

**Application of Genetic Algorithm in solving
Tourist Routing Problem**

By

Siti Sarah Binti Yusof

13915

Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information & Communication Technology)

SEPT 2013

Universiti Teknologi PETRONAS
Bandar Sri Iskandar,
31750 Tronoh,
Perak Darul Ridzuan, Malaysia.

CERTIFICATION OF APPROVAL

**APPLICATION OF GENETIC ALGORITHM IN SOLVING
TOURIST ROUTING PROBLEM**

By

Siti Sarah Binti Yusof

13915

Dissertation submitted to the
Information & Communication Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
Bachelor of Technology (Hons) (Information & Communication Technology)

Approved by,

(Mr. Izzatdin Abdul Aziz)

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SRI ISKANDAR,
31750, TRONOH, PERAK
SEPT 2013

CERTIFICATION OF ORIGINALITY

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

SITI SARAH BINTI YUSOF

ABSTRACT

Normally, tourist will experience dilemma in planning their tour route especially when they visited foreign country for the first time. Manually mapping the cities and searching the information on the Internet can be very exhaustive. Besides these, tourist also faced a dilemma on how to travel across different cities efficiently and at shortest distance. This can also be known as Tourist Routing Problem (TRP). TRP is a variance of Travelling Salesman Problem (TSP) which can defined by finding the optimal path to travel from point A to point B by going through the same place not more than twice at a shortest distance. After completing a thorough comparative study, the author decided to apply Genetic Algorithm (GA), which is one of the best heuristic solutions to date in solving TRP. A rapid-prototyping methodology had been chosen because the author can immediately alter the prototype if there are any changes in the requirements. An Android mobile application will be utilized as a platform to test the effectiveness of GA in solving TRP. To support this, simulation and experiments will be conducted to evaluate the performance and speedup of the algorithm. Besides focusing on finding the best shortest distance route to travel, this application will enable tourist to select places to visit according to their preferences and activities that will be happening at that particular place.

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to Allah S.W.T for His blessings and mercy that He instills in myself to face many challenges and thus complete this work for my Final Year Project (FYP).

Secondly, I am grateful that for the past 28 weeks, I had managed to complete this project with the guidance of Mr. Izzatdin Abdul Aziz. Under his supervision, he constantly encourages me to complete the project with outmost dedication and commitment. Besides, Mr. Izzatdin ensure that there is no gap between a student and a lecturer that made us, the students very comfortable to tell him any problem we faced while completing this project. I am grateful that I have a very understanding lecturer that always gives different opinion or perspective whenever we are discussing an issue or problem. Not to mention his dedication, I cannot be more thankful for his great support and believe in me that I can complete this project successfully. May Allah S.W.T always bless your for your kindness and commitment.

Together with this, I would like to thank my mother, Ms. Robiah Sariff and my family for understanding the late nights and the importance of completing this project in order for me to complete my degree.

Last but not least, I would like to acknowledge my friends and other personnel not mentioned above whom gave me such a great support in completing this project. I would also like to apologize for all the wrongdoings or problems arise. All of your kindness and cooperation are highly appreciated and will be fondly remembered.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL	i
CERTIFICATION OF ORIGINALITY	ii
ABSTRACT	iii
ACKNOWLEDGEMENT	iv
LIST OF FIGURES	vii
LIST OF TABLES	viii
ABBREVIATIONS	ix
 CHAPTER 1: INTRODUCTION	 1
<i>1.1 Background of The Study</i>	<i>1</i>
<i>1.2 Problem Statement</i>	<i>2</i>
1.2.1 Problem Identification	2
1.2.2 Significance of the Project	3
<i>1.3 Objectives and Scope of Study</i>	<i>4</i>
1.3.1 Objective	4
1.3.2 Scopes of Study	4
<i>1.4 Feasibilities of the Project</i>	<i>5</i>
1.4.1 Technical feasibility	5
1.4.2 Economic Feasibility	5
1.4.3 Operational Feasibility	5
1.4.4 Schedule Feasibility	5
 CHAPTER 2: LITERATURE REVIEW	 6
<i>2.1 What is Travelling Salesman Problem (TSP)?</i>	<i>6</i>
<i>2.2 What is Travelling Routing Problem (TRP)?</i>	<i>7</i>
<i>2.3 Malaysia as a Tourist Destination Place</i>	<i>7</i>
<i>2.4 Famous local attractions in Malaysia</i>	<i>9</i>
<i>2.5 Travel Destination Decision Making</i>	<i>10</i>
<i>2.6 Main application of TRP & TSP</i>	<i>11</i>
<i>2.8 What is Genetic Algorithm?</i>	<i>13</i>
<i>2.9 What is Android?</i>	<i>14</i>
<i>2.10 Why Android?</i>	<i>14</i>
<i>2.11 Comparative Study on Existing Mobile Application</i>	<i>15</i>
 CHAPTER 3: RESEARCH METHODOLOGY	 16
<i>3.1 Project Method & Activities</i>	<i>16</i>
3.1.1 Planning and Requirements Gathering	17
3.1.2 Analysis & Design	17
3.1.3 Prototype Development	17
3.1.4 Prototype Evaluation & Testing	18
3.1.5 Implementation	18

3.2.1 Overview of Project Activities	18
3.2.2 Qualitative Survey	19
3.3 <i>System Architecture</i>	19
3.4 <i>Requirement Analysis and Specification</i>	20
3.4.1 Genetic Algorithm	20
3.5 <i>Tools and Equipment</i>	21
3.5.1 Hardware	21
3.5.2 Software	21
3.6 <i>Key milestones & Gantt chart</i>	22
 CHAPTER 4 : RESULTS AND DISCUSSION	 23
4.1 <i>Genetic Algorithm Calculation for Shortest Distance Route</i>	23
4.2 <i>Eliminating the possibility of redundancy in path planning in solving TRP</i>	25
4.3 <i>Performance Evaluation of GA in solving TRP</i>	26
4.4 <i>Speedup of the GA in solving TRP with increasing number of cities selected</i>	27
4.5 <i>System Design</i>	29
 CHAPTER 5 : CONCLUSION	 32
5.1 <i>Conclusion</i>	32
5.2 <i>Recommendation</i>	32
 REFERENCES	 33
 APPENDICES	 35
Appendix 1: Gantt Chart	36
Appendix 2: Key Milestones	37
Appendix 3: Algorithm Codes in Java Language	38

LIST OF FIGURES

Figure 1.1 Mr. Steve’s scenario	2
Figure 1.2 Example of travel destination planning using TRP	3
Figure 2.1: The cluster of cities before and after going through Genetic Algorithm	13
Figure 3.1: Research Methodology Diagram	16
Figure 3.2: Overview of overall project activities	18
Figure 3.3 System Architecture for the Mobile Application	19
Figure 4.1 Genetic Algorithm cycle	23
Figure 4.2 Offspring route generated from crossover and mutation	24
Figure 4.3 Maps that consists of 20 cities	25
Figure 4.4 Shortest distance route for 5 cities with GA.	25
Figure 4.5 Shortest distance route for 10 cities with GA.	25
Figure 4.6 Shortest distance route for 15 cities with GA	25
Figure 4.7 Shortest distance route for 20 cities with GA	26
Figure 4.8 Performance Evaluation of GA in solving TRP with different sets of cities selected.	27
Figure 4.9 Speedup of GA in solving TRP with different sets of cities selected.	28
Figure 4.10 Welcome page of the application	29
Figure 4.11 Place of interest selection according to the categories	29
Figure 4.12 List of cities included in the Shopping Heaven category	29
Figure 4.13 Shortest distance calculation is projected to the user	30

LIST OF TABLES

Table 2.1 10 highest tourist nationalities that visited Malaysia in 2012	8
Table 2.2: Tourist attractions in Malaysia according to categories	10
Table 2.3 Possible solutions to TRP&TSP	13
Table 2.4 Important elements of Genetic Algorithms	14
Table 2.5: Existing mobile application that is related to the project	15
Table 3.1 Analysis and comparative study conducted before project development	19
Table 3.2 Important elements in Genetic Algorithm	20
Table 3.3 Constant parameters that need to be declared in Genetic Algorithm	21

ABBREVIATIONS

3D	3 – Dimensional
AR	Augmented Reality
FYP	Final Year Project
GA	Genetic Algorithm
GPS	Global Positioning System
RAM	Random Access Memory
TRP	Tourist Routing Problem
TSP	Travelling Salesman Problem
UTP	Universiti Teknologi PETRONAS
VRP	Vehicle Routing Problem

CHAPTER 1

INTRODUCTION

1.1 Background of The Study

Hwang *et. al.* (2008) reported that millions of tourist utilized information provided by more than 70 thousands travel-related website when the search “Travel” keyword in Google to plan for their holiday. Most of the tourist faced problem on how to maximized number of tourist attractions that they can visit in their itinerary. They need to consider many factors such as time, cost, interest and mode of transportation before deciding which places to visit.

When tourist faced dilemma on which place to travel during their visit, they prioritize on how to minimize the total distance that need to be travel based on their interest. This dilemma is known as Tourist Routing Problem (TRP). It is a variance of Travelling Salesman Problem (TSP). This problem can be defined by, given N set of cities or places to be visited, how to travel from point A to point B by going through various cities with the best shortest distance possible. In this research, the author will be focusing on suggesting the best route to travel based on shortest distance by applying Genetic Algorithm (GA).

To test the effectiveness of GA in solving TRP, an Android application will be utilized as a platform. In order to proof the concept, the author will also evaluate the performance and speedup of the algorithm in solving TRP. This application target user will be foreign tourist that is visiting Malaysia for the first time. The application will also function as information center where tourist can get information about the places to visit such as the location, opening hours, and ongoing activities at that particular place. Tourist can also choose places that they would like to visit based on their preferences in the mobile application.

1.2 Problem Statement

1.2.1 Problem Identification

Given a scenario, Mr. Steve is an American tourist who comes to Malaysia for his summer holiday. This is his first visit to Malaysia and he is travelling alone for 4 days and 3 nights. Mr. Steve is a person who is very passionate about history and he would love to learn about the history of our unique multi- cultural country.



