Analysis of using Genetic Algorithm to solve the Traveling Salesman Problem with a modified mutation operator and population initialisation method

Avriaj Singh Bevli(18MA20009), Yuvraj Singh(18MA20053)

Abstract

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. Genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation. Among the wide range of problems which genetic algorithms are used to solve, they are particularly useful to solve optimisation problems that conventional algorithms may take a long time to solve because of the large search space.
The travelling salesman problem (TSP) is an NP hard combinatorial optimization problem. Genetic algorithms are useful for NP-hard problems. The genetic algorithm depends on population initialisation, selection criteria, crossover and mutation operators.
For genetic algorithms performance is highly dependent on these control parameters. Early research was carried out using constant control parameters to find optimal parameter values for GA. In more recent research, it was shown that the convergence rate can be increased by adaptable control parameters, e.g. mutation rate can be varied during the optimization run.

1 Introduction

The travelling salesman problem has been a topic that has been studied for a long time because of the extreme significance of the problem in problems relating to computer science and operations research. This paper investigates the effects on GA performance or the optimization results by balancing control parameters to the chromosome fitness. We propose mutation functions that increase the performance of the simple genetic algorithm. We analyse how population initialisation with other construction heuristic methods is a viable option to get decent results in a much shorter time. Further we find out how various mutation rates and initialisation methods perform on some standard data sets available for the travelling salesman problem in TSPLIB. We finally analyse how we should be looking to choose our set of parameters to suit the problem at hand the best. While there has been work on trying to solve the travelling salesman problem using genetic algorithms, we propose a few improvements in this paper. In this paper, we analyse the symmetric travelling salesman problem. The contributions of the paper shall be as follows:

- Propose a modified evolutionary approach to solve the travelling salesman problem, using the solution obtained from a construction heuristic method
- Propose custom mutation operators and initialisation methods designed for the travelling salesman problem and compare its performance with some of the standard mutation operators that have been used in literature to solve the travelling salesman problem
- Analyse the performance of our methodology compared to the standard approaches.
- Attempt to identify which set of parameters would be most suitable to our example of travelling salesman problem given the amount of accuracy required and the time at hand

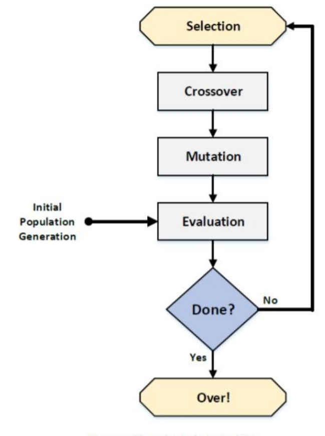


Figure 5. Flow chart of a typical GA.

2 Genetic Algorithm

2.1 Population initialisation

The population is generally assigned randomly and the the goal of the genetic algorithm is to improve the quality of the population over generations. Another approach is to use the solution obtained by another faster search heuristic to initialise the population. The solution can then be improved by the genetic algorithm over time. This greatly speeds up the convergence of the genetic algorithm and is a great option if not much time is available. Here, we will compare the random initialisation method with the case of initialising the population with the solution obtained from the nearest neighbour search heuristic.

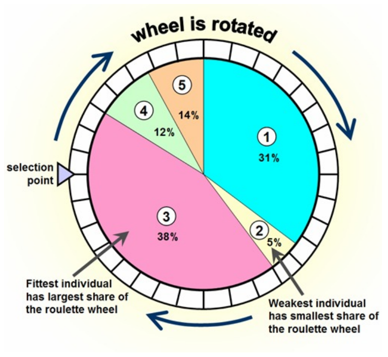
2.2 Fitness evaluation

We define a fitness function to evaluate how good the chromosome is in context of the problem at hand. The probability that an individual will be selected for reproduction is based on its fitness score. For the travelling salesman problem, we define the fitness of a chromosome to be proportional to the inverse of the total distance of the path.

