1 TSP using Dantzig-Fulkerson-Johnson Formulation

In this problem, we study the classic formulation for the TSP problem, namely the Dantzig-Fulkerson-Johnson (DFJ) formulation, and implement a constraint generation algorithm to solve it.

1.1 DFJ Formulation

The DFJ formulation was first proposed by G. Dantzig, D. Fulkerson, and S. Johnson. All three made pioneering contributions to operations research and optimization. The original paper where they first proposed the formulation has become a classic ("Solutions of large scale travelling salesman problem", *Operations Research*, 2:393-410, 1954.)

$$
\begin{aligned}
(DFJ) \quad \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_{ij} && (1)\\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = 1 \qquad \text{for all cities } i = 1, 2 \ldots, n && (2)\\
& \sum_{j=1}^{n} x_{ji} = 1 \qquad \text{for all cities } i = 1, 2 \ldots, n && (3)\\
& \sum_{i\in S}\sum_{j\in S} x_{ij} \le |S| - 1 \qquad \text{for all subset } S \subset N \text{ and } S \ne N && (4)\\
& x_{ij} \in \{0,1\} \qquad \text{for all cities } i, j = 1, 2, \ldots, n
\end{aligned}
$$

The decision variables $x_{ij} = 1$ if the tour goes directly from city i to city j, $x_{ij} = 0$ otherwise. The objective is to minimize the total travel distance of the tour $\sum_{j=1}^{n} d_{ij}x_{ij}$, where d_{ij} is the distance between cities i and j. Constraint (2) ensures that the tour goes from city i to exactly one other city, and Constraint (3) ensures that the tour goes into city i exactly from one other city. These two types of constraints together are called the *assignment constraints*. However, as we know, they are not enough to characterize a correct tour, because the solution to the assignment constraints may contain subtours of smaller cycles around subsets of cities. We need constraints (4) to eliminate all the subtours. Here $N = \{1, 2, \ldots, n\}$ is the set of all n cities, and S is any strict subset of N with $|S|$ number of cities.

The DFJ formulation has n^2 binary variables. However, the number of constraints is astronomical! Indeed, we need a subtour elimination constraint (4) for each non-empty subset S of N, excluding N itself (why?). There are $2^n - 2$ such constraints. That's why we say the number of constraints is exponentially many in the number of variables. For a problem of 48 cities, there will be $2^{48} - 2 = 281,474,976,710,654$ (281 trillion) subtour elimination constraints. It will be completely nonsensical to enter such a model into Xpress!

1.2 Constraint Generation Algorithm

In the following, we outline a **constraint generation** procedure that can be implemented quite easily in Xpress and can find an optimal (yah!) solution fairly fast. The strategy is as follows:

1. First form the restricted master problem with a subset of constraints (4). In the following we will explore different options for the initial set of constraints.

2. Solve the restricted master problem.

3. Check if any subtour elimination constraint is violated by the current solution using the following method: Find a tour starting at City 1 (which is Atlanta). Such a tour always exists and is unique (Why?). Then, if this tour contains less than 48 cities, then this is a subtour and we can add the corresponding subtour elimination constraint (4) to the restricted master problem. Go to Step 2 and

repeat. Otherwise if the tour has all 48 cities, then it is a legitimate TSP tour, and in fact optimal, then we are done.

1.3 Steps to Implement the Algorithm

The above outlined framework of constraint generation is partially coded in TSP-partial.mos. The data file US48.dat contains coordinates of 48 state capitals. You need to fill in the parts that are marked "!!!!!!!!! fill in your code here !!!!!!!!!!!". In particular, you need to do the following to complete the code:

1. Formulate the objective function, all the assignment constraints.
2. Write small subtour elimination constraints for subsets of 1 city, 2 cities, 3 cities, and 4 cities.
3. Finish the constraint generation part:
   (a) Write code to find a subtour starting at Atlanta (City 1). Hint: One way to find such a tour is to start at Atlanta and see which city comes next, then see which city comes after that, etc, until you trace back to Atlanta. Mark all the cities on the subtour.
   (b) Write code to generate the corresponding subtour elimination constraints and add it to the restricted problem.

1.4 Computational Experiments

1. Begin with all of the small subtour elimination constraints commented out (So, these will be generated by your constraint generation code if necessary.) Use your Xpress code to solve the problem. If it runs for more than 20 minutes, stop Xpress — report "Reach time limit of 20 min". If it does finish quickly, record how long it takes and how many constraints were generated. You can find the running time in the "Stats" tab in Xpress.
2. Now un-comment the 1-city subtour elimination constraints, so that the formulation begins with those constraints already in it. Use your Xpress code to solve the problem. If it runs for more than 20 minitues stop Xpress in the middle and report "Reach time limit of 20 min". If it does finish quickly, record how long it takes and how many constraints were generated.
3. Now start the formulation with both 1-city and 2-city subtour elimination constraints. Use your Xpress code to solve the problem. If it runs for more than 20 minitues stop Xpress and report "Reach time limit of 20 min". If it does finish quickly, record how long it takes and how many constraints were generated.
4. Now start the formulation with 1-city, 2-city, and 3-city subtour elimination constraints. Use your Xpress code to solve the problem. If it runs for more than 20 minitues stop Xpress and report "Reach time limit of 20 min". If it does finish quickly, record how long it takes and how many constraints were generated.
5. Now start the formulation with 1-city, 2-city, 3-city, and 4-city subtour elimination constraints. Use your Xpress code to solve the problem. If it runs for more than 20 minitues stop Xpress and report "Reach time limit of 20 min". If it does finish quickly, record how long it takes and how many constraints were generated.
6. For each of (i)-(v), how many subtour elimination constraints did the formulation begin with?
7. Plot the optimal TSP tour you found, using the Matlab code US48TourPlot.m. You can use V-Lab to access Matlab. You do not need to do any Matlab programming to run this code. Download US48TourPlot.m to the same folder as your other .mos files. Open the .m file in an Matlab editor