Advanced algorithms **Master CPS2/DSC/MLDM**

Project - 2018-2019

---

1 Description

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

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 version adding and removing edges as seen in the exercise sheet.
- 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 (designed on your own or based on a publication found on the web that you must cite and detail properly the algorithm and complexities in your report)
- A version based on a randomized approach.
- A version based on a genetic programming or ant colony approach.
- Another personal version.

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, ...). For this purpose, you can implement a random generator able to create automatically some TSP problems. The generator 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 available at the given URL where the optimal solution is generally known (you may propose other existing databases):
`https://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95`

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 of 4 or 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.

3 What to send

Each group must send before **Friday November 9, 11:59 AM (noon)** an email containing the members of the group, 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 potential different elements used for testing the different programs. This email must be sent to **Leo Gautheron:** `leo.gautheron@univ-st-etienne.fr` - Leo will supervise the progress of the projects. **Note that all the questions and mails related to the project must be sent to him**.
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 quality of the presentation (report, oral, ...) and the quality of the source code will be taken into account.

The final version of the projects must be uploaded on Claroline Connect, in the `Project_archive` resource, before **Tuesday December 11, 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 **Thursday December 13** morning (the exact schedule will be given later), be careful to come to the defense with a working implementation.

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 the answers to the questions must be correct.
- at least 12/20 : in addition to the previous point, evaluate the algorithms on a 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 and try to find on which problem instances each method is efficient. Optionally, a graphical user interface can be proposed.

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. **Be careful, the grading scale is given for information purpose.**