11/19/2024

MET CS 767 Assignment 5: Genetic Algorithms

*replace this with your name*

This is an exercise in evolutionary learning and representation. You are to implement the *traveling salesman problem* using a genetic algorithm in the manner outlined below—preferably in Python, otherwise Java. We’ll call this project TravOrder because it has a unique way to represent routes. You can assume that there is a direct route connecting every pair of cities, and that no two distances are equal[[1]](#footnote-1).

Eric Braude created this technique to perform seamless crossover. It may be published elsewhere (although we have not located any such reference), and if so, please do not use such work for this assignment. You are encouraged to leverage AI generation as usual, however. The instructions abut red font and attached AI transcript are the same as in the previous assignments.

# Representation of the Data

## Read part 3 which describes the crossover you must use. Describe how TravOrder will *represent* the given data (e.g., “a list consisting of …”). As an example, describe how the following given data are to be represented using your representation:

## B(oston) to L(ondon) 3(k miles),

## L to M(umbai) 4.5,

## M to S(hanghai) 3.1,

## S to L 5.7,

## B to M 7.6, and

## B to S 7.8.

The above is only an example: your assignment should apply to any traveling salesman problem with unique node distances. You can refer to the nodes as “cities” if you wish.

your response replaces this

# Representation of a Route

## Explain how TravOrder will represent a route. Include, as an example based on the above example data, how the following route will be represented: *B to L to M to S to B*.

your response replaces this

# Crossover

## Provide a crossover function consistent with the following.

## TravOrder should create a child from two parents by simple cuts, as in the example below. (Bold and italics are added to clarify what part of the child comes from what parent.)

## Parent 1: Boston 🡪 2nd closest unvisited city[[2]](#footnote-2) 🡪 2nd closest unvisited city[[3]](#footnote-3) 🡪 closest unvisited city 🡪 Boston

## *Parent 2: Boston 🡪 closest unvisited city 🡪 2nd closest unvisited city 🡪 closest unvisited city 🡪 Boston*

## Child route from these parents—with crossover point at 1:

## Boston 🡪 2nd closest unvisited city *🡪 2nd closest unvisited city 🡪 closest unvisited city 🡪 Boston*

your response replaces this

# Mutation

## Explain (clearly) how your TravOrder performs mutation.

your response replaces this

# Result on the Given Data

## Describe the result from executing your GA on the following example data:

## *B(oston) to L(ondon) 3(k miles), L to M(umbai) 4.5, M to S(hanghai) 3.1, S to L 5.7, B to M 7.6, and B to S 7.8*.

## What do you think of this result? Explain.

your response replaces this

# Result on Your Data

## Describe the result from executing your application on illustrative data of your choice. What do you think of the result? Explain.

your response replaces this

# Key Code

## Paste below the most important code snippets.

your response replaces this

# Source Code

## Paste your source code below—or refer to an appendix. It should accompany this doc as well.

your response replaces this

# Comments on Performance

## Compare the performance of TravOrder with at least one known Traveling Salesman GA implementation.

your response replaces this

# References

[1] your first reference replaces this

[2] …

# Evaluation



# Appendix 1

# Appendix 2

1. The latter simplifies coding, as you will see. [↑](#footnote-ref-1)
2. i.e., compared to all other direct hops from the city just visited (Boston in this case) [↑](#footnote-ref-2)
3. i.e., compared to all other direct hops from the city just visited [↑](#footnote-ref-3)