Genetic Algorithm Solution for Travelling Salesman Problem
COMS3101 section 2

Bernardo Abreu (bd2440)
Felipe Rocha (flt2107)

1. Overview

This is a simulation GUI that allows the user to insert an unlimited number of cities coordinates, which will be shown on a "map", and then see the solution for the Travelling Salesman Problem (called TSP from this point) given by a Genetic Algorithm (called GA from this point). The user can also change the algorithm parameters (described below) to check how the algorithm will (not) converge given a number of iterations.

Given a map containing 13 cities (as the example available on the project), an exhaustive algorithm written in Python would take several hours to check all possibilities and find the best one, while the GA implemented on this project is able to reach same result within 15 seconds.

2. Definitions for GAs

Genetic Algorithm is a search heuristic used to solve optimization problems. By encoding a defined number of possible solutions as individuals (Population parameter), a generation is formed. Then, by crossing individuals and doing mutation on each iteration, it forms new generations. By the end of the iterations, the best individual of the last generation is picked up as the solution.

To improve the algorithm's convergence, some extra parameters are available:

- Mutation Ratio: controls how often gene mutations happen;
- Elitism Ratio: defines the portion of the previous generation (sorted from best to worst individual) that will be kept the same for the following generation (without crossing and/or mutation);
- Crossover Slice: defines the portion of the previous generation (sorted from best to worst individual) that will be used as parents for the following generation individuals.

3. Usage

The program can be run with the command "python main.py" on the terminal.

The interface presents a map of the cities to be covered by the GA that is scaled to better fit the size of the screen. It also presents a menu on the right side of the screen with the functionalities of the software. It is possible to add cities through the button "Add city", which will open a window to enter the name of the city and its coordinates. It is also possible to remove the city with the "Remove city" button. Clicking this button will open a window with a list of the cities in the map, which can be selected and removed. The starting city is the first city to be added to the map, but it can be changed using the "Change start" button. It is also possible to change the parameters with which the GA will run. The program also presents an example map to demonstrate its operation, which can be loaded using the "Load example" button.

4. Architecture

This solution uses the Tkinter graphical library, available on the Python 3.4 Anaconda Distribution. The source code contains the following files:

- main.py: main file, which binds engine and UI;
- ga.py:
  - class City: class that holds each city information (name and coordinates);
  - class Map: class that holds the list of cities;
  - class Population: class that holds the population and the GA methods;
- gatest.py: script to run and test the GA from command line;
- mapview.py: contains class for the map visualization frame;
- menuview.py: contains class for the visual menu frame;
- newcitypane.py: contains class for the "Add City" window;
- rmcitypane.py: contains class for the "Remove City" window;
- changestartpane.py: contains class for the "Change Start City" window.

5. Conclusion

This project provided an interesting and instructive experience. The python language is very helpful and easy to use, and it facilitates the employment and learning of resources that the programmer might have never used before. The work done on this project allowed for a better understanding and dominium of Python, increasing the familiarity and with language, besides showing its usefulness.

6. Screenshot