(Meta-)Heuristic Separation of Jump Cuts for the Bounded Diameter Minimum Spanning Tree Problem

Martin Gruber and Günther R. Raidl

Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, 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 inequalities by a Branch&Cut algorithm. As the separation subproblem of identifying currently violated jump inequalities is difficult, we approach it heuristically by two alternative construction heuristics, local search, and optionally tabu search. The overall algorithm performs excellently, and we were able to obtain proven optimal solutions for some test instances that were too large to be solved so far.

1 Introduction

The *bounded diameter minimum spanning tree* (BDMST) problem is a combinatorial optimization problem appearing in applications such as wire-based communication network design when quality of service is of concern and, for example, a signal between any two nodes in the network should not pass more than a fixed number of routers. It also arises in ad-hoc wireless networks [1] and in the areas of data compression and distributed mutual exclusion algorithms [2, 3].

The goal is to identify a tree structure of minimum costs connecting all nodes of a network where the number of links between any two nodes is limited by a maximum diameter D. More formally, we are given an undirected connected graph $G = (V, E)$ with node set V and edge set E and associated costs $c_e \geq 0$, $\forall e \in E$. We seek a spanning tree $T = (V, E_T)$ with edge set $E_T \subseteq E$ whose diameter does not exceed $D \geq 2$, and whose total costs $\sum_{e \in E_T} c_e$ are minimal. This problem is known to be NP-hard for $4 \leq D < |V| - 1$ [4].

The algorithms already published for this problem range from greedy construction heuristics to various exact *(mixed) integer linear programming* (ILP) approaches. The latter include formulations based on Miller-Tucker-Zemlin inequalities [5], a compact Branch&Cut approach strengthened by cycle elimination cuts [6], and in particular hop-indexed multi-commodity network flow models [7, 8] whose linear programming (LP) relaxations yield tight bounds but which involve a huge number of variables. 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 [9] and a variable neighborhood search (VNS) [10]. The so far leading metaheuristics to address instances up to 1000 nodes are to our knowledge the evolutionary algorithm and the ant colony optimization from [11], which are based on a special level encoding of solutions and strong local improvement procedures.

Strongly related to the BDMST problem is the *hop constrained minimum spanning tree* (HCMST) problem, in which a root node is specified and the number of edges (hops) on each path from the root to some other node must not exceed a limit H. An overview on several ILP models and solution approaches for this problem can be found in [12]. A well working approach in particular for smaller H is the reformulation of the problem as a Steiner tree problem on a layered graph [13]. Another strong formulation is based on so-called *jump inequalities* [14]. Unfortunately, their number grows exponentially with $|V|$, and the problem of separating them in a cutting plane algorithm is conjectured to be NP-hard. Therefore, Dahl et al. [14] exploited them in a Relax&Cut algorithm where violated jump inequalities only need to be identified for integer solutions, which is straightforward.

In this work, we adopt the concept of jump inequalities to formulate a strong model for the BDMST problem, which we then solve by Branch&Cut. A hierarchy of two alternative construction heuristics, local search, and tabu search is used for efficiently separating jump cuts.

2 The Jump 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 special 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 diameter D, and two outgoing arcs in the case D is odd. From a feasible HCMST $T^+ = (V^+, A^+_T)$, the associated BDMST T on G is derived by choosing all edges for which a corresponding arc is contained in A^+_T. In the odd diameter case, an additional *center edge* connecting the two nodes adjacent to the artificial root is further included.

We use the following variables: Arc variables $x_{u,v} \in \{0, 1\}$, $\forall (u, v) \in A^+$, which are set to 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 are set to 1 iff (u, v) forms the center of the BDMST.

The even diameter case is formulated as follows:

$$\text{minimize} \quad \sum_{(u,v)\in A} c_{u,v} \cdot x_{u,v} \tag{1}$$

$$\text{subject to} \quad \sum_{u|(u,v)\in A^+} x_{u,v} = 1 \qquad \forall\, v \in V \tag{2}$$

$$\sum_{v\in V} x_{r,v} = 1 \tag{3}$$

$$\sum_{(u,v)\in\delta^+(V')} x_{u,v} \geq 1 \qquad \forall\, V' \subset V^+ \mid r \in V' \tag{4}$$

$$\sum_{(u,v)\in J(P)} x_{u,v} \geq 1 \qquad \forall\, P \in P(V^+) \mid r \in S_0. \tag{5}$$

The objective is to minimize the total costs of all selected arcs (1). All nodes of the original graph (without artificial root node r) have exactly one predecessor (2), and just one node is successor of r (3). To achieve a connected, cycle free solution we include the widely used directed connection cuts (4).

