Algorithms for the Maximum Weight Connected Subgraph and Prize-collecting Steiner Tree Problems

Ernst Althaus[1] and Markus Blumenstock[1]

[1]Institut für Informatik, Johannes Gutenberg-Universität, Mainz, Germany
ernst.althaus@uni-mainz.de
markusblumenstock@hotmail.com

Abstract. We present new exact and heuristic algorithms for the prize-collecting Steiner tree problem. The exact algorithm first reduces the size of the input graph while preserving equivalence of the optimal solutions and then uses mixed integer linear programming to solve the resulting instance. For our heuristic, we reduce the size of the instance graph further, but without guaranteeing the equivalence of the optimal solutions and then use the same integer linear programming based approach to solve the remaining instance.

Keywords: linear programming, heuristics, prize-collecting Steiner tree

1 Introduction

The Steiner tree problem and its variants are among the most investigated combinatorial optimization problems. Here we are primarily interested in the prize-collecting Steiner tree problem (PCST), i.e. we are given a graph $G = (V, E)$ with vertex weights $w : V \mapsto \mathbb{R}$ (the prizes) and non-negative edge weights $c : E \mapsto \mathbb{R}_{\geq 0}$ and we are supposed to compute a subset $U \subseteq V$ and a spanning tree T of the graph induced by U with minimal total weight $\sum_{u \in U} w(u) + \sum_{e \in T} c(e)$. This NP-complete problem has many applications, see e.g. Johnson et al. [JMP00]. In some variants, the vertices that are *not* selected incur a penalty.

Our original interest was in the Maximum-Weight Connected Subgraph problem (MWCS), which is a special case of the prize-collecting Steiner tree problem with all edge weights being zero, but most algorithms generalize to the general problem. Furthermore, in PCST, typically only a small fraction of the vertices have a weight different from zero, whereas in the MWCS almost all vertices have a non-zero weight. MWCS arises when trying to identify functional modules in protein-protein interaction networks (see Dittrich et al. [DKR+08]).

In this paper, we describe exact and heuristic algorithms for this problem. The exact algorithm is based on a mixed integer linear programming (MILP) formulation, originally developed for a k-cardinality variant of MWCS [ABD+14]. It uses the combination of two ILPs, the approach by Lucena and Resende [LR00],

which uses generalized subtour elimination constraints (GSec), and an adoption of the approach by Cohen [Coh10] for spanning tree problems. The ILP formulation by Chimani et al. [CKLM10] is equivalent to the Gsec formulation, and it allows a more efficient separation algorithm than that proposed by Fischetti et al. [FHJM94]. These ILPs are restated in Section 2.

To improve the efficiency of the ILP-based algorithm, we propose some techniques to reduce the size of the instance. Reductions play a very important role to solve large Steiner tree problems [Pol03]. We derive a sufficient criterion in Section 3.

If the input graph after these reductions is still too large to be solved with our ILP-based algorithm, we reduce it further, but without guaranteeing that the solution found is still optimal. This heuristic is explained in Section 4.

We evaluated our algorithms on the Mwcs instances of the 11th DIMACS Implementation Challenge[1] in Section 5 and give a conclusion. We also test the exact MILP formulation on the Steiner tree (Stp) and (rooted) prize-collecting Steiner tree (Pcst, Rpcst) instances of the competition.

In the following, we assume that the input graph is connected. Otherwise, we start our algorithm for every connected component and return the best solution.

2 An MILP Formulation for Pcst and Mwcs

2.1 Previous MILP Formulations

We propose a mixed integer linear programming formulation for the Mwcs problem based on a recent k-cardinality tree formulation by Althaus et al. [ABD+14]. This in turn is based on formulations by Fischetti et al. [FHJM94], Chimani et al. [CKLM10], and Cohen [Coh10]. Other formulations for k-cardinality trees or connected subgraphs are by Backes et al. [BRK+11] and Quintão et al. [QdCM08, QadCML10], the latter uses the Miller-Tucker-Zemlin constraints [MTZ60].

