School of Digital Technologies

CST3170

Developing Artificial Intelligence

The Travelling Salesman Problem: Friday 11th December 2020, 17:00hrs (End of Week 11)

Student ID: M00677939

Name: SEEROO Muhammad Shaad Ouwesh

CAMPUS: Mauritius

Tutor: Mr Suraj Juddoo

Table of Contents

Table of Figures

Introduction

This project is focused on using algorithms written with java to solve the traveling salesman problem (TSP). The salesman starts the journey in any city and visits the other cities only once. Once the trip ends, he should return to the initial city, making a complete circuit. This problem is known to be an NP problem and therefore is no algorithm to solve it in polynomial time (Leena & Amit, 2004). In this project, the following was used:

- **Adaptive Genetic Algorithm**
- **Simulated Annealing Algorithm**

Problem Definition

- The starting point can start at any given
- A city can only be visited once
- Optimal results need to be produced both in terms of time and distance
- All data sets should run in less than 1 min

Proposed Solution

Brute force algorithms are well known to provide optimal solutions. However, when it comes to large data sets and cities, the cost and time increase exponentially (Lumburovska, 2018). Secondly, nearest neighbour algorithms produce reasonable solutions in a very short amount of time. But the result depends on the starting point, and it might not offer the optimal solution but reach the worst case. The alternative proposed solution should be genetic algorithms due it produces better solutions in terms of cost compared to nearest neighbour according to a case study by Bisan, et al (Refer to Figure 1). It would be improved by using Adaptive Genetic Algorithms and using Multi-Heuristics to achieve performance improvements. Simulated annealing will decrease the mutation rate and/or crossover rate over a period.

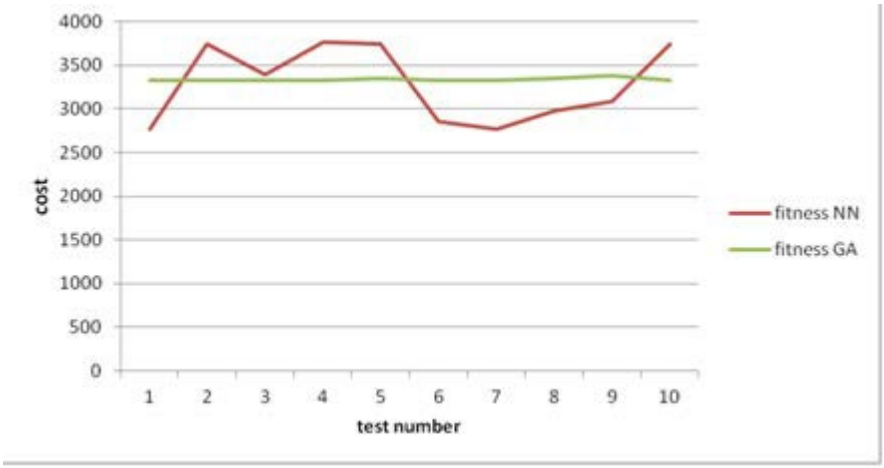
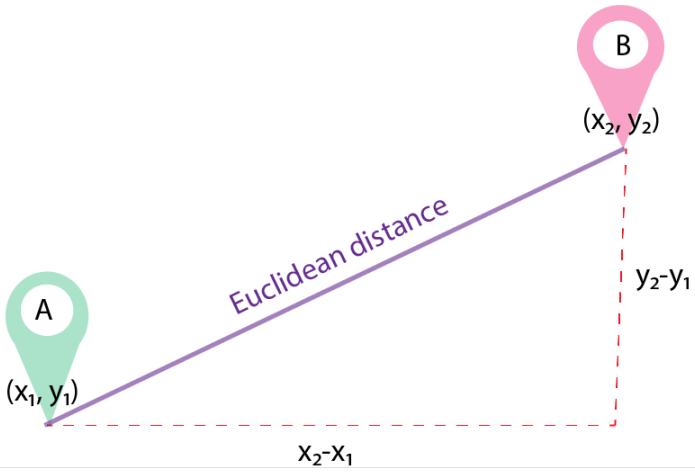


Figure 1-Cost comparison between NN and GA

To calculate the distance between the cities, the Euclidean distance formula is used:



(Banerjee, 2020)

Figure 2-Visual Representation of Euclidean Distance

$$D_e = \left(\sum_{i=1}^{n} (p_i - q_i)^2 \right)^{1/2}$$

(Banerjee, 2020)

Figure 3- Euclidean Distance Formula

Where,

n = number of dimensions

pi, qi = data points

Generic Algorithms

Pseudocode

BEGIN AGA

WHILE NOT STOP DO

BEGIN

Choose parents from the population.

