**Traveling Salesman Problem: Wisdom of Crowds Using Genetic Algorithms**

Jacob Taylor Cassady

Computer Engineering & Computer Science

Speed School of Engineering

University of Louisville, USA

[jtcass01@louisville.edu](mailto:NameID@louisville.edu)

# Introduction

The Traveling Salesman Problem (TSP) is a well-known non-deterministic polynomial-time hard problem that has been studied within mathematics since the 1930s. The "salesman” is given a list of cities with their locations and is asked the shortest route to travel to each city once and then return to the starting point. A program was developed using Python 3.7 and accompanying 3rd party libraries: NumPy, Pandas, and matplotlib to determine the shortest path.

# Approach

The approach taken to solving the TSP was to use the ‘wisdom of crowds’ principle along with a series of differing genetic algorithms. A population of 180 chromosomes were split evenly between 6 genetic algorithm variations. The chromosomes were represented as a series of alleles. Throughout this paper, alleles and vertices will be used interchangeably. Edges will be used to refer to adjacent pairs of alleles.

The genetic algorithm variations were generated from cross joining the mutation methods and crossover methods described in sections 2.1.1 and 2.1.2 of this document. The genetic algorithms were run until their population had 25 consecutive generations without improvement. A generation was defined as a series of both performing cross over and then performing mutation. A crowd was then generated by taking a percentage of each algorithm’s population with best performance.

To develop an aggregate answer from the crowd of chromosomes, the crowd was examined to determine common consecutive pairs of alleles by calculating the relative frequency of each edges. A threshold was then used to determine the required number of recurrences of the same edge required throughout the crowd for the edge to be included in the aggregate answer. If two edges were found frequently and included and same starting vertex or the same ending vertex, the edge with the smallest distance was kept and the edge with the longer distance was discarded. If the resultant graph was not complete, a greedy heuristic was used to select the nearest unvisited vertex from an edge already included. The greedy heuristic algorithm is explained in more detail in section 2.3 of this document.

## Genetic Algorithm

The genetic algorithm implemented is inspired by sexual reproduction of gametes in biology. This algorithm retains a constant population of “chromosomes” which are representations of possible solutions/agents for/within the given problem. These chromosomes are a set of alleles that describe its performance. The algorithm makes use of two functions to evolve the population overtime to weed out the poor performers and mate the good performers.

### Crossover Methods

The implemented crossover methods have been shown to improve performance in a genetic algorithmic approach to TSP (ABDOUN & ABOUCHABAKA, 2011). Each algorithm was run with an 80% crossover probability, meaning that with each generation the top 20% of the population was used to generate replacements for the bottom 80%.

#### Uniform

The uniform crossover forms a child by randomly alternating between the two parents. For reference to the implementation of this method please see **Figure x** in the appendix.

#### Ordered Crossover

The ordered crossover breaks each parent into three sequences, S1, S2, and S3 with matching indices for both parents. The child is then produced by taking S2 from one parent and filling S1 and S3 with alleles from the other parents starting at S1 and leaping genes already included. For reference to the algorithm as formalized in the literature, please refer to **Figure x** below. For refence to the implementation of this method please see **Figure x** in the appendix.

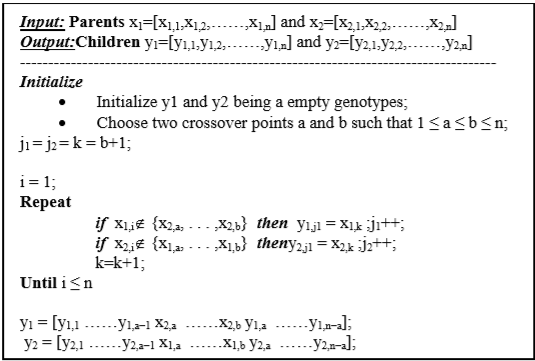


Figure : Ordered Crossover Algorithm (ABDOUN & ABOUCHABAKA, 2011)

#### Partially Mapped

The partially mapped crossover breaks each parent into three sequences, S1, S2, and S3 with matching indices for both parents. The child is then produced by taking S1 and S3 from one parent and filling in S2 with alleles from the other parent starting at S2 and leaping genes already included. For reference to the algorithm as formalized in the literature, please refer to **Figure x**. For reference to the implementation of this method please see **Figure x** in the appendix.