The formulation of Lucena and Resende [LR00] is based on generalized subtour elimination constraints (Gsec). More precisely, it uses binary variables y_u for the vertices $u \in V$ and b_e for the edges $e \in E$ in the undirected sense. The constraints are $\sum_{u \in V} y_u = \sum_{e \in E} x_e + 1$ and $\sum_{e \in \gamma(S)} x_e \leq \sum_{u \in S \setminus \{s\}} y_u$ for all subsets $S \subseteq V$ with $|S| \geq 2$ and all $s \in S$. The formulations for the k-cardinality tree problem by Fischetti et al., Chimani et al., and Ljubić [Lju04] use generalized subtour elimination or equivalent contraints.

The formulation by Cohen [Coh10] uses the fact that graphs with a maximum average degree of at most $2 - \frac{2}{|V|}$ are acyclic. Cohen proves that for a graph of maximum average degree z, we can distribute a value of 2 for each edge (the degree generated by it) to its endpoints, i.e. define continuous edge flow values $f_{uv,u}$ and $f_{uv,v}$ with $f_{uv,u} + f_{uv,v} = 2$, such that the total amount assigned to a vertex is at most z, i.e. $\sum_{uv \in E} f_{uv,v} \leq z$ for all $v \in V$. Furthermore, this is not possible for any value z smaller than the maximum average degree.

[1] `http://dimacs11.cs.princeton.edu/`

The ILPs of Cohen and Althaus et al. can be adapted for the MWCS and PCST problems, which leads to the following formulation where linear objective functions can be added in a straightforward manner. We use a flow of one per selected edge in (3) instead of two since the model is linear:

Variables

$$\begin{aligned} y_v \in \{0, 1\} \quad & \forall\, v \in V \\ b_e \in \{0, 1\} \quad & \forall\, e \in E \\ f_{uv,u}, f_{uv,v} \in [0, 1] \quad & \forall\, uv \in E \end{aligned}$$

Constraints

$$\sum_{v \in V} y_v = \sum_{e \in E} b_e + 1 \tag{1}$$

$$b_{uv} \leq y_u, y_v \quad \forall\, uv \in E \tag{2}$$

$$f_{uv,u} + f_{uv,v} = b_{uv} \quad \forall\, uv \in E \tag{3}$$

$$\sum_{uv \in E} f_{uv,v} \leq 1 - \frac{1}{|V|} \quad \forall\, v \in V \tag{4}$$

Note that this formulation does not allow to select zero vertices because of (1). As in the paper of Althaus et al., we can strengthen the constraints (4) to

$$\sum_{uv \in E} f_{uv,v} \leq \left(1 - \frac{1}{|V|}\right) y_v \quad \forall\, v \in V. \tag{4a}$$

We can further add the constraint

$$f_{uv,v} \geq \frac{1}{|V|} b_{uv} \quad \forall\, v \in V \tag{5}$$

and the restriction $f_{uv,v} \in [0, 1 - \frac{1}{|V|}]$ to forbid more solutions of the relaxed model: if an edge is in the solution, it generates a flow of one, but its end vertices (which must also be selected by (2)) can each only take $(1 - \frac{1}{|V|})$ at most.

This formulation can be combined with GSEC or equivalent formulations to obtain a smaller polyhedron. Althaus et al. [ABD+14] showed that in the k-cardinality variant, the polyhedra are not comparable if $k \geq 2$.

The directed cut (DCUT) formulation by Chimani et al. [CKLM10] is favorable because there is a very efficient separation algorithm for it. The formulation solves a generalization of the problem to arborescences. In addition to vertex variables y_v, it uses directed edge variables $x_{u,v}$ for every $uv \in E$. An artificial root vertex $r \notin V$ is introduced which acts as the source with directed edges $x_{r,v} \in \{0, 1\}$ to every $v \in V$, emitting a total flow of exactly one. Every vertex selected for the solution must receive a flow of one. There are exponentially many cut constraints that ensure the connectivity,

$$\sum_{u \in S, v \in V \setminus S} x_{u,v} \geq y_v \quad \forall v \in S, \forall S \subseteq V. \tag{6}$$

For single-vertex sets, we can require (6) with equality, i.e. the ingoing flow is exactly one for every selected vertex.