Generate children from the chosen parents

Mutate the individuals

Expand the population combining children to it

Decrease the extended population

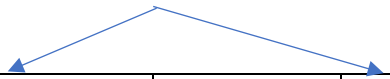
End

Output the best route

END AGA

Genetic algorithms are based on biological notions of evolution. There are some common terminologies.

- Population- set of candidate solution which utilises generic functions such as mutation and crossover.
  The larger the population, the better the search space, hence more optimised results. However, this would result in slower execution.
- Candidate Solution- possible result in a problem.
- Gene - blocks that are invisible to build up a chromosome. (1 and 0)
- Chromosome - a string of genes that makes up a possible solution.
- Mutation – method in which candidate solutions are changed to form new traits.
  A higher mutation rate would result in a variety of results. Lower mutation would result in the algorithm taking too long.
  A method of mutation would be to swap the genetic information at two points. An example is shown below.
  Individual:



1	**3**	5	**2**	5	4

  New individual:

1	**2**	5	**3**	5	4

- Crossover – a function of combining chromosomes to create a new candidate solution.
  Higher crossover value allows for a considerate number of new solutions to interact with the new generation.
  Below shows an example of a cross over
  Parent 1, Parent 2, and an offspring which is to be added.
  The values are compared

Parent 1

1	2	5	6	3	4

Parent 2

5	1	4	6	3	2

Offspring

		5	3	6	0

Values 2, 1 and 4 are missing from the offspring when compared to the parent 1 and parent 2

Parent 1

1	2	5	6	3	4

Parent 2

5	1	4	6	3	2

Offspring

1	2	5	3	6	4

- Selection- choosing candidate solutions for the next breeding of solutions

- Fitness- means to measure how a candidate solution has been adapting given the problem. Improved fitness function would be an average of the current fitness function when induvial fitness is greater than average fitness. The formula can be found below:

$$p_m = (f_{max} - f_i) / (f_{max} - f_{avg}) * m, f_i > f_{avg}$$
$$p_m = m, f_i £ f_{avg}$$

f_{max} -Best fitness of population

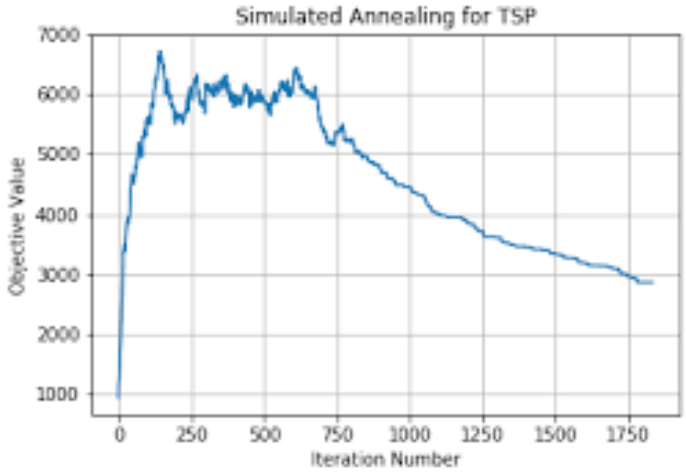
f_i – Current individual fitness

f_{avg} – Fitness Average

It is a well-known formula to improve results and it stable, it can also be used to improve the TSP solution (Lee & Burak, 2015).

Simulated annealing

Simulated annealing is an efficient and generalised form of optimisation. The algorithm is a hill-climbing, but it does not pick the best move but a random one. The selected solution is either approved, or it chooses one with probability less than one. In TSP, the mutation and cross over start with a high rate then progressively decreasing it.



(buseyarentekin, 2020)

Figure 4-Simulated annealing TSP

Figure 4 shows that using Simulated annealing the number of iterations decreases, and it thus makes the algorithm more efficient in terms of cost. Therefore, it was used to improve the current algorithm.

Pseudocode

BEGIN AGA

WHILE NOT STOP DO

BEGIN

Choose parents from the population.

Generate children from the chosen parents

IF Mutation rate is greater than temperature

Mutate the individuals

Expand the population combining children to it

Decrease the extended population

End

Output the best route

END AGA

Limitations

One of the main limitations is to choose the correct population and generations for the algorithms. Increasing the values will bring more optimal solutions; however, the time would be affected, and it would take more computational power. Furthermore, the algorithm was tested using 40 cities +, and the results gap was considerate, ranging from 1000 to 8000. The configuration parameters could be improved based on the number of cities available, and thus results could be improved in both in terms of time and best route.

