The performance of RiskerWise agent was displayed. The game data was used to illustrate the relationship among score, utility and cost: The higher the utility and the lower the cost, the higher the score.

Because the open-loop agents cause high price of the hotel rooms, it is wise to avoid the price war. The RiskerWise agnet used clients' preference, current price and historical data to search the solution for its clients. The algorithm the RiskerWise agnet used was Breadth-First search algorithm. Because the open-loop agents don't change their decisions, the RiskerWise agnet sents buying bids once it gets the search results to avoid high flight tickets' price. It updates its bids arrocrding to the current price if the price is not great than its true value. The RiskerWise agent achieved a score of 2691.3 ± 745.8 by playing 58 games against seven aggressive DummyAgents provided by the TAC classic server.

In a highly competitive environment and with many aggressive competitors, it is hard to satisfy the clients' preferences. In order to achieve high score, the RiskerWise agent had to reach a compromise between its clients' preferences and the overall performace such as shortening its client trip and choosing cheap hotel for its clients.

Chapter 4: Using Fuzzy Logic to Predict the Hotel Price Increase

If the agents can predict hotel price or hotel price change, the agents could identify profitable auction and assembling trips for clients is much easier. Because of the random nature of the customers' preferences and the way other agents deal with their hotel bidding, uncertainties are associated with the hotel auctions. This increases the difficulties of hotel room auction price prediction. Fuzzy logic-based method has been proven to be a practicable solution in solving decision making problem under uncertainty (He and Jennings, 2004). It has been successfully used in real world applications such as manufacturing, financial trading. There are alternative techniques for handling uncertainty. For example, possibility-based approach, this relies on a case base. The dynamic nature of TAC doesn't guarantee there always have similar situations available. The Bayesian learning method has also been used to model multi-issue negotiation in a sequential decision making model based on Bayesian probbilities. The dynamic and uncertain nature of TAC makes it very difficult to assign prior probabilities of a bid (He and Jennings, 2004).

In this chapter, the general knowledge of Fuzzy Logic will be introduced in the following sections and the specifics of the Fuzzy Logic hotel price prediction system design will be discussed in detail.

4.1 The basic concepts of Fuzzy Logic

A fuzzy set describes vague concepts. It does not have a clearly defined boundary. It can have elements with only a partial degree of membership. The degree an object belongs to a fuzzy set is represented by a membership value between 0 and 1.

Fuzzifying inputs is the process which converts all fuzzy statements in the antecedent to a degree of membership between 0 and 1.

Reasoning in fuzzy logic generalizes Boolean logic. If "true" has the numerical value of 1 and "false" has the numerical value of 0, fuzzy logic could have values like 0.9 or 0.15.

A membership function maps input values to their appropriate membership values (or degree of membership), which is between 0 and 1 (Negnevitsky, 2005).

$A = \{x, \mu_A (x) \mid x \in X\}$

$\mu_A (x)$ is called the membership function (MF) of x in A. The membership function maps each element of X to a membership value between 0 and 1 (Negnevitsky, 2005).

Fuzzy rules operate using a series of if-then statements. For example, if X then P, where P is set of X. The order of those rules is not important because all rules are evaluated in parallel.

Fuzzy inference is a method that interprets the values in the input space and assigns values to the output space by using fuzzy logic (Negnevitsky 2005).

A typical process in developing the fuzzy logic system includes the following steps (Negnevitsky 2005):

1. Specify the problem and define variables
2. Determine fuzzy sets
3. Elicit and construct fuzzy rules
4. Encode the fuzzy sets, fuzzy rules and procedures to perform fuzzy inference into the system
5. Evaluate and tune the system

4.2 Developing the hotel room price prediction system

4.2.1 Specify the problem and define variables

The prices of Tampa Towers and Shoreline Shanties hotel rooms are not available before the hotel auction closes. If the agents could predict the hotel price changes, agents could choose appropriate approach to bid goods. Although the strategies of the competitors play an important role in affecting the price, a good game design needs to ensure externalities, which means that game players don't know other's strategies. Assuming the agent doesn't know the strategies of its competitors. There are some information is available during the game. The available information for an agent include the information of the current ask price of Tampa Towers hotel room auction, the change of the ask price of Tampa Towers hotel room auction, the current ask price of Shoreline Shanties hotel room auction, the change of the the current ask price of Shoreline Shanties hotel room auction, time period since counterpart auction closed and the game time. All these information could be used to predict the hotel price changes. For example, if the agent wants to get the hotel room on day 2 for its clients and both of the Tampa Towers hotel room and Shoreline Shanties hotel room auctions have not been closed, there are more chances for agents to get hotel room on day 2. So the price increase should be slow. If Shoreline Shanties hotel room auction on day 2 has closed, there less chance for the agent. The price increase should be fast.

The factors might be used to predict the hotel price changes are listed below:

- Current ask price of Tampa Towers hotel (PT2)
- Ask price increase of Tampa Towers hotel in previous minute (TC2)
- Current ask price for Shoreline Shanties hotel (PS2)
- Ask price increase of Shoreline Shanties hotel in previous minute (SC2)
- Current time (time)
- Time period since counterpart auction closed (Sclose)

Linguistic Variable: PT2 when SSClosed	
Linguistic value	Numerical range
Low	[0, 50]
Medium	[50, 150]
High	[80, 250]
Linguistic Variable: PT2 when CounterpartNotClosed	
Linguistic value	Numerical range
Low	[0, 30]
Medium	[30, 100]
High	[80, 200]
Linguistic Variable: PS2 when TTClosed	
Linguistic value	Numerical range
Low	[0, 50]
Medium	[50, 300]
High	[400, 1000]
Linguistic Variable: PS2 when CounterpartNotClosed	
Linguistic value	Numerical range
Low	[0, 30]
Medium	[10, 90]
High	[50, 150]
Linguistic Variable: TC2 when SSClosed	
Linguistic value	Numerical range
Slow	[0, 50]
Fast	[5, 150]
Linguistic Variable: TC2 when CounterpartNotClosed	
Linguistic value	Numerical range
Slow	[0, 50]
Fast	[5, 100]
Linguistic Variable: SC2 when TTClosed	
Linguistic value	Numerical range
Slow	[0, 5, 20, 50]
Fast	[15, 30, 50, 100]
Linguistic Variable: SC2 when CounterpartNotClosed	
Linguistic value	Numerical range
Slow	[0, 50]
Fast	[20, 80]
Linguistic Variable: Time when TTClosed	
Linguistic value	Numerical range
Early	[40, 50]

Late	[45, 55]
Linguistic Variable: Time when CounterpartNotClosed	
Linguistic value	Numerical range
Early	[40, 50]
Late	[45, 55]
Linguistic Variable: Time when SSClosed	
Linguistic value	Numerical range
Early	[0, 45]
Late	[30, 59]

Table 18: Linguistic Variables and their ranges

Output: The bid increase amount is constant

- Small: 10
- Medium: 50
- Big: 100

4.2.2 Determine fuzzy sets

Fuzzy sets can have a variety of shapes. The Gaussian membership function, s shape membership function can provide an adequate representation of the system and they have characteristic of smoothness.

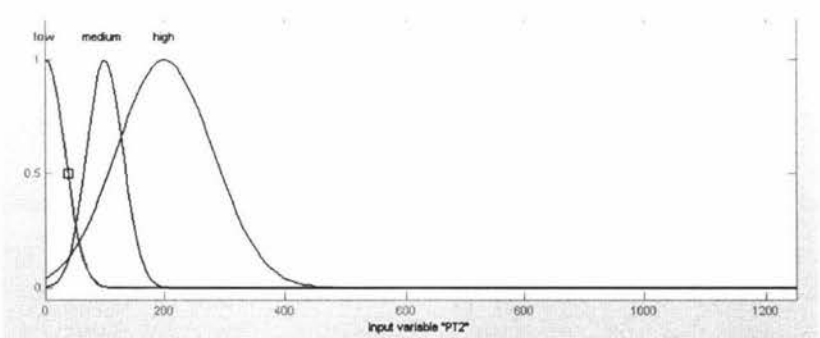


Figure 27: Fuzzy sets of PT2 when Counterpart auction has not been closed.

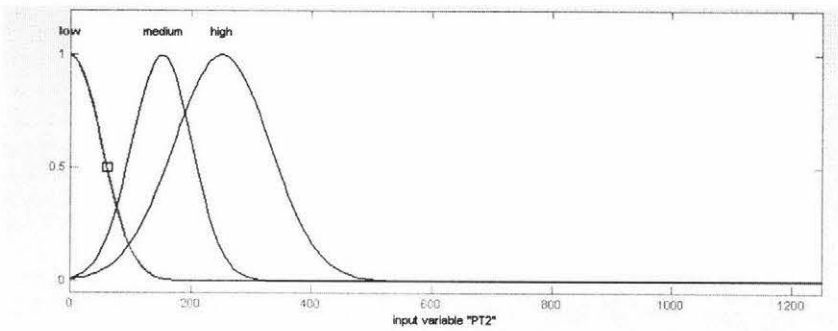


Figure 28: Fuzzy sets of PT2 when Counterpart auction SS2 has been closed

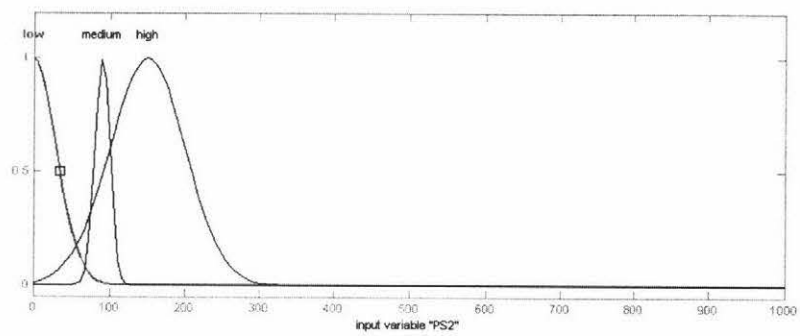


Figure 29: Fuzzy sets of PS2 when Counterpart auction TT2 has not been closed.

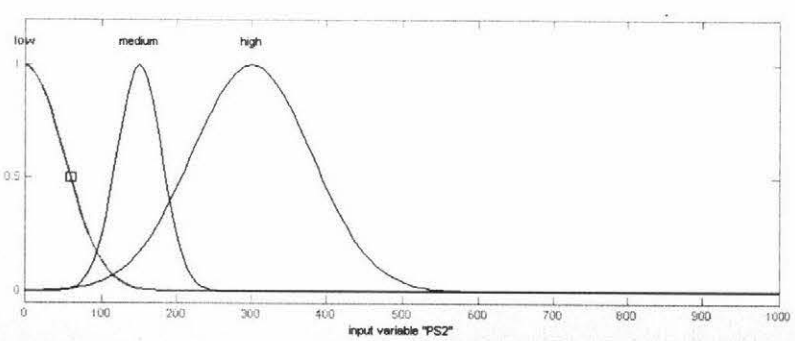


Figure 30: Fuzzy sets of PS2 when Counterpart auction TT2 has been closed.

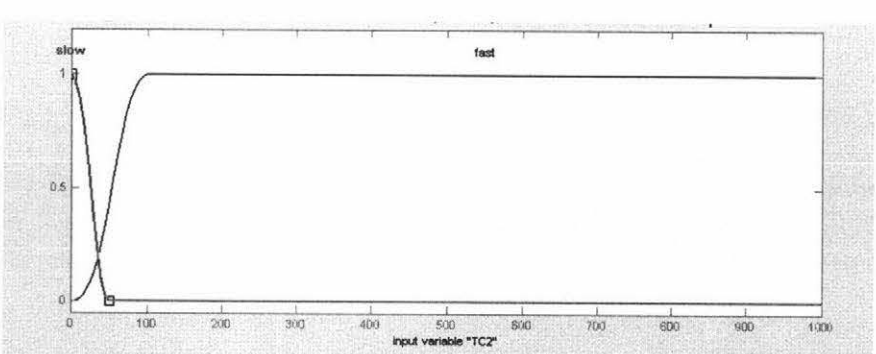


Figure 31: Fuzzy sets of TC2 when Counterpart auction SS2 has not been closed

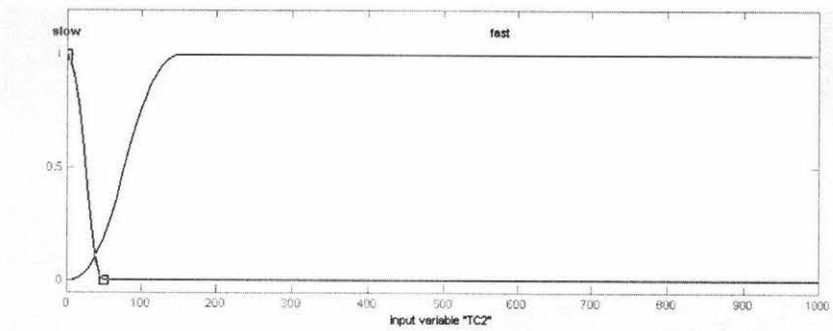


Figure 32: Fuzzy sets of TC2 when Counterpart auction SS2 has been closed

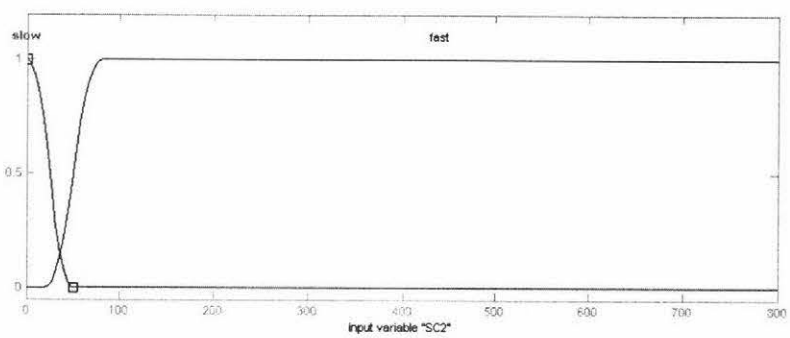


Figure 33: Fuzzy sets of SC2 when Counterpart auction TT2 has not been closed

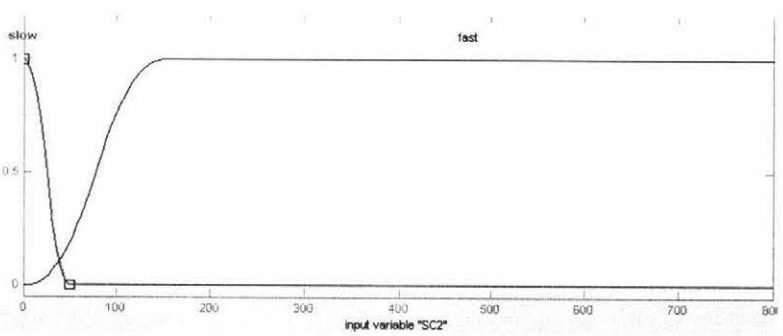


Figure 34: Fuzzy sets of SC2 when Counterpart auction TT2 has been closed

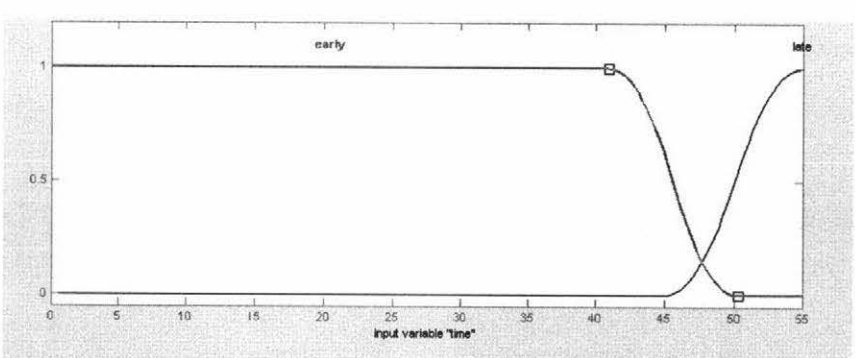


Figure 35: Fuzzy sets of time when counter part of auction has not been closed

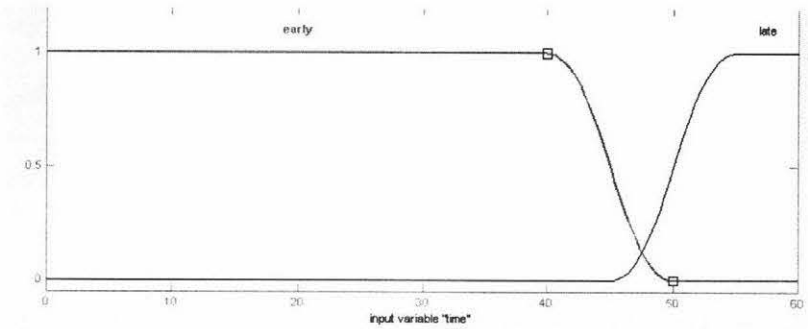


Figure 36: Fuzzy sets of time when SS has been closed

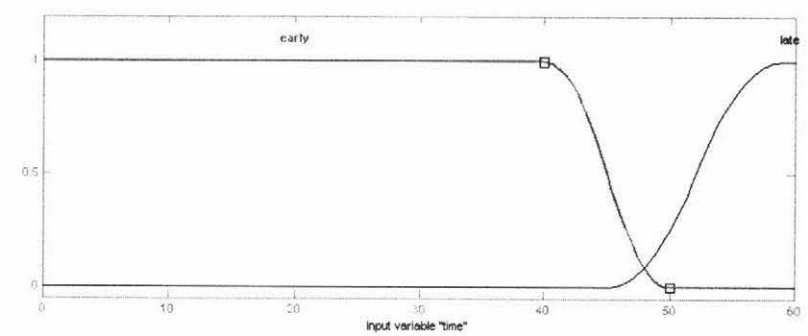


Figure 37: Fuzzy sets of time when TT has been closed

4.2.3 Elicit and construct Fuzzy Rules

The starting point is to find the relationship between the inputs and the outputs. For example, if both the Tampa Towers hotel room auction and Shoreline Shanties hotel room auction have not been closed, both of the ask price of the Tampa Towers hotel room auction and Shoreline Shanties hotel room auction are low and the time is early, the agent only need to increase the small bid price to keep the opportunity to win the hotel room. If Shoreline Shanties hotel room auction has been closed, the the ask price increase of the Tampa Towers hotel room auction is fast and time is late, there has less chance for the agent to get the hotel room. The agent needs to increase the bid price big to win the hotel room auction. If Tampa Towers hotel room auction has closed, the the ask price increase of the Shoreline Shanties hotel room auction is slow and time is late, the agent needs to increase medium bid price to win the hotel room auction. Because the winning of the Shoreline Shanties hotel room will not gain the bonus for the agent, the

price of Shoreline Shanties hotel room is normally lower than the the price of Tampa Towers hotel room.

