Using Genetic Algorithms to solve the Travelling Salesperson Problem

Millan Uka  
17981567

*Abstract*— In this paper I present a Genetic Algorithm to solve the Travelling Salesperson Problem for two hundred randomly generated cities on a map of the United States of America. I implement a Genetic Algorithm, which goal is to determine the optimal route with the shortest distance for a salesperson to travel. I will also provide an analysis of different Genetic Algorithm parameters such as Selection Method, Scaling Method, Population Size of the chromosomes and max number of generations.

Keywords—Genetic Algorithms, Nature Inspired Computing, Evolutionary Algorithms, The Salesperson Problem,

# Introduction (*Heading 1*)

The Travelling Salesperson/Salesman problem (TSP) is a well-established problem. The problems involves a number of cities with varying distances between each city. The idea behind this problem is to determine the best route for a salesperson to take where they travel the least amount of distance, assuming the salesperson must visit each city at least once. Although this problem, on the surface, looks simple. It is incredibly difficult to solve and requires intense computational capacity,

I will present a genetic algorithm (GA) to solve The Traveling Salesperson problem. A genetic algorithm is a nature inspired algorithm inspired by the process of natural selection. It is mostly used for optimisation and search problems. Each GA initially begins with a population of randomised individuals, called, chromosomes, which contains certain parameters. The algorithm simulates biological processes such as reproduction, mutation and crossover. The GA will repeat these processes repeatedly. For each iteration or generation, the algorithm will select the most promising individuals (in accordance to Darwin’s theory of natural selection) to survive and breed for the next generation. It determines the most promising and fittest individuals, by calculating their respective fitness value, and using this value to select the fittest. This simulates Charles Darwin’s theory of Natural Selection and the principle of “Survival of the fittest.”

In this paper. I present a specific implementation of The Salesperson Problem with two hundred cities. These cities are randomly generated and randomly located on a simplified map of the United States of America. I use a genetic algorithm to find the shortest route between each city. I will investigate and also provide an analysis of the different parameters such as varying Scaling and Selection Methods. And the effects of increasing population size and increasing the number of generations that the simulation can run for. I investigate and analyse these changes in these parameters and provide a final GA with optimal parameters for a network of two hundred cities.

# TRAVELLING SALESPERSON PROBLEM

That Salesperson Problem is a well-known problem. The definitive origin of the problem is unknown, but it is thought to originate from an Austrian mathematician, Karl Menger. The problem was technically first published in 1948 by Merril Meeks Flood [1], since than the problem has been widely studied and researched. It is a simple problem to conceptualise and understand, but extremely difficult to solve. Requiring a lot of computational resources to solve. The Salesperson Problem is a situation where there is number of nodes(cities). With varying distances between each node. The goal of The Travelling Salesperson Problem is to find the optimal route between all the cities in the problem, assuming that every single city has to be visited at least once.

One approach to try and solve the TSP is to do an exhaustive search by calculating all the distances for every route and find the solution with the lowest distance, however there can be many possible routes in TSP. The number of possible solutions can be express by the equation *(n - 1)! / 2*. Where *n* is number of cities. If we were to do this for five cities the number of possible solutions will be *(5 - 1)! /2* which is 12*.* Since there are only 12 possible solutions it would be easy to find the best possible solution. However, with the specific situation in this paper, of two hundred cities. The number of possible routes can be expressed as *(200-1)!/2* which will be 1.97 x 10372 possible solutions. This would be virtually impossible with our current computational resources. Therefore, an alternative approach must be taken. This is the reason, for in which I will use a GA to solve this problem.

# Methodology

A GA is inspired by Charles Darwin’s theory of natural selection. So, the algorithm’s methodology takes inspiration from this. In TSP a chromosome is a solution meaning that it is a specific route. The algorithm for a GA is:

### Intialisation of an Intial State

The algorithm initialises a randomised initial state. The initial population is made of randomised chromosomes. The population size will be set to a specified number, which determines the number of chromosomes that survive each generation.

