An Ant Colony Simulator

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

Delan Ren

2012

Table of Contents

Abstract 4

Figures 5

Chapter 1: Introduction 6

1.1 Ant Colony Optimisation 6

1.1.1 Basic Theory 7

1.2 Previous Research 8

1.2.1 Double Bridge Lab 8

1.2.2 Ant Colony Simulator 9

1.2.3 Problem Detect From Previous Research 10

1.3 Improve ant colony algorithms 13

Chapter 2 TSP (Travel Salesman Problem) 14

2.1 Introduction 14

2.2 Overview 15

2.3 Algorithms 16

Chapter 3 Ant Colony Algorithms in the discrete space optimizations 17

3.1 Background and Algorithms 17

3.2 Ant-Q System 18

3.2 Sensational and consciousness ant colony algorithm 25

3.2.1 Ant Search First Stage 26

3.2.2 Ant Search Middle Stage 27

3.2.3 Ant Search Final Stage 29

3.2.4 Self-Adaptive Update Pheromone Policy 30

3.2.5 Pseudo Code For Algorithms And Structure 32

Chapter 4 Simulator with New Algorithms 35
4.1 Introduction 35
4.2 Algorithm Design 37
4.3 New Simulator Implementation 40
4.3.1 Class diagram structures 40
4.3.2 New features for simulator 44
4.4 Empirical Observations 51

Chapter 5 Lessons Learned and Future Work 57
5.1 Lessons Learned from the Research 57
5.2 Future Work 58

Chapter 6 Conclusion 59

Bibliography 61

Abstract

In recent years, ant colony algorithms have become more popular research topics in the artificial intelligence area of computer science. This biological modeling algorithm simulates the natural behavior of an ant colony looking for food among the insect kingdom. This algorithm was initially proposed by Marco Dorigo in 1992 in his PhD thesis – the first algorithm was aiming to search for an optimal path in a graph based on the behavior of ants seeking a path between their colony and a source of food. The original idea has since been diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behavior of ants (Ant colony optimization, 2010).

The famous science journal “Nature” has published articles relating to ant colony algorithms several times, and lots of other publishers around the world have produced quite a few books for ant colony optimization. These days, ant colony algorithms have become a hot topic for the international artificial intelligence computing.

The biological modeling optimization algorithm is an important branch in the artificial intelligence research area, which includes simulation biosphere natural selection and heredity mechanism genetic algorithm (Duan, 2005). This thesis continues research on the original ant colony algorithm, and creates a simulator to handle the ant colony’s natural behavior to find food.

Figures

Diagram 1.1: Ant Colony Theory ……. ……. 7
Diagram 2.1: Travel Salesman Problem Graph…….. 14
Diagram 3.1: Basic ant colony algorithms under different parameters…….…… 22
Diagram 3.2: Ant -Q algorithms under different parameters ……..…… 23
Diagram 3.3: The best iteration of Ant-Q algorithm……..…… 24
Diagram 3.4: The worst iteration of Ant-Q algorithm…….. 24
Diagram 3.5: Weber's law…… 27
Diagram 4.1: Classes structure for the console application project…….. 41
Diagram 4.2: Classes Structure for entity project…… 42
Diagram 4.3: Class Structures for algorithms project…….. 43
Diagram 4.4: Boundary of the Map…… 44
Diagram 4.5 Boundary of the infinite map…….. 45
Diagram 4.6 Log ant movement data…….. 47
Diagram 4.7: Obstacle Model - Initial Stage…….. 48
Diagram 4.8: Obstacle Model - Middle Stage…….. 49
Diagram 4.9: Obstacle Model – Final Stage…….. 50
Diagram 4.10: Run simulator using default settings…… 52
Diagram 4.11: Simulator Observation Series Ⅰ …… 52
Diagram 4.12: Simulator Observation Series Ⅱ …… 55

Chapter 1: Introduction

Ant colonies, and more generally social insect societies, are distributed systems that, in spite of the simplicity of their individuals, present a highly structured social organization. As a result of this organization, ant colonies can accomplish complex tasks that in some cases far exceed the individual capabilities of a single ant (Dorigo & Stutzle, Ant Colony Optimization, 2004). Biological scientists have discovered that the intelligence of evey single ant is not higher under their long term research. In addition, they do not appear to be under a central control; however, they all work together to find food, and build up their nest for the new generation. The ant colony presents higher intelligence and more ability than each individual ant in their society (Duan, 2005).

1.1 Ant Colony Optimisation

The field of "ant algorithms" studies and models derived from the observation of real ants' behavior, and uses these observations as a source of inspiration for the design of novel algorithms for the solution of optimization and distributed control problems. The main idea is that the self-organizing principles which allow the highly coordinated behavior of real ants can be exploited to coordinate populations of artificial agents that collaborate to solve computational problems. Several different aspects of the behavior of ant colonies have inspired different kinds of ant algorithms. Examples are foraging, division of labor, brood sorting, and cooperative transport. In all these examples, ants coordinate their activities via stigmergy, a form of indirect communication mediated by modifications of the environment (Dorigo & Stutzle, Ant Colony Optimization, 2004).

One of the most successful examples of ant algorithms is known as "ant colony optimization," or ACO (Dorigo & Stutzle, Ant Colony Optimization, 2004). Ant Colony Optimization (ACO) is a paradigm for mathematics algorithms to sort the combinatorial optimization problems. Ant colony optimization had successfully applied to solve the travel salesman problem (TSP), and this algorithm of the ant colony optimization had used distributional parallel computation mechanism.

1.1.1 Basic Theory

The basic theory of finding the food sources of the ant colony based on the following criteria:

1. Artificial ants communicate based on pheromones (Animal Behavior/Pheromones in ants and bees, 2007). Each artificial ant reacts depending on the local environment, and its behavior only affects the local environment.

2. The behavior of the ant depends on its own mode, because the ant is the biological animal, its behavior in fact is self-adaptation according to the genes.

3. From the ant colony view, each ant has random behavior; however, the ant colony can self-organize to form the highly cohesive group behavior. On the other hand, from the ant individual view, each ant makes independent choices depending on the environment.

Ant colony optimization includes two basic steps: the adaptation stage and cooperate stage (Thomas & Dorigo, 2004). Under the adaptation stage, all the ants choose the path to go depending on the pheromone and environment. The more ants pass from the node, the node has more pheromone, and this node's path gets easier to be chosen by other ants. If the node has no ant pass for a period of time, the node has less or no pheromone, and this node's path gets more difficult to be chosen by other ants. Under the cooperate stage, all the ants share the information, which expect to find a better solution to solve the problem. The ant colony behavior is the behavior that moves from disorder towards order.

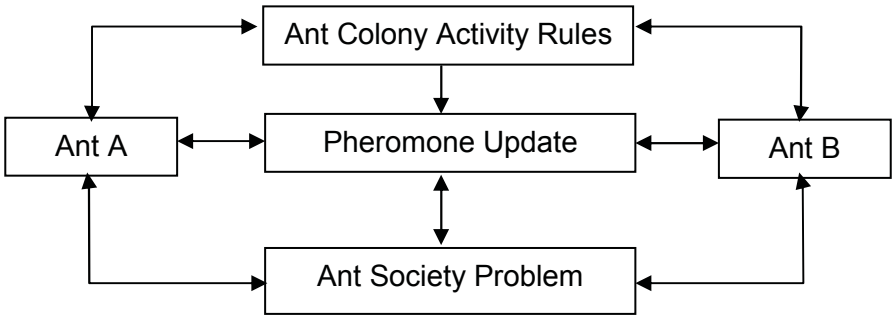


Diagram 1.1 Ant Colony Theory

The above diagram (2.1) presents the ant colony to solve the problem, which transfer into the standard format. According to decision maker of each ant depends on the pheromone values, which has been computing from the algorithms between "exploration" and "exploitation" procedures. In addition, each ant's behavior follows the overall plan to solve the particular problem, and the ant behavior loops to find the optimized answer to solve the problem.

1.2 Previous Research

Investigate the ant colony optimization basic algorithms, and construct the simulator, which simulates the original and nature ant behavior based on the ant colony algorithms. The simulator will take the parameters to initialize it, and random put the ant's nest and food sources on the map under certain limitation. The simulator defines the pheromone value as numeric number, draws a few policies for ant colony society, and restricts the ant's moving action, which can only move from one cell to another neighbor's cell from the map, cannot jump into the cell which is not belong its neighbors.

1.2.1 Double Bridge Lab

This famous lab was designed and operated by Goss S and his colleagues in 1989. They had used double bridge to connect the nest of the ants and the food sources. They ran experiments varying the ratio "r ¼ ll=ls" between the length of the two branches of the double bridge, where "ll" was the length of the longer branch and ls the length of the shorter one (Marco & Thomas, 2004). They had run two major experiments for this lab.

The first experiment was to design the bridge's branch had the same length. The ants moved from their nest to the food source randomly when the experiment started, after a while, Goss S had noticed that there were more ants move using one branch than the another branch. Finally, all the ants used one branch as their path between nest and food sources. The results showed us the following factor. When the experiment started, there was no pheromone on both the branches; hence the ants moving depended on the probability. Therefore, a few more ants would use one branch than another branch, and because of this, one branch would contain more pheromone than another one. At last, according to the ant natural behavior, all the ants would move using the more pheromone branch.

The second experiment was to design the bridges with different lengths, and the longer branch was twice as long as the short one. The result of this experiment was that all the ants used the shorter branch as their path between nest and food source. The ants choosing the short branch were the first to reach the food and to start their return to the nest. However on their way back to nest, they must make a decision between the short and the longer branch,

the higher level of pheromone on the short branch would bias their decision in its favor. Therefore, pheromone starts to accumulate faster on the short branch, which will eventually be used by all the ants because of the autocatalytic process described previously (Marco & Thomas, 2004).

1.2.2 Ant Colony Simulator

The simulator project runs as a windows form, and it contains three forms, which are parameter settings, simulator status and ant colony optimization procedure. The simulator starts with the parameter settings form to allow the user custom the simulator. The simulator status form lists the information about the simulator and updates the status information while the ant colony procedure form is running to trigger the updatable condition. The ant colony optimization procedure form simulates the ant's behavior to find the food sources.

The simulator has a two-dimension map, which is windows form in this research project. This map consist a list map cells. Map cell will have four different statuses: 1) ant current at the cell; 2) the map cell is food source; 3) the map cell is nest; 4) the map cell without above objects. The map cell has the "position" property, which specifies the X and Y coordinate locate in the map. In addition, the map cell contains a very important property, which is pheromone value. The ants choose the cell to enter which depends on this pheromone value. The ant is another important object in this project. The ant has "unique ID" to identify itself for simulator. The ant has the "position" property, which represents the ant currently locate the position in the map, and also has the list of the "position" as its property. This list of the position contains all the positions which the ant has passed, and this is simulating the ant's working memory. The ant should not enter the cell with which position matches the position in the ant's position list, and this is one of the basic ant activity rules. However, the working memory needs to be cleaned after the ant changing its target. For example, if an ant travel along the map, then arrive the food source, it picks up the food, change its target back to home, the ant's position list need to be reset.

There are a few activity rules that defined how the ants move. These are listed below:

1. The ant will always attempt to enter the cell with the highest pheromone value. If the pheromone values of all the ant's neighboring cells are the same, then the ant will randomly choose a cell to enter. If rules 2 or 3 (below) prevent the ant entering the cell

with the highest pheromone value, the ant will attempt to enter the cell with the next highest value.

2. The ant will not enter a cell if the cell's position exists in the ant's working memory (position list). This is to stop the ant going backwards.
3. The ant will not enter a cell if the cell contains another ant. However, if the ant's neighboring cells all contain an ant, then the ant will still enter one of them.

1.2.3 Problems Detected From Previous Research

1.2.3.1 Ant Movement Loop Cycle Problem

Ant movement loop cycle problem was the one of the biggest major problem which had been detected during the implementation. The problem was discovered by the stage for scoping to code the ant colony algorithms. By that time, this problem became the major problem to struggle with the basic theory of the ant colony algorithms. The problem had discovered by the observation result that more than one artificial ant follow the previous ant to trace the highest pheromone values, they tends to loop in the certain routine and never enter into other different routines. According to the basic theory of the ant colony algorithms, the artificial ant should always pick up the path with highest pheromone value to enter, and this theory caused the ant movement loop cycle problem. Assume the below simple example: If there are 10 ants for the simulator map, the nest locate in the position of (500,500) and the pre-defined length is 5. The first artificial ant randomly choose the cell to enter, for this sample, it choose the position of (505,500) to enter. There is a small question here, which is "should the first artificial ant update the pheromone value after enters the map cell". If the first artificial ant updates the pheromone value after entering the map cell, then the following ants will all choose this map cell to enter because of this map cell contains the pheromone value 1, and the rest of the map cells are all having the pheromone value 0, and under the ant colony algorithms basic theory, the rest of ants restrict to choose the map cell with the pheromone value 1 to enter at this case. Therefore, any artificial ant will only update the pheromone value when it leaves the map cell. The second ant choose the position of (500,505) to enter, third ant choose the position of (495,500) to enter, the rest of the ant choose different position to enter, however, there are 10 ants, and the total number of the neighbor map cells of the nest map cell is 8, hence, there should be at least two neighbor map cells contains the pheromone value of 2. In this case, assume the position of (505,500) and (495,500) contains the pheromone value of 2 and one of the ants at the position of the

(505,500) chooses the next neighbor cell with the position of the (510,500) to enter. After that, the artificial ant will choose the next neighbor cell with the position of (505,500) from current map cell with the position of (510,500) to enter because the entering map cell contains the highest pheromone value among the all the other neighbor cells. Because the artificial ant has already visited the map cell with the position of (505,500), also when it leaves this map cell again, it will update the pheromone value which restricted by the basic theory of the ant colony algorithms, and it will continue to choose the map cell with the position of (510,500) to enter. Hence, there appears an ant movement loop cycle. At least two artificial ants in above sample will always choose those two map cells to enter, and never choose the different routine to enter.

This problem struggle with the basic theory of the ant colony algorithms, however, in the real world, the ant will always go forward to the path, will not go backward to the path unless there appears the obstacle on the moving path. This shows us one thing, which the real ant should have some memory to encourage them to not go back the way which they have already passed. Therefore, the artificial ant should have some memory as well.

There are two additional logic have been developed to solve this problem. Enhance the memory for the artificial ant is the first additional logic. The artificial ant contains one more property to remember all of map cells which previous entered. However, there are two small issues raised for this new property. According to the two issues, only the short term memory can be developed for the artificial ants. In addition, the short term memory capacity is also important, if the short term capacity is too small, which will increase the possibility to come cross the ant achievement problem, and if the short term capacity is too big, which will increase the possibility to come cross the performance problem. Hence, this capacity value should be some kind of the balance value of these two issues, and the final capacity values has been set after a few lab observations. Limited the pheromone value of each map cell is the second addition logic to solve ant movement loop cycle problem. The maximum pheromone value has been set in the simulator parameter, which restricts the pheromone value to be increased unlimited. If the pheromone's value can be increased unlimited, then the artificial ant will always choose the map cells with highest pheromone values to enter, and this tends to be the big routine of the ant movement loop cycle. Hence, the limitation of the pheromone will force this do not happen. If the artificial ant enters the map cell which the neighbor map cells all have the maximum pheromone values, then the artificial ant will randomly choose the map cell to enter.