There are 96 possible relationships in total. They are the fuzzy rules. All the possible rules are listed in Appendix 16.

The rules can be simplified. For example, the following three rules can be replaced by: IF TC2 is slow and time is early THEN bid price increase is small.

1. IF PT2 is medium and TC2 is slow and time is early THEN bid price increase is small.
2. IF PT2 is low and TC2 is slow and time is early THEN bid price increase is small.
3. IF PT2 is high and TC2 is slow and time is early THEN bid price increase is small.

The following two rules can be replaced by: IF PT2 is not low and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

1. IF PT2 is medium and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

2. IF PT2 is high and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

After combining and simplifying the rules, the rules are reduced to 40:

- When SS2 has closed:

1. IF TC2 is slow and time is early THEN bid price increase is small.
2. IF TC2 is fast and time is late THEN bid price increase is big.
3. TC2 is slow and time is late THEN bid price increase is medium
4. TC2 is fast and time is early THEN bid price increase is medium

- When TT2 has closed:

1. SC2 is slow and time is early THEN bid price increase is small.
2. SC2 is slow and time is late THEN bid price increase is medium
3. SC2 is fast and time is early THEN bid price increase is medium
4. SC2 is fast and time is late THEN bid price increase is big.

- When SS2 and TT2 has not yet closed:

1. IF TC2 is slow and PS2 is low and SC2 is slow and time is early THEN bid price increase is small.
2. TC2 is slow and PS2 is low and SC2 is slow and time is late THEN bid price increase is medium.
3. IF PT2 is not low and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.
4. IF PT2 is low and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is small.
5. TC2 is slow and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.
6. TC2 is fast and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.
7. TC2 is fast and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.
8. TC2 is fast and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.
9. TC2 is fast and PS2 is low and time is late THEN bid price increase is big.
10. TC2 is fast and PS2 is medium and SC2 is slow and time is early THEN bid price increase is medium.
11. TC2 is fast and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.
12. TC2 is fast and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

13. TC2 is fast and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.
14. TC2 is slow and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.
15. TC2 is slow and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.
16. TC2 is fast and PS2 is low and SC2 is slow and time is early THEN bid price increase is medium.
17. IF PT2 is not low and TC2 is slow and PS2 is high and time is late THEN bid price increase is big.
18. IF PT2 is low and TC2 is slow and PS2 is high and time is late THEN bid price increase is medium.
19. IF PT2 is low and TC2 is slow and PS2 is not low and SC2 is slow and time is late THEN bid price increase is small.
20. IF PT2 is high and TC2 is slow and PS2 is not low and SC2 is slow and time is early THEN bid price increase is medium.
21. IF PT2 is low and TC2 is slow and PS2 is not high and SC2 is fast and time is late THEN bid price increase is medium.
22. IF PT2 is medium and TC2 is slow and PS2 is not high and SC2 is fast and time is early THEN bid price increase is big.
23. IF PT2 is medium and TC2 is slow and PS2 is medium and SC2 is slow and time is early THEN bid price increase is small.
24. IF PT2 is medium and TC2 is slow and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.
25. IF PT2 is high and TC2 is slow and PS2 is low and SC2 is fast and time is late THEN bid price increase is medium.
26. IF PT2 is low and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.
27. IF PT2 is low and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.
28. IF PT2 is medium and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is big.

29. IF PT2 is medium and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is big.
30. IF PT2 is high and TC2 is slow and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.
31. IF PT2 is high and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is big.
32. IF PT2 is high and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is big.

4.2.4 Encode the fuzzy sets, fuzzy rules and procedures

The fuzzy sets, fuzzy rules and procedures need to be encoded to perform fuzzy inference into the system after fuzzy sets and fuzzy rules are defined. To accomplish this task, the fuzzy logic system could be built by using a programming language such as C or to apply a fuzzy logic development tool such as MATLAB Fuzzy Logic Toolbox from the MathWorks. MATLAB Fuzzy Logic Toolbox provides complete environments for building and testing fuzzy logic system. It has five integrated graphical editors: the fuzzy inference system editor, the rule editor, the membership function editor, the fuzzy inference viewer, and the output surface viewer. These features make designing fuzzy systems much easier. For rapid developing and prototyping the fuzzy logic system, MATLAB Fuzzy Logic Toolbox was chosen.

4.2.5 Evaluate the system

If the predicted price is not very higher than actual price, the agent will not suffer since the agent will not change its plan and it only pays the 16^{th} price. If the predicted price is much higher than actual price, the agent will not suffer if the agent will not change its plan but the agent may suffer if the agent changes its plan. If the predicted price is lower than actual price, the agent will suffer since the agent will not change its plan and it may lose the hotel auction. The predicted price is a bit higher than actual price is ideal. There are three different situation will be tested by this Fuzzy logic model.

- When the counter part has not been closed
- When Shoreline Shanties hotel auction has been closed
- When Tampa Towers hotel auction has been closed

In order to test this Fuzzy Reasoning model, some real game data was recored.

4.2.5.1 When the counter part has not been closed

17610	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
TT Day 3	1	134	142	147	172	175	Close	
SS Day 3	200	203	214	215	253	312	312	Close

Table 19: TAC game 17610 Hotel Auction on Day 3 data

The fuzzy logic system predicting the bid increase in TAC game 17610 at minute 5 is shown in Appendix 17.

- The ask price of Tampa Towers hotel auction was 147
- The ask price increase of Tampa Towers hotel auction compare to the previous minute was 5
- The ask price of Shoreline Shanties hotel auction was 215
- The ask price increase of Shoreline Shanties hotel auction compare to the previous minute was 1
- Current time was 42
- The fuzzy logic system predicted the bid increase was 50

The agent could get the Tampa Towers hotel room at day 2 by increasing the bid 50 if the auction closed on minute 3. The actual price increase was 25. The fuzzy logic system over predictd the hotel price change by 25.

17622	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
SS Day 1	117	129	134	145	Close			
TT Day 1	0	1	42	49	49	close		

Table 20: TAC game 17622 Hotel Auction on day 1 data

The fuzzy logic system predicting the bid increase in TAC game 17622 at minute 3 is shown in Appendix 18.

- The ask price of Tampa Towers hotel auction was 1
- The ask price increase of Tampa Towers hotel auction compare to the previous minute was 1
- The ask price of Shoreline Shanties hotel auction was 129
- The ask price increase of Shoreline Shanties hotel auction compare to the previous minute was 12
- Current time was 55
- The fuzzy logic system predicted the bid increase was 50

The agent could get the Tampa Towers hotel room at day 2 by increasing the bid 50 if the auction closed on minute 3. The actual price increase was 41. The fuzzy logic system over predicted the hotel price change by 9.

17624	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
SS Day 1	89	89	89	89	Close			
TT Day 1	1	1	44	44	44	44	44	44

Table 21: TAC game 17624 Hotel Auction on day 1 data

The fuzzy logic system predicting the bid increase in TAC game 17624 at minute 3 is shown in Appendix 19.

- The ask price of Tampa Towers hotel auction was 1
- The ask price increase of Tampa Towers hotel auction compare to the previous minute was 0

- The ask price of Shoreline Shanties hotel auction was 89
- The ask price increase of Shoreline Shanties hotel auction compare to the previous minute was 0
- Current time was 55
- The fuzzy logic system predicted the bid increase was 50

The agent could get the Tampa Towers hotel room at day 1 by increasing the bid 50 if the auction closed on minute 3. The actual price increase was 43. The fuzzy logic system over predicted the hotel price change by 7.

4.2.5.2 When Shoreline Shanties hotel auction has been closed

The fuzzy logic system predicting the bid increase in TAC game 17622 at minute 5 is shown in Appendix 20.

- The ask price of Tampa Towers hotel auction was 49
- The ask price increase of Tampa Towers hotel auction compare to the previous minute was 7
- Current time was 45
- The fuzzy logic system predict the bid increase was 30

The agent could get the Tampa Towers Hotel room at day 1 by increasing the bid 30 when the auction closed on minute 5. The actual price increase was 0. The fuzzy logic system over predicted the hotel price change by 30. The fuzzy logic model doesn't take travel day as input variable. Travel day is an important factor which affects the hotel price. Day 2 and day 3 are more competitive than day 1 and day 4.

17614	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
SS Day 2	148	171	177	186	186	Close		
TT Day 2	0	94	131	141	145	145	145	Close

Table 22: TAC game 17614 Hotel Auction on Day 2 data

The fuzzy logic system predicting the bid increase in TAC game 17614 at minute 7 is shown in Appendix 21.

- The ask price of Tampa Towers hotel auction was 145
- The ask price increase of Tampa Towers hotel auction compare to the previous minute was 0
- Current time was 45
- The fuzzy logic system predict the bid increase was 30

The agent could get the Tampa Towers Hotel room at day 1 by increasing the bid 30 when the auction closed on minute 7. The actual price increase was 0. The fuzzy logic system over predicted the hotel price change by 30. The fuzzy logic model just takes the the price change in the previous minute. It doesn't take the whole price change history into consideration. If the fuzzy logic model had all the price change data, it could predict the hotel price change more precise.

4.2.5.3 When Tampa Towers hotel auction has been closed

17612	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
SS Day 2	218	220	221	272	275	292	Close	
TT Day 2	0	1	132	132	Close			

Table 23: TAC game 17612 Hotel Auction day 2 data

The fuzzy logic system predicting the bid increase in TAC game 17612 at minute 6 is shown in Appendix 22.

- The ask price of Shoreline Shanties hotel auction was 275
- The ask price increase of Shoreline Shanties hotel auction compare to the previous minute was 3
- Current time was 59
- The fuzzy logic system predict the bid increase was 30

The agent could get the Shoreline Shanties room at day 2 by increasing the bid 30 when the auction closed on minute 6. The actual price increase was 17. The fuzzy logic system over predicted the hotel price change by 13.

17610	Min. 1	Min. 2	Min. 3	Min. 4	Min. 5	Min. 6	Min. 7	Min. 8
SS Day 2	193	193	Close					
TT Day 2	1	close						

Table 24: TAC game 17610 Hotel Auction on Day 2 data

TAC game 17610 at minute 2 is shown in Appendix 23.

- The ask price of Shoreline Shanties hotel auction was 193
- The ask price increase of Shoreline Shanties hotel auction compare to the previous minute was 193
- Current time was 45
- The fuzzy logic system predicted the bid increase was 30

The agent could get the Shoreline Shanties room at day 2 by increasing the bid 30 when the auction closed on minute 2. The actual price increase was 0. The fuzzy logic system over predicted the hotel price change. The fuzzy logic system could adjust the output according to the different hotels.

4.2.6 Ideas of improving the system

The prices pridicted by the fuzzy logic prediction system were generally higher than the actual data. This could be improved by adjust the rule execution weights. The force of rule can be reduced by adjusting its weight. For example, if I specify

> TC2 is fast and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big * (0.8).

Then the rule's force will be reduced by 20 per cent.

The range of the output variable also could be modified. The use of narrow fuzzy sets may increase the fuzzy system's performance.

The fuzzy logic prediction system could have more model input variables. For example, the types of competitors, types of day, types of hotels and the price change history starting from minute 1 to the current time.

4.3 Summary

In this chapter, the basic concepts of fuzzy logic was introduced and the actual main processes of developing the fuzzy logic price prediction system for hotel room auction in TAC game were described.

In order to establish bid price prediction for the hotel room auction in the TAC game, a set of fuzzy rules are constructed and inference is provided by fuzzy logic reasoning. There are six linguistic variables were defined: the ask price of Tampa Towers hotel auction, the ask price increase of Tampa Towers hotel auction compare to the previous minute, the ask price of Shoreline Shanties hotel auction, the ask price increase of Shoreline Shanties hotel auction compare to the previous minute, current time, time period since counterpart auction closed. There were 40 fuzzy rules constructed. The knowledge base for the predictor is established from observation of the TAC games and the general knowledge about the game. The fuzzy logic prediction system was built by using MATLAB Fuzzy Logic Toolbox, a fuzzy logic development tool.

While the test results presented in this work show some good results of fuzzy logic approach, the fuzzy logic prediction system designed here is somewhat simplistic. It needs to be tuned. There are improvements needed to be done before this fuzzy logic system could be used in the real game. A list of thoughts was given to improve the system including adding more input and output variables, fuzzy sets, new rules, adjust the rule execution weights, etc. A practical system needs to consider all the possible criteria, including the types of competitors.

Chapter 5: Conclusion and Future Development

5.1 Overall conclusions

The Internet is experiencing an exponential growth. But searching, accessing, filtering and integrating information is not easy for a person or current computer systems. Software agent is expected to be the solution. Trading agent is one kind of the software agents. The interests have growing because of the development and success of electronic commerce. Trading agents are more suitable for e-business comparing with humans in the situation when the decisions are based on an enormous amount of data or using complex strategies. Online auction is a successful exmple of e-businesses. It has become very popular. In order to understand effectiveness of agent strategies, the possible influences of trading agents to online auctions and also to stimulate research in trading agents with an emphasis on developing a successful strategy for maximizing profits in a constrained environment, the Trading Agent Competition (TAC) game was designed. TAC provides infrastructure and directs researchers' energy to a common problem leading to cooperation and promoting originality and creativity.

This thesis provided an overview of the intelligent software agents and a close investigation of TAC game and the strategies used in the TAC.

From an autioneer's point of view, TAC design is a good design. The goal of an auctioneer is to make maximum profit. Collusion between bidders could hinder an auctioneer achieving this goal. Preventing collusion is one of the important aspects of TAC design. TAC closes randomly chosen hotel room auction. This strategy prevents last-minute collusion. TAC doesn't provide enough hotel rooms especially in day 2 and day 3. This strategy prevents low-price collusion. Externalities between bidders in TAC make collusion harder.

TAC game design is complicated. The test designed in this thesis shows that the TAC game needs to be improved by adding the functions to detect and stop abnormal

activity. Using anonymous names of trading agents in the TAC could be one of the answers. It makes collusion even harder for agents.

TAC is not a real e-commerce setting. For example, the infinite supply of the flight tickets and the periodic closing of randomly selected hotel auctions are not typically seen in real world market mechanisms. It is not usual for the related items to be sold in different auctions in the real on-line auction. The design of periodic closing of randomly selected hotel auctions in TAC prevents the last minutes bidding, which is common problem in the online auction. This idea could be adopted by the real online auctions if the auctioneers want to increase the profit. Humans are normally interested in buying many items at a time (Sardinha, et al., 2005). In some cases, goods can have a high interdependency, such as flight ticket and hotel room. What is the point to buy hotel room without a flight ticket? With the development of online auction, it is possible to build more combined auctions. TAC provides a valuable framework around which could eventually be deployed into commercial settings. The design of TAC has succeeded.

TAC has stimulated successful strategies, it incorporates many technologies. It is challenging to design a software agent for the TAC. In order to find the strategy for playing against seven open-loop greedy agents in the TAC game, the practical agent called RiskerWise agent was designed after investigating the strategies used in TAC. In a highly competitive environment and with many aggressive competitors, it is hard to satisfy the clients' preferences. The open-loop agents cause all the players in the market to suffer. Because the open-loop agents tend to bid high price for the hotel rooms in TAC, it is wise for the RiskerWise agent to avoid the price war. In order to achieve high score, the RiskerWise agent had to reach a compromise between its clients' preferences and the overall performace such as shortening its client trip and choosing cheap hotel for its clients. The RiskerWise agent won most of the top scores of the games in which it participated. The RiskerWise agent achieved a score of 2691.3 ± 745.8 by playing 58 games with seven aggressive DummyAgents. The RiskerWise agnet used Breadth-First search algorithm to search the solution for its clients. The information the RiskerWise agnet used include clients' preference, current price and historical data. It sents buying

bids once it gets the search results to avoid high flight tickets' price. It updates its bids arrocrding to the current price if the price is not great than its true value.

In order to predict bid price change of the hotel room auction in the TAC, a set of fuzzy rules are constructed and inference is provided by fuzzy logic reasoning. There are six linguistic variables were defined: the ask price of Tampa Towers hotel auction, the ask price increase of Tampa Towers hotel auction compare to the previous minute, the ask price of Shoreline Shanties hotel auction, the ask price increase of Shoreline Shanties hotel auction compare to the previous minute, current time, time period since counterpart auction closed. There were 40 fuzzy rules were constructed. The knowledge base for the predictor is established from observation of the TAC games and the general knowledge about the game. The fuzzy logic prediction system was built by using MATLAB Fuzzy Logic Toolbox, a fuzzy logic development tool.

While the test results presented in this work show some good results of fuzzy logic approach, the fuzzy logic prediction system designed here is somewhat simplistic. It needs to be tuned.

5.2 Future development

There are improvements needed to be done for the hotel room price change prediction fuzzy logic system. The future work includes:

- Adding more input variables
    - Types of competitors
    - Types of day
    - Types of hotels
    - The price change history starting from minute 1 to the current time
- The range of the output variable also could be modified.
- Adding more output variables

- Adding more fuzzy sets,
- Adding more new rules
- Adjust the rule execution weights

References

Anderberg, A (2003). *History of the Internet and Web*. Retrieved December 1, 2004, from http://www.anderbergfamily.net/ant/history/

Arrow, K. J. and Hahn, F. I. (1971). General Competitive Analysis, *Mathematical Economics Texts*. Holden-Day, inc., San Francisco, Volume 6, 1971.

Aurell, E., Boman, M., Carlsson, M., Eriksson, J., Finne, N., Janson, S., Kreuger, P., and Rasmusson, L. (2002). A trading agent built on constraint programming. In *Eighth International Conference of the Society for Computational Economics: Computing in Economics and Finance*, Aix-en-Provence.

Boadway, J. and Precup, D. (2001). Reinforcement learning applied to a multiagent system. Presentation at TAC Workshop.

Bose, P., Maheshwari, A., & Morin, P. (2002). Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry: Theory and Applications, 24*, 135–146.

Boyan, J., & Greenwald, A. (2001). Bid determination in simultaneous auctions: An agent architecture. In *Third ACM Conference on Electronic Commerce*, pp. 210–212, Tampa, FL.

Bradshaw, J. M. (Ed.) (1997) Software Agents, *AAAI Press/The MIT Press,* p 8

Braziunas, D., Chalkiadakis, G., Hyafi, N., Poupart, P., Ross, D. Boutilier, C. (2002). *TOMAhack Trading Agent '02.* Department of Computer Science, University of Toronto

Brustoloni, J.C. (1991). *Autonomous agents: characterization and requirements,* Carnegie Mellon technical report CMU-CS-91-204. Pittsburgh:

Carnegie Mellon University.

Caillaud, B. and Jehiel, P. (1998) Collusion in Auctions with Externalities. *Rand journal of Economics*(29), 680-702.

Cheng, S.-F., Leung, E., Lochner, K. M., O'Malley, K., Reeves, D. M., and Wellman, M. P. (2005). Walverine: A Walrasian trading agent. *Decision Support Systems*, 39:169-184.