$$Fitness = ScalingFactor/TotalPathlength$$

2.3 Selection

The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation. We use the Roulette wheel selection process here. The probability of each chromosome to be picked to be a part of the next generation is proportional to its fitness value


wheel is rotated
1
31%
2
5%
3
38%
4
12%
5
14%
selection point
Fittest individual has largest share of the roulette wheel
Weakest individual has smallest share of the roulette wheel

2.4 Crossover

Crossover is a significant phase in a genetic algorithm. Two chromosomes of the same generation exchange their genetic information to produce off springs. Taking inspiration from [8], we use the Partially Mapped Crossover(PMX). After choosing two random cut points on parents to build offspring, the portion between cut points, one parent's string is mapped onto the other parent's string and the remaining information is exchanged. Consider, for example, the two parents tours with randomly one cut point between 3rd and 4th bits and other cut point between 6th and 7th bits are as follows (the two cut points marked with "/"):

P 1 = (3 4 8 / 2 7 1 / 6 5) ,
P 2 = (4 2 5 / 1 6 8 / 3 7) .
The mapping sections are between the cut points. In this example, the mapping systems are 2 1, 7 6, and 1 8. Now two mapping sections are copied with each other to make offspring as follows:
O 1 = (x x x / 1 6 8 / x x) ,
O 2 = (x x x / 2 7 1 / x x) .
Then we can fill further bits (from the original parents), for those which have no conflict as follows:
O 1 = (3 4 x / 1 6 8 / x 5) ,
O 2 = (4 x 5 / 2 7 1 / 3 x) .
Hence, the first in the first offspring is 8 which comes from first parent but 8 is already in this offspring, so we check mapping 1 8 and see again 1 existing in this offspring, again check mapping 2 1, so 2 occupies at first . Similarly, the second in first offspring is 6 which comes from first parent but 6 exists in this offspring; we check mapping 7 6 as well, so 7 occupies at second . Thus the offspring 1 is:
O 1 = (3 4 2 / 1 6 8 / 7 5) .
Analogously, we complete second offspring as well:
O 2 = (4 8 5 / 2 7 1 / 3 6) .

2.5 Mutation

The genetic algorithm might get trapped in sub optimal solutions. A way to get out of these local minimas is to introduce mutation in the population, which helps maintain diversity within the population and prevent premature convergence. This is a very sensitive parameter and has considerable impact on the performance of the performance of the genetic algorithm. We use various types of mutation operators in this paper and compare their performance.
More on mutation has been discussed in section 4

3 Background, Preliminary, and Related Work

Over the years, researchers have suggested several approached to solve TSP most efficiently. [8] demonstrated the use of new crossover operator for solving the TSP with genetic algorithm and performed detailed analysis of the results. [7] and [6] analysed the use of the simple nearest neighbour construction heuristic search method to solve TSP. In [10] the 2-opt and Particle Swarm Optimization are used and the results are compared with those obtained using GA. Their results indicate that the 2-opt approach with the nearest neighbour as initial starting point for the search yields superior performance. [3] used the RGIBNNM mutation, which we use in our implementation. [9] talks about a 2-Opt based evolution strategy. We derive inspiration from this work and propose our 2-opt based mutation operator. Works [5] and [11] talk about a simulated annealing approach for TSP. We use the simple simulated annealing approach in our analysis. [2], [4], [1] providing an in depth study about potential mutation operators for the TSP and analysed the performance over data sets from TSPLIB.
But even with so much of research on this topic, there is not one mutation operator that is fit for all. Hence, we propose our mutation operator and analyse its performance versus other mutation operators. We also explore the strategy of non-random population initialisation and analyse it.

