Intelligent Route Optimization for Travelling Salesman Problem

BY

LIONG KAH MEE

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

In partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Perak Campus)

JAN 2014

DECLARATION OF ORIGINALITY

I declare that this report entitled "**Intelligent Route Optimization for Travelling Salesman Problem**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____

Date : _____

ACKNOWLEDGEMENTS

Thanks to my supervisor, Ms. Mano, without her guidance and instruction I would not be able continue and finish my thesis. She give me a lot of insight of the genetic algorithm, how to do a proper thesis report, how to collect the sampling result data, giving me some example how should the algorithm might work and so on. There are many things that I never learn before she teaches me. She gives me the direction to continue on the thesis, and I gain confident step by step to finish the thesis. I would like to thanks my friends who support me during the thesis, giving me motivation when I facing the problems. Thanks to my family always support and cheer me up when I face the difficulties. Without all of them above, I definitely won't be able to finish this thesis. Thank you very much.

ABSTRACT

Travelling Salesman Problem (TSP) is one of the famous studied in optimization problem. Many techniques and application apply to solve the TSP in order to gain the optimization result and compare which is the best solution under some circumstances. Genetic algorithm is the one of the solution to solve this problem by using the processes observed in natural evolution to solve problem optimization and search problems. In this thesis, there have a basic genetic algorithm of TSP program that will give a result of best path length base on it types of operator, and I had modified the existing program to a new version that adding a special mutation function and to prove that the result from the modifying is much better than the original program. The better result is in term of getting the lesser amount of best path length from certain of criteria or circumstances, the more optimization of the algorithm. This thesis is can be treat as the guideline for future research on this area. Guideline as if using some of the specific operators by mention in the thesis, the result can be predict as how much can be improve and what is the expected result by using the operators as mention in the thesis.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
TSP	Travelling Salesman Problem
Program A	Existing basic genetic algorithm TSP program
Program B	Modified own genetic algorithm TSP program

Chapter 1: Introduction

1-1 Problem Statement and Motivation

1-1-1 Problem Statement

Many existing algorithm can be choose to implementing the TSP problem.

Currently there have many algorithms that already tested and done the research for the TSP problem. Which are the best algorithms to obtain the best solution, optimization and search for TSP problem? For this project, is need to study many journals and papers to do the research on each algorithm and get the best view which can be apply for the TSP problem.

Many different algorithms have been applied in solving the TSP over the years, yet question on cost efficiency and path optimization still arise due to the weaknesses or disadvantages of these applications.

Due to the increasing size of population, some of the algorithm didn't scope of the increasing size, so the performance is getting worst and unreliable. To verify a suitable algorithm, there must have testing, studying on how the algorithm can be suit for the circumstances.

From a thesis, author Kylie Bryant (2000) mention on the conclusion part that even though that genetic algorithm is prove that suitable to solve the TSP problem throughout many research but genetic algorithm is difficult to maintain structure from the parent chromosomes and still end up with a legal tour in the child chromosomes. Author also mentions that maybe come out with a good solution by having the better operators of mutation or crossover routine that retain the structure from the parent chromosomes.

By analysis the algorithms, optimization result is depend on the algorithm or technique you use. Different combination of operators from genetic algorithm example like mutation, selection or crossover will give different optimization result.

This is the fact that apply every single algorithm into the TSP problem will get the different optimization result. Some are good and optimum result, some are worst. Some algorithm may not fix into the TSP problem. Combine different operators between the mutation, selection and crossover will give the different optimization result. This is a very important step to choosing the operators in genetic algorithm. It will affect the optimization result and the outcome. From the journal by Geetha, Nishaa, and Vasumathy (2009) and title of "A perspective view on Travelling Salesman Problem using Genetic Algorithm", there have the listed different research done by some researches regarding on genetic algorithm and the total outcome is different. From this journal we can be prove that by combining different operators, different perspectives under some circumstances, the optimization result will be different.

If the solution is not reliable by users, it will be abundant and forget.

Develop a new structure algorithm is not a problem, but will it be reliable to the user. Technology keep on changing from time to time, we need to keep track on it, always coming out new solution to be able to cope up with the market.

1-1-2 Motivation

There are many real life examples that will dependent on example of TSP. As the era change to becoming more and more advance, people want to get the best and faster solution due to any problems. It's very important to us for make the improvement to get the better solution, optimization route to minimize the cost, time consuming and etc.

Technology is moving fast and better day by day, if the solutions are not reliable to the user, it will be abundant and forget. They will come out with a better solution to solve the problems. It same goes to the real life TSP example. Take a real life example may happen, Courier Services. If the courier service didn't plan well on scheduling and which route can be faster and shortest in time consuming by making sure going to every town to deliver the goods, this will cost them a lot of waste resources, example like petrol money, time consuming and etc. The motto of courier services is to be fast in delivery. If they cannot meet this condition, how would they competing on the market if their competitor has the better solution on optimization.

1-2 Project Scope

The fundamental algorithm will be choosing for this project will be genetic algorithm. There have many way of doing and experiments using the genetic algorithm apply in TSP problem. This project is to prove that different combination of genetic algorithm's operator will give the different optimization result. Sometime will get the better optimization and result by comparing the existing basic genetic algorithm version and the modified genetic algorithm version.

This project separate into 2 modules:

I. First module: Finding the solution on combine either existing algorithms or technique operators and test it by doing some analysis to produce a better optimization and accurate result for TSP under some circumstances.
II. Second module: Generate result and comparing the existing basic genetic algorithm program and the modified version of genetic algorithm. The modified version of genetic algorithm program should get a better and optimization result than the existing basic genetic algorithm program.

1-3 Project Objective

I. To proposed an effective and optimization solution for TSP using genetic algorithms.

Implementing a new solution from an algorithm is the main focus and contribution for this project. There have many tested solution from different algorithms but I think there still have spaces to improve and getting more optimization result for the TSP problem. There always a new way of by generating new operator or find a new solution to solve a problem.

II. To research on which operator will be suitable to solve the TSP problem.

Since genetic algorithm have many operators or ways to solve the TSP problem, and each may give the different result under some circumstances. So in this project, will be do the research on which operators is suitable or maybe there have other ways like combining different algorithm in the genetic algorithm to produce a more optimization result.

III. To verify the proposed solution in term of better optimization and better result.

By using two different structures of program A (basic genetic algorithm program) and program B (modified genetic algorithm program), testing them by using different type of criterions or properties and get the result of the program's behavior. The propose solution (program B) should be always get the optimization and better result compare with program A.

IV. To development and test the prototyping applying the genetic algorithm.

Not only propose the new style of algorithm for the TSP problem but also develop and testing is it one of the main purposes of this project. Develop will get better to know the fundamental and what characteristic of the operators choose and testing the algorithm also will know how is the accurate optimization will be under some circumstances. This is to overcome the problem of getting the worst result by applying the different algorithm in TSP. Development the propose solution into prototype application. Demonstrate how to find the shortest or optimum path with the minimum cost between the cities.

1-4 Project Background

Travelling Salesman Problem (TSP) is widely discussed in artificial intelligent search and optimization problem. According to the history, the fundamental form of TSP was firstly discovered and studied by the mathematician Karl Menger (1920) in Vienna and Harvard.

TSP problem explanation as if there are N number of cities and the M distance path between the cities are given and define exactly the shortest tour where each city is visited only once and returning to the starting point. The larger of the number of cities, the larger probability and it's not possible to calculate all the ordered combination. The formula for the number of combination is (n-1)!/2. TSP problem is nondeterministic polynomial time (NP-Hard).

There are many ways to solve the TSP problem by applying the programming structure example linear programming, inclusion-exclusion and etc. But in artificial intelligent way, had some algorithms in current trend is ready for implementing and applying to the TSP problem, such as genetic algorithm, ant colony algorithm, and etc.

The real life system example that using the TSP problem as fundamental are:

a) School Bus Scheduling
b) Food Delivery
c) Routing of tracks purpose for delivery
d) Logistic purpose and etc.

Travelling salesman problem have two types, which are symmetric travelling salesman problem and asymmetric salesman problem. Symmetric TSP is mean by the distance between the two cities is the same in each opposite direction. For the asymmetric TSP, paths between maybe not exist in either directions or the distances might be different. Symmetric TSP is forming the undirected graph but asymmetric TSP is forming the directed graph.

Asymmetric TSP also can be explain as when the cost of travelling from city J to city I is not the same when city I to city J. TSP have not only symmetric TSP and asymmetric TSP but also have different variations example Hamiltonian path problem, bottleneck TSP, multiple TSP, dynamic TSP and etc. Since TSP is a NP-Hard problem, there are no known algorithms that will solve it in polynomial time.

In this project, I will be using genetic algorithm to solve the TSP problem. In generally, genetic algorithm is using to selection of parameters to optimize the performance of a system. Here are some of the examples that genetic algorithm apply in the real life operating system:

- Gas distribution pipeline system
- Travelling Salesman Problem
- Traffic Light system
- Allocation of funds to projects and etc.

Chapter 2: Literature Review

2-1 Genetic Algorithm

The genetic algorithm is an optimization and searching technique based on the principle of genetics and neutral selection. It includes the idea of maximizing the fitness function (e.g. minimizing the cost function). It also stand for survival of the fittest idea into a search algorithm which provide a method of searching which do not need to go through every possible outcome to obtain the good result. Genetic algorithm is a natural selection, by applying the fitness function, the fittest individual should be surviving and the next generation will be healthier because they bred from healthy parents.

According to Haupt & Haupt (2004) wrote that, the advantage of using the genetic algorithm are optimizes with continuous or discrete variables, can be deal with large variables, is well suit on parallel computers, provide a list of optimum variables but not a single solution and etc. These advantages list can be provide a great list of optimization compare to the traditional optimization.

Firstly we discuss about the fitness functions. Applying a good fitness function in the genetic algorithm is very important since this function is determine the how optimization you want in the program. It's also known as objective function. By applying the fitness function, you will get the fitness value for each of the chromosome that create, from this value, your operation will choose which chromosome is good chromosome will survive become the parents and mutate the next generation as children.

There have 3 types of operators that can determine the characteristic and the optimization of a genetic algorithm. Those are selection operator, crossover operator and mutation operator. These 3 operators must be existing in order to do a complete cycle of the genetic algorithm.

Selection operator is the process of choosing 2 parents/ chromosome from the population initially generate by some random number function (depend on the problem case) and get choose by the fitness value they own. In word, the highest of the fitness value, the highest chances to be choose by the selection operator. This appears of selection pressure that will make sure the quality of the genetic algorithm. The higher of the selection pressure, the better of the chromosome are favored. The selection operator drive the genetic algorithm makes sure to improve the population fitness over the successive chromosomes.

Below are few examples of different types of selection operators:

- Roulette Wheel Selection Operator

  It is the commonly use and tradition operator in selection operator for genetic algorithm. Roulette wheel selection is kind of linear search for the chromosome in a list that contains wheel weight proportional to the chromosome's own weight (fitness value). A target value which is the random proportional of the sum fitness in population will be set. The population will step through in until the target value is reach.

- Random Selection Operator

  This technique of operator is randomly choosing the chromosome from the population. It does not choose depend on the fitness value.

- Tournament Selection Operator

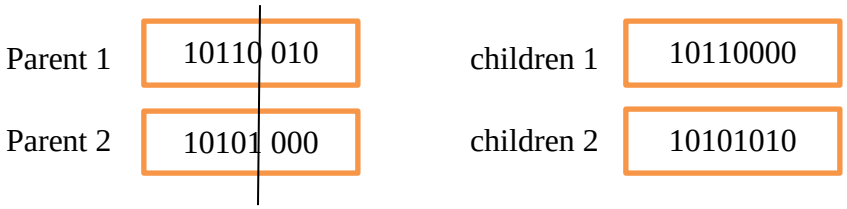
  N chromosome will be selected randomly from the population and the selected chromosome will be competing within each other. The chromosome with the highest fitness value will win and will be include in the next generation of the population. The number of chromosome competing each other is referred to the tournament size, as commonly set to 2.

Crossover operator is also can be name as recombination operator. It is the process of selecting 2 parents of chromosomes and produces a children chromosome. After of the selection operator that choose the ideal batch of parent chromosomes, the crossover operator will mate 2 parent chromosomes and produces a children chromosome as a new batch of population. 2 parent chromosomes will cross site each other randomly, and position values are swap between each other.

Below are few techniques of crossover operator can be used in genetic algorithm:

- Single Point Crossover Operator

  Two mating chromosomes are separate with a single corresponding points and the section after separate exchange.



- Partially Matched Crossover Operator (PMX)

  It is designed to preserve absolute position from both parents. Partially matched crossover works as two parent chromosomes are aligned, and two crossing sites are selected uniformly at random along the strings as the matching section.

- Cycle Crossover (CX)

  Cycle crossover preserve the information contain the both parent chromosomes that all elements of the offspring are taken from one of the parent chromosome. For the cycle crossover the offspring inherits all the elements found at the same position in two parent chromosomes. Starting with a randomly chosen position in the offspring, an element from one parent chromosome will be randomly selected. Make sure there are no implicit mutation occurs. Then, the unassigned position to the right is proceed in the same way until the entire element been considers.

Mutation operator plays a role that to avoid the genetic algorithm trap at the local minimum. It is view as background operators to maintain the genetic diversity in the population. Mutation involves the modification of the value of each gene of a solution with some mutation probability.

Below are some of the mutation operators that been use in genetic algorithm:

- Inversion Mutation

  Inversion mutation is to selecting two positions within a chromosome at random and then inverts it in the substring between two positions.

  Chromosome A: (1 2 3 4 5 6 7 8)

  The position of 3 4 5 is selected at random using two position:

  Chromosome A: (1 2 | 3 4 5 | 6 7 8) after inverting, the chromosome B will become

  Chromosome B: (1 2 | 5 4 3 | 6 7 8)

- Displacement Mutation

  This mutation operator will select a sub chromosome and inserts it at a random position outside the sub chromosome.

