Programming Exercise for Mathematical Programming 186.835

Benjamin Schwendinger
e1225371

May 5, 2019

1 k-Node Minimum Spanning Tree (k-MST) Problem

Given:

- graph $G = (V, E, w)$
- edge weights $w(e) \in \mathbb{R}^+_\nu, \forall e \in E$
- integer $0 \leq k \leq |V|$

Goal: Find a minimum weight tree, spanning exactly k nodes.

I use a directed graph to model the problem. Hence for each edge in E, I use two directed arcs instead. Moreover, the weight of each arc is the same as the corresponding weight of the undirected graph. Furthermore a root node (I call it 0) and edges from 0 to each other node with weight 0 are added. I write the set of neighbors of a vertex v as $N(v)$.

2 Formulations

2.1 Variables:

2.1.1 Binary variables:

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is in the tree} \\ 0 & \text{otherwise} \end{cases}$$

$$y_a = \begin{cases} 1 & \text{if arc } a \text{ is in the tree} \\ 0 & \text{otherwise} \end{cases}$$

$$z_v = \begin{cases} 1 & \text{if vertex } v \text{ is in the tree} \\ 0 & \text{otherwise} \end{cases}$$

2.1.2 Integer variables:

order variables u_i are used to indicate the order in which the nodes are explored (start and end node is 0)
flow variables $0 \leq f_{ij} \leq n - 1$; amount of flow on arc $(i, j) \in A$

2.2 MTZ

$$\min_{e \in E} c_e x_e \tag{1a}$$

$$\text{subject to} \sum_{a \in A} y_a = k + 1 \tag{1b}$$

$$\sum_{v \in V} z_v = k + 1 \tag{1c}$$

$$\sum_{j \in N(i)} y_{ij} \leq 1 \qquad \forall v \in V \tag{1d}$$

$$\sum_{i \in N(0)} y_{i0} = \sum_{i \in N(0)} y_{0i} = 1 \tag{1e}$$

$$y_{ij} \leq z_i, z_j \qquad \forall i, j \in V \tag{1f}$$

$$y_{ij} + y_{ji} = x_e \qquad \forall e = \{i, j\}, e \in E \tag{1g}$$

$$u_i + y_{ij} \leq u_j + (n-2)(1 - y_{ij}) \qquad \forall i, j \in V \setminus \{0\}, i \neq j \tag{1h}$$

$$1 \leq u_i \leq n - 1 \qquad \forall i \in V \setminus \{0\} \tag{1i}$$

$$z_v \in \{0, 1\} \qquad \forall v \in V \tag{1j}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{1k}$$

$$y_a \in \{0, 1\} \qquad \forall a \in A \tag{1l}$$

2.3 SCF

$$\min_{e \in E} c_e x_e \tag{2a}$$

$$\text{subject to} \sum_{a \in A} y_a = k + 1 \tag{2b}$$

$$\sum_{v \in V} z_v = k + 1 \tag{2c}$$

$$\sum_{j \in N(i)} y_{ij} \leq 1 \qquad \forall v \in V \tag{2d}$$

$$\sum_{i \in N(0)} y_{i0} = \sum_{i \in N(0)} y_{0i} = 1 \tag{2e}$$

$$y_{ij} \leq z_i, z_j \qquad \forall i, j \in V \tag{2f}$$

$$y_{ij} + y_{ji} = x_e \qquad \forall e = \{i, j\}, e \in E \tag{2g}$$

$$\sum_{j \in N(i)} f_{ij} - \sum_{j \in N(i)} f_{ji} = y_i \qquad \forall i \in V \setminus \{0\} \tag{2h}$$

$$0 \leq f_{ij} \leq (n-1) y_{ij} \qquad \forall (i, j) \in A \tag{2i}$$

$$z_v \in \{0, 1\} \qquad \forall v \in V \tag{2j}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{2k}$$

$$y_a \in \{0, 1\} \qquad \forall a \in A \tag{2l}$$

The objective is modelled by Equation (a) which states the weight of the found tree has to be minimal.
The constraints (b) to (f) are the same for both formulations.

(b) Our found directed graphs has to have $k+1$ arcs. Hence I get a tree on by deleting the arcs from and to 0.

(c) Our found tree has to have $k+1$ nodes. This is true, since I am searching for a tree on k for graph without our added root node.

(d) The number of ingoing arcs to every node is less or equal 1. This enforces that the found graph is a forest.

(e) The number of ingoing and outgoing arcs from 0 is 1. This enforces that really $k-1$ edges of the input graph are used and not only our added zero weight edges from and to 0.

(f) An arc can only be in the tree if both ends of it are in the tree.

(g) An edge is in the graph if one and at most one of the corresponding arcs is in the directed graph.

The constraints $(1h)$ to $(1i)$ model the MTZ constraints.
The constraints $(2h)$ to $(2i)$ model the SCF constraints. Here a node only consumes one if it is in the tree.
The constraints j to k set the domains of my binary variables.

3 Computational Results

The implementations are done in Python and I use Gurobi 8 to solve our ILP approach. All tests were performed on a single core of an Intel Xeon E5540 processor with 2.53 GHz using at most 3GB RAM.

According to Table 1 and Figure 1 I see that MTZ has to explore more nodes for bigger k ($k \geq 100$). The running times show similiar results according to Table 1 and Figure 2, although it seems that MTZ is favorable for smaller k. This is probably due to the high number of nodes which are explored for bigger k. Moreover, we see that all optimal values are reached with both developed formulations.

