Project // IND E 599 // Due: March 7th 11:59pm via Canvas

1 Introduction and Software

The purpose of project is for you to learn how to use a commercial MIP solver. The goal is to familiarize yourself with the basics, so that you will be more comfortable implementing more advanced approaches (e.g., branch-and-cut). You can use any software tool to implement it, like CPLEX, Xpress or Gurobi (in conjunction with a supported coding language like C++ or Python). I prefer Gurobi or Xpress since it is very easy for academics to get free licenses, and it is also faster than CPLEX for MIP[1]. There is also a nice collection[2] of sample code (which is perhaps the best way to learn).

2 The Problem: Traveling Salesman Problem

The project problem this quarter is the asymmetric traveling salesman problem: given a complete directed graph $G = (V, E)$ and a cost c_{ij} for each directed edge (i, j), find a directed Hamiltonian cycle of G of minimum total cost. Consider the following formulations/relaxations for this problem.

2.1 Formulation 0: The Assignment Relaxation

What follows is the assignment relaxation. This formulation is integral (i.e., solving its LP relaxation gives an integral solution), however it may result in subtours.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1a}$$

$$\text{s.t.} \sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \tag{1b}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \tag{1c}$$

$$x_{ii} = 0 \qquad \forall i \in V \tag{1d}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{1e}$$

The three formulations below add some constraints (and sometimes variables) to ensure that subours will not be formed.

[1] `http://plato.asu.edu/bench.html`

[2] `https://www.gurobi.com/jupyter_models`

2.2 Formulation 1: The Cut Formulation

The cut formulation was (essentially) introduced by Dantzig, Fulkerson, and Johnson in a 1954 paper published in OR. They used it to solve a TSP instance with 49 cities, one node for each state in the contiguous USA and another for Washington D.C. (Actually, they were able to collapse some nodes based on optimality arguments, so they ended up working with a 42-node instance.) The formulation is the same as the Assignment Relaxation, except that we add the following constraints.

$$\sum_{i \in S} \sum_{j \in V \setminus S} x_{ij} \geq 1 \qquad \forall S \subseteq V,\ 2 \leq |S| \leq n-2. \tag{2}$$

The Concorde TSP Solver uses this formulation (with some additional cuts) to solve the classical 42-city instances from the Dantzig, Fulkerson, and Johnson paper in 0.09 seconds[3]. This time was obtained in December 2003 using a computer with a dual-core 2.8 GHz Intel Xeon processor and 2 GB RAM and CPLEX version 6.5.

Since this formulation is non-trivial to implement, you are *not* tasked with implementing it for this project. Still, it is an important formulation to keep in mind.

2.3 Formulation 2: The MTZ Formulation

The MTZ formulation was introduced by Miller, Tucker, and Zemlin in a 1960 paper published in the Journal of the ACM. It uses additional variables u_i representing the position of vertex i in the tour. For example, $u_v = 2$ means that vertex v is the second city to be visited. The formulation is the same as the Assignment Relaxation, except that we add the u_i variables and the following constraints, where a vertex $r \in V$ is arbitrarily selected to be the first vertex visited in the tour.

$$u_i - u_j + 1 \leq n(1 - x_{ij}) \qquad \forall i, j \in V \setminus \{r\} \tag{3a}$$

$$u_r = 1 \tag{3b}$$

$$2 \leq u_i \leq n \qquad \forall i \in V \setminus \{r\}. \tag{3c}$$

It is acceptable to enforce integrality on the u_i variables, although this is not strictly necessary. The "big-M" constraints (3a) enforce the vertex ordering and can be interpreted as follows. If $x_{ij} = 0$, then the constraint does nothing. However, if $x_{ij} = 1$, meaning that vertex j is visited directly after visiting vertex i, then we should enforce $u_j \geq u_i + 1$, i.e., $u_i - u_j + 1 \leq 0$.