In short, the ant movement loop cycle problem is the biggest challenge for this research topic, it takes quite a while to solve this problem and it requires lots of time to practice the theory of the ant colony algorithms. In addition, this problem leads to another major problem which is ant working memory problem.

1.2.3.2 Ant Working Memory Problem

Artificial ant will simulate the real ant, which has the working memory. Working memory is very important which solve the ant movement loop cycle problem. There are two issues raised for the ant working memory problem. The first issue is the performance problem. Assume this sample; if the simulator contains 50 artificial ants, and the map contains 1000 map cells, then all of ants travel 500 map cells. Each of the artificial ants contains 500 map cell objects in their memory property, and when the artificial ant needs to determine next map cell to enter, it will look up memory to ignore the candidate map cell exists in the memory, and this action apply for all other artificial ants, which means the single ant movement will loop at least 500 times for calculation, and the whole ants list move one cycle will calculate 25000 times, and the calculation time will potential increase quite lot during the ants moving. The second issue is the ant achievement problem. If one of the artificial ants almost travels the whole map cells without finding the food source cell, then it may enter the map cell which all the neighbor map cells has been already visited. In this situation, the artificial ant cannot move to any other map cells because of the memory restrict the moving action, and that artificial ant cannot achieve the task to find the food source.

Therefore, the working memory has the problem to balance the performance and achievement. This problem caused quite a lot troubles during this research, and the updating working memory process has been re-designed three times to solve the issues. The fully memory process structure has been designed for artificial ants at the first time for designing the working memory, and which seems works fine at early stage, but then the artificial ants move much more slower when it visited lots of the map cells, and in fact this causes the performance problem. For the second time, the short term memory has been designed which only to memorized a few steps for the map cells, and this logic works fine during the time of testing. However, during the time of observation, the major issue, which is the problem of the ant movement loop cycle, has been detected. The artificial ants seem to travel into the big loop cycle routine. After that, the capacity of the artificial ants' working memory has been modified into the fixed value for many times to find out the best the value to balance the performance and achievement. In addition, the working memory for the

artificial ant will do the behavior as first in first out. For example: the working memory capacity has the fixed value, and first map cell (“cell A”) save in the working memory, the map cells continue to save into the working memory until the capacity is full. If a new map cell needs to be saved into the memory when the capacity is full, the working memory will remove the map cell of the “cell A”, and save the new map cell.

In short, the working memory problem for artificial ant is the major problem for this ant colony simulator project, fixing this problem will also fix ant movement loop cycle problem.

1.3 Improve ant colony algorithms

According to the previous discussed problem, lots of researchers have done analysis work, and make a few improved algorithms. Dorigo.M had researched on the “Ant – Q” system based on the principle ant colony algorithms. This algorithm will always update the pheromone value on the shortest path of iteration, and only the path, which has the highest pheromone value, will has the high possibility to be chosen. This algorithm fully used ant learning policy, and maximized the feedback policy for the optimized path which has the highest pheromone value. German researchers “Stutzle T” & “Hoos H” advanced another theory to improved ant colony algorithms, MAX-MIN ant system (MMAS), MMAS defined the highest allowance pheromone value and the lowest allowance pheromone value, and use trail smoothing mechanism in the algorithm.

There are two major types of improved ant colony algorithms:

a) The improvement of the ant colony algorithms in the discrete space optimization. Using pre-defined search algorithms to find out the group of the points (steps), which are close to solve the problem, can be chosen as increasing the possibility, and finally to work out the optimization algorithms.

b) The improvement of the ant colony algorithms in the continuous space optimization based on the distribution function of the amount of pheromone, and working out the optimization function value. The distribution functions include the information, which all the ants travel through all the nodes as well as pheromone values.

Chapter 2 TSP (Travel Salesman Problem)

2.1 Introduction

The Travelling Salesman Problem (TSP) was first formulated as a mathematical problem in 1930 and is one of the most intensively studied problems in optimization, and it is a problem in combinatorial optimization studied in operations research and theoretical computer science (Travelling salesman problem, 2009).

TSP can be modeled as an undirected weighted graph in the computing field. (Dorigo & Gambardella, Ant colonies for the traveling salesman problem, 1996) This kind of undirected weighted graph includes cities and paths, which consider cities as the graph's vertices and paths as the graph's edges. In addition, the path's distance between two cities is the edge's length.

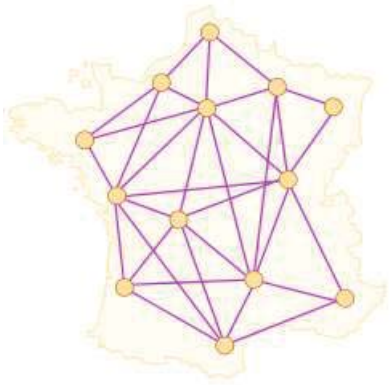


Diagram 2.1 Travel Salesman Problem Graph

Travel salesman problem is a famous benchmark for many optimization methods, which includes ant colony algorithms, and the ant colony optimization to solve the travel salesman problem had been first introduced by Dorigo & Gambardella on 1996. Therefor, lots of the improved optimiszation for ant colony algorithms to slove the travel salesmane problem.

The following chapter, which discusses ant colony algorithms in the discrete space optimizations has listed a few algorithms. These algorithms work out the travel salesman problem to collect the observation data.

2.2 Overview

The goal of this problem is that to find the shortest “Hamiltonian cycle” from the directly graphic. (Travelling salesman problem, 2009) This problem can be resolved by ant colony system. In the ant colony system, the artificial ant, which is an agent, moves from city to city on the graph of the TSP. The artificial ant chooses the city to move depends on the probability, which should be calculated based on the cities that are connected by edges with a lot of pheromone. (Thomas & Silva) In fact, the artificial ants find a shortest closed tour which visits all the cities in a given set for the graph. In the ant colony system, at first, numbers of ants have been initialized on randomly selected cities. These ants move to the new cities, which linked to current city by the edges, and modify the pheromone values on the used edges. After all the ants travel around the tour/graph, each of them made the shortest tour modifies the edges belonging to its tour by adding an amount of pheromone trail that is inversely proportional to the tour length (Dorigo & Gambardella, 1996). There are two terms describe in this ant colony system, which are local trail updating, that updating the pheromone values on the used edges, and global trail updating, that adding an amount of pheromone trail that is inversely proportional to the tour length.

The below list the basic algorithms steps for ant colony to solve the travel salesman problem from Duan’s MatLab application (Duan, 2005):

1. Initialized the parameters, which includes setting the default value time (t=0), loop cycle numbers(N = 0), setting the max loop cycle number, setting the number of ants (A) in the number of cities(c), setting the pheromone values to be 0 on all the connected edges between cities.
2. Loop cycle numbers (N) = N + 1
3. Ant’s working memory table (w), and make the index number as the ant number (K).
4. Ant’s number(K) = K + 1
5. Each ant according the probability function to forward the next city.
6. Modified the working memory table (w), to add the new city, which just moved into the working memory.
7. If ant number (K) less than the parameter’s ant number (A), which K < A, then go to the step 3. Otherwise go to next step.

8. Update the pheromone values for each edge which ant has used passing from one city to another, which local trail updating.
9. If the loop cycles (N) less than the max loop cycle number, clean the working memory table and go to the step 2. Otherwise finish the loop and output the compute result.

2.3 Algorithms

The key to the application of ACS to a new problem is to identify an appropriate representation for the problem (to be represented as a graph searched by many artificial ants), and an appropriate heuristic that defines the distance between any two nodes of the graph. Then the probabilistic interaction among the artificial ants mediated by the pheromone trail deposited on the graph edges will generate good, and often optimal, problem solutions (Dorigo & Gambardella, Ant colonies for the traveling salesman problem, 1996).

There are many ways in which ACS can be improved so that the number of tours needed to reach a comparable performance level can diminish, making its application to larger problem instances feasible. First, a local optimization heuristic like 2-opt, 3-opt or Lin-Kernighan (Lin and Kernighan, 1973) can be embedded in the ACS algorithm (this is a standard approach to improve efficiency of general purpose algorithms like EC, SA, NNs, as discussed in (Johnson and McGeoch, in press)). In the experiments presented in this article, local optimization was just used to improve on the best results produced by the various algorithms. On the contrary, each ant could be taken to its local optimum before global trail updating is performed. Second, the algorithm is amenable to efficient parallelization, which could greatly improve the performance for finding good solutions, especially for high-dimensional problems. The most immediate parallelization of ACS can be achieved by distributing ants on different processors: the same TSP is then solved on each processor by a smaller number of ants, and the best tour found is exchanged asynchronously among processors. A preliminary implementation (Bolondi and Bondanza, 1993) of a similar scheme (Dorigo, Maniezzo and Colorni, 1996) on a net of transputers has shown that it can make the complexity of the algorithm largely independent of the number of ants. Third, the method is open to further improvements such as the introduction of specialized families of ants, tighter connections with reinforcement learning methods (Gambardella and Dorigo, 1995; Dorigo and Gambardella, 1996), and the introduction of more specialized heuristic functions to direct the search.

Chapter 3 Ant Colony Algorithms in the discrete space optimizations

3.1 Background and Algorithms

Everything has two sides, which includes the bionic optimization algorithm - “Ant Colony Algorithms”. During the progress for the ant colony algorithms, random selection policy has a few disadvantages: the whole process takes more time, easy to get into the loop cycle problem and leading to the local optimization results. According to these problems for the ant colony algorithms, lots of the researchers spent huge amount of time to improve the ant colony algorithms which can solve those problems.

Among those improved ant colony algorithms which had been implemented by other researchers, two type of the mathematic research study seems more popular. One type of the research is based on the discrete space optimizations for the ant colony algorithms, and another type of the research is based on the continuous space optimizations. Bilchev G A was the first researcher who had introduced the continuous space optimization for ant colony algorithms. The basic theory of his algorithms is that using genetic algorithms to search the entire space at first, then using the ant colony algorithms to optimized the results which had been calculated from the genetic algorithms to get the final optimization result. (Bilchev & Parmee, 1995)

This chapter will discuss the ant colony algorithms in the discrete space optimizations, and introduces the Ant-Q system which had been implemented by the Dorigo M, and investigates the sensational and consciousness ant colony algorithm. From all these, we should obviously know that the discrete space optimizations of the ant colony algorithms should able to avoiding the local optimization results, reduce the time to calculating the optimization results and improve the efficiency for the ant colony algorithms.

3.2 Ant-Q System

Dorigo M had introduced the Ant-Q algorithms after he introduced the basics ant colony algorithms. This new algorithms combine the determinacy choose policy & random choose policy to avoid the algorithms staleness. (Gambardella & Dorigm, 1996) In addition, the algorithms will also dynamic adjust the state transform possibility. (Gambardella & Dorigm, 1996)

The easiest way to introduce the Ant-Q algorithm is to the traveling salesman problem (TSP) or the more general asymmetric traveling salesman problem (ATSP). Let's defined as follows:

TSP

Let V = {v1, ... , vn} be a set of cities, A = {(i ,j) : i,j ∈V} be the edge set, and dij = dji be a cost measure associated with edge (i,j) ∈ A. The TSP is the problem of finding a minimal length closed tour that visits each city. In the case cities v i ∈ V are given by their coordinates (xi, yi) and dij is the Euclidean distance between i and j, then we have a Euclidean TSP. (Gambardella & Dorigo, 1996)

ATSP

If dij ≠ dji for at least some (i,j) then the TSP becomes an ATSP. In the following of this section we will talk generically of ATSP problems, which include TSP as a special case. Let k be an ant whose task is to make a tour: visit all the cities and return to the starting one. Associated to k there is the list Jk(r) of cities still to be visited, where r is the current city (this is equivalent to say that ant k remembers already visited cities). An ant k situated in city r moves to city s using the following rule, called pseudo-random-proportional action choice rule (or state transition rule): (Gambardella & Dorigm, 1996)

$$s = \begin{cases} \arg \max_{u \in J_k(r)} \left\{ [AQ(r,u)]^{\delta} \cdot [HE(r,u)]^{\beta} \right\} & \text{if } q \le q_0 \\ S & \text{otherwise} \end{cases}$$

where:

- AQ(r,s), read Ant-Q-value, is a positive real value associated to arc (r,s) and is the Ant-Q algorithm counterpart of Q-learning Q-values. AQ(r,s)'s are changed at run time and are intended to indicate how useful it is to make move s (i.e., to go to city s) when in state r.
- HE(r,s), is a heuristic function which evaluates the goodness of move s when in city r. For example, in the ATSP HE(r,s) is the inverse of the distance between cities r and s.
- Parameters δ and β weigh the relative importance of the learned AQ-values and the heuristic values.
- q is a value chosen randomly with uniform probability in [0, 1], and q0 (0≤q0≤1) is a parameter: the smaller q0 the higher the probability to make a random choice.
- S is a random variable selected according to the distribution given by formula (2) which gives the probability with which an ant in city r chooses the city s to move to. (Gambardella & Dorigo, 1996)

$$p_k(r,s) = \begin{cases} \dfrac{[AQ(r,s)]^{\delta} \cdot [HE(r,s)]^{\beta}}{\sum_{z \in J_k(r)} [AQ(r,z)]^{\delta} \cdot [HE(r,z)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases}$$

As we said, the goal of Ant-Q is to learn AQ-values such that they can favour, in probability, the discovery of good ATSP solutions. AQ-values are learned by the following rule1: (Gambardella & Dorigo, 1996)

$$AQ(r,s) \leftarrow (1-\alpha)\cdot AQ(r,s) + \alpha \cdot \left(\Delta AQ(r,s) + \gamma \cdot \underset{z \in J(s)}{Max}\ AQ(s,z) \right)$$

The update term is composed of a reinforcement term and of the discounted evaluation of the next state. In general, the reinforcement ΔAQ can be local (immediate) or global (delayed). In the current version of Ant-Q local reinforcement is always zero, while global reinforcement, which is given after all the ants have finished their tour, is computed by the following formula: (Gambardella & Dorigo, 1996)

$$\Delta AQ(r,s) = \begin{cases} \dfrac{W}{L_{Best}} & \text{if } (r,s) \in \text{ tour done by the best agent} \\ 0 & \text{otherwise} \end{cases}$$

where L-Best is the length of the tour done by the best ant, that is the ant which did the shortest tour in the current iteration, and W is a parameter. (Gambardella & Dorigo, 1996)

Ant-Q Algorithm (Gambardella & Dorigo, 1996)

/* Initialization phase */

Set an initial value for AQ-values

/* Main algorithm */

Loop /* This loop is an iteration of the algorithm */

1. /* Initialization of ants data structures */

Choose a starting city for ants

2. /* In this step ants build tours and locally update AQ-values */

Each ant applies the state transition rule (1) to choose the city to go to, updates the set Jk and applies formula (3) to locally update AQ-values (in formula (3) ΔAQ(r,s)=0)

3. /* In this step ants globally update AQ-values */

The edges belonging to the tour done by the best ant are updated using formula (3) where ΔAQ(r,s) is given by formula (4)

Until (End_condition = True)

According the above algorithms and formulas have been introduced by "Dorigo" & "Gambardella", Dr "Duan" have shown some improved algorithms based on their research. His improved algorithms changed the pheromone volatile element (p value) to increase the algorithms' general situation.

