Asn 1: The Salesperson that Math Can't Follow

Ethan Heavey
St. Francis Xavier University
Department of Computer Science
Antigonish, Nova Scotia, Canada
x2018zbf@stfx.ca

***Abstract*—The Traveling Salesperson Problem is a rigorously studied problem with a great deal of data and research surround it. There is a multitude of ways to solve the problem, either by specifically tailored algorithms for the Problem[3]. It is possible to do by hand, however this approach is not recommended. Enter genetic algorithms. In this report, it will be shown how a genetic algorithm can be implemented to solve a wide variety of Traveling Salesperson Problem instances, along with the results achieved. Modifications have been added to the basic implementation in order to provide further insight into the inner workings of a genetic algorithm, and how it searches a state space for an optimal solution. By tweaking parameters and attempting unique methods of mutation, the distance of the best solution achieved by this genetic algorithm will come very close to achieving the optimal solution to a Problem instance.**

***Index Terms*—Evolutionary Computation; Traveling Salesperson Problem; Genetic Algorithm;**

I. Introduction

The Traveling Salesperson Problem is a well known problem, but once the number of cities becomes sufficiently large, attempting to solve the problem by hand becomes tedious at best, impossible at worst. Using genetic algorithms,this tedious process becomes child's play, as one can press "Run", sit back, relax, and enjoy the lovely graphs the algorithm will generate for you as you plan your holiday.

II. Data

The data used for this problem was taken from Florida State University (FSU) [1]. The text files are available for download at the FSU Department of Scientific Computing website, with each one containing an adjacency matrix for a given number of cities. The text files ATT48, DANTZIG42, FIVE, and FRI26 were all used in the initialization and testing of the TSP model.

A. The Problem

The Traveling Salesperson Problem (TSP) is a routing and scheduling problem, which calls for a minimum cost Hamiltonian cycle on a given graph[2]. Typically referred to as "cities", the vertices on a given graph will have a set value for all edges, creating a network of cities the salesperson can travel through.

B. Data files and contents

The FSU data files, as previously mentioned, each contain an adjacency matrix pertaining to the distance between some number of cities. Formatted as text files, each row of a given file represents one city, with the integer values in the row representing the distance to all other cities. Being an adjacency matrix, a city's distance to itself is always represented as 0.

III. Algorithm

A GA was implemented to solve TSP, which used two different mutation methods, inversion and swap, to promote diversity within the population and each new generation.

A. Parameters

The genetic algorithm has a multitude of parameters that can be tweaked and updated. All parameters are significant, and fine-tuning a single parameter will be a tedious task, as once the value of one parameter changes, the optimal value of all other parameters shifts to create a new optimal permutation of values. The one parameter that has minor leeway regarding this permutation change is the *MAX_GENERATIONS* parameter. Once this parameter is set to an sufficiently large value, and the genetic algorithm has converged on a point in the problems state space, this parameter has minimal effect. However, too small of a value for *MAX_GENERATIONS* could result in premature convergence, or no convergence at all. Due to hardware limitations this value has been set to s static 1000, as a higher value takes too long to compute and crashes the machine.

Another parameter of significant importance to the functionality of this GA is the value of *SELECTION_SIZE*. This parameter, as the name suggests, represents the number of chromosomes that will be selected from Generation*X*, and, after some mutation or crossover, will be added to Generation*X+1*. The chance of mutation and/or crossover occurring is determined by the *MUTATION_RATE* (*MR*) and *CROSSOVER_RATE* (*CR*) parameters respectively.

B. Structure

The GA was designed with the structure of the chromosomes in mind. Using nested lists to represent the population, each chromosome represented a different permutation of the cities from the adjacency matrix provided. After initialization of the population and fitness lists, a preset number of random