Figure 1.1 Mr. Steve's scenario

He did his own research on best places or tourist attractions to visit that suit his preference and he try to manually plan his itinerary. However, due to certain limitations, he cannot find the best route planning application on the Internet that can help him to arrange his tour based on his preference and at shortest travel distance possible.

Realistically, a normal tourist would like to travel through various places in Malaysia efficiently, the distance between each location will be random; therefore the time and cost taken to travel through each location will be different as well. Therefore, by applying TRP, tourist will be able to find the best route to travel from initial location A to location B through various

locations, shortest distance to travel will be able to help tourist to travel throughout Malaysia at their own leisure time.



Figure 1.2 Example of travel destination planning using TRP

Even though this problem can be solved with a map, pencil and ruler, it can be very exhaustive if there is more than 6 cities or locations that the tourist would like to visit. Besides, some tourist would also like to travel through various locations according to their preference and interest during their stay in Malaysia. Providing a selection of attractions segregated based categories such as shopping malls, museums, national parks, beaches and historical sites, these can assist tourist in making a better decision during their visit.

1.2.2 Significance of the Project

Researchers have been conducting many studies on the application of TSP in daily life. Many studies had been conducted in various areas such as detection of cancerous cells, computation of DNA, and vehicle-routing problem. In this project, the author will focus on solving TRP, on the scope of tourism and tourist routing problem. First, the significance of this project is to prevent redundancy in path planning by eliminating the possibility of travelling across same city twice. Known as one of heuristic algorithm, the author will also evaluate the performance of GA in solving TRP by

conducting experiments and simulations as number of cities increases. With the results projected by the testing conducted, the author will also be able to measure the speedup of the application in suggesting the best shortest distance route for tourist to travel.

1.3 Objectives and Scope of Study

1.3.1 Objective

- **To study the effectiveness of Genetic Algorithm in solving TRP**

A study will be conducted on the Genetic Algorithm in order to proof the concept that this algorithm is effective in solving TRP even though is it's a heuristic algorithm. In this project, we will focus on the scope of tourism. Simulations and experiments will be conducted to measure the performance and speedup of the algorithm.

- **To implement GA in finding the shortest distance to travel across different cities without going through same city twice.**

The algorithm will be tested on an Android mobile application that will act as a platform to suggest the best shortest distance route for tourist to travel across different cities or places without going through the same place twice. The application will also enable tourist to select places that they would like to travel based on their preferences.

1.3.2 Scopes of Study

The scope of this research is to focus on finding the shortest distance route travel destination planning by applying TRP. It is assume that location choices for the destination are already pre-defined and the distance of the city is calculated based on the coordinate between the cities. Termination criteria will also need to be defined by tourist, which in this case, maximum distance that the tourist would like to travel in a day/period since Genetic Algorithm is a heuristic algorithm that will require a parameter to terminate the algorithm once the criteria has been met.

1.4 Feasibilities of the Project

1.4.1 Technical feasibility

This project requires knowledge and skills in programming in Java in which the algorithm will be developed and building a mobile application using Android platform. The content can be extracted from any third-party website that is available on the Internet regarding tourism.

1.4.2 Economic Feasibility

Currently, there is no extra cost that needs to fund the project because the algorithm and mobile application is developed using a personal computer workstation and simple mobile application developer that is free of charge.

1.4.3 Operational Feasibility

This project will be able to meet the objective since the authors have narrowed the scope. Once it is completed, it will be able to assist foreign and local tourist to plan their tour when they visited Malaysia.

1.4.4 Schedule Feasibility

The limitation of the project will be its tight schedule since the duration is only limited to 28 weeks (FYP1 and FYP2 together). Below are the timeframes, which can be considered as timely feasible:

FYP1

- Data gathering
- Development of the algorithm
- Interface development

FYP2

- Development of the prototype
- User and functionality testing
- Final documentation

CHAPTER 2

LITERATURE REVIEW

2.1 What is Travelling Salesman Problem (TSP)?

TSP is a very famous mathematical problem has been firstly introduced in the 18th Century. Sir William Rowan Hamilton, an Irish Mathematician and Thomas Penyngton, a British mathematician introduced it in the early stage and further developed by Hassler, Whitney & Merrill in Princeton in a more normal form.(Matai, Mittal&Singh,2011). Yong (2009) also mentioned that TSP is one of the benchmark for optimization problem. Gupta and Khurana (2012) believed that TSP has been proven that it is an NP-complete problem that has no definite optimal solution or efficient way to solve if there is a large size problem. It is as defined finding the optimum path to visit all cities starting from the initial point and returned back at the same point with the consideration to minimize the travel cost at shortest distance.

According to Matai, Mittal and Singh, (2011):

“ The complexity of the TSP is defined as given n is the number of cities to be visited, the total number of possible routes covering all cities can be given as a set of feasible solutions of the TSP and is given as $(n-1)! / 2$. ”

It is clearly explained, given a set 6 cities to be visited, there will be 6! Sets of possible routes that equals to 720 possible routes that we can derive from this travel. However, in TSP, we need to focus on finding the best possible path that can minimized the distance that we need to travel between the cities without going through the same point twice. An exact TSP can also be classified into three types, which are symmetric travelling salesman problem (sTSP), asymmetric travelling salesman problem (aTSP), and multi travelling salesman problem (mTSP). TSP has

many applications in our real life that will be further explained in the next part of the literature review.

2.2 What is Tourist Routing Problem (TRP)?

According to Yong (2009), manual path planning can be very exhaustive and time consuming. He believed that this is due to the fact that we need to use brute-force to find all the best route possibilities. It will take a lot of time to simply find the best route to travel across different places at the same time. The dilemma has been described as Tourist Routing Problem (TRP). TRP is a variance of Travelling Salesman Problem (Hashimoto et al., 2006). It is similar to the Vehicle Routing Problem (VRP), which has been introduced by Danziq and Ramser in 1960.









VRP can be defined by delivering a set of customers with known demands or time – window (Chang &Chen, 2007). The cost constraints of the problem are limited to soft time window and time to travel (Hashimoto et. al, 2006). They also stated that VRP use local search to determine the possible route. Both constraints in VRP and TRP have the same objectives, which is to visit the same place once besides finding the minimum distance for them to travel. Since TRP and VRP are both derivative of TSP, it is still considered as NP- complete problem (Non-deterministic polynomial-time complete) because when we put the constraints normally, it can be solve by using a simple dynamic programming and vice versa.

2.3 Malaysia as a Tourist Destination Place

Malaysia, ranked as 9th most visited place in the world are apparent to the tourist due to their rich in culture heritage, multi-cultural community, various annual festivities, wonderful cuisines and traditional crafts (Ledesma et al, 2012). Boasting beautiful sandy beaches and oldest tropical rainforest in the world as the main attraction for tourist to flood in every year. They also believed that the national parks are perfect for water rafting, cave exploration and wildlife watching that will provide challenge for tourist who loves adventures. According to Home Minister Ng Yen Yen (2013), Malaysian's tourism will boost up due to our rich culture and nature heritage which

can be clearly seen in famous cities such as in Penang and Malacca which has been certified as UNESCO'S World Historical Sites.

It was estimated that over 25.03 millions of visitors flood in Malaysia in 2012, which is an increase of 1.3% compared to 2011. (UNWTO, 2013). Malaysia is an all- year round humid weather country are suitable to visit anytime and the peak season is during the summer holiday season with visitors from Middle East, Singapore, and Australia. Table below shows the 10 highest tourist nationalities that visited Malaysia in 2012.

		Number of Tourists	Total of Tourist
		Arrivals	Arrivals
		(million)	(%)
Singapore		13.01	51.99
Indonesia		2.38	9.52
China (including Hong Kong & Macau)		1.56	6.23
Thailand		1.26	5.05
Brunei		1.25	5.03
India		0.69	2.76
Philippines		0.51	2.03
Australia		0.5	2.03
Japan		0.47	1.88
United Kingdom		0.4	1.61

*Table 2.1 10 highest tourist nationalities that visited Malaysia in 2012
(Source: Tourism Malaysia)*

Besides, Malaysia is estimated to gain a net profit of RM 60.3 billion due to tourism each year and will be the third largest foreign earning section for the country's economy. Tourism also flourishes the country's related tourism occupational opportunity for the locals. The country not only boost their culture and natural heritage as the main attraction, but local foods, shopping heavens, homestay, theme parks , fine arts and creativity and other parts of truly Asian tourism. Malaysia aimed

to be the best place to visit in the world that with the help of regional and international collaboration that can lead to development of tourism in Malaysia.

2.4 Famous local attractions in Malaysia

Below is the list of famous local attractions in Malaysia that has been frequently by foreign and local tourist each year.

Islands and Beaches	<ul style="list-style-type: none"> • Langkawi, Kedah • Pangkor, Perak • Penang Island • Redang Island • Tenggol Island, Terengganu • Tunku Abdul Rahman National Park • Perhentian Islands • Kapas Island • Lang Tengah Island • Rantau Abang Beach • Mabul • Tioman Island • Sipadan
National Parks	<ul style="list-style-type: none"> • Bako National Park, Sarawak • Batang Ai National Park, Sarawak • Gunung Mulu National Park, Sarawak • Gunung Gading National Park, Sarawak • Lambir Hills National Park, Sarawak • Niah Caves National Park, Sarawak • Loagan Bunut National Park, Sarawak • Kinabalu National Park, Sabah • Taman Negara National Park – spanning from Kelantan, Pahang and Terengganu (World Oldest Rainforest) • Endau Rompin National Park, Johor
Shopping Malls and Centers	<ul style="list-style-type: none"> • Berjaya Times Square KL, KL • Bukit Bintangwalk, KL • Suria KLCC, KL • Pavillion, KL • Midvalley Megamall, KL • Queensbay Mall, Penang

	<ul style="list-style-type: none"> • Johor Premium Outlets, Johor
Theme Parks	<ul style="list-style-type: none"> • A' Famosa Resort, Malacca • Sunway Lagoon, Selangor • LEGOLAND, Johor Bahru • Hello Kitty Land, Johor Bahru • Lost World of Tambun, Perak • Genting Highlands, Pahang • Bukit Merah Laketown Resort, Perak
Zoo and Animal Protection Park	<ul style="list-style-type: none"> • National Zoo of Malaysia (Zoo Negara), KL • Kuala Gandah Elephant Sanctuary • Kuala Lumpur Bird Park, KL • Kuala Lumpur Butterfly Park, KL • Sepilok Orang Utan Sanctuary
High Hills	<ul style="list-style-type: none"> • Masjid Negara • Merdeka Square • Cameron Highlands • Bukit Larut • Bukit Fraser • Genting Highlands
Historical Sites or Monuments	<ul style="list-style-type: none"> • Masjid Negara • Merdeka Square • Tugu Negara • Stadium Negara • Malacca • Georgetown

Table 2.2: Tourist attractions in Malaysia according to categories

2.5 Travel Destination Decision Making

According to Hwang et al. (2008), complicated, multifaceted decision process is required to make a decision in travelling. It is defined that in tourist decides which location to visit according to their place of interest and also any tourism services available at the location. Considering the place's attractions, timing, transportation, activities, accommodation and other facilities are also important. They conclude that destination choice is selecting a focal place to visit among numerous alternative places in order to satisfy the tour goals.

