UNIVERSITI TUNKU ABDUL RAHMAN

Assignment 1

Course Code: UECS2153

Course Name: Artificial Intelligence

Lecturer: Dr. Ng Oon-Ee

Academic Session: 2019/05

Title: Genetic Algorithms

Student ID	Student Name	Major
16UEB03890	Chin Kai Xiang	Software Engineering (SE)

Problem Definition

The problem that I am going to solve in this project is travelling salesman problem (TSP) which is an NP-hard problem. The question posted by TSP is "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?". TSP is frequently used to study the problems involved in finding an optimal solution from a finite set of possible sets of solutions. Therefore, it is also frequently used to benchmark the performance of various optimization methods such as genetic algorithms, simulated annealing, particle swarm optimization and so on.

In its purest formulation, TSP concept can widely be applied in many fields such as planning, logistics as well as microchip manufacturing. One of the famous applications of TSP is DNA sequencing. In this application, the concept city in TSP represents a DNA fragment whereas the similarity measure between DNA fragments was represented by the concept distance which is the travelling cost in TSP.

Using the brute force approach, to search for an optimal solution for TSP requires us to test all the possible routes which are 14051! = 1.611458753e+54049 possible routes. This process will require a computer with very high processing power and will consume a lot of times and spaces. Therefore, in order to find a near optimal solution in this extremely complex search space, it is better to use a metaheuristic approach and, in this case, we will use the genetic algorithm in order to determine the best route chromosome i.e. with the shortest distance.

Genetic Algorithm for Traveling Salesman Problem

1. Initialization

In this step, we will create a large population of randomly generated chromosomes i.e. routes. Each chromosome will represent a possible route which is a sequence of cities. Each gene in the chromosome represents a city with x, y coordinates. Last but not least, each chromosome or route will have a distance and fitness.

2. Selection

a) Parent Selection

The following are the two parent selection methods that I have used in this project:

i. **Roulette Wheel Selection**

The first parent selection method that I used is called roulette wheel selection. In this approach, every chromosome will be given a selection probability based on its fitness. Thus, the fitter chromosome will have a greater chance to be selected as a parent for mating and transferring their features to the next generation. The selection pressure in this approach is high as it tends to select individuals with higher fitness in the hope to produce better chromosomes over time. In roulette wheel selection, each chromosome gets a portion of the wheel that is proportional to its fitness value. A fixed point is chosen on the wheel circumference and the wheel is rotated. The region of the wheel which comes in front of the fixed point is chosen as the parent and the same process is repeated until a desired size of the mating pool is reached.

ii. **Tournament Selection**

In the tournament selection approach, we will randomly select K chromosomes from the current population and run a tournament among them. From the sample chosen, the fittest chromosome will be selected as a parent for mating purpose. This process is repeated until we get the desired mating pool size. The selection pressure i.e. the degree of fitness influence chromosome selection is dependent on the tournament size, K. With larger K, weak candidates will have a smaller chance of being selected as it has to compete with many stronger candidates while smaller K will increase the degree of randomness. Last but not least, tournament selection works for negative fitness values as well.

b) Survivor Selection

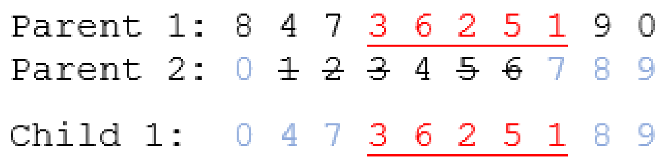
The survivor selection method that I have used is fitness-based survivor selection. In this fitness-based selection method, the fittest chromosome from the current population will be selected and propagated to the next generation. This approach in the genetic algorithm is also known as elitism. This approach is used to ensure that good chromosomes i.e. solutions will not be lost due to crossover and this can drastically improve the performance (quality and speed) of the genetic algorithm search process.

3. Crossover

The following are the two crossover methods that I have used in this project:

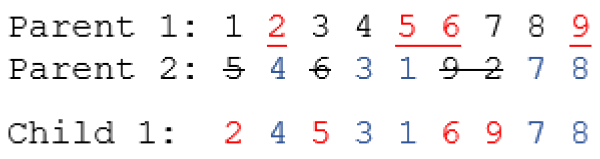
i. **Davis' Order Crossover (OX1)**

This crossover method constructs an offspring by randomly choosing a substring of one parent e.g. parent1 and copy this section to the offspring. After that, this method will copy the elements from another parent e.g. parent2 that is not in the offspring into the offspring. This method tends to preserve the relative ordering of elements in another parent e.g. parent2 when copying the elements to the offspring as it aims to transmit possible useful ordering information to the offspring. As a result, this method tries to strike a balance between improving the diversity of the offspring as well as to maintain some useful original sequence from the parents. The figure below shows the high-level view of OX1 process and note that the order of the elements in parent2 is preserved when its elements are copied to the child/offspring.