  Chromosome A: (1 2 3 4 5 6 7 8)

  Randomly choose sub chromosome is 3 4 5 and the randomly selected insertion position is between 7 and 8, then the result will be,

  Chromosome A: (1 2 3 4 5 6 7 | 8)

  Chromosome B: (1 2 6 7 3 4 5 8)

Flow Chart of the Genetic Algorithm

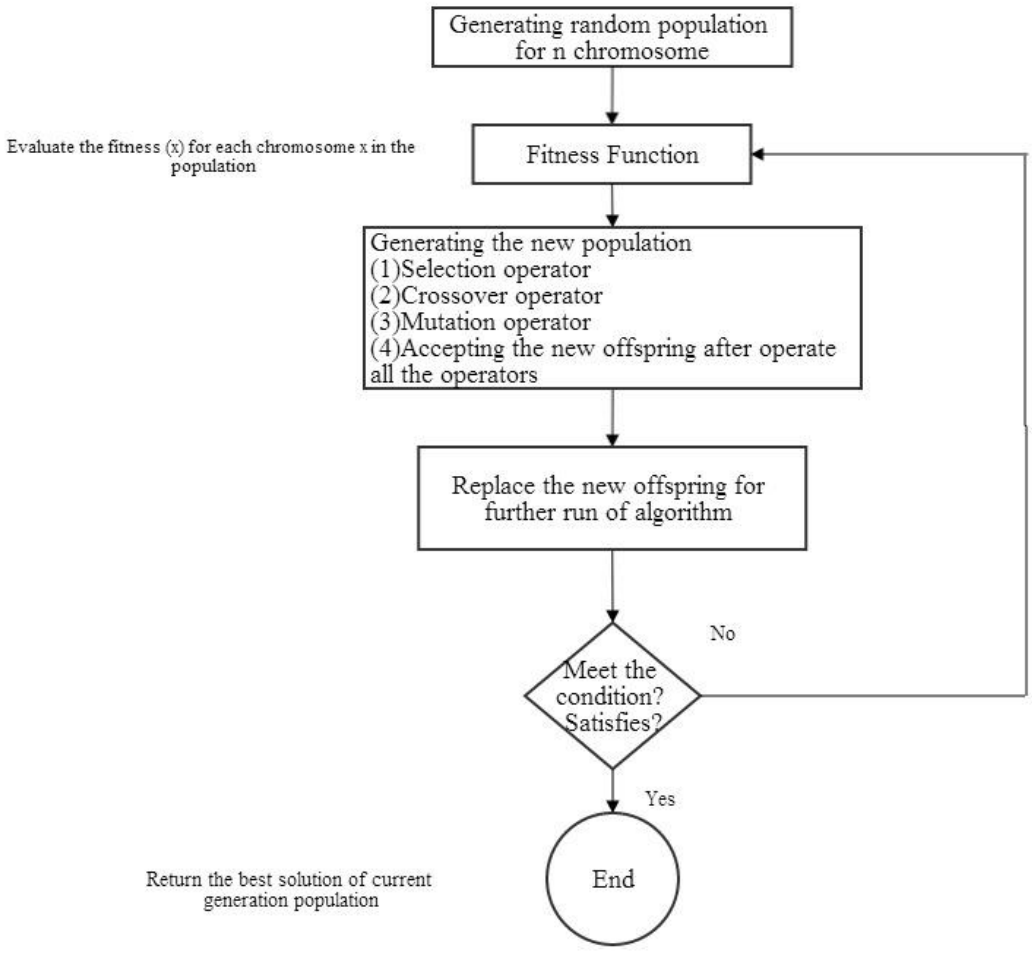


Figure 2-1 Genetic Algorithm Program Flow chart

Figure above showing that sequences on how a basic genetic algorithm program execute and get the optimization result.

2-2 Summarize of previous research done for Travelling Salesman Problem (TSP)

There have been many intelligent algorithms to solving the TSP problem, due to Wang Hui (2012), the author trying to use three algorithms to solving the TSP problem and compare each of them which one come out the optimization result. Three of the algorithms are Basic Genetic Algorithms, Hopfield Neural Network, and Basic Ant Colony Algorithm. By using these three algorithms, the author is compare them by differentiate with the time complexity, space complexity, advantage and disadvantage of calculation result and realization difficulty level.

Below is the table of the comparison three of the algorithms.

Title	Basic Genetic Algorithm	Hopfield Neural Network	Basic Ant Colony Algorithm
Time Complexity	O(Tn_0n^2)	O(TT_1n^2)	O(Tmn^2)
Space Complexity	O(n_0n)	O(n^2)	O(n^2)
Comparison the merit of result (Standard optimal solution: 426)	Global optimal: 1007.9	Best path length: 1392.3	Best path length: 480.13
Level of difficulty (Using code line standard by MATLAB)	186 lines	93 lines	96 lines

Table 2-2-1-T1: Comparison between Basic Genetic Algorithm, Hopfield Neural Algorithm, and Basic Ant Colony Algorithm.

From the above table of comparison the merit of result by using the standard optimal solution eil51 published on A Library of Standard TSP Instances (TSPLIB) provide is 426. The ant colony algorithm approaches to the optimal solution. From the research, by comparing the level of difficulty that made standard by using the code line standard by MATLAB is the genetic algorithm will be more difficult comparing to the Hopfield Neural algorithm and ant colony algorithm.

Many form of TSP problem, not only the single form but also in dynamic form, multiple form and etc. The difference for dynamic form is the N number of cities and the cost matrix is time dependent. And the difference for multiple TSP form is the N number of salesman who visited M number of cities and each salesman is defined to start and end at the same destination.

'A perspective view on Travelling Salesman Problem using Genetic Algorithm' (Geetha RR, Nishaa B & Vasumathy S 2009), is a journal basically discuss about the fundamental knowledge on genetic algorithm, how the genetic algorithm be able to solve the TSP problem and etc. They also state that there had many solution way of using the genetic algorithm. This paper did some research and recommendation which algorithm and techniques from genetic algorithm can be applied in the TSP problem.

This paper discuss on different researches's work done on solving the TSP problem and indicate how the researches implementing and applying the genetic operators with different technique to solve the TSP problem. They state for the different operators like mutation, selection, crossover and combination used by them to produce a better solution to solve the TSP problem.

Take an example from Nilesh Gambhava and Gopi Sanghani (2003), they proposed a paper where using genetic algorithm with suitable probability crossover and mutation operator for small population size and less number of generations. They explained two encoding techniques, matrix encoding and straight encoding. Good choices of encoding to be able to use in the crossover process.

Below is the summary of the genetic operator that used by researchers in their papers to implementing and optimizing the TSP problem.

Journal Title and Author Name	List of Genetic Operator used
'Travelling Salesman Problem using Genetic Algorithm', Nilesh Gambhava and Gopi Sanghani (2003).	a. Encoding operator: Matrix encoding, Straight Encoding b. Selection operator: Rank Selection c. Crossover operator: Edge Communication Operator d. Mutation operator: Reciprocol Exchange Mutation
'Enhanced Travelling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA)', Buthainah Fahran, Al-Dulaimi & Hamza (2008).	a. Selection operator: Roulette Selection , Deterministic Sampling b. Crossover operator: Order crossover, Partially Matched Crossover, Cycle Crossover
'An Improved Genetic Algorithm to Solve the Travelling Salesman Problem', Omar M.Sallabi & Younis El-Haddad (2009).	a. Crossover operator: Swapped Inverted Crossover b. Mutation operator: Partial Local Optimal Mutation Operation c. Other operator use: i. Rearrangement operation ii. Population Reformulates

Table 2-2-2-T2: Summary of the genetic operator to implementing and optimizing the TSP problem.

Beside the genetic algorithm, another algorithm Ant Colony Optimization Algorithm also has been discussed for the TSP. It first introduced by Dorigo et al (1991). It approaches for solving combinatorial optimization problem. The fundamental idea of ant colony optimization algorithm is based on the natural ants that succeed to finding their shortest path to reach their destination of food allocation by communicating via a collective memory.

There also a way by study the wandering salesman applicability on solving the TSP problem base on genetic algorithm (Javad, 2010). According to him, there have many ways on solving the TSP, but the most optimization and good technique is genetic algorithm. However, it depends on mainly on how the problem encoded and which type of crossover and mutation are used.

According to Choi In-Chan, Kim Seong-In, and Kim Hak-Soo (2001), they doing the research on asymmetric TSP by using genetic algorithm with a mixed region search. They extend the search space of the genetic algorithm by purposefully generating infeasible solution with sub tours. The sub tours in this paper define as a closed circuit formed by sub set node within the salesman's visiting list.

This paper mention that patching algorithm works well when assignment relaxation problem contain sequences of good arcs, which potentially reside the optimal solution on asymmetric TSP. Infeasible solution with sub tours can be generated so that their solution almost close to optimal value of the assignment relaxation problem then the patching algorithm can be effective applied in to them to obtain the optimal solution in asymmetric TSP.

According to the other conference paper (Wang,Zhang,Li 2007), they propose a solution that to improve the performance of the genetic algorithm for TSP. The one new operator they proposed was Untwisted Operator. This operator can untie the knots between the route so it can shorten the length of route and quicken the convergent speed.

The paper mentioned that sometime the route of the cities twist frequently. In order to optimize the route result, they were used the untwisted operator to untwist the route between the cities. The untwist operator is applied after the mutation operator to get the optimize route. Untwist operator pick up the 2 positions from an individual X_n that picking from the population set. The position is mean 2 cities positions, k and m by picking randomly while k < m. K and M value need to satisfied the distance formula they are set and a new individual Y_m will be generate. At the final result, they claimed that by adding the untwist operator inside the genetic algorithm of TSP, the result is getting optimum and improve become better.

A conference paper (Yu,Chen,Li, 2011), propose a new design of Genetic Algorithm for solving the TSP. They are using the roulette wheel selection mechanism of survival of the fittest strategy, a heuristic crossover operator, and an inversion mutation. The most different from a normal operator, they are adopting both the roulette wheel selection and the strategy of best survival to survive. By making sure that the individual with the highest fitness value to survive and preserve to next generation and calculate again the formulate rate to choosing again from the population and get the best survival. They are using coordinates of 50 cities and prove that improve in term of efficiency, stable and closer to the optimization.

Chapter 3: Methodology and Technologies Involved

3-1 Methodology

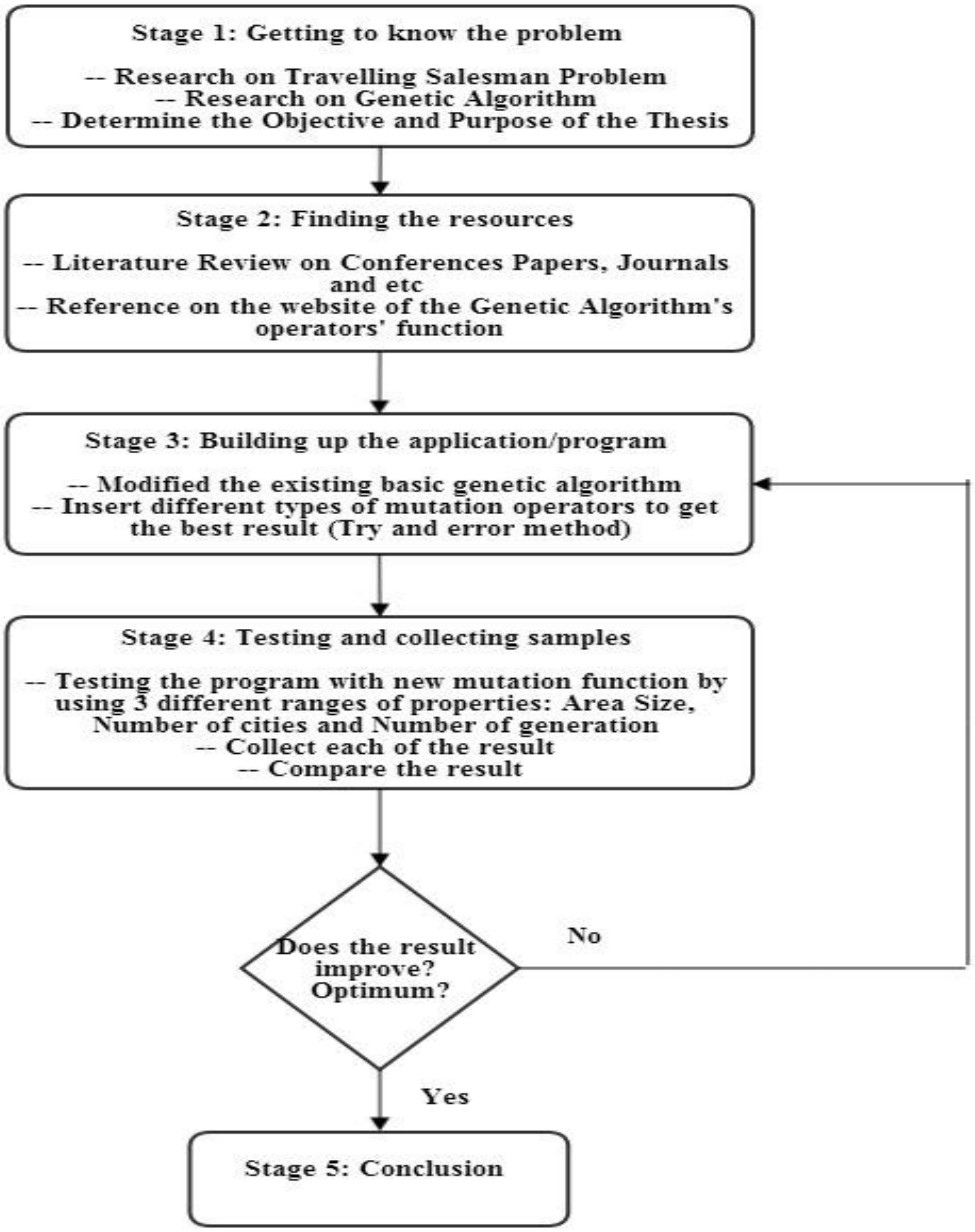


Figure 3-1 the methodology flow chart of thesis

3-2 Technologies Involved

There are 2 MATLAB programs involve in this thesis. One program is the basic genetic algorithm TSP program (Program A) and one program is a modified version of the genetic algorithm TSP program (Program B). Both are in MATLAB code and will produce total 2 diagrams with a single figure frame. The 2 diagram is the graph of area size n x n includes the number of cities as the black dot and the color graph of path number (number of population) inverse with index of sequence cities.

3-2-1 Description of the Basic Genetic Algorithm program for Travelling Salesman Problem (Program A)

This basic genetic algorithm is follow by below sequence operations and repeated in generation loop:

- Get the length of the paths
- Find the probabilities of the chromosome
- Prepare the crossover operation of the chromosome according to their probabilities
- Do the crossover operation, the parent chromosome replace by the children chromosome
- Do the 3 mutation operations to be able find the optimum path solution.

Since this program generating the chromosome/path, so there will be a lot of paths that contain different lengths. A path will be crossover with another path and do the mutation operation. After the crossover operation, the chromosome/path will be replaced by its children path. All paths record in a form of Matrix.

2	1	3	4
1	2	3	4
4	3	2	1
3	4	2	1

← path 1
← path 2
← path 3
← path 4

Figure

Above is the figure of example population with 4 chromosomes/paths (population size = 4) and it's contain 4 cities. Each row of matrix is describing 1 path. For path 1, the cities no.2 is connected to city no.1, city no.1 is connected to city no.3, and city no.3 is connected to city no.4. The last city in the list will be always connecting to the first one. This is to fulfill the requirement of the Travelling Salesman Problem. And each city only repeated one time in the path.

In this program, by generating the initial population of random path, it using the randperm matlab function. This function will return a randomly permutated number by example:

>>randperm(4)

The answer will be = 4 3 2 1

To calculation of path lengths before the iteration, it is calculated matrix of distance dsm. It's nn x nn matrix which nn is number of cities and dsm(a, b) is the distance between the a city and b city.

Example of the matrix:

	a	b	c	d
a	0	2	1	6
b	2	0	4	7
c	1	4	0	3
d	6	7	3	0

The inverse distance is used in this program, that is $\frac{1}{distance}$. The higher of the distance, the smaller of the inverse distance. In order to maximize the distance, then it need to minimize the inverse distance.

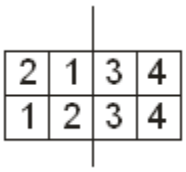
This program contain of roulette_wheel_indexes.m function that work in such way:

Using the matlab function that given the random number form [0:1] range. Split this range in to 6 pieces and length of each piece is equal to probability. The probability of choice of a piece is proportional to its length, because rand function gives random number from uniform distribution. This process will repeat 6 times to choose 6 parents. Then the 6 parents will be replaced with the 6 children.