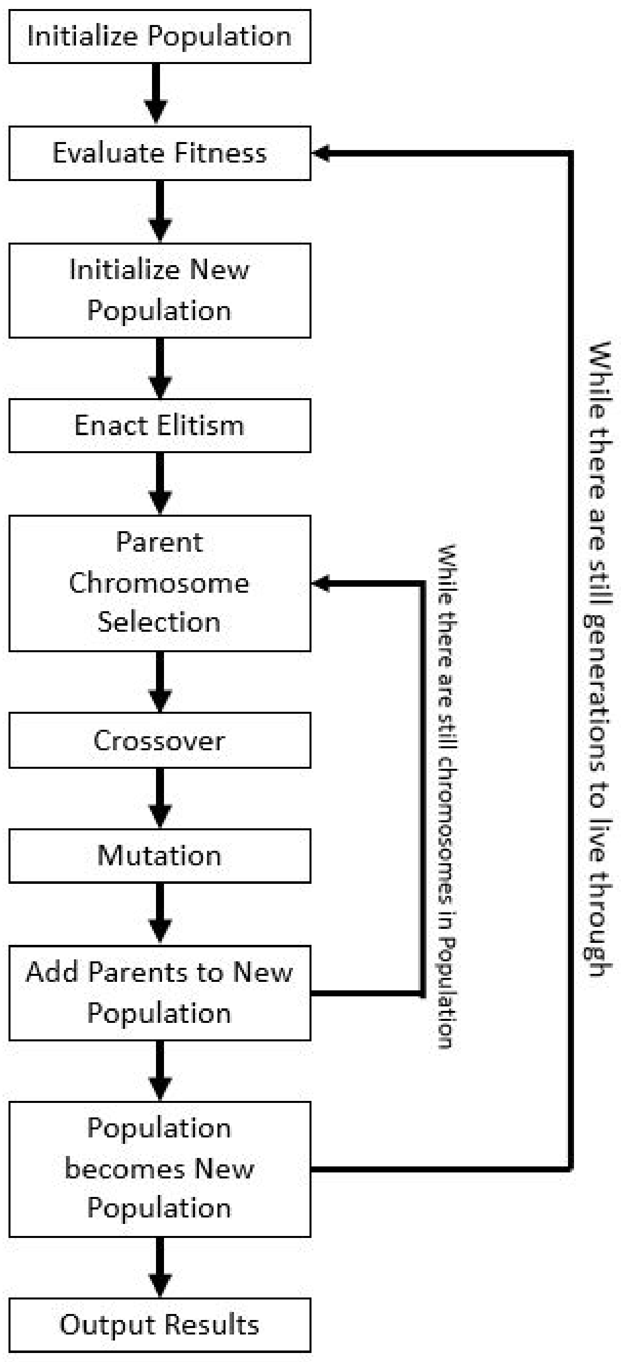
Initialize Population
Evaluate Fitness
Initialize New Population
Enact Elitism
Parent Chromosome Selection
Crossover
Mutation
Add Parents to New Population
Population becomes New Population
Output Results
While there are still chromosomes in Population
While there are still generations to live through

Fig. 1: The high-level structure of the GA

chromosomes would be selected to use for the next generation. As mentioned in III-A, there are a multitude of parameters and variables that affect the performance of the GA. However, these parameters have no effect on the inherent structure of the GA.

C. Crossover

There are a multitude of ways to perform a permutation crossover [4], however the one implemented in this GA was a simple one-point crossover function. By performing a deep copy of parent chromosomes, and selecting a random index, the new chromosomes that are produced preserve their permutation properties and the parents ”breed” to create two new child chromosomes.

D. Modifications

A NUMBER of modifications were added to the GA, their descriptions and affect on the results solving TSP are described below. In all instances, the parameters of the GA were set to the values in Table I

TABLE I: This table depicts the chosen default values for the listed parameters.

MR	*CR*	Max Generations
0.1	0.8	1000

1) Elitism: Elitism is the act of selecting the best of any given generation and ensuring those chromosomes are passed onto the next generation. The danger of elitism is how one can get caught in a local minima by selecting too many of the top chromosomes, and not allowing for crossover and mutation of inferior chromosomes. Crossing chromosomes linked to a high fitness value with those associated with a low fitness value could lead to new patterns and routes in the search space that would be undiscoverable if only the best of each generation are considered. Due to the risk associated with becoming trapped in a local minima, only a single chromosome, with the highest fitness value, would be selected... if we were constructing the best GA for TSP. Time to play with that number and see what happens.

The figures below are the TSP graphs from the first, middle, and final generations of the following parameter values.

TABLE II: This table displays the default values for elitism tests.

