Programming Exercise 2

e11921655 Fabian Holzberger

July 12, 2021

Introduction

For this Project we compare the single commodity flow formulation (SCF-formulation) , multi commodity flow formulation (MCF-formulation) and the Miller Tuckker Zemlin formulation (MTZ-formulation) of the k-minimum spanning tree problem (k-mst) solved by the IBM Cplex solver. The tests are performed on a Virtual Machine in a Lubuntu20.10-Linux distribution with Intel-i5-4300M 2.6GHz CPU. We investigate 10 different instances for which we solve the k-mst with the time restriction of 1 hour and memory-restriction of 6GB.

SCF-Formulation

Model

In the directed SCF-formulation we deal with arc variables $x_{i,j}$ defined by (9), flow variables $f_{i,j}$ defined on each arc by (10) and binary node variables z_i defined by (11) on each node. The directed SCF-formulation is given as:

$$\min \sum_{\{i,j\}\in E} (x_{i,j}+x_{j,i})c_{i,j} \tag{1}$$

$$\sum_{(i,j)\in\delta^+(0)} x_{i,j} = 1 \tag{2}$$

$$\sum_{(i,j)\in\delta^+(0)} f_{i,j} = k \tag{3}$$

$$\sum_{(i,j)\in\delta^-(0)} x_{i,j} = 0 \tag{4}$$

$$\sum_{(i,j)\in\delta^-(0)} f_{i,j} = 0 \tag{5}$$

$$f_{i,j} \leq k x_{i,j} \quad \forall (i,j)\in A \tag{6}$$

$$\sum_{j\in N} z_j = k+1 \tag{7}$$

$$\sum_{(i,j)\in\delta^-(l)} f_{i,j} - \sum_{(i,j)\in\delta^+(l)} f_{i,j} = z_l \quad l \in N\setminus\{0\} \tag{8}$$

$$x_{i,j}\in\mathbb{B} \quad \forall (i,j)\in A \tag{9}$$

$$f_{i,j}\in\mathbb{N}\cap[0,k] \quad \forall (i,j)\in A \tag{10}$$

$$z_i\in\mathbb{B} \quad \forall i\in N \tag{11}$$

$$\tag{12}$$

Here in (1) the cost of all arcs is summed, where by optimality the formulation only allows to select one arc at each undirected edge. Next in (2) we set the number of outgoing arcs from the dummy root node to exactly 1 and similarly in constraint (3) the outflow from the dummy root to exactly k. Since we don't want an ingoing edge into the dummy root and further no flow back into the root we state these restrictions in the constraints (4) and (5). Constraint (6) states that whenever we have a nonzero flow we must select the corresponding arc for that flow. Any selected node consumes one unit of flow, such that the sum of the outgoing flow variables is one less than the sum of the ingoing flow variables, which is formulated in (8). Note that the latter constraint is not formulated for the root node since there all incoming flows are 0.

Results on Instances

instance	k	total_nodes	solution_status	b&b_nodes	best_obj	cpu_time [s]	optimality_gap [%]
1	2	10	Optimal	0	46	0	0.0
1	5	10	Optimal	0	477	0	0.0
2	4	20	Optimal	81	373	0	0.0
2	10	20	Optimal	0	1390	0	0.0
3	10	50	Optimal	2013	725	1	0.0
3	25	50	Optimal	0	3074	1	0.0
4	14	70	Optimal	1008	909	1	0.0
4	35	70	Optimal	504	3292	2	0.0
5	20	100	Optimal	810	1235	3	0.0
5	50	100	Optimal	512	4898	3	0.0
6	40	200	Optimal	3678	2068	47	0.0
6	100	200	Optimal	622	6705	18	0.0
7	60	300	Optimal	596	1335	26	0.0
7	150	300	Optimal	833	4534	37	0.0
8	80	400	Optimal	4314	1620	119	0.0
8	200	400	Optimal	723	5787	47	0.0
9	200	1000	Optimal	832	2289	2648	0.0
9	500	1000	Optimal	1010	7595	608	0.0
10	400	2000	Feasible	80	10526	TL	100.0
10	1000	2000	Feasible	970	16085	TL	7.2

Discussion of Results

Except for the last instance, we are able to obtain optimal solutions for the SCF-formulation. Compared to the MTZ-formulation we need less b&b nodes, but more than for the MCF-formulation, meaning that it is a stronger formulation than SCF but weaker than MCF. Performance wise it is in second place, because of its large optimality gaps for the last instance, that are not present in the MTZ-fomulation, but can solve more instances than MCF-formulation.