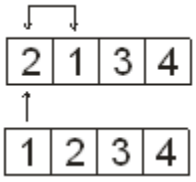
The crossover operation of this program:

There have 2 parent chromosomes and want to replace with the 2 children chromosomes.

First, randomly choose the point of cross-section, let's pick 2nd and 3rd positions:



Make the children that have first part like in parent 2. Copy the cities from parent 2 to parent 1, cross-section first then coping and make swap.



Then after that will be

1	2	3	4

1	2	3	4

Each time need to find where is the number of city of parent 2 in parent 1 to make swap. Now go to the next position, do nothing here because it already 2. So get the 1st child,

1	2	3	4

← child 1

1	2	3	4

The same goes to child 2 but for child 2, copy cities from parent 1 to parent 2.

The more big inverse distance of some path, then it will have bigger probability to be parent in crossover.

Roulette_wheel_indexes.m function was made to able to compute the probabilities as inverse distances divided by sum of inverse distances:

$$p_i = \frac{\frac{1}{d_i}}{\sum_i \frac{1}{d_i}}$$

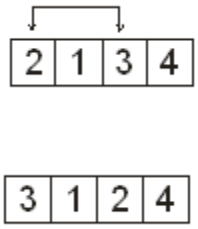
p_i = probability of i path to crossover

d_i = i path length

After the crossover operation, there will be 3 types of mutation to be used in this program:

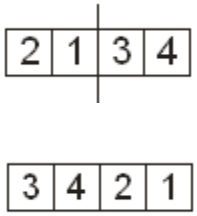
1. Mutation of exchange 2 random cities in the path

   From a single path, choose 2 random cities to do the swapping operation.



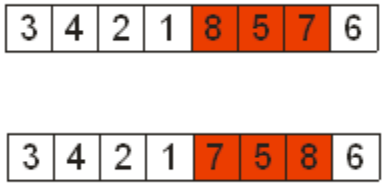
2. Mutation of exchange 2 paths

   Randomly choose the point of split, example chooses the position between 2 and 3 and do the swapping operation.



3. Mutation of flipping the random piece of path

   First choose some random piece of path, and then flip it left to right.



This program contains the Elitism function to make sure of improving of best path each iteration or keep the same path. It can save the best path and put it manually to next generation without any changes.

```
G (1,:)=Gb; % elitism
```

3-2-2 Description of the Modified Version Genetic Algorithm program for Travelling Salesman Problem (Program B)

The original basic genetic algorithm having 3 types of mutation, exchange 2 random cities, swap between 2 paths and flipping random piece between the paths. And the modified version for the mutation part is that, remain the first 2 part of mutation operators, eliminate the flipping random piece between the path mutation operator, add on the modified type of the mutation operators call as mutation binary string bit by bit. This mutation function is reference from the (Stanoyevitch, 2011), book from Discrete Structure with Contemporary Applications. Below shows that the print screen of the mutation functions:

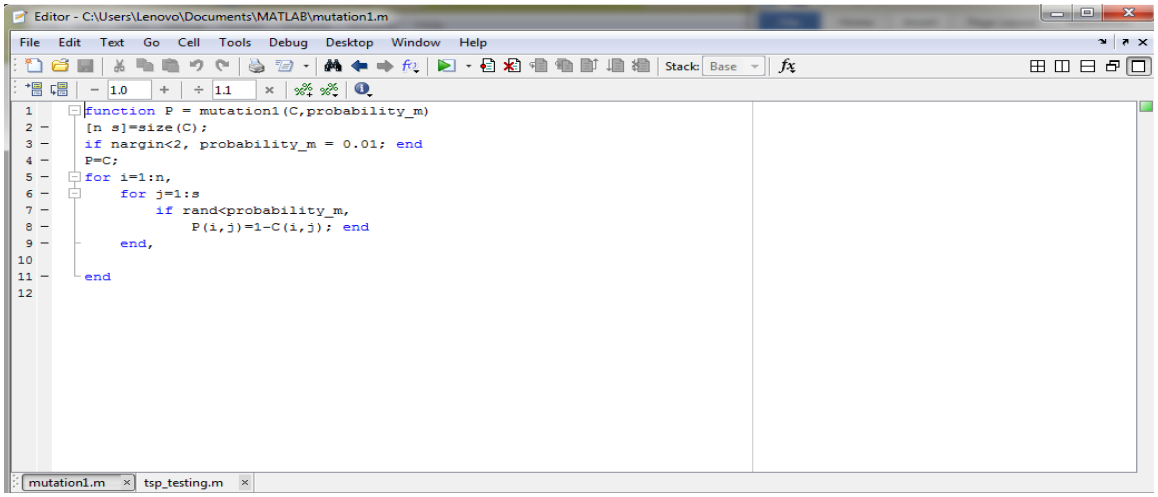


Figure 3-2-2 Mutation binary string bit by bit function

This mutation is almost the same concept with the Bit-flip mutation. This function is taking the area size n x s as the input matrix C as the psc in the program, the row vector with cities number of path. This function is goes through the binary string of the population, mutate it bit by bit example from bit 1 change to 0 and bit 0 change to 1.The mutation probability of this function is set to default as 0.01. The output size will get the same as the input size, it does not effect and lesser the input size.

This type of mutation also get apply by the Cornell University students in the MIT press (Chicano, M.Sutton, Whitley, Alba, 2013) of Fitness Probability Distribution of Bit-Flip mutation. But they are applying this method to the different perspective, theory of landscapes and Krawtchouk polynomials to conpute the distribution of the fitness value.

Chapter 4: Testing

Below is the laptop information details while doing the testing and retrieve the result:

Laptop Series Lenovo Z460

CPU Processor: Intel®Core™ i5 CPU M480 @2.67GHz

HDD: 500GB

VGA: INTEL Int GMA 4500M

Wireless: 802.11n

ODD: DVD Sup.MTI

Installed memory (RAM): 4.00GB

System type: 64-bit Operating System

Operating System: Window 7 Home Premium

I'm using the existing basic genetic algorithm TSP program version (Program A) and a modified own genetic algorithm TSP program version (Program B) to perform the testing process. I want to differentiate the performance result between both of the programs and the modified own version will be getting the better and optimum result than the existing basic genetic algorithm program. For the whole testing the probability of mutation (exchange 2 random cities) is 0.01, probability of mutation (exchange 2 piece path) is 0.02, probability of mutation (flip random piece of path) is 0.08 and the probability of the modified version mutation is 0.01.

4-1 Testing A: Number of cities: 40, Population Size: 3000, Number of generation: 500 – 2000, Area size: 10 x 10

Properties	Program A	Program B
Number of cities	40	40
Population Size	3000	3000
Area Size	10 x 10	10 x 10
Number of Generation	Case 1: 500 Case 2: 1000 Case 3: 1500 Case 4: 2000	Case 1: 500 Case 2: 1000 Case 3: 1500 Case 4: 2000

Table 4-1-1 The List of the properties to be tested in the Genetic Algorithm TSP Program

The result for both will be shown as separate table below:

Program A:

Number of Case	First Execute (Best Path Length)	Second Execute (Best Path Length)	Third Execute (Best Path Length)	Average (Best Path Length)
Case 1: 500	121.16	105.55	109.07	111.93
Case 2: 1000	94.53	75.68	94.42	88.88
Case 3: 1500	59.33	70.81	69.30	66.48
Case 4: 2000	66.03	95.96	61.26	74.42

Table 4-1-2 Result of the best path length by testing different numbers of generation for Program A

Screenshot of the result for Case 4 Program A:

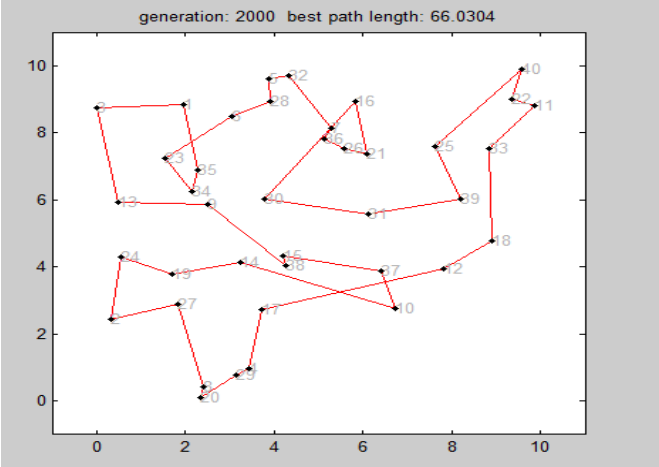


Figure 4-1-1-F1: Best path length graph of Program A with number of generation 2000 (first execute)

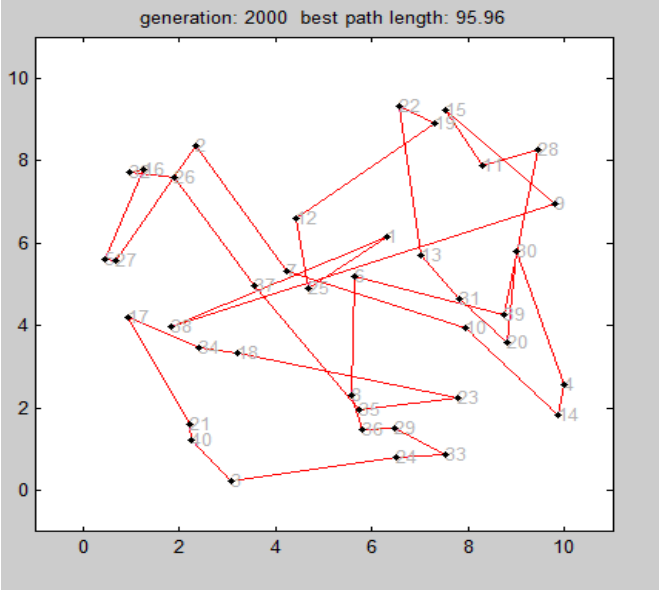


Figure 4-1-1-F2: Best path length graph of Program A with number of generation 2000 (second execute)

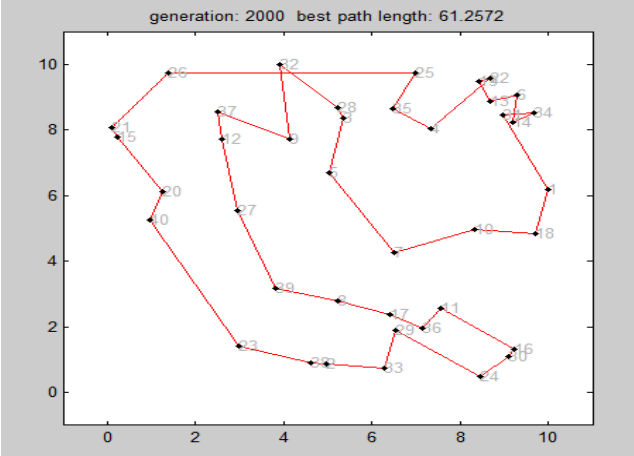


Figure 4-1-1-F3: Best path length graph of Program A with number of generation 2000 (third execute)

Program B:

Number of Case	First Execute (Best Path Length)	Second Execute (Best Path Length)	Third Execute (Best Path Length)	Average (Best Path Length)
Case 1: 500	98.61	110.07	98.86	102.51
Case 2: 1000	54.48	76.65	66.55	65.89
Case 3: 1500	54.37	60.51	73.35	62.74
Case 4: 2000	60.95	59.51	55.71	58.72

Table 4-1-3 Result of best path length by testing different numbers of generation from Program B

Screenshot of the result for Case 4 Program B:

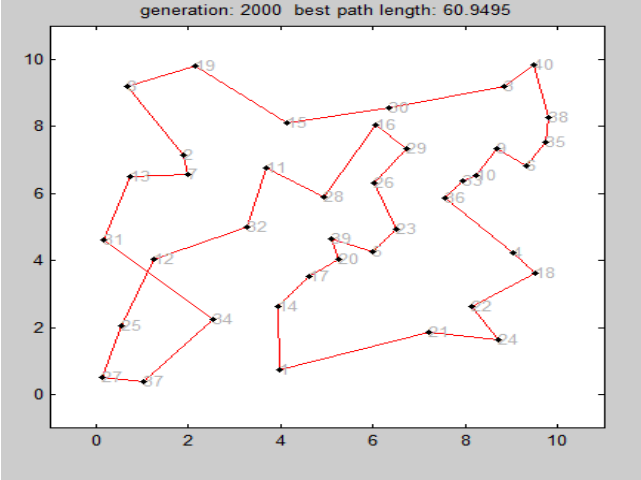


Figure 4-1-2-F1: Best path length graph of Program B with number of generation 2000 (first execute)

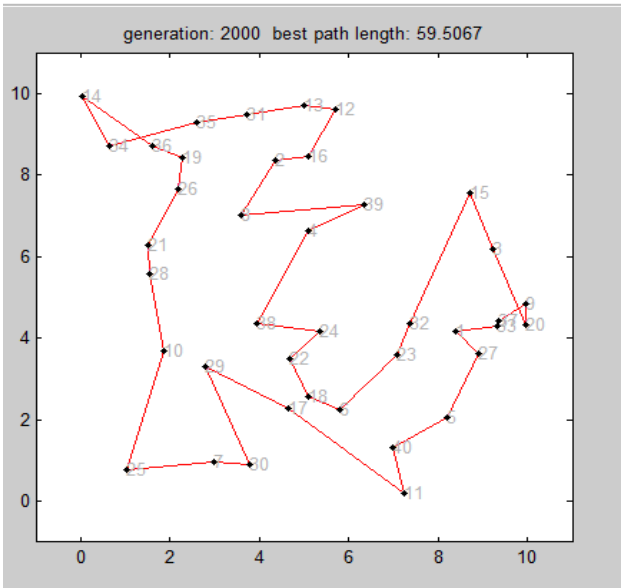


Figure 4-1-2-F2: Best path length graph of Program B with number of generation 2000 (second execute)

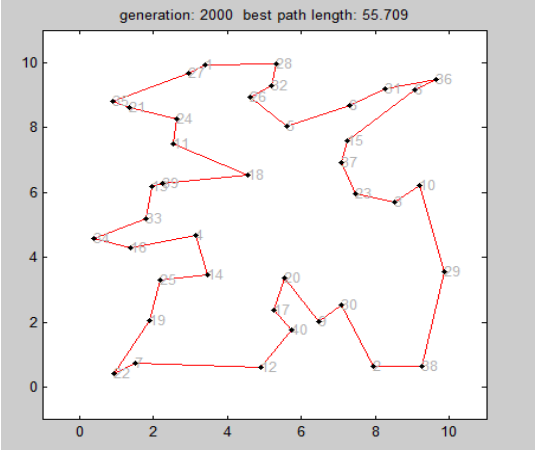


Figure 4-1-2-F3: Best path length graph of Program B with number of generation 2000 (third execute)

The Comparison of the average best path length between Program A and Program B for 4 Cases

Number of cases	Program A Average Best Path Length	Program B Average Best Path Length	Differentiate between average of Program A and B (A – B)	Percentage (%) of the differentiate of Program A and B $(\frac{(A-B)}{A}\ x\ 100\%)$
Case 1	111.93	102.51	9.42	8.42
Case 2	88.88	65.89	22.99	25.87
Case 3	66.48	62.74	3.74	5.63
Case 4	74.42	58.72	15.7	21.10

Table 4-1-4 Comparison Result between Program A and B

Explanation:

The lesser of the best path length, the better and optimize the result for TSP. TSP is to searching the shortest of path length with traveling all the cities and visits all the cities only in 1 time. By comparing both of the programs, with set the certain of number of generation, within the generation, how much it can search for the best path length result.