Below shows the pseudo code for his improved algorithms: (Duan, 2005)

Begin
Initialization
While (Not satisfied the algorithm stop condition)
{For (α_k = 1; α_k < m; α_k ++)
    Put the with the total of number m on the start position randomly
    For (i= 1; i < n; i++)
        {For(α_k = 1; α_k < m-1; α_k ++)
            The ant α_k choose next city depends on the possibility p_{ij}
        }
    Get the optimised result
    Set this result to the ant m
    If the optimised result = N the previous optimised result
    Update the pheromone volatile element p value according the formula
    Update the value of $\Delta\tau_{ik}$, which based on the "Dorigo" formula.
    Update the value of τ_{ik}
    Set the $\Delta\tau_{ik}$ = 0
    $N_c = N_c + 1$
}
Get the optimised result.
End

Below tables list the lab data to show the different between the basic ant colony algorithms and Ant – Q algorithms. This lab had used Oliver30TSP of the TSPLIB, and has been changed into different parameters.

α	β	ρ	Shortest Path	Iteration
1	4	0.5	395.2774	327
1	4	0.9	398.7039	341
2	2	0.5	390.5832	339
2	2	0.9	394.2361	308
4	1	0.5	316.6900	345
4	1	0.9	317.0540	322

Diagram 3.1 basic ant colony algorithms under different parameters
(Duan, 2005)

α	β	ρ_{min}	Shortest Path	Iteration
1	4	0.001	310.8758	162
1	4	0.5	322.5945	147
1	4	0.1	308.1685	150
2	2	0.001	271.1854	152
2	2	0.5	303.8851	143
4	1	0.1	287.8268	119
4	1	0.001	267.7795	168
4	1	0.1	290.4864	137

Diagram 3.2 Ant -Q algorithms under different parameters

(Duan, 2005)

According to the above lab data, it shows that the worst result from Ant-Q algorithms is still be optimised result for the basic ant colony algorithms, in addition the iteration for the Ant-Q algorithms are almost less than 200, for the basic ant colony algorithms are almost more than 300. Below diagrams show the Ant-Q system best and worst iteration process.

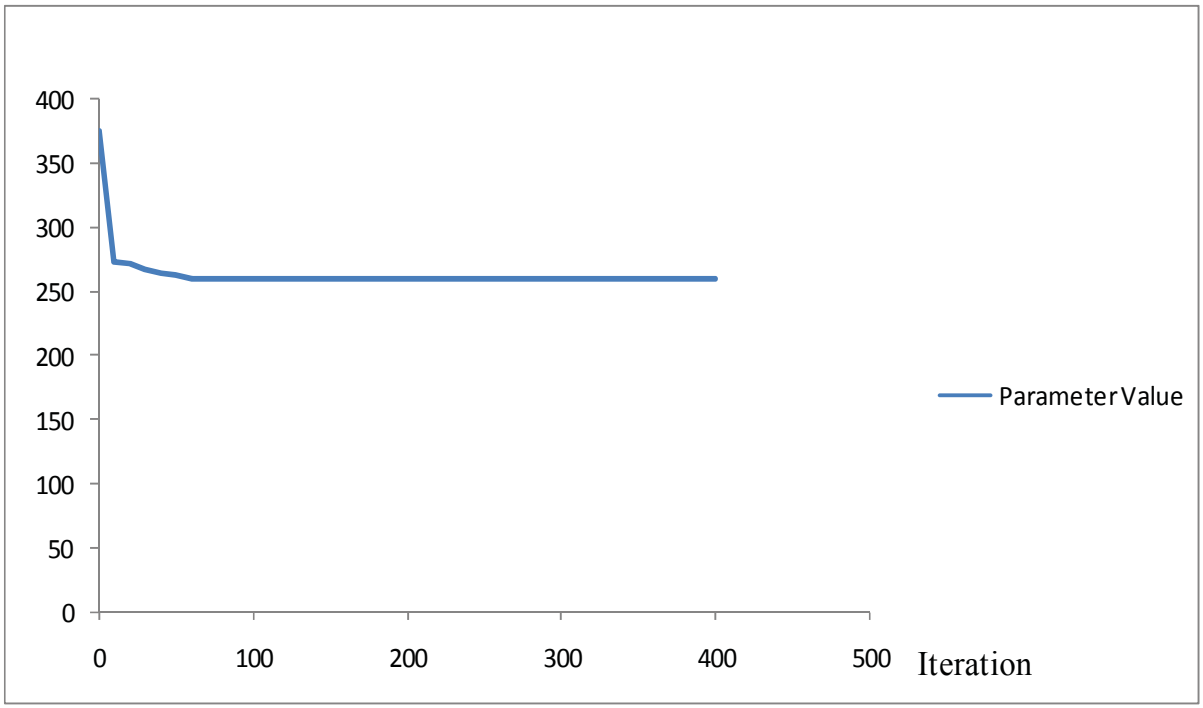


Diagram 3.3 the best iteration of Ant-Q algorithm

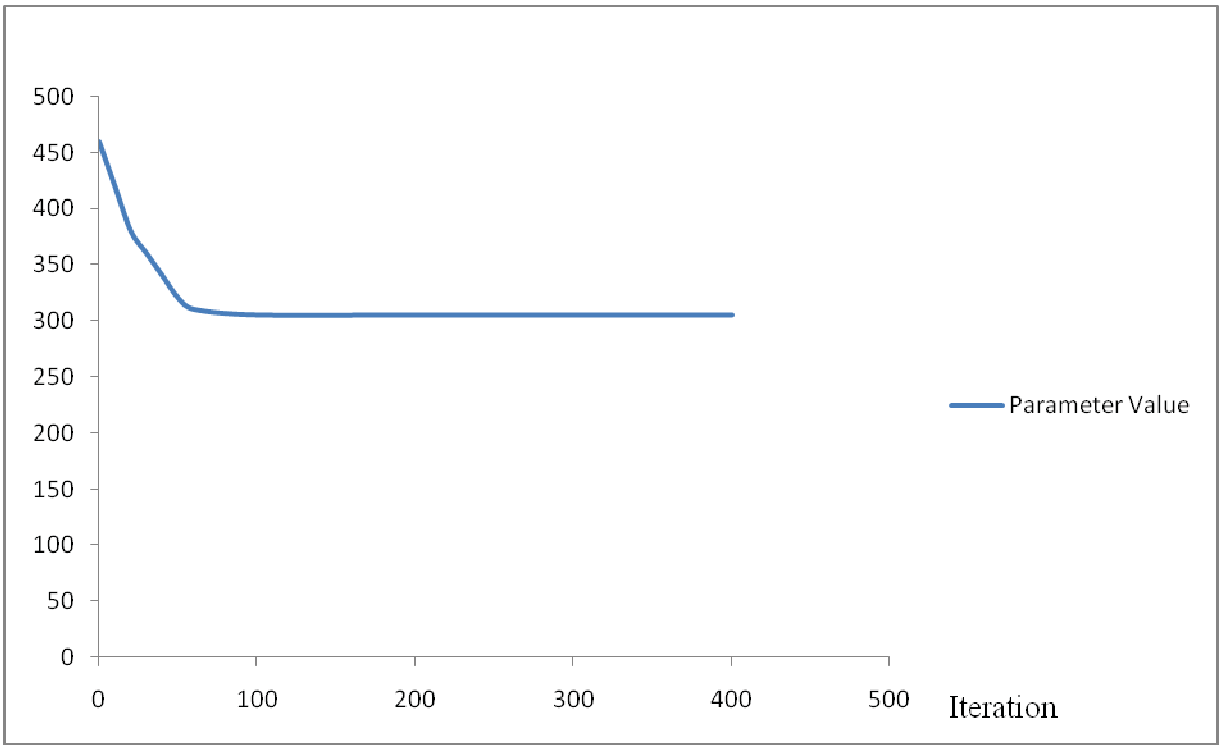


Diagram 3.4 the worst iteration of Ant-Q algorithm

3.2 Sensational and consciousness ant colony algorithm

This algorithm simulates the ant sensational and consciousness behaviour, and the ant makes decision to choose the path under reveals consciousness and subconscious affect each other. According to this, the ant chooses the path depends on its own experience on the first stage, after that the ant step by step to changing depends on the pheromone values and adjust the pheromone values on the passed path.

From the biological research, ants have powerful sensational and consciousness for stimulations from the environment. Individual ant has limit ability and intelligence; however the ant colony has enough ability to achieve multipliable and complicate tasks, they can coordinate with each other under sensational and consciousness control to accomplish the task. Therefore, ant colony algorithms simulate this kind of consciousness behaviour, let the ant make decision to choose the path depends on the sensational and consciousness in the discrete space, this will increase the speed of the convergence of the algorithms and keep the vary optimization results.

Overall, we call this kind of algorithms as sensational and consciousness algorithms (SCA). (Qin, Chen, & Chen, 2003, 15(10)) Base on the consciousness law, we can divide ant search process into 3 stages under SCA. (Duan, 2005)

3.2.1 Ant Search First Stage

Psychologist pointed out that nerve system produced the sensational and consciousness while the object things affect the nerve system. (Chen & Ma, 1996) The nerve organization, which produces the sensational and consciousness, calls neural analyser, and the feeling, is that the nerve analyser has the feeling ability by the suitable stimulation. Sensational reflects the individual attribute which object act on the organ, and consciousness reflects the overall attributes, Sensational and consciousness highly work in close cooperation. Absolute sensor threshold (AST) defined the minimum stimulation caused the sensational in the psychology way.

The ant search first stage, ants move into some routine, and some routine never has ants move into. During this stage, the ant choose the highly probability based on the highest pheromone value on the path as their choosing path policy, hence, the most ants centralized a few shortest paths compared with other paths depends on the probability. According the observation, the overall figure of the length for these paths which the ants has chosen always become bigger, which leads us the part optimization, not the global optimization. In this situation, the other paths, which have no ant move into, may or may not get the global optimization. In lab observations have been collected by Mr Duan (Duan, 2005); show us that most of the case the global optimization will always has these paths which have no ants move into in the search first stage.

In order to avoid losing ants choosing path variety from search at first stage and enlighten us on AST, the ant will consider the path, which has pheromone value less than the AST, as this has no pheromone value. In this way, the ants choose path policy will not affect based on the lower pheromone value, and the ants only consider path has the pheromone value greater than AST as their choosing path policy, which choose the highest pheromone value based on the probability. Therefore, ants can choose most different way on their search first stage; this can get variety results and the global optimization result. In addition, using this way to get the global optimization result can let ant choose the unnecessary way as less as possible and this also can reduce the search time.

3.2.2 Ant Search Middle Stage

After sensational caused by the simulation, if the simulation change, then the sensational will chance as well, however, not all the variation from the simulation change can cause the sensational change. According to the experience and sub-consciousness's affection, if the simulation changes among the certain range, and the special consciousness can still remain un-change. If the simulation change up to certain level, then can feel the difference, hence, this called contrast sensor threshold (CST) which defines the minimum change of the simulation can caused the sensational change. (Duan, 2005)

A German physician "Weber E H" who is considered one of the founders of experimental psychology found one factor during his research on CST in 1846. The factor defines the value of CST change depends on the simulation change, and this changes trend to the constant value under certain range. This can be formulated as below:

$$\frac{\Delta I}{I} = K$$

Where *I* is the intensity of the simulation,

ΔI is the CST

K is the constant value which called Weber Fraction

This is famous weber's law. (Weber's Law and Fechner's Law)

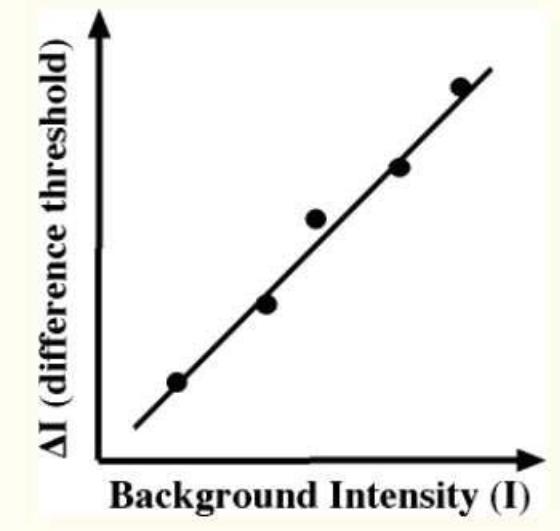


Diagram 3.5 Weber's law

During the progress for the ants searching the food sources, the pheromone value can be increased if the ant carried the food, which just left path. Moreover, the pheromone value can be decreased if the path has pheromone value and there is no ant to choose this path for certain time period, this logic based on the ant natural behaviour. Hence, the pheromone values continuous change of all the paths. According to the CST, the search algorithms terminate the pheromone affection policy, which is the change of pheromone value can directly affect the ants' consideration to choose travel path. In this stage, there exists CST among the pheromone values change under certain range for the ant colony. While the pheromone values increase or decrease below the CST, the ants still follow the regular policy and ignore the change of the pheromone value from the path. If the change of the pheromone values above the CST, then the ants consider the new pheromone value for the path, and this affects the decision for the ant to choose the travel path.

According to the Weber's law, this algorithm for search middle stage can be implemented based on the following formulas.

$$\mathrm{CST} = \tau_{ij}(t) \cdot K$$

Where K is the Weber fraction, and k = 1/50. In general, the Weber fraction for weight sensational is 1/30, the Weber fraction for listening sensational is 1/10, and the Weber fraction for vision sensational is 1/100. (Duan, 2005)

$$\theta_{ij}^{k} = \begin{cases} \dfrac{\alpha_{ij}^{k}}{\beta_{ij}^{k}}, & \text{If} \quad \tau_{ij} \leqslant \mathrm{CST} \\ \tau_{ij}(t)\,\eta_{ij}, & \text{Or} \end{cases}$$

$$\alpha_{ij}^{k} = \frac{\tau_{ij}^{k}(t)}{\sum_{h \in \mathrm{allowed}_k} \tau_{ih}^{k}(t)}$$

Where αij is the value of the attraction for the ant K under its sensational of the path (i, j), and βij is the value of the attraction for the other ant of the path (i, j).

The possibility for the ant k choose the path (i, j) as

$$p_{ij}^{k} = \begin{cases} \dfrac{\theta_{ij}^{k}}{\sum_{h \in \text{allowed}_k} \theta_{ih}^{k}}, & \text{If} \quad j \in \text{allowed}_k \\ 0, & \text{Otherwise} \end{cases}$$

The change of the pheromone value for the path (i, j) less than the CST, the ant choose travel path under its sensational. For example, if the ant k travels the path (i, j) many times, the ant is familiar with the path (i, j), and it is more possible for the ant k to choose this path under its sensational next time. On the other hand, if path (i, j) has more pheromone value which are not belong to the ant k, the ant ignore this path, and it is less possible for the ant k to choose this path under its sensational. This kind of the policy significant prevents the single ant always follow the other ants pheromone value during this stage, and this can decrease the ant's amount in the certain local area which can only leads local optimization result. In addition, this also can solve the ant movement loop problem which has been detected from my previous research.