### Evaluation of fitness

A fitness value can be determined by a fitness function. The fitness function will determine a value. This value in The Travelling Salesperson Problem is the total distance travelling on specific route. It is calculated by using the distances between cities. Since the fitness value is the total distance of a route, the individuals with a smaller fitness value in TSP, would be considered fitter.

### Reproduction

In adherence to the theory of Natural Selection the fittest individuals are the most likely to survive and reproduce. There is two main ways for changes in chromosomes to occur. One is mutation, which is just a random change in the chromosome, And the other is crossover which is when two chromosomes swap genetic information to create new offspring.

### Termination Condition and Iteration

The algorithm will check the current solution with the

termination condition. The termination condition can include factors like max number of generations and number of generations before any changes in the best solution. If the termination condition is reached the algorithm will terminate. If the termination condition is not reached, the simulation will continue, and the individuals will reproduce again, until the termination condition is met.

# implementation

### Program Details

The program is written in MATLAB. It is a modified version of a Travelling Salesman Problem provided as an example with MATLAB. The original example was first ran once, to randomly generate the locations and distances matrices for 200 cities. The variables are saved to a local file called “variables.mat”. Which are loaded in when the program starts.. This ensures that the locations and the distances are fixed and cannot change.

I will also be using the optimtool provide by MATLAB GA Toolbox. This allows me to change the parameters of the genetic algorithm. The parameters I will change are Selection and Scaling Methods. population size and max number of generations. I will change these parameters to determine the desired parameters that return the optimal result. The optimtool allows me to maintain accuracy and control over the trials that are undergone.

I will run at least ten trials for each change in the parameters, and record results. Each trial will produce a fitness value(fval) which is the total distance in kilometres(km) the salesperson has to travel. I will then use the average fitness values to determine the best parameters for the GA.

# Results and Discussion

## Default Parameters

. The first set of trials I ran were for a network of 200 cities. The parameters were kept at their default of 60 population size, Rank Scaling and Stochastic Uniform selection methods and 500 max number of generations. The results are shown below

Table I. Results for 200 cities with default parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **trials** | **population size** | **fval** | **generations** |
| 1 | 60 | 30883.3 | 500 |
| 2 | 60 | 29906.8 | 500 |
| 3 | 60 | 32180 | 500 |
| 4 | 60 | 31983.5 | 500 |
| 5 | 60 | 28399.3 | 500 |
| 6 | 60 | 29086.9 | 500 |
| 7 | 60 | 31554.6 | 500 |
| 8 | 60 | 32389.5 | 500 |
| 9 | 60 | 30432.3 | 500 |
| 10 | 60 | 31542 | 500 |
| Average | 60 | 30835.82 | 500 |

As we can see in Table 1, the average fitness value is 30,835.82km. We will use these results as a benchmark. And compare our optimal parameters to these results.

## The effect of Population Size

One of the parameters that I found that decreased the average fitness value was the population size. I conducted two sets of ten trials for a network of 40 cities with a population size of 40 and 100 respectively. I also did another two set of ten trials for a network of 100 cities. The average fitness value for the 40 cities network, was 5,035.04km for 40 generations and 4,915.26km for 100 generations. The difference is only 119.78km. However, this difference is exacerbated when the number of cities increase. Below is a graph showing the difference between the average fitness values of 40 and 100 generations for a network of 100 cities.

Figure 1: The comparison of different population sizes on average fitness value for a network of 100 cities.

As we can see in Figure 1, the average fitness value for the population size of 40 is much higher at 13,780.61km while the average fitness value for 100 is 10,567.02km this gives a significant difference of 3,213.59km. So, we can see when the number of cities increases, having a larger population size seems to decrease the fitness value at a higher rate. This would mean it would be better for our network of 200 cities to have a larger population size of 100 rather than a small population size of 40. Further trials will have an increased population size of 100.