If an edge variable is selected ($x_{u,v} = 1$), the edge in the opposite direction ($x_{v,u}$) cannot be selected, and a non-selected vertex cannot take more than zero flow. These constraints can be generated from the GSEC constraints for two-element sets, so they do not strengthen the formulation; however, Chimani et al. choose to include them from the beginning:

$$x_{u,v} + x_{v,u} \leq y_u \quad \forall uv \in E. \tag{7}$$

2.2 Combination and Strengthening of Existing ILP Approaches

We now discuss in detail how we combine the constraints and which improvements can be made. To combine the formulations, we set

$$x_{u,v} + x_{v,u} = b_{uv}, \tag{8}$$

and by (2), the constraint (7) is automatically fulfilled. Furthermore, it is now sufficient to restrict one of the opposite directed-edge variables $x_{u,v}$ to $\{0,1\}$, the other is immediately determined since b_{uv} is binary and can thus take continuous variables in the formulation.

We further note that the root variables $x_{r,v}$ can be relaxed as well. For every $v \in V$, the ingoing edges $x_{u,v}$ with $u \neq r$ take binary values and as stated in the previous section, the total ingoing flow is exactly one if the vertex is selected, zero otherwise. If the ingoing flow is zero, or it is one and the flow comes from a non-root vertex, the flow $x_{r,v}$ must be zero. Otherwise, we must have $x_{r,v} = 1$ since the root emits a total flow of one. Therefore, this set of $|V|$ variables does not need to be restricted to integer values.

We note that another constraint type can be added that strengthens both the Cohen and the DCUT formulation, but only if more than one vertex must be selected: In a tree with at least two vertices, every vertex has at least one adjacent edge. For our problems, the number of selected vertices is not restricted (as in the k-cardinality problem), but we can easily check single-vertex solutions and then proceed with the assumption that at least two vertices are selected. The constraints are

$$\sum_{uv \in E} b_{uv} \geq y_v \quad \forall v \in V, \tag{9}$$

and they do strengthen the relaxation, as can be seen in the example in Figure 1. A similar constraint was introduced by Lucena and Resende, but they only conjectured a strengthening of the formulation.

A question for future work is if some constraints of the Cohen formulation can be omitted when combined with DCUT, including this constraint.

The constraints of the Cohen formulation are less tight than in the k-cardinality variant, where (1) is split into two new constraints for k vertices and $k - 1$

edges, and the quantity $|V|$ can be replaced by k in the other constraints. Constraint (4a) can then be required with equality. In the general problem, the lower bound for the flow a vertex receives is zero since a single vertex can be selected. However, the constraints can be strengthened when k-cardinality solutions are checked separately for consecutive values $1, ..., k'$ and $k'', ..., |V|$, which can be done efficiently for small ranges, e.g. we can add the constraint

$$\sum_{uv \in E} f_{uv,v} \geq \left(1 - \frac{1}{k'}\right) y_v \quad \forall v \in V \tag{10}$$

and tighten the constraints (4a) and (5) with k''.