The change of the pheromone value for the path (i, j) great than the CST, the possibility of the ant k choose the path, which affected by the entire pheromone value, under its sensational, and the ant follows the policy, which choose the path that has the greatest pheromone value, and guarantee the chosen path is the optimal path, and increase the convergence speed for this algorithm.

3.2.3 Ant Search Final Stage

The Weber's law can only apply to the algorithms if the values of simulation under certain figure (IT). Hence, if the pheromone values greater than IT, this should change the policy for ants to choose the travelled path. (Duan, 2005) Therefore, this needs to determine the "IT" value.

According to determine the "IT" value, first thing to do which estimates the maximum pheromone values τ_{max} on all the paths. Follow defines the estimates formula:

$$\tau_{max} = Nc_{max} * m * AST$$

Where Nc_{max} is maximum iteration for the ant to search the travel paths,

m is the total amount of the ants

AST is the value of the absolute sensor threshold

$$IT = h * \tau_{max} = h * Nc_{max} * m * AST$$

Where h is the constant value, can be defined 0.5, 0.66 or 0.75. (Duan, 2005)

During this stage, the pheromone value appears huge on some certain paths, there might be in the situation which the algorithms lead to local optimization in some area. However, there will be less possible that the algorithms for local optimization in some area happened, because the middle search stage has already avoided this happening. The most possible reason for the huge pheromone values on certain paths is because these paths are absolutely supremacy, and the ant colony algorithms tend to be completed. In addition, in this stage, the choosing path policy of the Ant-Q algorithms can be introduced to use in order to increase the speed of the calculation for the algorithms. (Duan, 2005)

3.2.4 Self-Adaptive Update Pheromone Policy

There are always two ways to update the pheromone policy for improved ant colony algorithms. One common way is the algorithm increasing the pheromone value when the ant passed the path during iteration; another way is the algorithm increasing the pheromone value only for the optimized travelled path, and it decreasing the pheromone value for the rest of the paths. Both of these two ways update the pheromone value under certain percentage, and ignore the distribution of the pheromone value.

Sensational and consciousness ant colony algorithm raise the new self-adaptive policy to update the pheromone value. This policy depends on the distribution of the pheromone value, dynamic adjust the pheromone value for all the paths in order to control the distribution for the pheromone value, and the distribution should not be too much centralize or decentralize.

The pheromones do not volatilize for the ant choosing one path to another each cycle under iteration, hence the volatilization should not be considered of the algorithm in some area while updating the pheromone value. According to the algorithms, the ant will choose the path which has the highest pheromone values, if the multiple ants choose the same path at the same certain time, the pheromone values will be increased quite a lot, and this leads to the results which all other ants nearby will choose this path under high probability. Therefore, for this situation, the algorithms should choose the $1/d_{ij}$ as the increasing pheromone value. On the other hand, if there is half of the total ants choose the same path or quarter of the total ants finish iteration after this path because the length of this path is longer than the latest optimised path, then the algorithm need to decrease the pheromone value by 5 / d_{ij} .

These two mechanisms of the algorithms can increase the possibility for the ants to choose rest of the path, and result tends to diversification. This algorithm updates the pheromone value using following formula: (Duan, 2005)

$$\tau_{ij}^{k}(t+1) = (1-\rho)\tau_{ij}^{k}(t) + \psi_k \cdot \Delta\tau_{ij}^{k}(t)$$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \psi_k \cdot \Delta\tau_{ij}^{k}(t)$$

Where ψ_k is the ant K's affection for updating the pheromone value of the path (i, j)

$$\psi_k = s/2 - rank[k] + 1$$

Where s is the total ants which travelled the path (i, j), array rank stores the position index value of all the paths for s in length descend order under current iteration. Rank[k] means the position index value of the ant k.

According to this formula, if the position index value is bigger then the ψ_k become negative value, and the algorithms decrease the pheromone value on the relative path while updating globally. The rank[k] is bigger which means the travelled path is longer and the pheromone value on the path decrease more in order to keeping less pheromone value on the non-optimised path and keeping more pheromone value on the optimised path. On the opposite side, if the value of rank[k] is smaller, the length of travelled path is shorter, then the value of ψ_k becomes bigger, the pheromone value increases more in order to intensify pheromone value on the shortest paths.

In short, this self-adaptive update pheromone policy can dynamic adjust pheromone value globally, and dynamic keep the balance of searching speed and result variation of the algorithms. (Duan, 2005)

3.2.5 Pseudo Code For Algorithms And Structure

Main()

{

a. Initiation

Random create initial the adaptive results of the total ant m, which are the reverse order of the travelled lengths, and record the biggest adaptive result as f_{max} .

Set AST=C * f_{max} where c is constant value and set c = 5

Set there are number of s result for travelling path (i,j), and consider the total length L_1, L_2,…, L_s, therefore the initial pheromone value should be as follow:

$$\tau_{ij}(0) = \sum_{k=1}^{s} \frac{1}{L_k}$$

b. Iteration

While not finish condition do

{

For i = 1 to n do

{

For k = 1 to m do

{

If (the average pheromone value for the relative paths which start from position i less than AST) then

{

Ant choose the next position j at the current position i; using below formula to update the pheromone value.

$$\tau_{ij}^{k}(t+1) = (1-\rho)\tau_{ij}^{k}(t) + \psi_k \cdot \Delta\tau_{ij}^{k}(t)$$

}

Else

{

/*using formulas to calculate the probability for the ant to choose the next position j.*/

If (the average pheromone value for the relative paths which start from position i less than IT) then

{

$$p_{ij}^{k} = \begin{cases} \dfrac{\theta_{ij}^{k}}{\sum_{h \in \text{allowed}_k} \theta_{ih}^{k}}, & \text{If} \quad j \in \text{allowed}_k \\ 0, & \text{Otherwise} \end{cases}$$

}

Else

{

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{h \notin \text{tabu}_k} [\tau_{ih}(t)]^{\alpha}[\eta_{ih}(t)]^{\beta}}, & \text{If} \quad j \in \text{allowed}_k \\ 0, & \text{Otherwise} \end{cases}$$

}

Updating the pheromone value on the path (i, j) depends on the self-adaptive updating policy;

}

}

End for k;

If the total travelled length for the ant m greater than the latest optimised travel length, then terminate the iteration of the ant m to choose next position;

}

End for i;

Using below formula to update the pheromone value globally.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \phi_k \cdot \Delta\tau_{ij}^{k}(t)$$

}

End while

Chapter 4 Simulator with New Algorithms

4.1 Introduction

This simulator project has been modified from my post graduate simulator project with constructing the new ant colony optimization procedure form and implementing new algorithms. This simulator project has been implemented using .net framework 4.0, it contains three windows forms which are parameter settings form, simulator status form and ant colony optimization procedure. The simulator starts with the parameter settings form to allow the user custom the simulator. The simulator status form lists the information about the simulator and updates the status information while the ant colony procedure form is running to trigger the updatable condition. The ant colony optimization procedure form simulates the ant's behavior to find the food sources. The parameter settings form has been adjusted to suit this research topic, add two parameter settings which are "New Algorithm" and "Obstacle Model". The "new algorithm" setting allows the simulator to run the improved ant colony algorithm, and the "obstacle Model" setting allows the simulator to run the obstacle lab observation using the improved ant colony algorithm. In addition, the simulator status form will turn off if the simulator using these two new settings. Moreover, LINQ (Box & Hejlsberg, 2007) has been used quite a lot in my previously research project, however, I have found these may cause the performance issue if LINQ had used quite often. Hence, I have removed a few LINQ statements which are not necessary.

The simulator has a two-dimension map, which is windows form in this research project. The map consist a list map cells. Map cell will have four different states: 1) ant current at the cell; 2) the map cell is food source; 3) the map cell is nest; 4) the map cell without above objects. The map cell has the "position" property, which specifies the X and Y coordinate locate in the map. In addition, the map cell contains two very important properties, which are pheromone value and food pheromone value. The ants choose the cell to enter which depends on these two pheromone values. The ant is another important object in this project. The ant has "unique ID" to identify itself for simulator. The ant has the "position" property as well, which represents the ant currently locates the position in the map, and also has the list of the "position" as its property. This list of the position contains all the positions which the ant has passed, and this is simulating the ant's working memory. My post graduate simulator project defines that the ant should not enter the cell with which position matches the position in the

ant's position list. This research project defines a memory queue for the artificial ant, and the memory queue only can contains maximum fifty previous positions which the ant has passed. The ants split into two groups: ant carrying food group and ant without carrying food group. These two groups of ants' nature behavior are kind of similar. The ant without carrying food will consider the food pheromone on the cell and ignore the pheromone, and it will leave the pheromone on the cell which just passed. The ant carrying food works opposite way; it will consider the pheromone on the cell and leave the food pheromone on the cell which just passed. Both of them are consider one type of pheromone value and leave another type of pheromone on the cell after passing. Ant search first stage, all the artificial ants are placed into their nest. These ants without carrying the food choose their travel path to find the food source depends on the possibility and at this stage there are on food pheromone around the cells. Ant search second stage, carrying food ant group choose their travel path back to nest depends on the highest pheromone value or random selection.

There is only one ant's nest can be initialized for the simulator, a few food sources can be initialized depends on the parameters' settings, and by default there is only one food source can be initialized for the simulator. Because the position of the food sources and nest are randomly generated, there may be the food sources position locates too close to the ant nest position, which will affect the results of the research observation. Hence, there is a limitation for randomly generation for the position of nest and food sources. This limitation restricts the minimum distance between any position of the food sources and position nest. The minimum distance can be initialized by the simulator parameters. The position of the nest will be generated before the food sources, and then the simulator generates the food sources' positions. For every position of the food source, which does not satisfied with the minimum distance, needs to be re-generated. In addition, the position of the nest and food sources should not locate nearby the edge of the simulator form because of this may affect the obstacle model and lab observation's result.

The basic work flow for the new simulator project describe as follow:

1. Call construct method to initialize a new form object for the simulator, and sets the simulator parameter for this new form object. There a few steps involve creating simulator form object.

    - Initialize the form component, which create GUI (Graphical User Interface).
    - Create map, map cell, ant, nest object, and food source objects etc.

2. Loop through all ant objects, ant determine the cell to enter depends on the algorithms, ant to move into the cell and paint to the map.

   - If the ant enters the cell which is food source, then the ant change to the carrying food group, and paint into dark green colour.

   - If the ant enters the cell which is nest, then the ant change to the without carrying food group, and paint into black colour.

   - Continues the loop logic until manual termination.

4.2 Algorithm Design

There is a new algorithm has been implemented for this research project. This algorithm still simulates the ant's nature behaviour, however, a few artificial features has been added to the algorithm. Below list these features:

1. Loop through all the ants instead of loop through all the map cells. The algorithm for my post graduate project loops through all the map cells, and determine the cells for ant's entering.

2. Two type pheromone values are attached for map cell.

3. Artificial ants has been divided into different groups depends carrying the food.

4. Artificial ants determine the cell to enter, and paint different colour itself.

5. Ants' communication policy.

All above features has been designed by the thinking of the ant's communication policy based on the research of the ant's sensational and consciousness. The ant should have ability to communicate with other nearby ants to exchange the information such like food

sources etc. According to the research project and communication policy, the ants should be divided into two groups (ant carrying food group and ant without carrying food group).
The ant A will look all the neighbour cells which contain other ants, basically, the ant A will ignore all other ants which are the same group as the ant A, and enter the cell which contains the ant that is from different ant group.

The map cell contains two integer values, which are ant carry food and ant without carrying food. These two values describe how many ants are in the current map cell, and identify into different groups. There is a property for the map cell to show that the cell whether has ant with food, this property is read only and is a boolean value, and can only be calculated by comparing the number of ants carrying food and the number of ants without carrying food at the time. This property is very important for ant communication policy, because the map cell can contains more than one ant at the same time, the ants can be different groups, and the algorithm need to clearly know the cell contains ant carrying food or not. If the number of ant carrying food greater than the ant without carrying food, the algorithm consider this map cell contains the ant carrying food, which can be used for ant communication policy.

The algorithm for individual ant without carrying food determining movement has been designed as follow structure:

1. Get all the neighbour cells and sort the food pheromone value on the cells by descend order.

2. Determine food source in the neighbour cells, if neighbour cell contains the food source, the ant straight enters this cell.
   a) Clear ant's memory, which delete all the previous passed cells in its memory.
   b) Change the ant group which from without carrying food group to carrying food group.

3. If all the neighbour cells do not contain the food source, retrieve the cells have highest food pheromone value and the cells have ant with food. If there are more than one cells can be entered, then the ant random determine the cell to enter.

4. If the highest pheromone value from the neighbour cells is equal to zero, then the ant retrieve the neighbour cells that have ant with food property. If there is no such cell found, then the ant random enter all its neighbour cells. If there exists this kind of cells, then the ant random enter these cells if more than one have been found.

5. The ant makes the movement, and set its current cell as the cell which just entered. Add the previous map cell into its memory.

The algorithm for the ant carrying food determining its movement has been designed opposite way compare with above algorithm, and the algorithm consider the pheromone value instead of the food pheromone value.

There are two import methods has been designed for this algorithm, which are ant entering map cell and ant leaving map cell. These two methods have been implemented into map cell object rather than ant object.

- Ant entering map cell method

  This method will add the ant into the ant list for the cell which entering. If the ant is from carrying food group, then the property of the number of ant carrying food need to be added as well. If the ant is from without carrying food group, the property of the number of ant without carrying food needs to be added.

- Ant leaving the map cell method

  This method will remove the ant from the ant list for the cell. If the ant is from carrying food group, then the property of the number of the ant carrying food need to be decreased. If the ant is from without carrying food group, the property of the number of ant without carrying food needs to be decreased.

The above two methods are important; because these methods change the two integer values for the map cell, which are ant carry food and ant without carrying food. These two values are the Meta data for ant communication policy which has been described in the previously paragraphs.

The two types of the pheromone are the most important sources for this algorithm, because the artificial ant will make decision to enter cell depends on these values. For this research topic, the pheromone values of each map cells should be updated during certain period if the map cell contains the pheromone values. Before the artificial ant leaves the map cell, it increases one type of the pheromone value for that map cell depends on the ant group. There is no limitation to increase the pheromone values, which has been modified from the post graduate project. Decreasing the pheromone values is very complicate in the real world; the pheromone will disappear if it is raining. In addition, the pheromone value decreasing speed depends on the natural environment. For example, if the temperature of the weather

is very high and the land is very dry, then the pheromone will volatile quickly, and pheromone values accelerate to disappear, and on the other hand, if the temperature of the weather is standard and land is not too dry, then the pheromone value will remain on land longer. Because of all these complicate condition, it is very difficult for simulating this nature environment randomly of this research topic. Hence, the simulator needs to simply the method, which decreases pheromone values using certain value. Therefore, the pheromone values of the map cell will be decreased for every loop cycle of the ant's movement until the pheromone value sets to be zero.

4.3 New Simulator Implementation

The new simulator has been implemented by .net framework 4.0. There are three C# projects that have been developed for this research project. One project is console application, which includes three form classes: parameter, simulator and simulator status. Another project is the class library project, which is the entity project. The console application references the class library project, and the class library project contains lots of the classes structure such as: ant class, map class, map cell class, nest class and so on. The last project is the algorithms project, and the ant colony algorithm has been developed in this project.