Program Structure

TSP_CW Class

The class will run the program. It will use FileManagement, GeneticAlgorithm, City.

Stopwatch Class

This class starts the timer and stops it when required.

City Class

This class contains the values for the position, x, and y coordinates. Additionally, it has distanceFrom() which calculates the distance from two cities.

GeneticAlgorithm Class

This class contains methods and variable to handle the operations such as crossover, mutation, fitness, and check if the algorithm termination condition.

Individual Class

This class will keep one candidate solution, stores and modifying chromosomes. The class also keeps track of individual's fitness value.

Population Class

This class keeps track of the array of Induvials and methods such as getFittest() and setIndividual(). Additionally, the population's total fitness is stored.

Route Class

This class stores the route and calculates the distance of each way.

Analysis of Parameter values

According to (Rexhepi, et al., 2013) increasing mutation value does not contribute to a better fitness result as it increases exclusion time and cost. Furthermore, the larger the preliminary population, the fitness value dropped. Simulated annealing was introduced in the algorithm, and therefore the mutation value would need to change. After some trial tests, the mutation rate of 0.004 was found to give more optimal solutions compared to the previous values.

Reflection

The first challenge that was faced reading from the files; they had different delimiters, and therefore it wasn't easy to take care of all these different file structures. Space was used as a delimiter so that all the files have the same standards. Secondly was choosing an algorithm based on the solution and time it takes. After some research and testing, most algorithms were struggling with the 40+ cities files and taking more than one minute. After some research, an algorithm was chosen and implemented.

Furthermore, it was a challenge to test the average fitness function, whether it was correct or not. Google OR tools was used to make sure that the paths are corrected and working. The cooling temperature value was destinated by trial-and-error basis. The genetic algorithm was chosen to explore how it works, and thus this knowledge could be used to expand and solve other complex problems. It was a challenge to bring a solution to such a problem and satisfy both time and optimal distance.

Results

Below are the results for the cities. The time is in nanoseconds.

Test data

Test File 1

```
Enter your file name, make sure it is in project folder: test1tsp.txt
Algorithm Started
Start Distance: 26.069053496924262
Finished after: 911306.4
Stopped after 4000 generations.
Best distance: 24.293023070189598
Best Route:4,2,3,1,
```

```
Enter your file name, make sure it is in project folder: test1tsp.txt
Algorithm Started
Start Distance: 26.069053496924262
Finished after: 725370.7
Stopped after 4000 generations.
Best distance: 24.293023070189598
Best Route:4,2,3,1,
```

Test File 2

```
Enter your file name, make sure it is in project folder: test2atsp.txt
Algorithm Started
Start Distance: 80.88885995272898
Finished after: 883496.2
Stopped after 4000 generations.
Best distance: 65.65395780324545
Best Route:8,7,1,3,2,4,6,5,
```

```
Enter your file name, make sure it is in project folder: test2atsp.txt
Algorithm Started
Start Distance: 80.88885995272898
Finished after: 926626.9
Stopped after 4000 generations.
Best distance: 65.65395780324545
Best Route:7,8,5,6,4,2,3,1,
```

Test File 3

```
Enter your file name, make sure it is in project folder: test3atsp.txt
Algorithm Started
Start Distance: 450.6746633601859
Finished after: 935549.8
Stopped after 4000 generations.
Best distance: 229.50916652583456
Best Route:1,2,9,6,8,3,4,5,7,
```

```
Enter your file name, make sure it is in project folder: test3atsp.txt
Algorithm Started
Start Distance: 450.6746633601859
Finished after: 910810.8
Stopped after 4000 generations.
Best distance: 229.50916652583456
Best Route:8,6,9,2,1,7,5,4,3,
```

Final data

Final data 1

```
Enter your file name, make sure it is in project folder: test1-20.txt
Algorithm Started
Start Distance: 1042.4060998275854
Finished after: 1121796.7
Stopped after 4000 generations.
Best distance: 463.95774842296623
Best Route:10,11,9,2,12,7,1,3,5,8,4,6,
```

```
Enter your file name, make sure it is in project folder: test1-20.txt
Algorithm Started
Start Distance: 1042.4060998275854
Finished after: 1084522.0
Stopped after 4000 generations.
Best distance: 463.95774842296623
Best Route:3,5,8,4,6,10,11,9,2,12,7,1,
```

