CS454 AI Based Software Engineering

Autumn 2020

Coursework #2: Stochastic Optimisation
Due by 23:59, 5 October 2020

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.

2 Travelling Salesman Problem

The goal is to solve TSP instances as well as possible. Your submission will be graded using a hidden problem instance. Problem instances in TSPLIB can be found here: `http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html`; use the plaintext format in the Symmetric travelling salesman problem subsection (`http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html`).

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 order 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, a `python` example would be like:

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 Classum or email 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 comparisons.

- 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 Deliverables

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

- **Implementation**: solver for TSP 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/ ..... a directory containing a TSP solver and dependencies
```

5 Guidelines

- **Obviously**, do your own work. We take academic integrity very seriously. Any wrongdoing will be penalised accordingly.
- 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 predeveloped 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.
- Do reasonable documentation, so that we can use your solver for grading.