4.3.1 Class diagram structures

The .net object-oriented architecture gives this research topic of the ant colony algorithms simulator a good clear view of the whole simulator work flow for all the relative objects. This also reduced the coding complexity and reduced the total time spent on the development work.

Below is the list of diagrams for these three .net projects structure:

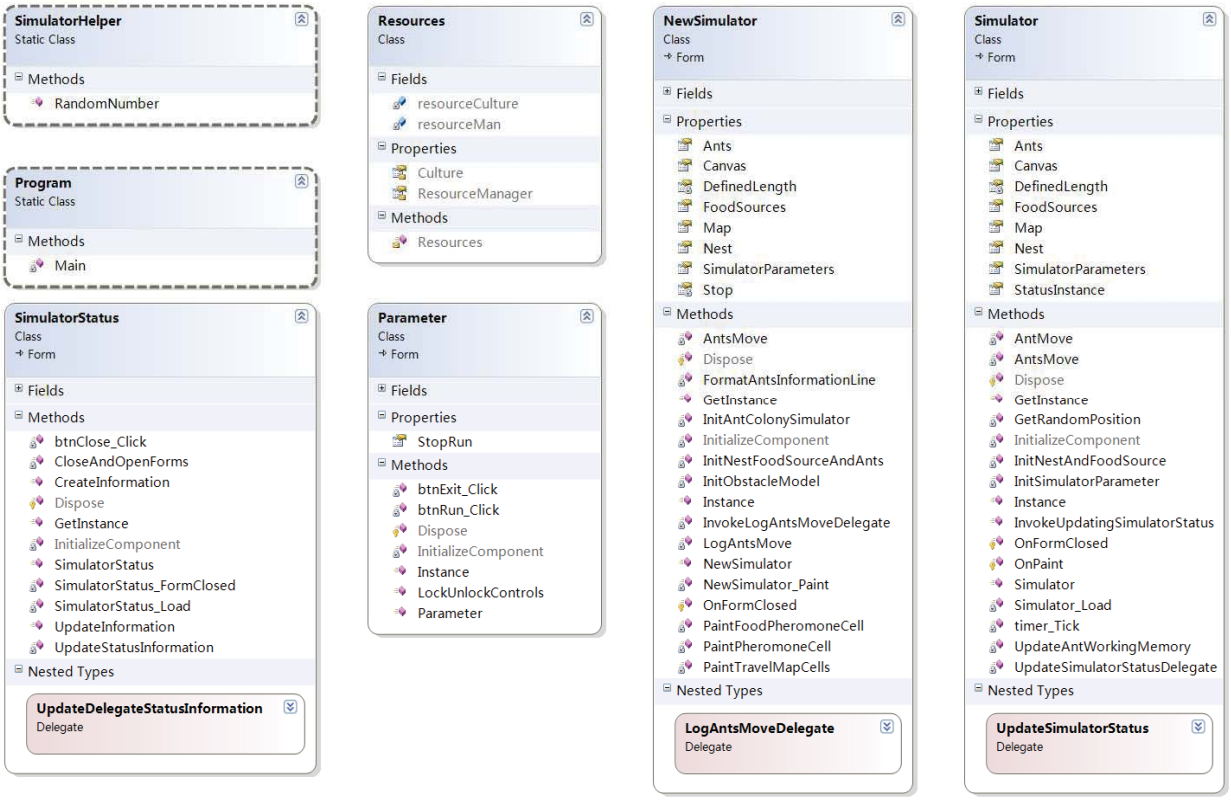


Diagram 4.1 Classes structure for the console application project

The above diagram show the class structure for console application project. Please note that the “Simulator” class is used for my research project in the post-graduate diploma, and the “NewSimulator” class is used for this research project. The “NewSimulator” class has implemented to correspond with ant’s communication policy that been designed as well as all the other new features for the ant colony algorithms. The “LogAntsMoveDelegate” delegate type has implemented to handle logging all the movement position for each ant. Delegate types are derived from the delegate class in the .NET Framework. Delegate types are sealed—they cannot be derived from— and it is not possible to derive custom classes from Delegate. Because the instantiated delegate is an object, it can be passed as a parameter, or assigned to a property. This allows a method to accept a delegate as a parameter, and call the delegate at some later time. This is known as an asynchronous callback, and is a common method of notifying a caller when a long process has completed. When a delegate is used in this fashion, the code using the delegate does not need any knowledge of the implementation of the method being used. (MSDN)

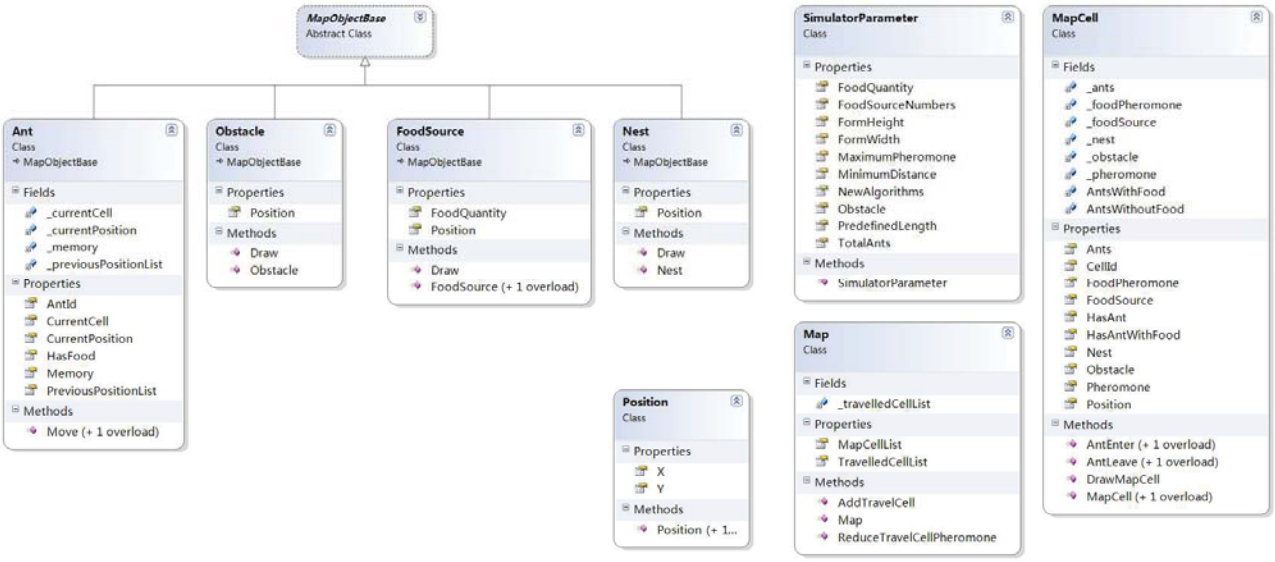


Diagram 4.2 Classes Structure for entity project

The above diagram lists the fundamental entity classes for the ant colony simulator. The “ant”, “Obstacle”, “FoodSource” and “Nest” classes are all inherited from the base abstract map object class. The base abstract class contains the virtual “draw” method, which allows child objects to draw different colours on the map cell depending on its own method. In addition, this abstract class also contains “DefinedLength” properties which allow the simulator to set the unified length of all the objects represent on the two-dimension map. “Position”, “MapCell” and “Map” classes are all base class for this research project. The map class contains “TravelledCellList” property. This property stores all the map cells that have been travelled by the ants, which mean those map cells have either food pheromone value or pheromone value. This property is helpful for the algorithm to decrease the pheromone values for iteration rather than calculating the cell lists contains pheromone value on the run time. When the ant object enters the cell, the ant object calls “AddTravelCell” method from the map object, this method will add the map cell into the “TravelledCellList” property if this map cell does not exist in the property. In addition, the map cell will be removed from the “TravelledCellList” property if the map cell has no food pheromone or pheromone value during iteration.

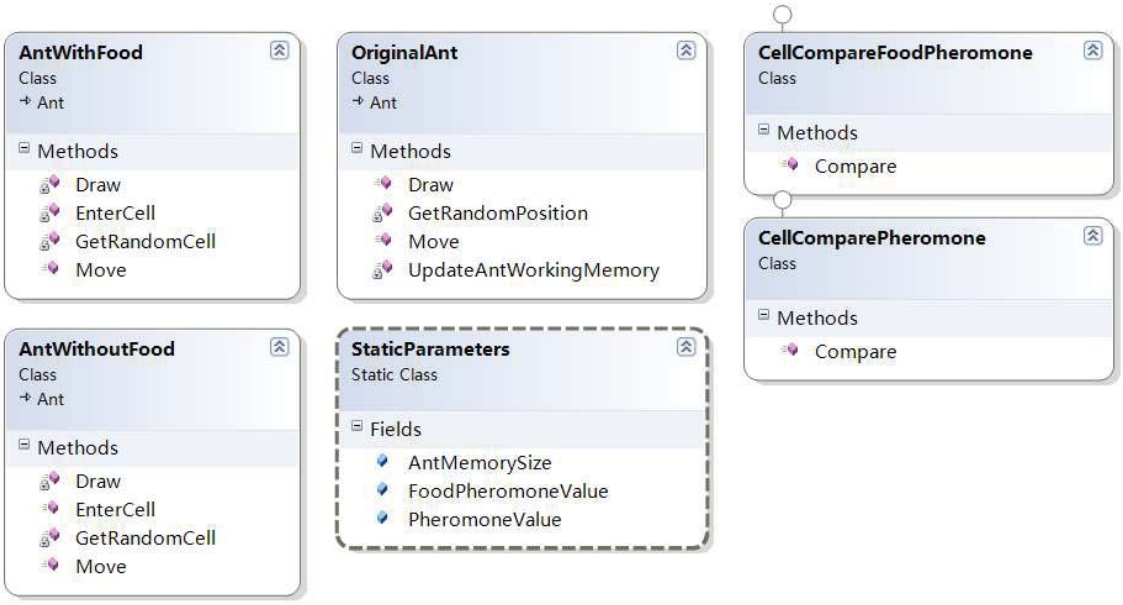


Diagram 4.3 Class Structures for algorithms project

The above diagram shows the major algorithms which have been implemented for this research project. The “OriginalAnt” class was used for the research project in my post-graduate diploma. The overall structure for the “Simulator” solution has been modified. There was no algorithms project for my post graduate research solution, and the ant colony algorithms were implemented within the console application. However, it seems that is not quite easy to attach into different algorithm for the simulator. Hence, I have modified the structure which allows to set up the “simulator” form and to attach different ant colony algorithms easily. “AntWithFood” and “AntWithoutFood” are the class which work out the algorithms for this research project. These two classes basically represent two ant groups which have described in previous sections. In addition, all these three “ants” classes all inherits from the “Ant” class which from entity project. The “StaticParameters” class contains three fields: Ant memory size, food pheromone value and pheromone value. These are fixed value which can be adjusted manually. Ant memory size is used for algorithms to restrict the memory size for the ant. Two different pheromone values are values can be straight increased or decreased on the map cell. The reason to put these static parameters is that I can easy to adjust these values to observe the algorithms behaviour on the simulator.

Another two compare classes are implemented the “ICompare” interface from .net metadata, which allows algorithms to sort pheromone values quickly.

4.3.2 New features for simulator

There are three new major features that have been implemented for the ant colony simulator, which are infinite map boundary, log ant movement data and obstacle model. Obstacle model is used to represent the artificial ant choose the ideal routine to pass the obstacle on the map under ant colony algorithms.

4.3.2.1 Infinite map boundary

From my post graduate project, the map object contains four edges, which the ant cannot pass through as shown in the diagram below.

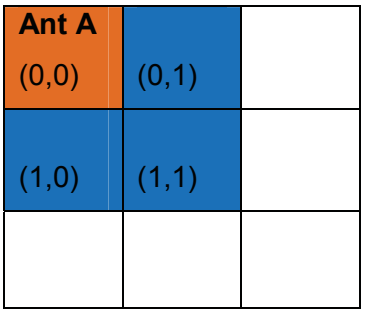


Diagram 4.4 Boundary of the Map

The orange map cell represents the “Ant A” locates on and the blue map cells represent the neighbour cells for the “Ant A”.

The ant A can only make decision to go through depends on its three neighbour cells. This restricts the ant movement, and the real ant should find the food source or nest in the infinite space. According to this, the map object should not restrict the ant movement under its boundary, and this kind of two-dimensional infinite map is quite regular to use for simulator.

The following diagrams describe the basic properties for the infinite map.

Ant A (0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
(2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

Diagram 4.5 Boundary of the infinite map -1

The orange map cell represents the “Ant A” locates on and the blue map cells represent the neighbour cells for the “Ant A”.

There are eight neighbour cells can be selected by the Ant A from above diagram which locates in the position of (0,1), (1,0), (1,1), (0,3), (1,3), (3,0), (3,1) and (3,3), and the Ant A currently position which locates on the top-left corner map. The ants can easy pass the map boundary for the other three corners under this criterion.

(0,0)	(0,1)	(0,2)	(0,3)
(1,0)	(1,1)	(1,2)	(1,3)
Ant A (2,0)	(2,1)	(2,2)	(2,3)
(3,0)	(3,1)	(3,2)	(3,3)

Diagram 4.5 Boundary of the infinite map – 2

The orange map cell represents the “Ant A” locates on and the blue map cells represent the neighbour cells for the “Ant A”.

The above diagram shows the Ant A located on the left edge of the map. The surrounding cells are located in the positions (1,3), (1,0), (1,1), (2,3), (2,1), (3,3), (3,0) and (3,1). The ants choose the eight neighbour cells using this way for all the edges on the map.

4.3.2.2 Log ant movement data

During the implementation of the ant colony simulator, a major problem has been discovered. The problem is that it is very hard to track the artificial ant movement. Obviously, the simulator paints all the ant into different colour depends on its group using GUI; however, I still need the ant movement information for debugging, observations and modifying the algorithms. Hence, I had implemented the log feature for the ant colony simulator to capture the ant movement data. In addition, I had used the third party library for the log function, because writing log function is not key part for this research project. Apache “log4net” is the third party library which I have used to capture the ant movement data.