Figure	A	B	C
Num. Elite Chromosomes	1	4	10
MR	0.1	0.1	0.1
CR	0.8	0.8	0.8
Max Generations	1000	1000	1000

As one can see from Fig. 2 through Fig. 4, even though Fig. 4 selects the 10 best chromosomes out of the population each generation, it appears to be stuck in a local minima. Fig. 2 and Fig. 3 have more widespread edges connecting their vertices, leaving room for crossover and mutation to have a last impact on the up-and-coming generation.

2) Mutations: Two different mutations were implemented and tested extensively within the GA. The methods tested were random swap and inversion. Random swap was implemented to randomly select two different indices of a given chromosome, and swap the values of them. This can cause disruption, which can either help or hinder the development

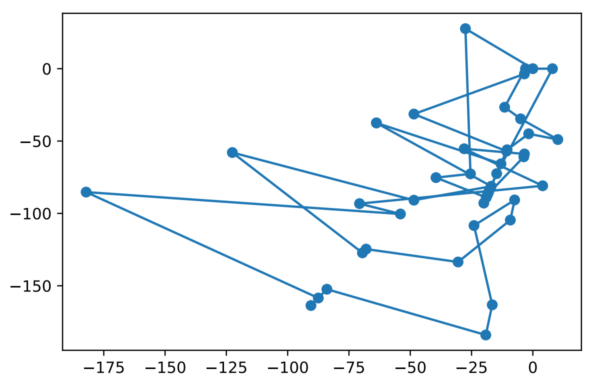

Fig. 2: Test A - After 5000 Generations

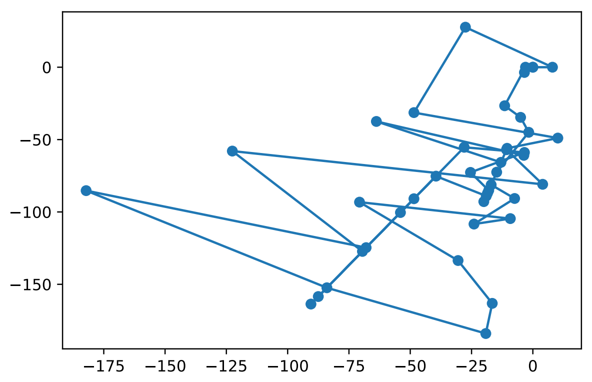

Fig. 3: Test B - After 5000 Generations

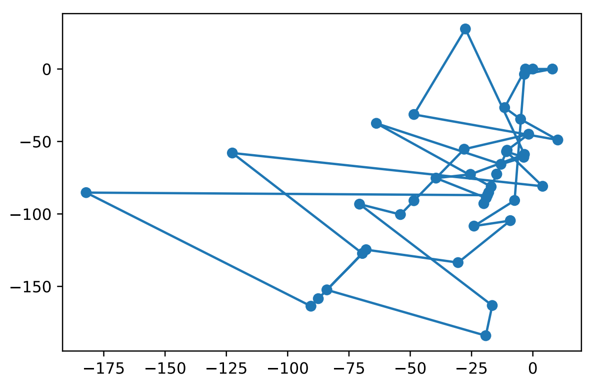

Fig. 4: Test C - After 5000 Generations

of a population, as disruption can help the GA move out of a local minima, but it can also disrupt a fit permutation and thus hinder the progress of the GA.

Inversion however, creates minor disruptions in the per-

TABLE III: This table displays the results of a multitude of tests for analysis.

Cities	No. Elite	Max Gen	*MR*	*CR*	Best Solution	Result
26	0	1000	0.1	0.8	937	1098.06
26	0	1000	0.3	0.8	937	1018.0
26	1	1000	0.1	0.8	937	984.4
26	1	1000	0.3	0.8	937	975.9
26	4	1000	0.1	0.8	937	967.5
26	4	1000	0.3	0.8	937	964.2
42	0	1000	0.1	0.8	699	983.1
42	0	1000	0.3	0.8	699	962.4
42	1	1000	0.1	0.8	699	1005.7
42	1	1000	0.3	0.8	699	1027.4
42	4	1000	0.1	0.8	699	1267.2
42	4	1000	0.3	0.8	699	943.9

mutation. Inversion selects two random indices of a given chromosome, and reverses the order of the genes between those indices, preserving a lot of the order of the pre-mutated chromosome. A drawback of this minimized disruption is that the population could trend towards a local minima, and would have difficulty leaving the local minima without a radical change in population.