MCF-Formulation

Model

We formulate a direct MCF-formulation. It uses the binary arc variables $x_{i,j}$ defined in (21), binary node variables z_i defined by (23), where we exclude the root node from these variables and the positive flow variables

$f_{i,j}^{\alpha}$ for each arc and node (exept the dummy root) that are defined in (22).

$$\min \sum_{\{i,j\} \in E} (x_{i,j} + x_{j,i}) c_{i,j} \tag{13}$$

$$\sum_{(i,j) \in \delta^+(0)} x_{i,j} = 1 \tag{14}$$

$$\sum_{(i,j) \in \delta^+(0)} f_{i,j}^{\alpha} = z_{\alpha} \quad \forall \alpha \in N \setminus \{0\} \tag{15}$$

$$\sum_{(i,j) \in \delta^-(0)} x_{i,j} = 0 \tag{16}$$

$$\sum_{(i,j) \in \delta^-(0)} f_{i,j}^{\alpha} = 0 \quad \forall \alpha \in N \setminus \{0\} \tag{17}$$

$$f_{i,j}^{\alpha} \leq x_{i,j} \quad \forall (i,j) \in A \; \forall \alpha \in N \setminus \{0\} \tag{18}$$

$$\sum_{j \in N \setminus \{0\}} z_j = k \tag{19}$$

$$\sum_{(i,j) \in \delta^-(l)} f_{i,j}^{\alpha} - \sum_{(i,j) \in \delta^+(l)} f_{i,j}^{\alpha} = \begin{cases} z_l & \text{if } \alpha = l \neq 0 \\ 0 & \text{if } \alpha \neq l \quad \forall \alpha, l \in N \setminus \{0\} \end{cases} \tag{20}$$

$$x_{i,j} \in \mathbb{B} \quad \forall (i,j) \in A \tag{21}$$

$$f_{i,j}^{\alpha} \in \mathbb{N} \cap [0,1] \quad \forall (i,j) \in A \; \forall \alpha \in N \setminus \{0\} \tag{22}$$

$$z_i \in \mathbb{B} \quad \forall i \in N \setminus \{0\} \tag{23}$$

$$\tag{24}$$

As in the SCF-formulation we select exactly one outgoing arc from the root and no ingoing arc by the constraints (14) and (16). For this formulation we can have for each node a different commodity of flow and therefore we only need to send out one unit of a commodity from the root, that corresponds to a selected node, as can be seen in constraint (15). Again we don't want any commodity of flow back into the root, which is stated in (17). By (18) we select an arc if it has a nonzero flow in any commodity. Since we don't include the root in the node variables we only need to select k of them in (19). Constraint (20) is stating that when we select a node variable, that node consumes its assigned commodity by none of the other commodities.

Results on Instances

instance	k	total_nodes	solution_status	b&b_nodes	best_obj	cpu_time [s]	optimality_gap [%]
1	2	10	Optimal	0	46	0	0.0
1	5	10	Optimal	0	477	0	0.0
2	4	20	Optimal	0	373	0	0.0
2	10	20	Optimal	0	1390	0	0.0
3	10	50	Optimal	0	725	1	0.0
3	25	50	Optimal	0	3074	1	0.0
4	14	70	Optimal	76	909	5	0.0
4	35	70	Optimal	0	3292	4	0.0
5	20	100	Optimal	0	1235	11	0.0
5	50	100	Optimal	0	4898	17	0.0
6	40	200	Optimal	164	2068	558	0.0
6	100	200	Optimal	0	6705	342	0.0