Apache log4net is a tool to help the programmer output log statements to a variety of output targets. In case of problems with an application, it is helpful to enable logging so that the problem can be located. With log4net it is possible to enable logging at runtime without modifying the application binary. The log4net package is designed so that log statements can remain in shipped code without incurring a high performance cost. It follows that the speed of logging (or rather not logging) is crucial. At the same time, log output can be so voluminous that it quickly becomes overwhelming. One of the distinctive features of log4net is the notion of hierarchical loggers. Using these loggers it is possible to selectively control which log statements are output at arbitrary granularity. Log4net is designed with two distinct goals in mind: speed and flexibility. (Log4net, 2011)

The log file for the ant colony simulator stored into the “Log” folder under application root directory, and the name is “AntColony.log”. The log file contains the movement for all the ants, which represents the map cell position (x, y). Below is the screen shot for the log file format and information has been capture.

```
Ant Colony Start @2012/1/11 13:25
|Id:0(1577-190,95)|Id:1(1577-190,95)|Id:2(1576-185,95)|Id:3(1576-185,95)|Id:4(1576-185,95)|Id:5(1576-185,95)|Id:
|Id:0(1576-185,95)|Id:1(1576-185,95)|Id:2(1575-180,95)|Id:3(1575-180,95)|Id:4(1575-180,95)|Id:5(1575-180,95)|Id:
|Id:0(1494-180,90)|Id:1(1494-180,90)|Id:2(1574-175,95)|Id:3(1574-175,95)|Id:4(1574-175,95)|Id:5(1574-175,95)|Id:
|Id:0(1495-185,90)|Id:1(1495-185,90)|Id:2(1654-170,100)|Id:3(1654-170,100)|Id:4(1654-170,100)|Id:5(1654-170,100)
```

Diagram 4.6 Log ant movement data

From the above diagram, log format and information describes as “Id: **A** (**B** – **X**, **Y**)”, where **A** represents the Id of the artificial ant; **B** represents the Id of the map cell; **X/Y** represents the X/Y coordinator of the map cell position.

In short, the log feature had implemented using delegate method, which fires different safe thread and does not cause the performance issue for the main algorithms. This delegate method will be used after looping through the artificial ants list each time to capture the ant’s position.

4.3.2.3 Obstacle model

The obstacle model has been implemented to prove the artificial ants choosing the shortest path from nest to food sources under new ant colony algorithms. In order to simplify the implementation, the obstacle model has fixed parameter values, which has set the total number of the artificial ants to be 10 and only has one food source. The width and height of the simulator form are fixed values as well. In addition, the position of the food sources, nest and the obstacle are all fixed locate on the map. The Infinite map boundary feature is disabled for the obstacle model. The diagram below shows the initial stage of the obstacle model.

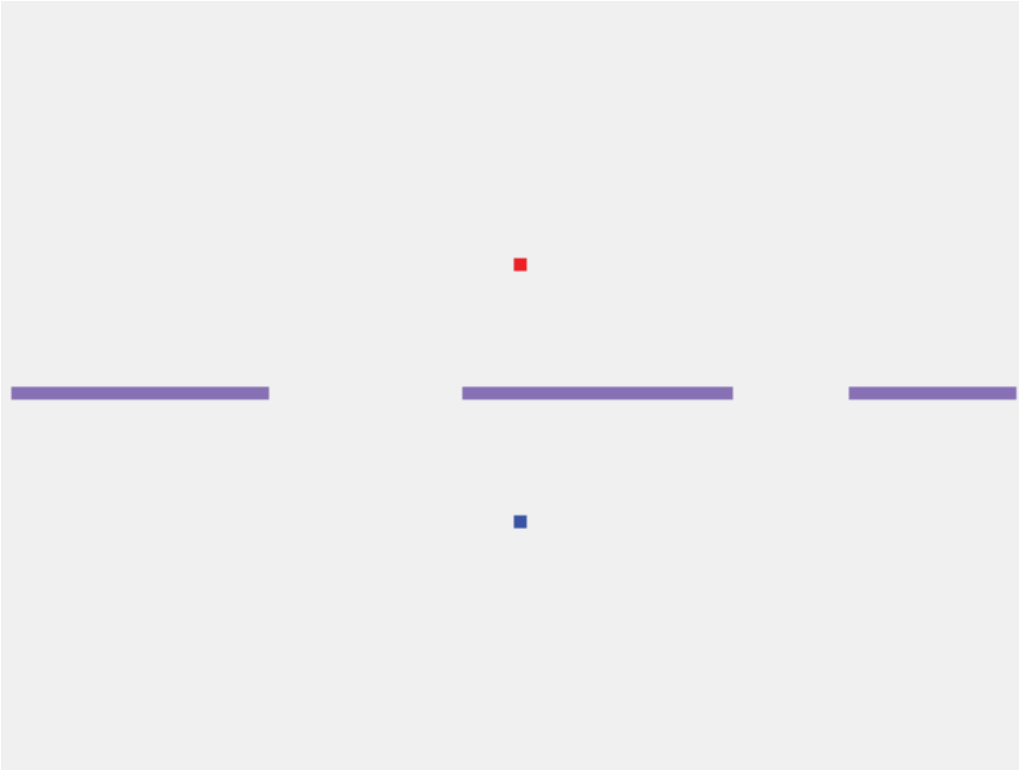


Diagram 4.7 Obstacle Model - Initial Stage

From above diagram, the red map cell represents the ant's nest; the blue map cell represents the food source and the purple map cells which look like a broken line represents the obstacle. According to the obstacle model environment, there are only two major routines connected from the nest to the food source. Obviously, the left routine on the diagram 4.7 is shorter than the right routine.

The diagram below shows the middle stage of the obstacle model.

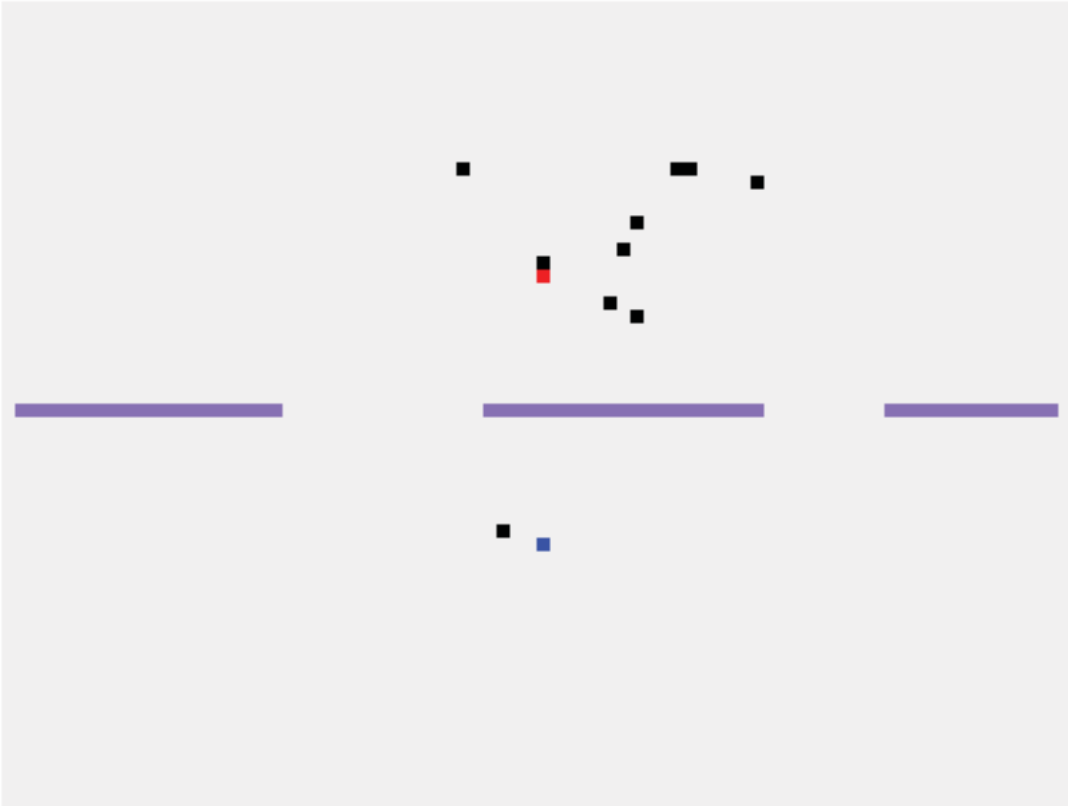


Diagram 4.8 Obstacle Model - Middle Stage

From diagram 4.8, the black map cell represents the ant which current stay in the map cell. The artificial ants are on its way to find the food sources.

The diagram below shows the final stage of the obstacle model.

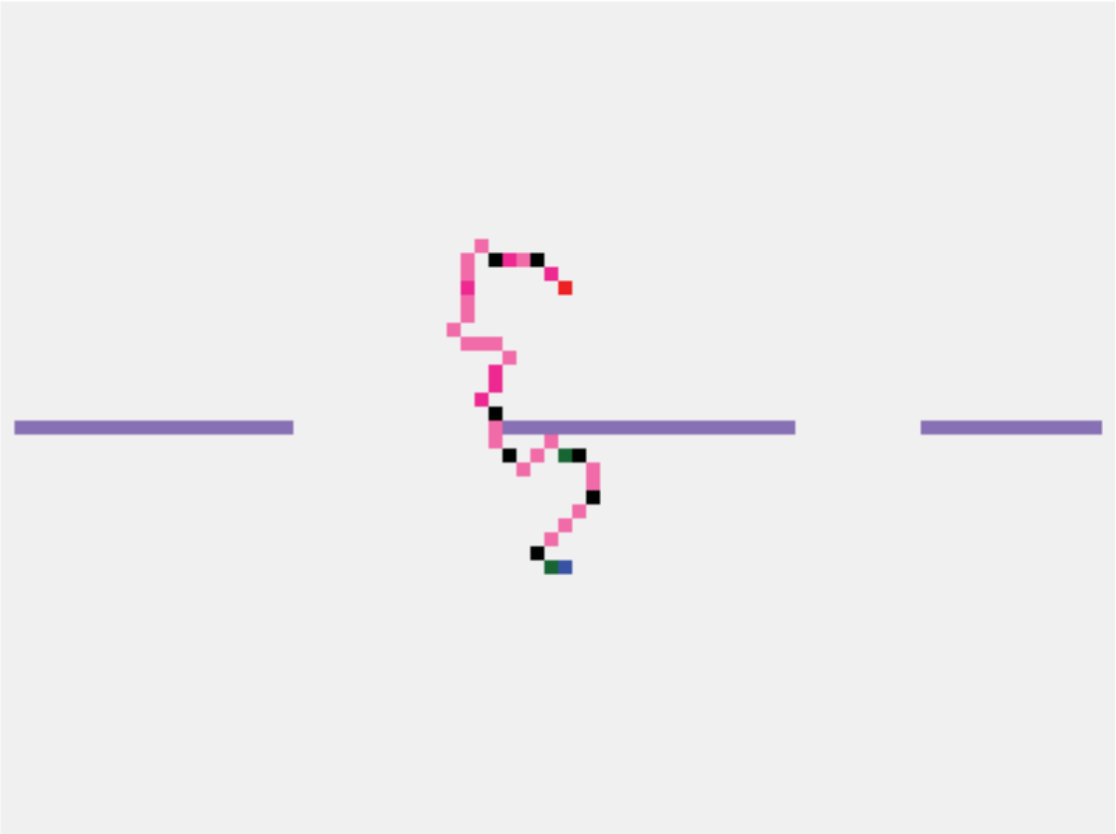


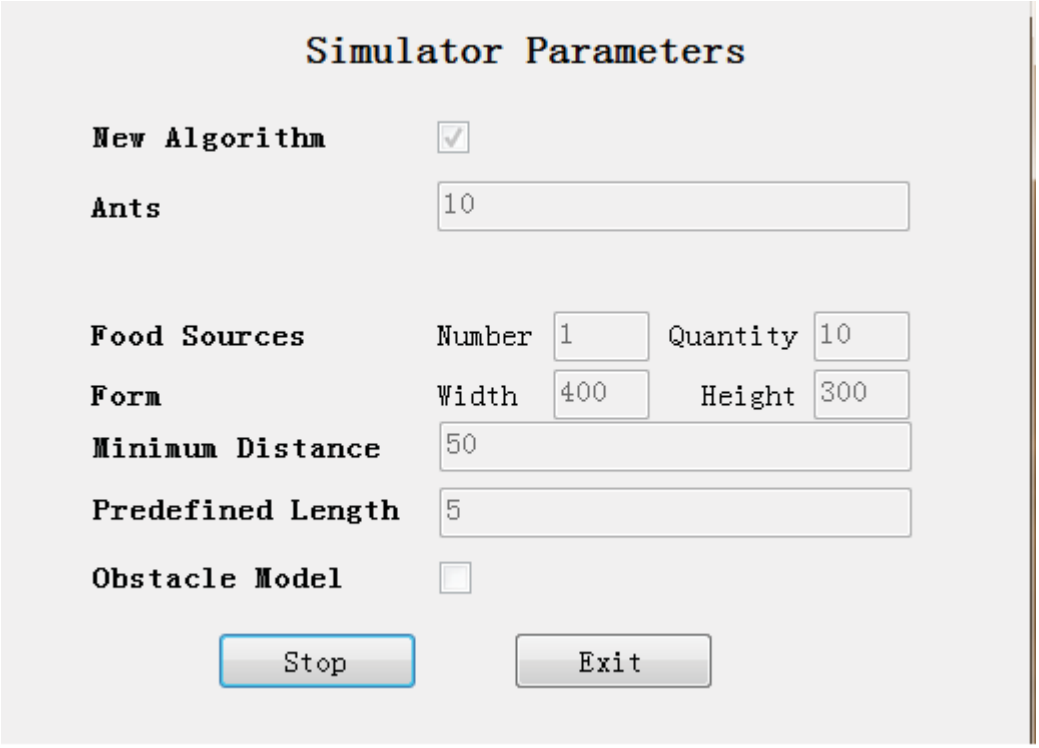
Diagram 4.9 Obstacle Model – Final Stage

The diagram 4.9 clearly shows that the artificial ants has chosen the left routine from their nest to the food sources, which has approved that the new ant colony algorithms works well under the obstacle model.

In general, there are two major reasons to implement the obstacle model. One reason is that this model proves the ability of the new ant colony algorithms, and another is used for the research lab experiment and observations.

4.4 Empirical Observations

This chapter lists the diagram and description for empirical observations. In order to observe the pheromone values remains on the map cell, the simulator paints the map cell into different colour depends on the pheromone values on the cell. If the pheromone value greater than 100 and less than 300, the map cell paint into light pink, if the pheromone value greater than 300 and less than 500, the map cell paint into pink, if the value greater than 500 and less than 1000, it paint into hot pink and if greater than 1000, it paint into deep pink. On the other hand, the map cells paint into Light Goldenrod, Goldenrod, Gold, and Dark Goldenrod if the map cell contains the food pheromone values and these values are under the same condition as the pheromone value.