2.4 Formulation 3: The MCF Formulation

A multi-commodity flow (MCF) formulation for TSP is as follows. As before, arbitrarily select a vertex $r \in V$ to be the "first" vertex visited in the tour. Introduce (new) flow variables f_{ij}^v representing the decision to route one unit of flow from r to v across arc (i, j). The formulation is the same as the Assignment Relaxation, except that we add the flow variables and the following

[3] `http://www.math.uwaterloo.ca/tsp/concorde/benchmarks/bench.html`

constraints.

$$\sum_{j \in V \setminus \{r\}} f^v_{rj} - \sum_{j \in V \setminus \{r\}} f^v_{jr} = 1 \qquad \forall v \in V \setminus \{r\} \tag{4a}$$

$$\sum_{j \in V \setminus \{i\}} f^v_{ij} - \sum_{j \in V \setminus \{i\}} f^v_{ji} = 0 \qquad \forall i, v \in V \setminus \{r\},\ i \neq v \tag{4b}$$

$$0 \leq f^v_{ij} \leq x_{ij} \qquad \forall i, j \in V,\ \forall v \in V \setminus \{r\}. \tag{4c}$$

The constraints (4a) ensure that one unit of flow (of commodity v) leaves the "root" vertex r and the constraints (4b) ensure that this flow is conserved at all nodes besides v (meaning that this unit of flow will be consumed at v). The constraints (4c) ensure that these flows can only be sent across chosen arcs. Together, these constraints ensure the existence of a path from r to every other node in the graph. This, along with the indegree and outdegree constraints from the Assignment Relaxation, ensures that the selected edges will form a tour.

The MCF formulation is equivalent in strength (in the x-space) to the cut formulation. It is also compact (i.e., has polynomially many variables and constraints). These properties make it seemingly a good formulation to use. However, its large size $\Omega(n^3)$ means that (good implementations of) the Cut formulation should outperform the MCF formulation.

2.5 The Test Instances

The data for this project comes from TSPLIB, a well-known library of TSP benchmark instances.

The instances can be downloaded here:
`http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html`

However, I have posted four of these instances to Canvas in my own format. In them, the first line indicates the number of nodes, and the next lines give the entire distance matrix. It is symmetric and has zeros on the diagonal.

Optimal objective values for these instances are available here:
`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html`

This website can help you to verify that your code is working properly.

3 Your Tasks

This project entails the following:

1. Code up Formulation 0, Formulation 2, and Formulation 3 for Gurobi or for CPLEX, using a language like C++ or Python. You should not hard-code any of the problem data; your codes should work for any values of n and c. If you have to recompile your codes between instances, you are doing it wrong. Your code should read data from an external file.

2. Test Formulation 0 on the four instances: bays29, dantzig42, pr76, and rat99. (The number at the end refers to the number n of nodes.) Report the solution time and the objective value.

3. Test Formulation 2 and Formulation 3 on the same four instances as above[4]. Specifically, solve each instance:

   (a) once with default settings;

   (b) once with presolve turned off (see the Gurobi parameter *Presolve*[5]);

   (c) once with cuts turned off (see the Gurobi parameter *Cuts*);

   (d) once with both presolve and cuts turned off.

   Do this for each of the two formulations. Set the acceptable MIP gap to zero (see the Gurobi parameter *MIPGap*). Impose a time limit of one hour (3600 seconds). In each test run, report the time (in seconds) to solve the MIP (or its best LB and UB when it timed out), the number of branch-and-bound nodes explored, and the number of cuts added by the solver. You will likely need to "do some googling" to find out how to impose time limits, etc. Be careful that you do not report an instance "solved" when it actually timed out. You should also report the root LP relaxation bound for each formulation on the instances.

4. Based on the results from your tests, what can you say about:

   (a) the relative strength of the LP relaxations of the two formulations (2 and 3)? (*how much* stronger)

   (b) the effect of the stronger formulation on the number of branch-and-bound nodes?

   (c) the effect of the stronger (but larger) formulation on the *total* runtime?

   (d) the importance of presolve?

   (e) the importance of good cuts?

   Also, how does your code compare to Concorde? Why do you think this is?

Submit the following to Canvas:

1. Your code.

2. A typed report that clearly presents your results that answers the questions from above. (If you use LaTeX in your research, I expect you to use LaTeX for your report.)

[4] When applying Formulation 3, it helps to use an *interior point* (a.k.a. *barrier*) method to solve the root LP relaxation. This is controlled by the Gurobi parameter *Method*. (Or, use the *concurrent* option which tries barrier, primal simplex, and dual simplex simultaneously on different threads.)

[5] `http://www.gurobi.com/documentation/8.0/refman/parameters.html`