## The effects of different Scaling methods

I also found that the default scaling method of Rank Scaling. Was the best option for the network of 200 cities. I conducted an investigation, for the best scaling method. I found that Rank was the best out of three options. Which are the following Rank, Proportional and Top. To come to this conclusion, I ran six sets of ten trials. Two sets for each scaling method on 40 and 100 cities network respectively. The figures below show a graphical visualisation of the results.

Figure 2: The comparison of the effect of different scaling methods on average fitness value for a network of 40 cities.

Figure 3: The comparison of the effect of different scaling methods on average fitness value for a network of 100 cities.

As we can see from the Figure 2, although Rank has the lowest fitness value. The difference is not very large. With the average fitness value for Proportional having the highest average fitness value at 5,207.96km, compared to Top’s average being, 4994.90km. Rank has slightly better average at 4915.26km. From this alone it is difficult to determine which is the best scaling method. However, in Figure 3, we can see that there is a huge difference in the average fitness value, between the scaling methods. Proportional has a significant larger average fitness value at 16,705.71km. With Top having an average fitness value of 12,200.53km. With Rank having the best average fitness value at 10,567.02km. These results are consistent with other literature on the subject, in a paper from the University of Minnesota [2]. They also found that for small and uncomplicated problems the Top Scaling method was able to achieve a solution quickly and a fitness value close to Rank scaling. However, Top scaling in larger and complicated problems makes the GA perform worse than Rank scaling. This is because by default in MATLAB, Top scaling only keeps 40% of the fittest individuals to become parents [3], this leads to a lack of diversity and a increased amount of selection pressure and early convergence. This explains the similar results for the smaller network of 40 cities, and a much more varied results with the larger and more complicated network of 100 cities. Overall, this shows that for larger networks, Rank Scaling can make a larger impact. This would mean that Rank Scaling would be the best option for a network of 200 cities.

## The effects of different Selection methods

The optimal Selection method that I found for the TSP GA was the default selection method of Stochastic Uniform, however its impact is relatively minimal which will most likely only save a couple hundred kilometres in our specific implementation of TSP. To come to this, I investigated two other methods, Roulette and Tournament. I ran an additional four sets of ten trials for Roulette and Tournament, to determine the average fitness value.

Figure 4: The comparison of the effect of different selection methods on average fitness value for a network of 40 cities.

Figure 5: The comparison of the effect of different selection methods on average fitness value for a network of 40 cities.

As we can see from Figure 4. The difference between the average fitness values is very minimal. The average for Roulette is 4,924.61km, which has a similar value to Stochastic Uniform’s average fitness value of 4,915.26km. With Tournament having an average fitness value of 4,940.09km. As you can see the difference is minimal for the network of 40 cities This is further exemplified in the Figure 5. We can see there isn’t much difference between the selection methods either. With Stochastic Uniform having an average fitness value of 10,567.02km. Roulette having an average of 10,938.12km, and lastly Tournament’s average fitness value of 10,721.68km. The difference is relatively small between the different selection methods. An interesting thing to note is that Tournament Selection has a slightly lower average fitness value then Roulette, from previous literature, this should not be the case [4]. Tournament selection typically performs poorly for more complicated problem, which our TSP is. This is likely because this is a minimising problem, as we are trying to minimise the distance that the salesperson travels. Which Roulette selection usually has issues with. Potentially the Roulette selection, could perform better than Rank and Tournament selection, if we make modifications to the fitness functions and adding specifications for scaling parameters. This could be the subject of further research. However, our results show that for both networks show that the Stochastic Uniform selection method is most likely the best method for the network of 200 cities. Further trials will continue using Stochastic Uniform selection.

## The effects of the number of generations

A major parameter that I found that dramatically affected the results of the GA, was the max number of generations. Generally, for a larger network of cities. An increased number of generations that the GA can run for. Leads to a decreased average fitness value. To get to these results I conducted two sets of ten trials for a network of 40 and 100 cities respectively. With an increased number of max generations of 1000.

