**Traveling Salesman Problem: Genetic Algorithm**

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 a genetic algorithm. Development of the algorithm was aided by a graphical user interface (GUI). Throughout this document cities from TSP will be referred to as “vertices” and the route between a pair of vertices as an “edge.”

## 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. The algorithm then makes use of two functions to evolve the population overtime to weed out the poor performers and mate the good performers. The mating process is called “crossover”. An additional method is used after cross over to add more randomization into the process, this method is called “mutation”. For the high-level implementation of the algorithm please refer to **Figure ?** in the appendices.

### Crossover

d

### Mutation

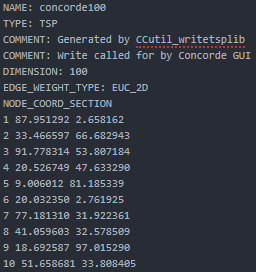
d

# Results

The genetic algorithm was successfully implemented along with its functions: crossover and mutate. The algorithm produces a route that visits each city; although, the output is not necessarily the shortest route. The algorithm did not have any issues running with only 4 GB of RAM. No mitigation techniques were needed to reduce program memory usage or improve runtime efficiency.

## Data

The algorithm was tested on one dataset of 100 randomly generated cities. Within the test file, cities are enumerated, and x and y coordinates are provided. The input data was formatted like the example shown in **Figure ?** below. Additionally, a graphical representation of the graph of cities is shown in **Figure ?**.



A close up of a map

Description automatically generated

## Results

T

# Discussion

# References

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

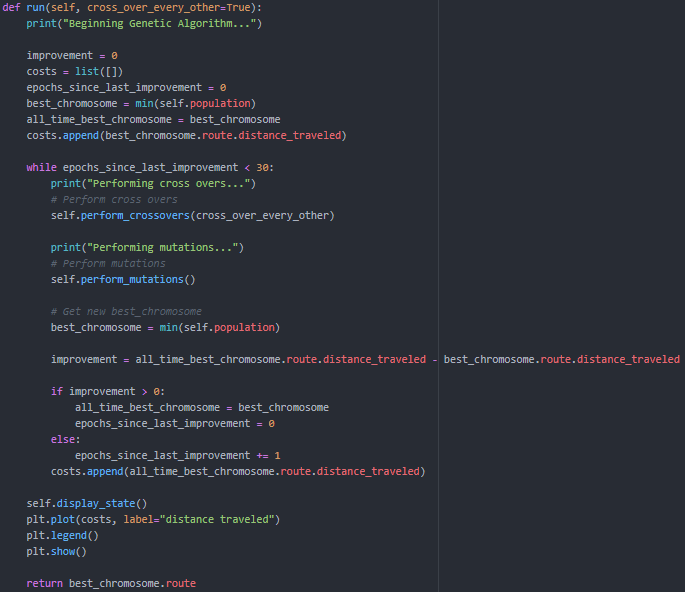


Figure : Genetic Algorithm High Level Function

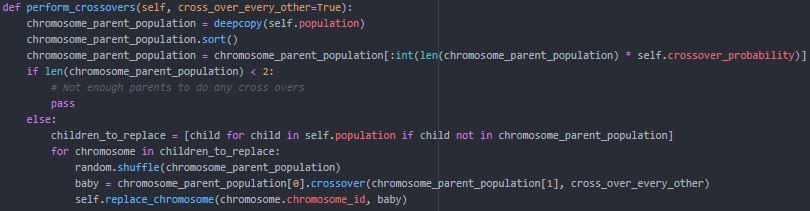


Figure : Perform Crossover Method (From perspective of population)