Heuristic Cut Separation in a Branch&Cut Approach for the Bounded Diameter Minimum Spanning Tree Problem

Martin Gruber and Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Austria
{gruber|raidl}@ads.tuwien.ac.at

Abstract

The bounded diameter minimum spanning tree problem is an NP-hard combinatorial optimization problem arising for example in network design when quality of service is of concern. We solve a strong integer linear programming formulation based on so-called jump cuts by a novel Branch&Cut algorithm, using various heuristics including tabu search to solve the separation problem.

1. Introduction

The bounded diameter minimum spanning tree problem (BDMST) is a combinatorial optimization problem appearing in applications such as wire-based communication network design when certain aspects of quality of service have to be considered (limit interfering hops a signal has to pass between any two nodes in a network), in ad-hoc wireless networks, and in the areas of data compression and distributed mutual exclusion algorithms [12, 1].

More formally, the BDMST problem can be stated as follows: Given an undirected, connected graph $G = (V, E)$ with node set V and edge set E and associated costs $c_e \geq 0$, $\forall e \in E$, the goal is to determine a spanning tree $T = (V, E_T)$ with edge set $E_T \subseteq E$ whose diameter – the maximum number of edges between any two nodes – does not exceed a given upper bound $D \geq 2$, and whose total costs $\sum_{e \in E_T} c_e$ are minimal. This problem is known to be NP-hard for $4 \leq D < n - 1$ [5].

Here we present a new Branch&Cut approach, in which a hierarchy of heuristics is used for efficiently separating so-called jump cuts.

2. Previous Work

The algorithms published for this problem range from greedy construction heuristics, e.g. [10], to various exact (mixed) integer linear programming (ILP) approaches. The latter include tight multi-commodity hop-indexed network flow models [5], formulations based on Miller-Tucker-Zemlin inequalities [4], and a compact Branch&Cut approach [7] utilizing cycle elimination constraints which was the basis for this work. Due to the complexity of the problem, exact algorithms are limited to relatively small instances with considerably less than 100 nodes when dealing with complete graphs. For larger instances, metaheuristics have been designed, for example evolutionary algorithms (EAs) [11] and a variable neighborhood search (VNS) [8]. The so far leading metaheuristics to address instances up to 1000 nodes are to the best of our knowledge the EA and ant colony optimization algorithm (ACO) from [9], which are based on a special level encoding of solutions and strong local improvement procedures.

Several publications address the related hop constrained MST (HCMST) problem where the root of the tree is fixed in advance, see for example [2, 3]. A well working approach for smaller hop limits is the reformulation of the problem as a Steiner tree problem in a layered graph [6].

3. The ILP Model

Our ILP model is defined on a directed graph $G^+ = (V^+, A^+)$, with the arc set A^+ being derived from E by including for each undirected edge $(u, v) \in E$ two oppositely directed arcs (u, v) and (v, u) with the same costs $c_{u,v} = c_{v,u}$. In addition, we introduce an artificial root node r that is connected to every other node with zero costs, i.e. $V^+ = V \cup \{r\}$ and $\{(r, v) \mid v \in V\} \subset A^+$. This artificial root allows us to model the BDMST problem as a directed outgoing HCMST problem on G^+ with root r, hop limit (i.e., maximum height) $H = \lfloor \frac{D}{2} \rfloor + 1$, and the additional constraint that the artificial root must have exactly one outgoing arc in the case of even D, and two outgoing arcs in the case D is odd. From a feasible HCMST $T^+ = (V^+, A_T^+)$, the corresponding BDMST T on G is derived by choosing all edges for which a corresponding arc is contained in the T^+. In the odd diameter case, an additional *center edge* connecting the two nodes adjacent to the artificial root must further be included.

