Large Scale Optimization for Transport & Mobility

Assignment 3

Deadline: Friday October 24, 17:00

Rolf van Lieshout

1. This is an individual assignment. Therefore, it is **not** allowed to collaborate with others (helping each other with small coding questions is allowed). Copying code or text from others is considered plagiarism. If you use online sources (such as Stack Overflow, but also Python libraries), indicate this clearly in the comments of your code.

2. Test your implementations on small instances you constructed yourself, and verify that the results are correct. It is often good practice to write a little bit of verification code to verify the correctness of your solutions.

A Lagrangian Heuristic for UFL

In this assignment, you will implement a Lagrangian Heuristic for the Uncapacitated Facility Location Problem (UFL). For information on this heuristic, please consider the slides on MIP-Based Heuristics. In contrast to the previous assignment, the assessment is mainly on the solution correctness. You are asked to only write a small report.

To get you started, we have prepared code for the structure and basic functionalities of the program, which you can download via Canvas. The *zip-file* contains the following files:

- **`UFL.py`:** the Python file that contains classes for a UFL instance, a UFL solution and for a Lagrangian heuristic. You will have to extend these classes.

- The folder "Instances": a folder with five benchmark instances.

Throughout the assignment you are advised to work with numpy arrays. You are asked to perform the following steps:

1. Complete the methods `isFeasible(self)` and `getCosts(self)` in the UFL_Solution class. (1 point)

2. Compute the lower bound $\theta(\lambda)$ in the method `computeTheta(self,labda)` in the LagrangianHeuristic class. This method should return a double. (2 points)

3. In `computeLagrangianSolution(self,labda)`, compute the Lagrangian solution. This method should return a solution as a UFL_Solution. (2 points)

4. In `convertToFeasibleSolution(self,lagr_solution)`, convert the input solution to a feasible one. Return a UFL_Solution. (1 point)

5. In `updateMultipliers(self,labda_old,lagr_solution)`, write a method that updates the Lagrangian multipliers based on the old multipliers and the obtained Lagrangian Solution. Return the new multipliers. To ensure convergence, it is advised that the updates to the multipliers become smaller as the algorithm progresses. (2 points)

6. Combine all written methods in `runHeuristic(self)`. (1 point)

7. Write a short (max. 2 pages) report where you explain your update rule, your termination criterion, and where you present results of your heuristic. Report your best obtained lower and upper bounds for all instances (note that you can still obtain bounds even if you did not complete step 6). Make a figure of the progression of the bounds over the iterations. Add the code in the appendix. (1 point)

Your submission should contain both the report in pdf format, and a zip-file that contains your code. In case anything about the assignment is unclear, please use the designated discussion board on Canvas.