AI - Assignment 5

Patrick Nagel: patrick.nagel@h-brs.de
Amirhossein Pakdaman: amirhossein.pakdaman@smail.inf.h-brs.de

Due Date: 01.12.2019

0.1 Programming Assignment: The Traveling Salesman Problem

- Inside src/data you will find two input files (cities throughout the world along with their locations) for this weeks programming assignment.
- Solve the Traveling-Salesperson problem by implementing random-restart hill-climbing. You can assume **Cartesian distances** between the cities.
- You can use the 49_cities.txt to test your program. Submit the final results on all cities (cities_full.txt). Check travis.yml for command line arguments to include txt files while running your python code.
- Russel and Norvig describe the steepest-ascent/descent version of hill climbing: Given a state, all possible sucessors are being evaluated and the one with the highest/lowest objective function is chosen as the new state. The simple hill climbing variant is: Start evaluating the sucessors of your current state and as soon as any better one is found, take is as the new state and repeat. (For more information see variants here: `https://en.wikipedia.org/wiki/Hill_climbing`).
- The goal of your agent is to find a cycle (a roundtrip) which **visits every city once**, while traveling the minimal possible distance. For more information see: `https://en.wikipedia.org/wiki/Travelling_salesman_problem`.
- Do random-restart hill-climbing atleast 5 times in both your implementations and for every 2000 iterations. This means that you restart each hill-climbing algorithm after every 2000 iterations and run until 10000 iteration to report the best result.
- Generate plots of results for with iterations from 1 to 10000.

0.2 Implementation Details

- Use *get_successors* to generate **100 random successors** by swapping **any two cities** (except for the first and last)
- Use *get_distance* to calculate the total roundtrip distance. This can be done easily by consecutively adding the distances between each of the two subsequent cities in roundtrip path.
- Efficient inference of distances between two cities can be obtained by storing the distances between all cities in a distance matrix D using *get_distance_matrix*. The number of rows and columns of distance matrix are equal to the number of the cities and each entry D_{ij} represents the distance between cities i and j. Note the diagonal elements D_{ii} will be 0 because the distance between a city and itself is 0.
- Using the generated successors from *get_successors* and the roundtrip distance from *get_distance*, the different version of the hill-climbing algorithm should be implemented in *hill_climb_simple* and *hill_climb_steepest_descent*.
- Use *plot_cost_function* to generate the plots (Hint: Use matplotlib for plotting). Look at the sample plot in results/plots to get an idea of how the plot should look like.

- Use 49_cities.txt during initial development, testing and experimentation of your algorithm. The hill-climbing search may take quite some time to run based on your implementation. Final results should be available for both txt files. Refer to travis.yml to switch between files using command line arguments.
- Please **do not change your first and last city** during generation of successors or random restart, because the salesman always starts and ends in the same city throughout the entire search (it is a roundtrip!).
- Please add comments explaining your code as and when required. Also fill in the docstrings for the implemented functions.
- You may add additional functions if required and modify the parameters of the existing functions.
- Any extra functions required as per your implementation should be included in helper.py. Please add the necessary docstrings and keep your code as clean as possible.
- Please refer to the travis.yml for command line options to run your code.