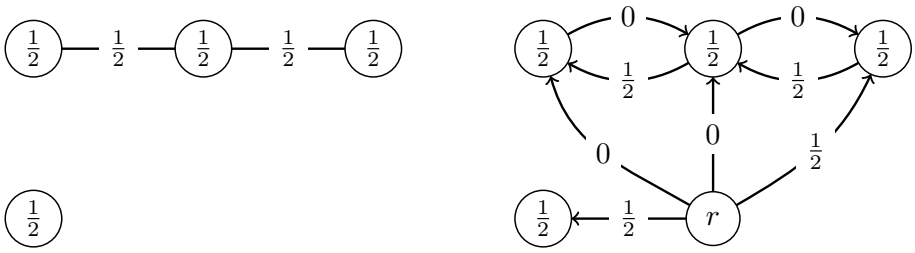

Fig. 1. A graph with four vertices that consists of a path of three vertices and an isolated vertex. For the assignment $y = (1/2, 1/2, 1/2, 1/2)$, GSEC (left) and the equivalent DCUT (right) LP solutions are shown for $k = 2$ selected vertices. These solutions are not feasible if constraint (9) is added to the model, because the isolated vertex does not have any incident edges (the artificial root edges in the DCUT formulation are not part of the projection in (8)).

2.3 Separation Algorithm

To find violated inequalities of the DCUT formulation in a branch-and-cut approach, we use an implementation of the Edmonds-Karp algorithm. Once a relaxed node in the MILP has been solved, the algorithm is used on a flow network with the edge variable assignments $x_{u,v}$ as capacities. Zero-capacity edges are dropped to keep the network sparse. There are |V| flow problems, one for each vertex as the target, while the artificial root vertex serves as the source. However, vertices with $y_v = 0$ can be skipped safely. The minimum cut corresponding to the maximum flow constitutes a violated inequality if its value is less than the value assigned to the target's variable in the MILP. However, we only added the most violated cuts among the flow networks, e.g. a single one or a very small number. Chimani et al. note that it is possible to extract several minimum cuts from a single flow problem, however, we did not implement this technique. Furthermore, there are faster algorithms for the maximum flow problem than the Edmonds-Karp algorithm, but since the lion share of the runtime goes to MILP solving for most instances, this is often negligible.

3 Reduction of the Input Graph

In this section, we present some methods to reduce the size of the input graph, so that it is easy to obtain the optimal solution of the original graph given the solution of the reduced graph. Some reduction methods are already given in [LR00] and not repeated here. Clearly, these reductions can be applied repeatedly until no further simplification is possible. Keep in mind that the objective in PCST is to minimize, which we will discuss in the following, while it is maximizing for MWCS.

3.1 Adjacent non-negative vertices

We can safely contract adjacent non-positive vertices if the edge between them is the cheapest among the edges adjacent to either vertex. The weight of the contracted vertex is the sum of the weights of the vertices that were contracted plus the cost of the edge between them.

The reason is that if one node is part of the optimal solution, the other is too and the edge between them is in the minimum spanning tree of the selected nodes.

3.2 Vertices of degree 1

The following test was already described in [LR00]. We add it, as we will refer to it later. We can remove a vertex u with degree 1 and $w(u) + c(uv) \geq 0$, where v is the neighbor of u, as it will never be contained in an optimal solution.

3.3 Vertices of degree 2

We can remove a vertex u of degree 2 if its profit $(-w(u))$ is less than the cost of the cheaper adjacent edge. If we do so, we have to add an edge between the two neighbors v and w of the node u of weight $c(vu) + c(uw) + w(u)$.

The reason is that the vertex u can not be of degree one in the optimal solution as it then could be removed. Hence u is only in the optimal solution if both edges are also in it. This generalizes a test given in [LR00].

3.4 Non-profitable unique path from a vertex of degree 1

A generalization of the "vertex of degree 1" test is as follows. Consider the unique path from a vertex of degree 1 to the next vertex that either has non-negative profit or a degree of at least three. If this path has non-negative total weight, all nodes on this path except the last one can be removed.

The reason is again, that any sub-path of the path can be removed from any solution so that the remaining edges still form a tree and have smaller cost.

Similarly, the "vertex of degree 2" test can be extended to longer paths.

3.5 Neighborhood Subsets of Neighbored Vertices

This is an extension of the "mirrored hubs" rule by El-Kebir and Klau [EKK14]: If two adjacent vertices have exactly the same neighbors[2], then the one with less (or equal) weight is always preferred and the other can be removed safely.

