The Implementation of Genetic Algorithm in Path Optimization

By

Suriana Jumali

Dissertation submitted in partial fulfillment of
the requirement for the
Bachelor of Technology (Hons)
Business Information System

JULY 2005

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Implementation of Genetic Algorithm in Path Optimization

By

Suriana Jumali

A project dissertation submitted to the
Information System Programme
Universiti Teknologi PETRONAS
In partial fulfill of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(BUSINESS INFORMATION SYSTEM)

Approved by,

_____

(Vivian Yong Suet Peng)

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR, TRONOH
PERAK DARUL RIDZUAN
JULY 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work of my own except as specified in the references and acknowledgements, and that the original contained here in have not been undertaken or done by unspecified sources of persons.

SURIANA JUMALI

ABSTRACT

Traveling Salesman Problem (TSP) is a classical problem in Artificial Intelligence (AI) field. Since 1800s when first mathematical problems related to TSP was treated, it became an interesting topic of optimization problem to be studied. In this project, TSP will be used to model and easy visualize the path optimization problem and Genetic Algorithm (GA) was chosen to be implemented in resolving the problem. This project will focus on the static variable referring to the length of distance as the fitness function of optimization. The idea of resolving TSP study is to come out with the shortest path among all possible solutions of tour to be taken. However, the major concern here is how to ensure that the optimum result is obtained. Therefore, the operators and parameters of GA itself were studied in depth particularly the mutation operator. Experiments were conducted to measure the effectiveness of two different types of mutation method namely swapping method and inversion method. The comparison of both performances in achieving optimum result had been analyzed in detail. Therefore, the implementation of GA in path optimization can be ascertained offering a compelling result.

ACKNOWLEDGEMENT

First and foremost, I would like to thank my family members and loved ones, who consistently given their love, support, and encouragement in the course of completing this project.

I also would like to express my deepest gratitude to my supervisor, Mrs Vivian Yong Suet Peng for her guidance, knowledge, experience, and feedback throughout the process of completing this project. Her valuable advice and constructive feedback have positively contributed to the success of this project.

I am extremely grateful for generous help, freely given by the other lecturers of AI and Java programming field namely, Dr Ahmad Kamil, Mr Jale Ahmad and Mr Anang Hudaya who gave invaluable assistance and ideas towards the project. They have dedicated effort to solve raised issues in completing the project. Their comment and suggestion were invaluable. Thanks also to Dave Todd, where his work becomes a major reference in this project. Special note gratitude is extended to Mr Rasky, IT Technician, for his help and support in provision of computer equipments and software.

Last but not least, I would like to thank those who gave generously provided information and comments in completing this project especially Dr. Mazlan Abbas who gave constructive comments towards a better production of this project. The support and assistance, coming from all parties involved in this project, have contributed to the success of this project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

TSP	:	Traveling Salesperson Problem
AI	:	Artificial Intelligence
HIG	:	Hamilton's Icosian Game
NP	:	Nondeterministic Polynomial
GA	:	Genetic Algorithm
SA	:	Simulated Annealing
GP	:	Genetic Programming
IGA	:	Interactive Genetic Algorithm
GIS	:	Geographical Information System
DA	:	Dijkstra Algorithm
HA	:	Heuristic Algorithm
DNA	:	Deoxyribonucleic Acid
PE	:	Permutation Encoding

CHAPTER 1
INTRODUCTION

1.1 Background of Study

Even though the origin of Traveling Salesperson Problem (TSP) is obscure, TSP is classic to Artificial Intelligence (AI) and computer science. Mathematical problems related to the TSP were treated in the 1800s by the Irish mathematician Sir William Rowan Hamilton and by the British mathematician Thomas Penyngton Kirkman [2]. Their early work named Hamilton's Icosian Game (HIG) requires players to complete tours through the 20 points using only the specified connections. A nice discussion of HIG can be found in book "Graph Theory 1736-1936" [3]. Since then, TSP became an interesting topic of AI study and a Nondeterministic Polynomial (NP) Complete Problem with many ramifications of search strategies.

Figure 1.0: The Hamilton's Icosian Game [2]

Subsequently, the general form of TSP appears to be first studied by mathematician and economist Karl Menger starting in the 1920's who publicized it among his colleagues in Vienna. In the 1930's, the problem reappeared in the mathematical circles of Princeton. In the 1940's, it was studied by statisticians Mahalanobis , Jessen in 1942, Gosh in 1948, Marks in 1948 in connection with an agricultural application and the mathematician Merill Flood popularized it among his colleagues at the RAND Corporation. The growth of the TSP as a topic of study can be found in Alexander Schrijver's paper ``On the history of combinatorial optimization (till 1960)" [3]. The pictorial history of TSP can be depicted as follow:

Table 1.0: The Pictorial History of Traveling Salesperson Problem (TSP) Study [3]

Year	Research Team
1954	G. Dantzig, R. Fulkerson, and S. Johnson
1971	M. Held and R.M. Karp
1975	P.M. Camerini, L. Fratta, and F. Maffioli
1977	M. Grötschel
1980	H. Crowder and M.W. Padberg
1987	M. Padberg and G. Rinaldi
1987	M. Grötschel and O. Holland
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook
2004	D. Applegate, R. Bixby, V. Chvátal, W. Cook and K. Helsgaun

Along the period of time, researchers have tried various algorithms to solve TSP problem. To name a few, the algorithms include simulated annealing [5, 6], discrete linear programming [7], neural networks [8], branch and bound [9], 2-opt [10], Markov chain [11] as well as Genetic Algorithm (GA).

1.2 Problem Statement

1.2.1 Problem Identification

In solving path optimization problem, TSP is used to easy visualize and modeled the case. The statement of TSP is simple and easy to state as the following:

Given a finite number of destinations, the salesperson is required to visit each destination. There could be many possible routes and the challenge is to find the most effective route between the destinations. To do so, the order of destination visited is crucial. The salesperson will pass through all the destinations and coming back to the starting point in the most effective way. The effectiveness will be gauged by the distance taken which lead to find the shortest path among all possible routes.

Segregated below are the main problems associated with path optimization. However, it is without taking into consideration other dynamic factor namely traffic congestion, type of transportation used as well as cost incurred. Assuming all the factors are constant, the shortest path among all is the biggest concern here.

Scenario 1

An individual who needs to travel from one place to another in carrying out their jobs.

Scenario 2

A transportation company who has a traveling tour to be completed in a routine basis.

Scenario 3

A logistic or courier company who needs to deliver goods to their customers from one place to another daily.

1.2.2 Significant of the Project

Nowadays, people become very sensitive about their time constraint and need to be more efficient. Everything needs to be fast even when it comes to travel from one place to another. Path optimization is the answer to overcome the issue and TSP is used to model the problem. There are numerous type of algorithm can be used in solving TSP problem. Each has its own advantages as well as disadvantages.

In this project, GA is chosen as a tool to find the optimize path. Before it is chosen, a thorough research was done comparing each possible algorithms can be applied. However, once the type of algorithm to be used is chosen, it is always become a concern on how optimize the optimum result can be? In this case, how optimize the optimum result GA can offer in order to solve the problem? Therefore, each element of GA will be studied in depth comprises of its operators and parameters. So, the implementation of GA in achieving path optimization can be ensured offering a compelling result.

1.3 Objective and Scope of Study

1.3.1 Objective of Study

The objectives of this project can be summarized as follows:

- To understand on TSP study together with the underlying concept of GA as an optimization tool applied to it.
- To aid decision making in path optimization problem by providing the shortest distance tour among all possible solutions.
- To compare different types of mutation methods used as an important operators in GA to achieve optimum result.
- To observe on various important parameters used in GA that encroach on GA functions to achieve optimum result.

1.3.2 Scope of Study

The simulated system will be developed using JAVA language since JAVA programs are executable on many platforms. The scope of study for the system will be focusing on the mutation operators and parameters used in GA in order to achieve optimize result. All the components will be the determine factors of the optimization result. The components are as the following:

Mutation Operators

The mutation operators used in this system will be subdivided into two different methods as follows:

- Swapping Method
- Inversion Method

Parameters

There are six parameters of GA used in this system as follows:

- Maximum number of iterations
- Mutation rate
- Display every Xth iteration
- Population size
- Convergence percentage
- Number of cities

The features of the system prototype will be as follows:

- Give the user flexibility to determine the type of mutation operator and the value of parameters to be used.
- Calculate the fitness of each possible solution in order to come out with the shortest path to be taken among all.
- Display the result that comprises of the final tour (the shortest path) together with the city locations, the length of distance as well as simulation of the route to be taken graphically.

1.3.3 Feasibility of the Project within the Time and Scope

Schedule Feasibility

The system will be completed according to the budgeted time frame depicted by the Gantt Chart. There are altogether 14 weeks allocated to produce both the dissertation as well as complete system. The research will be carried out prior to the development process. However, the project report will be written in parallel with both research as well as development process to ensure that it is inline with the to-be system.

Scope Feasibility

The project system focuses on the implementation of GA that will be used to solve the path optimization problem. The scope is to focus on the static variable without taking into consideration the other dynamic variables. Therefore, the optimization will be gauged by the length of distance solely. The system prototype will be built then to simulate the result. Experiments will be conducted in order to ensure the effectiveness of the algorithm in achieving optimize result.

Technical Feasibility

Due to the point that the system developer is well verse with Java programming language, the technical feasibility can be assured. There is no cost related to the project as the system will be developed using available software namely JAVA 2 Standard Development Kit 1.4.2_08 and Forte for Java. There are also adequate resources available to support the project gained. The source of information can be gained via books, journals, papers or even online resources on Internet.

CHAPTER 2
LITERATURE REVIEW AND THEORY

From the research conducted, TSP is one of the most intensely studied problems in computational mathematics and yet no effective solution method is known for the general case [3]. Although the complexity of the TSP is still unknown, for over 50 years its study has led the way to improved solution methods in many areas of mathematical optimization. Among all techniques, GA is chosen as TSP solver because GA based solutions are currently available for simultaneous search of multiple routes while the other algorithms only produce the best one at a time. The subsequent sections will be discussing in detail on how GA is applied to TSP as well as the comparison of GA with other type of algorithms.

2.1 The Brief History of GA

GA first developed by John Holland in the 1970s [12] is search algorithm based on the mechanics of natural selection and natural genetics. The algorithm works based on biological background nature. The process by which successive generations of animals and plants are modified so as to approach an optimum form. It is used to search large, non-linear search spaces where expert knowledge is lacking or difficult to encode and where traditional optimization techniques fall short [13, 14, 15]. Since the 80's, much work has been done in applying GA to the TSP [13,16, 17, 18, 19]. Until now GA is still the most widely used algorithm to cater optimization problem and TSP is an easily visualize problem solved by exploring the interaction in genetic search.

2.2 The Theory of GA

Marek Obitko's Site stated that GAs are a part of evolutionary computing, which is a rapidly growing area of AI. GAs are inspired by Darwin's theory about evolution and solution to a problem solved by GA is evolved [25]. From Generation5.org [26], the general algorithm of GA can be summarized as follows:

Create a Random Initial State

An initial population is created from a random selection of solutions which are analogous to chromosomes. This is unlike the situation for Symbolic AI systems, where the initial state in a problem is already given instead.

Evaluate Fitness

A value for fitness is assigned to each solution referring to chromosome depending on how close it actually is to solving the problem. Thus, the solutions are evaluated whether they arrive to the answer of the desired problem. These solutions are the possible characteristics that the system would employ in order to reach the answer.

Reproduce and Children Mutate

Those chromosomes with a higher fitness value are more likely to reproduce offspring which can mutate after reproduction. The offspring is a product of the father and mother, whose composition consists of a combination of genes from them. This process is known as "crossing over".

Next Generation

If the new generation contains a solution that produces an output that is close enough or equal to the desired answer then the problem has been solved. If this is not the case, then the new generation will go through the same process as their parents did. This will continue until a solution is reached.

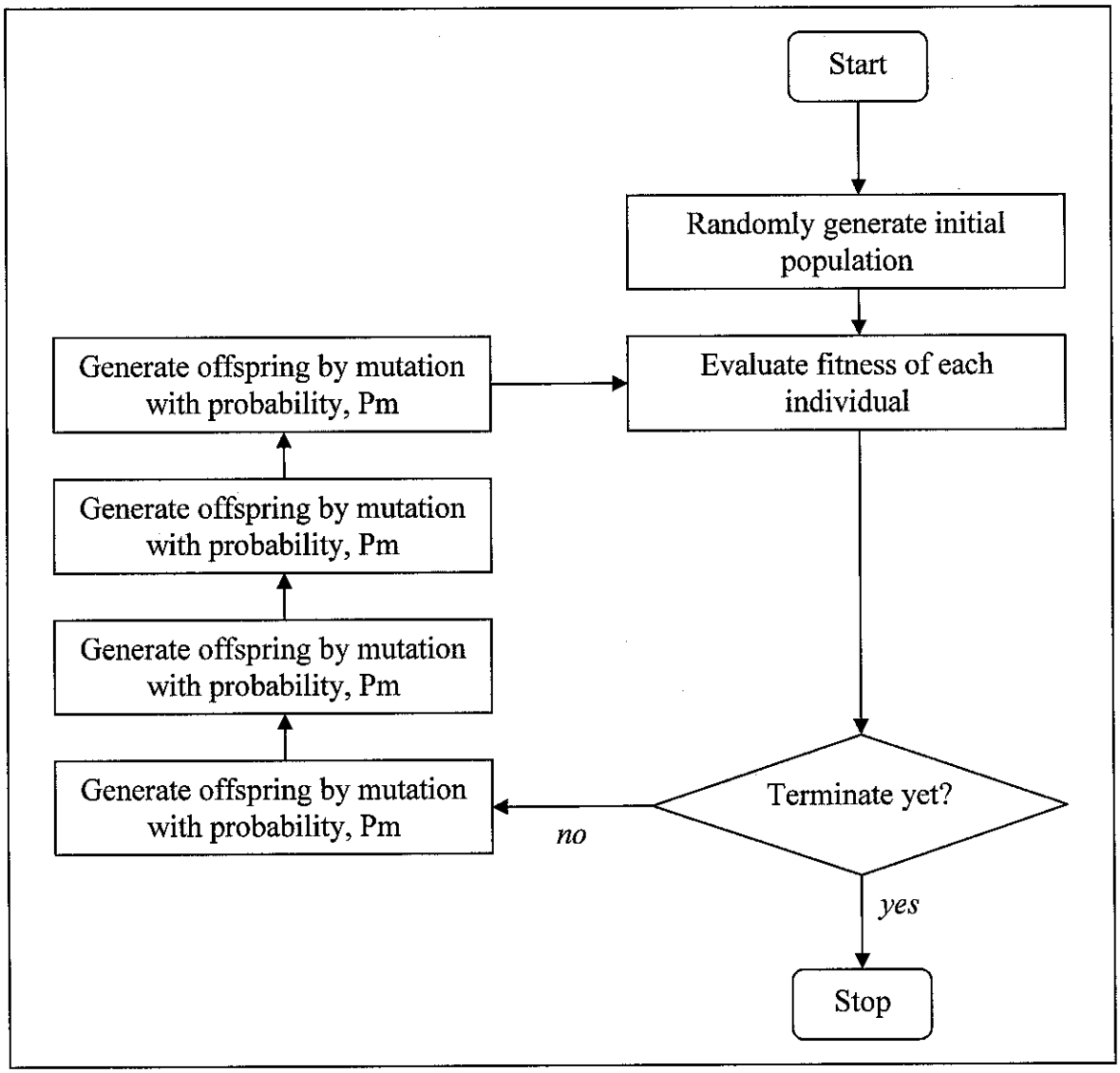

Figure 2.0: The Basic Genetic Algorithm (GA) [25]

In general, how GA works can be depicted from Figure 2.0. However, GA is still can be used together with other type of algorithms in order to enhance the effectiveness of optimization result offered. Even more, GA itself can be customized accordingly for optimization purpose. The subsequent section will be discussing on possible related techniques which can be used along with GA in solving TSP problem.

2.3 Related Techniques with GA in Solving TSP

2.3.1 Simulated Annealing (SA)

In the 1985, Kirkpatrick and Toulouse used simulated annealing to solve TSP followed by Learhoven and Aarts in the 1987[3]. SA is a related global optimization technique which traverses the search space by testing random mutations on an individual solution. A mutation that increases fitness is always accepted. A mutation which lowers fitness is accepted probabilistically based on the difference in fitness and a decreasing temperature parameter. In SA parlance, one speaks of seeking the lowest energy instead of the maximum fitness. SA can also be used within a standard GA algorithm, simply by starting with a relatively high rate of mutation, which decreases over time along a given schedule.