Our ILP makes use of the following variables: Depth variables $y_{v,l} \in \{0,1\}$, $\forall v \in V$, $\forall l \in \{1,\ldots,H\}$, where $y_{v,l}$ is set to 1 iff node v appears at depth l in the HCMST T^+, arc variables $x_{u,v} \in \{0,1\}$, $\forall (u,v) \in A^+$, where $x_{u,v}$ is 1 iff $(u,v) \in T^+$, and center edge variables $z_{u,v} \in \{0,1\}$, $\forall (u,v) \in E$, which are only relevant for the odd diameter case and $z_{u,v}$ is 1 iff (u,v) forms the center of the BDMST. We can now formulate the ILP as follows:

$$\text{minimize} \sum_{(u,v)\in A} c_{u,v} \cdot x_{u,v} + \sum_{(u,v)\in E} c_{u,v} \cdot z_{u,v} \qquad (1)$$

$$\begin{aligned}
\text{s.t.} \quad \sum_{l=1}^{H} y_{v,l} &= 1 \quad \forall v \in V && (2)\\
\sum_{v\in V} y_{v,1} &= (D \bmod 2) + 1 && (3)\\
x_{r,v} &= y_{v,1} \quad \forall v \in V && (4)\\
\sum_{u|(u,v)\in A^+} x_{u,v} &= 1 \quad \forall v \in V && (5)\\
x_{u,v} &\leq 1 - y_{v,l} + y_{u,l-1} && (6)\\
&\quad \forall (u,v) \in A,\ \forall l = 2,\ldots,H &&\\
\sum_{v|(u,v)\in E} z_{u,v} &= y_{u,1} \quad \forall u \in V,\ \text{for odd } D && (7)
\end{aligned}$$

The objective is to minimize the total costs of all arcs in T^+ plus the costs of the BDMST's center edge in the odd diameter case (1). Each node $v \in V$ is assigned to exactly one depth $\in \{1,\ldots,H\}$ according to (2). Equation (3) ensures the correct number of nodes having depth one, i.e. forming the BDMST's center. Note that variables $x_{r,v}$ and $y_{v,1}$ express the same fact namely node v belongs to the center. Therefore, they are equal (4), and in our implementation only one set of these variables is actually used. Here, we keep both for notational convenience. Every node except r has exactly one predecessor (5), and a node u can only be the predecessor of a node v if u's depth is exactly one less than v's depth (6).

Strengthening Inequalities

The model presented so far already correctly describes the BDMST problem. Its linear programming (LP) relaxation can, however, substantially be strengthened by including further classes of valid inequalities. At first place, we consider the widely used directed connection cuts

$$\sum_{(u,v)\in\delta^+(V')} x_{u,v} \geq 1 \qquad \forall V' \subset V^+ \mid r \in V'. \qquad (8)$$

In a Branch&Cut approach they can be efficiently separated by max-flow/min-cut computations.

In [2] Dahl et al. proposed a Relax&Cut approach for the HCMST based on so-called jump inequalities. We adopt them to further strengthen our ILP.

We consider a partitioning of V^+ into pairwise disjoint nonempty sets S_0 to S_{H+1}, with $r \in S_0$. Let $\sigma(v)$ denote the index of the partition a node v is assigned to. Then $J(P)$ is defined as the set of arcs $(u,v) \in A^+$ with $\sigma(u) < \sigma(v) - 1$. The jump inequality associated with this partitioning states that in a feasible HCMST T^+ at least one of the arcs in $J(P)$ must appear in T^+. Otherwise, there would be a path connecting nodes from S_0 to S_{H+1} of length $H+1$ violating the hop constraint. Considering all possible partitionings $P(V^+)$ of V^+, we can write:

$$\sum_{(u,v)\in J(P)} x_{u,v} \geq 1 \qquad \forall P \in P(V^+) \mid r \in S_0. \qquad (9)$$

4. Jump Cut Separation

Our main focus now is to separate (find) a jump inequality violated by the current solution to the LP relaxation of our ILP. Thus, we seek a partitioning P of all nodes into sets $S_0, \ldots, S_{H+1}$ such that the LP values $x^{\mathrm{LP}}_{u,v}$ of all arc variables in $J(P)$ sum up to a value less than 1. Dahl et al. [2] utilized the jump formulation within a Relax&Cut approach where violated jump inequalities only need to be separated in integer solutions, which is straightforward. Here we must solve this separation problem on fractional solutions, which has been conjectured to be hard [2].