Coen. M. (1995). *SodaBot: A Software Agent Environment and Construction System.* Massachusetts Institute of Technology. Retrieved July 7, 2004 from ftp://publications.ai.mit.edu/ai-publications/pdf/AITR-1493.pdf

D. C. Smith, A. Cypher and J. Spohrer (1994) KidSim: programming agents without a programming language, in *Communications of the ACM*, 37(7), pp. 54-67.

Das, R., Hanson, J. E., Kephart, J. O.and Tesauro., G. (2001) Agent-human interactions in the continuous double auction. In *Seventeenth Inter-national Joint Conference on Artificial Intelligence*, pages1169–1176, Seattle, WA, 2001.

Dijkstra, Edsger W. (2001). The End of Computing Science? *Communications of The ACM, Vol. 44* (No. 3).

Ding, L., Finin, T., Dhi, Y. M., Zou, Y.Y., Ding, Z. L., and Pan, R. (2003). Strategies and Heuristics Used by the UMBCTAC Agent in the third Trading Agent Competition. *IJCAI-03 Workshop on Trading Agent Design and Analysis,* Acapulco, 2003

Ding, L., Shi, Y., Ding, Z. L., Pan, R. and Finin, T. (2002) *UMBCTAC: A Balanced Bidding Agent*, UMBC Technical Report TR-02-15, 2002.

Dixit and Skeath (1999). *Games of Strategy*, Norton, W. W. & Company, Inc

eBay (2004) Retrieved January 1, 2005 from
http://www.ebay.com/

Eriksson, J. and Janson, S. (2002). The Trading Agent Competition - TAC 2002. *ERCIM News*, 51, October 2002.

Fama, E.F (1969). Efficient Capital Markets: A Review Of Theory And Empirical Work. *The Journal of Finance Efficient Market.* Vol.25, Issue 2. pp. 383-417.

Fornara, N., and Gambardella, L. M. (2001). An autonomous bidding agent for simultaneous auctions. *Fifth International Workshop on Cooperative Information Agents*, number 2182 in Lecture Notes on Artificial Intelligence, 130–141.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1993). *AMPL: A Modeling Language for Mathematical Programming*. Boyd & Fraser.

Franklin.S et.al (1996). Is it a Agent or Just a program?: A taxonomy for autonomous agents, *Proceeding of the Third International workshop on Agent theories, Architectures and Languages*, Springer-Verlag,.

Friedman, D. and Rust, J., editors (1993). *The Double Auction Market: Institutions, Theories, and Evidence.* Addison-Wesley.

Fritschi, C., and Dorer, K. (2002). Agent-oriented software engineering for successful TAC participation. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, Bologna.

Gadomski, A. M. (1998) *Agents and Intelligence*, Retrieved on December 24, 2002 from http://erg4146.casaccia.enea.it/wwwerg26701/gad-ag.html

Game Overview (n.d.) TAC , Retrieved February 10, 2004 from
http://www.sics.se/tac/page.php?id=3

Genetic algorithm (n.d.) Retrieved Judy 26, 2004 from http://www.stanford.edu/~buc/SPHINcsX/bkhm15.htm

Greenwald, A. & Stone, P. (2001). Autonomous Bidding Agents in the Trading Agent Competition, *IEEE Internet Computing*, March-April 2001

Greenwald, A. (2002). The International Trading Agent Competition: Focus on RoxyBot. *Computing Research News.*

Greenwald, A. (2003). The 2002 trading agent competition: An overview of agent strategies. *AI Magazine*, 24(1), 83–91.

Greenwald, A. (2003). Bidding under uncertainty in simultaneous auctions. In *IJCAI-03 Workshop on Trading Agent Design and Analysis*, Acapulco.

Greenwald, A., and Boyan, J. (2001). Bidding algorithms for simultaneous auctions: A case study. In *Third ACM Conference on Electronic Commerce*, 115–124. Tampa, FL.

Greenwald, A., and Boyan, J. (2004). Bidding under uncertainty: Theory and experiments. In *Twentieth conference on Uncertainty in Artificial Intelligence,* pages 209-216,Banff.

Guilfoyle, C., (1995). *Vendors of Agent Technology*, UNICOM Seminar on Intelligent Agents and their Business Applications, 8-9 November, London,

Harmon, P. (Ed) (1995). Software agents. Intelligent software strategies, 11(1), 1-13

Hayes-Roth, B., and Larsson, J. E. (1996) A Domain-Specific Software Architecture for a Class of Intelligent Monitoring Agents. *Journal of Experimental and Theoretical Artificial Intelligence*(8), 149-171

He, M. H. and Jennings, N. R. (2002). SouthamptonTAC: Designing a successful trading agent. *In Fifteenth European Conference on Artificial Intelligence*, pages 8–12, Lyon, 2002.

He, M. H. and Jennings, N. (n.d.) Strategy for SouthamptonTAC-02, Retrieved on July 24, 2004 from http://tac.eecs.umich.edu/research_reports/SouthamptonTAC-02.pdf

He, M. and Jennings, N. R. (2003). SouthamptonTAC: An adaptive autonomous trading agent. *ACM Transactions on Internet Technology*, 3, 218–235.

He, M. and Jennings, N. R. (2004). Designing a Successful Trading Agent:A Fuzzy Set Approach. *IEEE Transaction on Fuzzy Systems*, 12(3).

Healey, C. G. and Wurman, P. R. (2001). Visualizing Market Data, *IEEE Internet Computing.*

Healey, C. G., St. Amant, R., and Chang, J. (2001). Assisted visualization of e-commerce auction agents. *Graphics Interface,* 201–208.

Hildenbrand, W. and Kirman, A. P. (1976). *Introduction to Equilibrium Analysis: Variations on Themes by Edgeworth and Walras.* North-Holland Publishing Company, Amsterdam.

Hillas, J. (n.d.). *Sc Microeconomic Theory II Introduction To General Equilibrium Theory,* UNIVERSITY OF AUCKLAND. Retrieved April 28, 2004 from http://yoda.eco.auckland.ac.nz/~jhillas/616.781/ge_notes_2.pdf

Internet Domain Survey. (2004). *Internet Software Consortium.* Retrieved 25 September 2004 from http://www.isc.org

Rust, J., Miller, J. and Palmer, R. (1993). *Behavior of trading automata in a*

computerized double auction market. In Friedman and Rust [1993], pages 155–198.

Katzner, D. W. (1989). *The Walrasian Vision of the Microeconomy.* University of Michigan Press.

Klemperer, P. (1999) Auction Theory: A Guide to the Literature. *Journal of Economic Surveys*, Vol. 13(3), July 1999 pp. 227-286

Lanzi, P. L. and Strada, A. (2002). A Statistical Analysis of the Trading Agent Competition 2001, In *SIGecom Exchanges* 3(2):1-8,2002

Laffont, J.-J. (1987): Externalities. In: J. Eatwell, M. Millgate & P. Newman (eds.): *The New Palgrave: A Dictionary of Economics.* London and Basingstoke: Macmillan

Machine learning. (n.d.). AAAI. Retrieved 14 Jun 2004 from http://www.aaai.org/AITopics/html/machine.html

Maes, P. (1995) Artificial Life Meets Entertainment: Lifelike Autonomous Agents, *Communications of the ACM* 38, No. 11, 108-111.

Marczyk, A. (2004) *Genetic Algorithms and Evolutionary Computation.* Retrieved April 28, 2004 from http://www.talkorigins.org/faqs/genalg/genalg.html

Mas-Colell, A., Whinston, M. D., and Green, J. R. (1995). *Microeconomic Theory.* Oxford University Press, New York.

Maskin, E. S. and Riley, J. G. (1999). Asymmetric Auctions. *Review of Economic Studies*, forthcoming.

Mitchell, T.M. (1997), *Machine Learning*, Boston: WCB/McGraw Hill.

Negnevitsky, M. (2005). Artificial Intelligence-A Guide to Intelligent Systems, Addison - Wesley.

Negroponte, N (1997). Being Digital. Alfred A. Knopf, Inc.

Norman Sadeh, Raghu Arunachalam, Joakim Eriksson, Niclas Finne, and Sverker Janson. TAC-03: A supply-chain trading competition. *AI Magazine*, 24(1):92–94, 2003.

Ozmioz, M. (2002) *To snipe or not to snipe, Do People Bid Irrationally On eBay?* Ozmioz games

PainInNEC: *A Genetic algorithm based approach*. Pennsylvania State University, NEC Research, Rutgers. Retrieved July 26, 2004 from http://tac.eecs.umich.edu/research_reports/PainInNEC02.pdf

Paul Klemperer (1999) Auction Theory: A Guide to the Literature. *Journal of Economic Surveys,* Vol. 13(3), July 1999 pp. 227-286

Peter R. Wurman, Michael P. Wellman, and William E. Walsh. A parametrization of the auction design space. *Games and Economic Behavior*, 35:304–338, 2001.

Porter, R., Nudelman, E., Chen, H., Shoham, Y., Powers, R. (2002.) *TACSMAN,* Stanford University. Retrieved June 1, 2004 from http://tac.eecs.umich.edu/research_reports/TOMAhack02.pdf

Probability density function. (n.d.) TheFreeDictionary.com. Retrieved December 20, 2004 from http://encyclopedia.thefreedictionary.com/probability%20density%20function

Putchala, R. P., Morris, V. N., Kazhanchi, R., Raman, L., & Shekhar, S. (2002). kavayaH: *A trading agent developed for TAC-02*. Tech. rep., Oracle India.

Raghu Arunachalam, et al, (2002) *The TAC Supply Chain Management Game* (Draft version 0.5).

Rajarshi Das, James E. Hanson, Jeffrey O. Kephart, and Gerald Tesauro. Agent-human interactions in the continuous double auction. In *Seventeenth International JointConference on Artificial Intelligence*, Pages 1169–1176, Seattle, WA, 2001.

Russell, S. and Norvig, P. (1995). *Artficial Intelligence: A Modern Approach.* Prentice-Hall, Saddle River, NJ.

Rust, J.; Miller, J. H.; and Palmer, R. (1994). Characterizing effective trading strategies: Insights from a computerized double auction tournament. *Journal of Economic Dynamics and Control* 18:61–96.

Sadeh, N., Arunachalam, R., Eriksson, J., Finne, N., & Janson, S. (2003). TAC-03: A supply-chain trading competition. *AI Magazine*, 24(1), 92–94.

Sardinha, J. A. R. P., Milidiú, R. L., Paranhos, P. M., Cunha, P. M., Lucena, C. J. P. (2005). An Agent Based Architecture for Highly Competitive Electronic Markets. *Eighteenth International FLAIRS Conference*, pages 326–331, Clearwater Beach, Florida.

Schapire, R. E.; Stone, P.; McAllester, D.; Littman, M. L.; and Csirik, J. A. (2002). Modeling auction price uncertainty using boosting-based conditional density estimation. In *Nineteenth International Conference on Machine Learning*. Springer-Verlag, 2002: 143-160.

Stone, P., Schapire, R. E., Csirik, J., Littman, M. L. and McAllester, D. A. (2002) ATTac-2001: A learning, autonomous bidding agent. In *Agent-Mediated Electronic Commerce IV*, volume 2153 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002: 143-160.

Stone, P. (2002). Multiagent competitions and research: Lessons from RoboCup and TAC. In *Sixth RoboCup International Symposium*, Fukuoka, Japan.

Stone, P., and Greenwald, A. (2004). The first international trading agent competition: Autonomous bidding agents. *Journal of Electronic Commerce Researc*h.

Stone, P., Schapire, R. E., Csirik, J., Littman, M. L. and McAllester, D. A. (2002) ATTac-2001: A learning, autonomous bidding agent. In *Agent-Mediated Electronic Commerce IV*, volume 2153 of *Lecture Notes in Computer Scienc*e. Springer-Verlag, 2002: 143-160.

Stone, P., Schapire, R. E., Littman, M. L., Csirik, J. A., & McAllester, D. (2003). Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research*, 19, 209–242.

Stone, P.; Littman, M. L.; Singh, S.; and Kearns, M. (2001). Attac-2000: An adaptive autonomous bidding agent. *Journal of Artificial Intelligence Research* 15:189–206.

Strother, B (2000). *TAC A Trading Agent Competition,* North Carolina State University. Retrieved on July 7, 2004 from http://portal.acm.org/citation.cfm?id=844308

Sycara, K., Pannu, A., Williamson, M., Zeng,D.J., Decker, K. (1996) *Distributed Intelligent Agents*, IEEE Intelligent System. December 1996 (Vol. 11, No. 6) p. 36-46

Taylor, P. and Jonker, L. (1978) Evolutionary stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156.

The Agent Society (n.d.) Retieved January 1, 2004 from http://www.agent.org/

The Computer Information Center. (n.d.). Retrieved January 1, 2004 from http://www.compinfo.co.uk/ai/intelligent_agents.htm

Vetsikas, I. A., & Selman, B. (2003). A principled study of the design tradeoffs for autonomous trading agents. In *Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pp. 473–480, Melbourne.

Virdhagriswaran (n.d.) *Mubot.* Retrieved October 11, 2003 from: http://www.crystaliz.com

Wellman, M. P. and Wurman, P. R. (1999). A trading agent competition for the research community. In *IJCAI-99 Workshop on Agent-Mediated Electronic Commerce*, Stockholm.

Wellman, M. P., Cheng, S.-F., Reeves, D. M., Lochner K. M. (2003) Trading Agent Competing: Performance, Progress, and market Effectiveness. *IEEE Intelligent System.* November/December (Vol. 18, No. 6)

Wellman, M. P., Greenwald, A., Stone, P., & Wurman, P. R. (2002) *The 2001 Trading Agent Competition*, In Fourteenth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-02), pages 935-941, Edmonton, 2002

Wellman, M. P., Reeves, D. M., Lochner, K. M. and Vorobeychik, Y. (2004) Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research*, 21:19-36.

Wellman, M. P., Wurman, P. R., O'Malley, K., Bangera, R., Lin, S., Reeves, D.; and Walsh, W. E. (2001). Designing the market game for a trading agent competition. *IEEE Internet Computing* 5(2): 43–51.

Wellman, M. P., MacKie-Mason, J. K., Reeves, D. M., and Swaminathan, S. (2003)

Exploring bidding strategies for market-based scheduling. In *Fourth ACM Conference on Electronic Commerce*, San Diego.

Wilson, B. (2003) The Machine learning Dictionary for COMP9414. Retrieved October 11, 2002 from: http://www.cse.unsw.edu.au

Wooldridge, M and Jennings, N R. (1994) *Agent Theories, Architectures and Languages: A Survey* in Wooldridge and Jennings Eds., Intelligent Agents, Berlin: Springer-Verlag, 1-22

Worldridge.M and Jeannins, N (1995) Intelligent Agents: Theory and practice in *Knowledge Engineering Review*, Vol 10, No 2.

Wurman, P. R., Walsh, W. E, and Wellman, M. P. (1998) Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27, 1998.

Wurman, P. R.; Wellman, M. P.; and Walsh, W. E. 1998. The Michigan Internet AuctionBot: A configurable auction server for human and software agents. In *Second International Conference on Autonomous Agents*, 301–308.

Zakon, H. R. (2004). *Hobbes' Internet Timeline*. Retrieved September 1, 2004, from http://www.zakon.org/robert/internet/timeline/

Appendices

Appendix 1: Experiments of flight tickets price

```
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>

const int times = 100000;
int main()
{
        double x, Xt[times];
        int t, i;

        for (t=0; t<720; t++)
        {
                for (i=0; i<times; i++)
                {
                        x = 10.0 + 80.0 * double(rand())/double(RAND_MAX);
                        Xt[i] = 10.0 + (double(t)/double(720))*(x-10.0);
                }

                double min = 10000.0, max= -10000.0, mean = 0.0;
                for (i=0; i<times; i++)
                {
                        mean += Xt[i];
                        if (Xt[i] > max)
                        {
                                max = Xt[i];
                        }

                        if (Xt[i] < min)
                        {
```

```
                    min = Xt[i];
                }
            }
            mean = mean/(double)(times);
            cout << max << ' ' << min << ' ' << mean << endl;
        }
        return 0;
}
```