2.3.2 Genetic Programming (GP)

GP is a related technique popularized by John Koza, in which computer programs, rather than function parameters, are optimized [3]. GP often uses tree-based internal data structures to represent the computer programs for adaptation instead of the list, or array, structures typical of GA. GP algorithms typically require running time that is orders of magnitude greater than that for GA, but it may be suitable for problems that are intractable with GA. Thus, it is possible to be used as an extension of GA.

2.3.3 Interactive Genetic Algorithm (IGA)

IGA is part of GA that also based use human evaluation. It is usually applied to domains where it is hard to design a computational fitness function, for example, evolving images, music, artistic designs and forms to fit users' aesthetic preference [3]. Therefore, with the interactive factors involved, a more interesting problem can be solved which able to cater more than just a simple plain data.

2.4 Comparison of GA with Other Algorithms

As stated before, there are various possible algorithms can be used to solve TSP problem. Each has its own advantages as well as disadvantages. The chosen of GA is based on research conducted comparing all possible techniques can be used. Therefore, the subsequent section will be discussing on the comparison of GA to other algorithms followed by a conclusion of the advantages as well as the drawbacks of GA itself.

2.4.1 GA-Based Multiple Route Selection for Car Navigation [27]

Basabi Chakraborty used GA to find multiple route selection for car navigation. There are a lot of good solutions like Dijkstra Algorithm (DA), breadth-first search, Bellman-Ford algorithm and others available for optimization problem. But simultaneous search for multiple semi-optimal routes are difficult with the above mentioned algorithms as they produce the best one at a time. Therefore, GA is the answer for simultaneous search of multiple routes. However, the problem in finding multiple routes is that selected routes resemble each other, partly overlap. To overcome this problem, a GA based algorithm with a novel fitness function is used for simultaneous search of multiple routes for car navigation system avoiding overlapping. Three different algorithms were compared namely Inagaki algorithm, Inogue algorithm also known as DA as well as GA itself.

Inagaki [36] proposed an algorithm in which chromosomes are sequences of integers and each gene represents a node ID selected randomly from the set of nodes connected with the node corresponding to its locus number. The idea is to minimize the effect of overlapping solutions. But the proposed algorithm requires a large solution space to attain high quality solution due to its inconsistent crossover mechanism. On the other hand, Inogue [37] proposed a method for finding out multiple different (non overlapping) short routes by dividing the road map in multiple areas and putting different weights in each of them so that the selected routes are through different areas of the map. But as there is no direct method for comparing the overlapping of the

selected paths this method is not guaranteed to select minimally overlapped multiple shorter paths.

Then, GA has been evaluated against mentioned algorithm using a real road map. GA can be used effectively for searching multiple routes from a real road map with a rank order i.e., shortest, second shortest, 3rd shortest and so on (*k* shortest path problem). The road map is first converted into a connected graph, considering each road crossing as a node in the graph and all such nodes are numbered. The roads in map are represented by links in the graph. The distance between any two crossings is considered as the weight of the link between the corresponding nodes. The starting point and the destination on the map are defined as the starting node and goal node on the graph. Any possible path from start node to goal node or destination node via other nodes is a possible solution and coded as a chromosome by using node numbers. However, looping in the path is generally avoided. Figure 2.1 represents a simple graphical representation of a road map and the possible routes from source node 0 to destination node 9.

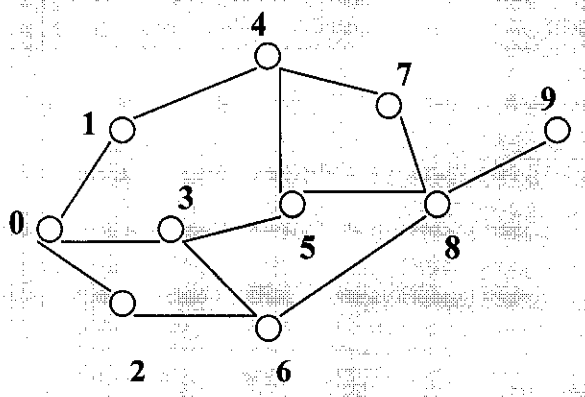

Figure 2.1: Graphical Representation of Road Map and Routes [27]

The comparative results of the three different algorithms showed that DA takes much shorter time compared to others for finding out the shortest route and also is able to find out better paths in terms of distance. But successive short routes are highly overlapped. Both Inagaki's method and GA take longer time than DA but alternate routes can be found out simultaneously. The GA is found to be better to avoid route overlapping compared to Inagaki's method. The average run time and the weight of the path in both methods are nearly equal with GA being slightly better. The average weight of the selected path is also close to the average weight of the paths found out by DA. For a better view, the result is summarized in Table 2.0 as the following:

Table 2.0: Comparative Performance of Different Algorithms [27]

Algorithm	**Time taken**	**Number of overlapping**	**Average weight of the path**
Inogue/DA	0.014 sec	15	243
Inagaki	0.182 sec	5	275
GA	0.177 sec	2	257

Above all, from the experiments conducted it is proven that simultaneous multiple route selection is difficult by popular optimization technique like DA. On the other hand, GA can be efficiently used for finding out multiple routes simultaneously with minimal overlapping by grouping m routes as one set of solution and designing fitness functions in such a way that it penalizes the function of overlapping.

2.4.2 Evaluation of Route Finding Methods in GIS Application [29]

Roozbeh Shad, Hamid Ebadi and Mohsen Ghods performed an evaluation of route finding methods in GIS applications. The processing time of DA, Heuristic Algorithm (HA) which is divided into graph growth algorithm implemented with two queues and DA implemented with double buckets as well as GA using Visual Basic, MapObject and Visual C++ programing languages was done.

Basically, DA [30] for computing the shortest path is based upon the calculation of values in three one dimensional arrays, each of size equal to the number of nodes in the network. Each row of each array corresponds to one of the nodes of the network. As the algorithm proceeds, paths are calculated from the start node to other nodes in the network, paths are compared and the best paths are chosen, given the state of knowledge of the network at that stage in the progress of the algorithm.

On the other hand, there are different heuristic and probabilistic methods that no always guarantee the optimal solution but are able to find a possible solution space, taking advantage of the particular attributes of the target problem. In a recent study, a set of two shortest path algorithms that run fastest on real road networks has been identified. These two algorithms are the graph growth algorithm implemented with two queues and DA implemented with double buckets [31].

In this context, GA is a metaheuristic technique [32] that could provides robust tools with optimal or quasi-optimal designing and programming of transportation networks and node locating. Because its excellent flexibility, robustness and adaptability characteristics, GA has been successfully applied in the non-linear and complex optimization problem solutions, and also it is much appropriated to face the noisy combinatorial problems associated to the real systems optimization and transportation networks.

The four algorithms were implemented using data sets with different number of nodes and links under three different condition namely one-to-one, one-to-all and all-to-all. The result of timing for each algorithm is shown as the following:

Table 2.1: Execution Times of Algorithms with One to One Condition [29]

Number of nodes	DA	HA		GA
		Graph Growth	DA with double buckets	
500	0.38 (sec)	0.42	0.41	0.92
1000	3.48	3.78	3.21	4.78
2000	12.23	11.22	10.56	14.6
3000	38.74	29.89	27.43	35.43
4000	50.23	44.65	41.23	53.34
5000	102.38	89.34	85.65	104.04

Table 2.2: Execution Times of Algorithms with One to All Condition [29]

Number of nodes	DA	HA		GA
		Graph Growth	DA with double buckets	
500	0.53 (sec)	0.59	0.61	0.58
1000	5.42	5.49	5.54	5.67
2000	17.45	15.23	15.58	18.23
3000	42.24	36.56	38.68	44.32
4000	50.23	44.65	47.23	48.34
5000	112.42	101.37	105.45	110.56

Table 2.3: Execution Times of Algorithms with All to All Condition [29]

Number of nodes	DA	HA		GA
		Graph Growth	DA with double buckets	
500	1.43 (sec)	0.63	0.67	0.59
1000	8.45	6.80	6.95	6.23
2000	32.96	18.23	19.05	18.83
3000	64.78	40.24	43.50	50.61
4000	104.89	52.76	57.87	73.04
5000	210.78	110.78	118.65	137.76

Based on the result shown, the evaluation of 4 shortest path algorithms using real road networks has identified that:

- Execution time of mentioned algorithms depends on the problem conditions and the number of nodes in the real road networks.
- When the number of nodes and the problem conditions increase, HA method performs better than others.
- For the small number of nodes and the complex problem conditions, GA performs better than others.
- For the small number of nodes and the easy problem conditions DA performs better than others.

2.5 Conclusions

2.5.1 Advantages of GA

As a high efficient search strategy for global optimization, GA demonstrates favorable performance on solving the combinatorial optimization problems. With comparing to traditional search algorithms, GA is able to automatically acquire and accumulate the necessary knowledge about the search space during its search process, and self-adaptively control the entire search process through random optimization technique. Therefore, it is more likely to obtain the global optimal solution without encountering the trouble of combinatorial explosion caused by disregarding the inherent knowledge within the search space. It has been used to solve combinatorial optimization problems and non-linear problems with complicated constraints or non-differentiable objective functions. Besides that, it also has the capability to provide multiple route selection simultaneously avoiding overlapping. It necessitates the application of GA into route finding algorithms.

2.5.2 Drawbacks of GA

Search in usual GA, based on neo-Darwinian evolutionary theory is conducted by crossover and mutation. Crossover is generally considered a robust search means. Offspring may inherit partial solutions without conflict from their parents, but no information to decide which genes are partial solutions is available. From a search strategic point of view this means that variables are randomly selected and then values which will be assigned to them are also randomly selected from genes contained within a population. Therefore search by crossover in GA can not be considered efficient. Furthermore, mutation is a random search method itself. When thinking about efficiency of global search that is the most important characteristic of GA; it is admitted that the rate of search is low, because GA stresses random search rather than directional search. However, the above problem can be considered as directly inherited from problems of the neo-Darwinian evolutionary theory.

CHAPTER 3
METHODOLOGY

3.1 Biological Background

3.1.1 Chromosomes

The underlying GA concept in used to solve the TSP problem involved biological background. There are a few terms need to be understood particularly chromosome which will represent the solutions of the TSP problem. All living organisms consist of cells. In each cell there is the same set of chromosomes. Chromosomes are strings of Deoxyribonucleic Acid (DNA) and serve as a model for the whole organism. A chromosome consists of genes, blocks of DNA. Each gene encodes a particular protein. Basically, it can be said that each gene encodes a trait, for example color of eyes. Possible settings for a trait (e.g. blue, brown) are called alleles. Each gene has its own position in the chromosome. This position is called locus [17].

Complete set of genetic material or all chromosomes is called genome. Particular set of genes in genome is called genotype. The genotype is with later development after birth base for the organism's phenotype, its physical and mental characteristics, such as eye color, intelligence and others [17].

3.1.2 Reproduction

During reproduction, recombination/crossover first occurs. Genes from parents combine to form a whole new chromosome. The newly created offspring can then be mutated. Mutation means that the elements of DNA are a bit changed. The changes are mainly caused by errors in copying genes from parents. The fitness of an organism is measured by success of the organism in its life or survival [17].

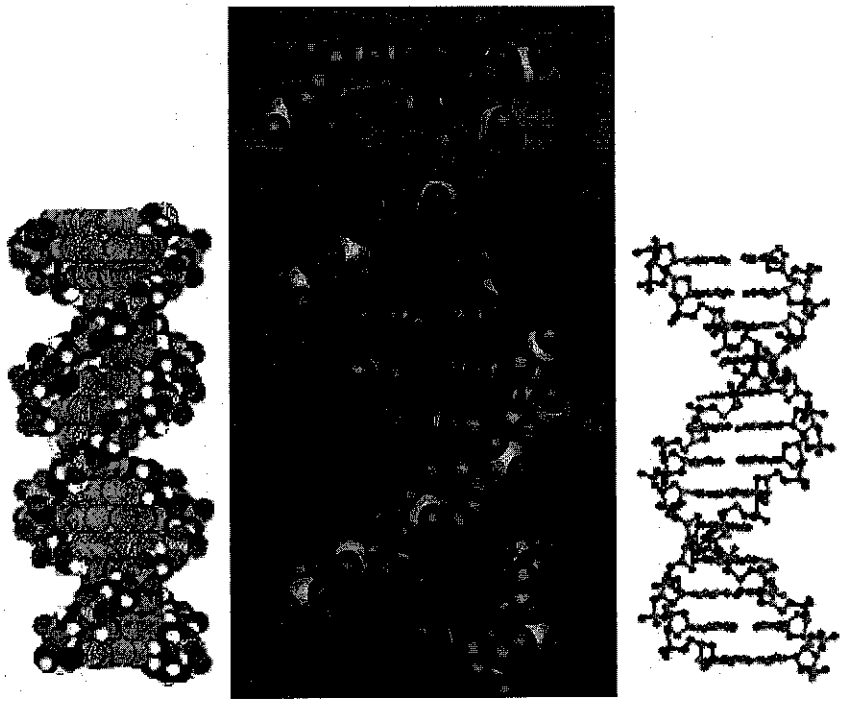

Figure 3.0: Deoxyribonucleic Acid (DNA) Structures [17]

3.2 Problem Area

Search Space

In solving the problem, some solutions which will be the best among others are looked for. The space of all feasible solutions or the set of solutions among which the desired solution resides is called search space or also state space. Each point in the search space represents one possible solution. Each possible solution can be "marked" by its value or fitness for the problem. With the implementation of GA, the best solution among a number of possible solutions is looked for by which represented by one point in the search space.

Looking for a solution is then equal to looking for some extreme value either minimum or maximum in the search space. At times the search space may be well defined, but usually only a few points in the search space are known. In the process of using GA, the process of finding solutions generates other points referring to the possible solutions as evolution proceeds.

3.3 System Concept

3.3.1 Basic Outline of GA

The implementation of the algorithm begins with a set of solutions represented by chromosomes called population. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are then selected to form new solutions; the offspring are selected according to their fitness. The more suitable they are the more chances they have to reproduce. This is repeated until some condition, for example number of populations or improvement of the best solution is satisfied. The basic outline of GA used can be depicted in Figure 3.1.

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in the new population
4. **[Replace]** Use new generated population for a further run of the algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

Figure 3.1: The Basic Outline of Genetic Algorithm (GA) [22]

3.3.2 Operators of GA

The two basic operators of GA namely crossover and mutation are the most important parts of GA. The performance is influenced mainly by these two operators. However, before addressing crossover and mutation, it is needed to first know how to create chromosomes and what type of encoding to choose. The type and implementation of these GA operators depends on the encoding and also on the problem.

Crossover

After what encoding to be used is already decided, then it can proceed to crossover operation. Crossover operates on selected genes from parent chromosomes and creates new offspring. The simplest way how to do that is to choose randomly some crossover point and copy everything before this point from the first parent and then copy everything after the crossover point from the other parent. Crossover can be illustrated as follows whereby (| is the crossover point) [22].

Table 3.0: The Illustration of Crossover [22]

Chromosome 1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

There are other ways how to make crossover, for example by choosing more crossover points. Crossover can be quite complicated and depends mainly on the encoding of chromosomes. Specific crossover made for a specific problem can improve performance of the GA.

Mutation

After a crossover is performed, mutation takes place. Mutation is intended to prevent falling of all solutions in the population into a local optimum of the solved problem. Mutation operation randomly changes the offspring resulted from crossover. In case of binary encoding it can switch a few randomly chosen bits from 1 to 0 or from 0 to 1 [22].

Table 3.1: The Illustration of Mutation [22]

Original offspring 1	1101111000011110
Original offspring 2	1101100100110110
Mutated offspring 1	1100111000011110
Mutated offspring 2	1101101100110110

The technique of mutation as well as crossover depends mainly on the encoding of chromosomes. For example when encoding permutations is used, mutation could be performed as an exchange of two genes.

3.3.3 Parameters of GA

The outline of the basic GA is very general. Therefore, many parameters and settings can be implemented differently in various problems. The two basic parameters of GA are crossover probability and mutation probability.

