Search Based Software Engineering

Exercise 01 - Traveling Salesman Problem

2019-04-23

Deadline: 2019-05-06 23:59

Submit to: andre.karge@uni-weimar.de

Submission details: compress your files (.zip or .tar.gz or .rar)
Include a text file with the names and matrikel numbers for each group member!

Name your compressed file: <lastname>_<firstname>_<matrikelnummer>-ex<exercise-number>(.tar.gz or .rar or .zip)
or for more than one student: please use this format for all group members
example: norris_chuck_123456-schwarzenegger_arnold_121212-ex01.tar.gz

Groups: submit your solved assignment in **groups of 2**

Language: Python 3

Hints: The introduction and problem set of the TSP can be found at: link
All exercises have to be applied to the *city100.txt* problem set!
All algorithms can be found in the lecture slides.

Problem Description
The Traveling Salesman Problem (TSP) is given by the following question:
"*Given is a list of cities and distances between each pair of cities - what is the shortest route that visits each city and returns to the original city?*"[1]
The TSP is an **NP-Hard-Problem** which does not mean an instance of the problem will be hard to solve. It means, there does not exist an algorithm that produces the *best* solution in polynomial time. We can not make predictions about how long it might take to find the *best* solution.
But, we can find a *good* solution which might not be the *best* solution. It is ok to find a route amongst 1000 cities that is only few miles longer than the *best* route. Particularly, if it would take an inordinate amount amount of computing time to get from our *good* solution to the *best* solution.

Use the *city100.txt* file to implement different Single-State Meta-Heuristics. You can also use the functions which were given in the lab class.

[1] source: `https://en.wikipedia.org/wiki/Travelling_salesman_problem`

Task 1. (5 points)
Implement Steepest Ascent Hill Climbing on the TSP.

Task 2. (5 points)
Implement Steepest Ascent Hill Climbing with replacement on the TSP.

Task 3. (5 points)
Implement Simulated Annealing on the TSP.

Task 4. (5 points)
Implement Tabu Search on the TSP.

Task 5. (5 points)
Implement Iterated Local Search (ILS) on the TSP.