Parent 1: 8 4 7 3 6 2 5 1 9 0
Parent 2: 0 ~~1~~ ~~2~~ ~~3~~ 4 ~~5~~ ~~6~~ 7 8 9
Child 1: 0 4 7 3 6 2 5 1 8 9

ii. **Order-Based Crossover (OX2)**

Order-based crossover method (OX2) is a modification of which aims to solve the schedule problems in OX1. In OX2, several positions of one parent e.g. parent1 are randomly selected. After that, the elements in another parent e.g. parent2 which is not in selected elements will be copied to the child at the original positions. Finally, the selected elements will be copied into the remaining empty spaces. Notice that, in this case, the relative ordering of the elements in parents are not preserved.


Parent 1: 1 2 3 4 5 6 7 8 9
Parent 2: ~~5~~ 4 ~~6~~ 3 1 ~~9~~ ~~2~~ 7 8
Child 1: 2 4 5 3 1 6 9 7 8

4. Mutation

The following are the two mutation methods that I have used in my project:

i. **Shuffle Mutation**

In shuffle mutation, a subset of genes is selected at random and their values are shuffled or scrambled. Shuffle mutation represents permutation-based encodings.

ii. **Inversion Mutation**

In inversion mutation, a subset of genes is selected and the entire sequence is inverted or reversed.

Analysis of Results

1. Parent Selection Methods

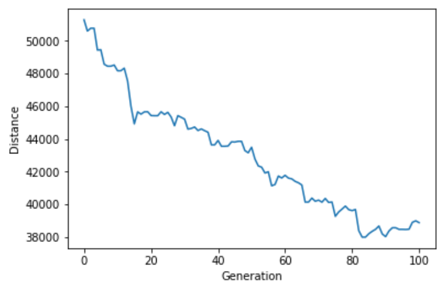
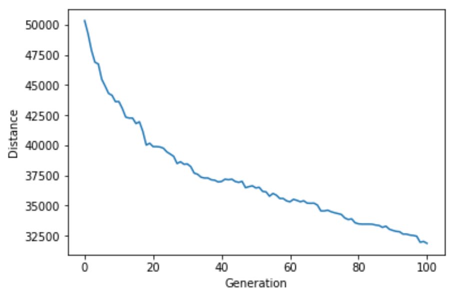
Parent Selection	***Roulette Wheel Selection***	***Tournament Selection***
Results	`'Best distance for population in iteration 99: 38883.85114310484'` 	`'Best distance for population in iteration 99: 31873.746251738965'` 

Table 1.1

Based on table 1.1, we can conclude that the genetic algorithm (GA) search process using tournament selection gave a shorter distance than roulette wheel selection. This is most probably because roulette wheel selection tends to apply higher selection pressure towards individuals with higher fitness values. As a result, the fittest chromosomes in the population will always be selected and this leads to the mating of similar chromosomes decreasing the diversity of the next generation. On the other hand, tournament selection with optimum K can strike a balance between fitness influence and randomness on the selection of individuals for mating. Therefore, tournament selection can neutralize the selection pressure and can improve the diversity of the population at the same time ensuring the quality of the population (high fitness).

Moreover, since the roulette wheel applies high selection pressure, thus, the convergence rate will be higher and might be causing premature convergence. Premature convergence will prevent the GA to explore and find more optimal solutions because the diversity of the population tends to be low when selection pressure is high as above mentioned.

2. Crossover Methods

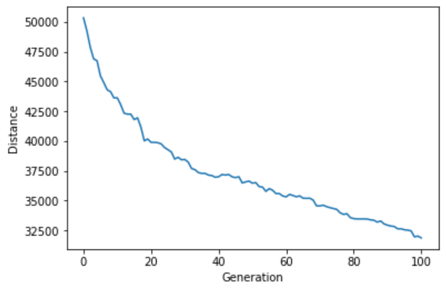
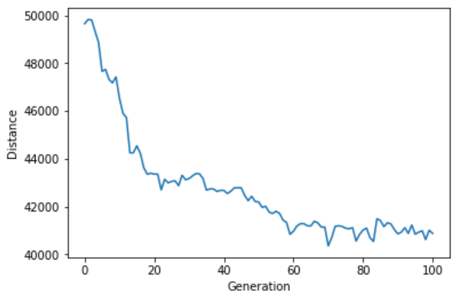
Crossover	***Davis' Order Crossover (OX1)***	***Order-Based Crossover (OX2)***
Results	`'Best distance for population in iteration 99: 31873.746251738965'` 	`'Best distance for population in iteration 99: 40879.32795803254'` 

Table 1.2

Based on table 1.2, we can conclude that the genetic algorithm (GA) search process using Davis' Order Crossover (OX1) gave a shorter distance than Order-Based Crossover (OX2). Based on the explanation above of how these two crossover methods work, we know that Davis' Order Crossover tries to preserve the relative order of the elements in the parents and transmit the ordering information to the offspring. This may be potentially useful in some situations as some of the sequences of genes in the chromosome might be good i.e. giving the shorter total distance between cities and we might want to maintain those sequences of genes in the next generation.