Crossover Probability

Crossover probability is on how often crossover will be performed. If there is no crossover, offspring are exact copies of parents. If there is crossover, offspring are made from parts of both parent's chromosome. If crossover probability is 100%, then all offspring are made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population. However, this does not mean that the new generation is the same. Crossover is made in hope that new chromosomes will contain good parts of old chromosomes and therefore the new chromosomes will be better. However, it is good to leave some part of old population survives to next generation [17].

Mutation Probability

Mutation probability is on how often parts of chromosome will be mutated. If there is no mutation, offspring are generated immediately after crossover or directly copied without any change. If mutation is performed, one or more parts of a chromosome are changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to random search [17].

Other Parameters:

Population Size

Population size is how many chromosomes are in population in one generation. If there are too few chromosomes, GA has few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if there are too many chromosomes, GA slows down. Research shows that after some limit which depends mainly on encoding and the problem it is not useful to use very large populations because it does not solve the problem faster than moderate sized populations [17].

3.4 System Implementation

3.4.1 Encoding of Chromosomes

A chromosome should in some way contain information about solution that it represents. The most used way of encoding is a binary string. A chromosome then could look like in Table 3.2.

Table 3.2: The Illustration of Chromosomes

Chromosome 1	1101100100110110
Chromosome 2	1101111000011110

Each chromosome is represented by a binary string. Each bit in the string can represent some characteristics of the solution. Another possibility is that the whole string can represent a number. Of course, there are many other ways of encoding. The encoding depends mainly on the solved problem.

Permutation Encoding

Among various encoding techniques namely Binary Encoding, Value Encoding, Tree Encoding and Permutation Encoding (PE), PE is chosen in this project to solve the TSP problem as it is useful for ordering problems. In PE, every chromosome is a string of numbers that represent a position in a sequence. The chromosome describes the order of cities, in which the salesman will visit them.

Table 3.3: The Illustration of Chromosomes with Permutation Encoding (PE)

Chromosome A	1 5 3 2 6 4 7 9 8
Chromosome B	8 5 6 7 2 3 1 4 9

Generate Random Initial Population of Chromosomes

With the use of GA, the system is capable of building a list of chromosomes representing a population. After the type of encoding technique to be used is determined, the initial population of chromosomes will be generated. Here, the population size value is crucial and need to be specified. Due to the point, a random initial population of tours will be generated according to the specified number of the population size. The flow of generating random initial population of chromosomes can be depicted from Figure 3.2.

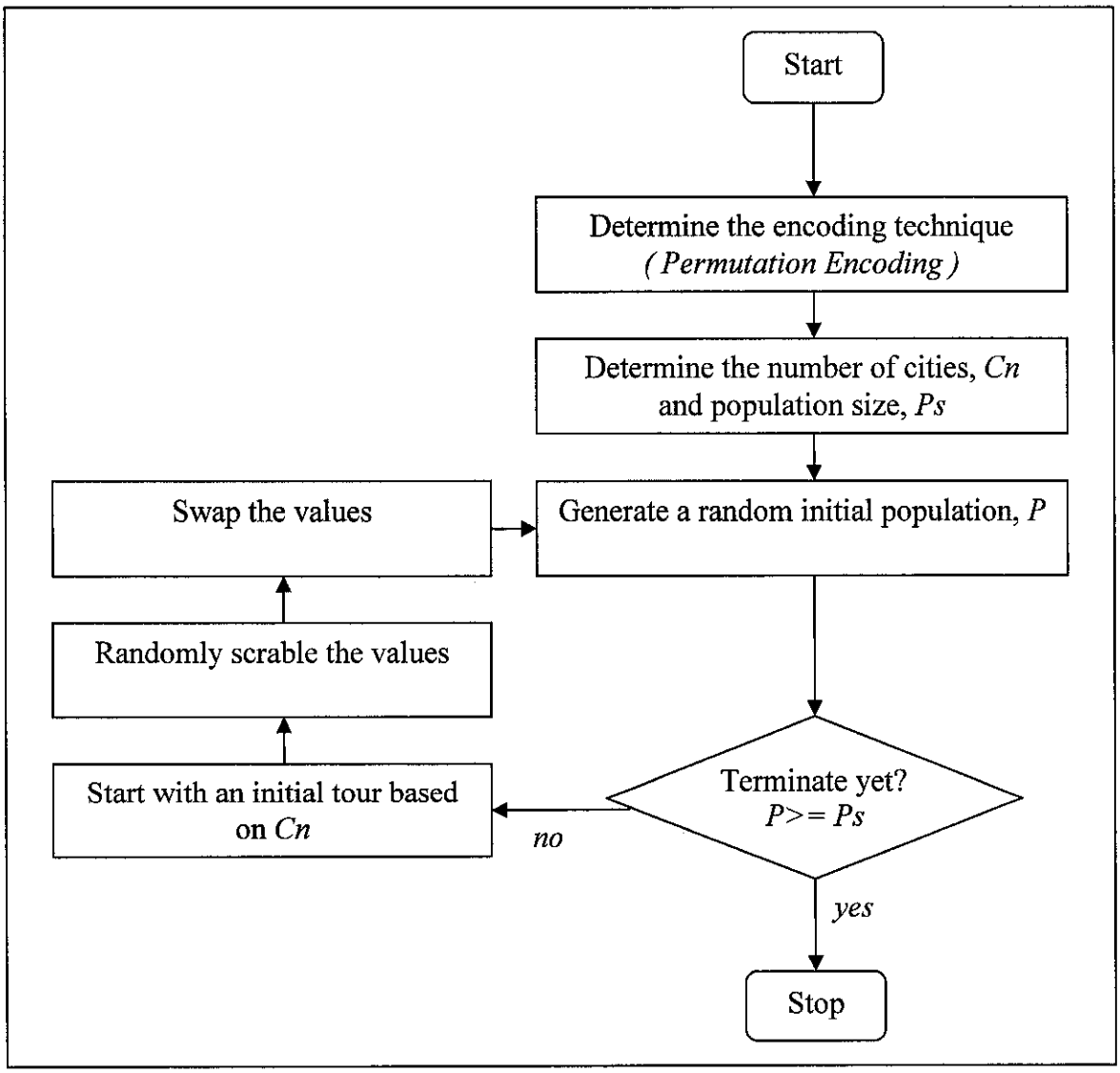

Figure 3.2: The Flow of Generating Random Initial Population

3.4.2 Evaluate the Fitness

After the population is already generated, the fitness of each chromosome in the population is evaluated. In this project, the fitness is determined by the length of distance. To do that, the value of each tour length distance is first calculated and added up. The values obtained are then used to compare how well one chromosome/member of the population stands against other chromosomes/members of the population. The process involved can be depicted in Figure 3.3.

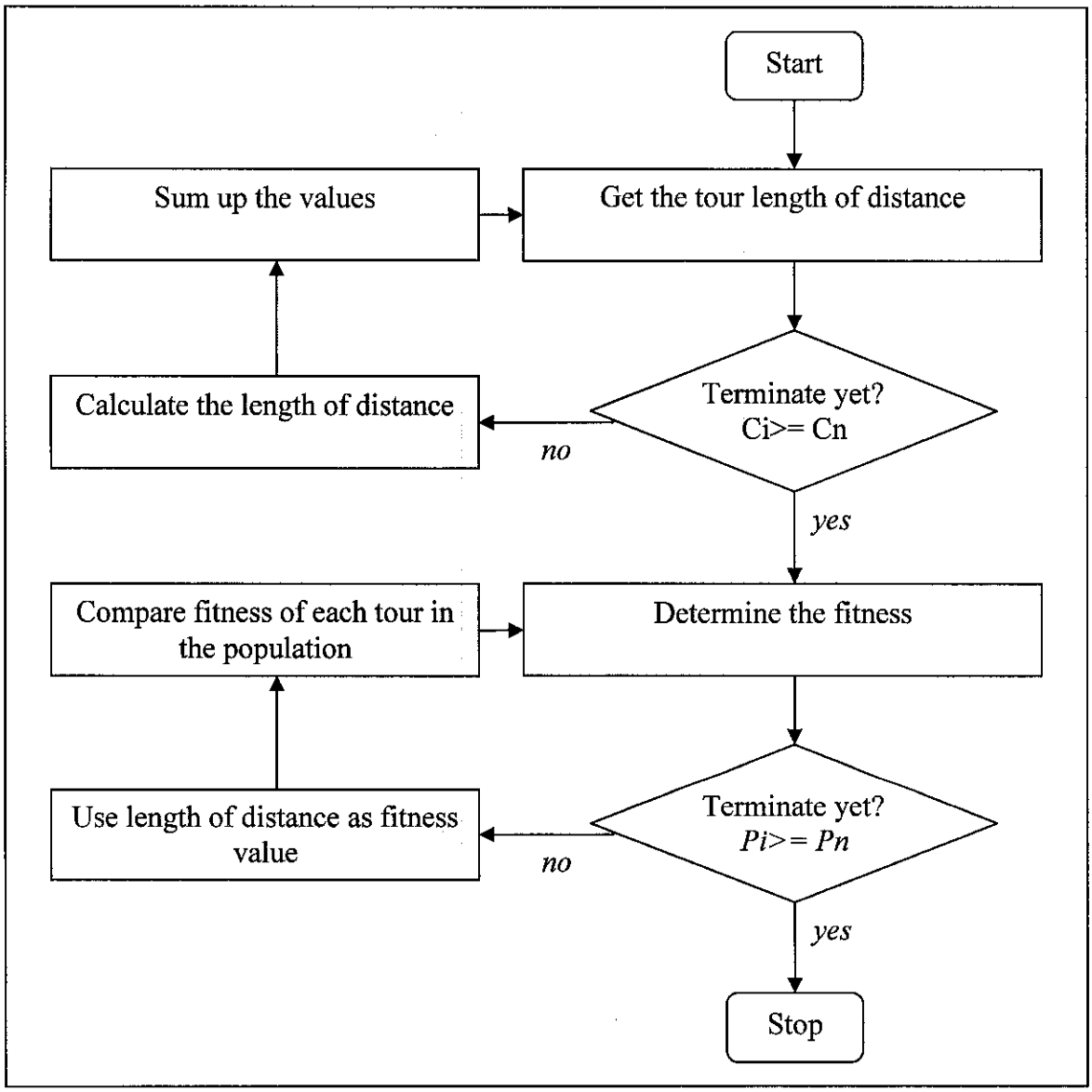

Figure 3.3: The Flow of Evaluating the Fitness

3.4.3 Generate New Population

Create the Child

Next, for the optimization purpose the new population will be generated. The process is done by creating the child referring to new chromosomes/members of the population. In order to create the child, selection and recombination/ crossover will be performed as the following:

- ***Selection of Parents***

  First is selecting parents for crossover. The main idea is to select the better parents (best survivors) in the hope that the better parents will produce better offspring. In this project, the parents are randomly chosen.

- ***Crossover***

  Next, the crossover points are randomly generated. In this case, the child carries with the traits exploited from the parents chosen from with the population. In this project, the crossover probability is set to null.

Sort the Population

Once the child has been created and inserted into the bottom of the population, a sorting of the population occurs. The sorting uses the fitness to determine where chromosomes/members of population are placed. The chromosomes/members will be ranked where the best fitness chromosomes/members will be at the top followed by the least fitness chromosomes/members.

Throw the Most Unfit Chromosome

Lastly, based on the fitness evaluated, the least fit chromosome/member will be thrown out from the population list. The idea is definitely to have a better list of population by inserting the new child replacing the unfit chromosome/member in the population.

Mutation

Mutation is introduced within the loop in order to bring new and random chromosomes/members to the population. Mutation is the process of randomly selecting chromosomes/members of the population can change their traits randomly in order to produce tour that has some traits new to the population. This keeps the population from converging too quickly upon a solution when there are much better solutions to be found. In essence, mutation keeps the search within the search space broad enough to encounter a feasible solution. In this project, two types of mutation method are available to be chosen. The idea is to compare the effectiveness of both methods in reaching the optimum result. The mutation methods provided are as follows:

- ***Swapping Method***

In swapping method, order changing is used whereby two numbers are selected and exchanged as depicted in Figure 3.4.

(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Figure 3.4: The Illustration of Swapping Method

- ***Inversion Method***

Where else in inversion method, the order of numbers will be inverted. This can be depicted in Figure 3.5.

(1 2 3 4 5 6 8 9 7) => (7 9 8 6 5 4 3 2 1)

Figure 3.5: The Illustration of Inversion Method

Evaluate the Fitness of New Tour

After the population is mutated according to the mutation rate defined by the user, the fitness of the new tour generated is evaluated. Lastly, once again the population will be sorted accordingly. The full process involved in generating new population can be depicted in Figure 3.6. The process loops until the maximum number of iterations is reached or the convergence occurred.

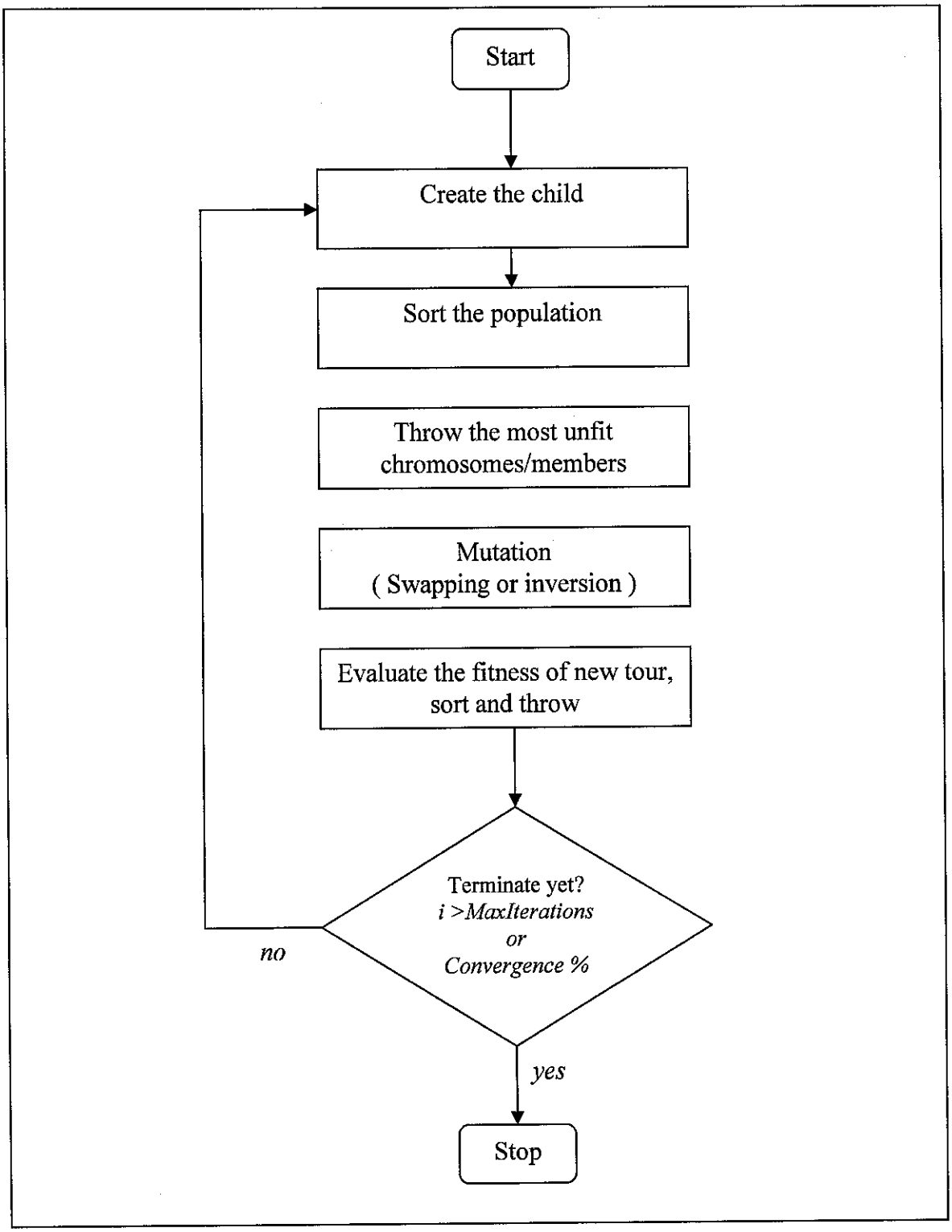

Figure 3.6: The Flow of Generating New Population

3.4.4 Save the Best Population

The best of population will be saved and then used to represent the best tour found by the GA. The algorithm loops through this process of spawning a child, replaces the weakness chromosome/member and then sorted. The detail process involved in achieving the optimize tour among all used in this project can be summarized as in Figure 3.7.