More generally, if the neighborhood of a vertex is a subset of a neighbor's neighborhood, then this vertex can be removed if its weight is greater or equal than the weight of the neighbor.

4 A Heuristic for the MWCS Problem

To reduce the search space of the MWCS problem, we use a heuristic to obtain a smaller graph (for the already reduced graph) on which the ILP is solved. We will call vertices with negative/nonnegative weights negative and nonnegative vertices for short, respectively.

To simplify the graph, we only keep nonnegative vertices and create edges among them by computing the shortest paths between these vertices: the weight of the edge uv is the total weight of a shortest (vertex-weight) path between u and v in the original graph. To keep the graph sparse, the shortest-path computation starting in u (a variant of the Dijkstra algorithm) does not go beyond a negative vertex, therefore usually less than $|V|(|V|-1)/2$ edges are present.

After the MWCS solution has been found, the edges are converted back to sets of negative vertices. It is then possible to remove cycles that can be present due to the fact that the shortest paths may not have been disjoint.

The heuristic solution that is found can be used as a starting point for the exact approach, which is used in the actual competition of the DIMACS Implementation Challenge.

5 Experiments

5.1 Benchmark Setting and Instances

The tests were carried out on an Intel Core i7 CPU @ 3.20GHz with 12GB DDR3-RAM. The algorithms were programmed in Java 7 [Ora12]. The mixed integer linear programs were solved using Gurobi 5.6 [Gur14], which is free for academic purposes. Gurobi was run with two threads.

The datasets we used were instances from the 11th DIMACS Implementation Challenge. We ran the exact and the heuristic approach on the MWCS datasets and report the best solutions found within one hour of computing time in Tables 1 and 2. We also ran the exact algorithm on PCST, SPG and RPCST test instances for half an hour each, the results can be seen in the appendix in Tables 3, 4 and 5.

[2] Here, we say that a vertex is a neighbor of itself.

5.2 Analysis of MWCS Results

On the `drosophila` datasets, which were the largest data sets by far (more than $5,000$ vertices and $90,000$ edges), our heuristic approach performed better than the exact approach. In the exact approach, the root relaxation could not be solved within one hour of computing time, which leaves us only with a solution found by some heuristic (e.g. by the MILP solver or a trivial solution such as a minimum spanning tree that is used as a starting point). On the smaller datasets, the exact algorithm produced better results.

The reductions that preserve optimality are able to decrease the graphs in size quite well. The neighborhood-subsets reduction (Subsection 3.5) yields a big improvement over the special case described by El-Kebir and Klau [EKK14] on the `drosophila` datasets (Table 1). The `metabol_expr_mice` datasets do not benefit from the generalization.

Table 1. Results of the Cohen-DCUT intersection on the MWCS dataset after reduction with one hour of computation per instance, rounded to two decimal places. Small connected components in the `metabol_expr_mice` datasets are not included in the runtime. The "mirrored hubs" (MH) and "neighborhood subsets" (NS) rules yield vastly different results for some instances, other reductions being equal.

	Input graph		MH reduction		NS reduction		Result on NS-reduced instance			
Dataset	Vertices	Edges	Vertices	Edges	Vertices	Edges	LB	UB	Gap	Time in s
`drosophila001`	5226	93394	3872	68311	2857	45802	10.02	∞	–	3600.08
`drosophila005`	5226	93394	3856	66922	2802	43859	17.53	∞	–	3600.06
`drosophila0075`	5226	93394	3833	65547	2738	40017	44.56	∞	–	3600.09
`HCMV`	3863	29293	2963	24883	2659	21414	7.55	7.55	0.00%	3427.00
`lymphoma`	2034	7756	1544	7160	1461	6895	70.17	70.17	0.00%	321.64
`metabol_expr_mice_1`	3523	4345	1813	2741	1813	2741	544.75	587.05	7.76%	2847.07
`metabol_expr_mice_2`	3514	4332	1808	2738	1808	2738	241.07	283.20	17.47%	2844.97
`metabol_expr_mice_3`	2853	3335	1175	1796	1175	1796	508.26	527.74	3.83%	2766.48

