Advanced Algorithms and Programming   Master DSC/MLDM
Project - 2024-2025

1 Description

The objective of this project is to implement some algorithms for solving the *Traveling Salesman Problem* (TSP) and to provide an experimental study of their running time and quality.

2 Work to do

You must program different algorithms for the TSP problem, including:

- The brute-force approach exploring all the possible solutions in a systematic way.
- A Branch-and-Bound version.
- The approximation based on the minimum spanning tree as seen in the exercise sheet.
- A version considering at each step the nearest unvisited choice (called the *greedy* approach).
- A dynamic programming approach, you take inspiration from the paper of *Bouman et al.* uploaded on claroline [1], and from the wikipedia page of the *Held-Karp algorithm*: `https://en.wikipedia.org/wiki/Held%E2%80%93Karp_algorithm`.
- The Branch-and-Bound algorithm based on edge selection and matrix reduction presented in the file `TSP_BB-edges-Reingold77.pdf`[2]
- A version based on a randomized approach.
- A version based on a genetic programming or ant colony approach.
- Another personal version of you choice

Note that all the algorithms must be able to output the solution - *i.e. the ordered sequence of nodes to visit* and the cost of the proposed solution.

A graphical user interface could be proposed to illustrate the behavior of the algorithm(s), but **this is not required** and must be done after the implementation of the algorithms.

The running time of the algorithms and the quality of the solutions given (in terms of optimality) have to be evaluated on problems of different sizes and structure (*i.e.* with different number of cities, different distances between cities, different underlying graphs, ...).

For this purpose, you can implement a random generator able to create automatically some TSP problems. It can take into account some options like the level of sparsity of the problem (*i.e.* level of connectivity), a range over possible edge weights, a distribution for generating the weights, ...
Your evaluation must also include existing benchmarks where the optimal solution is generally known (you may propose other existing databases), have a look here to the `TSP` and `ATSP` data:
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html`

[1]Full reference: *Bouman, Paul C., Niels A. H. Agatz and Marie Schmidt. "Dynamic programming approaches for the traveling salesman problem with drone." Networks 72 (2018): 528-542.*

[2]This document is extracted for educational purposes from the book: *Combinatorial algorithms: theory and practice, Edward M. Reingold, Prentice-Hall, 1977.*

You have to provide a full and rigorous experimental study to illustrate the different strong and weak points of each algorithm. In particular, it is expected that the efficiency and the quality of the solutions are studied with care.

The project can be done in a group from 3 to 5 students, and each student must program **at least** one algorithm. You are free to use any programming language among: python, C, C++, java, javascript, but the same language must be used by all the members of the group. It is recommended to maximize the number of students of the same master track in each group to ensure a similar availability between group members.

3 What to send

Each group must upload before **Friday November 8, 22:00** a text containing the members of the group on moodle, in the related resource. This text **must also contain a provisional planning** giving the milestones of the project with the tasks to achieve, the repartition between the group members and the time deserved to each task, if possible the group can mention the procedures used for testing and evaluating the programs. The ability to respect the schedule will not be used for evaluating the grade, but during the defense the students must present the real planning and discuss the reasons why the project has run late. The group must also indicate the workload between the members of the group in percentage[3]. The quality of the presentation (report, oral, ...) and the quality of the source code will be taken into account.

Erick Gomez Soto will supervise the advancement of the project, you will have 6 meetings for discussing your achievements. **Note that all the questions and mails related to the project must be sent to him:** `erick.gary.gomez.soto@univ-st-etienne.fr`.

The final version of the projects must be uploaded on moodle, in the related resource, before **Tuesday December 17, 22:00**, in a `.zip` or `.tar.gz` archive, **well-organized**. It must include the source code of the programs, information for installing and using them, and a report in PDF format. A project defense will scheduled on **Friday December 20** morning or afternoon (the exact schedule will be given later), be careful to come to the defense with a working implementation. The slides must be uploaded on the related ressource in moodle by **Friday December 20, just before your turn**.

Grade scaling:

- at least 7/20 : brute-force and branch-and-bound versions must work correctly, and be evaluated in a rigorous manner on artificial data, report and source code must be presented neatly.
- at least 10/20 : each member of the group must program a different approach, experimental evaluations on artificial data must be rigorous, one problem of the TSP database must be used, source code and report must be presented neatly and questions answered correctly.
- at least 12/20 : in addition to the previous point, evaluate the algorithms on a bigger part of the TSP database available and implement another method (the number of methods must be higher than the size of the group)
- at least 14/20 : in addition to the previous point, implement two new methods, and process a significant part of the TSP database, source code and report must be extremely neat and clear.
- at least 16/20 : program all the algorithms, process a large part of the TSP database, try to find on which problem instances each method is efficient. Option: propose a graphical user interface.

Note that for a given grade, the absence of one element can be compensated by the addition of an element associated to a higher grade. **<u>Be careful, the grading scale is given for information purpose.</u>**

[3]For a group of 4 students, if the workload was the same for every member, the group will indicate 25% for each, otherwise the group must give the distribution of the workload between the group members such that the quantities given sum up to 100%.