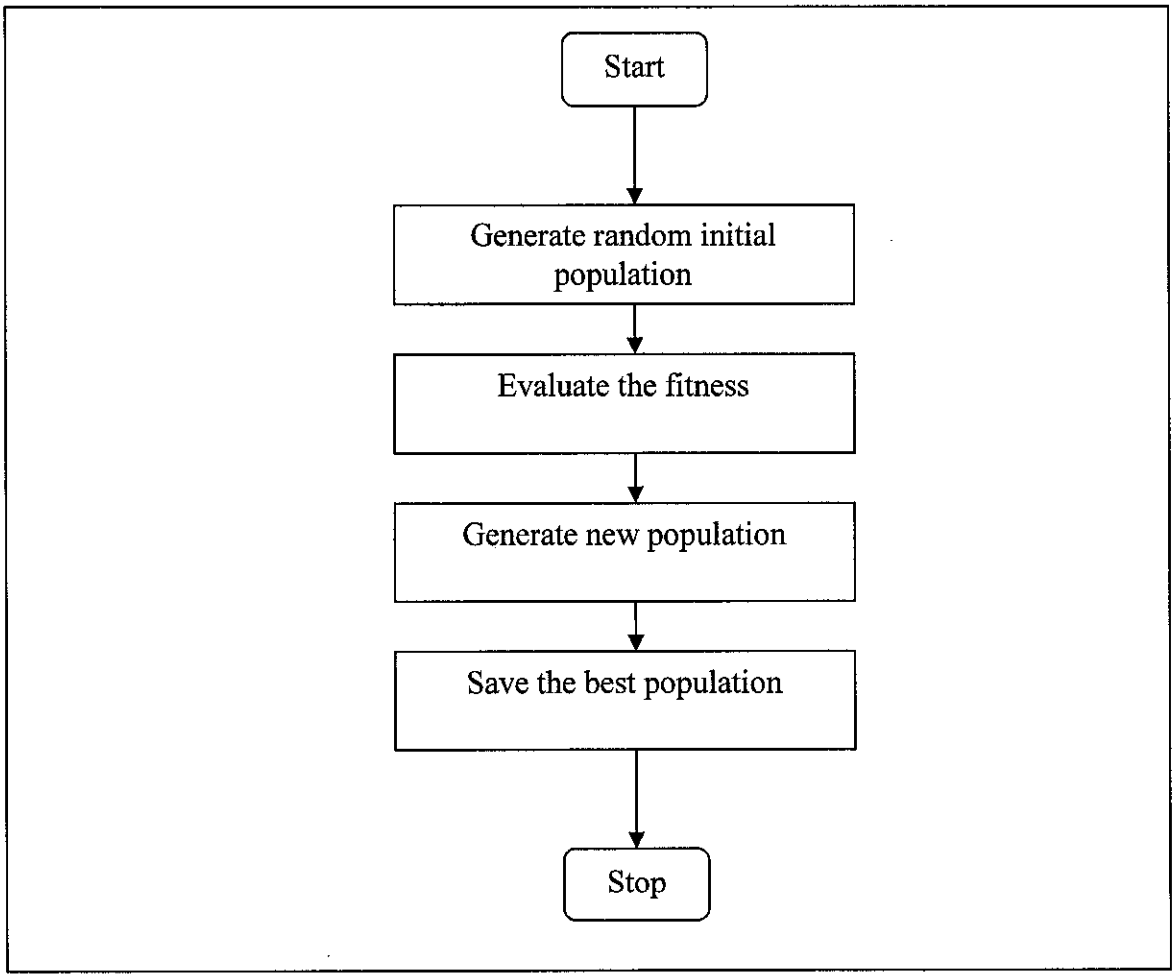

Figure 3.7: The Flow of Achieving Optimum Tour

3.4.5 Parameters

There are six user-defined parameters provided by the system as below:

- ***Maximum number of iterations***

  This will give the program a cut off point of when to quit by limiting the number iterations it is to loop for.

- ***Mutation rate***

  This is the percentage at which the every member of the population has a chance of being mutated.

- ***Display every Xth iteration***

  This is a display factor. Due to the point that GA solution for the TSP problem runs faster that screen can update, the solution will only be displayed every other iteration.

- ***Population size***

  This is obviously the size of the population used within the GA. This will determine the number of tours to be generated randomly.

- ***Convergence percentage***

  This will give the program a way to be able to determine if an algorithm has converged upon a solution to the problem. In this case, the system will be terminated not only when the maximum number of iterations is reached, but also when the population shows very little change from one generation.

- ***Number of cities***

  This will give the user flexibility to set the number of cities to be visited. Based on the number of cities the tour will be generated.

3.5 System Prototype

3.5.1 Run the System

The system prototype will be in a form of Java Applet. To run the system, at least JAVA version 2 or later is needed. The system will start running once the start button was pressed. The system will search for the optimum path according to the parameters that already had been set by the user. The status will indicate whether the system is still running or not. The system will be running as long as the currently running iteration is not exceeding the maximum iterations or the convergence percentage; unless the stop button was pressed to terminate the program intentionally.

3.5.2 Display the Result

After the entire process is completed, the system prototype will display the following result:

- ***The Best Path***

  This is used to display the best length of distance which is the shortest path after each of the iteration.

- ***The Worst Path***

  This is used to display the worst length of distance which is the longest path after each of the iteration.

- ***Final Report***

  This is used to display the city locations as well as the final tour which is the shortest path among all together with its length of distance after the maximum iteration is reached.

- ***Path Canvas***

  This is used to display the city locations as well as the simulation route to be taken graphically. The city locations can be displayed in two different forms as follows:

  1. **Radial city locations**

     - In radial city locations, the cities will be located in circle form.

  2. **Random city locations**

     - In random city locations, the cities will be located randomly in the canvas area.

3.6 Tools Required

The tools required in completing this project are as follows:

3.6.1 Hardware

- Processor: Intel Pentium-4 1.5GHz
- Memory: 256MB RAM
- Disk Storage: 30GB

3.6.2 Software

- Window XP Platform
- JAVA 2 Standard Development Kit 1.4.2_08
  – j2sdk-1.4.2_08-windows-i586-p
- Forte for JAVA
  – Community Edition

CHAPTER 4
RESULT AND DISCUSSION

4.1 System Testing

In this project, experiments were conducted to measure the effectiveness of system performance in achieving optimum result. The objective is to compare the two different methods of mutation operator namely swapping method and inversion method. The values of parameters for this experiment are set as follows:

Table 4.0: The Value of Parameters for Experiments

Parameters	Value	Justifications
Maximum number of iterations	10,000	High number of iterations is used to see the pattern of performance. It can be up to 30,000 but it will slow down the performance. A faster machine is needed
Display every Xth iteration	500	This will make the output to be displayed 21 times which is still considerable for a slower machine
Population Size	100	Moderate sized population is used because too few chromosomes will cause few possibilities to perform crossover and only a small part of search space is explored. If there are too many chromosomes, GA will slow down
Converge Percentage	-1	The converge is "off" to let the system iterates until the maximum number of iterations.

Number of cities	20	Moderate number of cities is used to test the system because if it is too small, there is no much chromosome can be generated for the population
Mutation rate (%)	0, 1, 10	The idea of choosing no mutation rate (0%), low value of mutation (1%) as well as high value of mutation (10%) is merely to see the effect of mutation rate on result and performance.

The experiments will be categorized according to the type of city locations and the results will be analyzed.