Order-based crossover (OX2), on the other hand, does not preserve and transmit the ordering information of elements in parents to the offsprings but OX2 promotes better diversity as the sequences of genes in offsprings tend to be more different than the that of parents. However, higher diversity might cause GA not able to find an optimal solution as the sequence of genes in chromosome changed a lot each new generation.

As a conclusion, the result using OX1 is better than that of OX2 most probably because of the retention of useful ordering information of parents in new generations. Moreover, we can see that the line graph for OX2 is fluctuating more than that of OX1 because each time the gene sequence changes a lot and therefore, the distance changes a lot too in each generation.

3. Mutation Methods

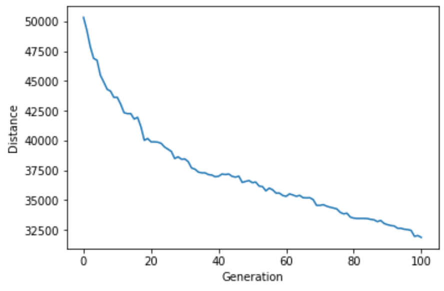
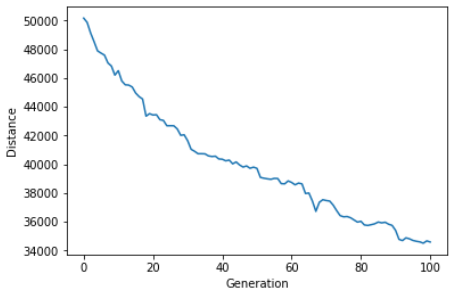
Mutation	***Shuffle Mutation***	***Inverse Mutation***
Results	`'Best distance for population in iteration 99: 31873.746251738965'` 	`'Best distance for population in iteration 99: 34592.36412706226'` 

Table 1.3

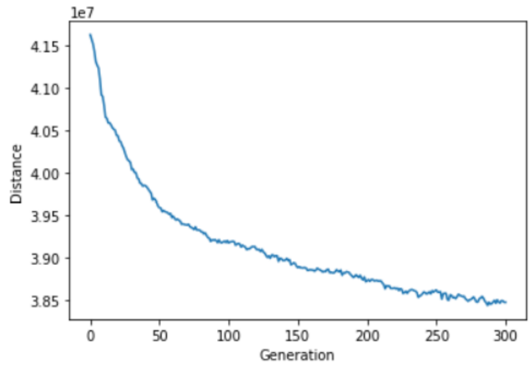
Based on table 1.3, we can conclude that the GA search process using shuffle mutation gave a shorter distance than the inversion mutation. As its name implied, shuffle mutation will shuffle or scramble the selected sequence of values in the chromosome. Indeed, this increases the diversity of the new population but at the same time, this might cause some good sequences of genes in the chromosomes to be altered. However, since we knew that mutation will only alter a small portion of the chromosome so the sequence of genes in the chromosomes will not be altered too much.

On the other hand, the inverse mutation will inverse the selected sequence of genes in the chromosome. In inverse mutation, the diversity of the new population will be lower than that of shuffle mutation as most of the cities are still connected to the same neighbour cities.

The Verdict

As a conclusion, I will construct my genetic algorithm using the combination of best methods which are tournament parent selection, Davis's Order Crossover (OX1) and shuffle mutation. The result that I have obtained by using this combination of methods is as follow:

```
'Best distance for population in iteration 299: 38470624.45009706'
```



```
'Time taken: 50599.10520219803 s'
```

The distance that I have obtained is 38470624.45. Note that the convergence starts to occur roughly from 210th iteration and onwards. The parameters that I have used are as follow:

1. Population size: 60
2. Elite size: 15
3. Number of iterations: 300
4. Size of mutation: 2% of the cities in a route

The performance of this model is considered ok. The fitness of the model most probably can be improved more by tuning the parameters mentioned just now to an optimal value. This can be done by trying the model several runs with different values. However, the time efficiency of the model is quite poor as it took around 14 hours to run this GA with the parameter values mentioned above. This is most probably due to the logic of code in the crossover section and thus, more efficient logic should be investigated.

Last but not least, there are some suggestions about the tuning of parameters which should be able to improve the result of this GA. First of all, the number of populations should be adjusted to a higher value especially for this complex search space so that the diversity of the populations will be increased. With improved diversity, a more optimal solution can be found as well as to prevent premature convergence. The size of mutation should not be too high as it might cause large fluctuation in the result preventing the convergence of solutions. A balance between the number of populations and mutation size should be strived to prevent premature (too low population size) and slow convergence (too high mutation rate). The number of iterations should be higher if the population size is high in order to allow sufficient time for convergence of solutions.