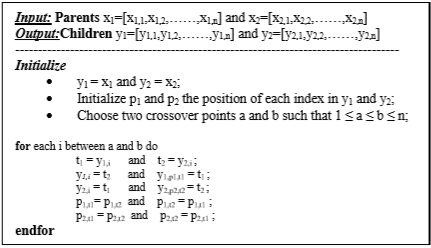


Figure : Partially Mapped Algorithm (ABDOUN & ABOUCHABAKA, 2011)

### Mutation Methods

The implemented mutation methods have been shown to improve performance in a genetic algorithmic approach to solving TSP (ABDOUN & ABOUCHABAKA, 2011). All genetic algorithms were run with a mutation rate of 2%, meaning that each generation had 2% of its chromosomes undergo mutation.

#### TWORS

The TWORS mutation method randomly swaps two alleles’ locations within the chromosome. For reference to the implementation of this method, please see **Figure x** in the appendix of this document.

#### Reverse Sequence

The Reverse Sequence mutation method reverses the sequence of the chromosome. For reference to the implementation of this method, please see **Figure x** in the appendix of this document.

## GUI

A GUI was developed to visualize different stages of the algorithm. A heat map was developed to understand the crowd’s edge frequency. Additionally, a route solution representation was generated to ensure proper connection of the final path.

### Heat Map

Heat maps were generated to help understand the crowd’s edge frequency. The edges were plotted with their RGB values denoting its frequency within the crowd. The most red colored edges are those that occur least frequent, while the most blue colored edges are those that occur most frequent. This was used to estimate an optimal frequency requirement of 65%, meaning

### Route Solution

A graphical representation of the final solution was generated to ensure it is reasonable. Each edge is colored using its vertices’ IDs to quantize its red and blue color magnitudes while the green magnitude is calculated from the modulus of the starting vertex id with respect to the ending vertex id. For reference to the implementation of this plotting method, please refer to **Figure x** in the appendix.

A picture containing indoor, object, sky

Description automatically generated

## Greedy Heuristic

# Results

## Data

## Results

# Discussion

# References

ABDOUN, O., & ABOUCHABAKA, J. (2011, October). A Comparative Study of Adaptive Crossover Operators for Genetic Algorithms to Resolve the Traveling Salesman Problem. *International Journal of Computer Applications, 31*(11). Retrieved from https://arxiv.org/ftp/arxiv/papers/1203/1203.3097.pdf

Baraglia, R., Hidalgo, J. I., & Perego, R. (2001, December). A Hybrid Heuristic for the Traveling Salesman Problem. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, 5*(6), 613-622. doi:10.1109/4235.974843

Yi , S. M., Steyvers , M., Lee, M. D., & Dry , M. J. (2011). Wisdom of the Crowds in Traveling Salesman Problems.

Wikipedia, Traveling Salesman Problem - <https://en.wikipedia.org/wiki/Travelling_salesman_problem#History>