4 Readings

Given below are the readings we obtained upon testing the performance of the four mutation operators and the 2 initialisation methods on some of the standard data sets from TSPLIB:

Data set	Population Initialisation	Mutation	Solution distance	Time for convergence
eil51	Random	Custom	447	30s
eil51	Random	Random	476	500s
eil51	Random	RGIBNNM	456	500s
eil51	Random	SA	440	150s
eil51	NN	SA	441	120s
eil51	NN	Custom	443	40s
eil51	NN	RGIBNNM	478	40s
eil51	NN	Random	490	70s
st70	Random	Custom	740	50s
st70	Random	Random	767	450s
st70	Random	RGIBNNM	735	450s
st70	Random	SA	690	350s
st70	NN	SA	710	200s
st70	NN	Custom	730	100s
st70	NN	RGIBNNM	761	200s
st70	NN	Random	760	200s
bayg29	Random	SA	9080	20s
bayg29	Random	Random	9546	30s
bayg29	Random	RGIBNNM	9540	20s
bayg29	Random	Custom	9190	20s
bayg29	NN	Custom	9100	20s
bayg29	NN	Random	9490	30s
bayg29	NN	RGIBNNM	9530	20s
bayg29	NN	SA	9080	20s
berlin52	NN	SA	8182	50s
berlin52	NN	Custom	8139	64s
berlin52	NN	RGIBNNM	8180	60s
berlin52	NN	Random	8182	76s
bier127	Random	Random,RGIBNNM,Custom		could not converge[1]
bier127	Random	SA	132000	150s
bier127	NN	Random,RGIBNNM,Custom		could not converge
bier127	NN	SA	124100	150s
kroA100	NN	SA	21800	110s
kroA100	Random	Random,RGIBNNM,Custom		could not converge
kroA100	NN	Random,RGIBNNM,Custom		could not converge
kroA100	Random	SA	23900	130s

Table 1.0, Readings taken from running the code in the repository - "https://github.com/avirajBevli/GaTSP"

5 Mutation methods used

Mutation is an important parameter determining the performance of a genetic algorithm. If the rate is too less, we can get stuck at local minimum. On the other hand, too high a mutation rate may lead to non-convergence of the algorithm. This section studies the effect that the used mutation operator and the mutation rate has on the performance of the genetic algorithm. We also introduce out custom mutation operator.

5.1 Random Mutation

In this method, two cities from the order are randomly chosen and swapped. This method upon testing yields poor results

5.2 RGIBNNM mutation

This mutation is a knowledge-based operator designed especially for the TSP problem. This operator uses the idea of the nearest neighbor cities, where this mutation selects a random gene (city), and finds its nearest city, then swap the random city with one of the neighbors of the nearest city.

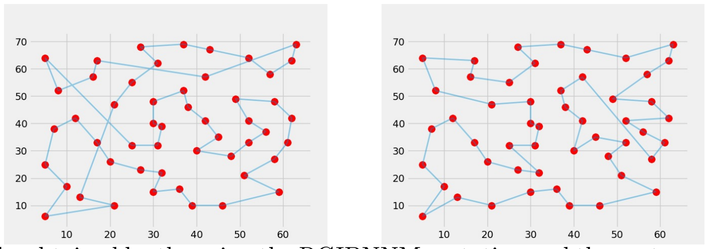
5.3 Proposed mutation operator(Custom)

We propose to use the mutation operator which takes inspiration from the 2-opt algorithm. There has been work previously on using the 2-opt algorithm as a search heuristic for the travelling salesman problem, including [9]. In this mutation, we look for 2 edges such that when switched, lead to a shorter path. If in a path, 2 such edges are found, we check whether a shorter path is being obtained by switching the 2 edges. If yes, we perform the mutation else we leave it. This is done for each individual a fixed number of times in each generation. This way we keep improving the quality of genes over generations. The implementation of this mutation operator involves selecting two random cities from the chain of cities of an individual. Invert the order of all the cities that lie between these 2 cities. If the new order has a shorter path length compared to the original order of cities that means by swapping the two edges, we have obtained a better fitness solution. Hence, we replace the individual by the newly created individual. Else, we do not perform mutation on the individual.

5.4 Simulated annealing inspired mutation

This mutation is a time varying mutation. We accept the mutation if the solution obtained is better than the previous order of the cities. If the solution is worse than the original solution, we choose whether to mutate with a certain probability that decreases with time. This is inspired by the process of another evolutionary algorithm, simulated annealing. We do not use crossover when this mutation operator is used.