Discussion of Results

The MCF-formulation can't solve all instances with the present machine-restrictions. This is due to the fact that it creates the most constraints of the three investigated formulations and therefore tops out from lack of memory when solving the instances 7 to 10 (for example when allowing 8GB memory instance 7 can be solved). On the other hand it is the strongest formulation, since it often doesn't require b&b nodes, which the other two formulations mostly do. This shows that the MCF-formulation's polyhedra might be very close tho the convex hull of the considered problem. Since we can't attempt to solve instances higher than 6 it is performance wise in last place of the considered formulations.

MTZ-Formulation

Model

For a directed MTZ-formulation we have binary arc-variables $x_{i,j}$, defnined in (32), binary node-variables z_i, defined in (33) and bfs-like level-variables f_i in (34), that are positive integers and mark the level of the tree starting with 0 at the dummy root (35). Our MTZ-formulation is given by:

$$\min \sum_{\{i,j\}\in E} (x_{i,j} + x_{j,i})c_{i,j} \tag{25}$$

$$\sum_{(i,j)\in\delta^+(0)} x_{i,j} = 1 \tag{26}$$

$$\sum_{(i,j)\in\delta^-(0)} x_{i,j} = 0 \tag{27}$$

$$\sum_{j\in N} z_j = k \tag{28}$$

$$f_i + x_{i,j} \leq f_j + k(1 - x_{i,j}) \quad \forall (i,j) \in A \tag{29}$$

$$\sum_{(i,j)\in\delta^-(l)} x_{i,j} = z_l \quad \forall l \in N \tag{30}$$

$$\sum_{(i,j)\in\delta^+(l)} x_{i,j} \leq (k-1)z_l \quad \forall l \in N \setminus \{0\} \tag{31}$$

$$x_{i,j} \in \mathbb{B} \quad \forall (i,j) \in A \tag{32}$$

$$z_i \in \mathbb{B} \quad \forall i \in N \tag{33}$$

$$f_i \in \mathbb{N} \cap [1,k] \quad \forall i \in N \setminus \{0\} \tag{34}$$

$$f_0 = 0 \tag{35}$$

Since we need to deals with the dummy root, we select by (25) the number of outgoing arcs from the root to exactly 1 and in (26) the number of ingoing root-arcs to 0. By (28) we select exacly k nodes, note that the other constraints force that the dummy root is not selected. In (29) the MTZ constraint forces in every selected arc the level variable to increase by 1 when going from the start to the end-node of the arc. Next the arc and node variables are connected by (30), which tells us that for every selected node there must be an ingoing arc and (31), that describes that there can only be an outgoing arc at some node when that node is selected. Note that the latter constraint is not formulated for the dummy root, which enables solvability.

Results on Instances

instance	k	total_nodes	solution_status	b&b_nodes	best_obj	cpu_time [s]	optimality_gap [%]
1	2	10	Optimal	0	46	0	0.00
1	5	10	Optimal	0	477	0	0.00
2	4	20	Optimal	7	373	0	0.00
2	10	20	Optimal	13	1390	0	0.00
3	10	50	Optimal	16	725	0	0.00
3	25	50	Optimal	2317	3074	2	0.00
4	14	70	Optimal	19	909	0	0.00
4	35	70	Optimal	857	3292	1	0.00
5	20	100	Optimal	440	1235	1	0.00
5	50	100	Optimal	2671	4898	4	0.00
6	40	200	Optimal	14399	2068	75	0.00
6	100	200	Optimal	10354	6705	48	0.00
7	60	300	Optimal	496	1335	7	0.00
7	150	300	Optimal	2133	4534	15	0.00
8	80	400	Optimal	280	1620	8	0.00
8	200	400	Optimal	18830	5787	343	0.00
9	200	1000	Feasible	110907	2289	TL	0.31
9	500	1000	Feasible	46082	7595	TL	0.35
10	400	2000	Optimal	30947	4182	1819	0.00
10	1000	2000	Feasible	22496	14991	TL	0.26