Many studies assume that tourist decides the best place to visit by assessing the cost, maximizing their utility and benefits that they will get from selecting specific location as their tour destination. According Hewitt (2012), tourist need to be able to carefully arranging their stay due to their first time experience visiting a foreign country. They need to carefully “map out” their travel itinerary either through they decision whether to visit the nearest location possible, undertook a cheaper cost travel tour or visiting places that they might find interesting activities conducted there.

Hewitt also suggested that tourist should also plan on how many cash that they would like to carry throughout their tour in case of emergency and it sometimes can be too dangerous to carry a lot of cash in hand. Hwang (2012) stated that socio demographic factors could also influence travel decision-making process such as their age, marital status, education and also level of income. In conclusion, selecting and planning travel destinations while travelling is not solely depend on utility maximization but also situational factors.

2.6 Main application of TRP & TSP

In real life, there were abundance of problems that can be modeled using TSP. Yong (2009) in his paper mentioned that there were varieties of TSP application such as planning, logistics, and manufacturing of micro-chip. He also stated that there are abundant sub-derivative of TSP such as DNA sequencing and also routing problem. Since TRP is a variance of TSP, it can be seen from the main application of routing planning for school bus, army mission planning, and museum visitor problem. In the instance school bus problem, found that some constraints need to be taken into consideration to solve the problem. In order to find the best possible route schedule, the number of route need to be small, have minimum travel distance, time consumed to travel using the route did not exceed maximum time constraints.

As for army mission planning, reported that it involves planning the best path for each army to reach their goal or mission in minimum possible time. The problem is a bit more complex since it will need to take into consideration of n army, m goals,

and base city or initial point where all of the army must return. It is considered that each salesman has a fixed cost, f by solving the algorithm through relaxing the SECs and performing a check as to whether any of the SECs are violated, after an integer solution is obtained.

Looking back at the museum visitor routing problem, the problem is applied when all of the visitors share the same interest (Yu, Lin, & Chou, 2010). In this case, some of the visitors will visit the same route and some will be divided into different route. Yu, Lin & Chou suggested that the routes are need to be plan in order to save the time to conduct to visit and divided into groups of visitors that share the same interest. The time to visit each place can vary due to certain exhibition might interest the visitor more than other exhibitions so time –window of different route and at certain place might vary. Therefore, a proper planning for the visitor’s route could help reduce the congestion at certain place or prolonged tour time.

2.7 Comparative study on the possible solution for TRP & TSP

Algorithm	Characteristics	Comparison
Heuristic Algorithm	- Simplification algorithm that reduce or limits the search for algorithm those are difficult to understand.	-Does not guarantee optimal solution
Memetic Algorithms	-Combination of several techniques (local search and crossover from genetic algorithm. -Use local heuristic to determine how to define iteration.	-Very difficult to program in a short time. -Very detailed and have many inputs to be processed
Ant colony algorithm	-Based on studies of ant colony. - Each colony can form new colony	- Can find optimal solutions up to 100 cities only
Genetic Algorithm	-Based on the theory of evolution - Break into smaller parts to find the fitness value. - Heuristically find optimal path until it reach the stopping criteria	- Very thorough algorithm that handles all possibilities to get optimal path until it reach the stopping criteria.

Table 2.3 Possible solutions to TRP&TSP

2.8 What is Genetic Algorithm?

Genetic algorithm is one of the best heuristic algorithms that is widely utilized to solve TSP derivative that includes TRP which also due to wide application of TSP in real life. Originating from the Greek word “genesis” in which literally translates to “to become” or “to grow”, it follows the principles of Genetics and Evolution (Gupta & Khurana, 2012). It is basically a step-by-step search technique to find approximate solutions to optimization problem. Basically, an optimization problem looks really simple.

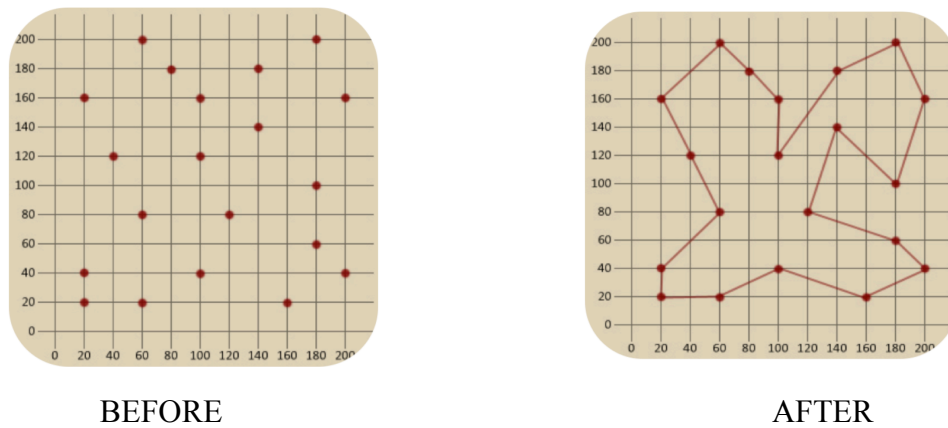


Figure 2.1: The cluster of cities before and after going through Genetic Algorithm.

One knows the form of all possible solutions corresponding to particular problem. Depending on the number of cities that will be visited, all the possibility can be known but the objective was to find the optimal path. Gupta and Khurana stress that genetic algorithm will go through each and one of possible population (routes). Each solution is represented by a chromosome, which in this project, the solution will refers to possible route. It uses the concept of mutation, crossover and recombination to solve the problem.

Due to fact that TRP is an NP-complete problem, Genetic Algorithm is considered as an exact method that can find the exact optimal solution (Kiraly and Abonyi, 2010). However, problem will still occur if the problem is too large in size. There are few elements that need to be considered in Genetic Algorithm in order to find the optimal path to travel throughout the entire place without going through the same place twice.

Elements	Explanation
Genetic coding	<ul style="list-style-type: none"> - How to represent the cities - Can choose either to represent it in path or array.
Fitness function	<ul style="list-style-type: none"> - Value that will be assigned to each node/route. - The higher the value, f_i, the greater chance it will be chosen as optimal path.
Selection	<ul style="list-style-type: none"> - Choose two cities to be crossover to form new route - Can either choose “roulette wheel” selection or random selection.
Crossover	<ul style="list-style-type: none"> - To ensure that none of the cities is repeated or missed out - Can use order crossover, cycle crossover, or partially matched crossover
Mutation	<ul style="list-style-type: none"> - To discard the less f_i value route and store it in their memory - To ensure that no important features are lost such as random selection of the cities.

Table 2.4 Important elements of Genetic Algorithms

2.9 What is Android?

Android is an operating system for mobile-based platform. It is Linux-based and designed to be used on touchscreen mobile devices such as smartphone and tablets. Introduced in 2005, when Google Inc bought it from Android Inc. In 2012, it is now currently the biggest operating system in terms of user demographic in Malaysia that overtook Symbian (Nokia) and iOS (Apple) and Blackberry OS by 43.16%.

2.10 Why Android?

In order to run this application, a platform is required. This project application will require a platform As Android is the fastest growing mobile application platform available in the market, it ensures sustainability and constant improvements of the application. Huge user demographic can ensure that application that is made available in the Google Play can be downloaded for free or per pay.

It features as an open-source operating system, which is free, and the development of application would be much easier with the support from online community. Creating an Android mobile application can also be made easy for non- experts through platform such as App Inventor for Android is a visual programming environment created by Google for application development.

2.11 Comparative Study on Existing Mobile Application

Application Name	Focus Category	Strength of Product	Weakness
Trip Advisor: Kuala Lumpur City Guide	Specific guide application	-Specific city guide -Works offline -GPS based application	- No best route suggestion to visit few places in one trip.
Trip It	Overall trip planning	- Timely –planner that organize all items into travel itinerary.	-User must manually input all information.
Poynt	Suggestion based application	-Directly connected with local business - Offer variety place of interest.	-Information and feedbacks are fixed to tailor to business needs.
Wikitude	Place of interest information	Use AR in disseminating information about the place.	-No nearest place suggestion, which is similar to user interest.
Route Planner	Routing planning	- User enter initial point and end point of the trip - User can mark their favorite route	-Functions almost similar to Google maps
Best Route Free	Routing planning	-Give a whole map view -Abundant information and functions	-The interface is very hard for novice user to use.
Google Maps	Comprehensive travel maps	-Have satellite and 3D view of the maps - Abundant information -Simple user interface	- Some specific features are limited to certain regions or continent only.