IV. Analysis Methodology

The results of the GA were compared to the best solution of the given adjacency matrix, and also so subsequent tests on the same matrix with different parameters. The results found in table III are the average of the best value from thirty executions. Other comparisons that were performed were T-Tests, and box plots, in an attempt to truly understand which iteration of the GA was more successful in determining the certain point.

V. Results

As can be seen in table III, the closest the GA comes to the optimal solution is in the instance represented by row six, with an average result of 964.2, a maximum value of 989, a range of 46, and a minimum value of 943.

A. Statistics

In testing this implementation, scipy.stats T-Tests were used to evaluate the statistical significance of the model. Multiple tests were conducted, between random and inversion mutation methods, and between each mutation method with and without elitism. As shown in IV, the Random and Inversion distributions are statistically significant, and, with a p-value significantly smaller than 0, the null hypothesis is rejected. The Random v Elitism Random and Inversion v Elitism Inversion results were equally interesting. With such large t-scores and p-values, the comparison between a mutation method and the same method with elitism (MvEM) implemented clearly are statistically significant. With the null hypothesis rejected twice more, one can see that the t-score of the MvEM methods is very large, with just over 30 standard deviations from the mean in one case.

TABLE IV: This table displays the results statistical tests.

Comparison	T-Score	P-Value
Random v Inversion	-4.1788	0.00056
Random v Elitism Random	30.29196	6.7561 e-17
Inversion v Elitism Inversion	29.56718	1.03625 e-16
Elitism Random v Elitism Inversion	3.2519	0.0044

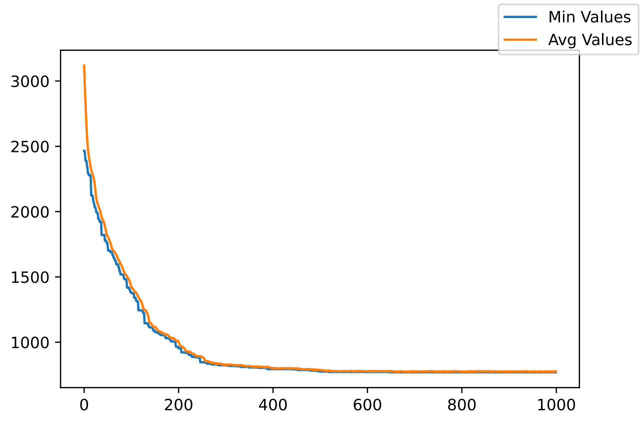

Fig. 5: Minimum and Average plot for a 42 City Instance

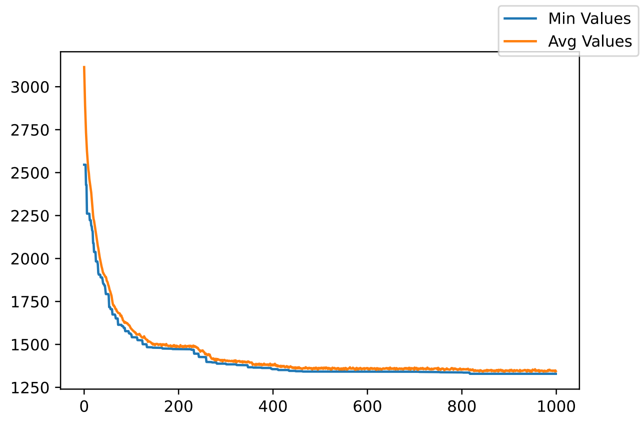

Fig. 6: Minimum and Average plot for a 26 City Instance

B. Averages, Minimums, and Paths

As can be seen in Fig. 5 and Fig. 6, after a 5000 generation cycle, the GA yields a much cleaner solution to TSP. As Fig. 5 represents the most fit of the initial permutations, Fig. 6 boasts the natural selection prowess only a permutation 5000 generations in the making could. One can see how the GA can completely pivot from the initial permutation, yielding a graph where the only similar characteristics are the placements of the vertices. Fig. 7 is my favourite solution for a TSP instance of 26 cities as it looks akin to a (slightly alien) dog, and I love dogs.