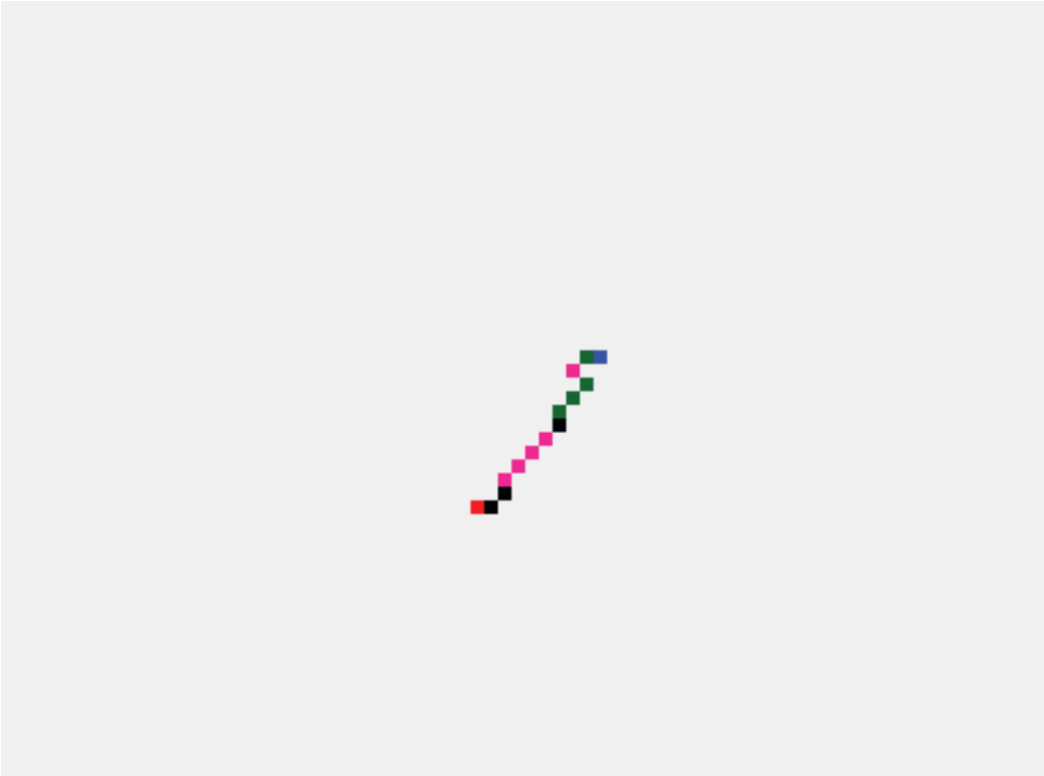


Diagram 4.10 Run simulator using default settings

The below a few diagrams show the simulator run 100 ants, form width is 800 and form height is 600

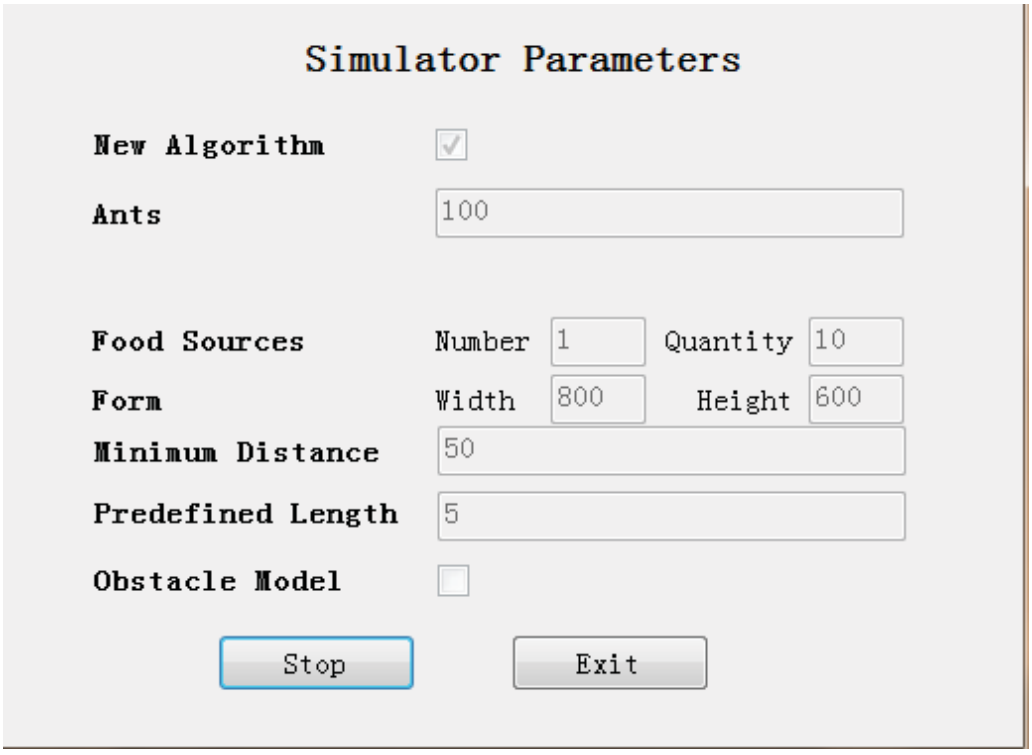


Diagram 4.11 Simulator Observation Series Ⅰ– Parameters

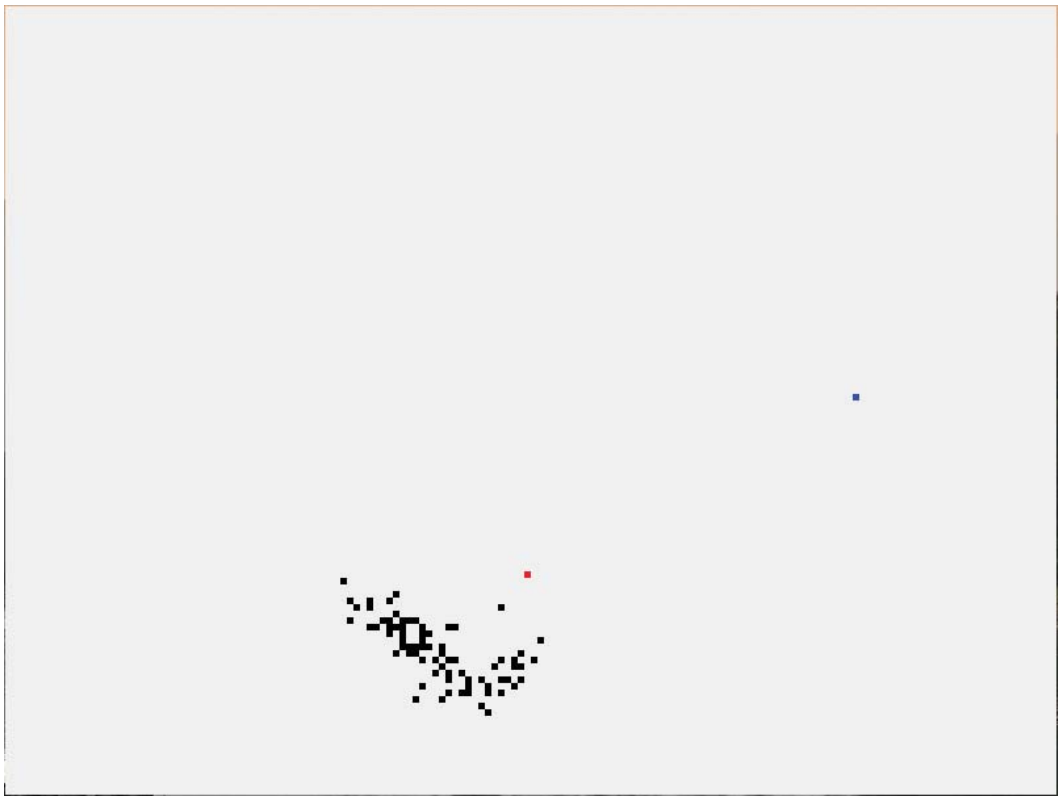


Diagram 4.11 Simulator Observation Series I – Initial Stage

Diagram 4.11 Simulator Observation Series I – First Stage

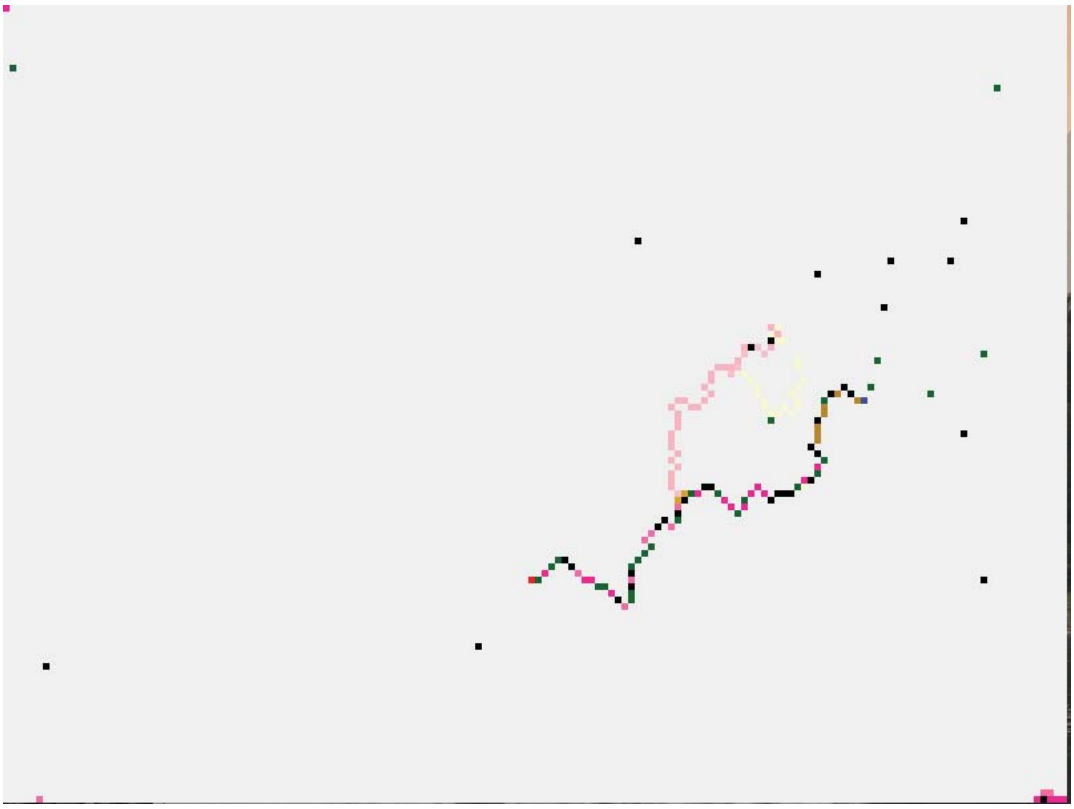


Diagram 4.11 Simulator Observation Series I – Middle Stage

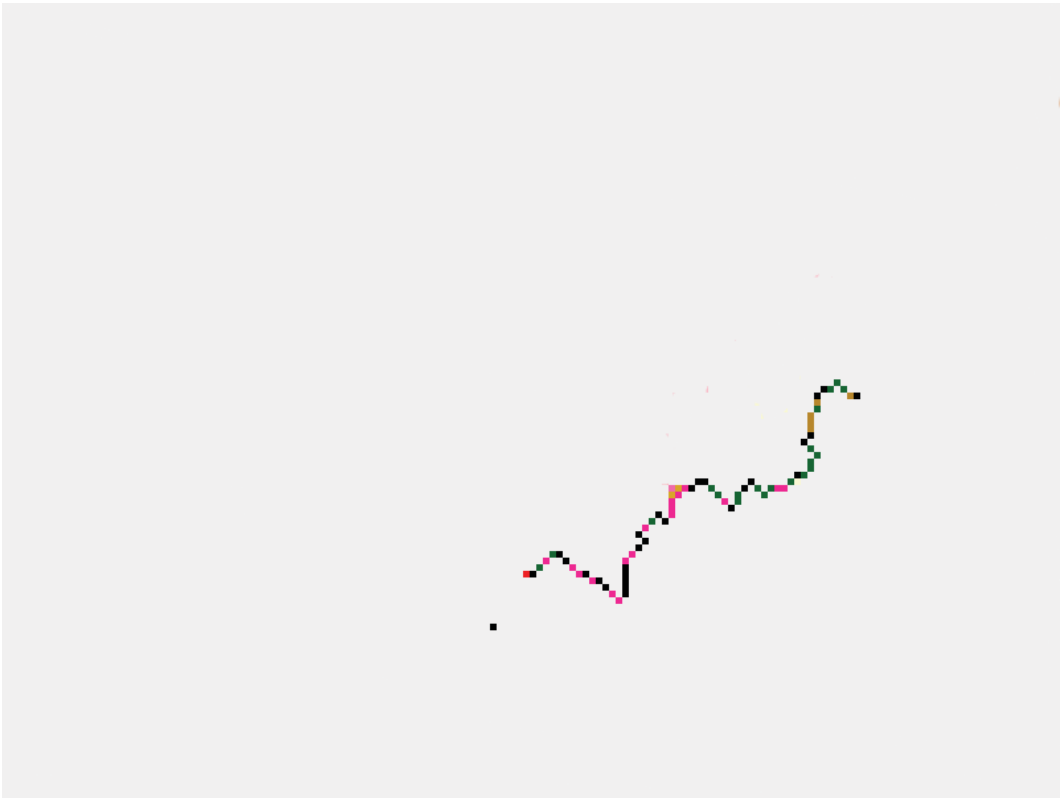


Diagram 4.11 Simulator Observation Series I – Final Stage

From above diagrams of the observation series Ⅰ, we can clearly see the artificial ants search the shortest path from nest to the food sources under different stage, and clearly see the change of the pheromone value and food pheromone values on the map cells. During the middle stage, there exits two optional paths from the nest to the food sources depends on two types of the pheromone values, however, one of the path has higher pheromone and food pheromone values. Therefore, this path is more attract for the artificial ants to go through, on the other hand, less ants select another path because of the lower pheromone values. Finally there is no artificial ant go through another path, and pheromone values of this path decrease until to be zero, and this can be seen on the final stage of the diagrams. The basic theory of this, the path has more ants go through, the more pheromone value can be added to the map cells of the path, and this path has more possibility for the rest of the ants to choose. On the opposite side, the path has less ant go through, the less pheromone value can be added to the map cells of the path, and this path has less possibility for the rest of the ants to choose until there is no ant to choose the path. In addition, it took approximately three and half minutes to run from the initial stage to final stage of the observation series Ⅰ.