The diameter restriction is enforced by the jump inequalities (5) as follows. Consider a partitioning P of V^+ into $H+1$ 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. Jump $J(P)$ is defined as the set of arcs $(u,v) \in A^+$ with $\sigma(u) < \sigma(v) - 1$, i.e. $J(P)$ contains all arcs leading from a partition to a higher indexed one and skipping at least one in-between. The jump inequality associated with this partitioning states that in a feasible HCMST T^+ at least one of these arcs in $J(P)$ must appear. Otherwise, there would be a path connecting the root contained in S_0 to a node in S_{H+1} with length $H+1$ violating the hop constraint. Such jump inequalities must hold for all possible partitionings $P(V^+)$ of V^+.

The odd diameter case additionally makes use of the center edge variables $z_{u,v}$:

$$\text{minimize} \quad \sum_{(u,v)\in A} c_{u,v} \cdot x_{u,v} + \sum_{(u,v)\in E} c_{u,v} \cdot z_{u,v} \tag{6}$$

$$\text{subject to} \quad \sum_{v\in V} x_{r,v} = 2 \tag{7}$$

$$\sum_{v|(u,v)\in E} z_{u,v} = x_{r,u} \qquad \forall\, u \in V \tag{8}$$

$$\frac{1}{2} \cdot \sum_{v\in V'} x_{r,v} + \sum_{(u,v)\in\delta^+(V-V')} x_{u,v} + \frac{1}{2} \cdot \sum_{(u,v)\in\delta(V')} z_{u,v} \geq 1 \qquad \forall\, \emptyset \neq V' \subset V \tag{9}$$

(2), (4), and (5) are adopted unchanged.

Now, two nodes are connected to the artificial root node r (7), and they are interlinked via the center edge (8). The costs of this edge are also accounted for in the new objective function (6). The special directed connection inequalities (9) are not necessary for the validity of the model but strengthen it considerably. They are essentially derived from observations in [8].

As there are exponentially many directed connection inequalities (4, 9) and jump inequalities (5), directly solving these models is not a practical option. Instead, we start without these inequalities and apply Branch&Cut, thus, separating inequalities that are violated by optimal LP solutions on the fly. While directed connection cuts – including our special variants (9) – can efficiently be separated by series of max-flow/min-cut computations, this subproblem unfortunately is conjectured to be NP-hard for the jump inequalities [14].

3 Jump Cut Separation

We have to identify a node partitioning P and corresponding jump $J(P)$ for which the current LP solution $(x^{\mathrm{LP}}, z^{\mathrm{LP}})$ violates $\sum_{(u,v)\in J(P)} x^{\mathrm{LP}}_{u,v} \geq 1$.

3.1 Exact Separation Model

In a first attempt we formulate the separation problem as an ILP, making use of the following variables: $y_{v,i} \in \{0,1\}$, $\forall v \in V^+,\ i = 0,\dots,H+1$, is set to value 1 iff node v is assigned to partition S_i, and $x_{u,v} \in \{0,1\}$, $\forall (u,v) \in A^{\mathrm{LP}}$, set to 1 iff arc (u,v) is contained in the jump $J(P)$, with $A^{\mathrm{LP}} = \{(u,v) \in A^+ \mid x^{\mathrm{LP}}_{u,v} > 0\}$. This leads to the following model:

$$\text{minimize} \qquad \sum_{(u,v)\in J(P)} x^{\mathrm{LP}}_{u,v} \cdot x_{u,v} \tag{10}$$

$$\text{subject to} \qquad \sum_{i=1}^{H+1} y_{v,i} = 1 \qquad \forall\ v \in V \tag{11}$$

$$y_{r,0} = 1 \tag{12}$$

$$\sum_{v\in V} y_{v,H+1} = 1 \tag{13}$$

$$y_{u,i} - 1 + \sum_{j=i+2}^{H+1} y_{v,j} \leq x_{u,v} \qquad \forall\ i \in \{1,\dots,H-1\},\ (u,v) \in A^{\mathrm{LP}} \tag{14}$$

$$\sum_{i=2}^{H+1} y_{v,i} \leq x_{r,v} \qquad \forall v \in V \mid (r,v) \in A^{\mathrm{LP}} \tag{15}$$

The objective is to minimize the total weight of the arcs in the jump $J(P)$ (10). Each node in V is assigned to exactly one of the sets S_1 to S_{H+1} (11), whereas the artificial root r is the only node in set S_0 (12). Exactly one node is assigned to set S_{H+1} (13), as Dahl et al. [14] showed that a jump inequality is facet-defining iff the last set is singleton. Finally, an arc (u,v) (14), respectively (r,v) (15), is part of the jump $J(P)$ iff it leads from a set S_i to a set S_j with $j \geq i+2$. Note that it is not necessary to explicitly address the condition that no partition may be empty, as this is implied when directed connection cuts are separated in advance and the objective value (10) is less than one, the case we are interested in. This model contains $O(|V| \cdot H + |A^{\mathrm{LP}}|)$ variables and $O(|V| + |A^{\mathrm{LP}}| \cdot H)$ constraints.