NumPy Documentation - <https://docs.scipy.org/doc/>

Pandas Documentation - <https://pandas.pydata.org/pandas-docs/stable/>

Matplotlib Documentation - <https://matplotlib.org/3.1.1/contents.html>

# Appendix

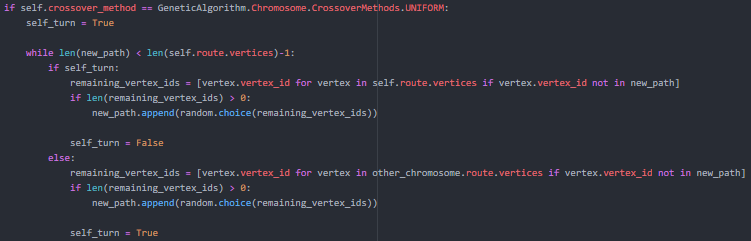


Figure : Uniform Crossover Method

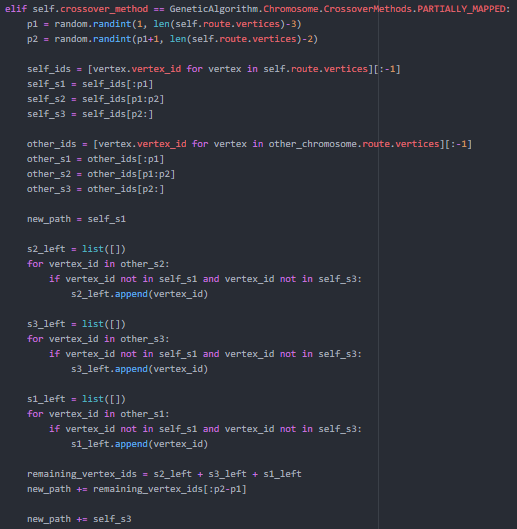


Figure : Partially Mapped Crossover Method

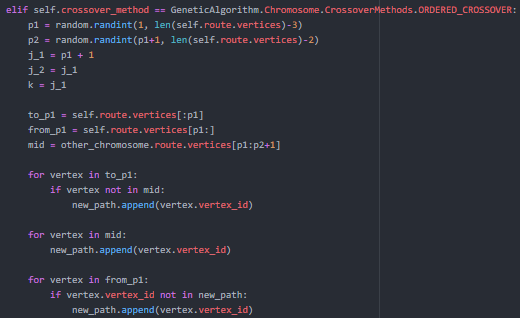


Figure : Ordered Crossover Method

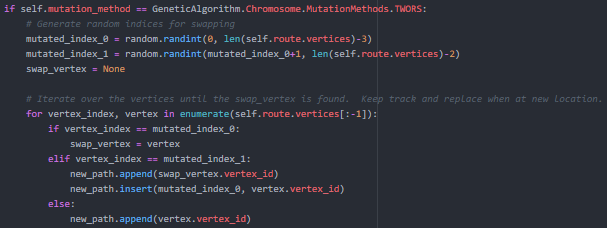


Figure : TWORS Mutation Method



Figure : Reverse Sequence Mutation Method

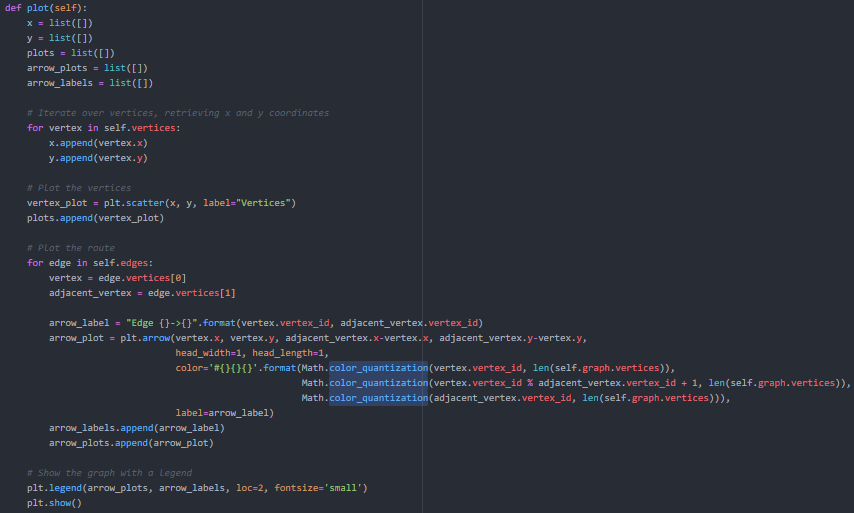


Figure : Route Solution Plotting Method