Discussion of Results

The MTZ-formaulation is the only formulation that could obtain optimal solutions in the given amount of time. Note that for some of the larger instances the optimality gap is not yet zero but the best solution is already optimal. Next we observe that the MTZ-formulation crates the most b&b nodes of all three formulations showing that it is the weakest. Further, it uses the least amount of variables of the three presented formulations. Performance wise it is the best with respect to the given machine requirements, since it can solve all instances to optimality.

Introduction

For this second Project we compare the cycle elimination cut formulation (CEC-formulation) and the directed cut formulation (DCC-formulation) of the k-minimum spanning tree problem (k-mst) solved by the IBM Cplex solver. The tests are performed on a Virtual Machine in a Lubuntu20.10-Linux distribution with Intel-i5-4300M 2.6GHz CPU. We investigate 10 different instances for which we solve the k-mst with the time restriction of 1 hour and memory-restriction of 6GB.

Basis-Formulation

Our CEC and DCC formulations are build on top of a common set of inequalities. Important is that this common set has a polynomial number of inequalities and is extended in the solution preocess by violated CEC or DCC inequalities. The Basis-Formulation is given by:

$$\min \sum_{\{i,j\} \in E} (x_{i,j} + x_{j,i}) c_{i,j} \tag{36}$$

$$\sum_{(i,j) \in \delta^+(0)} x_{i,j} = 1 \tag{37}$$

$$\sum_{(i,j) \in \delta^-(0)} x_{i,j} = 0 \tag{38}$$

$$x_{i,j} + x_{j,i} \leq 1 \quad \forall (i,j) \in A \tag{39}$$

$$\sum_{j \in N \setminus \{0\}} z_j = k \tag{40}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{i,j} = z_j \quad \forall j \in V \tag{41}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{i,j} \leq z_i(k-1) \quad \forall j \in V \setminus \{0\} \tag{42}$$

$$\sum_{(i,j) \in A} x_{i,j} = k \tag{43}$$

$$x_{i,j} \in \mathbb{B} \quad \forall (i,j) \in A \tag{44}$$

$$z_i \in \mathbb{B} \quad \forall i \in N \tag{45}$$

$$z_0 = 0 \tag{46}$$

In this formulation we use binary arc variables $x_{i,j}$ and binary vertex variables zi. The objective (36) is to minimize the costs $c_{i,j}$ for selected edges $x_{i,j}$ that we convert in our formulation into two directed arcs $x_{i,j}, x_{j,i}$. By optimality only one arc is selected, but nevertheless we have added the strenthening inequlities (39) that cut off integer-solutions with bidirectional arcs. Since we add an artificial root node with index 0, we state in (37) and (38), that there is only one outgoing arc from the root and no arc back into the root. The next three inequalities are based on tree properties. (41) is stating that there is exactly one ingoing arc into any selected vertex. Inequality (42) ensures that whenever we select an arc that leads to some vertex, we also need to select that vertex. Finally in (44) we ensure that there are k arcs selected in a solution, as required for a tree. This formulation is not valid for the k-mst since it still allows solutions that contain circles with disconnected sub-graphs.

CEC-Formulation

Model

To prevent circle-formulation in solutions we forbid it in this formulation explicitely by adding the CEC's:

$$\sum_{(i,j) \in C} x_{i,j} \leq |C| - 1 \quad \forall C \in A,\ |C| \geq 2,\ C \text{ cycle} \tag{47}$$

$$\tag{48}$$

The above set of inequalities grows exponential when increasing the nuber of vertices. Therefore we dont add all of them at once and rather identify them on the run. Whenever we generate a solution we check for violated

CEC's by solving the separation problem by the algorithm 1. If no violated *CEC* is found we conclude that the solution is valid and optimal.