Appendix 2: RiskerWise

```
package se.sics.tac.aw;
import se.sics.tac.util.ArgEnumerator;
import java.util.logging.*;
import java.lang.Math;
import java.lang.*;

public class RiskerWise extends AgentImpl {

 private static final Logger log =
  Logger.getLogger("se.sics.tac.aw.Risker");

 private float[] prices;
 private float[]p = new float[28];
 private int inFlight;
 private int outFlight;
 private int hotel;
 private int ET[] = {-1,-1,-1};
 private int entOwn[][] = new int[4][3]; //how many entertainment tickets owned on
particular day and type
 private int hotelOwn[][] = new int[4][2]; //how many hotel rooms owned on particular
day and type
 private int entBonus[][] = new int[8][3]; //the preference for a client on the particular
entertainment type
 private int entMaxBonus[] = new int [3];  // the Max entertainment Bonus (preference)
for a different entertainment type
 private int highest = 0; //the Max entertainment Bonus (preference) for all the
entertainment types
 private int entMean = 0;
 public  int []EO = {0,0,0};
 private int e1;
```

```
private int e2;
private int e3;
boolean allocationFlag = false;
boolean searchFlag = false;
protected void init(ArgEnumerator args) {
 prices = new float[agent.getAuctionNo()];
}
private Node ClientNode;
public int[] BArrival = new int[8];
public int[] BDeparture = new int[8];
public int[] Aday = new int[8];
public int[] Dday = new int[8];
public int[] type = new int[8];
public int hotelType = -1;
public int countClient = 0;
public Heap theHeap = new Heap(3);

public void gameStarted() {
 log.fine("Game risk " + agent.getGameID() + " started!");
 init(); //initialise varibales
 sellEnt();
 buyCheapHotels(); //try to buy cheap hotel when the game starts
  while (agent.getGameTime() < 24*1000) {
    search();
  }
 Allocation();
 sendBids();
}

private void init() {
 for (int j = 0; j <28; j++)
 {
  p[j]=0;
```

```
    }
    for (int j = 0; j <8; j++)//intialize 8 clients
    {
     for (int i = TACAgent.E1; i <= TACAgent.E3; i++) {
      entBonus[j][i-3] = 0;
      entMaxBonus[i-3] = 0;
     }
    }
    for (int d = 0; d < 4; d++) {
     for (int i = TACAgent.E1; i <= TACAgent.E3; i++) {
      entOwn[d][i-3] = 0;
     }
     for (int t = 0; t < 2; t++) { //0: good hotel; 1: cheap hotel
      hotelOwn[d][t] = 0;
     }
    }
  }

protected void search ()
{
    float smax = -10000;
    float total = 0;
    int count=0; //count all the possible value
    int [] sarrival = new int [8];
    int [] sdepartuer = new int [8];
    int [] shotel = new int [8];
    ClientNode = new Node();
    float [] s = new float [8];
    for (int j = 0; j <8; j++)//go through all the clients
    {
      s[j]= 0;
      sarrival[j] = -1;
      sdepartuer[j] = -1;
```

```
  shotel[j] = 0;
  BArrival[j]=-1;
  BDeparture[j]=-1;
  Aday[j]=-1;
  Dday[j]=-1;
  type[j]=-1;
}
for (int j = 0; j<28 ; j++) {
 int aID = agent.getAuctionID(j);
 Quote quote =agent.getQuote(j);
 p[j]=quote.getAskPrice();
 for (countClient = 0; countClient<8; countClient++){
  int i=countClient;
  int Arrival =agent.getClientPreference(i, TACAgent.ARRIVAL);
  int Departure = agent.getClientPreference(i, TACAgent.DEPARTURE);
  float Hotelbouns = agent.getClientPreference(i, TACAgent.HOTEL_VALUE);
  e1 = agent.getClientPreference(i, TACAgent.E1);
  e2 = agent.getClientPreference(i, TACAgent.E2);
  e3 = agent.getClientPreference(i, TACAgent.E3);
  float maxTemp = 0;
  for (Aday[i] = 1; Aday[i] <= 4; Aday[i] ++) {
   for (Dday[i] = Aday[i]+1; Dday[i] <=5; Dday[i] ++) {
    float temp = 1000+100*(Math.abs(Aday[i]-Arrival)+ Math.abs(Dday[i]
                - Departure))-p[Aday[i]-1]-p[Dday[i]+2]
                + hotelScore(Aday[i], Dday[i],Hotelbouns)
                + funbounds(Aday[i], Dday[i]);
     if (temp <0) continue;
      if (temp > maxTemp) //find local max for a client
      {
       maxTemp = temp;
       shotel[i] = hotelType;
       sarrival[i] = Aday[i];
       sdepartuer[i] = Dday[i];
```

```
                ClientNode.set(i, Aday[i], Dday[i], hotelType, temp);
              }
            }
          }
        }
      total = 0;
      theHeap.insertNode(ClientNode);
}

private int getHotelOwn(int day, int type) {
  int auction = agent.getAuctionFor(1, type, day);
  int auctionID = agent.getAuctionID(auction);
  int hotel = agent.getOwn(auctionID);
  return hotel;
}

private void getEntOwn(int day, int type) {
 int auction = agent.getAuctionFor(2, type, day);
 entOwn[day-1][type-3] = agent.getOwn(auction);
}

private void buyCheapHotels() {
 for (int day = 1; day <= 4; day++) {
   for (int type = 0; type <=1; type++) {
     int auction = agent.getAuctionFor(1, type, day);
         Bid bid = new Bid(auction);
      bid.addBidPoint(8,1);
            agent.submitBid(bid);
   }
 }
}

 private void sellEnt() {
```

```
  for (int client = 0; client < 8; client ++) {
    for (int i = 1; i <= 3; i++) {
      entBonus[client][i-1] = agent.getClientPreference(client, i);
      entMean += entBonus[client][i-1];
    }
  }
  entMean = entMean/24;

  for (int i = 1; i <= 3; i++) {
    for (int client = 0; client < 8; client ++) {
      if (entMaxBonus[i-1] < entBonus[client][i-1]) {
        entMaxBonus[i-1] = entBonus[client][i-1];
        if (highest < entMaxBonus[i-1]) {
          highest = entMaxBonus[i-1];
        }
      }
    }
  }
  for (int day = 1; day <= 4; day++) {
    for (int type = 1; type <=3; type++) {
      int auction = agent.getAuctionFor(2, type, day);
      entOwn[day-1][type-1] = agent.getOwn(auction);
      if (entOwn[day-1][type-1] > 0) {
        Bid bid = new Bid(auction);
        if (entMaxBonus[type-1] < 90) entMaxBonus[type-1]=90;
              bid.addBidPoint(-entOwn[day-1][type-1],entMaxBonus[type-1]+20);
              agent.submitBid(bid);
      }
    }
  }
}
```

```
protected float funbounds (int Aday, int Dday)
{
 int countDay = 0;
 float total_fun_bonus = 0;
 int j;
 for ( j = 0; j < (Dday-Aday) && j < 3; j++)
 {
  total_fun_bonus += ET[j];
  countDay++;
 }
 j=0;
 while (j<3) {
  ET[j]=-1;
  j++;
 }
 float EntScore = total_fun_bonus - 80*countDay;
 return EntScore;
}

protected float hotelScore (int Aday, int Dday, float Hotelbouns)
{
  int totaltower = 0;
  int totalshanty = 0;
  int aucid;
  for (int i = Aday; i < Dday; i++)
  {
   totaltower += p[i-1+12];
   totalshanty += p[i-1+8];
  }
  int g = (int)Hotelbouns - totaltower;
  int b = 0 - totalshanty;
  if (g > b){
   hotelType = TACAgent.TYPE_GOOD_HOTEL;
```

```
      return g;
    } else {
      hotelType = TACAgent.TYPE_CHEAP_HOTEL;
      return b;
    }
  }

  private float MaxHotelBonus () {
    float Max = 0;
    for (int Client = 0; Client<8; Client++){
        float Hotelbouns = agent.getClientPreference(Client,
                                    TACAgent.HOTEL_VALUE);
        if (Max < Hotelbouns) Max = Hotelbouns;
      }
      return Max;
  }

  private float MinHotelBonus () {
    float Min = 200;
    for (int Client = 0; Client<8; Client++){
        float Hotelbouns = agent.getClientPreference(Client,
                                    TACAgent.HOTEL_VALUE);
        if (Min < Hotelbouns) Min = Hotelbouns;
      }
      return Min;
  }

  public void quoteUpdated(Quote quote) {
    int auction = quote.getAuction();
    int auctionCategory = agent.getAuctionCategory(auction);
    if (auctionCategory == TACAgent.CAT_HOTEL) {
      int amount;
      float p = quote.getAskPrice(); //ask price for this auction
```

```
int alloc = agent.getAllocation(auction);
int day = agent.getHotelAuctionDay(auction);
int Type = agent.getHotelAuctionType(auction);
if (Type == 0 ) {
 Type =1;
} else {Type = 0; }
int hotel = getHotelOwn(day, Type);
int aauction = agent.getAuctionFor(1, Type, day);
int aalloc = agent.getAllocation(auction);
if (alloc > aalloc) {
 amount = alloc - hotel;
} else amount = aalloc - hotel;
if (alloc > 0 && quote.hasHQW(agent.getBid(auction)) &&
        quote.getHQW() < alloc && hotel < alloc  &&
            quote.hasHQW(agent.getBid(aauction)) &&
                                quote.getHQW() < alloc) {
 int aauctionID = agent.getAuctionID(aauction);
 Quote aQuote =  agent.getQuote(aauctionID);
 float aPrice = aQuote.getAskPrice();
 if ((p - aPrice) > MinHotelBonus()) {
  if (Type == 0) {
   for (double t = 3.9; t < 11.9; t++) {
    if ((agent.getGameTime() - t*60*1000) < 5*1000) {
     aPrice += 50;
    }
   }
   Bid bid = new Bid(aauction);
           bid.addBidPoint(amount, aPrice+20);
           agent.submitBid(bid);
 } else {
         prices[auction] = p + 20;
  for (double t = 3.8; t < 11.8; t++) {
   if ((agent.getGameTime() - t*60*1000) < 5*1000) {
```

```
            prices[auction] = p+150;//+= 150;
          }
        }
              if (prices[auction] > 1170) prices[auction] = 1170;
              Bid bid = new Bid(auction);
              bid.addBidPoint(amount, prices[auction]);
              agent.submitBid(bid);
        }
      }
    } else if (auctionCategory == TACAgent.CAT_ENTERTAINMENT) {
    int alloc = agent.getAllocation(auction) - agent.getOwn(auction);
    if (alloc != 0) {
            Bid bid = new Bid(auction);
            if (alloc < 0) {
              prices[auction] = 200f - (agent.getGameTime() * 120f) / 720000;
      }
            else {
              prices[auction] = 30f + (agent.getGameTime() * 100f) / 720000;
      }
      if (prices[auction]> entMean) {
        prices[auction] = entMean;
      }
      bid.addBidPoint(alloc, prices[auction]);
            agent.submitBid(bid);
    }
  }
}

public void quoteUpdated(int auctionCategory) {
  log.fine("All quotes for "
          + agent.auctionCategoryToString(auctionCategory)
          + " has been updated");
}
```

```
public void bidUpdated(Bid bid) {
  log.fine("Bid Updated: id=" + bid.getID() + " auction="
           + bid.getAuction() + " state="
           + bid.getProcessingStateAsString());
  log.fine("      Hash: " + bid.getBidHash());
}

public void bidRejected(Bid bid) {
  log.warning("Bid Rejected: " + bid.getID());
  log.warning("      Reason: " + bid.getRejectReason()
              + " (" + bid.getRejectReasonAsString() + ')');
}

public void bidError(Bid bid, int status) {
  log.warning("Bid Error in auction " + bid.getAuction() + ": " + status
              + " (" + agent.commandStatusToString(status) + ')');
}

public void gameStopped() {
  log.fine("Game Stopped! ");
}

public void auctionClosed(int auction) {
  log.fine("*** Auction " + auction + " closed! ");
}

private void sendBids() {
  for (int i = 0, n = agent.getAuctionNo(); i < n; i++) {
    int alloc = agent.getAllocation(i) - agent.getOwn(i);
    log.fine("in sendBids()   "+"auction: "+i+" alloc:  "+ alloc);
    float price = -1;
    switch (agent.getAuctionCategory(i)) {
```

```
      case TACAgent.CAT_FLIGHT:
            if (alloc > 0) {
              price = 1000;
            }
           break;
      case TACAgent.CAT_ENTERTAINMENT:
            if (alloc < 0) {
              price = 150;
            } else if (alloc > 0) {
              price = 50;
              prices[i] = 50f;
          }
           break;
      default:
           break;
    }
    if (price > 0) {
        Bid bid = new Bid(i);
          bid.addBidPoint(alloc, price);
          agent.submitBid(bid);
    }
  }
}

  private void Allocation() {
    for (int a = 0; a < 8; a++) {
      int auction=-1;
      auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                    TACAgent.TYPE_INFLIGHT,
                                  theHeap.heapArray[0].getArrival(a));
      agent.setAllocation (auction, agent.getAllocation(auction) + 1);
      auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                    TACAgent.TYPE_OUTFLIGHT,
```

```
                                theHeap.heapArray[0].getDepartuer(a));
    agent.setAllocation(auction, agent.getAllocation(auction) + 1);
    for (int d = theHeap.heapArray[0].getArrival(a); d <
                            theHeap.heapArray[0].getDepartuer(a); d++) {
      auction = agent.getAuctionFor(TACAgent.CAT_HOTEL,
                                theHeap.heapArray[0].getHotel(a), d);
      agent.setAllocation(auction, agent.getAllocation(auction) + 1);
    }
    int eType = -1;
    while((eType = nextEntType(a, eType)) > 0) {
      auction = bestEntDay(theHeap.heapArray[0].getArrival(a),
                                theHeap.heapArray[0].getDepartuer(a), eType);
            agent.setAllocation(auction, agent.getAllocation(auction) + 1);
    }
  }
}

private int bestEntDay(int inFlight, int outFlight, int type) {
  for (int i = inFlight; i < outFlight; i++) {
    int auction = agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                                        type, i);
    if (agent.getAllocation(auction) < agent.getOwn(auction)) {
            return auction;
    }
  }
  return agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                                type, inFlight);
}

private int nextEntType(int client, int lastType) {
  int e1 = agent.getClientPreference(client, TACAgent.E1);
  int e2 = agent.getClientPreference(client, TACAgent.E2);
  int e3 = agent.getClientPreference(client, TACAgent.E3);
```

```
    if ((e1 > e2) && (e1 > e3) && lastType == -1)
      return TACAgent.TYPE_ALLIGATOR_WRESTLING;
    if ((e2 > e1) && (e2 > e3) && lastType == -1)
      return TACAgent.TYPE_AMUSEMENT;
    if ((e3 > e1) && (e3 > e2) && lastType == -1)
      return TACAgent.TYPE_MUSEUM;
    return -1;
  }

  public static void main (String[] args) {
    TACAgent.main(args);
  }
}
```

Appendix 3: Heap

```
package se.sics.tac.aw;
import java.io.*;
import java.lang.Integer;

public class Heap
{
  public Node[] heapArray;
  private int maxSize;
  private int currentSize;

public Heap(int mx)
{
    maxSize = mx;
    currentSize = 0;
    heapArray = new Node[maxSize];
 }

public boolean isEmpty()
{ return currentSize==0; }

public boolean insertNode(Node A)
{
   if(currentSize==maxSize)
   return false;
   Node newNode = A;
   heapArray[currentSize] = newNode;
   trickleUp(currentSize++);
   return true;
  }
```

```
public boolean insert(int []arrival, int []departuer, int []hotel, float []score)
{
    if (currentSize==maxSize)
      return false;
      Node newNode = new Node(arrival,departuer,hotel, score);
      heapArray[currentSize] = newNode;
      trickleUp(currentSize++);
      return true;
}

public void trickleUp(int index)
{
    int parent = (index-1) / 2;
    Node bottom = heapArray[index];
    while( index > 0 &&
         heapArray[parent].total < bottom.total )
   {
      heapArray[index] = heapArray[parent];
      index = parent;
      parent = (parent-1) / 2;
    }
    heapArray[index] = bottom;
    }

public Node remove ()
{
    Node root = heapArray[0];
    heapArray[0] = heapArray[--currentSize];
    trickleDown(0);
    return root;
 }
```

```
public void trickleDown(int index)
{
    int largerChild;
    Node top = heapArray[index];
    while(index < currentSize/2)
    {
      int leftChild = 2*index+1;
      int rightChild = leftChild+1;
      if(rightChild < currentSize &&
                    heapArray[leftChild].total <
                    heapArray[rightChild].total)
        largerChild = rightChild;
      else
        largerChild = leftChild;
        if(top.total >= heapArray[largerChild].total)
        break;
        heapArray[index] = heapArray[largerChild];
      index = largerChild;
    }
    heapArray[index] = top;
  }
}
```

Appendix 4: Node

```
package se.sics.tac.aw;
import java.io.*;
import java.lang.Integer;

public class Node
{
 public float total = 0;
 public int [] arrival = new int [8];
 public int [] departuer = new int [8];
 public int [] hotel = new int [8];
 public float [] score = new float [8];
 public Node(int [] arrival, int [] departuer, int [] hotel, float []score)
 {
  for (int a = 0; a < 8; a++) {
   arrival[a]= arrival[a];
   departuer[a] = departuer[a];
   hotel[a] = hotel[a];
   score[a] = score[a];
   total += score[a];
  }
 }

 public Node()
 {
  for (int a = 0; a < 8; a++) {
   arrival[a]= -1;
   departuer[a] = -1;
   hotel[a] = -1;
   score[a] = 0;
   total = 0;
```

```
  }
}

public void set(int i, int arrival, int departuer, int hotel, float score)
{
  this.arrival[i]= arrival;
  this.departuer[i] = departuer;
  this.hotel[i] = hotel;
  this.score[i] = score;
}

public int getArrival(int i)
{
  return this.arrival[i];
}

public int getDepartuer(int i)
{
  return this.departuer[i];
}

public int getHotel(int i)
{
  return this.hotel[i];
}

public float getScore(int i)
{
  return this.score[i];
}

}
```

Appendix 5: Scores table of the 58 games RiskerWise played

Game	Utility	Score		Game	Utility	Score
25863	8569	2822.69		25864	8307	3485
25867	8194	2725		25868	8373	1927
25871	8652	3324		25872	8395	2753
25875	8300	3710		25876	7801	1852
25879	8044	2290		25880	8084	2679
25883	8819	3004		25884	7620	2893
25887	7821	2995		25888	8353	2531
25891	7655	2410		25892	8489	2858
25895	8335	2482		25896	7949	2683
25899	8765	3966		25900	6400	257
25903	8565	3305		25904	8452	3631
25907	8643	3285		25908	8018	1903
25911	8757	3649		25912	8359	2545
25915	8217	2525		25916	7780	1902
25919	7808	2859		25920	8846	2865
25865	8506	3505		25866	7806	1677
25869	8537	2874		25870	8385	3679
25873	8020	2642		25874	8382	3256
25877	8058	2546		25878	8467	3022
25881	8846	3552		25882	7386	708
25885	8760	2536		25886	8135	2451
25889	7718	1430		25890	7284	2442
25893	8054	2736		25894	7583	1224
25897	8078	2517		25898	8161	2490
25901	7959	2676		25902	8063	2887
25905	8146	2458		25906	8239	2407
25909	8254	3499		25910	7852	1794
25913	8924	3387		25914	9147	3959
25917	8772	3439		25918	7900	2188

Appendix 6: Game results for RiskerWise in game 25899

Client	Arrival	Departure	Hotel	Entertainment	Utility	Cost	Score
1	Day 1	Day 2	TampaTowers Hotel	Day 1 AlligatorWrestling	1083	669.00	414.00
2	Day 4	Day 5	TampaTowers Hotel	Day 4 AmusementPark	1270	596.00	674.00
3	Day 1	Day 2	TampaTowers Hotel	Day 1 AlligatorWrestling	1018	669.00	349.00
4	Day 4	Day 5	TampaTowers Hotel	Day 4 AlligatorWrestling	1264	776.00	488.00
5	Day 1	Day 2	TampaTowers Hotel	Day 1 AmusementPark	769	669.00	100.00
6	Day 1	Day 2	TampaTowers Hotel	Day 1 AmusementPark	1094	669.00	425.00
7	Day 1	Day 2	TampaTowers Hotel	Day 1 AlligatorWrestling	1195	669.00	526.00
8	Day 1	Day 2	TampaTowers Hotel	Day 1 AlligatorWrestling	1072	669.00	403.00
				Sum	8765	5386.00	3379.00
				Other costs (unused goods, transaction losses, etc)		-587.00	
				Total	8765	4799.00	3966.00

Appendix 7: Client Preferences for RiskerWise in game 25899

Client	Arrival	Departure	Hotel	AlligatorWrestling	AmusementPark	Museum
1	Day 2	Day 3	143	140	24	88
2	Day 4	Day 5	78	143	192	23
3	Day 1	Day 4	145	73	39	48
4	Day 4	Day 5	91	173	160	22
5	Day 3	Day 5	113	141	156	24
6	Day 1	Day 4	118	109	176	14
7	Day 1	Day 3	100	195	12	123
8	Day 2	Day 3	82	190	81	87

