CS492B
Search Based Software Engineering, Autumn 2016

Coursework #1: Stochastic Optimisation
Due by 23:59, 30 September 2016

1 Aim

Research metaheuristic algorithms and implement solvers for the Travelling Salesman Problem (TSP). The only restriction in the choice of algorithm is that the basis has to be a stochastic optimisation. In fact, exact classical optimisation will not cope well with some of the problem instances we tackle, due to high computational cost. Hybridisation with other exact algorithms is encouraged if appropriate.

2 Travelling Salesman Problem

The goal is to solve TSP instances as well as possible: solutions will be measured by Feel free to use the example code, shown during the classes, as the starting point, but you are free to choose your own language and reimplement the algorithm. Problem instances in TSPLIB can be found here: `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`; please use the plaintext format.

Your program should take the `.tsp` file (exactly as it is downloaded from the above site) as the input, and create a single output file, named `solution.csv`. It should contain a single column of city indices, in the orde of your solution to the TSP. Also, print out the total distance travelled on the standard output. For example, if the solution is to visit cities in the order of 5, 4, 1, 3, and 2, and the distance travelled is 8934.12:

Listing 1: Example output

```
> python tsp_solver.py dj38.tsp
8934.12
> cat solution.csv
5
4
1
3
2
>
```

For any other implementational detail, please use the coursework forum on KLMS to interact with me and T/As.

3 Interface

Please provide ways to control the following parameters; if possible, use the flags given below. This is to promote fair competition.

- Population (`-p`): if you use population based algorithm, we should be able to control the parameter.
- Fitness evaluations (`-f`): you should be able to limit the total number of fitness evaluations.

If you implement other parameters, please document them in the report in appropriate detail.

4 Competition

There will be a class competition on the online leaderboard (`http://coinse.kaist.ac.kr:8000`). Submit a solution to the `rl11849` problem instance from the TSPLIB to the lieaderboard, using your solver. The person who has submitted the shortest tour by the due date will win a **prize**. Later, we as a group will have a chance to go through interesting solutions. Note that **the competition result is not directly linked to the grades**, as marking will consider other aspects, such as report/documentation, implementation quality, and the novelty of the optimisation idea.

5 Deliverables

Each person should submit the following deliverables by the submission deadline:

- **Implementation**: solvers for two problems, self-contained in separate directories (see below).
- **Report**: include a written report that contains detailed descriptions of how you approached the problems. Describe the optimisation you have implemented in as much detail as possible. There is no page limit.

For ease of marking, follow the following directory structure, and submit a zip file containing the top level directory, through KLMS.

```
[your student number]
├── report.pdf ........ Your report documenting both implementations
└── tsp ............... TSP solver and dependencies
```

6 Guidelines

- Examples are given with `python` only for illustrative purposes; you are free to choose any programming language.
- **BUT** you do have to implement the optimisation algorithm yourself; do not use pre-developed frameworks.
- Make sure your submission is self-contained. It should not depend on any file outside the submitted directory, such as files on your own hard drive or online. We expect the solvers simply to work out of the box. If you use Windows machines (or, in fact, any machine of your own), we strongly recommend that you test the submission on a separate machine, in order to test whether it is relaly self-contained.