After the result sampling collection, we can see that each of the test case by using different number of generations of program B getting less best path length than the program A. This mean program B can perform more optimum shortest path result than the program A. As the number of generation increase, the best path length will decrease until its optimum and it's the lowest path length between them. By comparing the result from the average table, the average best path length of program B is getting much lower than program A.

For the case 1, the program B is getting lesser best path length 8.42% from the program A. That's mean program B is getting improve by 8.42% comparing to program A. Program B in Case 2 and case 4 are getting improved much higher than other cases, which are 25.87% and 21.10% respectively.

In overall of the Testing A using different number of generations, program B is getting the lesser best path length than the program A. This mean that program B has the improvement on its optimization and will get the solution faster and best path length in the less number of generation than program A.

4-2 Testing B: Number of cities: 50 - 80, Population Size: 3000, Number of generation: 1500, Area size: 10 x 10

Properties	Program A	Program B
Number of generation	1500	1500
Population Size	3000	3000
Area Size	10 x 10	10 x 10
Numbers of Cities	Case 1: 50 Case 2: 60 Case 3: 70 Case 4: 80	Case 1: 50 Case 2: 60 Case 3: 70 Case 4: 80

Table 4-2-1 The List of the properties to be tested in the Genetic Algorithm TSP Program

The result for both will be shown as separate table below:

Program A:

Numbers of Cities	First Execute (Best Path Length)	Second Execute (Best Path Length)	Average (Best Path Length)
Case 1: 50	131.32	142.77	137.05
Case 2: 60	178.19	165.31	171.75
Case 3: 70	211.89	227.37	219.63
Case 4: 80	253.46	248.77	251.16

Table 4-2-2 Result of the best path length by testing different numbers of cities for Program A

Screenshot of the result for Case 4 Program A:

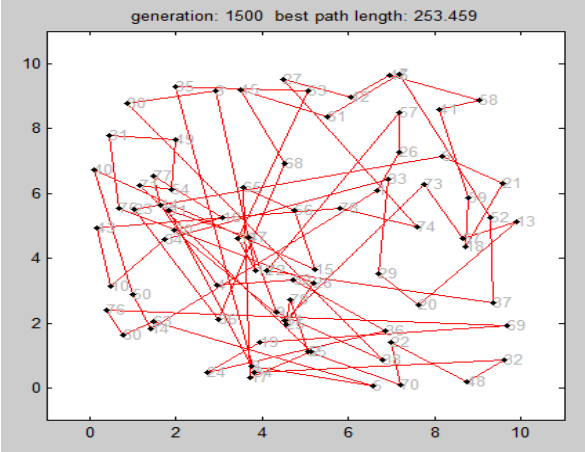


Figure 4-2-1-F1: Best path length graph of Program A with number of cities 80 (first execute)

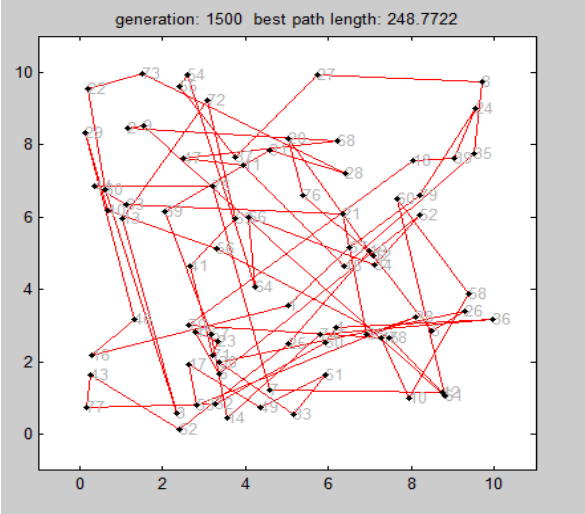


Figure 4-2-1-F2: Best path length graph of Program A with number of cities 80 (second execute)

Program B:

Numbers of Cities	First Execute (Best Path Length)	Second Execute (Best Path Length)	Average (Best Path Length)
Case 1: 50	78.14	63.04	70.59
Case 2: 60	152.81	179.19	166.00
Case 3: 70	204.21	196.03	200.12
Case 4: 80	244.72	228.55	236.64

Table 4-2-3 Result of the best path length by testing different numbers of cities for Program B

Screenshot of the result for Case 4 Program B:

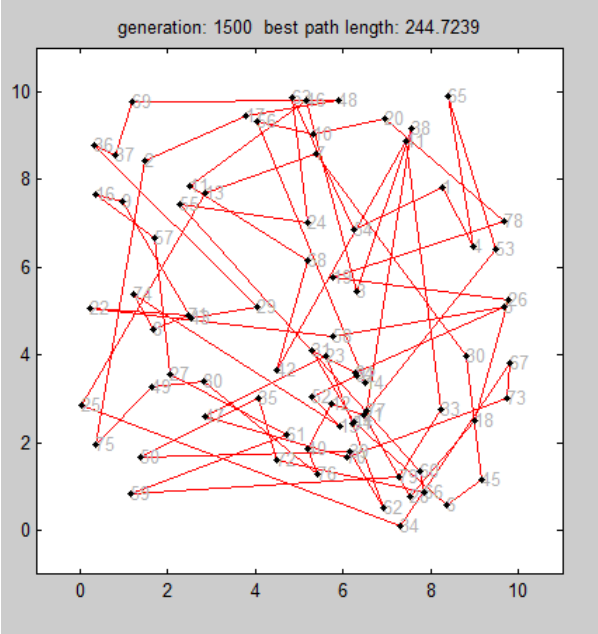


Figure 4-2-2-F1: Best path length graph of Program B with number of cities 80 (first execute)

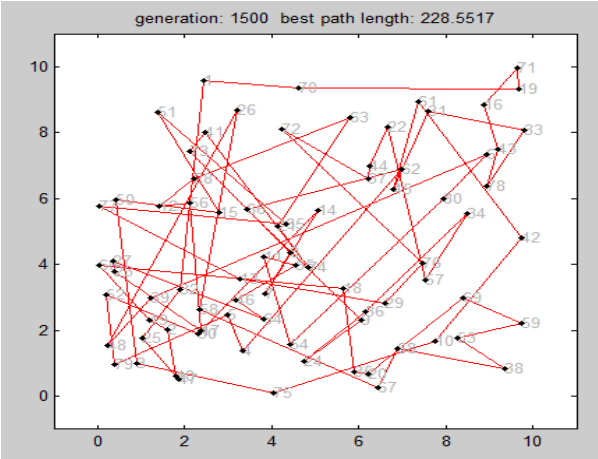


Figure 4-2-2-F2: Best path length graph of Program B with number of cities 80 (second execute)

The Comparison of the average best path length between Program A and Program B for 4 Cases

Number of cases	Program A Average Best Path Length	Program B Average Best Path Length	Differentiate between average of Program A and B (B – A)	Percentage (%) of the differentiate of Program A and B ($\frac{(A-B)}{A}\ x\ 100\%$)
Case 1	137.05	70.59	66.46	48.49
Case 2	171.75	166.00	5.75	3.35
Case 3	219.63	200.12	19.51	8.88
Case 4	251.16	236.64	14.52	5.78

Table 4-2-4 Comparison Result between Program A and B

Explanation:

When the number of cities increases, the path length between the numbers of cities will increase as will. The purpose of increasing the number of cities is that I need to see the program's behavior how to cope as the number of cities increase, the computation path length between comparing each node will increase, the complexity will increase and time consuming increase as the number of cities increase.

The most dramatically different between the program A and program B will be in case 1, number of cities 50. Program A is getting not optimum result than the program B. In program B, the result has improves the optimum by 48.49% comparing with program A.

As the number of cities become larger, the time execution of the program become higher because of the computation distance between the number of cities and the complexity of the comparison between each population. For both of the programs, the best path length result increase as the number of cities increase.

4-3 Testing C: Number of cities: 40, Population Size: 3000, Number of generation: 1500, Area size: 15 x 15, 20 x 20, and 25 x 25

Properties	Program A	Program B
Number of generation	1500	1500
Population Size	3000	3000
Number of cities	40	40
Area Size	Case 1: 15 x 15 Case 2: 20 x 20 Case 3: 25 x 25	Case 1: 15 x 15 Case 2: 20 x 20 Case 3: 25 x 25

Table 4-3-1 The List of the properties to be tested in the Genetic Algorithm TSP Program

The result for both will be shown as separate table below:

Program A:

Area Size	First Execute (Best Path Length)	Second Execute (Best Path Length)	Average (Best Path Length)
Case 1: 15 x 15	155.58	158.33	156.96
Case 2: 20 x 20	208.55	241.07	224.81
Case 3: 25 x 25	162.46	202.69	182.58

Table 4-3-2 Result of the best path length by testing different area size for Program A

Screenshot of the result for Case 3 Program A:

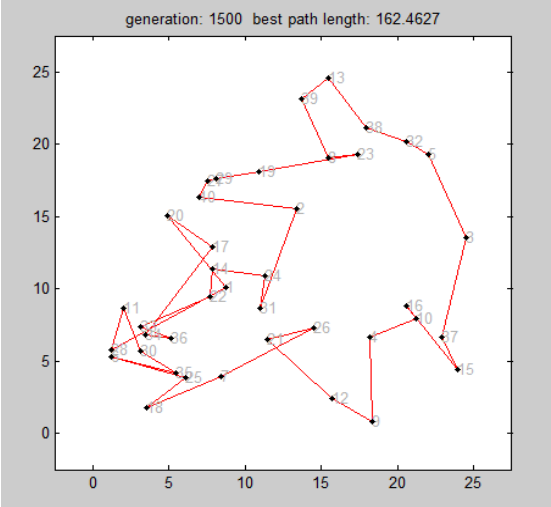


Figure 4-3-1-F1: Best path length graph of Program A with area size 25 x 25 (first execute)

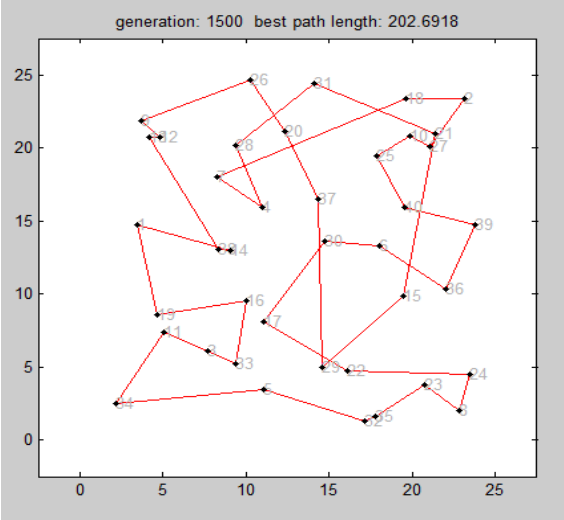


Figure 4-3-1-F2: Best path length graph of Program A with area size 25 x 25 (second execute)

Program B:

Area Size	First Execute (Best Path Length)	Second Execute (Best Path Length)	Average (Best Path Length)
Case 1: 15 x 15	87.65	85.04	86.35
Case 2: 20 x 20	116.87	169.97	143.42
Case 3: 25 x 25	188.38	162.05	175.22

Table 4-3-3 Result of the best path length by testing different area size for Program B

Screenshot of the result for Case 3 Program B:

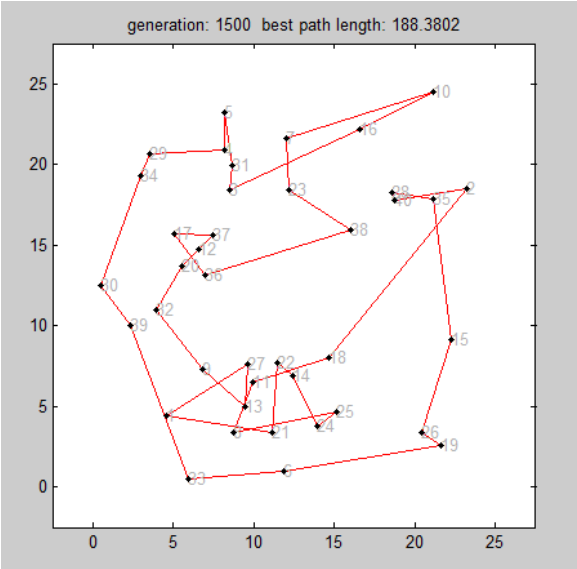


Figure 4-3-2-F1: Best path length graph of Program B with area size 25 x 25 (first execute)

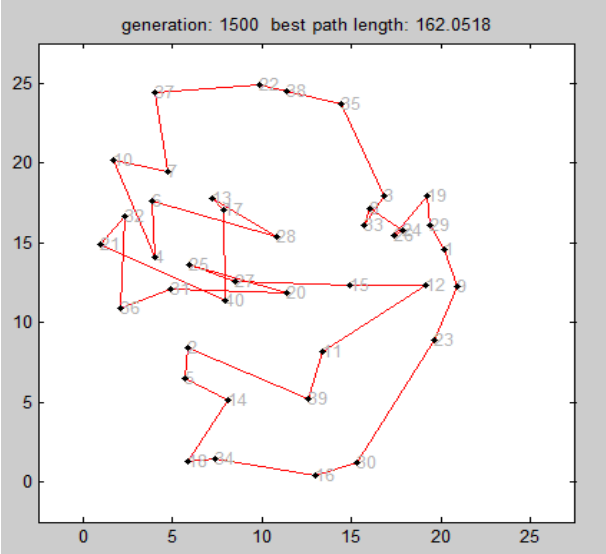


Figure 4-3-2-F2: Best path length graph of Program B with area size 25 x 25 (second execute)

The Comparison of the average best path length between Program A and Program B for 3 Cases

Number of cases	Program A Average Best Path Length	Program B Average Best Path Length	Differentiate between average of Program A and B (B – A)	Percentage (%) of the differentiate of Program A and B $(\frac{(A-B)}{A}\ x\ 100\%)$
Case 1	155.96	86.35	70.61	44.99
Case 2	224.81	143.42	81.39	36.2
Case 3	182.58	175.22	7.36	4.03

Table 4-3-4 Comparison Result between Program A and B

Explanation:

For this Testing C, I'm using different area size 15x15, 20x20 and 25x25 for the testing cases. I'm only using this 3 testing case because as the area size increase, the computation time for the program is increase and same goes to best length between the cities will increase. And by only getting the 3 result testing case, are enough to see the program's behavior. The area size is using the matrix nxn format and putting inside the number of cities by randomly in coordinate x and coordinate y.

From the result table, all the result from Program B is getting optimization result than the Program A. The first result by using area size 15x15 case 1, Program B improve by 44.99% compare with Program A. Program B is getting improve by 36.2% than program A in case 2, area size 20x20. And the case 3, program B is improve by 4.03% than program A by using area size 25x25.