Appendix 8: Endowments for RiskerWise in game 25899

Entertainment	Day 1	Day 2	Day 3	Day 4
AlligatorWrestling	0	0	4	0
AmusementPark	0	0	0	2
Museum	0	2	0	4

Appendix 9: Owned Goods for RiskerWise in game 25899

Goods	Day 1	Day 2	Day 3	Day 4	Day 5	Cost
InFlight	6	0	0	2	0	2298.00
OutFlight	0	6	0	0	2	2300.00
TampaTowersHotel	7 (1)	0	0	6 (4)	0	13.00
ShorelineShantyHotel	4 (4)	0	0	4 (4)	0	8.00
AlligatorWrestling	4	0	0	1	0	360.00
AmusementPark	2	0	0	1	0	180.00
Museum	0	0	0	0	0	-360.00
					Total	4799.00

(Unused goods is specified in parentheses)

Appendix 10: Transactions for game 25899

Time	Auction	Day	Buyer	Seller	Quantity	Price
19:42:00	InFlight	4	dummy-7	auction	1	366.00
19:42:00	OutFlight	3	dummy-7	auction	1	269.00
19:42:00	OutFlight	3	dummy-7	auction	1	269.00
19:42:00	InFlight	1	dummy-7	auction	1	261.00
19:42:00	InFlight	1	dummy-7	auction	1	261.00
19:42:00	OutFlight	4	dummy-7	auction	1	258.00
19:42:00	OutFlight	4	dummy-7	auction	1	258.00
19:42:00	OutFlight	4	dummy-7	auction	1	258.00
19:42:00	OutFlight	4	dummy-7	auction	1	258.00
19:42:00	InFlight	2	dummy-7	auction	1	382.00
19:42:00	InFlight	2	dummy-7	auction	1	382.00
19:42:00	InFlight	2	dummy-7	auction	1	382.00
19:42:00	InFlight	2	dummy-7	auction	1	382.00
19:42:00	OutFlight	5	dummy-7	auction	1	293.00
19:42:00	OutFlight	5	dummy-7	auction	1	293.00
19:42:00	InFlight	3	dummy-7	auction	1	279.00
19:42:00	InFlight	3	dummy-6	auction	1	279.00
19:42:00	OutFlight	4	dummy-6	auction	1	258.00
19:42:00	OutFlight	4	dummy-6	auction	1	258.00
19:42:00	OutFlight	5	dummy-6	auction	1	293.00
19:42:00	OutFlight	5	dummy-6	auction	1	293.00
19:42:00	InFlight	1	dummy-6	auction	1	261.00
19:42:00	InFlight	1	dummy-6	auction	1	261.00
19:42:00	InFlight	1	dummy-6	auction	1	261.00
19:42:00	InFlight	1	dummy-6	auction	1	261.00
19:42:00	OutFlight	3	dummy-6	auction	1	269.00
19:42:00	OutFlight	3	dummy-6	auction	1	269.00

19:42:00	OutFlight	3	dummy-6	auction	1	269.00
19:42:00	OutFlight	3	dummy-6	auction	1	269.00
19:42:00	InFlight	2	dummy-6	auction	1	382.00
19:42:00	InFlight	2	dummy-6	auction	1	382.00
19:42:00	InFlight	2	dummy-6	auction	1	382.00
19:42:00	InFlight	3	dummy-5	auction	1	279.00
19:42:00	OutFlight	3	dummy-5	auction	1	269.00
19:42:00	OutFlight	4	dummy-5	auction	1	258.00
19:42:00	OutFlight	4	dummy-5	auction	1	258.00
19:42:00	OutFlight	2	dummy-5	auction	1	294.00
19:42:00	InFlight	2	dummy-5	auction	1	382.00
19:42:00	InFlight	2	dummy-5	auction	1	382.00
19:42:00	InFlight	2	dummy-5	auction	1	382.00
19:42:00	InFlight	2	dummy-5	auction	1	382.00
19:42:00	OutFlight	5	dummy-5	auction	1	293.00
19:42:00	OutFlight	5	dummy-5	auction	1	293.00
19:42:00	OutFlight	5	dummy-5	auction	1	293.00
19:42:00	OutFlight	5	dummy-5	auction	1	293.00
19:42:00	InFlight	1	dummy-5	auction	1	261.00
19:42:00	InFlight	1	dummy-5	auction	1	261.00
19:42:00	InFlight	1	dummy-5	auction	1	261.00
19:42:00	OutFlight	3	dummy-4	auction	1	269.00
19:42:00	OutFlight	3	dummy-4	auction	1	269.00
19:42:00	OutFlight	3	dummy-4	auction	1	269.00
19:42:00	OutFlight	4	dummy-4	auction	1	258.00
19:42:00	OutFlight	4	dummy-4	auction	1	258.00
19:42:00	InFlight	2	dummy-4	auction	1	382.00
19:42:00	InFlight	2	dummy-4	auction	1	382.00
19:42:00	InFlight	2	dummy-4	auction	1	382.00

19:42:00	OutFlight	5	dummy-4	auction	1	293.00
19:42:00	OutFlight	5	dummy-4	auction	1	293.00
19:42:00	OutFlight	5	dummy-4	auction	1	293.00
19:42:00	InFlight	1	dummy-4	auction	1	261.00
19:42:00	InFlight	1	dummy-4	auction	1	261.00
19:42:00	InFlight	1	dummy-4	auction	1	261.00
19:42:00	InFlight	1	dummy-4	auction	1	261.00
19:42:00	InFlight	1	dummy-4	auction	1	261.00
19:42:00	OutFlight	3	dummy-3	auction	1	269.00
19:42:00	OutFlight	4	dummy-3	auction	1	258.00
19:42:00	OutFlight	4	dummy-3	auction	1	258.00
19:42:00	InFlight	2	dummy-3	auction	1	382.00
19:42:00	InFlight	2	dummy-3	auction	1	382.00
19:42:00	InFlight	2	dummy-3	auction	1	382.00
19:42:00	InFlight	3	dummy-3	auction	1	279.00
19:42:00	OutFlight	5	dummy-3	auction	1	293.00
19:42:00	OutFlight	5	dummy-3	auction	1	293.00
19:42:00	OutFlight	5	dummy-3	auction	1	293.00
19:42:00	OutFlight	5	dummy-3	auction	1	293.00
19:42:00	OutFlight	5	dummy-3	auction	1	293.00
19:42:00	InFlight	1	dummy-3	auction	1	261.00
19:42:00	InFlight	1	dummy-3	auction	1	261.00
19:42:00	InFlight	1	dummy-3	auction	1	261.00
19:42:00	InFlight	1	dummy-3	auction	1	261.00
19:42:00	OutFlight	3	dummy-2	auction	1	269.00
19:42:00	InFlight	1	dummy-2	auction	1	261.00
19:42:00	InFlight	1	dummy-2	auction	1	261.00
19:42:00	InFlight	4	dummy-2	auction	1	366.00
19:42:00	OutFlight	4	dummy-2	auction	1	258.00

19:42:00	OutFlight	4	dummy-2	auction	1	258.00
19:42:00	OutFlight	4	dummy-2	auction	1	258.00
19:42:00	InFlight	2	dummy-2	auction	1	382.00
19:42:00	InFlight	2	dummy-2	auction	1	382.00
19:42:00	InFlight	2	dummy-2	auction	1	382.00
19:42:00	InFlight	2	dummy-2	auction	1	382.00
19:42:00	OutFlight	5	dummy-2	auction	1	293.00
19:42:00	OutFlight	5	dummy-2	auction	1	293.00
19:42:00	OutFlight	5	dummy-2	auction	1	293.00
19:42:00	OutFlight	5	dummy-2	auction	1	293.00
19:42:00	InFlight	3	dummy-2	auction	1	279.00
19:42:00	OutFlight	3	dummy-1	auction	1	269.00
19:42:00	OutFlight	3	dummy-1	auction	1	269.00
19:42:00	InFlight	3	dummy-1	auction	1	279.00
19:42:00	InFlight	3	dummy-1	auction	1	279.00
19:42:00	InFlight	1	dummy-1	auction	1	261.00
19:42:00	OutFlight	4	dummy-1	auction	1	258.00
19:42:00	OutFlight	4	dummy-1	auction	1	258.00
19:42:00	OutFlight	4	dummy-1	auction	1	258.00
19:42:00	OutFlight	4	dummy-1	auction	1	258.00
19:42:00	OutFlight	5	dummy-1	auction	1	293.00
19:42:00	OutFlight	5	dummy-1	auction	1	293.00
19:42:00	InFlight	2	dummy-1	auction	1	382.00
19:42:00	InFlight	2	dummy-1	auction	1	382.00
19:42:00	InFlight	2	dummy-1	auction	1	382.00
19:42:00	InFlight	2	dummy-1	auction	1	382.00
19:42:00	InFlight	2	dummy-1	auction	1	382.00
19:43:05	InFlight	1	RiskerWise	auction	6	260.00
19:43:05	InFlight	4	RiskerWise	auction	2	369.00

19:43:05	OutFlight	2	RiskerWise	auction	6	288.00
19:43:05	OutFlight	5	RiskerWise	auction	2	286.00
19:46:00	ShorelineShantyHotel	4	dummy-7	auction	1	1.00
19:46:00	ShorelineShantyHotel	4	dummy-6	auction	1	1.00
19:46:00	ShorelineShantyHotel	4	dummy-5	auction	3	1.00
19:46:00	ShorelineShantyHotel	4	dummy-4	auction	2	1.00
19:46:00	ShorelineShantyHotel	4	dummy-3	auction	4	1.00
19:46:00	ShorelineShantyHotel	4	dummy-1	auction	1	1.00
19:46:00	ShorelineShantyHotel	4	RiskerWise	auction	4	1.00
19:47:00	TampaTowersHotel	1	dummy-6	auction	2	1.00
19:47:00	TampaTowersHotel	1	dummy-5	auction	1	1.00
19:47:00	TampaTowersHotel	1	dummy-4	auction	2	1.00
19:47:00	TampaTowersHotel	1	dummy-3	auction	1	1.00
19:47:00	TampaTowersHotel	1	dummy-2	auction	2	1.00
19:47:00	TampaTowersHotel	1	dummy-1	auction	1	1.00
19:47:00	TampaTowersHotel	1	RiskerWise	auction	7	1.00
19:48:00	ShorelineShantyHotel	2	dummy-7	auction	4	1000.00
19:48:00	ShorelineShantyHotel	2	dummy-6	auction	4	1000.00
19:48:00	ShorelineShantyHotel	2	dummy-5	auction	3	1000.00
19:48:00	ShorelineShantyHotel	2	dummy-4	auction	4	1000.00
19:48:00	ShorelineShantyHotel	2	dummy-3	auction	1	1000.00
19:49:00	TampaTowersHotel	2	dummy-7	auction	2	1000.00
19:49:00	TampaTowersHotel	2	dummy-6	auction	3	1000.00
19:49:00	TampaTowersHotel	2	dummy-5	auction	3	1000.00
19:49:00	TampaTowersHotel	2	dummy-4	auction	4	1000.00
19:49:00	TampaTowersHotel	2	dummy-3	auction	2	1000.00
19:49:00	TampaTowersHotel	2	dummy-2	auction	2	1000.00
19:50:00	TampaTowersHotel	3	dummy-7	auction	2	1000.00
19:50:00	TampaTowersHotel	3	dummy-6	auction	3	1000.00

19:50:00	TampaTowersHotel	3	dummy-5	auction	3	1000.00
19:50:00	TampaTowersHotel	3	dummy-4	auction	2	1000.00
19:50:00	TampaTowersHotel	3	dummy-3	auction	1	1000.00
19:50:00	TampaTowersHotel	3	dummy-2	auction	5	1000.00
19:50:36	AlligatorWrestling	1	RiskerWise	dummy-2	2	120.00
19:50:36	AlligatorWrestling	1	RiskerWise	dummy-4	2	120.00
19:50:36	AlligatorWrestling	4	RiskerWise	dummy-7	1	120.00
19:50:36	AmusementPark	1	RiskerWise	dummy-7	2	120.00
19:51:00	ShorelineShantyHotel	3	dummy-7	auction	3	1000.00
19:51:00	ShorelineShantyHotel	3	dummy-6	auction	1	1000.00
19:51:00	ShorelineShantyHotel	3	dummy-5	auction	3	1000.00
19:51:00	ShorelineShantyHotel	3	dummy-4	auction	3	1000.00
19:51:00	ShorelineShantyHotel	3	dummy-3	auction	6	1000.00
19:51:06	AlligatorWrestling	3	dummy-4	RiskerWise	1	60.00
19:51:06	AlligatorWrestling	3	dummy-5	RiskerWise	1	60.00
19:51:06	AlligatorWrestling	3	dummy-6	RiskerWise	1	60.00
19:51:06	AlligatorWrestling	3	dummy-7	RiskerWise	1	60.00
19:51:07	AmusementPark	4	dummy-7	RiskerWise	1	60.00
19:51:07	Museum	2	dummy-6	RiskerWise	1	60.00
19:51:07	Museum	2	dummy-7	RiskerWise	1	60.00
19:51:07	Museum	4	dummy-3	RiskerWise	1	60.00
19:51:07	Museum	4	dummy-4	RiskerWise	1	60.00
19:51:07	Museum	4	dummy-5	RiskerWise	1	60.00
19:51:07	Museum	4	dummy-7	RiskerWise	1	60.00
19:52:00	TampaTowersHotel	4	dummy-7	auction	1	1.00
19:52:00	TampaTowersHotel	4	dummy-6	auction	1	1.00
19:52:00	TampaTowersHotel	4	dummy-5	auction	1	1.00
19:52:00	TampaTowersHotel	4	dummy-4	auction	1	1.00
19:52:00	TampaTowersHotel	4	dummy-3	auction	1	1.00

19:52:00	TampaTowersHotel	4	dummy-2	auction	4	1.00
19:52:00	TampaTowersHotel	4	dummy-1	auction	1	1.00
19:52:00	TampaTowersHotel	4	RiskerWise	auction	6	1.00
19:53:00	ShorelineShantyHotel	1	dummy-7	auction	2	1.00
19:53:00	ShorelineShantyHotel	1	dummy-6	auction	2	1.00
19:53:00	ShorelineShantyHotel	1	dummy-5	auction	2	1.00
19:53:00	ShorelineShantyHotel	1	dummy-4	auction	3	1.00
19:53:00	ShorelineShantyHotel	1	dummy-3	auction	3	1.00
19:53:00	ShorelineShantyHotel	1	RiskerWise	auction	4	1.00

Appendix 11: Game results for RiskerWise at game 25900

Client	Arrival	Departure	Hotel	Entertainment	Utility	Cost	Score
1	Day 1	Day 2	TampaTowers Hotel	Day 1 AlligatorWrestling	996	990.00	6.00
3	Day 4	Day 5	TampaTowers Hotel	Day 4 AlligatorWrestling	1143	726.00	417.00
4	Day 1	Day 2	TampaTowers Hotel	Day 1 AmusementPark	1266	870.00	396.00
5	Day 1	Day 2	TampaTowers Hotel	Day 1 AmusementPark	995	870.00	125.00
6	Day 4	Day 5	TampaTowers Hotel	Day 4 AlligatorWrestling	1122	726.00	396.00
7	Day 1	Day 2	TampaTowers Hotel	Day 1 AmusementPark	878	870.00	8.00
				Sum	6400	5052.00	1348.00
				Other costs (unused goods, transaction losses, etc)		1091.00	
				Total	6400	6143.00	257.00

Appendix 12: Client Preferences for RiskerWise at game 25900

Client	Arrival	Departure	Hotel	AlligatorWrestling	AmusementPark	Museum
1	Day 2	Day 4	105	191	6	5
2	Day 3	Day 4	104	84	127	66
3	Day 3	Day 5	81	162	13	31
4	Day 1	Day 2	128	132	138	70
5	Day 1	Day 4	102	177	93	127
6	Day 3	Day 5	139	83	145	71
7	Day 2	Day 4	70	160	108	160
8	Day 2	Day 5	67	30	105	47

Appendix 13: Endowments for RiskerWise at game 25900

Entertainment	Day 1	Day 2	Day 3	Day 4
AlligatorWrestling	0	2	0	2
AmusementPark	4	0	4	0
Museum	0	0	0	0

Appendix 14: Owned Goods for RiskerWise at game 25900

Goods	Day 1	Day 2	Day 3	Day 4	Day 5	Cost
InFlight	6 (2)	0	0	2	0	3000.00
OutFlight	0	6 (2)	0	0	2	2770.00
TampaTowersHotel	4	0	3 (3)	4 (2)	0	607.00
ShorelineShantyHotel	0	0	0	6 (6)	0	6.00
AlligatorWrestling	1	0	0	2	0	0.00
AmusementPark	4 (1)	0	0	0	0	-240.00
Museum	0	0	0	0	0	0.00
					Total	6143.00

(Unused goods is specified in parentheses)

Appendix 15: Transactions at game 25900

Time	Auction	Day	Buyer	Seller	Quantity	Price
20:03:00	OutFlight	4	dummy-7	auction	1	372.00
20:03:00	InFlight	1	dummy-7	auction	1	398.00
20:03:00	InFlight	1	dummy-7	auction	1	398.00
20:03:00	InFlight	3	dummy-7	auction	1	337.00
20:03:00	OutFlight	5	dummy-7	auction	1	395.00
20:03:00	OutFlight	5	dummy-7	auction	1	395.00
20:03:00	OutFlight	5	dummy-7	auction	1	395.00
20:03:00	OutFlight	3	dummy-7	auction	1	335.00
20:03:00	OutFlight	3	dummy-7	auction	1	335.00
20:03:00	OutFlight	3	dummy-7	auction	1	335.00
20:03:00	OutFlight	3	dummy-7	auction	1	335.00
20:03:00	InFlight	2	dummy-7	auction	1	373.00
20:03:00	InFlight	2	dummy-7	auction	1	373.00
20:03:00	InFlight	2	dummy-7	auction	1	373.00
20:03:00	InFlight	2	dummy-7	auction	1	373.00
20:03:00	InFlight	2	dummy-7	auction	1	373.00
20:03:00	InFlight	2	dummy-6	auction	1	373.00
20:03:00	InFlight	3	dummy-6	auction	1	337.00
20:03:00	OutFlight	3	dummy-6	auction	1	335.00
20:03:00	OutFlight	3	dummy-6	auction	1	335.00
20:03:00	OutFlight	4	dummy-6	auction	1	372.00
20:03:00	OutFlight	4	dummy-6	auction	1	372.00
20:03:00	OutFlight	4	dummy-6	auction	1	372.00
20:03:00	InFlight	1	dummy-6	auction	1	398.00
20:03:00	InFlight	1	dummy-6	auction	1	398.00
20:03:00	InFlight	1	dummy-6	auction	1	398.00