Table 2.5: Existing mobile application that is related to the project

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Project Method & Activities

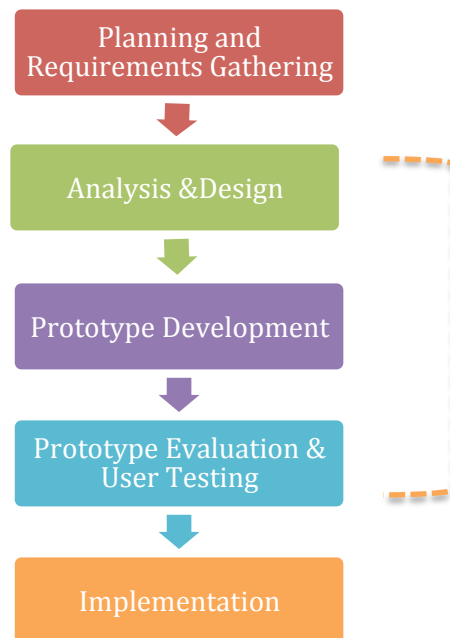


Figure 3.1: Research Methodology Diagram

For this research, the author has chosen rapid-prototyping methodology as an approach. Objective of choosing this methodology is to ensure that user requirements will be met, development is aligned according to the deadlines and each testing will be conducted once the prototype is ready. Requirements gathering and planning in the early stage will take part and the best algorithms to solve the problem will be taken into consideration to develop the prototype.

If project encounter any problem through any stage, the manager can easily go back one step back to ensure that it will meet its design capabilities. Upon testing, feedbacks and criticism will be taken into consideration to ensure that the project functions effectively and improvements immediately. The finalized prototype then will be converted as a system for user usage.

3.1.1 Planning and Requirements Gathering

Planning phase of the research involved doing a pilot study reading based on books, journals, articles regarding tourism, application of TSP, TRP and to identify the problem, scope and objectives of the study. A general idea on scope of the study such as tourism in Malaysia, possible solution to TRP and Android Mobile Application are also necessary. Narrowing the scope of study will ensure that the system produced will be able to meet the user demands and needs.

3.1.2 Analysis & Design

Analysis of the requirements is conducted based on the identification of the problem, scope and objective of the study. Objectives are further refined to identify necessary functions or value that need to be added to the system. The area of the study are then further researched and studied to gather necessary information such as decision making problem for travellers, how to model TRP, and how to embedded GA into the mobile application for testing and with the objective to produce the best result. Correct analysis is important to ensure that the prototype will function properly which will then produced the literature review.

3.1.3 Prototype Development

The prototype will be build and develop at this phase according to the results done on previous research or initial stage. The Genetic Algorithm will be first developed using Java language and tested using Netbeans 7.3.1 to test its effectiveness using a computer simulation. Then, it will be embedded in the mobile application to test its functionality once the prototype is completed,

any major problems will be change and improved after evaluation and experiments until it satisfies the user needs.

3.1.4 Prototype Evaluation & Testing

The developed prototype will be tested by a group of user that fits the target user demographic, which is tourist. The execution time and speedup of the application will be tested to see whether with an increase in number of cities, does the performance of the application increases or decreases. Tourist response and interaction towards the prototype will be closely monitored. Any feedbacks and criticism will be taken into accounts and improvements will be made immediately. Re-evaluation is required to ensure that the application will satisfy the tourist need.

3.1.5 Implementation

The finalized prototype that satisfies the user will be then called as system and implemented to real markets by uploading the application into Google Play.

3.2 Project Activities

3.2.1 Overview of Project Activities

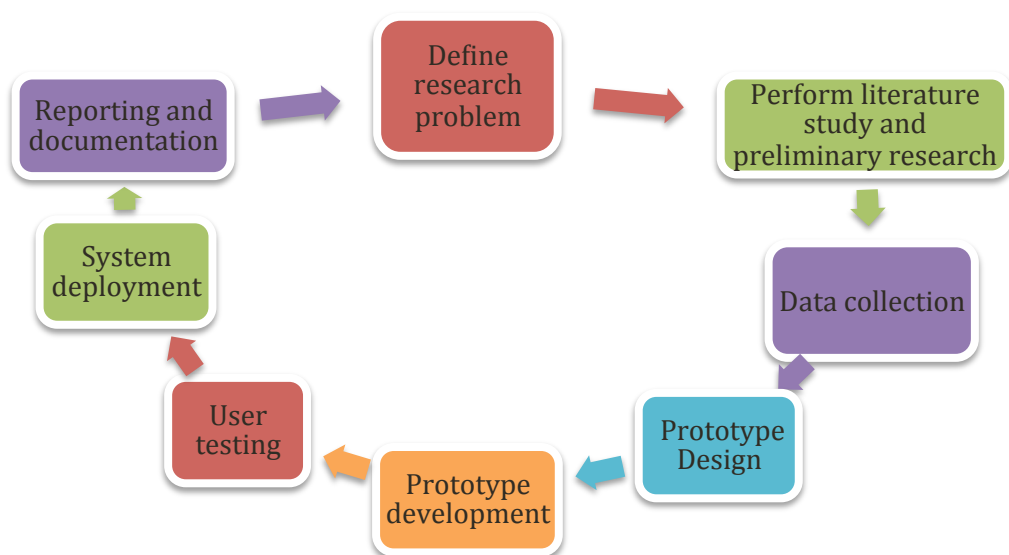


Figure 3.2: Overview of overall project activities

3.2.2 Requirement Gathering

Comparative study and analysis are conducted to gather requirements and information for project developments.

Analysis	Objective
Comparative study on travel destination planning application that uses TRP. The analysis focuses on criteria as follow: <ul style="list-style-type: none"> i. Available tools that focus on finding the shortest path to travel through multiple location ii. Tools that recommends user on best path to be taken based on their interest 	<ul style="list-style-type: none"> i. To review on the available application on the market. ii. To identify each strength and weakness of the available applications to apply in the system. iii. To collect data for system requirements. iv. To support problem statement
Analysis on TRP approach and its possible solutions.	<ul style="list-style-type: none"> i. To gather information on TRP ii. To review on the real application and their effectiveness. iii. To study and identify the best possible algorithm as a solution.

Table 3.1 Analysis and comparative study conducted before project development

3.3 System Architecture

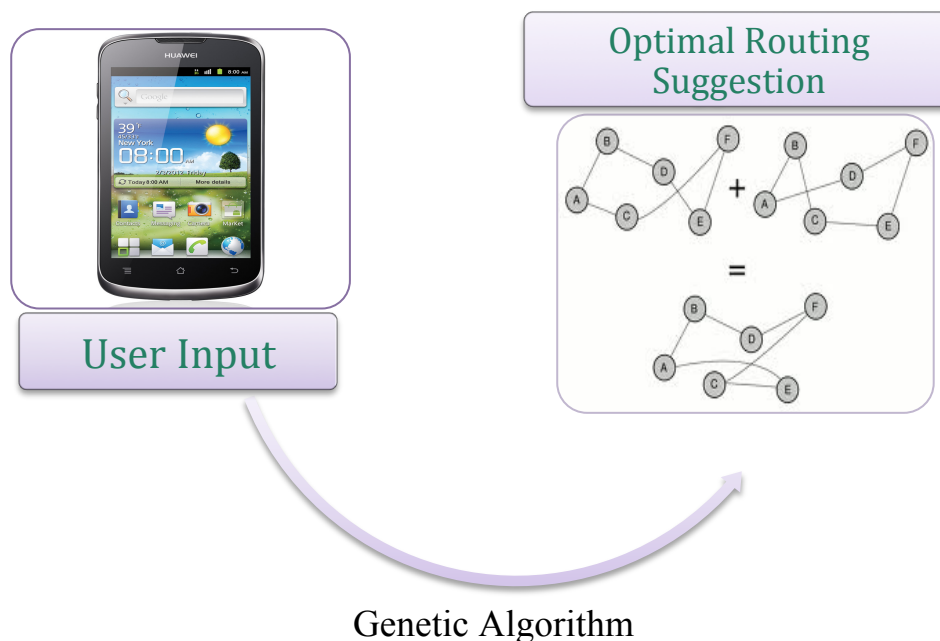


Figure 3.3 System Architecture for the Mobile Application

This is the system architecture for the prototype. Figure shown that user will key-in the necessary information into the application, for example: place of interest, maximum distance they can travel on that day, and the cost for them to travel. The application will then use the Genetic Algorithm to find the optimal path for the user to travel by going thorough the all the place once at a minimized distance. It will then project the optimal routing suggestion based on the calculation.

3.4 Requirement Analysis and Specification

Based on the above system architecture, there are few requirements that need to be listed out in order to build the prototype and thus complete the system. Since the back-end of the project will be using Genetic Algorithm, there are few requirements that need to be set to ensure that the calculation will produce almost accurate results.

3.4.1 Genetic Algorithm

In order to solve TRP using Genetic Algorithm, we have to make sure that all of these requirements are satisfied:

Elements	Explanation
Genetic coding	<ul style="list-style-type: none"> - How to represent the cities - Can choose either to represent it in path or route.
Fitness function	<ul style="list-style-type: none"> - Value that will be assigned to each node/route. - The higher the value, f_i, the greater chance it will be chosen as optimal path.
Selection	<ul style="list-style-type: none"> - Choose two cities to be crossover to form new route - Can either choose “roulette wheel” selection or random selection.
Crossover	<ul style="list-style-type: none"> - To ensure that none of the cities is repeated or missed out - Can use order crossover, cycle crossover, or partially matched crossover
Mutation	<ul style="list-style-type: none"> - To discard the less f_i value route and store it in their memory - To ensure that no important features are lost such as random selection of the cities.