Figure 6: The comparison of the effect of the number of generations on average fitness value.

The graph in Figure 6, shows that for the network topology of 40 cities, that the increase in generations causes a relatively small decrease of average fitness value. From 4,915.26km to 4,913.57km, this is relatively insignificant, and does not justify the increase in the number of generations for smaller networks. However, for the larger network of 100 cities. The difference is relatively significant. For 500 generations the average fitness value is 10,567.02km, which decreases to 8,926.41km. This is a significant difference. From these results we can determine that increasing the max number of generations decreases the fitness value at a higher rate for larger networks. Since our network is 200 cities, we can expect that increasing the number of generations will lead to significant decrease in the average fitness value.

## The effect of the optimal parameters on the network of 200 cities

From the results that I have gathered from the trials that I have done; I found a set of parameters which I have deemed as optimal. These were compared against the results of the GA with the default parameters. The default parameters were 60 population size, Rank scaling, Stochastic Uniform Selection and 500 maximum number of generations. The optimal parameters I found were, 100 population size, Rank scaling, Stochastic Uniform selection and 1000 generations.

Figure 7: The effect of the default parameters and optimal parameters on average fitness value.

As shown in Figure we can see the average fitness value, of the GA with optimal parameters is much lower than with the default parameters. The average fitness value for the GA with the default parameters was 30,835.82km which decreases with the optimal parameters to 19,260.21km. This is a huge difference of 11,575.61km which is a 37.50% decrease in the average fitness value. From these results it’s clear that we have found that the increase in population size and the number of generations leads to a decrease in the fitness value.

## Further work on the effect of larger popluation sizes on the Genetic Algorithm

The population size I had for the previous optimal results was 100, however this may not be optimal. Population size is an important parameter. If the population size is too low, then the selection pressure could be too high leading to early convergence and poor results. If the population size is too high then the selection pressure could be too low, leading to poor results. After some experimentation I found a new optimal population size of 10,000. The results are displayed in Figure 8

Figure 8: The effect of an increased population size on fitness value.

As we can see from Figure 8 the increased number of population size has lowered the fitness value by a large amount for every trial. The average fitness value is 10,900.10km compared to the previous optimal parameters average of 19,260.21km, which is a difference of 8,360.11km. I found that GAs with a population size greater than 10,000 either produced similar results, or in some cases worse. This is a very large population size, which I found surprising, as I expected the selection pressure to be too low. However, I think that an extremely large population size performs well, because the number of possible solutions in this specific TSP, is extremely large at 1.97 x 10372 possible solutions.

# Conclusion

In conclusion, I have found that the optimal parameters for a network of 200 cities was a GA with 10,000 population size, Rank Scaling, Stochastic Uniform Selection and 1,000 generations. My findings have found Rank Scaling to still be the best Scaling method as it maintains genetic diversity, which is not achieved in Top and Proportional Scaling methods. Stochastic Selection is the best Selection method because Tournament performs poorly because of the large tournament size and Roulette performs badly in minimising problems. However, there might be some areas for further research, such as the effect of increasing the max number of generations to a number beyond 1000. Overall, I believe that I have found the optimal or near optimal parameters for the GA to solve this TSP for 200 cities.

##### References

1. N. Christofides, *Worst-case analysis of a new heuristic for the traveling salesman problem*, Academic Press, New York, 1976.
2. F. A. Sadjadi, *Comparsion of fitness scaling functions in genetic algorithms with applications to optical process*, Proceedings of SPIE Vol. 55, 2004, doi:10.1117/12.5639
3. Unknown, *Fitness Scaling*, MATLAB, retrieved from <https://au.mathworks.com/help/gads/fitness-scaling.html> on 15th April 2020
4. N. M. Razali, J. Geraghty, *Genetic Algorithm Performance with Different Selection Strategies in Solving TSP*,