In a first attempt we formulated this subproblem as an ILP, but the required computation time prohibits this approach for practical use. One key issue is that the size of the separation problem increases quickly: more and more variables of the LP solution have values greater than 0 when jump cuts are added consecutively to the model.

Nevertheless, this approach gives an indication for the achievable strengthening of the model helping in the evaluation of the following heuristics.

4.1. Heuristics

To separate jump cuts we use a hierarchy of heuristics: A construction heuristic to find a first partitioning which is improved by local search, and – in case this was not successful – a tabu search procedure.

Construction Heuristic

Let $A^{\mathrm{LP}} = \{(u,v) \in A^+ \mid x^{\mathrm{LP}}_{u,v} > 0\}$. To avoid that an arc $(u,v) \in A^{\mathrm{LP}}$ becomes part of $J(P)$, $\sigma(u) \geq \sigma(v) - 1$ must hold in the partitioning P. Our heuristic iterates through all arcs in A^{LP} in decreasing LP value order and checks for each arc whether or not its associated constraint on the partitioning can be realized, i.e. if it is compatible with previously accepted ones. Compatible arcs are collected within a *constraint graph* $G_C = (V^+, A_C)$, while arcs raising contradictions w.r.t. previously accepted arcs stored in G_C will

Algorithm 1: Jump Cuts: Construction Heuristic

```
input  : $V^+, A^{\mathrm{LP}}$
output: partitioning $P$ of $V^+$
1  sort $A^{\mathrm{LP}}$ according to decreasing LP values;
2  forall nodes $v \in V$ do
3      all sets $S_i \leftarrow \emptyset$, except $S_0 \leftarrow \{r\}$, $S_{H+1} \leftarrow \{v\}$;
4      $b_r \leftarrow [0,0]$; $b_v \leftarrow [H+1, H+1]$;
5      forall $w \in V \setminus \{v\}$ do $b_w \leftarrow [1, H]$;
6      initialize $G_C$: $A_C \leftarrow \emptyset$;
7      initialize jump $J \leftarrow \emptyset$;
8      forall arcs $(u,v) \in A^{\mathrm{LP}}$ (sorted) do
9          if $A_C \cup (u,v)$ allows for a feasible assignment
           of all nodes then
10             $A_C \leftarrow A_C \cup (u,v)$;
11             perform recursive update;
12         else
13             $J \leftarrow J \cup (u,v)$;
14     assign nodes according to constraints in $G_C$;
15     evaluate partitioning and store it if best so far;
16 return best found partitioning;
```

be part of J. At the end, a partitioning P respecting all constraints associated with G_C is derived.

Note that only one node is assigned to set S_{H+1} as Dahl et al. proved that a cut is facet-defining iff the last set is singleton. See Algorithm 1 for a pseudo-code of this node partitioning heuristic.

In addition to the accepted arcs, G_C holds for each node $u \in V^+$ an interval $b_u = [\alpha_u, \beta_u]$, the feasible range of sets S_i for node u: $u \in S_i, i \in [\alpha_u, \beta_u]$. Inserting the arc (u,v) into G_C, the implied inequalities lead to the following updates of the lower and upper bounds at the involved nodes:

$$\begin{array}{ll} b_u\colon [\max(\alpha_u, \alpha_v - 1),\ \beta_u] & (\sigma(u) \geq \sigma(v) - 1), \\ b_v\colon [\alpha_v,\ \min(\beta_v, \beta_u + 1)] & (\sigma(v) \leq \sigma(u) + 1). \end{array}$$

An arc (u,v) can be added to the graph G_C without violating any stored constraints if the updates of the bounds at the nodes u and v do not cause an empty interval, i.e. $\alpha > \beta$. In case at least one interval is empty the inclusion of (u,v) would not allow for a feasible assignment of all nodes, therefore this arc is added to J. Otherwise, the arc is inserted into G_C, the bounds at u and v are updated, and these new bounds (if α and/or β were changed) need to be propagated through the graph according to the above rules.