4.1.1 Experiment 1 (Radial City Locations)

4.1.1.1 Mutation 1 (Swapping Method)

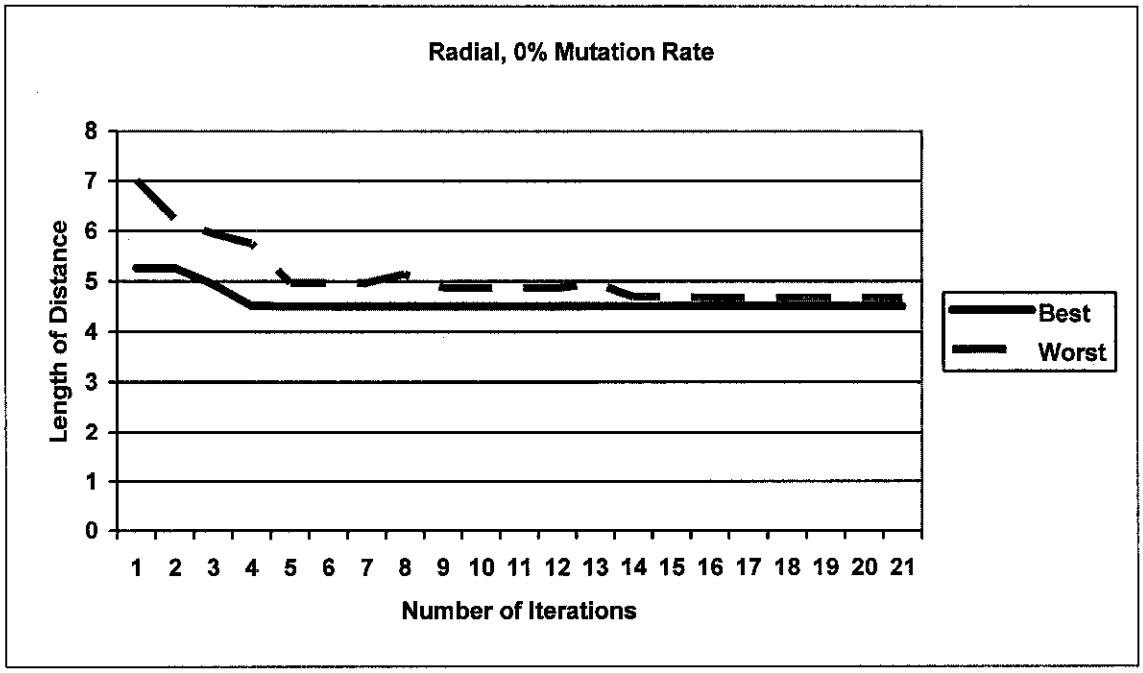

Figure 4.0: The Graph of Radial, 0% Mutation Rate for Mutation 1

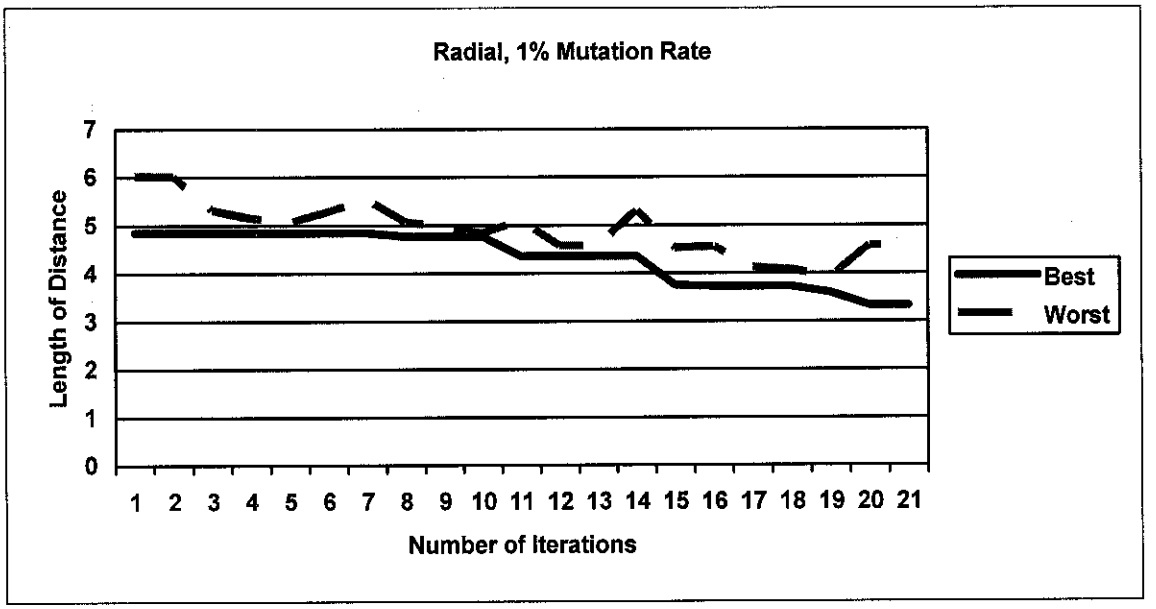

Figure 4.1: The Graph of Radial, 1% Mutation Rate for Mutation 1

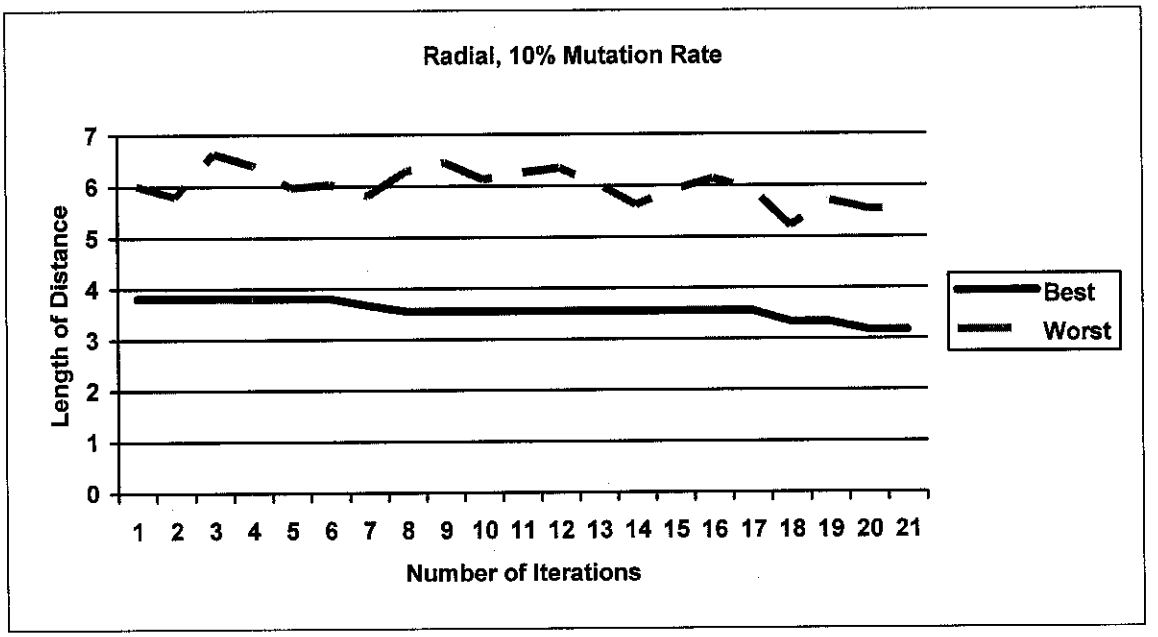

Figure 4.2: The Graph of Radial, 10% Mutation Rate for Mutation 1

4.1.1.2 Mutation 2 (Inversion Method)

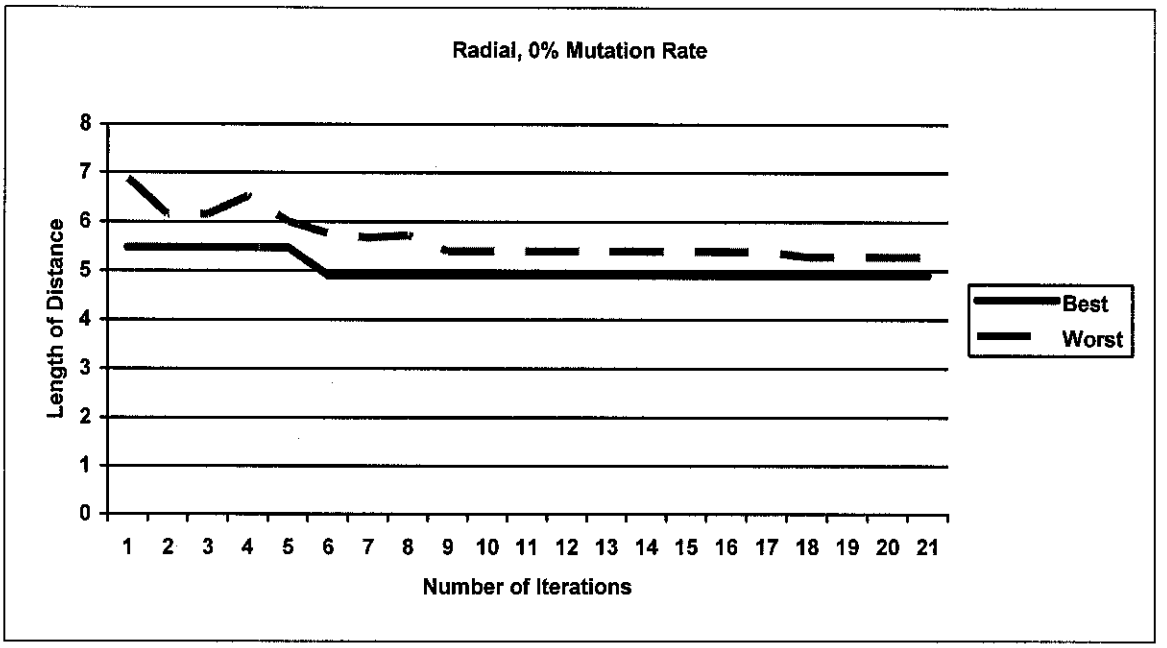

Figure 4.3: The Graph of Radial, 0% Mutation Rate for Mutation 2

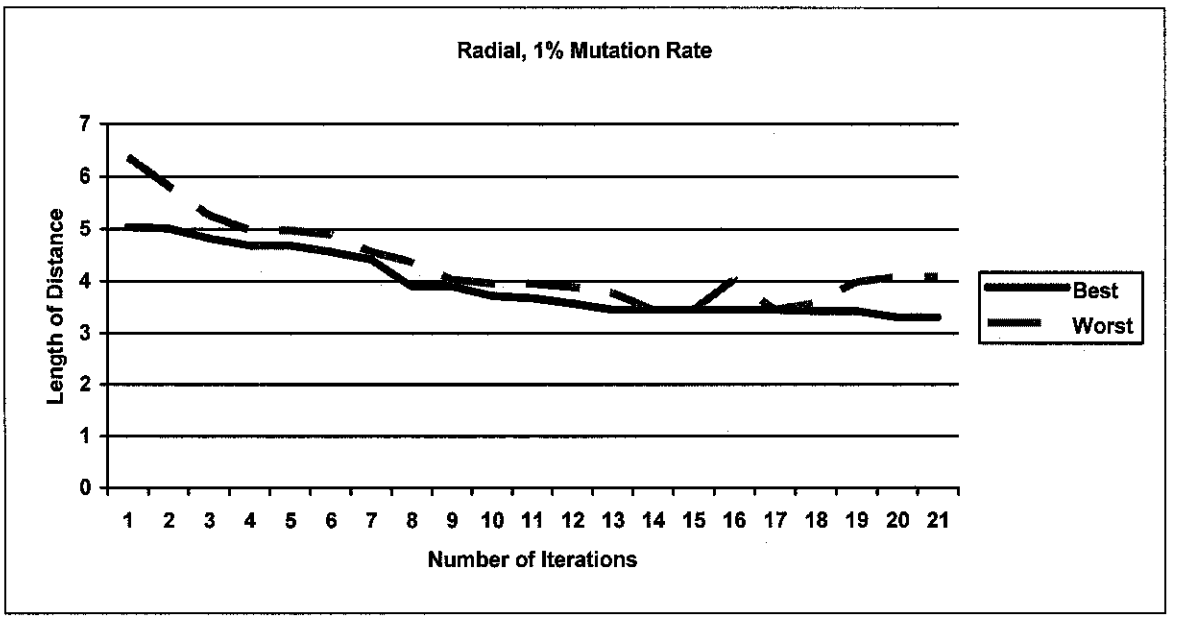

Figure 4.4: The Graph of Radial, 1% Mutation Rate for Mutation 2

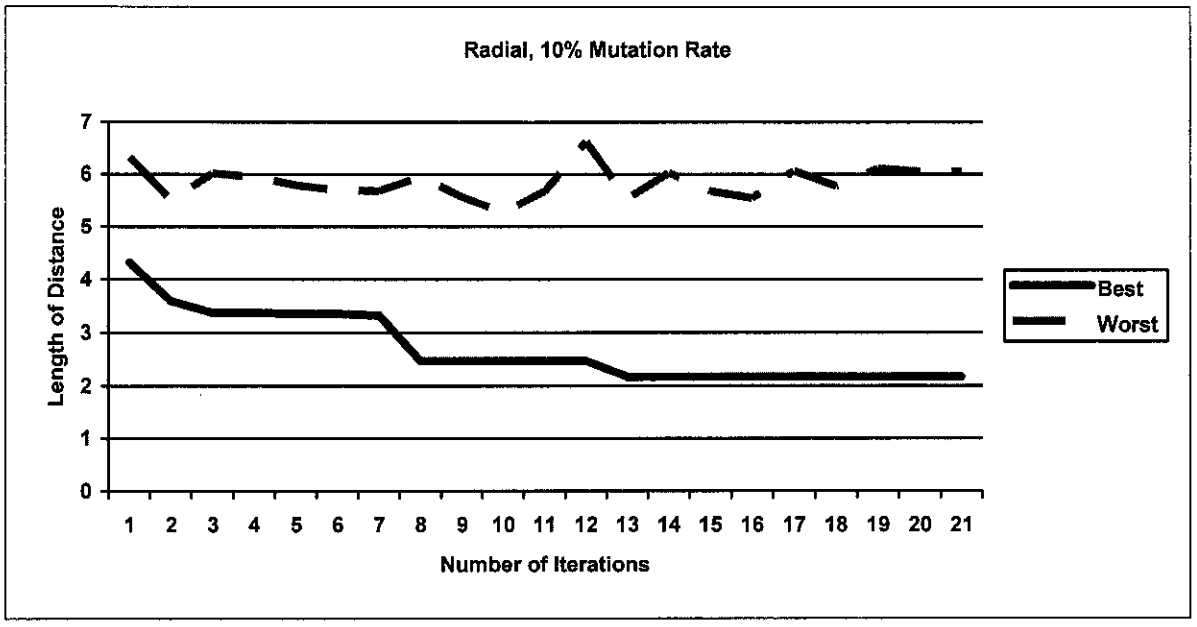

Figure 4.5: The Graph of Radial, 10% Mutation Rate for Mutation 2

4.1.2 Experiment 2 (Random City Locations)

4.1.2.1 Mutation 1 (Swapping Method)

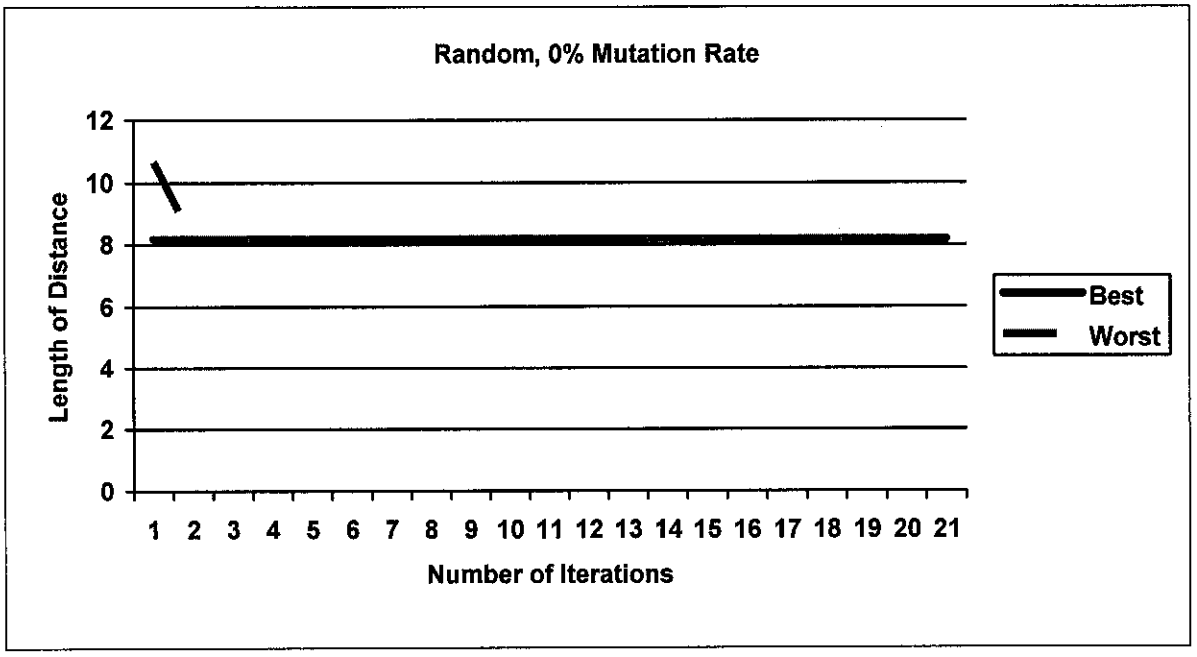

Figure 4.6: The Graph of Random, 0% Mutation Rate for Mutation 1

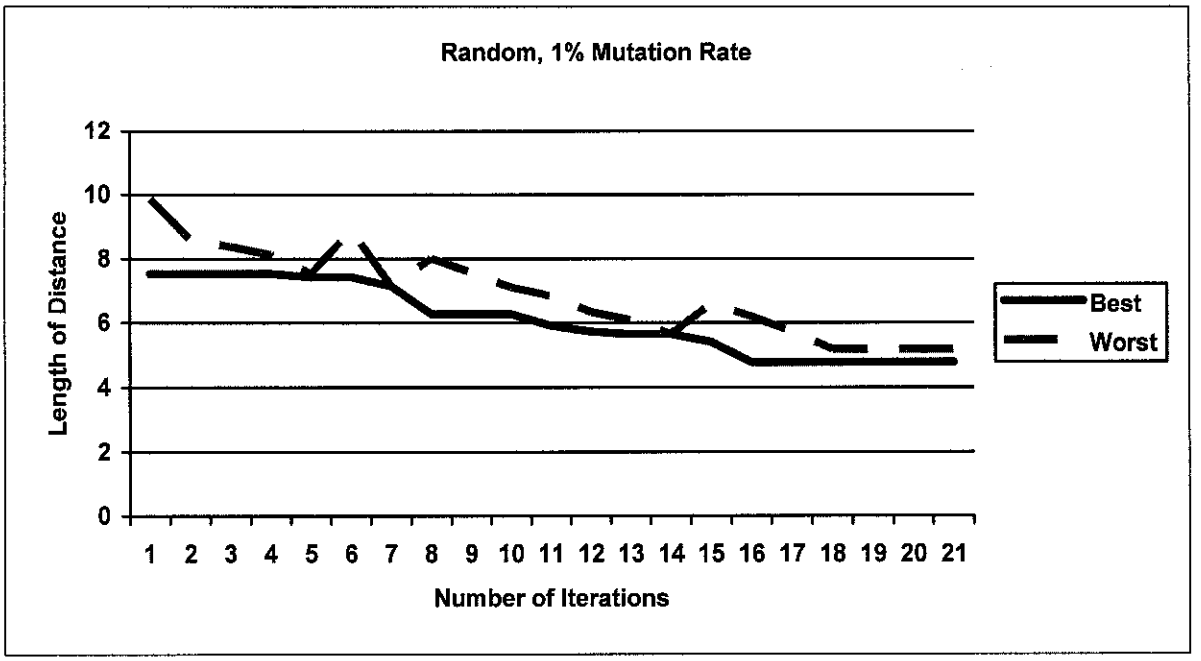

Figure 4.7: The Graph of Random, 1% Mutation Rate for Mutation 1

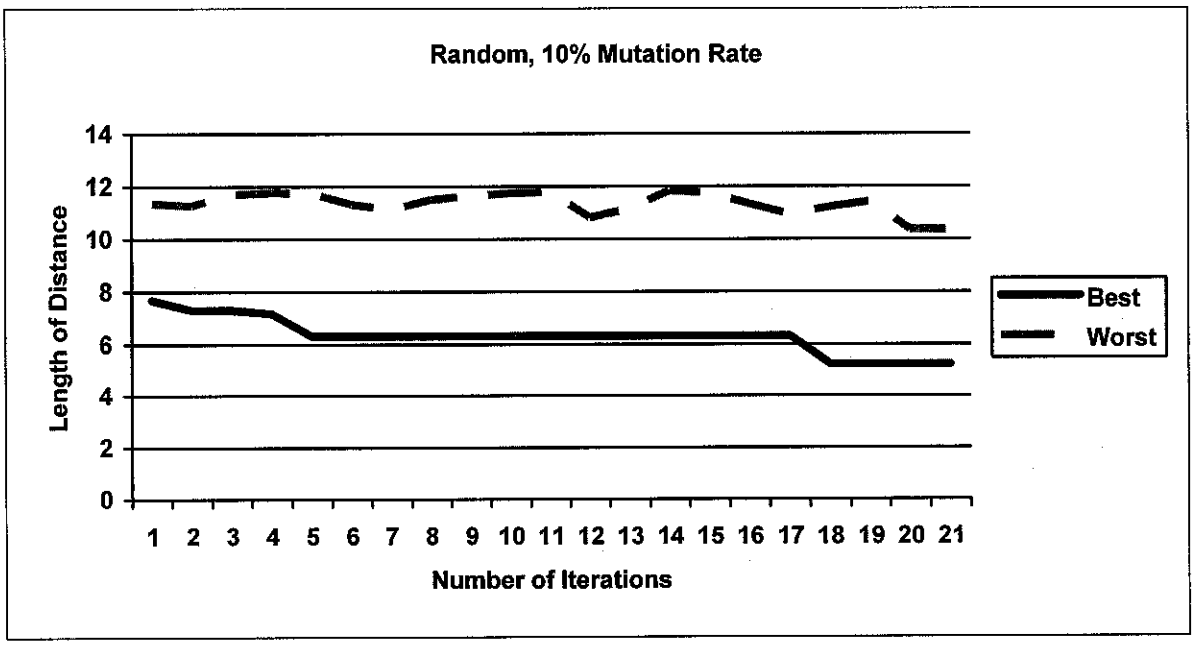

Figure 4.8: The Graph of Random, 10% Mutation Rate for Mutation 1

4.1.2.2 Mutation 2 (Inversion Method)

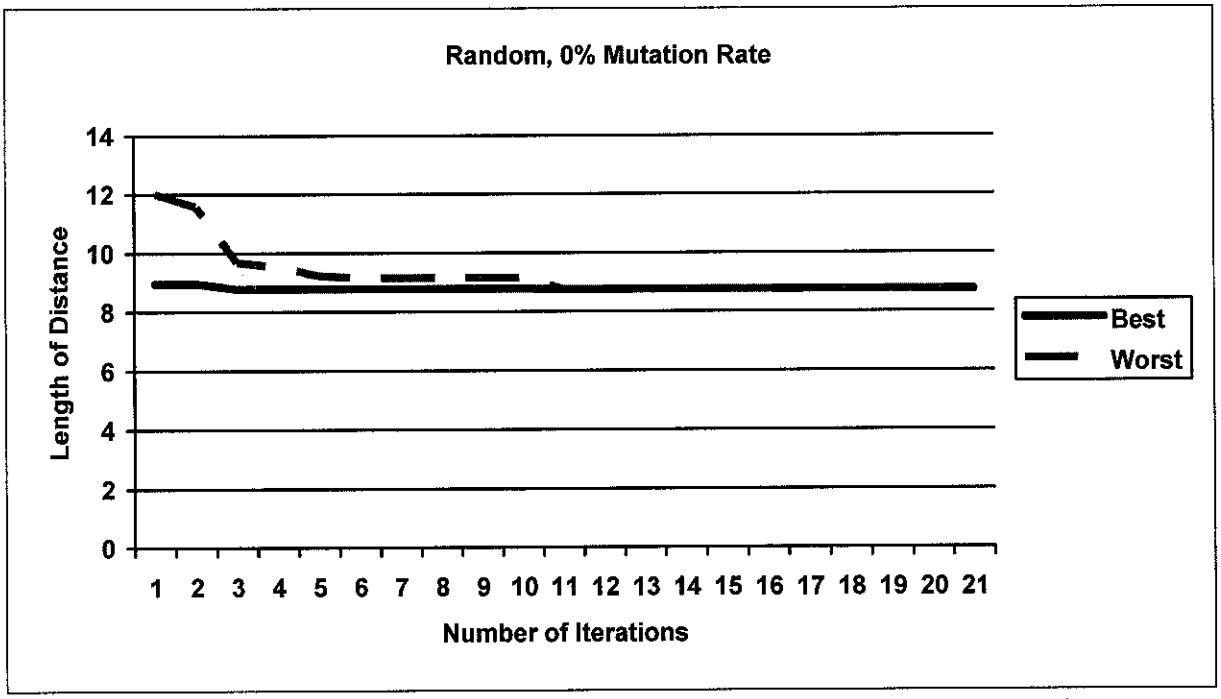

Figure 4.9: The Graph of Random, 0% Mutation Rate for Mutation 2

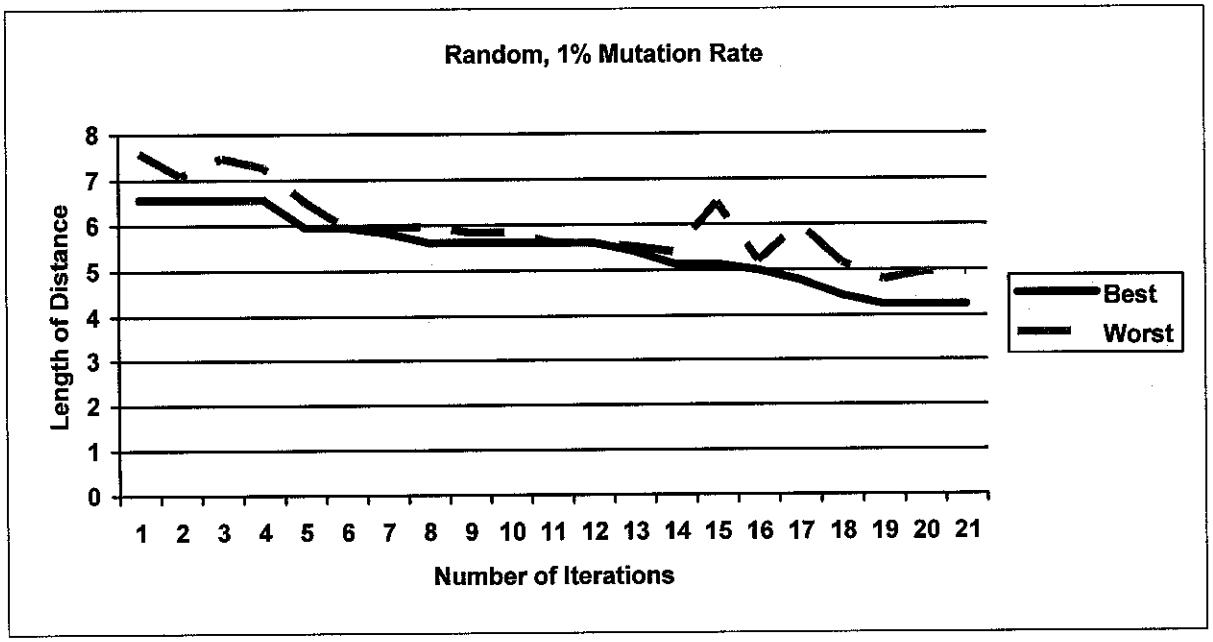

Figure 4.10: The Graph of Random, 1% Mutation Rate for Mutation 2

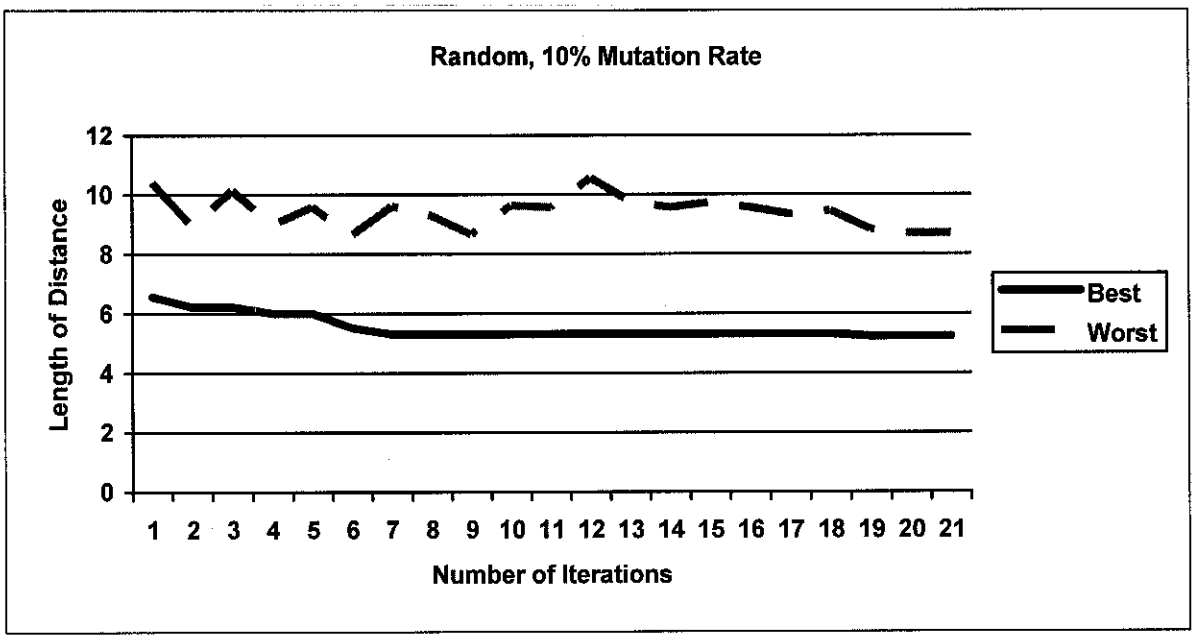

Figure 4.11: The Graph of Random, 10% Mutation Rate for Mutation 2

In the subsequent section, the result gained from Experiment 1 and Experiment 2 will be summarized and discussed in further detail. The datasets gained will be analyzed to answer two major questions as the following:

1. What type of mutation method to be used?

2. What is the appropriate mutation rate to be used?

4.2 Results

From Experiment 1 conducted, the results of the best path (the shortest distance) and the worst path (the longest distance) obtained from Mutation 1 and Mutation 2 operation can be summarized as in Table 4.1.

Type	Mutation Rate (%)	Mutation 1 (Swapping Method)	Mutation 2 (Inversion Method)
Best	0.0	4.50787	4.90436
	0.01	3.32732	3.31069
	0.1	3.17689	2.16706
Worst	0.0	4.67287	5.28178
	0.01	4.58081	4.08983
	0.1	5.54012	6.05351

Table 4.1: The Comparison of Result for Experiment 1

Where as from Experiment 2, the results of the best path (the shortest distance) and the worst path (the longest distance) obtained from Mutation 1 and Mutation 2 operation can be summarized as in Table 4.2.

Type	Mutation Rate (%)	Mutation 1 (Swapping Method)	Mutation 2 (Inversion Method)
Best	0.0	8.18551	8.75471
	0.01	4.77022	4.25060
	0.1	5.25729	5.23935
Worst	0.0	8.18551	8.75471
	0.01	5.12152	4.95347
	0.1	10.36468	8.69953

Table 4.2: The Comparison of Result for Experiment 2

4.3 Discussions

4.3.1 Experiment 1 (Radial City Locations)

From the result obtained in Experiment 1, it can be clearly seen that for no mutation rate (0%); Mutation 1 performs better than Mutation 2 in achieving optimize result. However, when the mutation rate takes place, Mutation 2 performs better than Mutation 1. For low level of mutation rate (1%) as well as high level of mutation rate (10%); Mutation 2 outperforms Mutation 1.

In term of the level of mutation rate, it is clearly seen that when the mutation rate is increasing, the better optimum result is obtained. The difference between the best path and the worst path is apparently getting wider as well when the mutation rate is increased. Regardless of the type of city locations, the performance of Mutation 1 and Mutation 2 towards the mutation rate in used is the same.

4.3.1 Experiment 2 (Random City Locations)

The same thing goes to Experiment 2. From the result obtained, it can be clearly seen that for no mutation rate (0%); Mutation 1 performs better than Mutation 2 in achieving optimize result. However, when the mutation takes place, Mutation 2 performs better than Mutation 1. For low level of mutation rate (1%) as well as high level of mutation rate (10%); Mutation 2 outperforms Mutation 1.

In term of the level of mutation rate, it is also clearly seen that when the mutation rate is increasing, the better optimum result is obtained. The difference between the best path and the worst path is apparently getting wider as well when the mutation rate is increased. Regardless of the type of city locations, the performance of Mutation 1 and Mutation 2 towards the mutation rate in used is the same.

4.4 Observations

From the experiments also some observation on the other parameters used was made. It can be seen clearly from all the graphs that the optimum result tends to be better when the number of iteration is increasing. The graph is sloping down across the number of iterations. The optimize result among all is getting shorter across the iterations which is awesome.

CHAPTER 5
CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

This project was successfully developed to find the path optimization by using the implementation of GA. The system prototype simulate the optimize result referring to the shortest length of distance among all possible tours. To ensure the effectiveness of the system, GA itself was studied in depth together with the operators and parameters. In this project, mutation operator functionality was focused in detail. Therefore, experiments were conducted to measure the effectiveness of two different types of mutation method namely swapping method and inversion method. The comparison of both performances in achieving optimum result had been analyzed. The result can be concluded as follows:

1. When there is no mutation rate implemented, Mutation 1 (Swapping Method) is more effective than Mutation 2 (Inversion Method).

2. When there is mutation rate implemented, Mutation 2 (Inversion Method is more effective than Mutation 1 (Swapping Method) regardless of the level of mutation rate in used.

3. The higher the level of mutation rate in used, the better result of optimum solution is obtained.

4. The type of mutation and rate of mutation to be used are important in resolving optimization problem and need to be determined accordingly.