20:03:00	InFlight	1	dummy-6	auction	1	398.00
20:03:00	OutFlight	5	dummy-6	auction	1	395.00
20:03:00	OutFlight	5	dummy-6	auction	1	395.00
20:03:00	OutFlight	5	dummy-6	auction	1	395.00
20:03:00	InFlight	4	dummy-6	auction	1	317.00
20:03:00	InFlight	4	dummy-6	auction	1	317.00
20:03:00	OutFlight	5	dummy-5	auction	1	395.00
20:03:00	InFlight	3	dummy-5	auction	1	337.00
20:03:00	OutFlight	2	dummy-5	auction	1	332.00
20:03:00	OutFlight	2	dummy-5	auction	1	332.00
20:03:00	InFlight	1	dummy-5	auction	1	398.00
20:03:00	InFlight	1	dummy-5	auction	1	398.00
20:03:00	InFlight	1	dummy-5	auction	1	398.00
20:03:00	InFlight	1	dummy-5	auction	1	398.00
20:03:00	InFlight	1	dummy-5	auction	1	398.00
20:03:00	OutFlight	4	dummy-5	auction	1	372.00
20:03:00	OutFlight	4	dummy-5	auction	1	372.00
20:03:00	OutFlight	3	dummy-5	auction	1	335.00
20:03:00	OutFlight	3	dummy-5	auction	1	335.00
20:03:00	OutFlight	3	dummy-5	auction	1	335.00
20:03:00	InFlight	2	dummy-5	auction	1	373.00
20:03:00	InFlight	2	dummy-5	auction	1	373.00
20:03:00	InFlight	3	dummy-4	auction	1	337.00
20:03:00	InFlight	2	dummy-4	auction	1	373.00
20:03:00	InFlight	2	dummy-4	auction	1	373.00
20:03:00	OutFlight	4	dummy-4	auction	1	372.00
20:03:00	OutFlight	5	dummy-4	auction	1	395.00
20:03:00	OutFlight	5	dummy-4	auction	1	395.00
20:03:00	OutFlight	5	dummy-4	auction	1	395.00

20:03:00	OutFlight	5	dummy-4	auction	1	395.00
20:03:00	OutFlight	3	dummy-4	auction	1	335.00
20:03:00	OutFlight	2	dummy-4	auction	1	332.00
20:03:00	OutFlight	2	dummy-4	auction	1	332.00
20:03:00	InFlight	1	dummy-4	auction	1	398.00
20:03:00	InFlight	1	dummy-4	auction	1	398.00
20:03:00	InFlight	1	dummy-4	auction	1	398.00
20:03:00	InFlight	1	dummy-4	auction	1	398.00
20:03:00	InFlight	1	dummy-4	auction	1	398.00
20:03:00	InFlight	4	dummy-3	auction	1	317.00
20:03:00	OutFlight	5	dummy-3	auction	1	395.00
20:03:00	OutFlight	5	dummy-3	auction	1	395.00
20:03:00	OutFlight	2	dummy-3	auction	1	332.00
20:03:00	OutFlight	3	dummy-3	auction	1	335.00
20:03:00	OutFlight	3	dummy-3	auction	1	335.00
20:03:00	InFlight	1	dummy-3	auction	1	398.00
20:03:00	InFlight	1	dummy-3	auction	1	398.00
20:03:00	InFlight	1	dummy-3	auction	1	398.00
20:03:00	InFlight	1	dummy-3	auction	1	398.00
20:03:00	InFlight	1	dummy-3	auction	1	398.00
20:03:00	OutFlight	4	dummy-3	auction	1	372.00
20:03:00	OutFlight	4	dummy-3	auction	1	372.00
20:03:00	OutFlight	4	dummy-3	auction	1	372.00
20:03:00	InFlight	2	dummy-3	auction	1	373.00
20:03:00	InFlight	2	dummy-3	auction	1	373.00
20:03:00	OutFlight	3	dummy-2	auction	1	335.00
20:03:00	OutFlight	3	dummy-2	auction	1	335.00
20:03:00	InFlight	1	dummy-2	auction	1	398.00
20:03:00	InFlight	1	dummy-2	auction	1	398.00

20:03:00	InFlight	1	dummy-2	auction	1	398.00
20:03:00	InFlight	1	dummy-2	auction	1	398.00
20:03:00	InFlight	3	dummy-2	auction	1	337.00
20:03:00	InFlight	3	dummy-2	auction	1	337.00
20:03:00	InFlight	3	dummy-2	auction	1	337.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	OutFlight	5	dummy-2	auction	1	395.00
20:03:00	InFlight	2	dummy-2	auction	1	373.00
20:03:00	OutFlight	5	dummy-1	auction	1	395.00
20:03:00	OutFlight	5	dummy-1	auction	1	395.00
20:03:00	OutFlight	5	dummy-1	auction	1	395.00
20:03:00	OutFlight	4	dummy-1	auction	1	372.00
20:03:00	OutFlight	4	dummy-1	auction	1	372.00
20:03:00	OutFlight	3	dummy-1	auction	1	335.00
20:03:00	OutFlight	3	dummy-1	auction	1	335.00
20:03:00	InFlight	2	dummy-1	auction	1	373.00
20:03:00	InFlight	2	dummy-1	auction	1	373.00
20:03:00	InFlight	2	dummy-1	auction	1	373.00
20:03:00	InFlight	2	dummy-1	auction	1	373.00
20:03:00	InFlight	2	dummy-1	auction	1	373.00
20:03:00	OutFlight	2	dummy-1	auction	1	332.00
20:03:00	InFlight	1	dummy-1	auction	1	398.00
20:03:00	InFlight	1	dummy-1	auction	1	398.00
20:03:00	InFlight	1	dummy-1	auction	1	398.00
20:04:04	InFlight	1	RiskerWise	auction	6	391.00

20:04:05	InFlight	4	RiskerWise	auction	2	327.00
20:04:05	OutFlight	2	RiskerWise	auction	6	329.00
20:04:05	OutFlight	5	RiskerWise	auction	2	398.00
20:07:00	TampaTowersHotel	2	dummy-7	auction	1	1000.00
20:07:00	TampaTowersHotel	2	dummy-6	auction	3	1000.00
20:07:00	TampaTowersHotel	2	dummy-5	auction	3	1000.00
20:07:00	TampaTowersHotel	2	dummy-4	auction	3	1000.00
20:07:00	TampaTowersHotel	2	dummy-3	auction	2	1000.00
20:07:00	TampaTowersHotel	2	dummy-2	auction	3	1000.00
20:07:00	TampaTowersHotel	2	dummy-1	auction	1	1000.00
20:08:00	TampaTowersHotel	1	dummy-6	auction	2	150.00
20:08:00	TampaTowersHotel	1	dummy-5	auction	2	150.00
20:08:00	TampaTowersHotel	1	dummy-4	auction	2	150.00
20:08:00	TampaTowersHotel	1	dummy-3	auction	1	150.00
20:08:00	TampaTowersHotel	1	dummy-2	auction	3	150.00
20:08:00	TampaTowersHotel	1	dummy-1	auction	2	150.00
20:08:00	TampaTowersHotel	1	RiskerWise	auction	4	150.00
20:09:00	ShorelineShantyHotel	4	dummy-7	auction	3	1.00
20:09:00	ShorelineShantyHotel	4	dummy-4	auction	1	1.00
20:09:00	ShorelineShantyHotel	4	dummy-2	auction	5	1.00
20:09:00	ShorelineShantyHotel	4	dummy-1	auction	1	1.00
20:09:00	ShorelineShantyHotel	4	RiskerWise	auction	6	1.00
20:10:00	TampaTowersHotel	4	dummy-6	auction	3	1.00
20:10:00	TampaTowersHotel	4	dummy-5	auction	1	1.00
20:10:00	TampaTowersHotel	4	dummy-4	auction	3	1.00
20:10:00	TampaTowersHotel	4	dummy-3	auction	2	1.00
20:10:00	TampaTowersHotel	4	dummy-2	auction	1	1.00
20:10:00	TampaTowersHotel	4	dummy-1	auction	2	1.00
20:10:00	TampaTowersHotel	4	RiskerWise	auction	4	1.00

20:11:00	TampaTowersHotel	3	dummy-6	auction	1	1.00
20:11:00	TampaTowersHotel	3	dummy-5	auction	2	1.00
20:11:00	TampaTowersHotel	3	dummy-4	auction	4	1.00
20:11:00	TampaTowersHotel	3	dummy-3	auction	2	1.00
20:11:00	TampaTowersHotel	3	dummy-2	auction	1	1.00
20:11:00	TampaTowersHotel	3	dummy-1	auction	3	1.00
20:11:00	TampaTowersHotel	3	RiskerWise	auction	3	1.00
20:11:36	AlligatorWrestling	1	RiskerWise	dummy-6	1	120.00
20:12:00	ShorelineShantyHotel	2	dummy-7	auction	6	1000.00
20:12:00	ShorelineShantyHotel	2	dummy-6	auction	2	1000.00
20:12:00	ShorelineShantyHotel	2	dummy-5	auction	2	1000.00
20:12:00	ShorelineShantyHotel	2	dummy-4	auction	2	1000.00
20:12:00	ShorelineShantyHotel	2	dummy-3	auction	4	1000.00
20:12:06	AlligatorWrestling	2	dummy-6	RiskerWise	1	60.00
20:12:06	AlligatorWrestling	2	dummy-7	RiskerWise	1	60.00
20:12:07	AmusementPark	3	dummy-3	RiskerWise	1	60.00
20:12:07	AmusementPark	3	dummy-4	RiskerWise	1	60.00
20:12:07	AmusementPark	3	dummy-5	RiskerWise	1	60.00
20:12:07	AmusementPark	3	dummy-6	RiskerWise	1	60.00
20:13:00	ShorelineShantyHotel	1	dummy-7	auction	2	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-6	auction	2	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-5	auction	3	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-4	auction	3	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-3	auction	4	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-2	auction	1	1000.00
20:13:00	ShorelineShantyHotel	1	dummy-1	auction	1	1000.00
20:14:00	ShorelineShantyHotel	3	dummy-7	auction	4	1000.00
20:14:00	ShorelineShantyHotel	3	dummy-6	auction	3	1000.00
20:14:00	ShorelineShantyHotel	3	dummy-5	auction	1	1000.00

20:14:00	ShorelineShantyHotel	3	dummy-4	auction	1	1000.00
20:14:00	ShorelineShantyHotel	3	dummy-3	auction	2	1000.00
20:14:00	ShorelineShantyHotel	3	dummy-2	auction	5	1000.00

Appendix 16: The Fuzzy rules for bid increase prediction

- When SS2 has not yet closed:

(1) IF PT2 is medium and TC2 is slow and PS2 is low and SC2 is slow and time is early THEN bid price increase is small.

(2) IF PT2 is medium and TC2 is slow and PS2 is low and SC2 is slow and time is late THEN bid price increase is medium.

(3) IF PT2 is medium and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(4) IF PT2 is medium and TC2 is slow and PS2 is low and SC2 is fast and time is late THEN bid price increase is big.

(5) IF PT2 is medium and TC2 is slow and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

(6) IF PT2 is medium and TC2 is slow and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.

(7) IF PT2 is medium and TC2 is slow and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.

(8) IF PT2 is medium and TC2 is slow and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

(9) IF PT2 is medium and TC2 is slow and PS2 is medium and SC2 is slow and time is early THEN bid price increase is small.

(10) IF PT2 is medium and TC2 is slow and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.

(11) IF PT2 is medium and TC2 is slow and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.

(12) IF PT2 is medium and TC2 is slow and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

(13) IF PT2 is medium and TC2 is fast and PS2 is low and SC2 is slow and time is early THEN bid price increase is medium.

(14) IF PT2 is medium and TC2 is fast and PS2 is low and SC2 is slow and time is late THEN bid price increase is big.

(15) IF PT2 is medium and TC2 is fast and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(16) IF PT2 is medium and TC2 is fast and PS2 is low and SC2 is fast and time is late THEN bid price increase is big.

(17) IF PT2 is medium and TC2 is fast and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

(18) IF PT2 is medium and TC2 is fast and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.

(19) IF PT2 is medium and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is big.

(20) IF PT2 is medium and TC2 is fast and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

(21) IF PT2 is medium and TC2 is fast and PS2 is medium and SC2 is slow and time is early THEN bid price increase is medium.

(22) IF PT2 is medium and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is big.

(23) IF PT2 is medium and TC2 is fast and PS2 is medium and SC2 is fast and time is early THEN bid price increase is big.

(24) IF PT2 is medium and TC2 is fast and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

(25) IF PT2 is high and TC2 is slow and PS2 is low and SC2 is slow and time is early THEN bid price increase is small.

(26) IF PT2 is high and TC2 is slow and PS2 is low and SC2 is slow and time is late THEN bid price increase is medium.

(27) IF PT2 is high and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(28) IF PT2 is high and TC2 is slow and PS2 is low and SC2 is fast and time is late THEN bid price increase is medium.

(29) IF PT2 is high and TC2 is slow and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

(30) IF PT2 is high and TC2 is slow and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.

(31) IF PT2 is high and TC2 is slow and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.

(32) IF PT2 is high and TC2 is slow and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

(33) IF PT2 is high and TC2 is slow and PS2 is medium and SC2 is slow and time is early THEN bid price increase is medium.

(34) IF PT2 is high and TC2 is slow and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.

(35) IF PT2 is high and TC2 is slow and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.

(36) IF PT2 is high and TC2 is slow and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

(37) IF PT2 is high and TC2 is fast and PS2 is low and SC2 is slow and time is early THEN bid price increase is medium.

(38) IF PT2 is high and TC2 is fast and PS2 is low and SC2 is slow and time is late THEN bid price increase is big.

(39) IF PT2 is high and TC2 is fast and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(40) IF PT2 is high and TC2 is fast and PS2 is low and SC2 is fast and time is late THEN bid price increase is big.

(41) IF PT2 is high and TC2 is fast and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

(42) IF PT2 is high and TC2 is fast and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.

(43) IF PT2 is high and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is big.

(44) IF PT2 is high and TC2 is fast and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

(45) IF PT2 is high and TC2 is fast and PS2 is medium and SC2 is slow and time is early THEN bid price increase is medium.

(46) IF PT2 is high and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is big.

(47) IF PT2 is high and TC2 is fast and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.

(48) IF PT2 is high and TC2 is fast and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

(49) IF PT2 is low and TC2 is slow and PS2 is low and SC2 is slow and time is early THEN bid price increase is small.

(50) IF PT2 is low and TC2 is slow and PS2 is low and SC2 is slow and time is late THEN bid price increase is medium.

(51) IF PT2 is low and TC2 is slow and PS2 is low and SC2 is fast and time is early THEN bid price increase is small.

(52) IF PT2 is low and TC2 is slow and PS2 is low and SC2 is fast and time is late THEN bid price increase is medium.

(53) IF PT2 is low and TC2 is slow and PS2 is high and SC2 is slow and time is early THEN bid price increase is small.

(54) IF PT2 is low and TC2 is slow and PS2 is high and SC2 is slow and time is late THEN bid price increase is medium.

(55) IF PT2 is low and TC2 is slow and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.

(56) IF PT2 is low and TC2 is slow and PS2 is high and SC2 is fast and time is late THEN bid price increase is medium.

(57) IF PT2 is low and TC2 is slow and PS2 is medium and SC2 is slow and time is early THEN bid price increase is small.

(58) IF PT2 is low and TC2 is slow and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.

(59) IF PT2 is low and TC2 is slow and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.

(60) IF PT2 is low and TC2 is slow and PS2 is medium and SC2 is fast and time is late THEN bid price increase is medium.

(61) IF PT2 is low and TC2 is fast and PS2 is low and SC2 is slow and time is early THEN bid price increase is medium.

(62) IF PT2 is low and TC2 is fast and PS2 is low and SC2 is slow and time is late THEN bid price increase is big.

(63) IF PT2 is low and TC2 is fast and PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(64) IF PT2 is low and TC2 is fast and PS2 is low and SC2 is fast and time is late THEN bid price increase is big.

(65) IF PT2 is low and TC2 is fast and PS2 is high and SC2 is slow and time is early THEN bid price increase is medium.

(66) IF PT2 is low and TC2 is fast and PS2 is high and SC2 is slow and time is late THEN bid price increase is big.

(67) IF PT2 is low and TC2 is fast and PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.

(68) IF PT2 is low and TC2 is fast and PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

(69) IF PT2 is low and TC2 is fast and PS2 is medium and SC2 is slow and time is early THEN bid price increase is medium.

(70) IF PT2 is low and TC2 is fast and PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium.

(71) IF PT2 is low and TC2 is fast and PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium.

(72) IF PT2 is low and TC2 is fast and PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

- When SS2 has closed:

(1) IF PT2 is medium and TC2 is slow and time is early THEN bid price increase is small.

(2) IF PT2 is medium and TC2 is slow and time is late THEN bid price increase is medium

(3) IF PT2 is medium and TC2 is fast and time is early THEN bid price increase is medium

(4) IF PT2 is medium and TC2 is fast and time is late THEN bid price increase is big.

(5) IF PT2 is low and TC2 is slow and time is early THEN bid price increase is small.

(6) IF PT2 is low and TC2 is slow and time is late THEN bid price increase is medium.

(7) IF PT2 is low and TC2 is fast and time is early THEN bid price increase is medium.

(8) IF PT2 is low and TC2 is fast and time is late THEN bid price increase is big.

(9) IF PT2 is high and TC2 is slow and time is early THEN bid price increase is small.

(10) IF PT2 is high and TC2 is slow and time is late THEN bid price increase is medium.

(11) IF PT2 is high and TC2 is fast and time is early THEN bid price increase is medium.

(12) IF PT2 is high and TC2 is fast and time is late THEN bid price increase is big.

- When TT2 has closed:

(1) IF PS2 is medium and SC2 is slow and time is early THEN bid price increase is small.

(2) IF PS2 is medium and SC2 is slow and time is late THEN bid price increase is medium

(3) IF PS2 is medium and SC2 is fast and time is early THEN bid price increase is medium

(4) IF PS2 is medium and SC2 is fast and time is late THEN bid price increase is big.

(5) IF PS2 is low and SC2 is slow and time is early THEN bid price increase is small.

(6) IF PS2 is low and SC2 is slow and time is late THEN bid price increase is medium.

(7) IF PS2 is low and SC2 is fast and time is early THEN bid price increase is medium.

(8) IF PS2 is low and SC2 is fast and time is late THEN bid price increase is big.