Table 3.2 Important elements in Genetic Algorithm

Besides this, there are also constant parameters that need to be declared for Genetic Algorithm to function:

Parameters	Functions
Population Size	- To decide how many generations (possible routes) to be generated from the crossover and mutation.
Crossover probability	Probability of crossover between 2 generated route.
Mutation probability	Probability of doing mutation between 2 specific cities (ex: between city A and city B., A-B, B-A)
Termination criteria	Need to be declared in order for the to terminate the search (can be distance or time)

Table 3.3 Constant parameters that need to be declared in Genetic Algorithm

3.5 Tools and Equipment

3.5.1 Hardware

- a) 4GB 1333 MHz DDR3 RAM
- b) 1.6 GHz Intel Core i5
- c) Samsung Galaxy S II GT-I9100 powered with Android 4.0.4 Ice Cream Sandwich Operating System.
- d) MacBook Air, 11 inch, Mid 2011 Edition

3.5.2 Software

Software to be used:

- a) Android Developer Tools (ADT)
ADT consist of:
 - i) Eclipse and ADT Plug-in.
 - ii) Android Software Development Kit (SDK) Tools
 - iii) Android Simulator
 - iv) Android Platform Tools
- b) NetBeans 7.3.1

- c) Macintosh Operating System, Mac OS X Mountain Lion 10.8.5

3.6 Key milestones & Gantt chart

A Gantt Chart (Appendix 1) is prepared to guide the development of the project. Deliverables and key milestones (Appendix 2) are attached together in the Gantt chart. Note that the timeline merge the timeline for Final Year Project I together with Final Year Project II to make sure that the project will be able to be delivered on time and tested with target user before released to the public.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Genetic Algorithm Calculation for Shortest Distance Route

To produce the best shortest distance route that allows the visitor to travel from the starting point across different cities without going through the same place twice, and return back to the same point with minimal distance, all of the requirements and parameters as stated in the requirement analysis and specifications need to be met. Since Genetic Algorithm is a step-by-step heuristic search, therefore there will be steps in order to produce the best result. Figure below describe how the algorithm work:

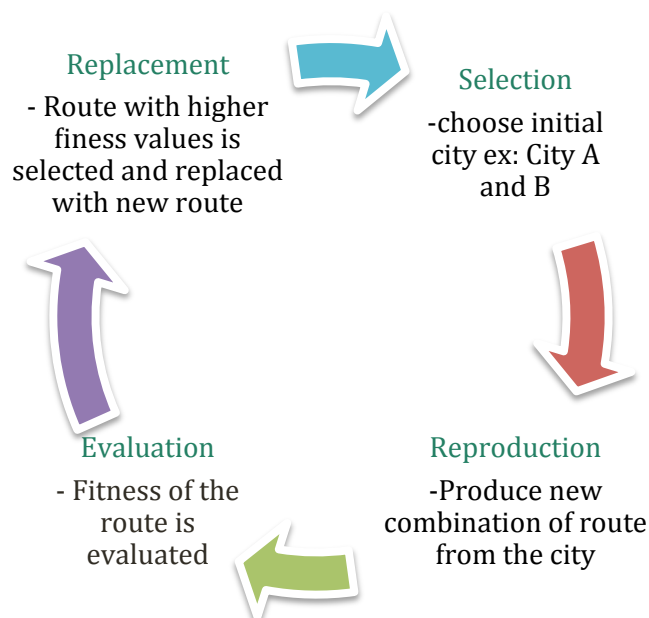


Figure 4.1 Genetic Algorithm cycle

Based on the algorithm, the first stage selection, an initial city will be choose based on this two technique, either random selection or “roulette wheel” selection. In the

application, the technique that is used is random selection in which it can select city by determining their fitness value, f_i . In the real application, the fitness value will be the distance of that particular city.

In the reproduction stage, an allele of two routes will be choose, crossover and mutate to form new route. The crossover type that will be used is cycle crossover in which it is a recombination of both parent route, R_x and R_y to form offspring route, R_z . Example:

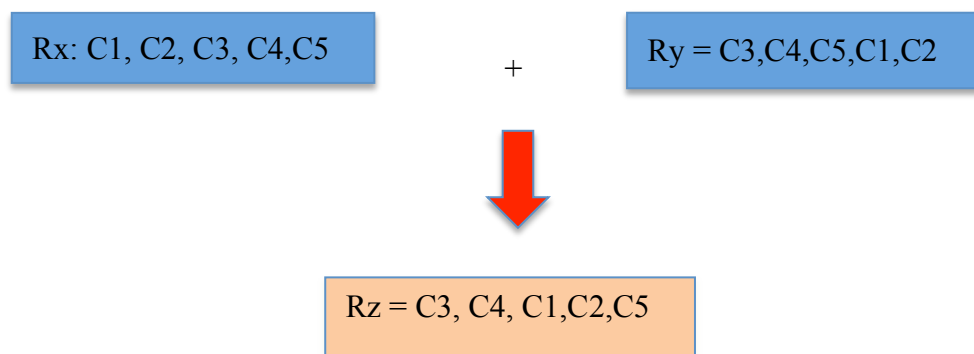


Figure 4.2 Offspring route generated from crossover and mutation

Each offspring route that has been produced in previous stage will then be given fitness value, f_i that represents its total distance that need to be travelled. In evaluation stage, offspring route that is more fit compared to others are choose. At the replacement stage, the fittest route will be choose as a parent and the cycle will run heuristically until it reach its terminating criteria which in this case, the minimized distance. The previous offspring route will then be discarded.

Since Genetic Algorithm is a heuristic solution to any derivation of TSP such as TRP, therefore it can suggest optimal path with rather moderate size of problem. However, with an increasing number of cities, the performance of the algorithm will increase and speedup of the execution will also increase. The author will conduct simulation and experiment to test the algorithm later in this report.

4.2 Eliminating the possibility of redundancy in path planning in solving TRP

The author have adjusted and remodeled the existing GA to fit into the scope of study. To proof that GA is the best heuristic solution to derivative of TSP that includes TRP, the author conducted simulation based on different number of cities. Below is a map that consists of 20 cities.

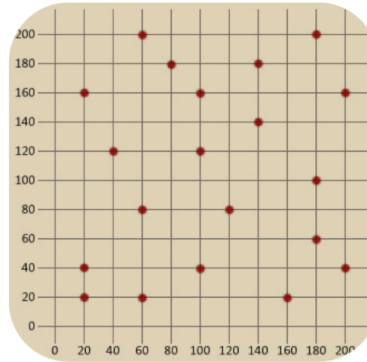


Figure 4.3 Map that consists of 20 cities.

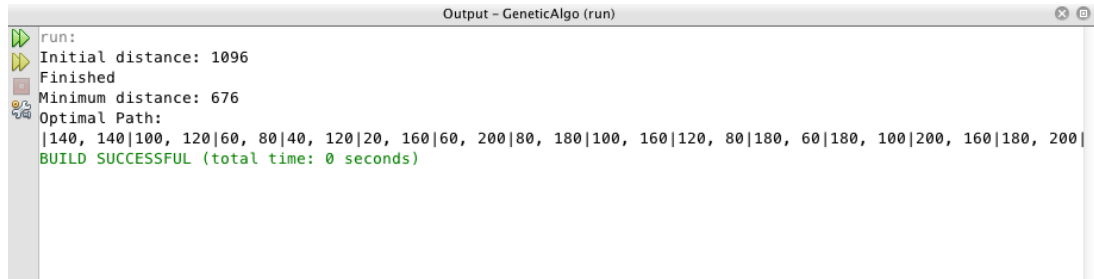
This simulation will also test that whether it is possible to eliminate the possibility to eliminate the redundancy in path planning which is travelling through same city twice. An increase of 5 cities, 10 cities, 15 cities and 20 cities was tested as a parameter. Figures below shows the results obtained with different variable of cities:

```
Output - GeneticAlgo (run)
run:
Initial distance: 343
Finished
Minimum distance: 343
Optimal Path:
|20, 160|80, 180|140, 180|180, 200|60, 200|
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4.4 Shortest distance route for 5 cities with GA

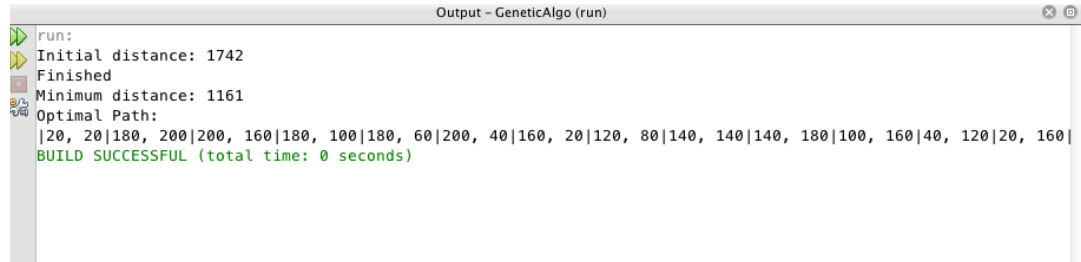
```
Output - GeneticAlgo (run)
run:
Initial distance: 545
Finished
Minimum distance: 455
Optimal Path:
|80, 180|60, 200|20, 160|40, 120|100, 160|140, 140|200, 160|180, 200|140, 180|
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4.5 Shortest distance route for 10 cities with GA.



```
run:
Initial distance: 1096
Finished
Minimum distance: 676
Optimal Path:
|140, 140|100, 120|60, 80|40, 120|20, 160|60, 200|80, 180|100, 160|120, 80|180, 60|180, 100|200, 160|180, 200|
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4.6 Shortest distance route for 15 cities with GA.



```
run:
Initial distance: 1742
Finished
Minimum distance: 1161
Optimal Path:
|20, 20|180, 200|200, 160|180, 100|180, 60|200, 40|160, 20|120, 80|140, 140|140, 180|100, 160|40, 120|20, 160|
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 4.7 Shortest distance route for 20 cities with GA.

Based on this simulation, we have found that there is a difference is the initial distance, which is the total distance of all cities before GA and minimum distance, which is the distance of the shortest distance as suggested by the GA. The route suggested by the application is also based on the shortest distance possible to travel. It has been proved that the route suggested does not show any redundancy in path planning since none of the city is repeated more than once in route.