6 Conclusion and Outlook

We present algorithms for the prize-collecting Steiner tree problem and the maximum-weight connected subgraph problem that are based upon several previous algorithms. We combine two non-comparable integer linear programming formulations to obtain tight bounds. Furthermore, we extend known reduction methods for the prize-collecting Steiner tree problem to reduce the size of the inputs, which turned out to be very effective on some MWCS instances. As a heuristic we propose to apply further reduction methods that do not necessarily preserve the optimality of the resulting solution.

Our experiments show that our algorithms are able to find solutions for all benchmark instances in limited time (an hour or less). For smaller instances, the exact method is superior, even if the branch-and-bound solver fails to prove

Table 2. Results for the heuristic with the Cohen-DCut intersection on the Mwcs dataset after full reduction with one hour of computation per instance, rounded to two decimal places. Small connected components in the `metabol_expr_mice` datasets are not included in the runtime.

	Path-graph of reduced graph					Cycles removed	
Dataset	Nonneg. vertices	Paths	LB	UB	Time in s	Solution	Gap to exact MILP
`drosophila001`	60	1770	13.47	13.47	0.75	14.62	73.07%*
`drosophila005`	130	8385	132.01	132.01	607.95	154.56	110.46%*
`drosophila0075`	152	11476	211.30	216.92	3600.07	239.52	788.50%*
`HCMV`	50	1225	5.17	5.17	0.46	5.50	37.27%
`lymphoma`	52	1326	45.53	45.53	1.63	49.43	41.95%
`metabol_expr_mice_1`	106	4308	421.19	421.19	93.72	460.63	27.45%
`metabol_expr_mice_2`	60	1713	223.03	223.03	0.43	233.79	21.13%
`metabol_expr_mice_3`	79	2709	417.26	417.26	3.56	443.24	19.06%

* This figure was computed by solving the exact MILP with the heuristic solution as a start value, which led to a feasible solution.

optimality of the best solution found. When instances get larger, the heuristic outperforms the exact approach, which sometimes needs more than an hour to solve the root relaxation.

References

[ABD+14] Ernst Althaus, Markus Blumenstock, Alexej Disterhoft, Andreas Hildebrandt, and Markus Krupp. Algorithms for the Maximum Weight Connected k-Induced Subgraph Problem. Accepted to The 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA'14), Maui, Hawaii, USA, 2014.

[BRK+11] Christina Backes, Alexander Rurainski, Gunnar W. Klau, Oliver Müller, Daniel Stöckel, Andreas Gerasch, Jan Küntzer, Daniela Maisel, Nicole Ludwig, Matthias Hein, Andreas Keller, Helmut Burtscher, Michael Kaufmann, Eckart Meese, and Hans-Peter Lenhof. An integer linear programming approach for finding deregulated subgraphs in regulatory networks. *Nucleic Acids Research*, 2011.

[CKLM10] Markus Chimani, Maria Kandyba, Ivana Ljubić, and Petra Mutzel. Obtaining Optimal k-Cardinality Trees Fast. *J. Exp. Algorithmics*, 14:5:2.5–5:2.23, January 2010.

[Coh10] Nathann Cohen. Several Graph Problems and their LP formulation (explanatory supplement for the Sage graph library). http://hal.inria.fr/inria-00504914, July 2010.

[DKR+08] Marcus T. Dittrich, Gunnar W. Klau, Andreas Rosenwald, Thomas Dandekar, and Tobias Müller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):223–231, 2008.

[EKK14] Mohammed El-Kebir and Gunnar W. Klau. Solving the Maximum-Weight Connected Subgraph Problem to Optimality. arXiv:1409.5308, 2014.