Chapter 5: Conclusion

Criteria	Program A (Basic genetic algorithm program)	Program B (Modified genetic algorithm program)
Testing A: Different number of generation	Poor. The best length from all set of number of generation always higher than program B.	Excellent. Get the result better than program A in all different set of number generation
Testing B: Different number of cities	Poor. The best path length from all set of number of cities always higher than program B.	Excellent. Get the result better than program A in all different number of cities.
Testing C: Different area size	Poor. The best path length from all set of area size always higher than program B.	Excellent. Get the result better than program A in all different setoff area size.

Table 5-1 Conclusion result table of Program A and Program B

There is the fact that travelling salesman problem can be solving in different type of algorithm as I mention some methods already done by the other scientist. Each of the algorithms will getting the different result and compares their optimization under some circumstances. Same goes to the genetic algorithm, by combine different type of operators, the different situation of the travelling salesman problem, will definitely getting the different optimization result. Although there are many scientist done for the research on genetic algorithm, but there are still have the space improve to getting the optimum result when technology is modern day by day.

I'm using the same situation of the travelling salesman problem, same technique of creating the area size so each time randomly created cities node will be standard for each time of testing analysis. This fulfill one of the project objective that research and identify on TSP.

After being tested using the 3 different test cases, by different number of generations, different number of cities in a specific area size, and different area size of a specific number of cities, I can conclude that program B is getting more optimization result than the program A.

This proves that genetic algorithm can be improved by using the new mutation that inserting in the program B which getting more effective and optimization for TSP. And this also prove the another point that by combination different kind of operators will get the different optimization result. Some of the operators may not get better or even become worse if apply in the program.

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3, Year 4	**Study week no.: Week 5 , 10/02/2014**
Student Name & ID: LIONG KAH MEE 09ACB05366	
Supervisor: Ms. Manoranjitham a/p Muniandy	
Project Title: Intelligent Route Optimization for Travelling Salesman Problem	

1. WORK DONE

- Analyse the basic of genetic algorithm.

2. WORK TO BE DONE

- Decide on which operators to use in the implement version of genetic algorithm.
- Implement the system of own version of genetic algorithm.
- Write the report on Genetic algorithm.

3. PROBLEMS ENCOUNTERED

- Need time to test and figure out which operators to use in genetic algorithm to optimise the result.

4. SELF EVALUATION OF THE PROGRESS

- Getting more familiar on the type of the operators in Genetic algorithm.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3, Year 4	**Study week no.: Week 7 , 24/02/2014**
Student Name & ID: LIONG KAH MEE 09ACB05366	
Supervisor: Ms. Manoranjitham a/p Muniandy	
Project Title: Intelligent Route Optimization for Travelling Salesman Problem	

1. WORK DONE

- Done for the example MATLAB program.
- Decide on which operators can be implementing to increase the optimization.

2. WORK TO BE DONE

- Implement the decided operators so it can be functional on the program.

3. PROBLEMS ENCOUNTERED

- Need time to test and implement on operators to use in genetic algorithm to optimise the result.

4. SELF EVALUATION OF THE PROGRESS

- Getting more familiar on the type of the operators in Genetic algorithm.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3, Year 4	**Study week no.: Week 8 , 03/03/2014**
Student Name & ID: LIONG KAH MEE 09ACB05366	
Supervisor: Ms. Manoranjitham a/p Muniandy	
Project Title: Intelligent Route Optimization for Travelling Salesman Problem	

1. WORK DONE

- Done on the implementing of the program.
- Getting on the result that better than the original previous version.

2. WORK TO BE DONE

- Testing on few criteria to make sure getting the optimal result every time.
- Testing report need to be done.

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

- Getting to know on how to evaluate and testing a program.

_____ _____

Supervisor's signature Student's signature

FINAL YEAR PROJECT BIWEEKLY REPORT

(Project I / Project II)

Trimester, Year: Trimester 3, Year 4	**Study week no.: Week 9 , 13/03/2014**
Student Name & ID: LIONG KAH MEE 09ACB05366	
Supervisor: Ms. Manoranjitham a/p Muniandy	
Project Title: Intelligent Route Optimization for Travelling Salesman Problem	

1. WORK DONE

- Done for the analysis result and collecting result sampling.

2. WORK TO BE DONE

- Analyse the result and get the conclusion.
- Finish the thesis report

3. PROBLEMS ENCOUNTERED

- Take times to reference the resources on internet to make sure to get the correct explanation under some circumstances.

4. SELF EVALUATION OF THE PROGRESS

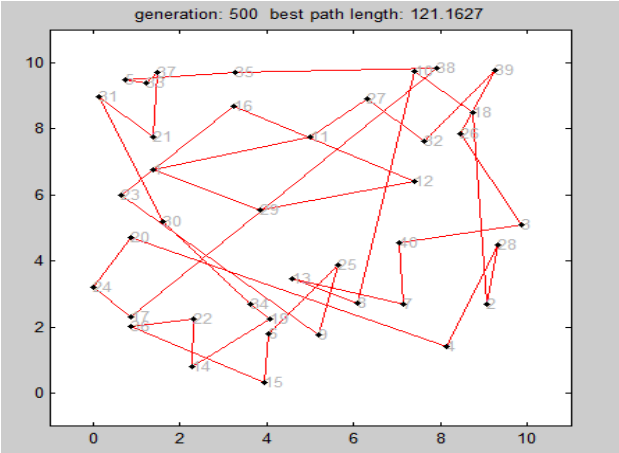
- Gain more knowledge on how to do the proper way of collecting sampling and analyse by the result table.

_____ _____

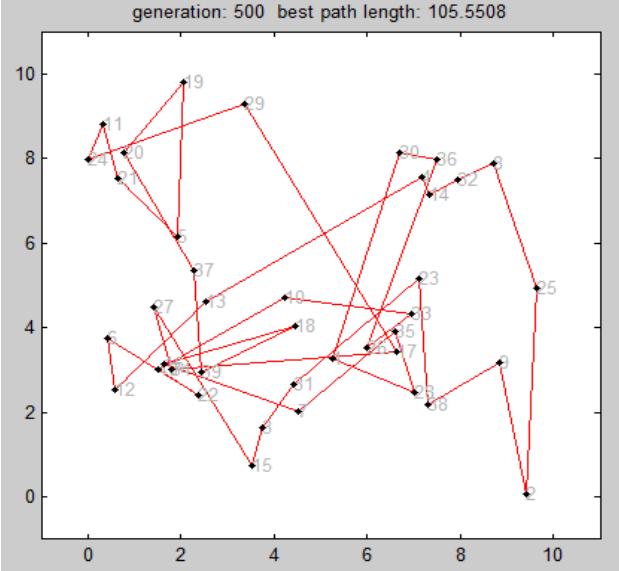
Supervisor's signature Student's signature

Appendixes

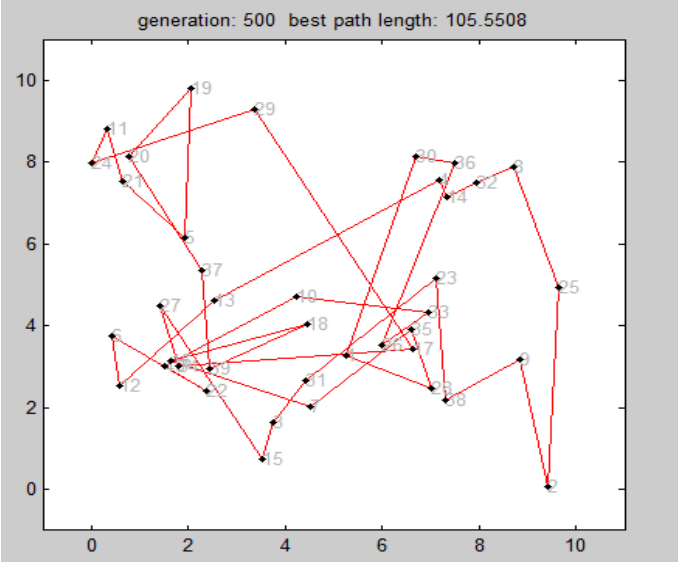
Screenshots for the Testing A Case 1 (number of generation=500):



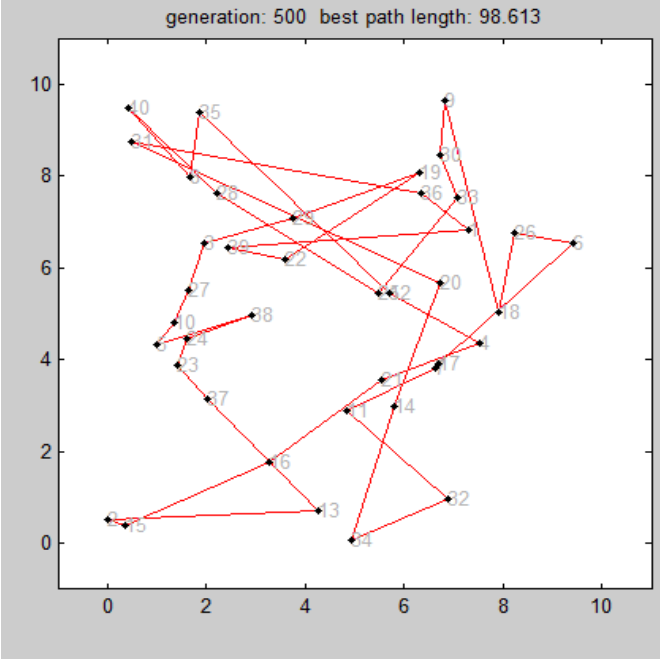
Best path length graph of Program A with number of generation 500 (first Execute)



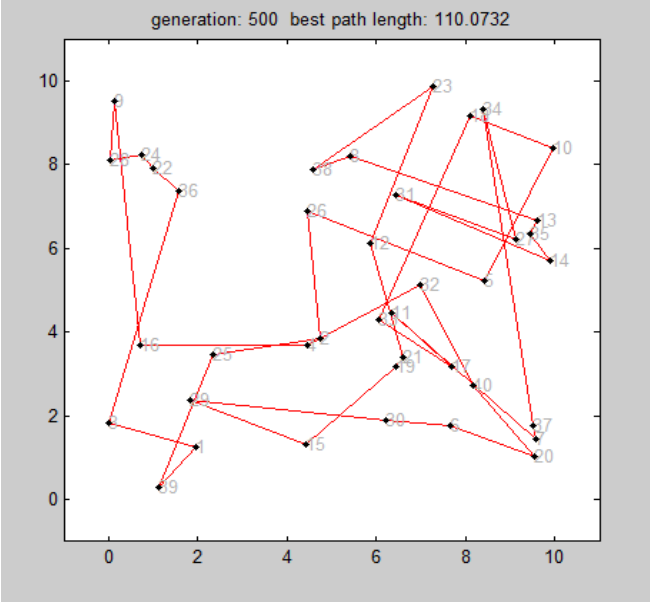
Best path length graph of Program A with number of generation 500 (second execute)



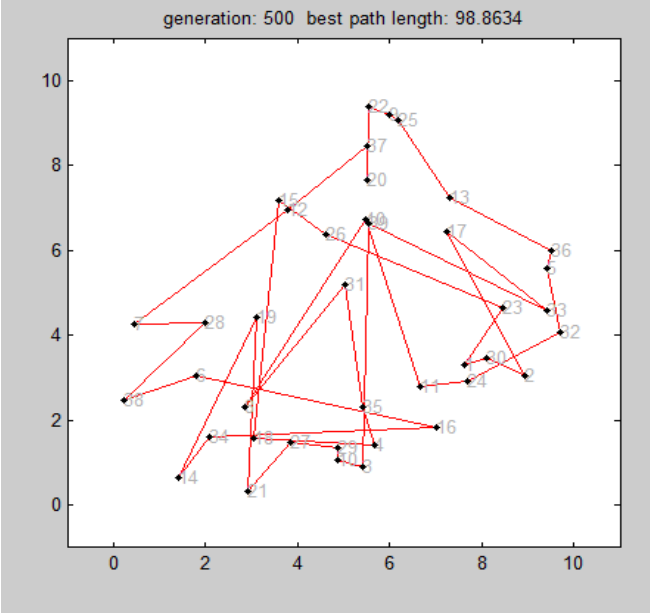
Best path length graph of Program A with number of generation 500 (third execute)



Best path length graph of Program B with number of generation 500 (first execute)

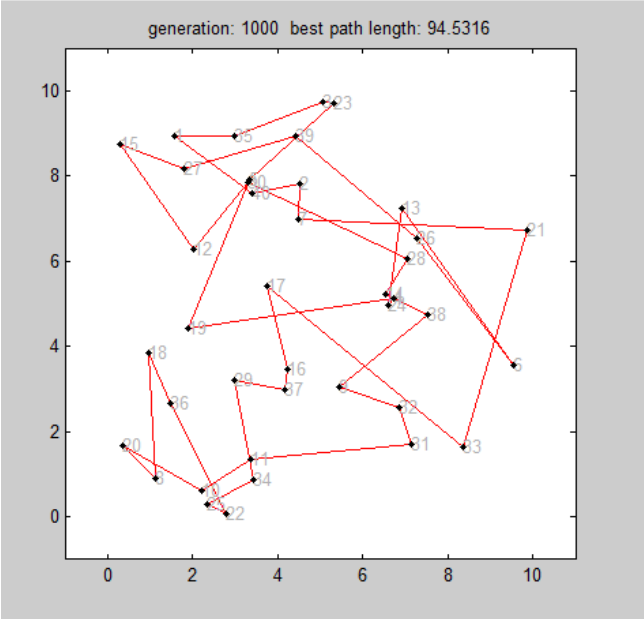


Best path length graph of Program B with number of generation 500 (second execute)

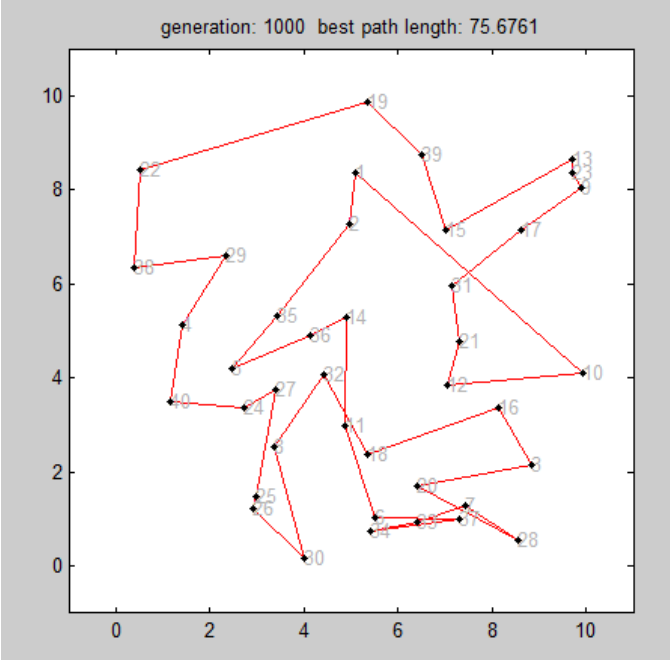


Best path length graph of Program B with number of generation 500 (Third execute)