4.3 Performance Evaluation of GA in solving TRP

Besides simulation, the author has also conducted an experiment, which is to evaluate the performance of GA in solving TRP. This experiment has been done by measuring the execution time of the application parallel to the number of cities. Line graph below shows the result obtained by the experiment:

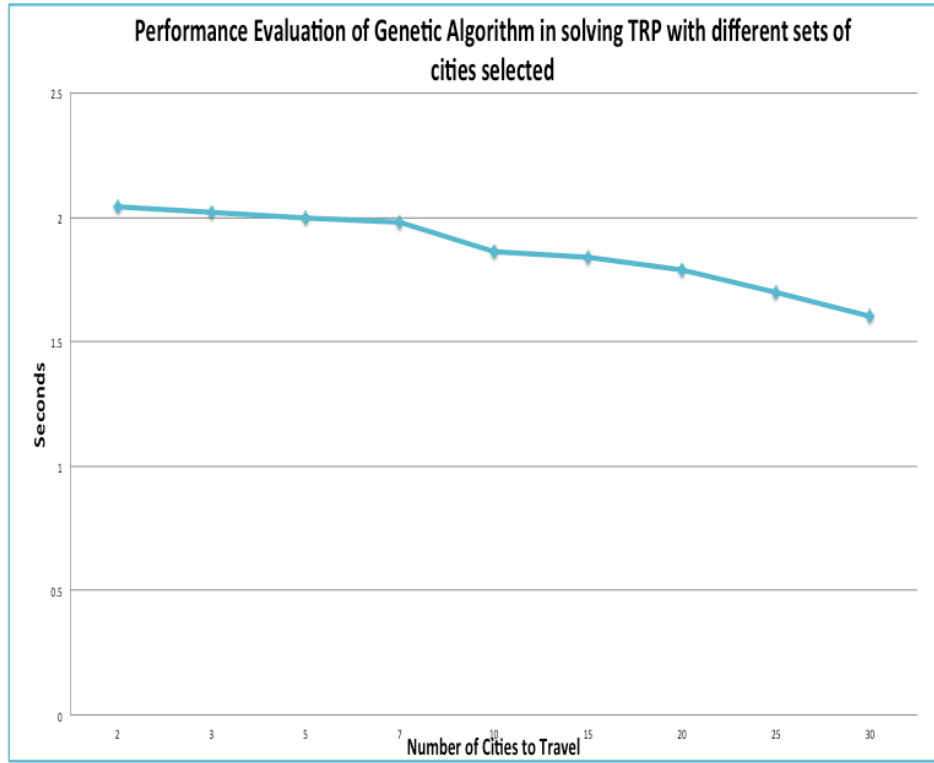


Figure 4.8 Performance Evaluation of GA in solving TRP with different sets of cities selected.

Based on the results, we can conclude that the execution time that the application requires in order to suggest the shortest distance route decreases even though number of cities selected by the user increases. These phenomena can be explained due to the memory hierarchy of the device used. In this case, results of initial experiments are stored in the system memory. Once the same or precedence calculation is being called, this reduces the execution time significantly.

4.4 Speedup of the GA in solving TRP with increasing number of cities selected

Once the author evaluates the performance of the algorithm by conducting experiment above, we can also tabulate the speedup of the algorithm. The calculation of the speedup is based on:

$$\text{Speedup} = \frac{\text{Execution time taken for 2 cities}}{\text{Execution Time taken for N cities}}$$

Figure below shows the tabulation of speedup of the algorithm versus the number of cities selected by the tourist:

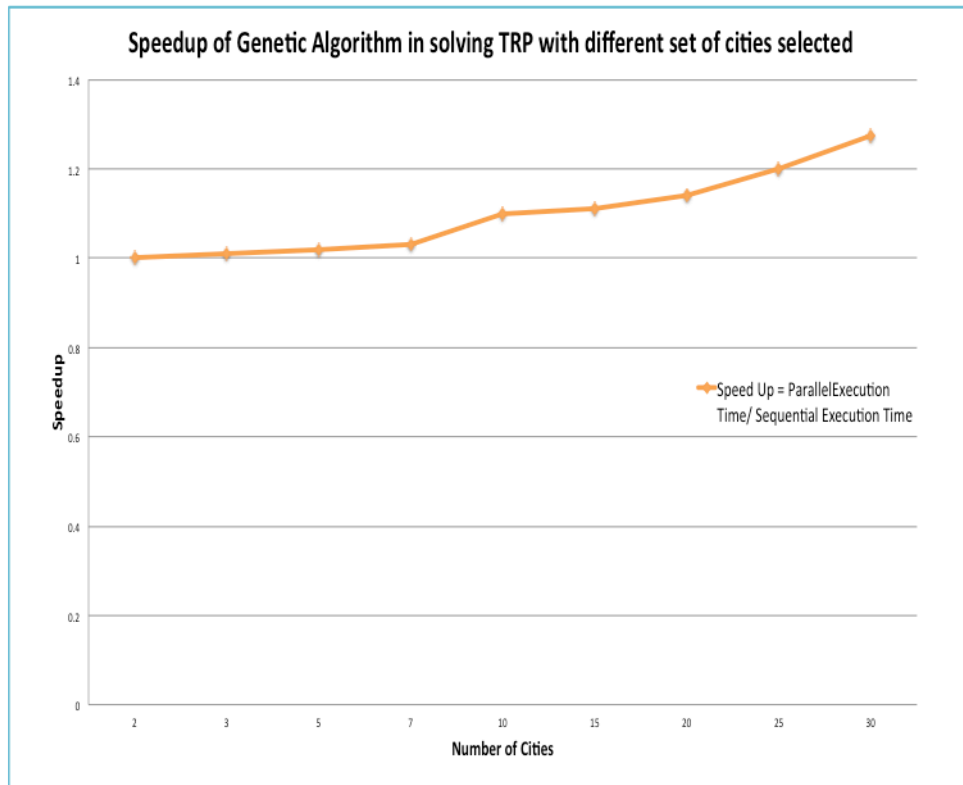


Figure 4.9 Speedup of GA in solving TRP with different sets of cities selected.

Based on the results, we can conclude that the speedup of the application increases even though number of cities selected by the user increases. Reduction in the execution time is also due to multi-processor architecture of the device being used for the simulation and experiments that processed information in parallel. Increased performance in execution time and less time is required to process a large number of cities can also be explained due to the memory hierarchy effect of the device used.

Previous route that had already been generated by the algorithm will be stored in the RAM. This is due to the temporal locality of reference, which means that there is chance that same information will be referred in the future. The temporal locality of reference in this case is due to the structure of GA, in which it will store previous route information in the RAM. With increasing number of cities selected, initial information can be simply called and thus reduce execution time.

Besides this, the microprocessors in the device also have trace cache in which it stored portions of instructions traces that have already been decoded previously. A trace cache stores instructions either after they have been decoded, or as they are retired. Therefore, it is concluded although the number of cities selected by the user increases, the performance and speedup of the algorithm in solving TRP increases.

4.5 System Design

The system is designed:

- a) To ensure tourist will be able to select their place of interest properly.
- b) To ensure that user can view previous tourist recommendation or rating of that particular place.

Below is the sample of interface design for the mobile application:



Figure 4.10 Welcome page of the application

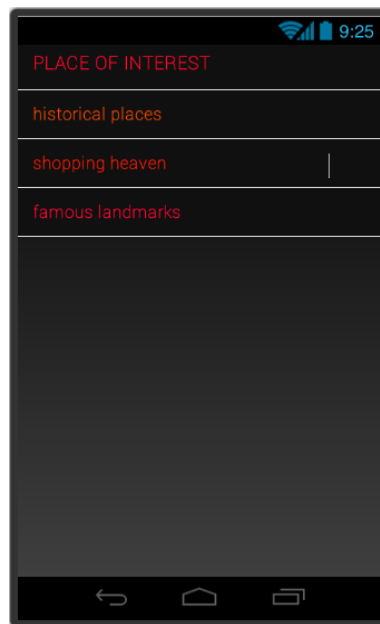


Figure 4.11 Place of interest selection according to the categories

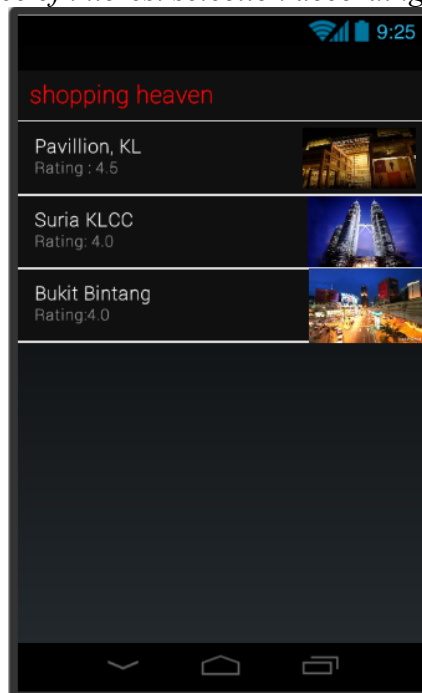


Figure 4.12 List of places included in the Shopping Heaven category

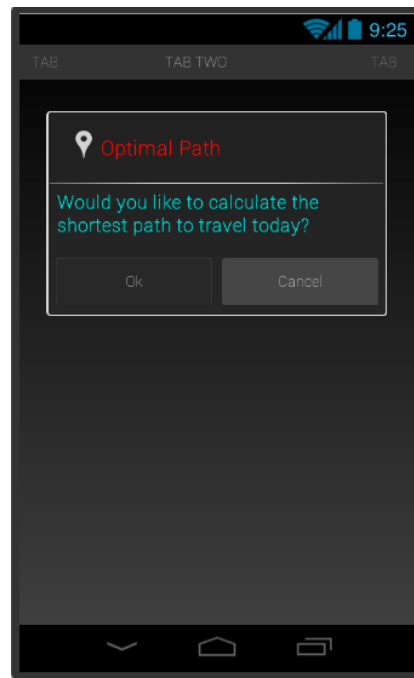


Figure 4.13 Shortest route calculation is projected to the user

CHAPTER 5

CONCLUSION

5.1 Conclusion

Solving TRP by applying GA as a solution is an active area of research. The objective of this research has been met by proofing the concept through simulations and experiments by utilizing the mobile application as a platform to test the algorithm. The effectiveness of GA in solving TRP has been proven due to an increase in performance and speedup of the algorithm even though the number of cities selected by the user increases. At the end of this project, the mobile application that applied GA in solving TRP will be able to assist tourist to plan their tour accordingly. Tourist will be able to decide which place to visit according to their preference and shortest distance route they would be able experience the true Malaysian experience visiting various interesting place.

5.2 Recommendation

For future enhancement, the author would like to recommend more parameters will be added by not suggesting the shortest distance route for tourist to travel, but taken into consideration of other factors such as time to travel, cost, and ratings for each tourist attraction. This will be able to provide tourist with an optimal path to travel that consist of shortest distance, time to travel, cost and ratings for each place. Adding more features for tourist such as tourism activities planner and travel checklist can also enhance functionality of the mobile application.

