FSF3827 Applied Linear Optimization, 2025
Assignment 1
Due Monday November 3 2025 23.59

Instructions for the assignments

- **Collaborations.** It is fine for students to collaborate on solving the assignment. However, every student needs to submit a report and if you have collaborated with someone you need to mention this clearly in the report. If you collaborate, it is essential that you all solve each task and fully understands all parts.
- **Using AI / building upon code you found online.** It is fine to use an AI tool or build upon code that you find online. However, you need to fully understand the code and be able to discuss all parts of the code without help from AI. The teacher will ask questions and make sure you understand what the code does. If you use AI, I recommend you ask it to help with writing parts of the code or subroutines and not full programs.
- The report should have a leading title page where the name, personal number, and e-mail address is clearly stated.
- You are encourage to write the reports in Latex.
- Content
  1. The solutions must be described in enough details that other students (and the teacher) can easily read the report. For example, clearly define variables and sets used in the formulations.
  2. Describe the main parts of the problem formulations. For example, what does the constraints do.
  3. Describe the main parts of the code (what it does and how it works).
- Everyone needs to upload on the Canvas page of the course no later than by the deadline of the assignment:
  - The report as a pdf file.
  - all the relevant code in a zip file.
- You will be graded on a pass/fail scale. If the reports is insufficient, you will get second chance to revise the report.

The purpose of this assignment is two folded I) you will become familiar with one of the most classical mixed-integer programming (MIP) problems, II) become familiar with solving MIP problems and the great computational difference between formulations.

The traveling sales person (TSP) problem is a classical MIP, or actually a combinatorial optimization problem. In this assignment you will work with TSP problems to get familiar with modeling problems, some important concepts/techniques used to solve MIP problems. TSP have actively been studied since the 1950s, and it is a very difficult class of problems. A basic branch-and-bound algorithm will perform very poorly on these, but if you do some smart things you can solve impressively large problems (optimal tour through all cities on earth).
After having worked on this assignment (or during), I recommend that you watch this presentation `https://youtu.be/5VjphFYQKj8?si=XqSqwVWXJpnT2u3j` by William Cook (who I would call the leading researcher on TSP).

On cavas you will find some files with the extension .tsp. These contains data about some symmetric TSP problems of varying size. These problems are taken from TSPLIB `https://github.com/mastqe/tsplib` where you can also find optimal solutions, which is useful for checking that your code is correct.

The .tsp files look like this

```
NAME: berlin52
TYPE: TSP
COMMENT: 52 locations in Berlin (Groetschel)
DIMENSION: 52
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 565.0 575.0
2 25.0 185.0
3 345.0 750.0
```

The first value is the node number (or city) and the second two values are its coordinates. You can use the coordinates to compute the distances between all nodes. In the TSPLIB `EUC_2D` convention, distances are rounded to the nearest integer by

$$d_{ij} = \text{int}\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + 0.5\right). \tag{1}$$

I recommend that you use Gurobi to solve the problems.

Tasks

1. Formulate the TSP problem using the Miller–Tucker–Zemlin formulation. Explain the formulation, specifically how the subtour elimination work. Write a code that reads the TSP problems, creates and solves the Gurobi model, solves the problems, and plots the solution. Try to solve the test problems in Canvas using this formulation. You can set a time limit of 5 minutes, after this you can stop (the larger problems should be quite difficult).
You can find the Miller, Tucker and Zemlin paper here `https://dl.acm.org/doi/pdf/10.1145/321043.321046` I recommend reading it (it is short).

2. Formulate the problem Dantzig–Fulkerson–Johnson (DFJ) formulation. Implement this in Gurobi and directly add all the subtour elimination constraints. Try to solve the smallest two problems using this formulation. You can find info about the DFJ formulation for example here `https://en.wikipedia.org/wiki/Travelling_salesman_problem` or in this chapter `https://link.springer.com/chapter/10.1007/978-3-540-68279-0_1` you can access and download it from a KTH network.

The DFJ formulation is known to have a much stronger continuous relaxation than MTZ. But, it results in a HUGE number of constraints.

Fortunately, most of the subtour constraints are not needed. An efficient way to use the this formulation is by cut generation (also called row generation or lazy cuts). Note, a cut is simply an inequality constraint that is satisfied for any feasible solution. Then you first solve the problem without any subtour elimination constraints, check if the solution contains subtours. If it contains subtours add constraints to exclude the subtours. The solver does not need to restart after adding the constraints, it can simply continue by adding the constraints to the subproblems and continue the branch and bound search. The main thing that changes is that the solver needs to be more carful with upper bounds and pruning on upper bounds (an upper bound is not given by all integer solutions found it also needs to be a valid tour).

This can be done directly with the Cut Callback functionality in Gurobi. It allows you to stop at nodes and add cuts. Online you can find many examples of this and the DFJ subtour elimination constraints is a typical example of this. ChatGPT can also help with this.

3. Write a code that reads the TSP problems, and generates the DFJ formulation without any subtour elimination constraints and add the subtour elimination constrains through cut callbacks. Solve the problems using this method with Gurobi (especially the larger problems should be much faster).

4. Implement a simple greedy heuristic. Start at a random city, and go to the closest city that you have not yet visited. From there, you again go to the closest city not yet visited. Do a few different random initializations (for example 100 starts), and save the best solution. How much worse are these solutions? (Fun fact, from what I have seen pure AI approaches struggles to find much better solutions)

5. Let's go back to the MTZ formulation, and we want to extend the model. Let's assume you start in city 1. For each city $i > 1$ there is an upper limit u^i on how far you are allowed to travel before you reach city i (for example, it should not take you longer than some time to reach the city). Similarly, there is a lower limit l^i on how far you must travel before reaching city i (for example, you should not arrive earlier than some specific time in the city). We assume that u^i and l^i are such that these distance constraints are not trivially satisfied by the shortest tour. How could you extend the model to handle these additional constraints? You only need to write down the mathematical model, but if you want you can test it for one of the TSP problems to test if it works. Describe what the additional constraint/variables do.

Good luck!