[FHJM94] Matteo Fischetti, Horst W. Hamacher, Kurt Jørnsten, and Francesco Maffioli. Weighted k-Cardinality Trees: Complexity and Polyhedral Structure. *Networks*, 24(1):11–21, 1994.

[Gur14] Gurobi Optimization, Inc. Gurobi Optimizer Reference Manual. http://www.gurobi.com/documentation/5.0/reference-manual/, 2014.

[JMP00] David S. Johnson, Maria Minkoff, and Steven Phillips. The prize collecting Steiner tree problem: theory and practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 760–769, 2000.

[Lju04] Ivana Ljubić. *Exact and memetic algorithms for two network design problems.* PhD thesis, Technische Universität Wien, 2004. https://www.ads.tuwien.ac.at/publications/bib/pdf/ljubicPhD.pdf.

[LR00] Abilio Lucena and Mauricio G.C. Resende. Strong lower bounds for the prize collecting steiner problem in graphs. Technical report, DISCRETE APPLIED MATHEMATICS, 2000.

[MTZ60] Clair E. Miller, Albert W. Tucker, and Richard A. Zemlin. Integer Programming Formulation of Traveling Salesman Problems. *J. ACM*, 7(4):326–329, October 1960.

[Ora12] Oracle corporation. Java Platform, Standard Edition 7. http://docs.oracle.com/javase/7/docs/api/, 2012.

[Pol03] Tobias Polzin. *Algorithms for the Steiner Problem in Networks.* Doctoral dissertation, Universität des Saarlandes, May 2003.

[QadCML10] Frederico P. Quintão, Alexandre Salles da Cunha, Geraldo R. Mateus, and Abilio Lucena. The k-Cardinality Tree Problem: Reformulations and Lagrangian Relaxation. *Discrete Appl. Math.*, 158(12):1305–1314, June 2010.

[QdCM08] Frederico P. Quintão, Alexandre Salles da Cunha, and Geraldo Robson Mateus. Integer Programming Formulations for the k-Cardinality Tree Problem. *Electronic Notes in Discrete Mathematics*, 30:225–230, 2008.

A Appendix

A.1 Steiner Tree Problem

Table 3. Results for the Cohen-DCut intersection on Stp instances within half an hour of computation time, rounded to two decimal places.

Dataset	LB	UB (primal)	Gap in %	Time in s
d18	222.00	223.00	0.5	1800
e18	$-\infty$	24977.00	–	1801
i640-211	11723.00	12066.00	2.9	1800
i640-314	34906.00	35783.00	2.5	1800
i640-341	$-\infty$	149895.00	–	1801
fnl4461fst	148172.00	459882.00	210.4	1801
alue7080	48312.00	383286.00	693.4	1803
alut2625	$-\infty$	451526.00	–	1802
es10000fst01	592276215.00	1442738340.00	143.6	1804
lin36	8225.00	471879.00	5637.1	1828
lin37	25569.00	2009807.00	7760.3	2349
hc12p	$-\infty$	448002.00	–	1801
hc12u	$-\infty$	4095.00	–	1801
cc12-2p	$-\infty$	861854.00	–	1801
cc12-2u	$-\infty$	8361.00	–	1801
cc12-2n	$-\infty$	4095.00	–	1801
cc3-12n	98.00	125.00	27.6	1801
cc3-12p	17363.00	356058.00	1950.7	1801
cc3-12u	171.00	3453.00	1919.3	1801
bipa2p	34666.00	37369.00	7.8	1800
bipa2u	330.00	348.00	5.5	1800
2r211c	74865.00	109000.00	45.6	1800
wrp3-83	8207387.42	12902865.00	57.2	1801
w23c23	684.00	695.00	1.6	1803
rc09	61001.00	187344.00	207.1	1804
rt05	21598.00	3082060.00	Large	1942
G106ac	27340466.80	151634246.00	454.6	1960
I064ac	181509367.00	408901305.00	125.3	1802
s5	$-\infty$	36414.00	–	1804

A.2 Prize-collecting Steiner Tree Problem

Table 4. Results for the Cohen-DCut intersection on Pcstp instances within half an hour of computation time, rounded to two decimal places.