REFERENCES

Gupta,A. & Khurana, S.(2012) .Study of Traveling Salesman Problem using Genetic Algorithm., *International Journal of Management, IT and Engineering*. Retrieved from http://www.ijmra.us/project%20doc/IJMIE_MAY2012/IJMRA-MIE1139.pdf

Hashimoto,H., Ibaraki,T., Imahori,S, & Yaguira, M. (2006) . The vehicle routing problem with flexible time windows and travelling times. *Discrete Applied Mathematics: The Journal of Combinatorial Algorithms , Informatics and Computational Sciences*.41. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0166218X06001879>

Hewitt, E. (2012) . 10 things to do before you travel. *Independent Traveller*. Retrieved from <http://www.independenttraveler.com/travel-tips/troubleshooting/10-things-to-do-before-you-travel>

Hwang, Y.H., Gretzel, U. ,Xiang, Z. , Fesenmaier,D.R.(2008). Travel destination choice models. *Destination Recommendation System : Behaviourial Foundations and Applications*. pp.45-52. Oxfordshire,UK :CABI

Kiraly, A., Abonyi,J.(2010). A novel approach to solve multiple Travelling Salesman Problem using Genetic Algorithm. *Computational Intelligence in Engineering*.313, pp 141-151. Springer

Ledesma, C., Leffman,D.,Lewis, M.,Lim, R.(2012). Introduction to Malaysia,Singapore, Brunei. *The Rough Guides to Malaysia, Singapore & Brunei*.pp. 15-20. Melbourne : Rough Guides Limited.

n.d. (2013, August 16). Yen Yen : Culture and heritage tourism gives Malaysia a boost. *The Malay Mail*. Retrieved from <http://www.themalaymailonline.com/travel/article/yen-yen-culture-and-heritage-tourism-gives-malaysia-a-boost>

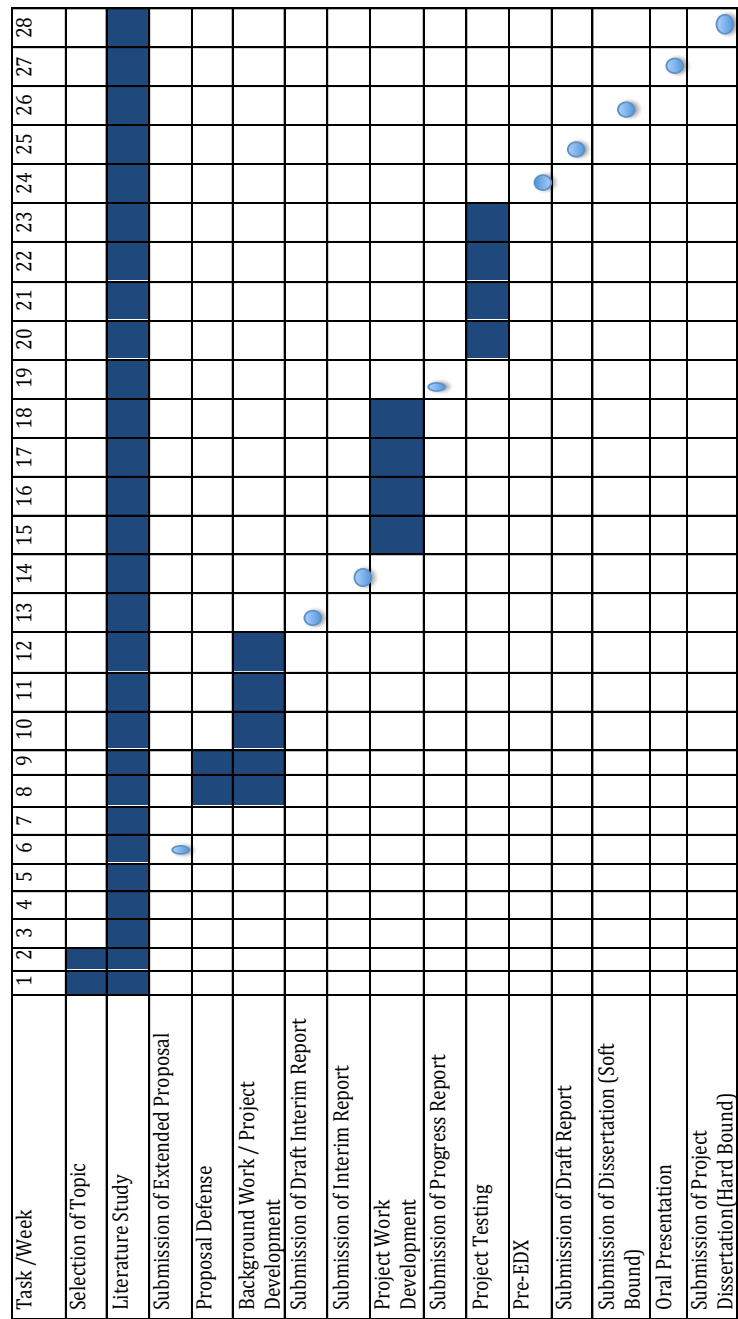
UNWTO 25th CAP-CSA Conference on Sustainable Tourism Development. (2013). *Malaysia Country Report*. Retrieved from http://dtx4w60xqpww.cloudfront.net/sites/all/files/pdf/malaysia_country_report_2012.pdf. Hyderabad:India

Yong,H.C. (2009) . Vehicle routing problem. *An Efficient Solving The Travelling Salesman Problem: Global Optimization Of Neural Networks By Using Hybrid Method*. Retrieved from http://cdn.intechopen.com/pdfs/12409/InTech-An_efficient_solving_the_travelling_salesman_problem_global_optimization_of_neural_networks_by_using_hybrid_method.pdf

Yu, V.F., Lin,W.S.,Chou,S.Y. (2010). The museum visitor routing problem.*Applied Mathematics and Computation*.216-3. 719-729. Retrieved from <http://www.sciencedirect.com/science/article/pii/S0096300310000792>

APPENDICES

Appendix 1: Gantt Chart



Appendix 2: Key Milestones

No	Week	FYP 1 Milestone	Date
1	1	FYP 1 Briefing	20 May 2013
2	1	Selection of Project Topic	20 May 2013
3	3	Submit Proposal to Research Cluster	4 June 2013
4	4	Topic and Supervisor Confirmation	13 June 2013
5	5	Ideation Makeweekend Rapid Prototyping	22 June 2013
6	6	Submission of Extended Proposal	25 June 2013
7	6	Briefing on Plagiarism	26 June 2013
8	12	Proposal Defense and Progress Evaluation	31 July 2013
9	13	Prototype Submitted	5 August 2013
10	14	Submission of Interim Report	20 August 2013

Appendix 3: Algorithm Codes in Java Language

TSP_GA.java (main program)

```
package geneticalgo;

/**
 *
 * @author sarah
 */
public class TSP_GA {
    /**
     * TSP_GA.java
     * Create a tour and evolve a solution
     */

    public static void main(String[] args) {

        // Create and add our cities
        City city = new City(60, 200); // cities represented like an array City(x-coordinate,y-coordinate)
        TourManager.addCity(city);
        City city2 = new City(180, 200);
        TourManager.addCity(city2);
        City city3 = new City(80, 180);
        TourManager.addCity(city3);
        City city4 = new City(140, 180);
        TourManager.addCity(city4);
        City city5 = new City(20, 160);
        TourManager.addCity(city5);
        City city6 = new City(100, 160);
        TourManager.addCity(city6);
        City city7 = new City(200, 160);
        TourManager.addCity(city7);
        City city8 = new City(140, 140);
        TourManager.addCity(city8);
        City city9 = new City(40, 120);
        TourManager.addCity(city9);
        City city10 = new City(100, 120);
        TourManager.addCity(city10);
        City city11 = new City(180, 100);
        TourManager.addCity(city11);
        City city12 = new City(60, 80);
        TourManager.addCity(city12);
        City city13 = new City(120, 80);
        TourManager.addCity(city13);
        City city14 = new City(180, 60);
        TourManager.addCity(city14);
        City city15 = new City(20, 40);
        TourManager.addCity(city15);
        City city16 = new City(100, 40);
        TourManager.addCity(city16);
        City city17 = new City(200, 40);
        TourManager.addCity(city17);
        City city18 = new City(20, 20);
        TourManager.addCity(city18);
        City city19 = new City(60, 20);
        TourManager.addCity(city19);
        City city20 = new City(160, 20);
        TourManager.addCity(city20);

        // Initialize population
        Population pop = new Population(50, true);
        System.out.println("Initial distance: " + pop.getFittest().getDistance());

        // Limited population for 50 generations only
        pop = GA.evolvePopulation(pop);
        for (int i = 0; i < 50; i++) {
```



```
        pop = GA.evolvePopulation(pop);
    }

    // Print final results
    System.out.println("Finished");
    System.out.println("Minimum distance: " + pop.getFittest().getDistance());
    System.out.println("Optimal Path:");
    System.out.println(pop.getFittest());
}
}
```

City.java

```
package geneticalgo;

/**
 * class City marks city
 * @author sarah
 */

public class City {
    int x;
    int y;

    // Constructs a randomly placed city
    public City(){
        this.x = (int)(Math.random()*200);
        this.y = (int)(Math.random()*200);
    }

    // Constructs a city at chosen x, y location
    public City(int x, int y){
        this.x = x;
        this.y = y;
    }

    // Gets city's x coordinate
    public int getX(){
        return this.x;
    }

    // Gets city's y coordinate
    public int getY(){
        return this.y;
    }

    // Gets the distance to given city
    public double distanceTo(City city){
        int xDistance = Math.abs(getX() - city.getX());
        int yDistance = Math.abs(getY() - city.getY());
        double distance = Math.sqrt( (xDistance*xDistance) + (yDistance*yDistance) );

        return distance;
    }

    @Override
    public String toString(){
        return getX()+" "+getY();
    }
}
```