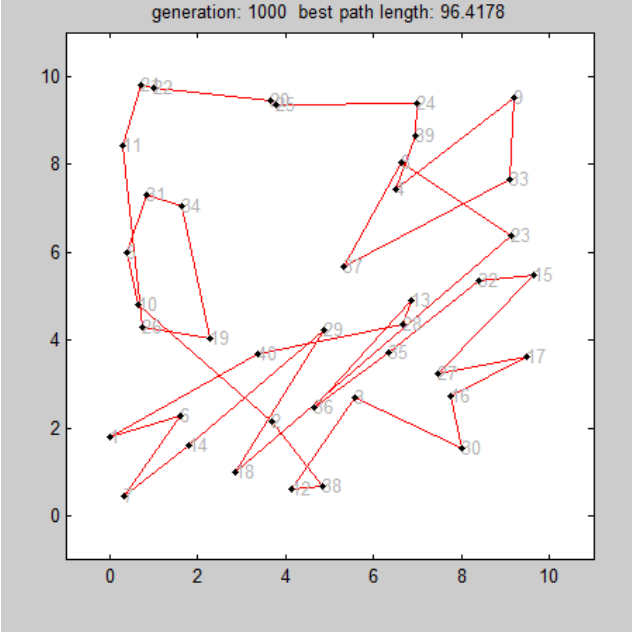
Screenshots for the Testing A Case 2 (number of generation=1000):



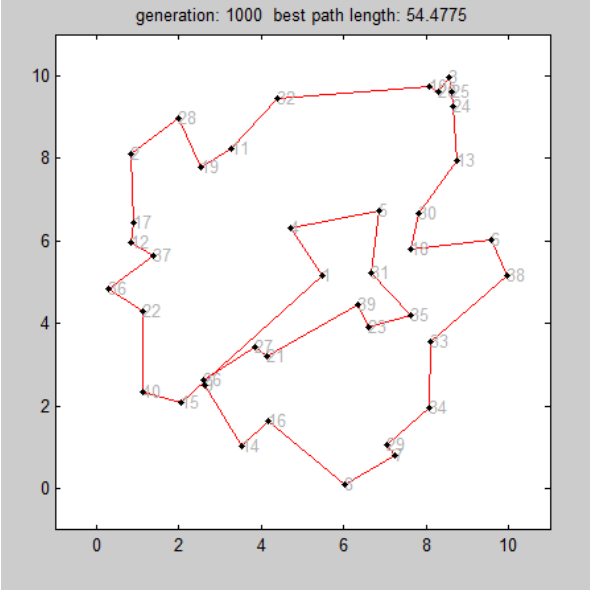
Best path length graph of Program A with number of generation 1000 (first execute)



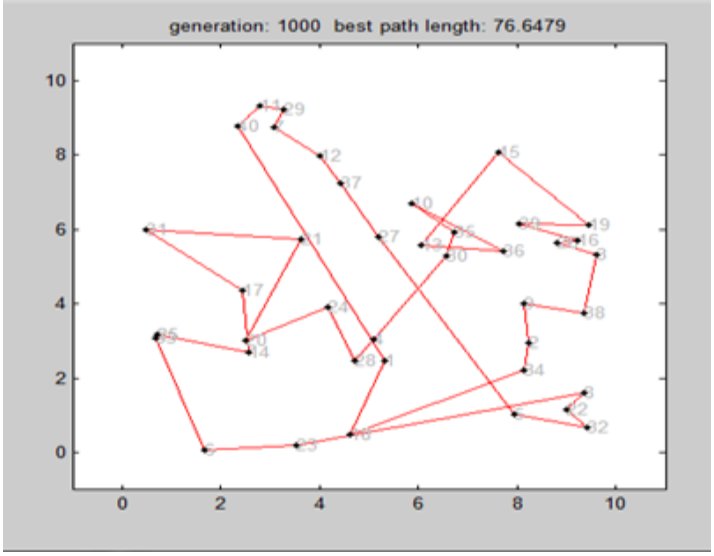
Best path length graph of Program A with number of generation 1000 (second execute)



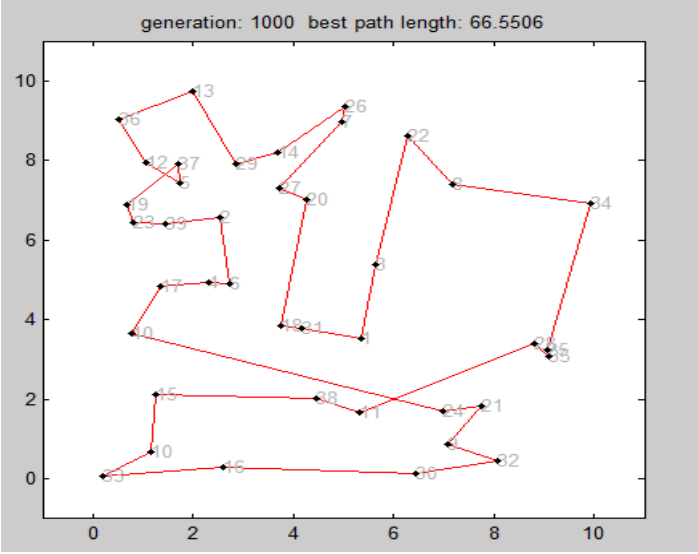
Best path length graph of Program A with number of generation 1000 (third execute)



Best path length graph of Program B with number of generation 1000 (first execute)

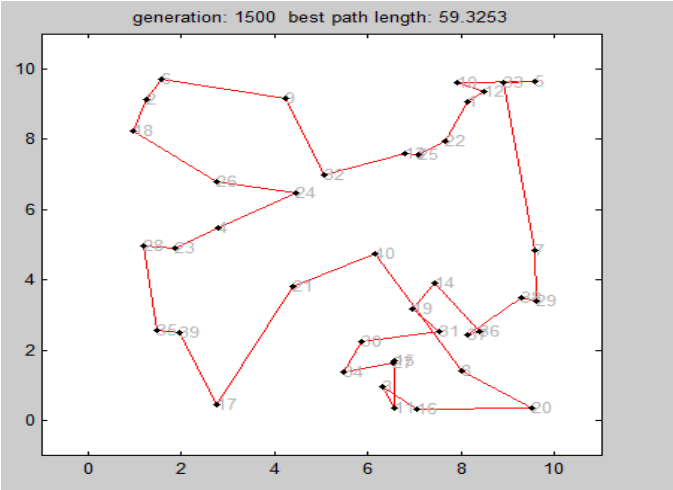


Best path length graph of Program B with number of generation 1000 (second execute)

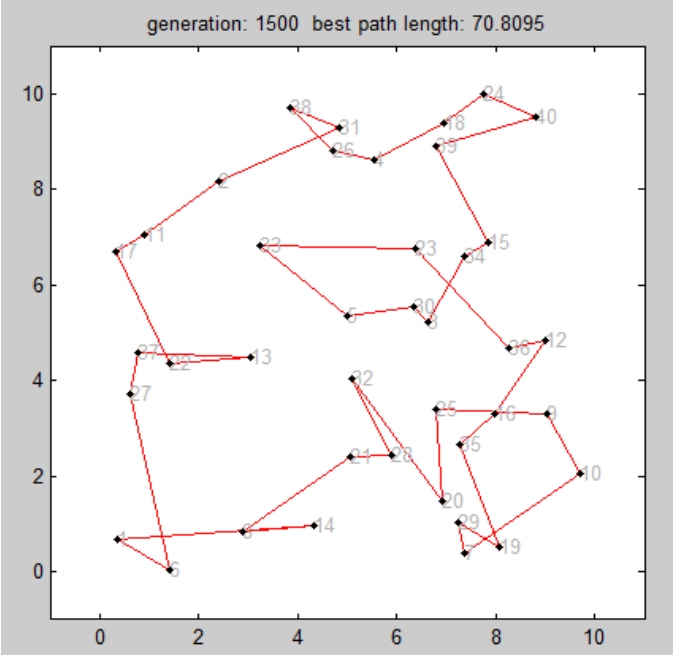


Best path length graph of Program B with number of generation 1000 (third execute)

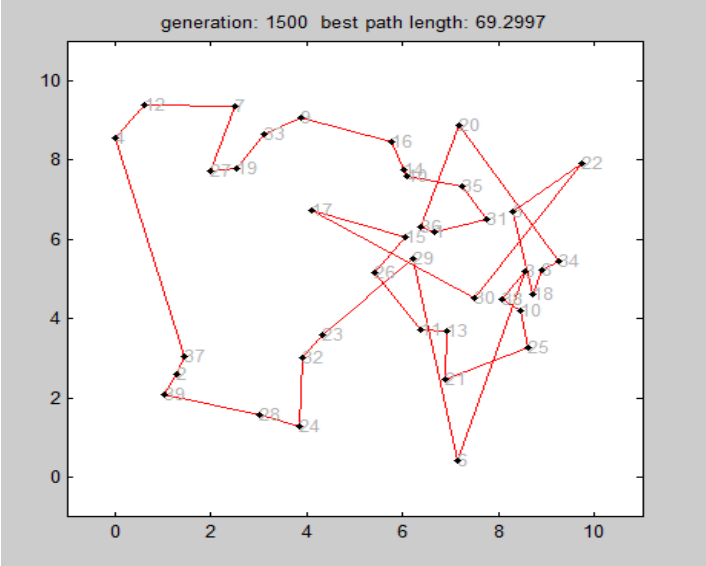
Screenshots for the Testing A Case 3 (number of generation=1500):



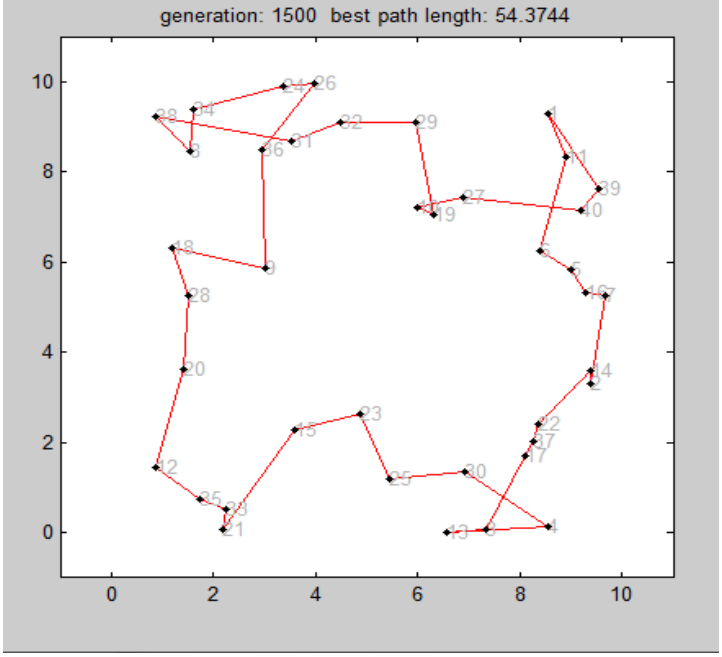
Best path length graph of Program A with number of generation 1500 (first execute)



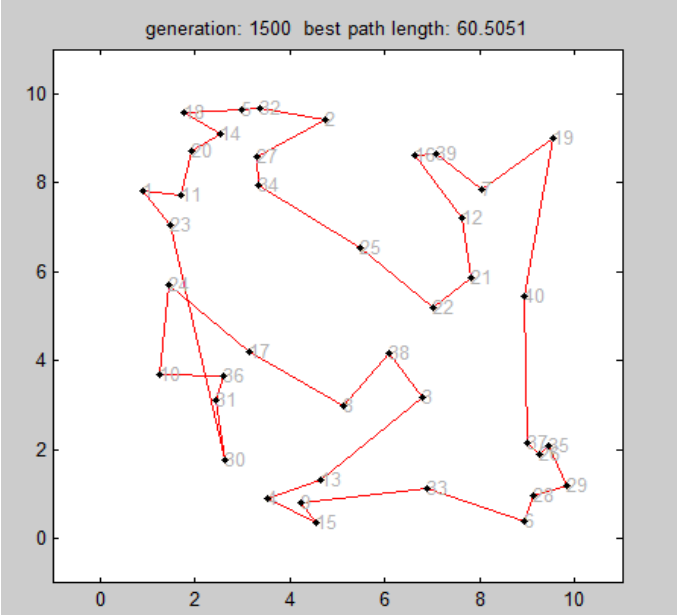
Best path length graph and of Program A with number of generation 1500 (second execute)



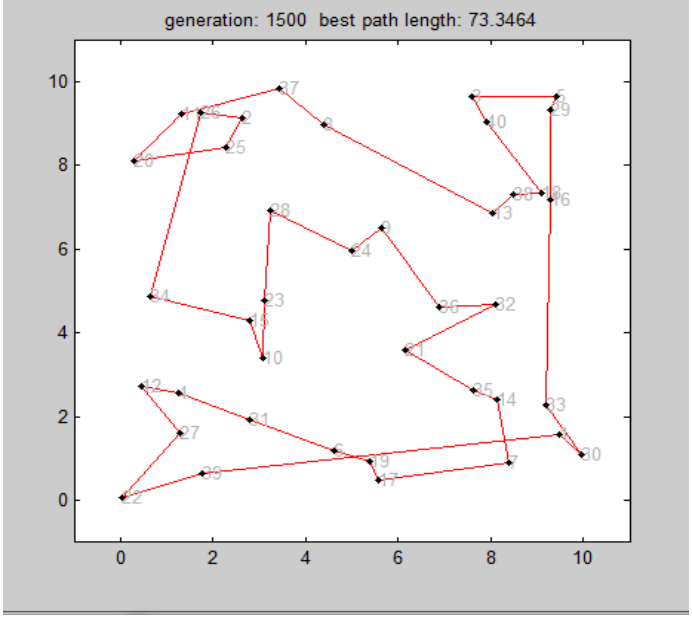
Best path length graph of Program A with number of generation 1500 (third execute)



Best path length graph of Program B with number of generation 1500 (first execute)

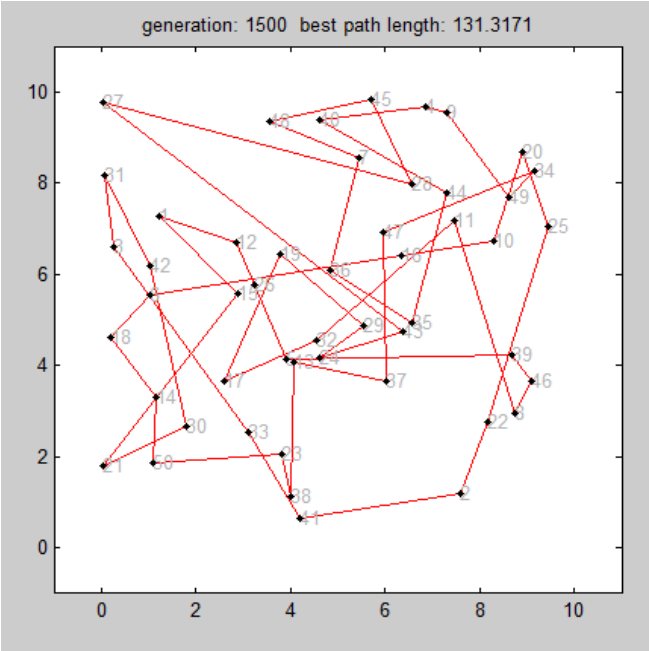


Best path length graph of Program B with number of generation 1500 (second execute)

