Traveling sales man problem

-solved using integer programming-

# Problem description

The Traveling Salesman Problem (TSP) is a classic problem in combinatorial optimization. It was first formulated as an integer program by Dantzig, Fulkerson and Johnson in 1954.

In this example, we consider a salesman traveling. The salesman starts in a city and has to visit a set of cities on a business trip before returning home. The problem then consists of finding the shortest tour which visits every city on the itinerary.

We will formulate this problem as an integer program and implement it in Cplex. The implementation will also demonstrate the use of lazy constraints in Cplex.

# Mathematical Model

We let the n selected cities in the salesman's tour be the set of vertices V of a graph. The set of edges E of the graph corresponds to the different connections between each city. Since we can travel from any city to another, the graph is complete. That is, there is an edge between every pair of nodes. For each edge in the the graph we associate a binary variable

Since the edges are undirected, we have that and it suffices to only include edges with in the model.

We want to minimize the total distance travelled during the tour. Therefore, we calculate the distance between each pair of nodes i and j. The total distance travelled is then the sum of the distances of the edges included in the tour

The tour should only pass through each city once. Therefore, each node in the graph should have exactly one incoming edge and one outgoing edge. In other words, for every node i exactly two of the binary variables should be equal to 1. We write this constraint as:

This constraint means that the saleman should enter and leave each city exactly once

With just this constraint on the number of edges in the tour entering and exit each node, we may produce solutions that are not connected tours. For example, the figure below shows a graph with 6 nodes and two disconnected subtours. The first subtour goes through nodes 1, 2, and 3, while the second subtour goes through nodes 4, 5, 6, and 7. Note that in this solution each node has exactly two edges in the tour incident to it. But there is no path for the salesman to travel between subtour. So we must add extra constraints to our model to eliminate these solutions.



To eliminate the subtours we add the following constraints:

These constraints require that for each proper (nonempty) subset S of the set of cities V, the number of edges between the nodes of S must be at most .

Indeed if the number of edges were equal to S, it would be possible to form a subtour. For example, in the figure above the subset of nodes S={1,2,3} has 3 edges in the tour: . So

Thus, the subtour elimination constraint above is violated.

So finally the integer program formulation becomes

If the set of cities V is of size n, there are subsets S of V, excluding S=V and S=∅. Instead of explicitally including a constraint.

for each S. We include above constraints implicitly as lazy constraints. That is, we generate and add these constraints to our model in a lazy fashion.

Initially we will have no subtour elimination constraints in our model. When Cplex finds a feasible solution that satisifies the other constraints, we compute the shortest cycle in edges included in the tour so far. If this cycle is of length n, then the model is solved. Otherwise, a cycle of length m<n defines a subtour, and a set S with |S|=m. We then add the corresponding constraint to eliminate this subtour as a lazy constraint, and Cplex continues solving this new modified model. This process continues until the shortest cycle is of length n, implying that all subtour elimination constraints have been satisfied.