5. The type of city locations in used gives no effect at all to the performance of both Mutation 1 (Swapping Method) and Mutation 2 (Inversion Method).

Therefore, to optimize the optimum result, it is recommended to use Mutation 1 (Swapping Method) when no mutation rate takes place where else it is better to use Mutation 2 (Inversion Method) when there is mutation rate involved.

Other than that, some observation was also made on the other parameters used. From the observation, it is also can be concluded that:

1. The higher the numbers of iterations, the better result of optimum solution it would be.

2. The maximum number of iterations is also vital in achieving optimum result.

Hence, it is also recommended to increase the maximum number of iterations in achieving a fairly good result. However, a faster machine is required because higher number of maximum iterations used will lead to a longer time taken in searching for the optimize result. The performance of current machine used is Intel Pentium-4 1.5GHz.

In this case, it is overseen that GA operators particularly mutation as well as parameters particularly mutation rate and number of iterations play a vital role in GA function for optimization purpose. External factor which is the speed of machine performance need to be considered. For example, the application performs better when it runs on faster machine such as Pentium 4 processor compared to slower machine such as Celeron processor.

5.2 Limitations of Project

The limitation of this project is it only caters for the static constraint referring to the length of distance in achieving the optimize path. The project only focuses in finding the shortest distance of tour among all possible solutions. Other dynamic factors namely traffic congestion, type of transportation used as well as cost incurred are assumed to be constant. Besides that, the project is also focusing mainly on the back-end of the algorithm. It is not applied yet to the real world situations.

5.3 Recommendations

Future enhancements recommended for the system are as the following:

1. Inclusion of dynamic constraints as fitness function in evaluating the solutions. Due to the point that the shortest distance of tour is not necessarily the best when traffic congestion comes into the place. The time taken to complete the tour will definitely be longer resulting the tour as no more the optimize tour to be used. The shortest path is no more the fastest tour among all.

2. Incorporate the use of real data to increase the efficiency of the system. In this case, the use of real map data as problem area together with the simulation of optimize path on the map is an awesome output to the user. Furthermore, the system is focused mainly on the back-end of the algorithm.

REFERENCES

1. George F Luger, 2005, ***Artificial Intelligence***, England, Pearson Education Limited.

2. N. L. Biggs, E. K. LLoyd, and R. J. Wilson, 1976, ***Graph Theory 1736-1936***, Clarendon Press, Oxford.

3. ***The TSP Problem***, Retrieved on July 11, 2005, <http://www.zlote.jabluszko.net/tsp/>

4. Sushil J. L. and Rilun T., ***Interactive Genetic Algorithms for the Traveling Salesman Problem***, University of Nevada, Retrieved on July 12, 2005, <http://www.cse.unr.edu/~sushil/papers/conference/newpapers/99/gecco99/iga/GECCO/gecco/node3.html>

5. Learhoven, P. V. and Aarts, E. H. L., 1987, ***Simulated Annealing:The Theory and Application***, Kluwer Academic Publishers.

6. Kirkpatrick, S. and Toulouse, G., 1985, ***Configuration space analysis of traveling salesman problems***, In Journal de Physiqu 46(8):1277-1292.

7. Crowder, H. and Padberg, M. W., 1980, ***Solving large scale symmetric traveling salesman problems to optimality***, In Management Science, 26:495-509.

8. Aarts, E. H. L. and Stehouwer, H. P., 1993, ***Neural networks and the traveling salesman problem***, In Proc. Int. Conf. on Artifical Neural Networks, Spring Verlog.

9. Padberg, M. and Rinaldi, 1987, ***Optimization of a 532-city symmetric traveling salesman problem by branch and cut***, In Operations Research Letters 6(1):1-7.

10. Lin, S. and Kernighan, B., 1973, ***An effective heuristic algorithm for the traveling-salesman problem***, In Operations Research, 21(2):498-516.

11. Martin, O., Otto, S., and Felten, E., 1991, ***Large-step Markov chains for the traveling salesman problem***, In Complex Systems, 5(3):299-326.

12. Holland, J., 1975, ***Adaptation In Natural and Artificial Systems***, The University of Michigan Press, Ann Arbor.

13. Goldberg, D. and Lingle, R., 1985, ***Alleles, loci and the traveling salesman problem***, In Proceedings of the Second International Conference on Genetic Algorithms, Mahwah, NJ. Lawrence Eribaum Associate.

14. Davis, L., 1985, ***Job shop scheduling with genetic algorithms***, In Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Eribaum Associates, Mahwah, NJ.

15. Louis, S. J., 1993, ***Genetic algorithms as a computational tool for design***, In PhD thesis. Indiana University, Indiana University.

16. Grefenstette, J., Gopal, R., Rosmaita, R., and Gucht., D., 1985, ***Genetic algorithms for the traveling salesman problem***, In Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Eribaum Associates, Mahwah, NJ.

17. Oliver, I. M., Smith, D. J., and Holland, J. R., 1987, ***A study of permutation crossover operators on the traveling salesman problem***, In Proceedings of the Third International Conference on Genetic Algorithms, London, Lawrence Eribaum Associates.

18. Jog, P., Suh, J. Y., and Gucht, D. V., 1991, ***Parallel genetic algorithms applied to the traveling salesman problem***, In SIAM J. Optimization 1:515-529.

19. Whitley, D., Starkweather, T., and Fuquay, D., 1989, ***Scheduling problems and traveling salesman: The genetic edge recombination operator***, In Proceedings of the Third International Conference on Genetic Algorithms, Los Altos, CA:Morgan Kaufmann Pulbishers.

20. Wonil Kim, Chuleui Hong and Yeongjoon Kim, ***Asynchronous Distributed Genetic Algorithm for Optimal Channel Routing***, Sangmyung University, Seoul, Korea.

21. ***Travelling Salesman Problem Using Genetic Algorithms***, Retrieved on July 9, 2005, < http://lalena.com/ai/tsp/>.