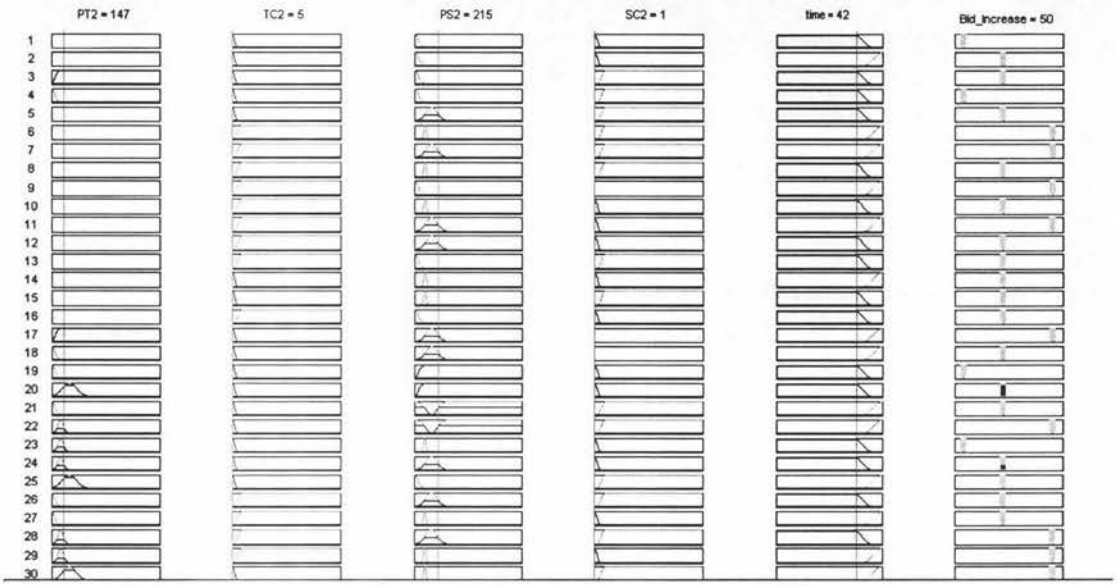
(9) IF PS2 is high and SC2 is slow and time is early THEN bid price increase is small.

(10) IF PS2 is high and SC2 is slow and time is late THEN bid price increase is medium.

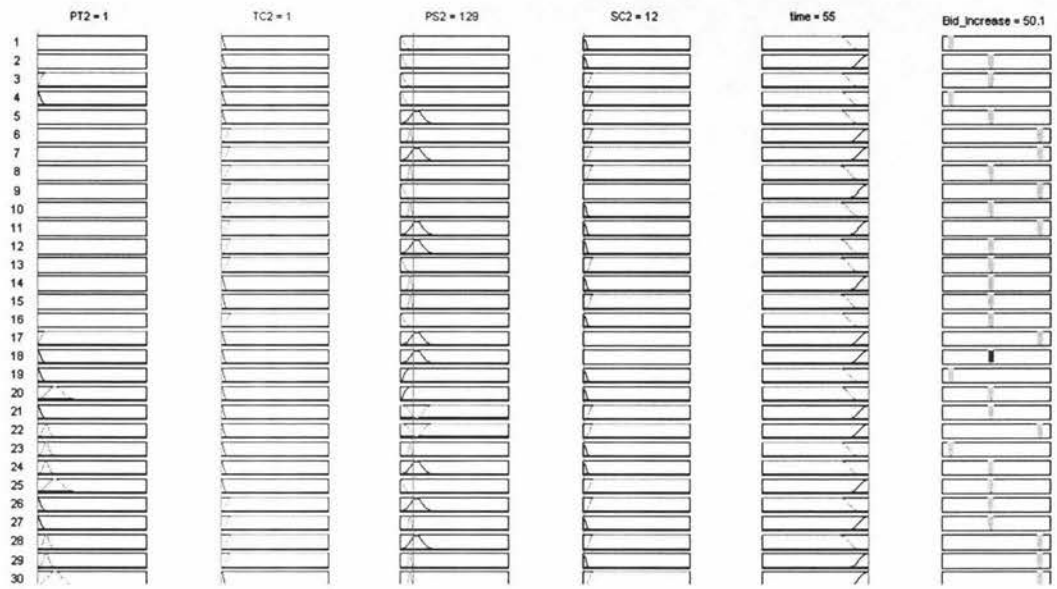
(11) IF PS2 is high and SC2 is fast and time is early THEN bid price increase is medium.

(12) IF PS2 is high and SC2 is fast and time is late THEN bid price increase is big.

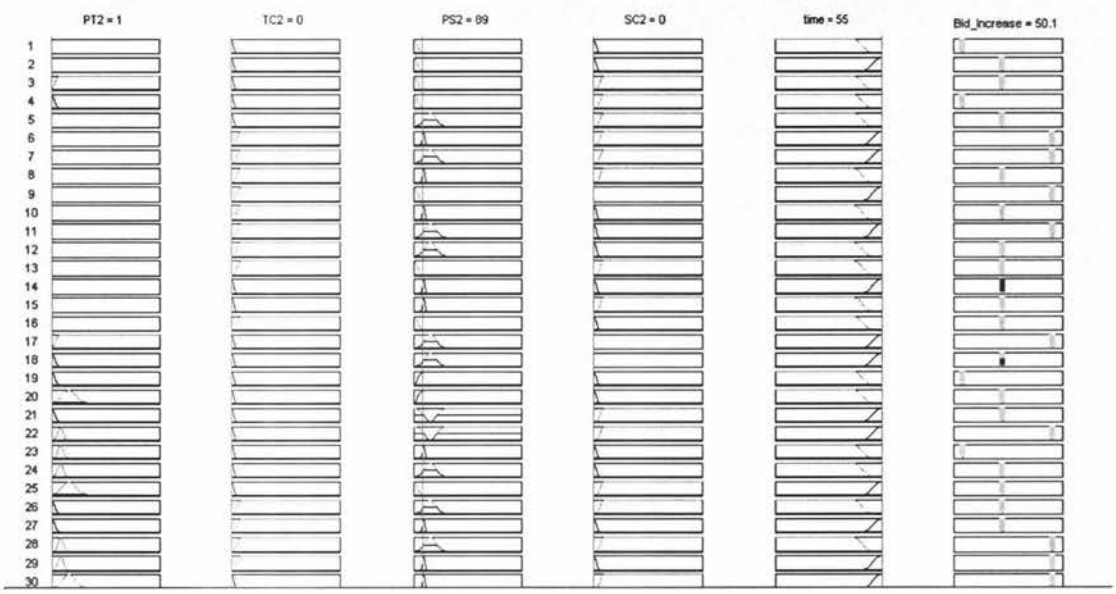
Appendix 17: Fuzzy logic system was tested with the data of TAC game 17610 at minute 5



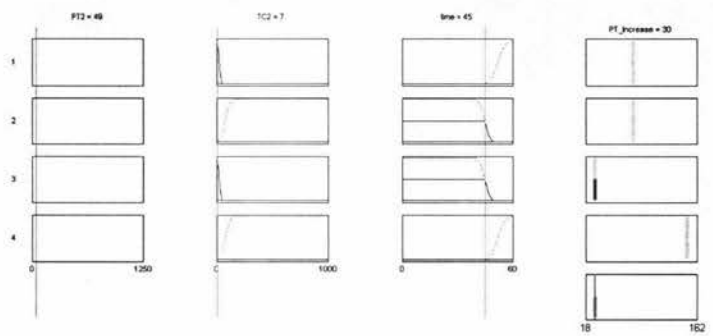
Appendix 18: Fuzzy logic system was tested with the data of TAC game 17622 at minute 3



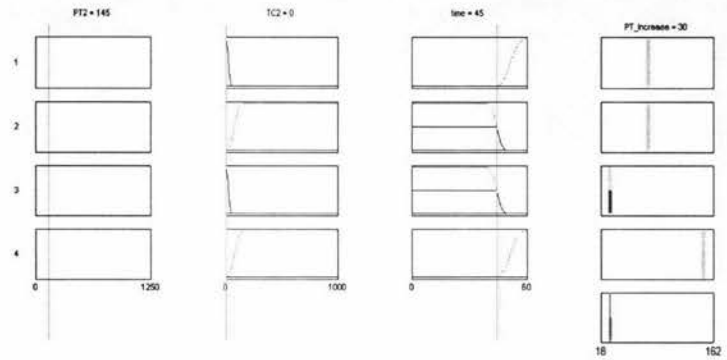
Appendix 19: Fuzzy logic system was tested with the data of TAC game 17624 at minute 3



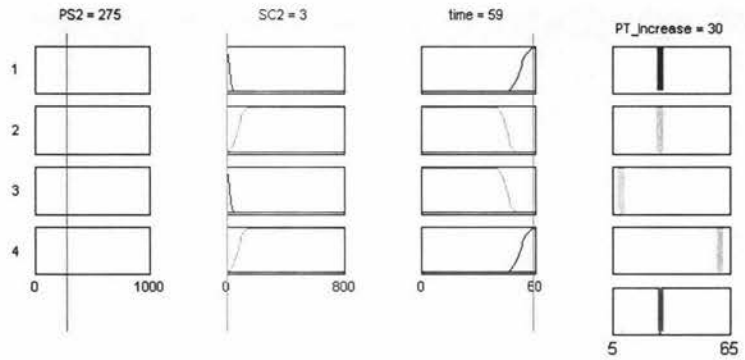
Appendix 20: Fuzzy logic system was tested with the data of TAC game 17622 at minute 5



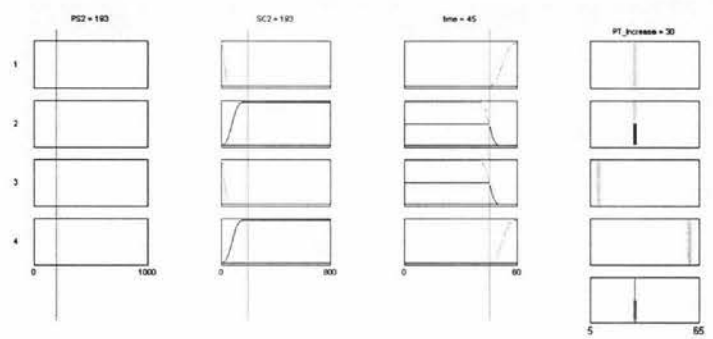
Appendix 21: Fuzzy logic system was tested with the data of TAC game 17614 at minute 7



Appendix 22: Fuzzy logic system was tested with the data of TAC game 17612 at minute 6



Appendix 23: Fuzzy logic system was tested with the data of TAC game 17614 at minute 2



Appendix 24: naughty

```
package se.sics.tac.aw;
import se.sics.tac.util.ArgEnumerator;
import java.util.logging.*;
import java.lang.Math;
import java.lang.*;

public class naughty extends AgentImpl {

  private static final Logger log =
    Logger.getLogger("se.sics.tac.aw.naughty");

  private float[] prices;
  private float[]p = new float[28];
  private int inFlight;
  private int outFlight;
  private int hotel;
  private int ET[] = {-1,-1,-1};
  private int entOwn[][] = new int[4][3];
  private int hotelOwn[][] = new int[4][2];
  private int entBonus[][] = new int[8][3];
  private int entMaxBonus[] = new int [3];
  private int highest = 0;
  private int entMean = 0;
  public  int []EO = {0,0,0};
  private int e1;
  private int e2;
  private int e3;
  boolean allocationFlag = false;
  boolean searchFlag = false;
  protected void init(ArgEnumerator args) {
```

```
  prices = new float[agent.getAuctionNo()];
}
private Node ClientNode;
public int[] BArrival = new int[8];
public int[] BDeparture = new int[8];
public int[] Aday = new int[8];
public int[] Dday = new int[8];
public int[] type = new int[8];
public int hotelType = -1;
public int countClient = 0;
public Heap theHeap = new Heap(3);

public void gameStarted() {
  log.fine("Game risk " + agent.getGameID() + " started!");
  init();
  sellEnt();
  buyCheapHotels();
    while (agent.getGameTime() < 24*1000) {
      search();
    }
  Allocation();
  sendBids();
}

private void init() {
  for (int j = 0; j <28; j++)
  {
    p[j]=0;
  }
  for (int j = 0; j <8; j++)
  {
    for (int i = TACAgent.E1; i <= TACAgent.E3; i++) {
      entBonus[j][i-3] = 0;
```

```
      entMaxBonus[i-3] = 0;
    }
  }
  for (int d = 0; d < 4; d++) {
    for (int i = TACAgent.E1; i <= TACAgent.E3; i++) {
      entOwn[d][i-3] = 0;
    }
    for (int t = 0; t < 2; t++) {
      hotelOwn[d][t] = 0;
    }
  }
}

protected void search()
{
  float smax = -10000;
  float total = 0;
  int count=0;
  int [] sarrival = new int [8];
  int [] sdepartuer = new int [8];
  int [] shotel = new int [8];
  ClientNode = new Node();
  float [] s = new float [8];
  for (int j = 0; j <8; j++)
  {
    s[j]= 0;
    sarrival[j] = -1;
    sdepartuer[j] = -1;
    shotel[j] = 0;
    BArrival[j]=-1;
    BDeparture[j]=-1;
    Aday[j]=-1;
    Dday[j]=-1;
```

```
    type[j]=-1;
  }
  for (int j = 0; j<28 ; j++) {
   int aID = agent.getAuctionID(j);
   Quote quote =agent.getQuote(j);
   p[j]=quote.getAskPrice();
   for (countClient = 0; countClient<8; countClient++){
    int i=countClient;
    int Arrival =agent.getClientPreference(i, TACAgent.ARRIVAL);
    int Departure = agent.getClientPreference(i, TACAgent.DEPARTURE);
    float Hotelbouns = agent.getClientPreference(i, TACAgent.HOTEL_VALUE);
    e1 = agent.getClientPreference(i, TACAgent.E1);
    e2 = agent.getClientPreference(i, TACAgent.E2);
    e3 = agent.getClientPreference(i, TACAgent.E3);
    float maxTemp = 0;
    for (Aday[i] = 1; Aday[i] <= 4; Aday[i] ++) {
     for (Dday[i] = Aday[i]+1; Dday[i] <=5; Dday[i] ++) {
      float temp = 1000+100*(Math.abs(Aday[i]-Arrival)+ Math.abs(Dday[i]-
Departure))-p[Aday[i]-1]-p[Dday[i]+2]
                  +hotelScore(Aday[i], Dday[i],Hotelbouns)
                  +funbounds(Aday[i], Dday[i]);
       if (temp <0) continue;
        if (temp > maxTemp) //find local max for a client
        {
         maxTemp = temp;
         shotel[i] = hotelType;
         sarrival[i] = Aday[i];
         sdepartuer[i] = Dday[i];
         ClientNode.set(i, Aday[i], Dday[i], hotelType, temp);
        }
       }
     }
    }
```

```
  }
   total = 0;
   theHeap.insertNode(ClientNode);
}

//get all the HOTEL rooms owned

private int getHotelOwn(int day, int type) {
  int auction = agent.getAuctionFor(1, type, day);
  int auctionID = agent.getAuctionID(auction);
  int hotel = agent.getOwn(auctionID);
  return hotel;
}

private void getEntOwn(int day, int type) {
 int auction = agent.getAuctionFor(2, type, day);
 entOwn[day-1][type-3] = agent.getOwn(auction);
}

//bid all the hotel for $1 each
private void buyCheapHotels() {
 for (int day = 1; day <= 4; day++) {
   for (int type = 0; type <=1; type++) {
     int auction = agent.getAuctionFor(1, type, day);
         Bid bid = new Bid(auction);
      bid.addBidPoint(8,1);
            agent.submitBid(bid);
   }
 }
}

 private void sellEnt() {
  for (int client = 0; client < 8; client ++) {
```

```
    for (int i = 1; i <= 3; i++) {
      entBonus[client][i-1] = agent.getClientPreference(client, i);
      entMean += entBonus[client][i-1];
    }
  }
  entMean = entMean/24;
  for (int i = 1; i <= 3; i++) {
    for (int client = 0; client < 8; client ++) {
      if (entMaxBonus[i-1] < entBonus[client][i-1]) {
      entMaxBonus[i-1] = entBonus[client][i-1];
        if (highest < entMaxBonus[i-1]) {
          highest = entMaxBonus[i-1];
        }
      }
    }
  }
  for (int day = 1; day <= 4; day++) {
    for (int type = 1; type <=3; type++) {
     int auction = agent.getAuctionFor(2, type, day);
     entOwn[day-1][type-1] = agent.getOwn(auction);
     if (entOwn[day-1][type-1] > 0) {
       Bid bid = new Bid(auction);
         bid.addBidPoint(-entOwn[day-1][type-1],18000);
            agent.submitBid(bid);
     }
    }
  }
}


protected float funbounds (int Aday, int Dday)
{
  int countDay = 0;
```

```
 float total_fun_bonus = 0;
 int j;
 for ( j = 0; j < (Dday-Aday) && j < 3; j++)
 {
  total_fun_bonus += ET[j];
  countDay++;
 }
 j=0;
 while (j<3) {
  ET[j]=-1;
  j++;
 }
 float EntScore = total_fun_bonus - 80*countDay;
 return EntScore;
}

protected float hotelScore (int Aday, int Dday, float Hotelbouns)
{
   int totaltower = 0;
   int totalshanty = 0;
   int aucid;
   for (int i = Aday; i < Dday; i++)
   {
    totaltower += p[i-1+12];
    totalshanty += p[i-1+8];
   }
   int g = (int)Hotelbouns - totaltower;
   int b = 0 - totalshanty;
   if (g > b){
    hotelType = TACAgent.TYPE_GOOD_HOTEL;
    return g;
   } else {
    hotelType = TACAgent.TYPE_CHEAP_HOTEL;
```

```
    return b;
   }
 }

private float MaxHotelBonus () {
 float Max = 0;
 for (int Client = 0; Client<8; Client++){
     float Hotelbouns = agent.getClientPreference(Client,
                                TACAgent.HOTEL_VALUE);
     if (Max < Hotelbouns) Max = Hotelbouns;
    }
    return Max;
 }

  private float MinHotelBonus () {
 float Min = 200;
 for (int Client = 0; Client<8; Client++){
     float Hotelbouns = agent.getClientPreference(Client,
                                TACAgent.HOTEL_VALUE);
     if (Min < Hotelbouns) Min = Hotelbouns;
    }
    return Min;
 }

public void quoteUpdated(Quote quote) {
 int auction = quote.getAuction();
 int auctionCategory = agent.getAuctionCategory(auction);
 if (auctionCategory == TACAgent.CAT_HOTEL) {
  int amount;
  float p = quote.getAskPrice(); //ask price for this auction
  int alloc = agent.getAllocation(auction);
  int day = agent.getHotelAuctionDay(auction);
  int Type = agent.getHotelAuctionType(auction);
```

```
int hotel = getHotelOwn(day, Type);
if (Type == 0 ) { //find the opposite hotel type
 Type =1;
} else {Type = 0; }

int ahotel = getHotelOwn(day, Type);
int aauction = agent.getAuctionFor(1, Type, day);
int aalloc = agent.getAllocation(auction);
amount = alloc - hotel - ahotel;
if (amount > 0 && quote.hasHQW(agent.getBid(auction)) &&
        quote.getHQW() < amount && quote.hasHQW(agent.getBid(aauction)) &&
        quote.getHQW() < amount) {
 int aauctionID = agent.getAuctionID(aauction);
 Quote aQuote =  agent.getQuote(aauctionID);
 float aPrice = aQuote.getAskPrice();
 if ((p - aPrice) > MinHotelBonus()) {
  if (Type == 1) {
   for (double t = 3.9; t < 11.9; t++) {
    if ((agent.getGameTime() - t*60*1000) < 5*1000) {
     aPrice += 50;
    }
   }
   Bid bid = new Bid(aauction);
   bid.addBidPoint(amount, aPrice);
   agent.submitBid(bid);
  } else {
   prices[auction] = p + 20;
   for (double t = 3.8; t < 11.8; t++) {
    if ((agent.getGameTime() - t*60*1000) < 5*1000) {
     prices[auction] = p+150;
    }
   }
   if (prices[auction] > 1350) prices[auction] = 1350;
```

```
    Bid bid = new Bid(auction);
    bid.addBidPoint(amount, prices[auction]);
    agent.submitBid(bid);
   }
  }
  int gauction = agent.getAuctionFor(0, Type, day);
  prices[gauction] = p + 20;
  for (double t = 3.8; t < 11.8; t++) {
   if ((agent.getGameTime() - t*60*1000) < 5*1000) {
    prices[gauction] = p+150;//+= 150;
   }
  }
    if (prices[gauction] > 1350) prices[auction] = 1350;
    Bid bid = new Bid(auction);
    bid.addBidPoint(amount, prices[gauction]);
    agent.submitBid(bid);
  }
} else if (auctionCategory == TACAgent.CAT_ENTERTAINMENT) {
 int alloc = agent.getAllocation(auction) - agent.getOwn(auction);
 if (alloc != 0) {
        Bid bid = new Bid(auction);
        if (alloc < 0 && (agent.getGameTime() > 4*60*1000)) {
         prices[auction] = 200f - (agent.getGameTime() * 120f) / 720000;
  }
        else {
         prices[auction] = 50f + (agent.getGameTime() * 100f) / 720000;
  }
  if (prices[auction]> entMean) {
   prices[auction] = entMean;
  }
  bid.addBidPoint(alloc, prices[auction]);
        agent.submitBid(bid);
 }
```

```
  }
}

public void quoteUpdated(int auctionCategory) {
  log.fine("All quotes for "
           + agent.auctionCategoryToString(auctionCategory)
           + " has been updated");
}

public void bidUpdated(Bid bid) {
  log.fine("Bid Updated: id=" + bid.getID() + " auction="
           + bid.getAuction() + " state="
           + bid.getProcessingStateAsString());
  log.fine("      Hash: " + bid.getBidHash());
}

public void bidRejected(Bid bid) {
  log.warning("Bid Rejected: " + bid.getID());
  log.warning("     Reason: " + bid.getRejectReason()
              + " (" + bid.getRejectReasonAsString() + ')');
}

public void bidError(Bid bid, int status) {
  log.warning("Bid Error in auction " + bid.getAuction() + ": " + status
              + " (" + agent.commandStatusToString(status) + ')');
}

public void gameStopped() {
  log.fine("Game Stopped! ");
}

public void auctionClosed(int auction) {
  log.fine("*** Auction " + auction + " closed! ");
```

```
}

private void sendBids() {
  for (int i = 0, n = agent.getAuctionNo(); i < n; i++) {
    int alloc = agent.getAllocation(i) - agent.getOwn(i);
    log.fine("in sendBids()  "+"auction: "+i+" alloc:  "+ alloc);
    float price = -1;
    switch (agent.getAuctionCategory(i)) {
      case TACAgent.CAT_FLIGHT:
              if (alloc > 0) {
                price = 1000;
              }
             break;
      case TACAgent.CAT_ENTERTAINMENT:
              if (alloc < 0) {
                price = 1800;
              } else if (alloc > 0) {
                price = 50;
                prices[i] = 50f;
            }
              break;
      default:
              break;
    }
   if (price > 0) {
         Bid bid = new Bid(i);
           bid.addBidPoint(alloc, price);
           agent.submitBid(bid);
   }
 }
}
```

```
private void Allocation() {
  for (int a = 0; a < 8; a++) {
    int auction=-1;
    auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                  TACAgent.TYPE_INFLIGHT,
                                  theHeap.heapArray[0].getArrival(a));
    agent.setAllocation(auction, agent.getAllocation(auction) + 1);
    auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                  TACAgent.TYPE_OUTFLIGHT,
                                  theHeap.heapArray[0].getDepartuer(a));
    agent.setAllocation (auction, agent.getAllocation(auction) + 1);
    for (int d = theHeap.heapArray[0].getArrival(a); d <
                        theHeap.heapArray[0].getDepartuer(a); d++) {
      auction = agent.getAuctionFor(TACAgent.CAT_HOTEL,
                                    theHeap.heapArray[0].getHotel(a), d);
      agent.setAllocation (auction, agent.getAllocation(auction) + 1);
    }

    int eType = -1;
    while((eType = nextEntType(a, eType)) > 0) {
      auction = bestEntDay(theHeap.heapArray[0].getArrival(a),
                           theHeap.heapArray[0].getDepartuer(a), eType);
      agent.setAllocation(auction, agent.getAllocation(auction) + 1);
    }
  }
}

private int bestEntDay(int inFlight, int outFlight, int type) {
  for (int i = inFlight; i < outFlight; i++) {
    int auction = agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                                      type, i);
    if (agent.getAllocation(auction) < agent.getOwn(auction)) {
          return auction;
```

```
    }
   }
   return agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                              type, inFlight);
  }

  private int nextEntType(int client, int lastType) {
   int e1 = agent.getClientPreference(client, TACAgent.E1);
   int e2 = agent.getClientPreference(client, TACAgent.E2);
   int e3 = agent.getClientPreference(client, TACAgent.E3);
   if ((e1 > e2) && (e1 > e3) && lastType == -1)
    return TACAgent.TYPE_ALLIGATOR_WRESTLING;
   if ((e2 > e1) && (e2 > e3) && lastType == -1)
    return TACAgent.TYPE_AMUSEMENT;
   if ((e3 > e1) && (e3 > e2) && lastType == -1)
    return TACAgent.TYPE_MUSEUM;
   return -1;
  }

  public static void main (String[] args) {
   TACAgent.main(args);
  }
}
```