Algorithm 1: separation problem for CEC: find a cycle $C \subseteq A$ with $\sum_{(i,j)\in C}(1 - x_{i,j}) < 1$

Input: Instance Graph $G = (V, A)$, current LP solution (x, z)
Output: cycle C in the current LP solution that violates a CEC

/* find violated CEC by< formulating a shortest path problem */
$w_{i,j} := 1 - x_{i,j}$ // arc weights for shortest path problem
$C \leftarrow \emptyset$ // for returning violated cycles
$P \leftarrow \emptyset$ // for saving shortest paths
for *all* $(u, v) \in A$ **do**
  /* compute shortest path P from v to u w.r.t w */
  $P \leftarrow \text{shortestPath}(v, u)$
  if $w_{u,v} + \sum_{(i,j)\in P} w_{i,j} < 1$ **then**
    /* violated CEC found */
    $C \leftarrow P \cup (u, v)$
    break
return C

When execution the branch and cut approach we have the option to add more CEC's in the LP-nodes when we solve for the relaxation and the solution is not integral. It was obsered that this slowed down the solution process on the testinstances. We therefore only generate violated inequalities when integral solutions are found.

Results on Instances

inst	k	tot_nodes	sol_stat	b&b_nodes	best_obj	cpu_time [s]	opt_gap [%]	u-cuts	c-cuts
1	2	10	Optimal	0	46	0	0.00	0	0
1	5	10	Optimal	0	477	0	0.00	0	0
2	4	20	Optimal	7	373	0	0.00	0	5
2	10	20	Optimal	0	1390	0	0.00	0	1
3	10	50	Optimal	0	725	0	0.00	0	3
3	25	50	Optimal	362	3074	0	0.00	0	16
4	14	70	Optimal	33	909	0	0.00	0	6
4	35	70	Optimal	86	3292	0	0.00	0	16
5	20	100	Optimal	4	1235	0	0.00	0	4
5	50	100	Optimal	1963	4898	1	0.00	0	47
6	40	200	Optimal	2533	2068	6	0.00	0	60
6	100	200	Optimal	4379	6705	10	0.00	0	201
7	60	300	Optimal	187	1335	4	0.00	0	15
7	150	300	Optimal	742	4534	4	0.00	0	30
8	80	400	Optimal	121	1620	5	0.00	0	15
8	200	400	Optimal	8492	5787	62	0.00	0	268
9	200	1000	Optimal	40393	2289	749	0.00	0	268
9	500	1000	Feasible	12712	7595	TL	0.34	0	1744
10	400	2000	Optimal	18698	4182	3549	0.00	0	193
10	1000	2000	Feasible	7035	14991	TL	0.27	0	140

Discussion of Results

As can be seen in the table above the CEC formulation can obtain optimal solutions for all instances exept two. For the two instances were nonoptmal solutions were obtained we see that the optimality bound is less than 0.5 %. As we can see below, this bound can be improved by considering the DCC formulation.

DCC-Formulation

Model

Another way to ensure that we create valid solutions is to generate solutions where all selected vertices are connected to the root node by some arcs. This DCC's can be formulated as:

$$\sum_{(i,j)\in\delta^+(S)} x_{i,j} \geq z_k \quad \forall S \in V : 0 \in S \text{ and } k \in V \setminus S \tag{49}$$

This time the separation problem can be formulated as an instance of the maxflow problem. The separation problem is given in algorithm 2

Algorithm 2: separation problem for DCC: find a set $S \subset V : 0 \in S$ and $k \notin S$ with $\sum\limits_{(i,j)\in\delta^+(S)} x_{i,j} < z_k$

Input: Instance Graph $G = (V, A)$, current LP solution (x, z)
Output: set S_s such that $\delta^+(S_s)$ violates a DCC