22. Dave Todd, 2002, ***TSP Genetic Algorithm***, Retrieved on July 9, 2005, <http://ouray.cudenver.edu/~da0todd/neural/third_homework/dave/test/Applet.htm>

23. ***The GA Playground***, Retrieved on July 10, 2005, <http://www.aridolan.com/ga/gaa/gaa.html.>

24. Emanuel Falkenauer, 1999, ***Genetic Algorithms and Grouping Problems***, England, John Wiley & Sons Ltd.

25. Marek Obitko, ***Introduction to Genetic Algorithms***, Retrieved on July 10, 2005, <http://cs.felk.cwt.cz/~xobitko/ga/.>

26. S.Hsiung and J.Matthews, ***Generation 5 - Genetic Algorithms and Genetic Programming***, Retrieved on July 10, 2005, <http://www.generation5.org/ content/2000/ga.asp/>

27. Basabi Chakraborty, ***GA-Based Multiple Route Selection for Car Navigation***, Iwate Prefectural University, Japan

28. M. R. Delavar, F. Samadzadegan, P. Pahlavani, ***A GIS-Assisted Optimal Urban Route Finding Approach Based On Genetic Algorithms***, University of Tehran, Iran, Retrieved on July 10, 2005, <http://www. isprs.org/istanbul2004/comm2/papers/144.pdf ->

29. Roozbeh Shad, Hamid Ebadi, Mohsen Ghods, ***Evaluation of Route Finding Methods in GIS Application***, University of Technology, Iran, Retrieved on July 15, 2005, <http://www.gisdevelopment.net/technology/gis/ma03202c.htm>

30. Dijkstra, E. W. , 1959, ***A Note on Two Problems in Connection with Graphs***, Numeriche Mathematik, 1:269-271

31. Zhan, F. B., and Noon, C. E., 1996, ***Shortest Path Algorithms: An Evaluation Using Real Road Networks***, Transportation Science (in press).

32. Diaz A., 1996, ***Optimization Hueristic Algorithms***, Madrid, 29-46

33. Golden B., 1976, ***Shortest-Path Algorithms- A comparison, Operations Research***, Vol.24, No. 9, 1164-1168.

34. Nakahara H., Sagawa T. and Fuke T., 1986, ***Virus theory of evolution***, Bulletin of Yamanashi Medical College, Vol.3, 14-18.

35. Satoh, H., Yamamura M. and Kobayashi S., 1996, ***Minimal Generation Gap Model for GAs Considering Both Exploration and Exploitation***, Proc. IIZUKA '96, 494-497.

36. Inagaki, J. 2002, ***A method of Determining Various Solutions for Routing Application with a Genetic Algorithm***, in Trans of IEICE, J82-D-I, No. 8, 11 02-1111 (in Japanese).

37. Inoue, Y., 2001, ***Exploration Method of Various Routes with Genetic Algorithm***, Master's Thesis, Information System Engineering, Kochi Institute of Technology