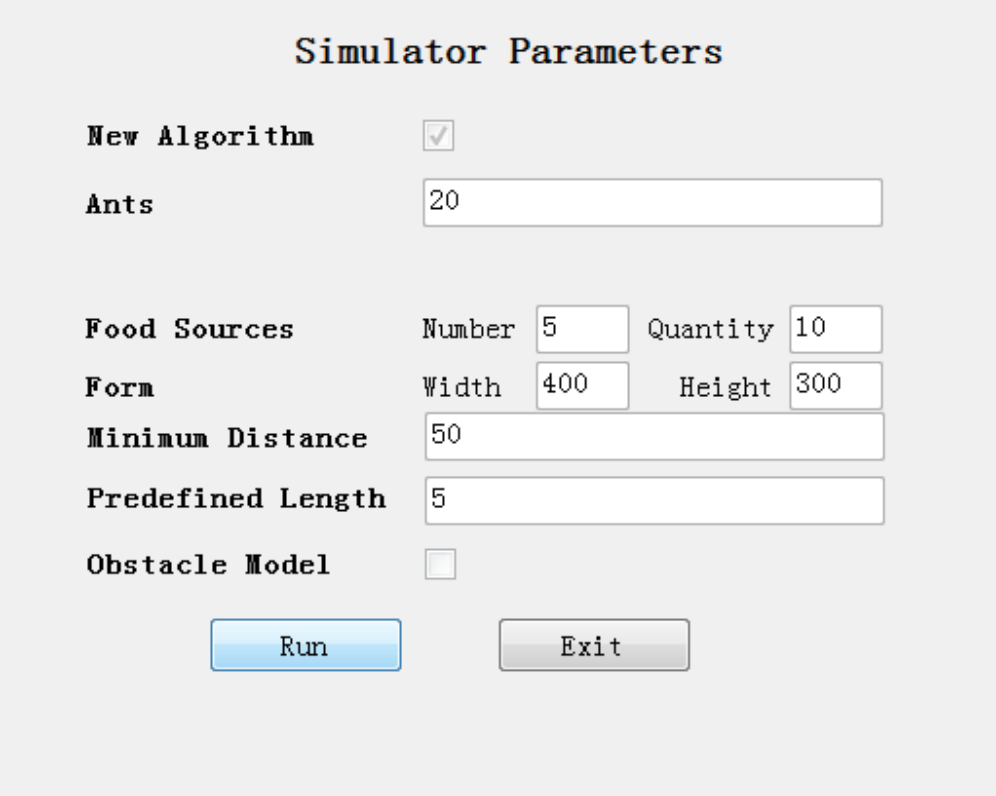


Diagram 4.12 Simulator Observation Series Ⅱ– Parameters

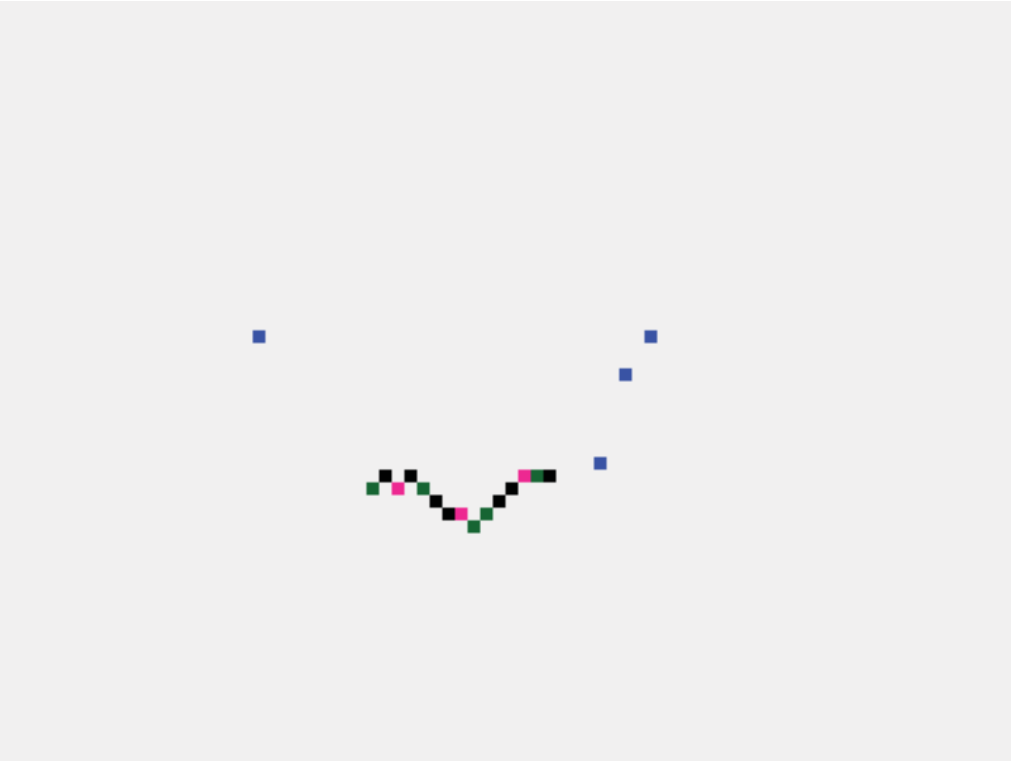


Diagram 4.12 Simulator Observation Series Ⅱ– Result

From the diagrams of the observation series Ⅱ, the simulator sets the parameters values to twenty ants and five food sources. We can clearly see that the artificial ants choose the nearest food sources from its nest. This observation describes the basic theory that if the distance between nest and food sources is shorter, then the artificial ants pass through quickly and the pheromone & food pheromone values become higher. Therefore, the artificial ants choose the shorter distance path under high possibility, and finally the ant colony establishes the shortest route from the nest to the nearest food source.

Chapter 5 Lessons Learned and Future Work

5.1 Lessons Learned from the Research

Two major difficulties became apparent during this research project, which are performance issues caused by LINQ for algorithms and the implementation of the obstacle model. Especially, full implementation of the obstacle model using all the parameters of the simulator is more complicated than was originally thought.

I have investigated and debugged through the entire set of algorithms to try and find out which part of the function slows down the whole process. Finally, some of the LINQ (Box & Hejlsberg, 2007) statements have been identified which use too much time to query the data. The execute time of each of these statements approximately more than few hundreds milliseconds, which means each ant will use one or two seconds to finish algorithms. If the simulator use ten ants which is the default setting, the iteration will cost 10-20 seconds to go through, and this is too slow for the algorithm to be completed, and this leads to around a few hours to get the optimization results of the ant colony algorithms. In fact, I have spent quite a lot time check the performance between LINQ and the method which I have written replacing some of the LINQ statements. According to the observation and calculation, I have used the method which I have written, and only replaced a few of the unnecessary LINQ statement. Finally, it seems the performance has dramatically improved and this performance suits the ant colony simulator.

Full implementation of the obstacle model is very complicated and this has made it more difficult to achieve the performance requirement. A few major issues are listed below:

- Nest and food sources can be dynamically located on the map, which cause conflicts in the obstacle model.
- The obstacle objects need to be dynamically located on the map but depend on the food sources and the nest.
- The simulator needs to work out the pre-set condition for the lab and observation purposes of the obstacle model.

Due to these issues the current simulator simplifies the obstacle model by using fixed parameter values and fixes the position of the food source and nest on the map for lab and observation purposes.

5.2 Future Work

There are several additional methods that can also be useful for ant colony algorithms. After studying a few improved ant colony algorithms from other research topics, particularly the topics of sensational and consciousness, I was enlightened on ant communication policy. Hence, the ant communication policy has been implemented for this research project, and this policy defines the artificial ants into two groups to ensure communication. This algorithm simulates the ant's natural behaviour, and solves a few problems that had previously occurred in the smaller research project for my post-graduate diploma.

According to the ants' searching first stage & middle stage for the algorithms of the sensational and consciousness, the ant communication policy can be modified to attach the AST (absolute sensor threshold) & CST (contrast sensor threshold). A few structures need to be modified to attach the AST & CST. Firstly, the simulator needs two parameters to define the values of the AST & CST and these two values need to be adjusted for observation purposes. Secondly, the ant colony algorithms need to be modified using the AST, which means if the pheromone value & food pheromone value of the map cell are less than the AST, then the artificial ant considers those two pheromone values to be zero for that map cell. Thirdly, the algorithms should be modified using CST and some examples are listed below:

- The memory of the artificial ant need to be changed which requires another queue to store all the paths which the ant has passed. This may cause a problem with memory capacity. The number of the times that the ant used a path also needs to be stored.
- To implement the sensational function of the artificial ant, which means that the ant will consider a neighbouring path which it had passed through previously and this path had been passed through many times.
- All the artificial ants which do not carry food leave the different pheromone on the map cell. This will help artificial ants only choose the map cell which has its own pheromone value for the sensational.
- If the change of the pheromone value is greater than the CST, the algorithms use the procedures of this research project, or they use the artificial ant's sensational function.

In general, the ant colony communication policy combined with AST and CST of the ant sensational and consciousness is the future work for the ant colony simulator.

Chapter 6 Conclusion

The ant colony simulator, which simulates the natural ant's behaviour to find the food sources and return to its nest, will assist in producing new computer science algorithms that solve some commercial problems such as: travelling salesman problem, vehicle routing problem, network routing problem etc. The ant colony optimization has been used for quite a few industries recently, and the logic works well to suit the requirement of those industries. The wide variety of algorithms (for optimization or not) seeking self-organization in biological systems has led to the concept of "swarm intelligence", which is a very general framework in which ant colony algorithms fit (Ant colony optimization, 2010).

This research project caused me to look deeply into simulator technology, and it is quite interesting to think about using the simulator to sort out our real life's problems. I have learned much from ant colony algorithms, especially using the math problems to deal with artificial intelligence for computer science. In addition, I have read many articles relating to ant colony algorithms and among these articles I have studied a few algorithms and understand clearly the fundamental mathematical model which underpins these algorithms. Based on all my reading and studying, I have defined a new improved ant colony algorithm which introduces the ant communication policy. Moreover, there are three new major features that have been implemented for the ant colony simulator, which are infinite map boundary, log ant movement data and the obstacle model. These new features make the simulator more flexible.

Many types of network optimization problems in the connection of real life situation can be tackled by computer science techniques. The comparative study helps to solve the large routing problem where the real data is not available. Simulation techniques being an important tool for research development can be implemented in such a real life situation (MOHD & KAANODIA, 2009), and can be used for major industry to get results from the simulation under certain conditions without modifying their current work frames and procedures. Research into the quality of service (QoS) routing of the internet is more important these days. In the field of computer networking and other packet-switched telecommunication networks, the traffic engineering term quality of service (QoS) refers to resource reservation control mechanisms rather than the achieved service quality (Botley, 2006). Quality of service has the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow (Quality Of Service, 2010).

According to the algorithms used for an ant to find the path, we can replace the routine table with the probability table for every node over the network, and the probability value is the pheromone value. There is a common problem for the logistic supply chain industry, which is finding the most cost efficient way to distribute goods across the logistic network under certain condition (Garn, 2007). Solving a vehicle routing problem means to find the best route to service all customers using a fleet of vehicles in the industry (Rizzoli, Oliverio, Montemanni, & Gambardella, 2007). This vehicle routine problem can be solved using ant colony optimization.

In conclusion, ant colonies, and more generally social insect societies, are distributed systems that, in spite of the simplicity of their individuals, present a highly structured social organization. As a result of this organization, ant colonies can accomplish complex tasks that in some cases far exceed the individual capabilities of a single ant (Dorigo & Stutzle, Ant Colony Optimization, 2004). An ant colony simulator simulates the natural ant's behavour, which accomplish the task for the ant colony finding the shortest path between nest and food sources in two dimentional graphic of the computer system. In addition, the biological scientest have discovered that the intelligence of evey single ant is not higher under their long term research. In addition, they do not appear to be under central direction; however, they can all work together to find food and build their nest for their new generation. The ant colony demonstrates higher intelligence and more ability than each individual ant in the society (Duan, 2005). Therefore, the ant colony algorithms are the most important part of the ant colony simluator,and I believe that the ant communication policy, which has been implemented for this reasearch project, improves the efficiency of the ant colony algoritms.

Bibliography

Animal Behavior/Pheromones in ants and bees. (2007, 4 25). Retrieved 4 19, 2010, from Wikibooks: http://en.wikibooks.org/wiki/Animal_Behavior/Pheromones_in_ants_and_bees

Travelling salesman problem. (2009, 2 20). Retrieved 4 1, 2009, from Wikipedia: http://en.wikipedia.org/wiki/Travelling_salesman_problem

Ant colony optimization. (2010, 4 13). Retrieved 4 19, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Ant_colony_optimization

Quality Of Service. (2010, 05 2). Retrieved 05 03, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Quality_of_service

Bilchev, G. A., & Parmee, I. C. (1995). The ant colony metaphor for searching continous space. 993: 25~39.

Botley, L. (2006, 5 4). *Artificial intelligence network load balancing using Ant Colony Optimisation*. Retrieved 2 10, 2010, from The Code Project: http://www.codeproject.com/KB/recipes/Ant_Colony_Optimisation.aspx

Box, D., & Hejlsberg, A. (2007, 2). *LINQ: .NET Language-Integrated Query*. Retrieved 4 20, 2010, from Microsoft: http://msdn.microsoft.com/library/bb308959.aspx

Chen, L., & Ma, J. (1996). *A New Psychology.* BeiJing: BEIJING Normal University Publishing Group.

Dorigo, M., & Gambardella, L. M. (1996). *Ant colonies for the traveling salesman problem.* Belgium: Universite Libre de Bruxelles.

Dorigo, M., & Stutzle, T. (2004). *Ant Colony Optimization.*

Duan, H. B. (2005). *Ant Colony Algorithms : Theory and Applications.* Bei Jing: Science Publisher.

Gambardella, L. M., & Dorigo, M. (1996). A study of some peoperties of Ant-Q. *Proceedings of PPSN IV–Fourth International Conference on Parallel*, (pp. 656-665). Berlin.

Gambardella, M. L., & Dorigm, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies. *Proceedings of the IEEE International Conference on Evolutionary Computation*, (p. 622~627).

Gambardellaa, L. M., Oliveriob, F., Lucibellob, E., & Montemannia, R. (2003). *Ant Colony Optimization for vehicle routing in advanced logistics systems.* Retrieved 3 20, 2010, from Dalle Molle Institute for Artificial Intelligence: http://www.idsia.ch/~luca/MAS2003_18.pdf

Garn, W. (2007, 7 20). *Vehicle Routing Problem*. Retrieved 3 10, 2010, from Argesim: http://osiris.tuwien.ac.at/~wgarn/VehicleRouting/vehicle_routing.html

Goss, S., Aron, S., & Deneubourg, J. L. (1989). *Self-organized shortcuts in the argentine ant.* Naturwissenschaften.

Hennigan, T. (2010, 3 14). *Ants Millimeter Messengers*. Retrieved 5 2, 2010, from Answers In Genesis: http://www.answersingenesis.org/articles/am/v5/n2/ants

Log4net. (2011, 11 24). *Apache log4net™ Features*. Retrieved 3 20, 2012, from Apache Logging Service: http://logging.apache.org/log4net/release/features.html

MOHD, R., & KAANODIA, K. (2009, 5). *A COMPARATIVE STUDY OF SIMULATED VEHICLE ROUTINE PROBLEM AND LINGO SOFTWARE SOLUTION IN LPG DISTRIBUTION SYSTEM.* Retrieved 4 20, 2010, from Journal of the Applied Mathematics, Statistics and Informatics: http://fpv.ucm.sk/jamsi/docs/v05n02_12_2009/v05_n02_03_KAANODIA_RIZWANULLAH.pdf

MSDN, M. . (n.d.). *Delegates (C# Programming Guide)*. Retrieved 2 25, 2012, from MSDN Library: http://msdn.microsoft.com/en-us/library/ms173172(v=VS.80).aspx

QinLing, ChenLing, & ChenHongJian. (2003, 15(10)). Sensational and consciousness ant colony algorithm. Journal of system simulation, 1418~1425.

Rizzoli, Oliverio, Montemanni, & Gambardella. (2007). *Ant Colony Optimisation for vehicle routing problems.* Retrieved 3 4, 2009, from Dalle Molle Institute for Artificial Intelligence: http://www.idsia.ch/idsiareport/IDSIA-15-04.pdf

Thomas, R., & Silva, C. (n.d.). Ant Colony Optimization for dynamic Traveling Salesman. Munich, Germany.

Thomas, S., & Dorigo, M. (2004). *Ant Colony Optimization.* MIT Press.

Weber's Law and Fechner's Law. (n.d.). Retrieved 2 22, 2012, from The Center for Neural Science : http://www.cns.nyu.edu/~msl/courses/0044/handouts/Weber.pdf