Appendix 25: Sacrifice

```
package se.sics.tac.aw;
import se.sics.tac.util.ArgEnumerator;
import java.util.logging.*;

public class Sacrifice extends AgentImpl {

 private static final Logger log =
  Logger.getLogger("se.sics.tac.aw.Sacrifice");

 private float[] prices;

 protected void init(ArgEnumerator args) {
  prices = new float[agent.getAuctionNo()];
 }

 public void quoteUpdated(Quote quote) {
  int auction = quote.getAuction();
  int auctionCategory = agent.getAuctionCategory(auction);
  if (auctionCategory == TACAgent.CAT_HOTEL) {
   int alloc = agent.getAllocation(auction);
   if (alloc > 0 && quote.hasHQW(agent.getBid(auction)) &&
         quote.getHQW() < alloc) {
       Bid bid = new Bid(auction);
       prices[auction] = 50;   bid.addBidPoint(alloc, prices[auction]);
       agent.submitBid(bid);
   }
  } else if (auctionCategory == TACAgent.CAT_ENTERTAINMENT) {
   int alloc = agent.getAllocation(auction) - agent.getOwn(auction);
   if (alloc != 0) {
       Bid bid = new Bid(auction);
       if (alloc < 0)
```

```
            prices[auction] = 30;
          else
            prices[auction] = 150;
          bid.addBidPoint(alloc, prices[auction]);
          agent.submitBid(bid);
      }
    }
  }

  public void quoteUpdated(int auctionCategory) {
    log.fine("All quotes for "
             + agent.auctionCategoryToString(auctionCategory)
             + " has been updated");
  }

  public void bidUpdated(Bid bid) {
    log.fine("Bid Updated: id=" + bid.getID() + " auction="
             + bid.getAuction() + " state="
             + bid.getProcessingStateAsString());
    log.fine("      Hash: " + bid.getBidHash());
  }

  public void bidRejected(Bid bid) {
    log.warning("Bid Rejected: " + bid.getID());
    log.warning("     Reason: " + bid.getRejectReason()
                + " (" + bid.getRejectReasonAsString() + ')');
  }

  public void bidError(Bid bid, int status) {
    log.warning("Bid Error in auction " + bid.getAuction() + ": " + status
                + " (" + agent.commandStatusToString(status) + ')');
  }
```

```
public void gameStarted() {
  buyEnt();
 calculateAllocation();
  while (agent.getGameTime() < 3*60*1000) {
   log.fine("Waiting...");
  }
 sendBids();
}

 private void buyEnt(){
  for (int day = 1; day <= 4; day++) {
   for (int type = 1; type <=3; type++) {
   int auction = agent.getAuctionFor(2, type, day);
      Bid bid = new Bid(auction);
      bid.addBidPoint(2,20000);
      agent.submitBid(bid);
    }
   }
}

public void gameStopped() {
  log.fine("Game Stopped!Realy");
}

public void auctionClosed(int auction) {
 log.fine("*** Auction " + auction + " closed! ");
}

private void sendBids() {
  for (int i = 0, n = agent.getAuctionNo(); i < n; i++) {
   int alloc = agent.getAllocation(i) - agent.getOwn(i);
   float price = -1f;
   switch (agent.getAuctionCategory(i)) {
```

```
    case TACAgent.CAT_FLIGHT:
        if (alloc > 0) {
          price = 1000;
        }
        break;
    case TACAgent.CAT_HOTEL:
        if (alloc > 0) {
          price = 50;
          prices[i] =50;
        }
        break;
    case TACAgent.CAT_ENTERTAINMENT:
        if (alloc < 0) {
          price = 50;
          prices[i] = 50f;
        } else if (alloc > 0) {
          price = 150;
          prices[i] = 150f;
        }
        break;
    default:
        break;
    }
    if (price > 0) {
        Bid bid = new Bid(i);
        bid.addBidPoint(alloc, price);
        agent.submitBid(bid);
    }
  }
}

private void calculateAllocation() {
      log.fine("I'm in calculation");
```

```
 for (int i = 0; i < 8; i++) {
  int inFlight = agent.getClientPreference(i, TACAgent.ARRIVAL);
  int outFlight = agent.getClientPreference(i, TACAgent.DEPARTURE);
  int hotel = agent.getClientPreference(i, TACAgent.HOTEL_VALUE);
  int type;

  int auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                    TACAgent.TYPE_INFLIGHT, inFlight);
  agent.setAllocation(auction, agent.getAllocation(auction) + 1);
  auction = agent.getAuctionFor(TACAgent.CAT_FLIGHT,
                                TACAgent.TYPE_OUTFLIGHT, outFlight);
  agent.setAllocation(auction, agent.getAllocation(auction) + 1);

  if (hotel > 70) {
      type = TACAgent.TYPE_GOOD_HOTEL;
  } else {
      type = TACAgent.TYPE_CHEAP_HOTEL;
  }

  for (int d = inFlight; d < outFlight; d++) {
      auction = agent.getAuctionFor(TACAgent.CAT_HOTEL, type, d);
      log.finer("Adding hotel for day: " + d + " on " + auction);
      agent.setAllocation(auction, agent.getAllocation(auction) + 1);
  }

  int eType = -1;
  while((eType = nextEntType(i, eType)) > 0) {
      auction = bestEntDay(inFlight, outFlight, eType);
      log.finer("Adding entertainment " + eType + " on " + auction);
      agent.setAllocation(auction, agent.getAllocation(auction) + 1);
  }
 }
}
```

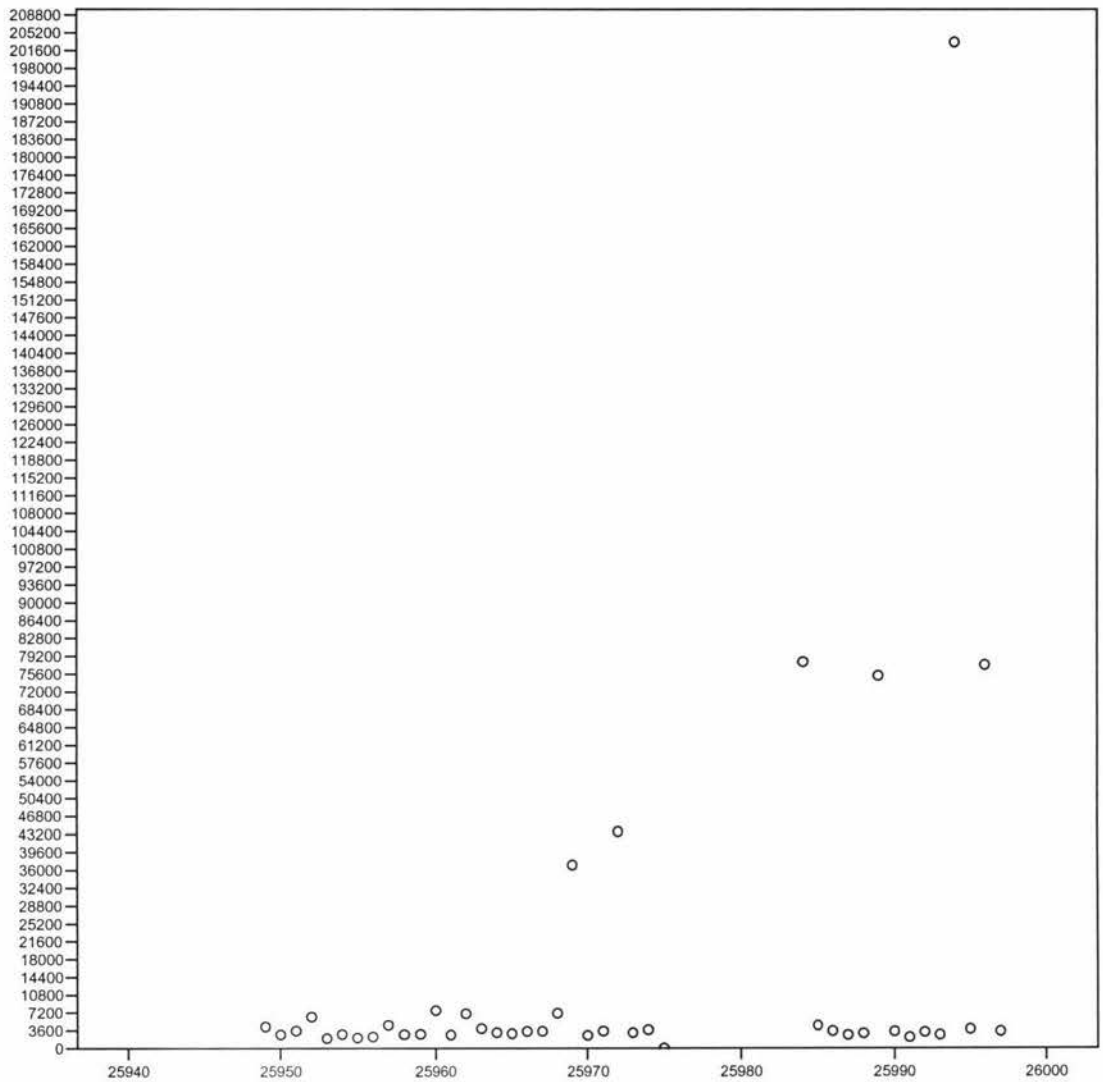
```
  private int bestEntDay(int inFlight, int outFlight, int type) {
    for (int i = inFlight; i < outFlight; i++) {
      int auction = agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                                        type, i);
      if (agent.getAllocation(auction) < agent.getOwn(auction)) {
          return auction;
      }
    }
  return agent.getAuctionFor(TACAgent.CAT_ENTERTAINMENT,
                             type, inFlight);
  }

  private int nextEntType(int client, int lastType) {
    int e1 = agent.getClientPreference(client, TACAgent.E1);
    int e2 = agent.getClientPreference(client, TACAgent.E2);
    int e3 = agent.getClientPreference(client, TACAgent.E3);

    if ((e1 > e2) && (e1 > e3) && lastType == -1)
      return TACAgent.TYPE_ALLIGATOR_WRESTLING;
    if ((e2 > e1) && (e2 > e3) && lastType == -1)
      return TACAgent.TYPE_AMUSEMENT;
    if ((e3 > e1) && (e3 > e2) && lastType == -1)
      return TACAgent.TYPE_MUSEUM;
    return -1;
  }

  public static void main (String[] args) {
    TACAgent.main(args);
  }
}
```

Appendix 26: Statistics for naughty



Scores of 41 games naughty played

Avg Score	Avg Score - Zero
15558.19	15947.14

The utility and score of the last 41 games naughty played

Game	Utility	Score		Game	Utility	Score
25949	7428	4367		25950	7322	2758
25953	8141	1989		25954	7914	2802
25957	7547	4681		25958	7316	2776
25961	7956	2693		25962	7868	6991.94
25965	7846	2947		25966	8604	3390.63
25969	6411	37009		25970	8062	2588
25973	8514	3110.72		25974	8081	3727.69
25985	8940	4593		25986	8687	3470
25989	4107	75205		25990	8205	3384
25993	7384	2697		25994	7807	203382
25951	7644	3526		25952	7522	6359
25955	7645	2103		25956	7462	2310
25959	7679	2855		25960	8234	7636
25963	8808	3975		25964	7716	3145
25967	8024	3425		25968	7748	7091
25971	7931	3458		25972	8793	43695
25975	0	0		25984	6756	78018
25987	8402	2680		25988	8420	2993
25991	7125	2188		25992	8592	3231.8
25995	8700	3850		25996	6843	77386
25997	8423	3399				

Appendix 27: Scores Table

This table is copied from http://www.sics.se/tac/server/ on 2.43pm 22-Apr-04

Position	Agent	Avg Score	Games Played	Zero Games
1	naughty	15862.17	40	1
2	stupidAgent	15417.92	6	1
3	RiskerWise	4648.75	87	2
4	awander	4581.29	2	0
5	sayliab	4494.40	4	0
6	myudd	4425.00	1	0
7	tangr	4401.37	1	0
8	yup	4332.71	3	0
9	mlspa	4310.74	1	0
10	mbbeaubi	4164.44	4	0
11	ndellofa	4146.43	3	0
12	sjtan	4140.71	2	0
13	splaza	4118.69	2	0
14	rushi	4103.15	3	0
15	kdormito	4095.77	4	0
16	eleftheria	4016.50	2	0
17	junzh	3962.41	4	0
18	Aristidis	3943.00	2	0
19	mhl5	3942.49	4	0
20	uhlr	3886.67	2	0
21	sccheung	3879.14	2	0
22	stubbsj	3850.02	1	0
23	bgkim	3838.60	2	0
24	jjwashbu	3802.34	4	0