5.5 Results and analysis of mutations



Images of the paths obtained by the using the RGIBNNM mutation and the custom mutation operator on the eil15 data set from TSPLIB

Effect of changing mutation rates, using the three mutation operators on the performance and time for a decent solution to be obtained. From the results, we make the following points:
1. For smaller data sets(eg bayg29), it is better to use a bigger population size and crossover with our custom mutation operator. The custom mutation operator and the simulated annealing mutation operator are both viable options.
2. For medium data sets(eg eil51, st70, berlin52), our custom mutation operator outperforms the other mutation operators easily. Using the random, RGIBNNM mutation methods, the solution does not converge to a good solution. Using the simulated annealing approach is a good option and generally yields better results compared to the combination of crossover and custom mutation operator but often takes a longer time to compute the best results.
3. For relatively larger data sets(eg bier127), our custom mutation operator easily outperforms the other mutation operators in terms of the fitness of solutions. But we are not able to obtain convergence after even an hour of letting the algorithm run. This is because crossover seems to be a bad option is general for a problem of this size. The method of crossover does not give give solutions even close to the quality given by the simulated annealing based approach. The simulated annealing approach gives by far the best results in terms of performance but the solution quickly settles into local minima. So an approach discussed in the conclusion is what we look to be working on in future.

6 Initialisation method used

The initialisation method used has a huge impact on the quality of solution and time taken for convergence Here, we compare the following two initialisation methods -

- Random initialisation wherein we randomly form orders of cities.
- The population is not initialised randomly. A certain percentage of the initial population is taken from the results obtained by using the nearest neighbour heuristic. These genes ensure that all the initial genes are not naive. We are able to maintain gene diversity as well as get faster convergence.

6.1 Statistical basis of the Nearest neighbour search heuristic

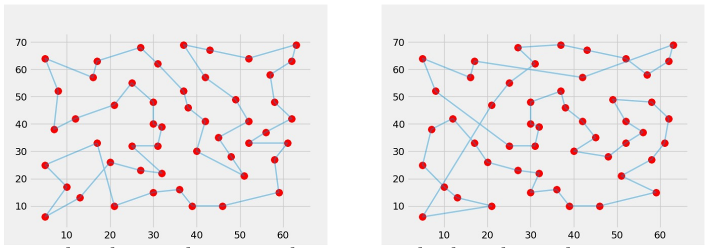
Work on the Nearest Neighbour heuristic for TSP has been going on. [7] and [6] established a tighter upper bound to the path obtained from the NN search heuristic.
We construct an ordering $c_{\pi(1)}, ..., c_{\pi(N)}$ of the cities with $c_{\pi(1)}$ being the initial city and in general $c_{\pi(i+1)}$ chosen to be the city c_k that minimises $d(c_{\pi(i)}, c_k) : k \neq \pi(j), 1 \leq j \leq i$. The corresponding tour traverses the cities in the constructed order, return to $c_{\pi(1)}$ after visiting $c_{\pi(N)}$. For a given heuristic A and TSP instance l, let A(l) denote length of the tour produced by A and OPT(l) denote the length of an optimal tour. According to [.], assuming P $\neq NP, thereexistssome\epsilon >$ 0 such that no polynomial-time TSP can guarantee

$$A(l)/OP(l) \leq 1 + \epsilon$$

With NN initialisation we can provide quite strong upper bound on the performance in $\Theta(\mathrm{N}^2)$ complexity. In particular we guarantee

$$NN(l)/OP(l) \leq (0.5)(\lceil log_2 N \rceil + 1)$$

6.2 Results and analysis of the used initialisation methods



Images of the paths given by the random initialisation method with random mutation and the NN initialisation method with random mutation operator on the eil15 data set from TSPLIB

From the results, we make the following points:
1. For smaller data sets(eg bayg29), the population initialisation method does not affect the results as much. Even if population is initialised randomly, since the search space is not as large, the genetic algorithm is able to arrive quickly at solutions comparable in fitness to those given by the nearest neighbour search heuristic.

