Methods and Models for Combinatorial Optimization

Lab exercise - Part II

L. De Giovanni

For the combinatorial optimization problem described in Part I, it is required to:

- design and implement an ad-hoc optimization algorithm, as an alternative to solving the implemented model with Cplex or OPL. Any meta-heuristic can be developed, as well as any of the approaches presented for the Travelling Salesman Problem (e.g., a constraint generation or a cut-and-branch approach based on separating sub-tour elimination constraints). You can also keep inspiration from any method you can find in literature for related problems, if it is related to the content of the course: if you decide to start from some algorithm you find in literature, *please ask to the teacher before* starting the implementation;

- it is allowed (in particular for "OPL guys") the use of available optimization libraries like Matlab genetic algorithms, or Python libraries. Of course, you can find ready-to-use code for the TSP but, in this case, you should add some functionalities to the available code (e.g. some more initialization options, intensification/diversification phases, ad-hoc mutations etc.);

- test the performance of the implemented method, present the computational results and compare the results to the ones obtained with the model solved by Cplex or OPL in Part I (use the same instances generated for Part I). Typical questions to be addressed during the performance test are: what are the running times? what is the gap with respect to the optimal solution? It is better to use the proposed heuristic or the Cplex/OPL model? etc. Notice that you should have more than one instance per size, so that average results should be provided. Moreover, in case the implemented algorithm has randomized components, you should either have a large number of instances, or rerun several (e.g. 10) times the algorithm on the same instance and collect statistics (average times or performance, standard deviations, min, max etc.);

- write a final report (10-20 pages) where you describe how you implemented the model of Part I in Cplex or OPL (some implementation details, for example, about the maps used to access variables, or how you created variables and constraints), some design issue concerning the algorithm of Part II (not the general heuristic or branch-and-bound that one can find in the notes or in a book, but how the method was customized), the feature of any used libraries (where applicable), and the computational results (summarizing tables and some comments) of both Part I and Part II, and their comparison.