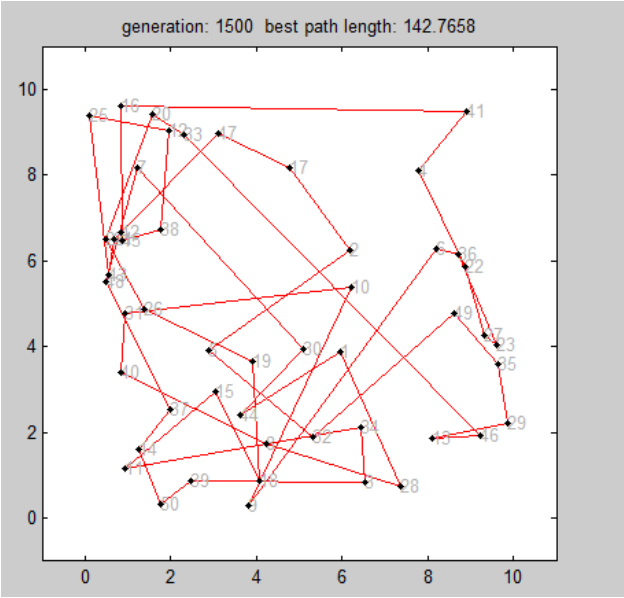


Best path length graph of Program B with number of generation 1500 (third execute)

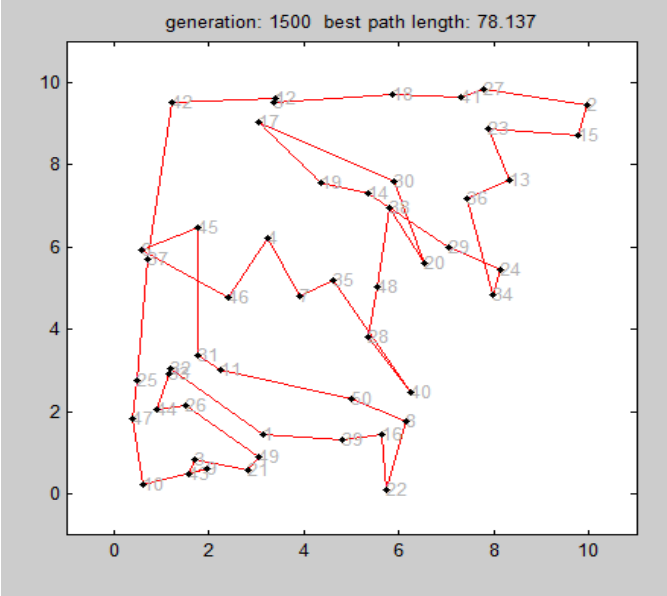
Screenshots for the Testing B Case 1 (number of cities=50):



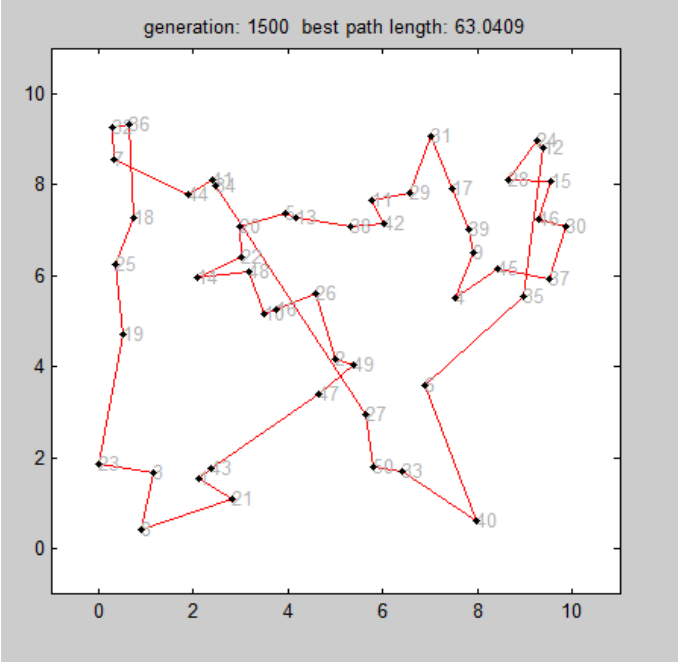
Best path length graph of Program A with numbers of cities 50 (first execute)



Best path length graph of Program A with number of cities 50 (second execute)

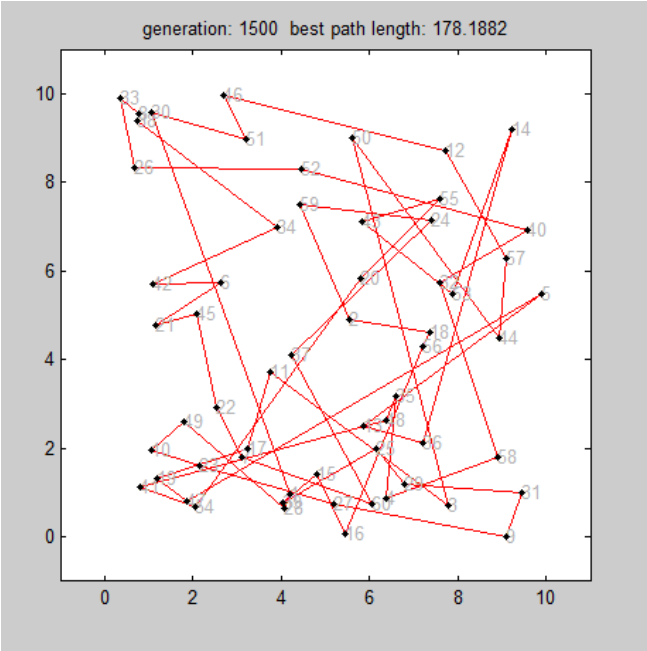


Best path length graph of Program B with numbers of cities 50 (first execute)

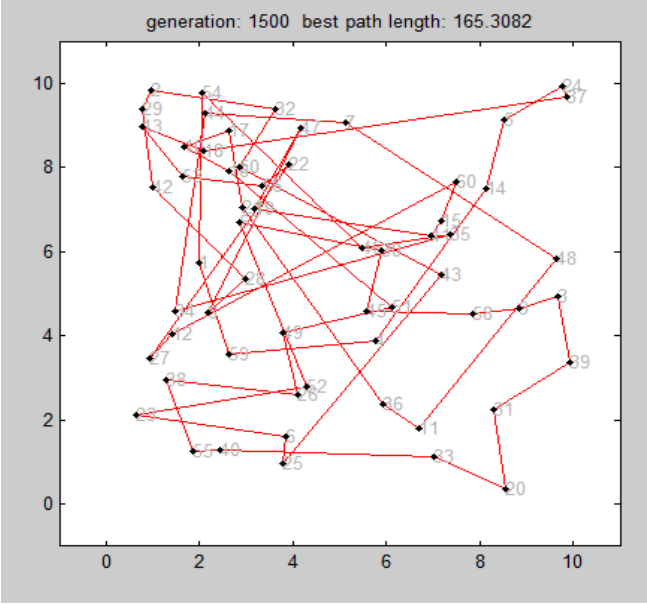


Best path length graph of Program B with numbers of cities 50 (second execute)

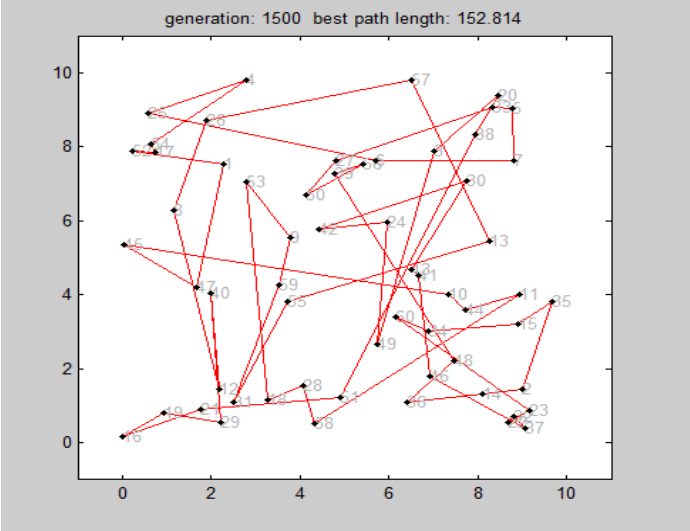
Screenshots for the Testing B Case 2 (number of cities=60):



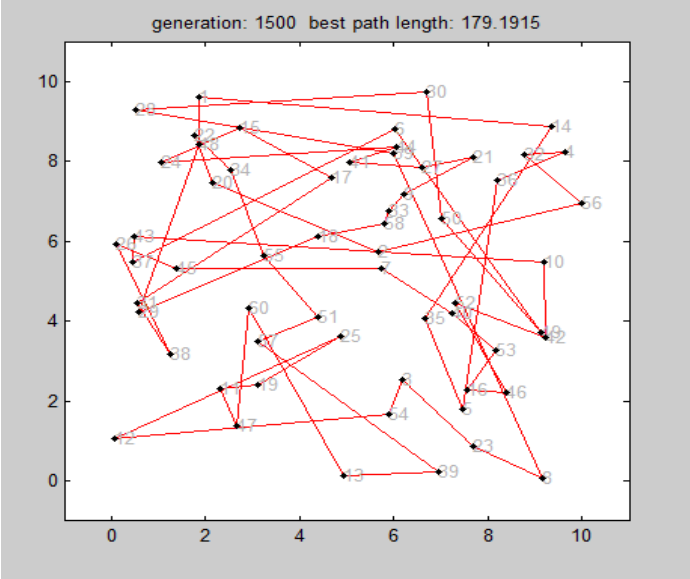
Best path length graph of Program A with numbers of cities 60 (first execute)



Best path length graph of Program A with numbers of cities 60 (second execute)

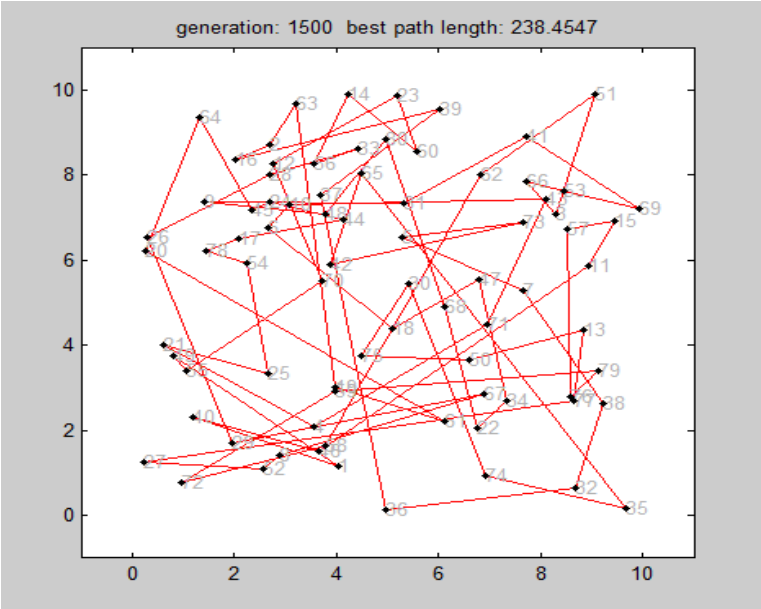


Best path length graph of Program B with numbers of cities 60 (first execute)

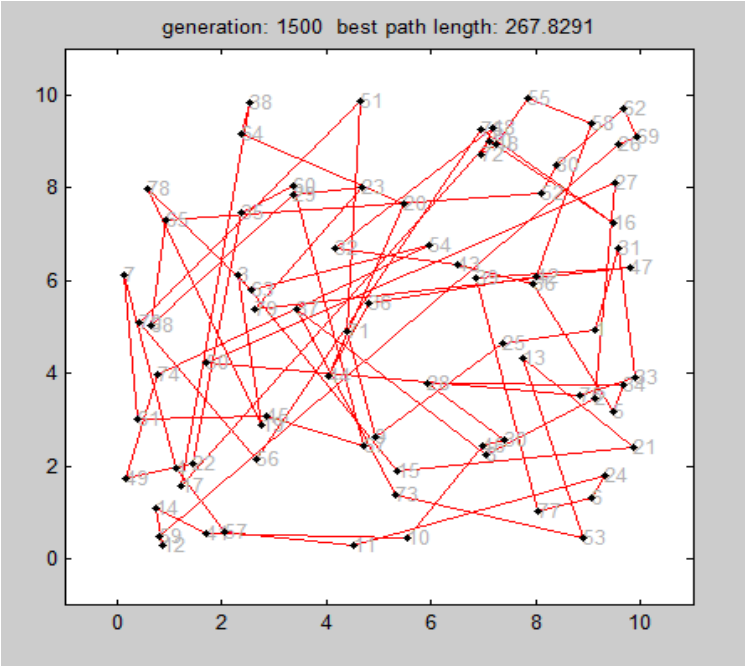


Best path length graph of Program B with numbers of cities 60 (second execute)

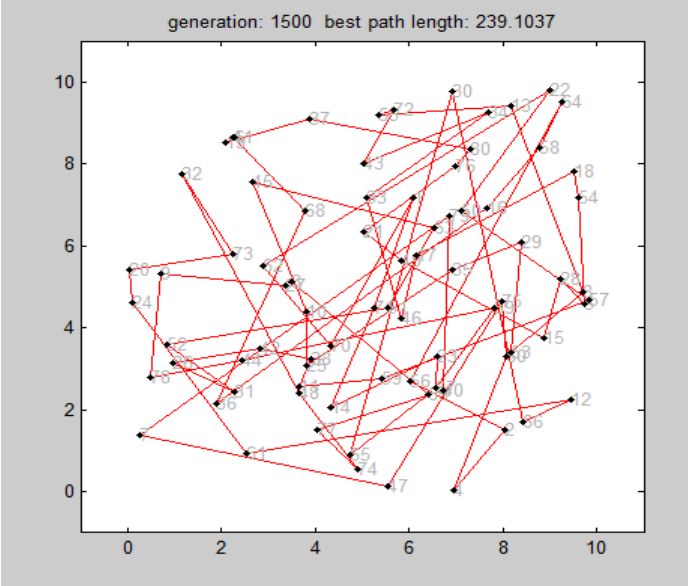
Screenshots for the Testing B Case 4 (number of cities=80):



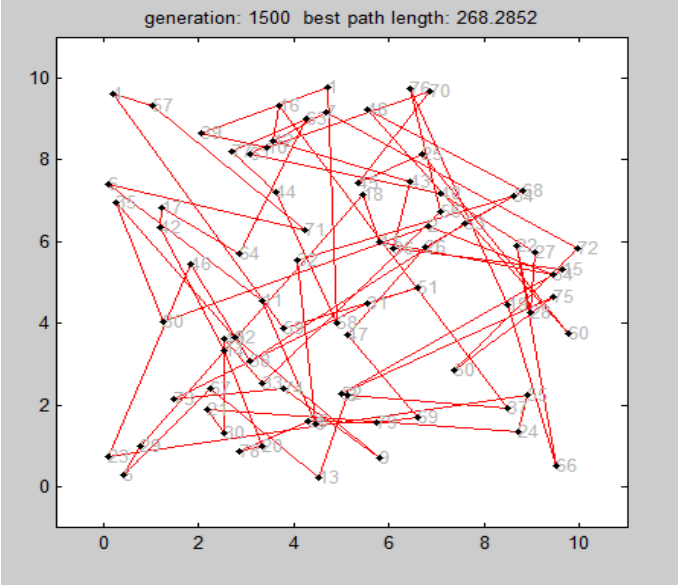
Best path length graph of Program A with number of cities 80 (first execute)