Solving this ILP by a general purpose solver each time when a jump cut should be separated is, however, only applicable for small problem instances as the computation times are high and increase dramatically with the problem size. According to our experiments, between about 85% and almost 100% of the total time for solving the BDMST problem is spent in this exact separation procedure for jump cuts.

To speed up computation we developed heuristic procedures for this separation problem and apply them in a hierarchical fashion: Two alternative construction heuristics are used to find initial partitionings; they are improved by local search and – in case a violated jump inequality has not yet been encountered – finally by tabu search.

3.2 Simple Construction Heuristic C^A

Heuristic C^A greedily assigns the nodes V^+ to sets $S_1, \ldots, S_{H+1}$ trying to keep the *number* of arcs that become part of the jump $J(P)$ as small as possible, see Algorithm 1. An independent partitioning is computed for each node $v \in V$ placed in the last set S_{H+1}, and the overall best solution is returned. To derive one such partitioning, all nodes u connected to r via an arc $(r, u) \in A^{\mathrm{LP}}$ with $x^{\mathrm{LP}}_{r,u}$ exceeding a certain threshold (0.5 in our experiments) are assigned to set S_1. Then the algorithm iterates through partitions S_{H+1} down to S_3. For each of these sets S_i all arcs $(w, u) \in A^{\mathrm{LP}}$ with target node $u \in S_i$ are further examined. In case w is still *free* (i.e., not already assigned to a set), it is placed in S_{i-1}, in

Algorithm 1: Simple Construction Heuristic C^A

```
   input  : $V^+, A^{\mathrm{LP}}$
   output: partitioning $P$ of $V^+$
1  forall nodes $v \in V$ do
2      $S_0 \leftarrow \{r\}$; $S_{H+1} \leftarrow \{v\}$; $\forall i = 1, \ldots, H:\ S_i \leftarrow \emptyset$;
3      forall arcs $(r, u) \mid u \neq v$ do
4          if $x^{\mathrm{LP}}_{r,u} > 0.5$ then $S_1 \leftarrow S_1 \cup \{u\}$;
5      for $i = H+1, \ldots, 3$ do
6          foreach node $u \in S_i$ do
7              foreach arc $(w, u) \in A^{\mathrm{LP}} \mid w$ not already assigned do
8                  $S_{i-1} \leftarrow S_{i-1} \cup \{w\}$;
9      forall still unassigned nodes $u \in V^+$ do
10         $S_1 \leftarrow S_1 \cup \{u\}$;
11     evaluate partitioning and store it if best so far;
12 return best found partitioning;
```

order to avoid (w, u) becoming part of $J(P)$. At the end, eventually remaining free nodes are assigned to set S_1.

Results achieved with heuristic C^A were encouraging, but also left room for improvement when compared to the exact separation. In particular, this heuristic does (almost) not consider differences in arc weights $x^{\mathrm{LP}}_{u,v}$ when deciding upon the assignment of nodes.

3.3 Constraint Graph Based Construction Heuristic C^B

To exploit arc weights in a better way, we developed the more sophisticated construction heuristic C^B which makes use of an additional *constraint graph* $G_C = (V^+, A_C)$. To avoid that an arc $(u, v) \in A^{\mathrm{LP}}$ becomes part of $J(P)$, the constraint $\sigma(u) \geq \sigma(v) - 1$ must hold in partitioning P. Heuristic C^B iterates through all arcs in A^{LP} in decreasing weight order (ties are broken arbitrarily) 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 arcs and their induced constraints. Compatible arcs are accepted and collected within the constraint graph, while arcs raising contradictions w.r.t. previously accepted arcs in G_C are rejected and will be part of $J(P)$. After checking each arc in this way, a partitioning P respecting all constraints represented by G_C is derived. Algorithm 2 shows this heuristic in pseudo-code.

In more detail, graph G_C not only holds compatible arcs but for each node $u \in V^+$ also an integer *assignment interval* $b_u = [\alpha_u, \beta_u]$ indicating the feasible range of partitions, i.e. u may be assigned to one of the sets $\{S_i \mid i = \alpha_u, \ldots, \beta_u\}$.