Dataset	LB	UB (primal)	Gap in %	Time in s
P400_3	5125476.00	2951725.00	73.6	1282
P400_4	4962336.00	2852956.00	73.9	353
K400_7	343269.01	492017.00	43.3	1805
K400_10	448964.00	405031.00	10.8	1745
cc12-2nu	$-\infty$	697.00	–	1801
i640-001	2932.00	2932.00	0.0	5
HCMV	7385.23	7371.54	0.2	1765
metabol_expr_mice_1	11901.88	11523.81	3.3	1544
C13-A	236.00	236.00	0.0	42
C19-B	146.00	146.00	0.0	127
D03-B	1509.00	1509.00	0.0	707
D20-A	536.00	536.00	0.0	321
hc10p	58887.00	60948.00	3.5	1801
hc11u	$-\infty$	1553.00	–	1800
hc12p	$-\infty$	308227.00	–	1800
hc12u	$-\infty$	3083.00	–	1800
bip52nu	220.00	224.00	1.8	1800
bip62nu	211.00	215.00	1.9	1800
cc3-12nu	$-\infty$	114.00	–	1801
i640-221	8269.00	8468.00	2.4	1805
i640-321	28604.00	29173.00	2.0	1805
i640-341	$-\infty$	95072.00	–	1801
a2000RandGraph_2	1483.77	1483.84	0.0	1366
a4000RandGraph_3	3406.30	3406.62	0.0	1643
a8000RandGraph_1_2	$-\infty$	4791.46	–	1806
a14000RandGraph_1_5	$-\infty$	10514.84	–	1803
handsd04	457.06	776.88	70.0	1802
handbd13	$-\infty$	13.24	–	1812
handsi03	$-\infty$	56.28	–	1802
handbi07	$-\infty$	151.07	–	1813
drosophila001	$-\infty$	8302.58	–	1802
lymphoma	3341.89	3341.89	0.0	870

A.3 Rooted Prize-collecting Steiner Tree Problem

Table 5. Results for the Cohen-DCut intersection on Rpcstp instances within half an hour of computation time, rounded to two decimal places.

Dataset	LB	UB (primal)	Gap in %	Time in s
i101M1	109271.50	109271.50	0.0	26
i101M2	236099.40	357341.64	51.4	1800
i101M3	244382.26	487898.93	99.6	1800
i102M1	104065.80	104065.80	0.0	31
i102M2	279347.59	357301.20	27.9	1800
i102M3	322883.35	1017848.42	215.2	1800
i103M1	118266.86	139749.41	18.2	1800
i103M2	337434.32	408966.24	21.2	1801
i103M3	348410.16	1106229.91	217.5	1800
i104M2	48585.88	89965.85	85.2	1800
i104M3	50994.38	98175.00	92.5	1800
i105M1	26717.20	26717.20	0.0	15
i105M2	41799.04	106889.22	155.7	1800
i105M3	42174.73	112205.66	166.0	1800
i201M2	272926.55	359774.09	31.8	1801
i201M3	453440.05	839473.04	85.1	1835
i201M4	531041.33	1719427.92	223.8	1801
i202M2	190789.49	298360.57	56.4	1801
i202M3	240626.84	1207417.48	401.8	1801
i202M4	240124.16	2033361.12	746.8	1801
i203M2	341459.03	1055699.26	209.2	1801
i203M3	445193.62	1817008.86	308.1	1801
i203M4	451333.83	3227562.06	615.1	1800
i204M2	86368.29	161700.54	87.2	1801
i204M3	104610.57	333652.54	218.9	1800
i204M4	104042.82	808502.72	677.1	1800
i205M2	460709.72	620194.17	34.6	1800
i205M3	534972.03	1447119.71	170.5	1801
i205M4	540902.89	3100970.80	473.3	1800