The recursive update of G_C after inserting arc (u,v) cannot fail if it succeeded at nodes u and v. This can be shown as follows: Let G_C be valid, i.e. it contains no contradicting inequalities, and it was possible to insert a new arc (u,v) into the graph without leading to an empty range of potential sets at u and v. Let (s,t) be any other arc $\in G_C$, implying $\alpha_s \geq \alpha_t - 1$, and $\beta_t \leq \beta_s + 1$. Now let us assume that α_t was updated, i.e. increased, consistently to α'_t, $\alpha'_t \leq \beta_t$.

If the lower bound of s has to be modified, it is set to $\alpha'_s = \alpha'_t - 1$ according to the update rules. To prove that the interval at s will not become empty we have to show that $\alpha'_s \leq \beta_s$:

$$\alpha'_s \overset{\text{(update rule)}}{=} \alpha'_t - 1 \overset{\alpha'_t \leq \beta_t}{\leq} \beta_t - 1 \overset{\beta_t \leq \beta_s + 1}{\leq} \beta_s \quad \square$$

The propagation of the upper bound can be shown in an analogous way. Note that this also proves that the recursive update procedure terminates (no infinite loop) even when there are cycles in G_C (intervals cannot become empty, updates increase respectively decrease bounds by at least 1). It can also be shown easily that no set S_i will be empty when directed connection cuts are separated first.

Local Search

Although the construction heuristic already finds a lot of violated jump inequalities there is still room for improvement using local search. The neighborhood of a current partitioning P is in principle defined by moving one node to some new set S_i. As this neighborhood would be relatively large, we restrict it as follows: Each arc $(u,v) \in J$ induces two allowed moves: node u to set $S_{\sigma(v)-1}$ and v to set $S_{\sigma(u)+1}$. The local search is performed in a first improvement manner until a local optimum is reached.

Algorithm 2: Jump Cuts: Tabu Search

```
input  : $V^+, A^{\mathrm{LP}}$
output: (improved) partitioning $P$ of $V^+$
1  tabu list $L = ()$;
2  repeat
3      find best move $m$ removing an arc from $J$;
4      execute $m$ and update $J$;
5      file tabu move $m^{-1}$ in tabu list: $L = (m^{-1}) \oplus L$;
6      truncate $L$ to length $\max(l_{\min}, \gamma \cdot |J|)$;
7  until stopping criterion met ;
```

4.2. Tabu Search

The described heuristics already perform well, but statistics using the exact separation approach show that there are still undiscovered violated jump inequalities in the LP solutions. Therefore, if the construction heuristic and the local search fail to identify a jump cut we give a tabu search implementation a try, see Algorithm 2.

The neighborhood structure as well as the valid moves are defined as in the local search, but now a best improvement strategy is applied. Having performed a move of node v, we file as tabu the node v together with the direction (towards set S_{H+1} or S_0, respectively) it came from. The tabu

Table 1. Jump cut separation statistics.

ILP **M**odel from [7], **E**xact separation, **C**onstruction heuristic, **L**ocal search, **T**abu search.

Instance		M	E		C	CL	CLT	
	$\|V\|/\|E\|/D$	t[s]	%	t[s]	%	%	%	t[s]
TE	30 / 200 / 6	$>$1h	99.8	$>$1h	96.0	98.9	99.5	27.8
TE	30 / 200 / 7	2961.8	99.6	$>$1h	92.3	95.7	99.1	45.1
TR	30 / 200 / 6	17.6	98.6	1786.1	98.4	98.4	98.4	1.8
Santos	25 / 300 / 6	44.4	98.6	640.1	95.4	95.4	95.4	2.7
Santos	25 / 300 / 9	108.8	90.0	$>$1h	74.8	76.9	80.4	6.1
Santos	40 / 100 / 6	11.5	95.6	1279.2	94.1	94.1	94.1	0.6

tenure is controlled by the length of the tabu list L, which depends directly on the number of arcs in the current jump J: L is limited to $\max(l_{\min}, \gamma \cdot |J|)$ stored moves, where $l_{\min}$ and γ are strategy parameters; the oldest entries are removed. We use the standard aspiration criteria where a tabu restriction is ignored in case the move would lead to the so far best node partitioning. Tabu search terminates when a predefined number of iterations without improvement of the overall best partitioning is reached ($i_{\max}$).

