Assignment 1: The Traveling Salesperson Problem

Core Body of Knowledge classification (http://tinyurl.com/acscbok): Abstraction (5), Design (5), Teamwork concepts & issues (3), Data (5), Programming (5), Systems development (3)

Due date: Week 4, Weight: 10% of the course

1 Overview

Assignments should be done in groups consisting of five to six (5–6) students. Please use the Groups feature in myUni. If you have problems finding a group use the Discussions forum to search for group partners.

Each student has to take major responsibility for one of the exercises and collaborate with the team members on the remaining exercises. Each exercise needs one student taking major responsibility. The group has to make sure that the workload is evenly distributed.

2 Assignment

The goal is to design a library and implement evolutionary algorithms for the traveling salesperson problem (TSP). The input for the TSP is given by n cities and distances d_{ij} for traveling from city i to city j, $1 \leq i, j \leq n$. A tour starts at one city, visits each city exactly once, and returns to the origin. The goal is to compute a tour of minimal cost. The TSP is one of the most famous NP-hard optimization problems and you should build an EA library and design evolutionary algorithms for this problem.

When designing your library for the TSP, it is desirable to follow object-orientated design practices and build a modular system. It should be possible to extend the system by using different methods for each part of the design, e.g., different individual representations, operators, and selection methods. In the following, we list the different modules that need to be implemented for the TSP problem. These are basic requirements and you may feel free to add additional features and operators to your library. If the parameter setting is not specified in detail then you are free regarding your choice. You should, however, take into account the recommendations given in the lecture.

You can use any of the following programming languages: Python or Matlab.

Exercise 1 *Team work: who has done what? (zero points)*

We'd like each team member to write one paragraph about what he or she has contributed to this assignment. We will not mark this, and it will not have any effect on the marking of the other exercises. You might now ask "why do this then?" — well, through this no-stakes approach, we'd like to encourage self-regulation within the group and cooperative learning. You can't lose, you can only win.

Exercise 2 *Problem Representation and TSPlib (5 marks)*

Write a class *TSPProblem* which represents the TSP problem. Your class should enable the construction of TSP problems from the files of the symmetric traveling salesperson problem of the TSPlib, which is available online:

```
For the problem files:
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/
--> Download the file ALL_tsp.tar.gz (this contains some of the instances
    that are not directly listed on that page)
The TSP main page has more information for you, if you want to read a bit:
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
```

Exercise 3 *Individual and Population Representation (10 marks)*

Represent a possible solution to the TSP as a permutation of the given cities and implement it in a class *Individual*.

Evolutionary algorithms often start with solutions that are constructed uniformly at random. Write a method that constructs such a solution in **linear time** – not in O(n log n) or even $O(n^2)$, but in O(n), where n is the number of cities.

A population in an evolutionary algorithms represents a set of solutions. Implement a class *Population* for representing a population which is a set of individuals. Make sure that you can evaluate the quality of a solution with respect to a given problem.

Exercise 4 *Variation operators (8+16 marks)*

- Implement the different mutation operators (insert, swap, inversion, scramble) for permutations given in the lecture.
- Implement the different crossover operators (Order Crossover, PMX Crossover, Cycle Crossover, Edge Recombination) for permutations given in the lecture.

Exercise 5 *Selection (15 marks)*

Implement the different selection methods (fitness-proportional, tournament selection, elitism) given in the lecture.

Exercise 6 *Evolutionary Algorithms and Benchmarking (9+10+5 marks)*

- Design three different evolutionary algorithms using crossover and mutation, based on the implementation of your different modules. Your algorithms should perform as best as possible. Justify your design choices.
- *(Experiment 1)* Run your three algorithms *once* with population sizes $10, 20, 50, 100$ on the instances EIL51, EIL76, EIL101, ST70, KROA100, KROC100, LIN105, PCB442, PR2392, and USA13509 from TSPlib. Report the outcomes after 5000, 10000, and 20000 generations. To clarify: these are 3*4*10=120 runs, i.e., 3 algorithms X 4 population sizes X 10 instances.
- *(Experiment 2)* From these three algorithms, select your "best" algorithm, justify your selection, and run your "best" algorithm with a population size of 50 for 20000 generations on the ten TSPlib instances mentioned above. Report for each instance either (1) the average cost and the standard deviation or (2) the median cost and the interquartile range.

Notes:

- For the large instances, e.g., USA13509: if you cannot finish a single run, please report the results that you can achieve within some time limit, e.g., "after 4 hours" or "after 8 hours".
- How often you repeat the experiment involving your best algorithm is up to you. 10 repetitions is a good start, 30 or 100 repetitions will result in more reliable means/medians, however, you might not have the time to run the experiments that often.
- How to report the outcomes: for each of *Experiment 1* and *Experiment 2*, submit the results in a plain text file or in an Excel spreadsheet (or similar).

It is absolutely critical that you provide detailed instructions on how to reproduce the results. With this, we mean that the TAs have to be able to run your code and get results that are somewhat similar to what you are reporting. This means that you might want to create a file with all the commands needed to run each experiment.

Exercise 7 *Inver-over Evolutionary Algorithm (17+5 marks)*

Read the paper "Inver-over Operator for the TSP" by Guo Tao and Zbigniew Michalewicz available at

`https://cs.adelaide.edu.au/~zbyszek/Papers/p44.pdf`

- Implement the algorithm as described in the paper.
- Run the inver-over algorithm with a population size of 50 for 20000 generations on the TSP instances mentioned above. Run the algorithm on each instance 30 times. Report the average cost of the tour you obtained for each instance as well the standard deviation.

3 General procedure for handing in the assignments in this course

Work should be handed in using the course website. The submission must include:

- pdf file of your solutions for theoretical assignments
- all source files (if you created any): if your code does not compile or if it is not sufficiently well documented, **we will cap the code-related marks at 50%**
- descriptions and additional files as required in the statement of the exercises
- a file name README.txt that contains instructions to run the code (if any), the names, student numbers, and email addresses of the group members
- for each group, there should be only 1 submission

Failure to meet all requirements of the 'General procedure for handing in the assignment' will lead to a reduction by twenty (20) marks.

Note: there will be a progress presentation session for this assignment. This is a big opportunity for you to get feedback on your progress, and to make last adjustments for the final submission. In these progress presentations, we are expecting each group to briefly outline their achievements. You should not repeat what the assignment is about – we all should know this by then. Outline your approach, your results (screenshots), and tell us about your challenges – maybe we can help you right away.

The following link contains general information that can be helpful:

- Feedback that we have given in the past:

  `https://cs.adelaide.edu.au/~markus/teaching/feedback.txt`

  (not everything is applicable here)