/* find violated DCC by solving a maxplow problem */
$f_{i,j} := x_{i,j}$ // arc flows for maxflow problem
$s := 0$ // source vertex is the root
$(S_s, S_t) \leftarrow (\emptyset, \emptyset)$ // for returning violated cycles
$f_{max} \leftarrow 0$ // for returning violated cycles
for *all* $t = 1$ *to* $|V|$ **do**
  /* compute maxflow between the vertices s and t w.r.t f */
  $(S_s, S_t, f_{max}) \leftarrow \text{maxflow}(s, t)$ // $s \in S_s$ are the source-side nodes and $t \in S_t$ are the target-side nodes
  if $f_{max} < 1$ *and* $k \in S_t$ *and* $z_k = 1$ **then**
    /* violated DCC found */
    break
return (S_s, S_t, f_{max})

This time we also add DCC's for nodes in the branch and cut tree where we dont have intergal solutions. Important is that we decrease the ammount of added inequalities when we find fractional solutions till we have reached an lower limit where we only add one inequalitie when we find a fractional solution.

Results on Instances

inst	k	tot_nodes	sol_stat	b&b_nodes	best_obj	cpu_time [s]	opt_gap [%]	u-cuts	c-cuts
1	2	10	Optimal	0	46	0	0.00	0	0
1	5	10	Optimal	0	477	0	0.00	2	0
2	4	20	Optimal	0	373	0	0.00	4	5
2	10	20	Optimal	0	1390	0	0.00	5	2
3	10	50	Optimal	0	725	0	0.00	8	1
3	25	50	Optimal	0	3074	0	0.00	13	4
4	14	70	Optimal	36	909	0	0.00	13	6
4	35	70	Optimal	0	3292	0	0.00	12	5
5	20	100	Optimal	0	1235	0	0.00	14	3
5	50	100	Optimal	0	4898	0	0.00	27	8
6	40	200	Optimal	47	2068	1	0.00	50	10
6	100	200	Optimal	1802	6705	6	0.00	109	102
7	60	300	Optimal	0	1335	1	0.00	22	4
7	150	300	Optimal	921	4534	9	0.00	131	27
8	80	400	Optimal	23	1620	3	0.00	67	9
8	200	400	Optimal	2164	5787	15	0.00	103	75
9	200	1000	Optimal	144	2289	34	0.00	168	11
9	500	1000	Feasible	98367	7595	TL	0.13	1951	2233
10	400	2000	Optimal	4366	4182	560	0.00	359	61
10	1000	2000	Feasible	31979	14991	TL	0.19	2569	3418

Results on Instances

The DCC formulation can obtain optimal solutions for the same instances as the CEC formulation. In addition to that the instances that are not solved to optimality have a lower optimality gap, which is now below 0.2 %. We can solso clearly see that the solution times for the optimal solutions of the DCC formulation as significantly lower than for the CEC formulation and that often a low number of branch and bound nodes are needed.

Results Project1 and Project2

inst	k	SCF	MCF	MTZ	CEC	DCC
1	2	0	0	0	0	0
1	5	0	0	0	0	0
2	4	0	0	0	0	0
2	10	0	0	0	0	0
3	10	1	1	0	0	0
3	25	1	1	2	0	0
4	14	1	5	0	0	0
4	35	2	4	1	0	0
5	20	3	11	1	0	0
5	50	3	17	4	1	0
6	40	47	558	75	6	1
6	100	18	342	48	10	6
7	60	26		7	4	1
7	150	37		15	4	9
8	80	119		8	5	3
8	200	47		343	62	15
9	200	2648		TL	749	34
9	500	608		TL	TL	TL
10	400	TL		1819	3549	560
10	1000	TL		TL	TL	TL

Here we show the efficiency of all formulations in one table.

The DCC formulation had the overall lowest solution times but could not solve instance 9 with $k = 500$, which is solved only by the single commodity flow formulation to optimallity.The CEC formulation aswell has very low solution times but slows down when it comes to larger instances like instance 10 with $k = 400$. FOr the compact formulations the SCF solved the most instances followed by the MTZ formulation. The MCF formulation showed the worst peformance, since we could not even solve instances larger than 6 due to the vast amout of inequalities created and their required memory for the solution process.