Regarding Fig. 8 and Fig. 9, I found two things interesting. 1) How fast the 42 city instance converged, and 2) the little plateau from generation 150 to generation 225 in the 26 city instance. I believe that elitism played a crucial role in allowing

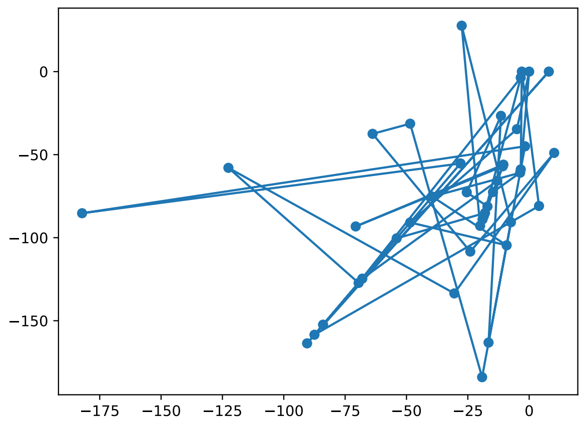

Fig. 7: After Initialization of a TSP Instance of 42 cities

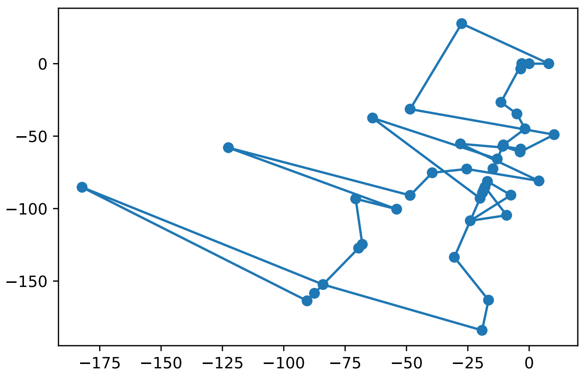

Fig. 8: Final Result of 5000 Generations on a TSP Instance of 42 cities

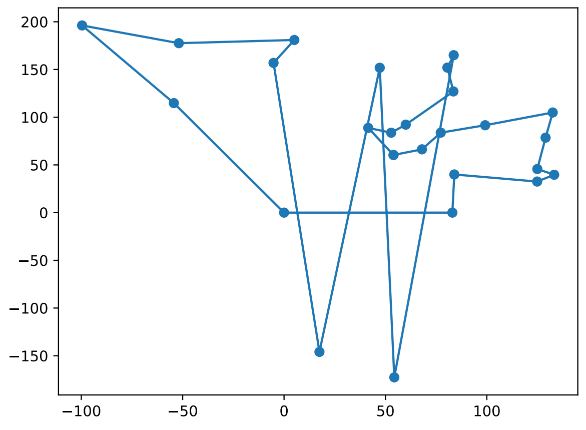

Fig. 9: Final Result of 5000 Generations on a TSP Instance of 26 cities

the 42 city instance to converge on an optimal solution within a quarter of the total allotted generations, whereas it appears the 26 city instance of Fig. 9 got trapped in a local minima and was fortunate enough to escape.

VI. Conclusions and Future Work

Regarding future work, this GA can be adapted to solve instances of the Vehicle Routing Problem (VRP), which is a generalized version of TSP. Having tweaked and adapted the GA, unfortunately there were issues and minor bugs preventing the problem from being solved. In particular, a tournament selection method would be added, where chromosomes must minimize both time spent traveling and the distance they travel. Other unique crossover methods could be added on top of the change in selection methods, namely multi-point crossover and cycle crossover. Moving back to TSP, future work would involve testing a multitude of different selection, crossover, and mutation methods, many of which were attempted but weren't debugged in sufficient time to provide insight into their effect on the results.

All in all, this GA provided some interesting solutions that came surprisingly close to the best possible solution. One just has to be careful it doesn't get caught in a local minima.

References

[1] Tsp data for the traveling salesperson problem. https://people.sc.fsu.edu/~jburkardt/datasets/tsp/tsp.html, July 2019.

[2] Gregory Gutin and Abraham P Punnen. *The traveling salesman problem and its variations*, volume 12. Springer Science & Business Media, 2006.

[3] Jakub Marecek. The traveling salesman problem: a computational study. *Interfaces*, 38(4):344, 2008.

[4] IM Oliver, DJd Smith, and John RC Holland. Study of permutation crossover operators on the traveling salesman problem. In *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*. Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.