IE 670: Selected Topics in Logistics Optimization
Fall 2020: Homework assignment 1
(Friday November 2nd)

Preliminaries

Throughout the semester, I would like you to type your solutions using LaTeX. If you are not familiar with it, this would be a great opportunity to learn how to use it. You will definitely find it helpful in the future. For those who are not used to type in LaTeX, I would like to ease the transition by asking you to type only a small percentage of your solutions. For this homework, I would like you to latex the answer of the first problem. Do not take this as an upper bound; you are more than welcome to latex more than one problem. For an introduction to LaTeX, please refer to http://www.ctan.org/tex-archive/info/gentle/gentle.pdf.

If you received some help to obtain the solution of a problem, you should acknowledge the source of help you received. In particular, for each question, I would like you to cite, if applicable, any book you consulted, any website you searched, or any individual you cooperated with (other than the instructor). This information will not be used to adjust your homework score provided that help is limited to a reasonable portion of the homework. Remember, it is acceptable to ask for hints if you feel "stuck" on a problem. However, by no means you should share complete/almost-complete solutions, codes, formulations, mathematical proofs, or any other material that represents a substantial portion of the problem being solved.

Finally, I would like you to candidly assess the amount of work it took you to complete each of the problems of the homework.

Test Instances

I created a test instance generator for this Homework. The test generator is written in javascript and embedded within an HTML file. It can be downloaded from UBLearns and you should be able to open it in your favorite web browser. I have successfully tested it in Chrome and Firefox, however, I am not sure of its performance in other browsers (e.g., Safari, IE). The generator works as follows: Given any natural number $n > 1$ and an instance number s, the generator will return to you a text file with the (x, y) coordinates—separated by a Tab—of n points (customers). Note that these instances are obtained using a fixed seed and therefore, the instance returned for a fixed n and a fixed s will never change. Note also that this generator allows you to generate problems of size as large as you want. The output is a `*.dat` file similar to the one depicted in Figure 1. The first line indicates the size of the instance. For all the problems, assume that the distances are Euclidean.

Independently of the programming language you use, your codes must receive as input, files with exactly the same format as the ones produced by the generator.

```
5
13.363731279969215        6.805373430252075
17.699855933897197        19.325826507993042
3.55214505456388          10.830293209291995
10.180472247302532        5.402365988120437
14.7881346847862          4.2188604129478335
```

Figure 1: Test file `TSP_instance_n_5_s_1.dat`

Part 1: TSP Formulations

Implement the following formulations for solving the TSP:

- Miller-Tucker-Zemlin
- Multicommodity flow formulation
- Shortest path with constraints
- Quadratic Formulation (linearized version)
- Dantzig, Fulkerson, and Johnson (subtour-elimination)
  - Plain loop (no callback)
  - callback with lazy cuts
  - (Bonus) callback with both lazy and user cuts

  For some advice on how to implement this, see the extra videos about Gurobi I posted on UB Learns.

To test the performance of these formulations, I would like you to incorporate into your report a table containing: the optimal value of the linear relaxation, the best integer solution, the optimality gap, and the running time. Solve 30 instances 5 of size 15, 5 of size 20, 5 of size 25, 5 of size 30, 5 of size 50, and 5 od size 70. Set the time limit of each run to 10 min. Your report must include a discussion of your findings.

Part 2: The Solving the Dantzing-Fulkerson-Johnson formulation using Lagrangian relaxation (Held & Karp bound)

The goal of this section is to get familiar with the Lagrangian relaxation technique from a practical perspective, and to experience how it can be helpful in solving discrete optimization problems. In order to gauge the strength of the Lagrangian dual, it would be best to compare the optimal value of the dual with respect to the optimal value of the problem. However, obtaining the optimal value of TSP might be difficult (I expect your results in Part 1 would have you convinced of this by now). You can use as an upper bound the best solution obtained with the formulations of part 1. Therefore your implementation will focus on obtaining a lower bound.

1. Implement an algorithm that generates a minimum cost 1-tree. For that you must code a minimum spanning tree algorithm (Kruskal's or Prim's). You can use the one I am posting on UB Learns as well.
2. Implement a Subgradient algorithm for solving the Lagrangian dual of the Dantzing-Fulkerson-Johnson formulation. Use as the step length sequence the function $h_k = M\rho^k$, for $0 < \rho < 1$. Test different values for M and ρ. As stopping criterion use:
   (a) If the current subgradient is 0
   (b) If $|L^k - L^{k+1}| < \epsilon$ for a small ϵ. Test different values for ϵ
   (c) If the number of iterations is larger than K. Test different values of K.
3. Use the Lagrangian relaxation to find lower bounds of the same instances you used for Part 1.

Part 3: Approximation Algorithms

Use the instance generator to produce a 10-node instance using as s the last three numbers of your UB person number. Use both approximation algorithms covered in class (by hand, you do not need to code them) to find feasible solutions to such instance. Solve the instance using any of the formulations from part 1 and use the optimal value to contrast the quality of the solution obtained with the approximation algorithms.