5. Results

In Table 1 the success rate (%) in separating violated jump inequalities for an LP solution and some runtimes (t[s]) are listed for a few representative benchmark instances described and used by Gouveia et al. in [5] and Santos et al. [4]. Tabu search was performed with the following strategy parameters: $l_{\min} = 5$, $\gamma = 0.1$, $i_{\max} = 100$.

As can be seen, the exact (**E**) approach almost always identifies a jump cut, and the heuristics, namely construction heuristic (**C**), local search (**L**), and tabu search (**T**), are close in general. The biggest differences between the heuristics can be observed when the cut separation procedure is called for more than 1000 times (e.g. TE-30/200/6, or Santos-25/300/9).

To give an impression about the overall improvement in runtime: The Branch&Cut ILP approach proposed in [7] (**M**), with comparable runtime results to [5] and [4], requires on an AMD Opteron 250 server about 3 hours to solve a set of 50 benchmark instances (1 hour time limit for each instance). Using exact jump cut separation the computation time increases to more than 14 hours, whereas with heuristic jump cuts all instances of the benchmark set can be solved to proven optimality within 5 minutes.

6. Conclusions

Based on the jump formulation for the HCMST by Dahl et al. we developed various heuristics to solve the jump cut separation problem within a Branch&Cut framework for the BDMST problem. Obtained results document that runtimes to gain optimal solutions are substantially reduced. Further investigations on larger instances can now be done.

References

[1] A. Bookstein and S. T. Klein. Compression of correlated bit-vectors. *Information Systems*, 16(4):387–400, 1991.

[2] G. Dahl, T. Flatberg, N. Foldnes, and L. Gouveia. Hop-constrained spanning trees: the jump formulation and a relax-and-cut method. Technical report, University of Oslo, Centre of Mathematics for Applications (CMA), 2005.

[3] G. Dahl, L. Gouveia, and C. Requejo. On formulations and methods for the hop-constrained minimum spanning tree problem. In *Handbook of Optimization in Telecommunications*, chapter 19, pages 493–515. Springer Science + Business Media, 2006.

[4] A. C. dos Santos, A. Lucena, and C. C. Ribeiro. Solving diameter constrained minimum spanning tree problems in dense graphs. In *Proc. of the Int. Workshop on Experimental Algorithms*, volume 3059 of *LNCS*, pages 458–467. Springer, 2004.

[5] L. Gouveia and T. L. Magnanti. Network flow models for designing diameter-constrained minimum spanning and Steiner trees. *Networks*, 41(3):159–173, 2003.

[6] L. Gouveia, L. Simonetti, and E. Uchoa. Modelling the hop-constrained minimum spanning tree problem over a layered graph. In *Proc. of the Int. Network Optimization Conference*, Spa, Belgium, 2007.

[7] M. Gruber and G. Raidl. A new 0–1 ILP approach for the bounded diameter minimum spanning tree problem. In L. Gouveia and C. Mourão, editors, *Proc. of the Int. Network Optimization Conference*, volume 1, pages 178–185, Lisbon, Portugal, 2005.

[8] M. Gruber and G. R. Raidl. Variable neighborhood search for the bounded diameter minimum spanning tree problem. In P. Hansen et al., editors, *Proc. of the 18th Mini Euro Conference on Variable Neighborhood Search*, Tenerife, Spain, 2005.

[9] M. Gruber, J. van Hemert, and G. R. Raidl. Neighborhood searches for the bounded diameter minimum spanning tree problem embedded in a VNS, EA, and ACO. In M. Keijzer et al., editors, *Proc. of the Genetic and Evolutionary Computation Conference 2006*, volume 2, pages 1187–1194, 2006.

[10] B. A. Julstrom. Greedy heuristics for the bounded-diameter minimum spanning tree problem. Technical report, St. Cloud State University, 2004. Submitted for publication in the ACM Journal of Experimental Algorithmics.

[11] G. R. Raidl and B. A. Julstrom. Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In G. Lamont et al., editors, *Proc. of the 2003 ACM Symposium on Applied Computing*, pages 747–752. ACM Press, 2003.

[12] K. A. Woolston and S. L. Albin. The design of centralized networks with reliability and availability constraints. *Computers and Operations Research*, 15(3):207–217, 1988.