GA.java (algorithm)

```
package geneticalgo;

/**
 *
 * @author sarah
 */
public class GA {
    /**
     * GA.java
     * Manages algorithms for evolving population
     */

    /** GA parameters */
    private static final double mutationRate = 0.015;
    private static final int tournamentSize = 5;
    private static final boolean elitism = true;

    // Evolves a population over one generation
    public static Population evolvePopulation(Population pop) {
        Population newPopulation = new Population(pop.populationSize(), false);

        // Keep our best route if elitism is enabled
        int elitismOffset = 0;
        if (elitism) {
            newPopulation.saveTour(0, pop.getFittest());
            elitismOffset = 1;
        }

        // Crossover population
        // Loop over the new population's size and create individuals from
        //current population
        for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
            // Select parents (previous route)
            Tour parent1 = tournamentSelection(pop);
            Tour parent2 = tournamentSelection(pop);
            // Crossover parents
            Tour child = crossover(parent1, parent2);
            // Add offspring route to new population
            newPopulation.saveTour(i, child);
        }

        // Mutate the new population a bit to add some new genetic material
        for (int i = elitismOffset; i < newPopulation.populationSize(); i++) {
            mutate(newPopulation.getTour(i));
        }

        return newPopulation;
    }

    // Applies crossover to a set of parents and creates offspring
    public static Tour crossover(Tour parent1, Tour parent2) {
        // Create new child route
        Tour child = new Tour();

        // Get start and end sub tour positions for parent1's tour
        int startPos = (int) (Math.random() * parent1.tourSize());
        int endPos = (int) (Math.random() * parent1.tourSize());

        // Loop and add the sub tour from parent1 to our child
        for (int i = 0; i < child.tourSize(); i++) {
            // If our start position is less than the end position
            if (startPos < endPos && i > startPos && i < endPos) {
                child.setCity(i, parent1.getCity(i));
            } // If our start position is larger
            else if (startPos > endPos) {
                if (!(i < startPos && i > endPos)) {
                    child.setCity(i, parent1.getCity(i));
                }
            }
        }

        // Loop through parent2's city route
```

```

    for (int i = 0; i < parent2.tourSize(); i++) {
        // If offspring doesn't have the city add it
        if (!child.containsCity(parent2.getCity(i))) {
            // Loop to find a spare position in the offspring's route
            for (int ii = 0; ii < child.tourSize(); ii++) {
                // Found empty slots in the route, add city
                if (child.getCity(ii) == null) {
                    child.setCity(ii, parent2.getCity(i));
                    break;
                }
            }
        }
    }
    return child;
}

// Mutate a tour using swap mutation
private static void mutate(Tour tour) {
    // Loop through tour cities
    for(int tourPos1=0; tourPos1 < tour.tourSize(); tourPos1++){
        // Apply mutation rate
        if(Math.random() < mutationRate){
            // Get a second random position in the tour
            int tourPos2 = (int) (tour.tourSize() * Math.random());

            // Get the cities at target position in tour
            City city1 = tour.getCity(tourPos1);
            City city2 = tour.getCity(tourPos2);

            // Swap them around
            tour.setCity(tourPos2, city1);
            tour.setCity(tourPos1, city2);
        }
    }
}

// Selects candidate tour for crossover
private static Tour tournamentSelection(Population pop) {
    // Create a tournament population
    Population tournament = new Population(tournamentSize, false);
    // For each place in the tournament get a random candidate tour and
    // add it
    for (int i = 0; i < tournamentSize; i++) {
        int randomId = (int) (Math.random() * pop.populationSize());
        tournament.saveTour(i, pop.getTour(randomId));
    }
    // Get the fittest tour
    Tour fittest = tournament.getFittest();
    return fittest;
}
}

```

Population.java

```
package geneticalgo;

/**
 *
 * @author sarah
 */
public class Population {
    /**
     * Population.java
     * Manages a population of candidate tours
     */

    // Holds population of tours
    Tour[] tours;

    // Construct a population
    public Population(int populationSize, boolean initialise) {
        tours = new Tour[populationSize];
        // If we need to initialise a population of tours do so
        if (initialise) {
            // Loop and create offspring route
            for (int i = 0; i < populationSize(); i++) {
                Tour newTour = new Tour();
                newTour.generateIndividual();
                saveTour(i, newTour);
            }
        }
    }

    // Saves generated offspring route
    public void saveTour(int index, Tour tour) {
        tours[index] = tour;
    }

    // Gets a route from population
    public Tour getTour(int index) {
        return tours[index];
    }

    // Gets the best route in the population
    public Tour getFittest() {
        Tour fittest = tours[0];
        // Loop through cities to find fittest
        for (int i = 1; i < populationSize(); i++) {
            if (fittest.getFitness() <= getTour(i).getFitness()) {
                fittest = getTour(i);
            }
        }
        return fittest;
    }

    // Gets population size
    public int populationSize() {
        return tours.length;
    }
}
```

Tour.java

```
package geneticalgo;

/**
 *
 * @author sarah**/

import java.util.ArrayList;
import java.util.Collections;

public class Tour {
    /*
    * Tour.java
    * Stores a candidate tour
    */

    // Holds our tour of cities
    private ArrayList tour = new ArrayList();
    // tour/ possible route is represented in array
    private double fitness = 0; // fitness of the route, the higher the better
    private int distance = 0; // distance of the city

    // Constructs a blank tour
    public Tour(){
        for (int i = 0; i < TourManager.numberOfCities(); i++) {
            tour.add(null);
        }
    }

    public Tour(ArrayList tour){
        this.tour = tour;
    }

    // Creates a random individual (select any initial city)
    public void generateIndividual() {
        // Loop through all our destination cities and add them into the pool
        for (int cityIndex = 0; cityIndex < TourManager.numberOfCities(); cityIndex++) {
            setCity(cityIndex, TourManager.getCity(cityIndex));
        }
        // Randomly reorder the tour (random selection)
        Collections.shuffle(tour);
    }

    // Gets a city from the tour
    public City getCity(int tourPosition) {
        return (City)tour.get(tourPosition);
    }

    // Sets a city in a certain position within a tour
    public void setCity(int tourPosition, City city) {
        tour.set(tourPosition, city);
        // If the tours been altered we need to reset the fitness and distance
        fitness = 0;
        distance = 0;
    }

    // Gets the tours fitness
    public double getFitness() {
        if (fitness == 0) {
            fitness = 1/(double)getDistance();
        }
        return fitness;
    }

    // Gets the total distance of the tour
    public int getDistance(){
        if (distance == 0) {
            int tourDistance = 0;
            // Loop through all possible cities in the routes
            for (int cityIndex=0; cityIndex < tourSize(); cityIndex++) {
                // Get last point of city where are travelling from
            }
        }
    }
}
```

```

        City fromCity = getCity(cityIndex);
        // City we're travelling to
        City destinationCity;
        // Check if not last city last city, then set course to travel back to initial city
        if(cityIndex+1 < tourSize()){
            destinationCity = getCity(cityIndex+1);
        }
        else{
            destinationCity = getCity(0);
        }
        // Get the distance between the two cities
        tourDistance += fromCity.distanceTo(destinationCity);
    }
    distance = tourDistance;
}
return distance;
}

// Get number of cities on our tour
public int tourSize() {
    return tour.size();
}

// Check if the tour contains a city
// Reconfirm to ensure that none of the city is missing (mutation probability)
public boolean containsCity(City city){
    return tour.contains(city);
}

@Override
public String toString() {
    String geneString = "|";
    for (int i = 0; i < tourSize(); i++) {
        geneString += getCity(i)+"|";
    }
    return geneString;
}
}

```

Tour Manager.java

```
package geneticalgo;

/**
 *
 * @author sarah**/

import java.util.ArrayList;
import java.util.Collections;

public class Tour {
    /*
    * Tour.java
    * Stores a candidate tour
    */

    // Holds our tour of cities
    private ArrayList tour = new ArrayList();
    // tour/ possible route is represented in array
    private double fitness = 0; // fitness of the route, the higher the better
    private int distance = 0; // distance of the city

    // Constructs a blank tour
    public Tour(){
        for (int i = 0; i < TourManager.numberOfCities(); i++) {
            tour.add(null);
        }
    }

    public Tour(ArrayList tour){
        this.tour = tour;
    }

    // Creates a random individual (select any initial city)
    public void generateIndividual() {
        // Loop through all our destination cities and add them into the pool
        for (int cityIndex = 0; cityIndex < TourManager.numberOfCities(); cityIndex++) {
            setCity(cityIndex, TourManager.getCity(cityIndex));
        }
        // Randomly reorder the tour (random selection)
        Collections.shuffle(tour);
    }

    // Gets a city from the tour
    public City getCity(int tourPosition) {
        return (City)tour.get(tourPosition);
    }

    // Sets a city in a certain position within a tour
    public void setCity(int tourPosition, City city) {
        tour.set(tourPosition, city);
        // If the tours been altered we need to reset the fitness and distance
        fitness = 0;
        distance = 0;
    }

    // Gets the tours fitness
    public double getFitness() {
        if (fitness == 0) {
            fitness = 1/(double)getDistance();
        }
        return fitness;
    }

    // Gets the total distance of the tour
    public int getDistance(){
        if (distance == 0) {
            int tourDistance = 0;
            // Loop through all possible cities in the routes
            for (int cityIndex=0; cityIndex < tourSize(); cityIndex++) {
                // Get last point of city where are travelling from
                City fromCity = getCity(cityIndex);
```



```

        // City we're travelling to
        City destinationCity;
        // Check if not last city last city, then set course to travel back to initial city
        if(cityIndex+1 < tourSize()){
            destinationCity = getCity(cityIndex+1);
        }
        else{
            destinationCity = getCity(0);
        }
        // Get the distance between the two cities
        tourDistance += fromCity.distanceTo(destinationCity);
    }
    distance = tourDistance;
}
return distance;
}

// Get number of cities on our tour
public int tourSize() {
    return tour.size();
}

// Check if the tour contains a city
// Reconfirm to ensure that none of the city is missing (mutation probability)
public boolean containsCity(City city){
    return tour.contains(city);
}

@Override
public String toString() {
    String geneString = "|";
    for (int i = 0; i < tourSize(); i++) {
        geneString += getCity(i)+"|";
    }
    return geneString;
}
}

```