Best path length graph of Program A with number of cities 80 (second execute)

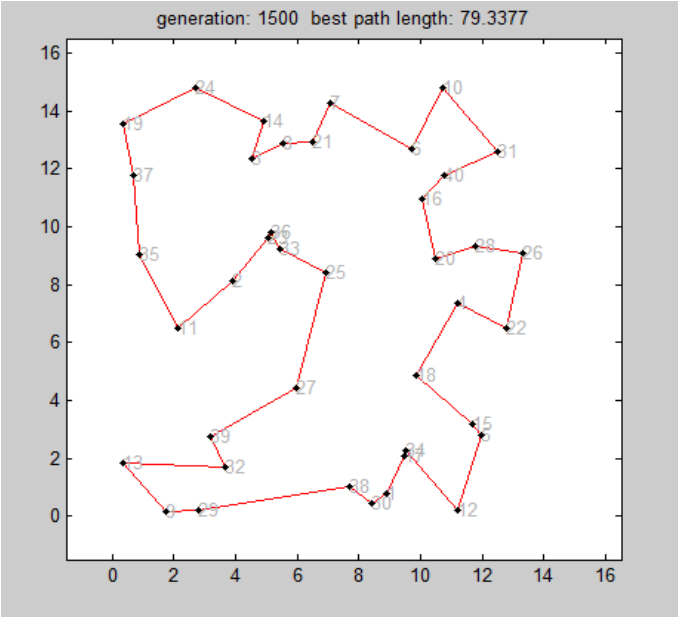


Best path length graph of Program B with number of cities 80 (first execute)

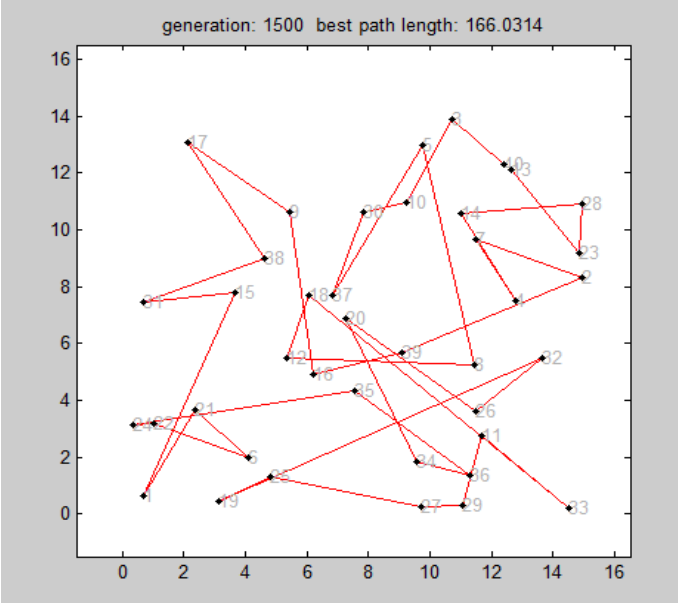


Best path length graph of Program B with number of cities 80 (second execute)

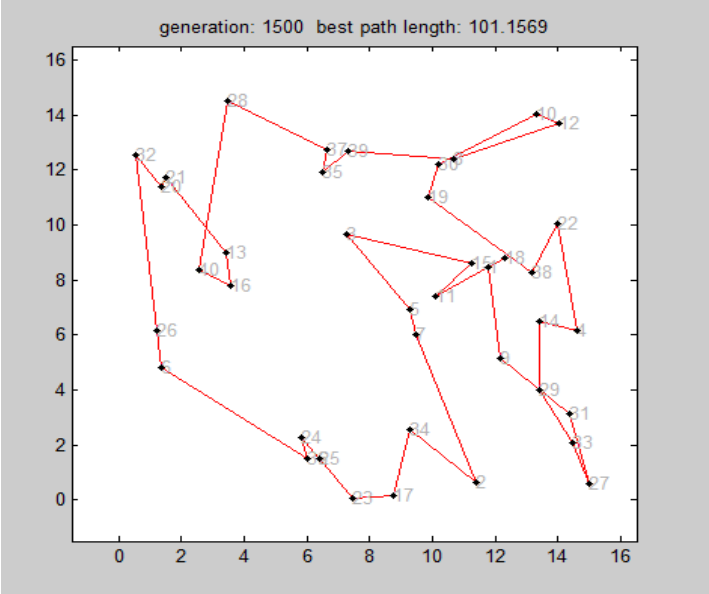
Screenshots for the Testing C Case 1 (area size=15x15):



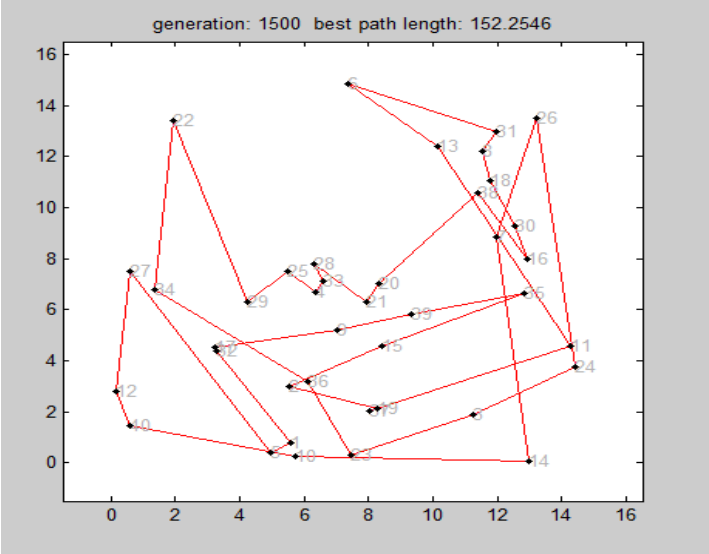
Best path length graph of Program A with area size of 15 x 15 (first execute)



Best path length graph of Program A with area size of 15 x 15 (second execute)

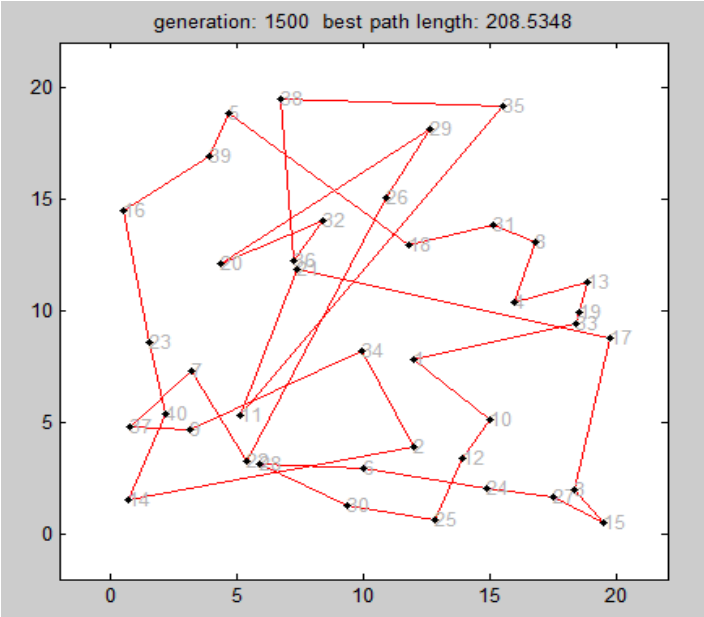


Best path length graph of Program B with area size of 15 x 15 (first execute)

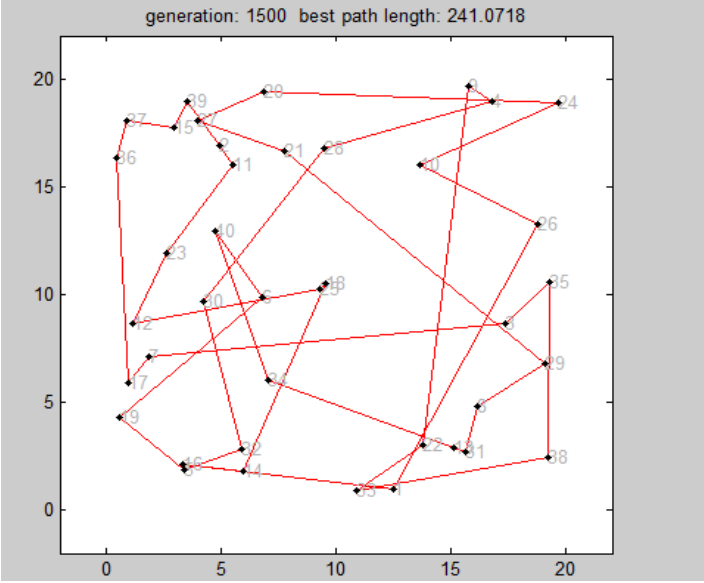


Best path length graph of Program B with area size of 15 x 15 (second execute)

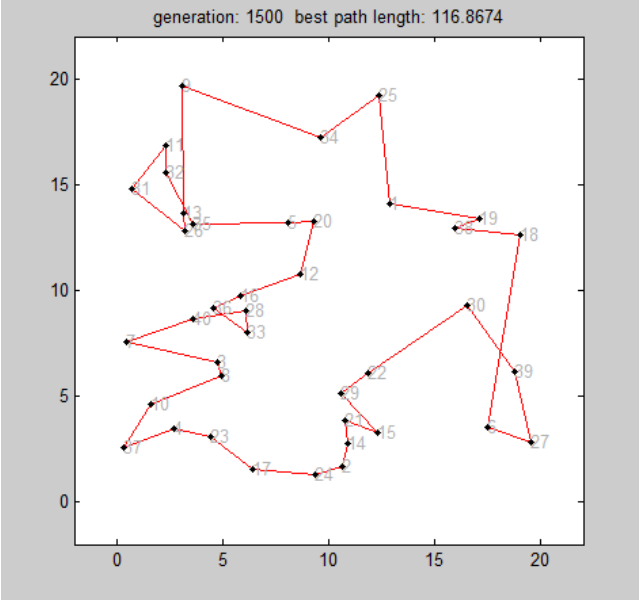
Screenshots for the Testing C Case 2 (area size=20x20):



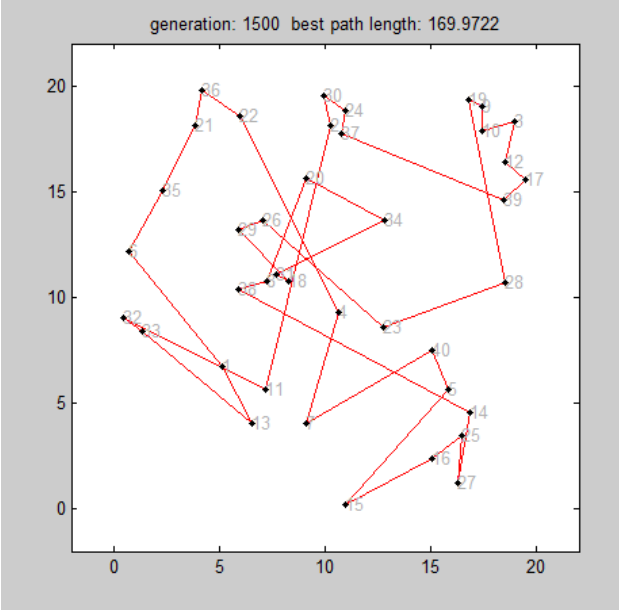
Best path length graph and color of city number of Program A with area size of 20 x 20 (first execute)



Best path length graph of Program A with area size of 20 x 20 (second execute)



Best path length graph of Program B with area size of 20 x 20 (first execute)



Best path length graph of Program B with area size of 20 x 20 (second execute)

Reference

Yingying Yu, Yan Chen & Taoying Li, 2011, '*A New Design of Genetic Algorithm*', 2011 Fourth International Joint Conference on Computational Science and Optimization, IEEE, pp 309 – 313.

Wang Hui, 2011, 'Comparison of several intelligent algorithms for solving TSP problem in industrial engineering', *System Engineering Procedia*, vol 4, 2012, pp.226-235.

Prototyping Methodology.

Available from: <http://www.wwetc.com/UoR/WhPp/Prototyping.html>

Geetha Ramani.R, Nishaa Bouvanasilan & Vasumathy Seenuvasan, 2009, '*A perspective view on Travelling Salesman Problem using Genetic Algorithm*', Nature & Biologically Inspired Computing 2009, pp. 356 – 361.

In-Chan Choi, Seong-In Kim, & Hak-Soo Kim, 2001, '*A genetic algorithm with a mixed region search for the asymmetric travelling salesman problem',* Computer & Operations Research 2003, pp 773 – 786.

Kylie, B 2000, *Genetic Algorithms and the Travelling Salesman Problem*. Harvey Mudd College.

Haupt, RL & Haupt SE 2004, *Practical Genetic Algorithm*, Johm Wiley & Sons, Inc., Hoboken, New Jersey.

Travelling Salesman Problem. Available from: <http://travellingsalesmanproblem.com/>

History of the TSP. Available from: < http://www.tsp.gatech.edu/history/index.html >. [January 2007]

Pan Junjie & Wang Dingwei, 2006, '*An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem*', Innovative Computing, Information and Control 2006, ICICIC'06, Vol 1, pp. 210–213.

Lishan Kang, Aimin Zhou, Bob McKay, Yan Li & Zhuo Kang, 2004, '*Benchmarking Algorithms for Dynamic Travelling Salesman Problem*', Evolutionary Computation 2004 CEC2004, Vol 2, pp. 1286 – 1292.

Nilesh Gambhava & Gopi Sanghami, 2003, '*Travelling Salesman Problem using Genetic Algorithm*'.

Ying-Hong Liao & Chuen-Tsia Sun (2001) *An Educational Genetic Algorithms Learning Tool*. Available from: < http://www.ewh.ieee.org/soc/es/May2001/14/Begin.htm>

Genetic Algorithm and Direct Search Toolbox User's Guide, 2004, MATLAB

Noraini Mohd Razali & John Geraghty, 2011, '*Genetic Algorithm Performance with Different Selection Strategies in Solving TSP*', World Congress on Engineering 2011 Vol II.

Alexander Stanoyevitch, 2011, Discrete Structure with Contemporary Applications, Chapman & Hall/CRC Press. [January 2011].

Matlab for TSP software, Maxim Vedenyov, 2011. Available from: http://www.mathworks.com/matlabcentral/fileexchange/31818-travelling-salesman-problem-with-genetic-algorithm

Li-Ying Wang, Jie Zhang, Hua Li, 2007, 'An improved genetic algorithm for TSP', in Machine Learning and Cybernetics: *proceedings of the sixth International Conference*, Hong Kong, pp. 925 – 928.

Francisco Chicano, Andrew M.Sutton, L.Darrell Whitley, Enrique Alba, 2013, 'Fitness Probability Distribution of Bit-Flip Mutation', in Discrete Mathematic: *Evolutionary Computation Journal (MIT press)*, arXiv:1309.2979[cs.DM].