Final data 2

```
Enter your file name, make sure it is in project folder: test2-20.txt
Algorithm Started
Start Distance: 1521.4681059848917
Finished after: 1318175.5
Stopped after 4000 generations.
Best distance: 733.5568464527814
Best Route:6,11,10,3,13,4,2,7,12,9,14,1,8,5,
```

```
Enter your file name, make sure it is in project folder: test2-20.txt
Algorithm Started
Start Distance: 1521.4681059848917
Finished after: 1148118.8
Stopped after 4000 generations.
Best distance: 733.5568464527814
Best Route:5,8,1,14,9,12,7,2,4,13,3,10,11,6,
```

Final data 3

```
Enter your file name, make sure it is in project folder: test3-20.txt
Algorithm Started
Start Distance: 313171.9337271361
Finished after: 1210825.4
Stopped after 4000 generations.
Best distance: 135664.84173227008
Best Route:6,11,14,3,15,10,4,12,1,17,13,7,8,18,9,2,5,16,
```

```
Enter your file name, make sure it is in project folder: test3-20.txt
Algorithm Started
Start Distance: 313171.9337271361
Finished after: 1328076.1
Stopped after 4000 generations.
Best distance: 135664.84173227008
Best Route:2,5,16,6,11,14,3,15,10,4,12,1,17,13,7,8,18,9,
```

Final data 4

```
Enter your file name, make sure it is in project folder: test4-20.txt
Algorithm Started
Start Distance: 2979112.067591119
Finished after: 1843943.8
Stopped after 4000 generations.
Best distance: 602852.0119077646
Best Route:14,28,10,26,20,11,32,30,5,17,2,23,6,9,4,22,7,16,8,24,27,21,19,15,13,12,18,31,3,1,25,29,
```

```
Enter your file name, make sure it is in project folder: test4-20.txt
Algorithm Started
Start Distance: 2979112.067591119
Finished after: 1731550.0
Stopped after 4000 generations.
Best distance: 607342.0469886772
Best Route:19,21,27,24,8,16,7,22,4,9,6,23,2,17,5,30,32,11,20,26,10,28,14,29,25,1,31,12,18,3,15,13,
```

Self-Marking Sheet

Point	Self	Reason	Area
10	10	I have been fair and critical about the marks I allocated.	Self-Marking Sheet
10	10	The test files were run serval times, and the result was the output was the same.	Solve First Training Problem
10	10	The test files were run serval times, and the result was the output distance was the same, but the routes were different.	Get Optimal Result for All Three Training Problems.
10	10	The algorithm was described, and the methods used were to explain how they work using examples.	Describe Algorithm(s) Used
10	8	The code is clean and well organised.	Quality of Code
20	16	The files were run several times, and output was similar in terms of distance only the routes changed.	Get Optimal Results for the First Three Tests
20	18	The files were run several times, and output was similar in terms of distance only the routes changed and it was under one minute.	Get Optimal Results for First Three Tests in under a minute.
10	8	The result is optimal however, the time could be improved	Best system on Fourth Test (Path length times time.)

References

Banerjee, W., 2020. *Medium.* [Online]
Available at: https://medium.com/analytics-vidhya/role-of-distance-metrics-in-machine-learning-e43391a6bf2e
[Accessed 10 Novemeber 2020].

Bisan, A., Marzieh, B. & Ibrahim, V., 2012. A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem. *International Journal of Computer Science and Electronics Engineering,* Volume 1, pp. 34-38.

buseyarentekin, 2020. *Global AI Hub.* [Online]
Available at: https://globalaihub.com/simulated-annealing-algorithm/
[Accessed 10 Novemeber 2020].

Leena, J. & Amit, B., 2004. Traveling Salesman Problem: A Case Study. *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY,* Volume 3, pp. 167-169.

Lumburovska, L., 2018. *Time-Efficient String Matching.* [Online]
Available at: http://eprints.fri.uni-lj.si/4287/1/63150349-LINA_LUMBUROVSKA-%C4%8Casovno_u%C4%8Dinkoviti_algoritmi_ujemanja_nizov_in_metoda_grobe_sile.pdf
[Accessed 15 Novemeber 2020].

Rexhepi, A., Maxhuni, A. & Dika, A., 2013. Analysis of the impact of parameters values on the Genetic Algorithm for TSP. *International Journal of Computer Science Issues,* Volume 10, pp. 158-164.