Inputs		MTZ			SCF		
Instance	k	Objective	Nodes explored	Time(s)	Objective	Nodes explored	Time(s)
1	2	46	0	0.078	46	0	0.036
1	5	477	0	0.033	477	16	0.043
2	4	373	19	0.128	373	89	0.115
2	10	1390	42	0.087	1390	0	0.155
3	10	725	258	0.221	725	459	0.418
3	25	3074	3687	0.581	3074	72	0.495
4	14	909	1465	0.498	909	1257	1.469
4	35	3292	1859	1.615	3292	2750	1.552
5	20	1235	1669	1.645	1235	1767	1.998
5	50	4898	3348	1.891	4898	4709	3.432
6	40	2068	17 374	14.874	2068	17 679	45.048
6	100	6705	19 768	19.761	6705	7934	27.087
7	60	1335	1631	32.169	1335	1743	56.405
7	150	4534	3337	33.055	4534	4548	108.332
8	80	1620	3138	36.994	1620	1805	115.925
8	200	5787	146 821	414.960	5787	3872	193.722

Table 1: Result table for different formulations

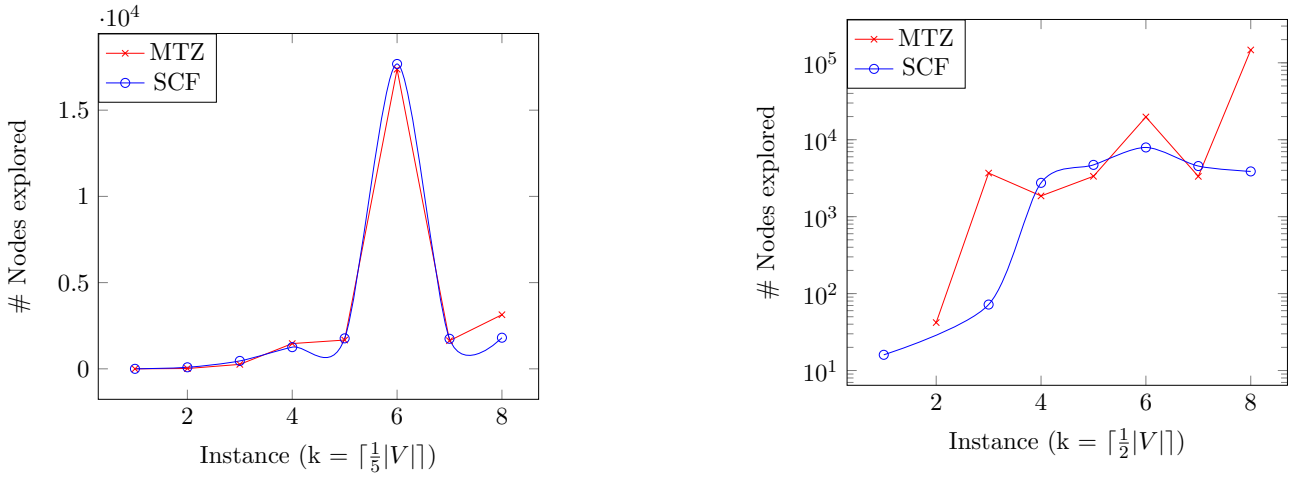

Figure 1: Number of explored nodes for different formulations

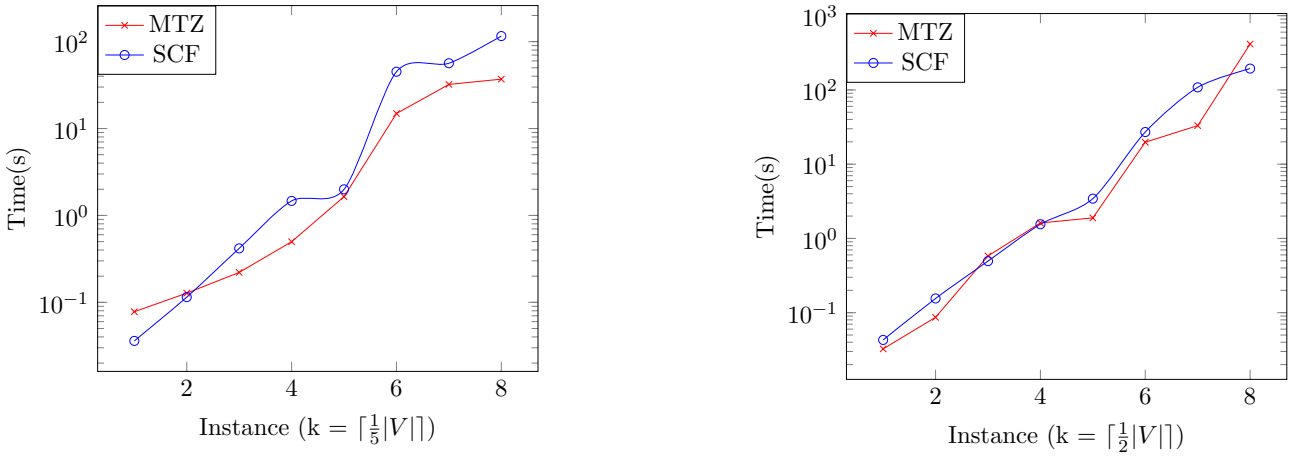

Figure 2: Running times for different formulations