2. For medium data sets(eg eil51, st70), initialisation using nearest neighbour search heuristic yields us with fitness values equivalent to those obtained by running the GA for some time. However, after the initialisation, the solution is not able to improve much because of the lack of diversity. On the other hand, the GA approach requires some time to give similar fitness solutions but if given some more time, may outperform the NN initialised method.

3. For relatively larger data sets(eg bier127) it is useful to initialise the population with the nearest neighbour construction heuristic to save time because the simple genetic algorithm might take a very long time to come up with solutions of the same fitness as those given by NN. In fact, in our implementation, if the population was initialised randomly, the solution took more than an hour to converge. Even after convergence the solutions were not as good as those given by the nearest neighbour construction heuristic.

7 Conclusion

While there are various available options of implementation, there is no single best algorithm for all the cases. This is generally the case with heuristic and meta-heuristic methods. From the statistical analysis of the NN search heuristic, we see that the path given by the NN search heuristic is guaranteed to give a path with a path length less than a certain upper bound ratio of the global best path length. As a rule of thumb, for a large number of cities, application of the genetic algorithm after population initialisation by another heuristic method(nearest neighbour search in our case) is a good option because of the huge time a simple genetic algorithm might take. This project is no where near complete and there is work we would look forward to doing after this as well. From this study one thing that seems to be pretty clear is that for the travelling salesman problem, simulated annealing seems to be a more promising option compared to the simple genetic algorithm approach. The only problem is it gets stuck in local minima and not be able to explore better potential solutions. In future, we would like to work on a simulated approach to TSP, where as soon as we hit a local minima, the temperature is increased for another round of simulate annealing and we continue to do this for a number of times and store the best candidate solutions from each round of the simulated annealing process. We wanted to try this approach but were not able to test it because of the lack of time.

The link to the source code written by us for the entire project is -
https://github.com/avirajBevli/GaTSP

References

[1] Otman Abdoun, Jaafar Abouchabaka, and Chakir Tajani. Analyzing the performance of mutation operators to solve the travelling salesman problem. *arXiv preprint arXiv:1203.3099*, 2012.

[2] Murat Albayrak and Novruz Allahverdi. Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. *Expert Systems with Applications*, 38(3):1313–1320, 2011.

[3] Esra' Alkafaween and Ahmad Hassanat. Improving tsp solutions using ga with a new hybrid mutation based on knowledge and randomness. *arXiv preprint arXiv:1801.07233*, 2018.

[4] Hock Hung Chieng and Noorhaniza Wahid. A performance comparison of genetic algorithm's mutation operators in n-cities open loop travelling salesman problem. In *Recent Advances on Soft Computing and Data Mining*, pages 89–97. Springer, 2014.

[5] Absalom El-Shamir Ezugwu, Aderemi Oluyinka Adewumi, and Marc Eduard Frîncu. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Systems with Applications*, 77:189–210, 2017.

[6] Stefan Hougardy and Mirko Wilde. On the nearest neighbor rule for the metric traveling salesman problem. *Discrete Applied Mathematics*, 195:101–103, 2015.

[7] Cor AJ Hurkens and Gerhard J Woeginger. On the nearest neighbor rule for the traveling salesman problem. *Operations Research Letters*, 32(1):1–4, 2004.

[8] Abid Hussain, Yousaf Shad Muhammad, M Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry, and Showkat Gani. Genetic algorithm for traveling salesman problem with modified cycle crossover operator. *Computational intelligence and neuroscience*, 2017, 2017.

[9] Kenan Karagul, Erdal Aydemir, and Sezai Tokat. Using 2-opt based evolution strategy for travelling salesman problem. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA)*, 6(2):103–113, 2016.

[10] Anoop Sathyan, Nathan Boone, and Kelly Cohen. Comparison of approximate approaches to solving the travelling salesman problem and its application to uav swarming. *International Journal of Unmanned Systems Engineering (IJUSEng)*, 3:1–16, 01 2015.

[11] Shi-hua Zhan, Juan Lin, Ze-jun Zhang, and Yi-wen Zhong. List-based simulated annealing algorithm for traveling salesman problem. *Computational intelligence and neuroscience*, 2016, 2016.