APPENDICES

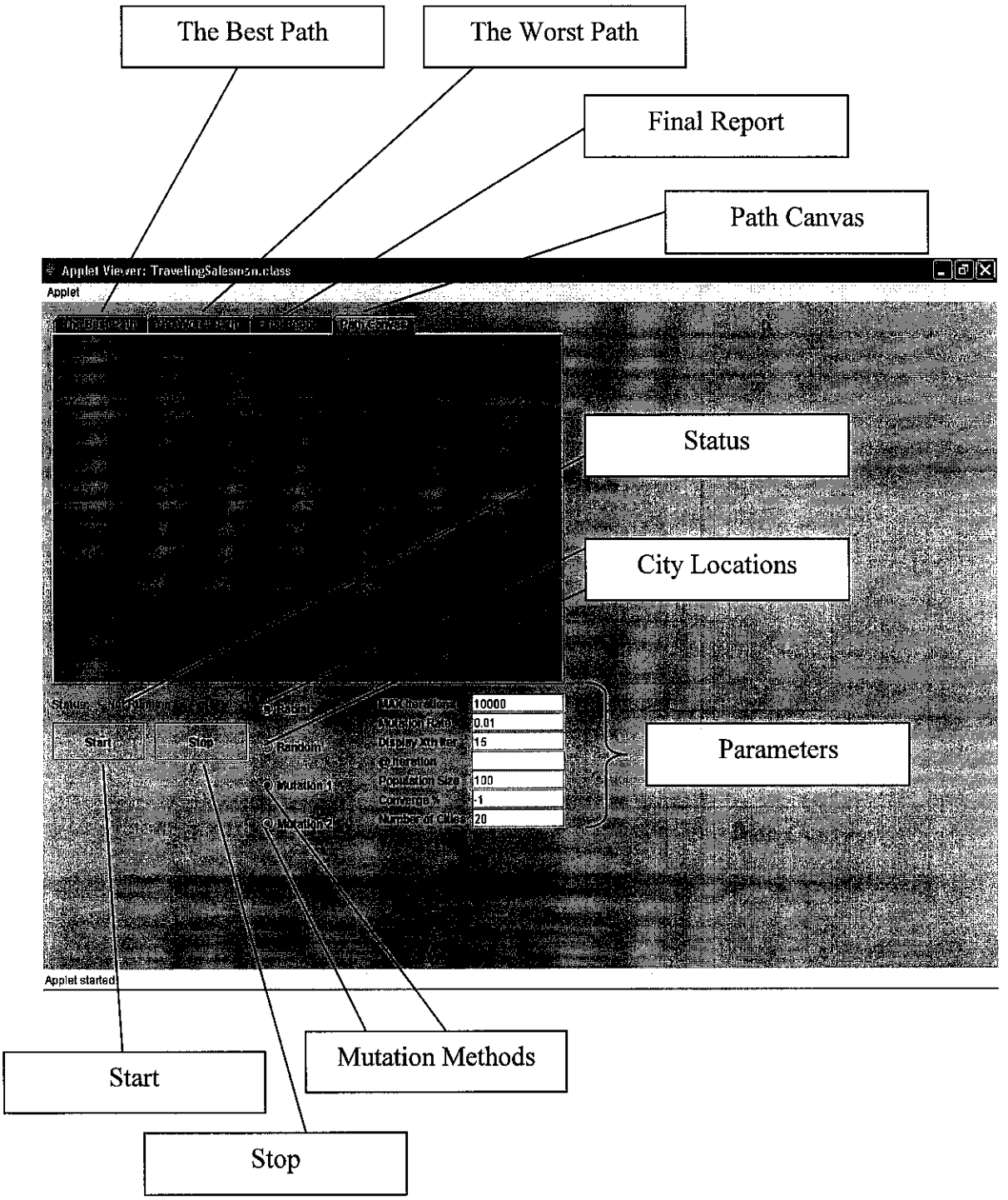

Figure: Graphical User Interface (GUI) of Simulated System

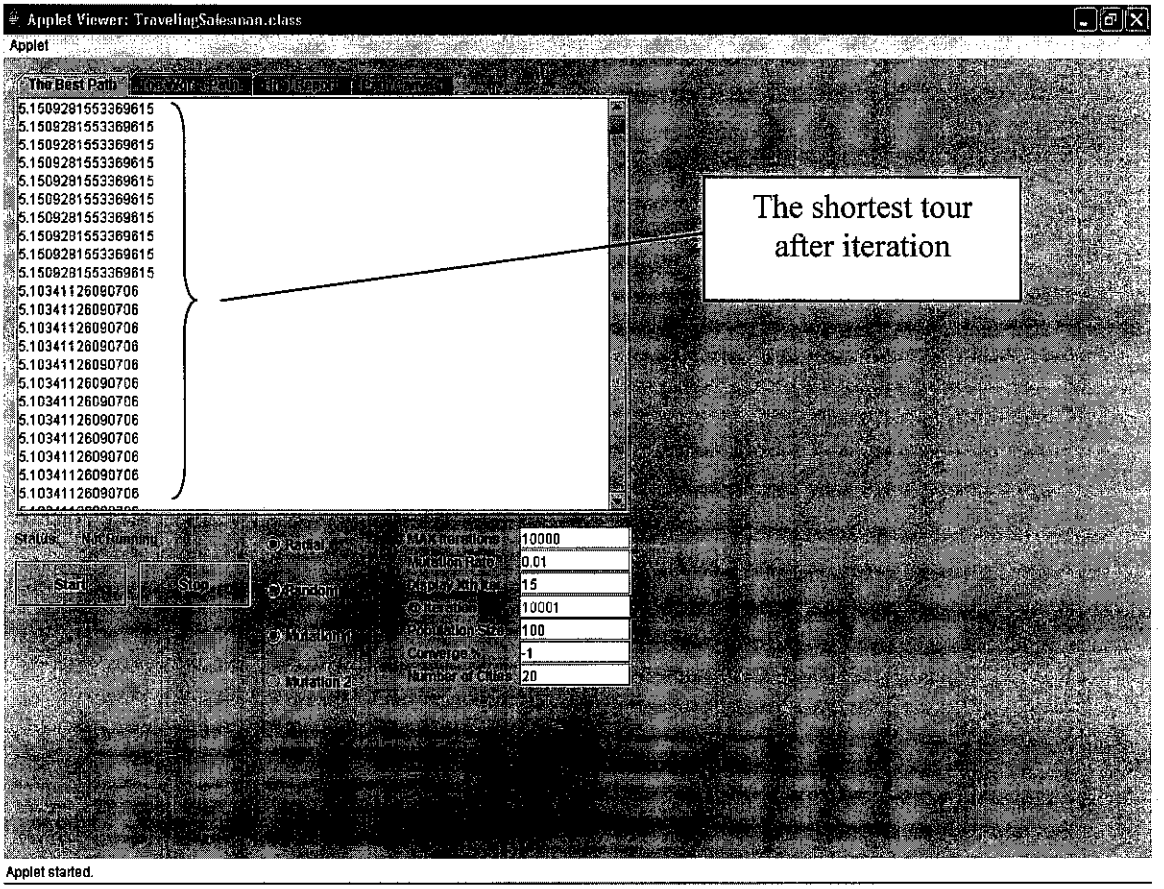

Figure: The Screenshot of the Best Path

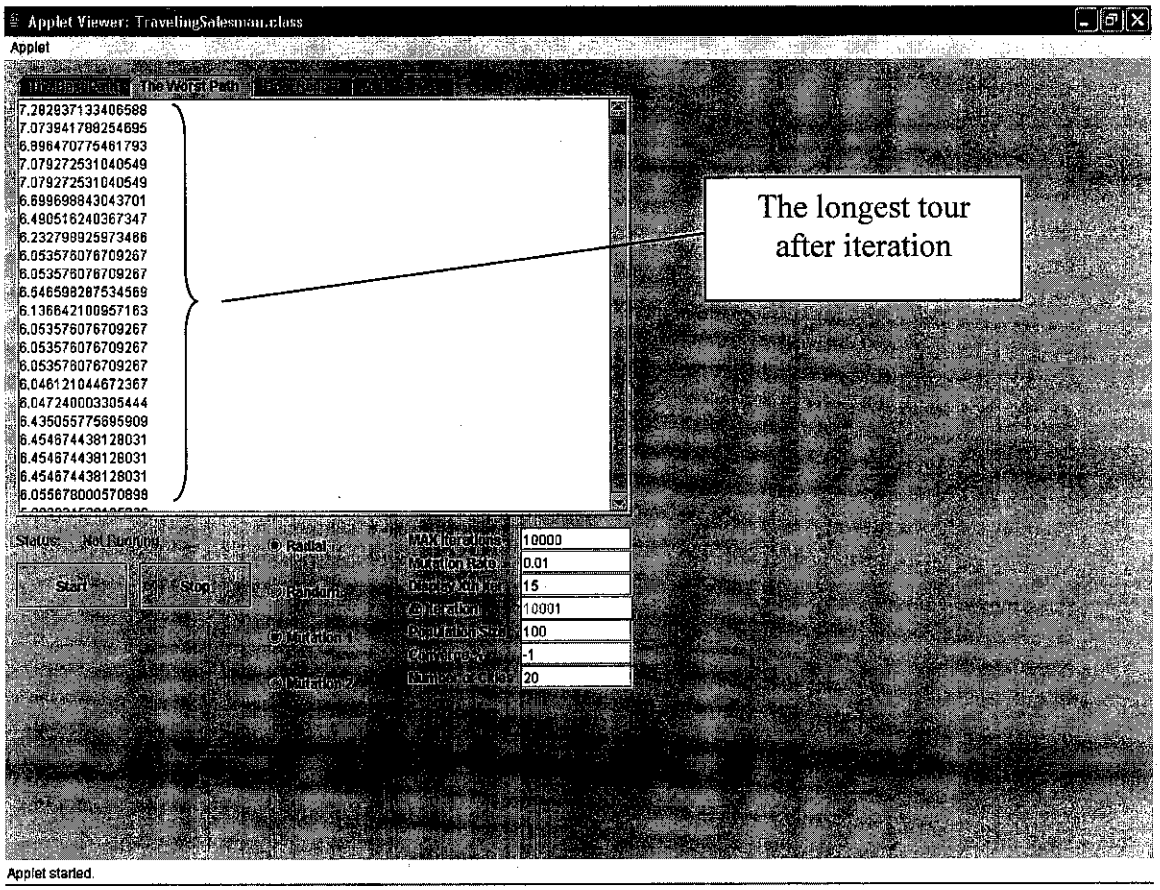

Figure: The Screenshot of the Worst Path

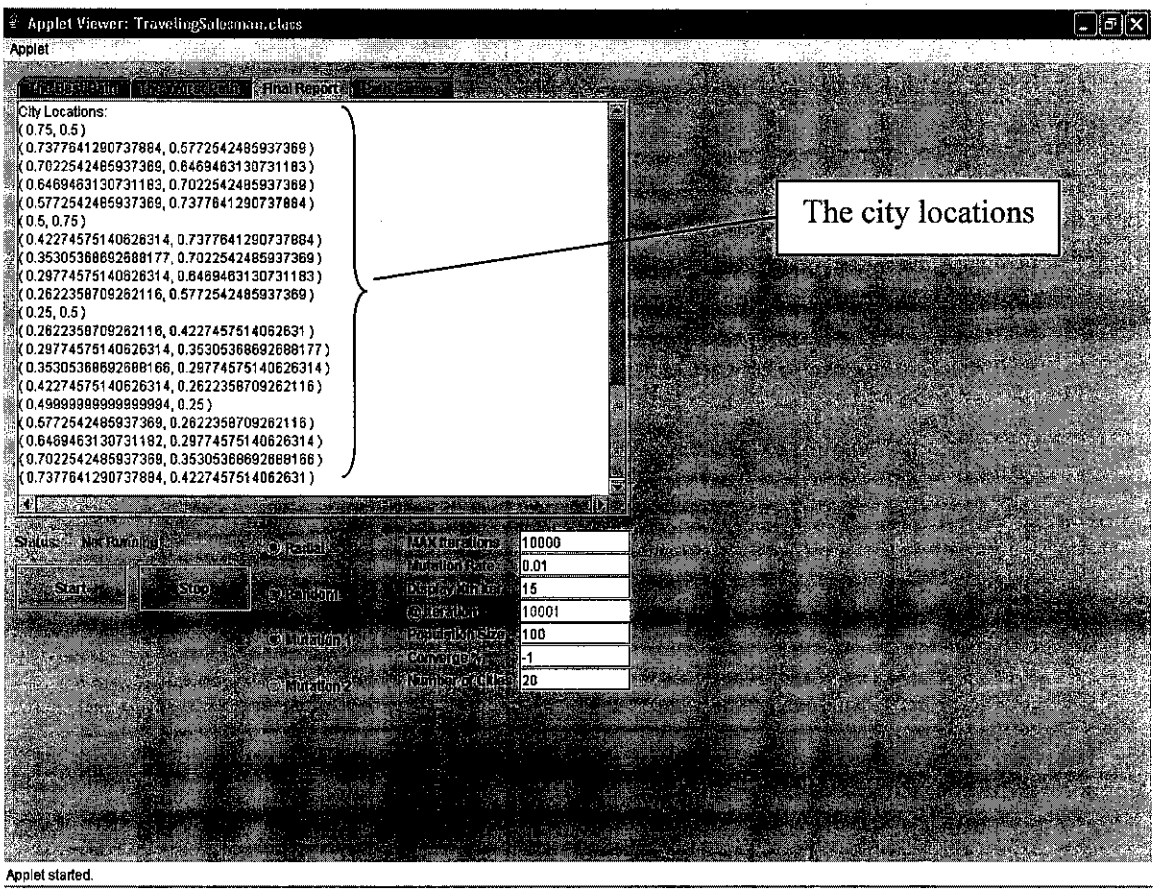

Figure: The Screenshot of Final Report

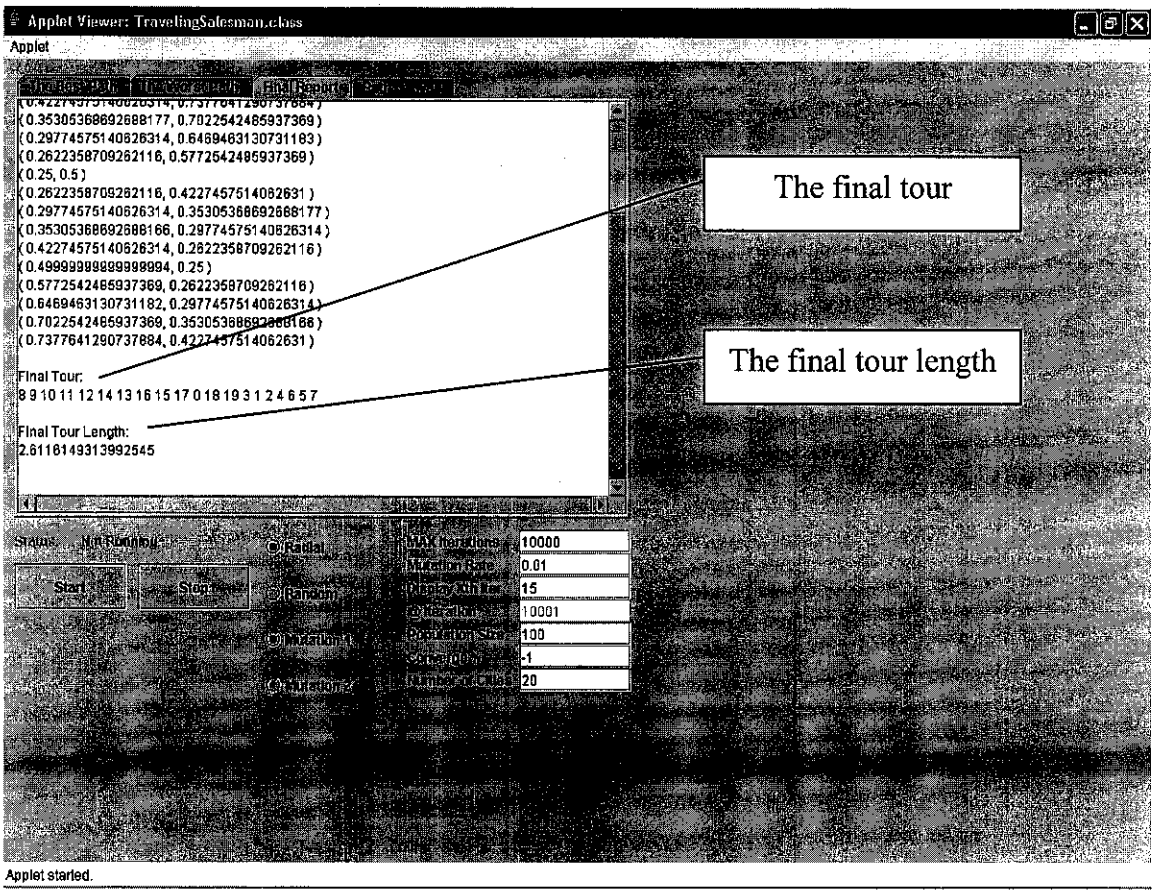

Figure: The Screenshot of Final Report (continued)

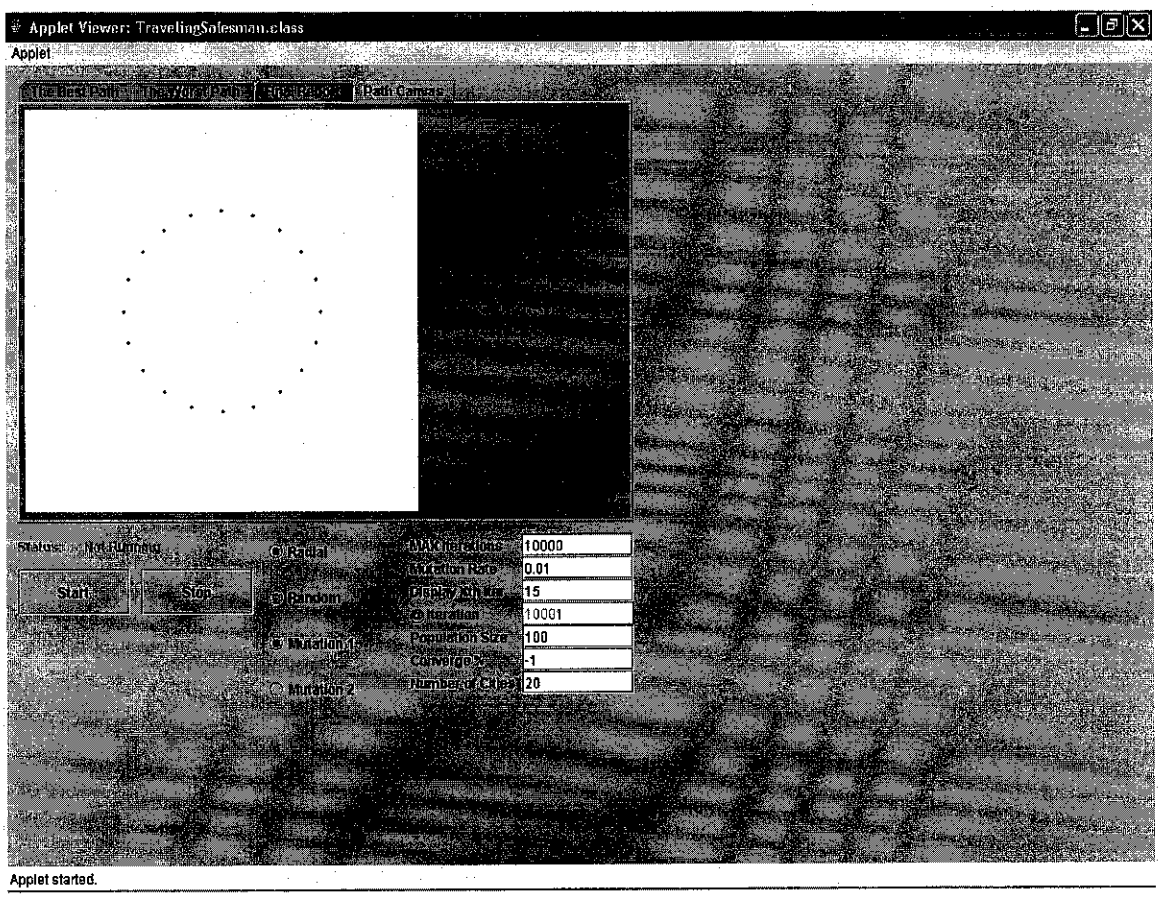

Figure: The Screenshot of Path Canvas with Radial City Location

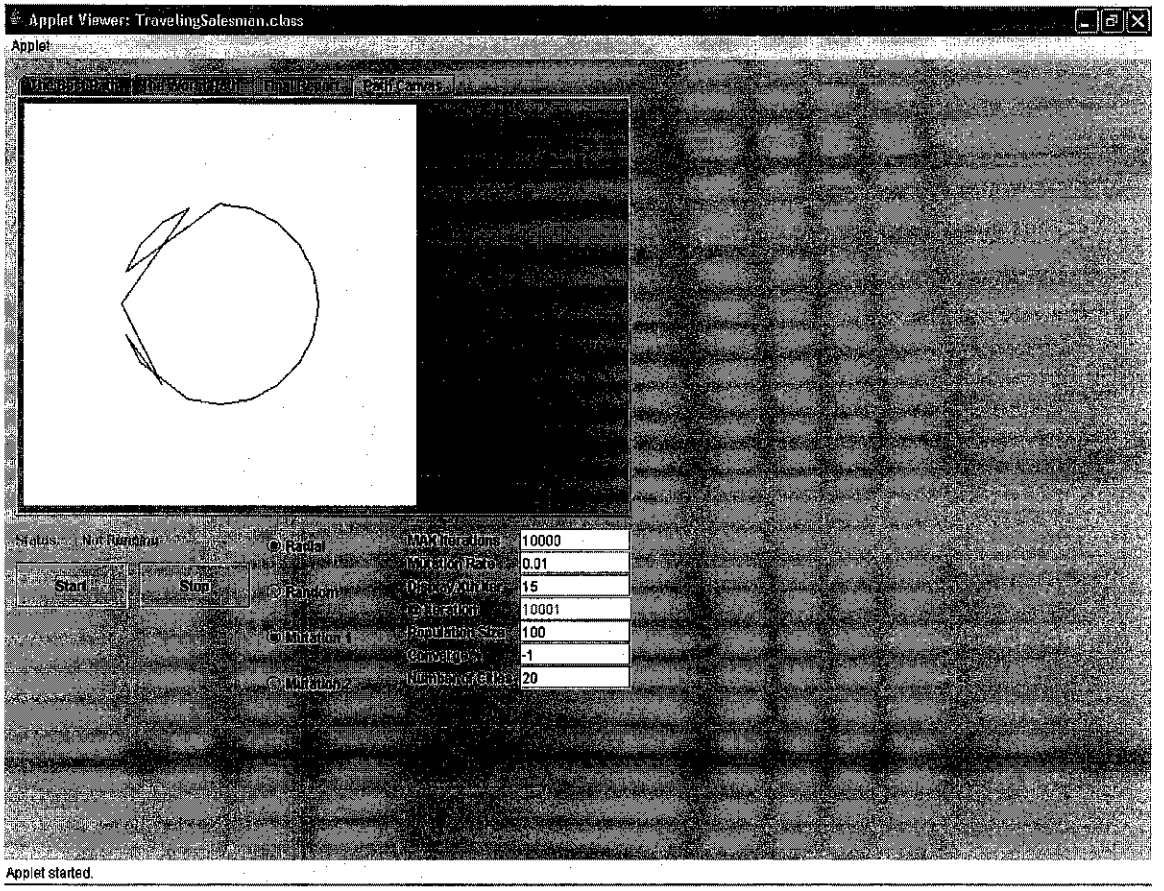

Figure: The Screenshot of Path Canvas with Radial City Location

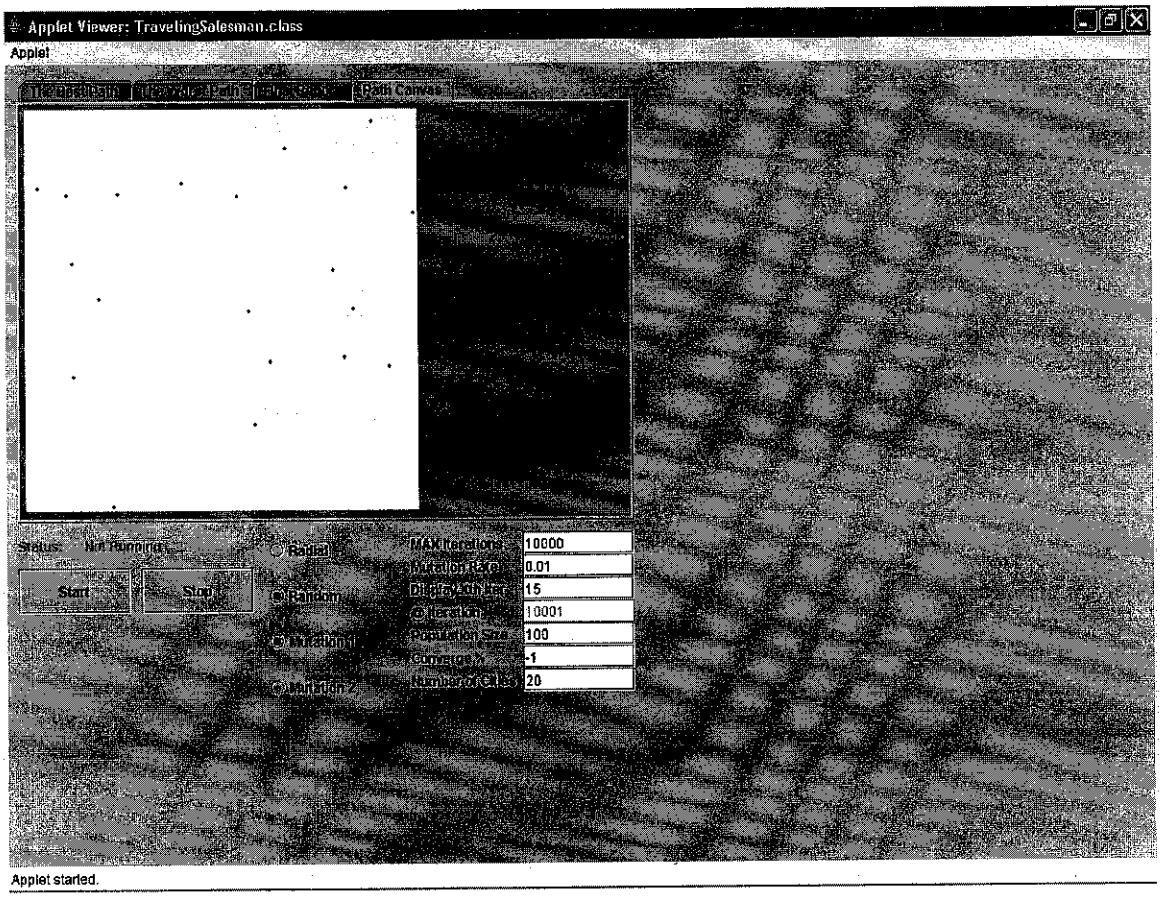

Figure: The Screenshot of Path Canvas with Random City Location

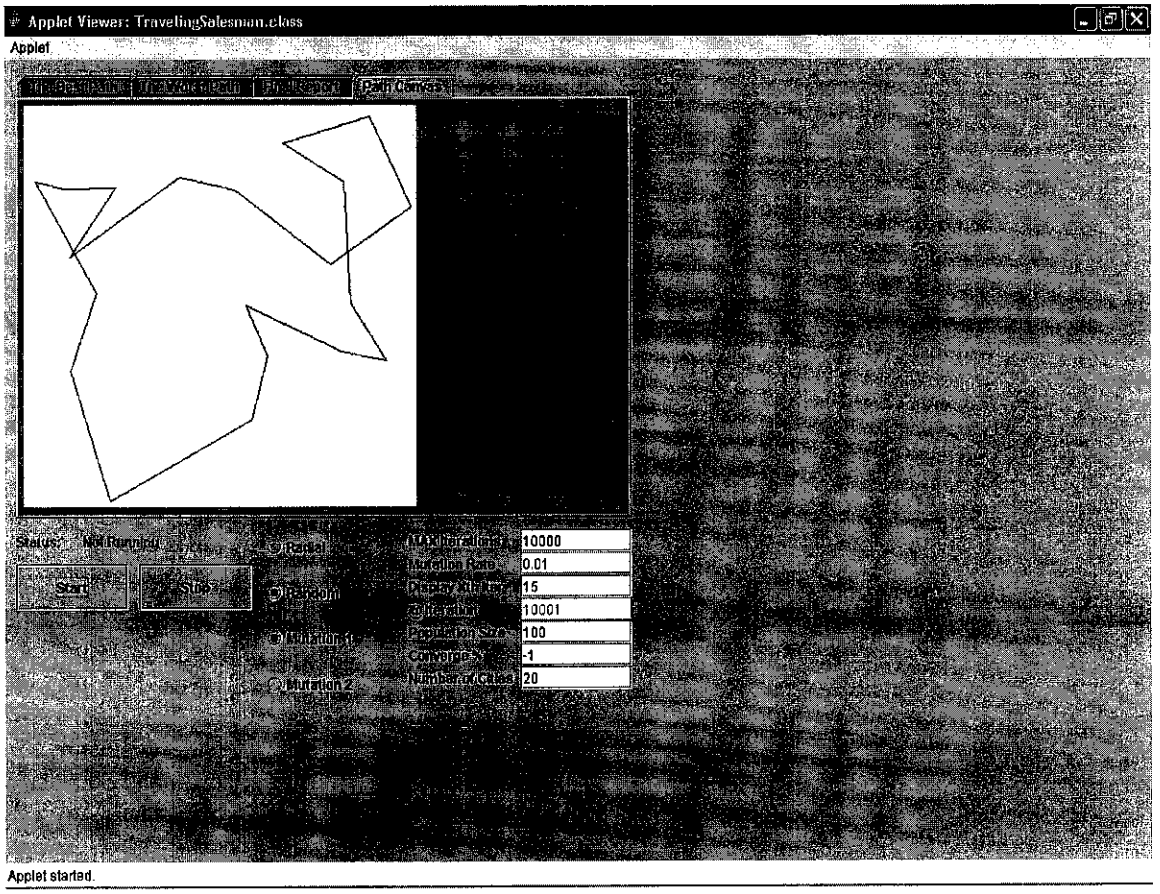

Figure: The Screenshot of Path Canvas with Random City Location