Algorithm 2: Constraint Graph Based Construction Heuristic C^B

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

When an arc (u, v) is inserted into A_C, the implied constraint $\sigma(u) \geq \sigma(v) - 1$ makes the following interval updates necessary:

$$b_u \leftarrow [\max(\alpha_u, \alpha_v - 1),\ \beta_u] \quad \text{and} \quad b_v \leftarrow [\alpha_v,\ \min(\beta_v, \beta_u + 1)]. \tag{16}$$

Changes of interval bounds must further be propagated through the constraint graph by recursively following adjacent arcs until all bounds are feasible again w.r.t. the constraints.

An arc (u, v) can be feasibly added to the graph G_C without raising conflicts with any stored constraint as long as the assignment intervals b_u and b_v do not become empty, i.e. $\alpha_u \leq \beta_u \wedge \alpha_v \leq \beta_v$ must always hold. In Algorithm 2 this condition is tested in line 8, and the arc (u, v) is either accepted for A_C or added to $J(P)$, respectively.

Theorem 1. *The recursive update of the assignment interval bounds in G_C after inserting an arc (u, v) always terminates and cannot fail if it succeeded at nodes u and v.*

Proof. Let G_C be valid, i.e. it contains no contradicting constraints, and it was possible to insert arc (u, v) into the graph without obtaining empty assignment intervals for nodes 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, feasibly to α'_t, $\alpha'_t \leq \beta_t$. If the lower bound of s must 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 $\alpha'_s \leq \beta_s$:

$$\alpha'_s \overset{\text{(update rule)}}{=} \alpha'_t - 1 \overset{\alpha'_t \leq \beta_t}{\leq} \beta_t - 1 \overset{\beta_t \leq \beta_s + 1}{\leq} \beta_s \tag{17}$$

The feasibility of the upper bound propagation can be argued in an analogous way. This also proves that the recursive update procedure terminates, even when there are cycles in G_C (intervals cannot become empty, updates increase respectively decrease bounds by at least one). □

3.4 Local Search and Tabu Search

Although the construction heuristics usually find many 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 other partition. As this neighborhood would be relatively large and costly to search, we restrict it as follows: Each arc $(u, v) \in J(P)$ induces two allowed moves to remove it from the jump: reassigning node u to set $S_{\sigma(v)-1}$ and reassigning node v to set $S_{\sigma(u)+1}$. Moves modifying S_0 or S_H are not allowed. The local search is performed in a first improvement manner until a local optimum is reached; see Algorithm 3.

In most cases, the construction heuristics followed by local search are able to identify a jump cut. In the remaining cases, we give tabu search a try to eventually detect still undiscovered violated jump inequalities. Algorithm 4 shows our tabu search procedure in pseudo-code.

Algorithm 3: Local Search

```
   input  : $V^+, A^{\mathrm{LP}}$, current partitioning $P$ and implied jump $J(P)$
   output: possibly improved partitioning $P$ of $V^+$
1  forall arcs $(u, v) \in J(P)$ do
2      if moving $u$ to $S_{\sigma(v)-1}$ or $v$ to $S_{\sigma(u)+1}$ is valid and improves solution then
3          perform move; update $P$ and $J(P)$ correspondingly;
4          restart at 1;
5  return partitioning $P$;
```

Algorithm 4: Tabu Search

```
   input  : $V^+, A^{\mathrm{LP}}$, current partitioning $P$ and implied jump $J(P)$
   output: possibly improved partitioning $P$ of $V^+$
1  tabu list $L = \{\}$;
2  repeat
3      search neighborhood of $P$ for best move $m$ considering tabu list $L$;
4      perform move $m$; update $P$ and $J(P)$ correspondingly;
5      file move $m^{-1}$ in tabu list: $L \leftarrow L \cup \{m^{-1}\}$;
6      remove from $L$ entries older than $\max(l_{\min}, \gamma \cdot |J(P)|)$ iterations;
7  until stopping criterion met ;
8  return best encountered partitioning;
```

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 movement of a node v, we file as tabu the node v in combination with its inverted direction of movement (towards set S_{H+1} or S_0, respectively).

The tabu tenure is dynamically controlled by the number of arcs in jump $J(P)$: Tabu attributes older than $\max(l_{\min}, \gamma \cdot |J(P)|)$ iterations are discarded, where $l_{\min}$ and γ are strategy parameters.

We consider the following aspiration criterion: The tabu status of a move is ignored if the move leads to a new so far best node partitioning. Tabu search terminates when a predefined number $i_{\max}$ of iterations without improvement of the overall best partitioning is reached.

4 Computational Results

For our computational experiments we utilize Euclidean (TE) and random (TR) instances as described and used by Gouveia et al. [7, 8] as well as complete and sparse Euclidean instances of Santos et al. [5]. This instance type, together with the number of nodes ($|V|$) and edges ($|E|$) and the diameter bound (D) is specified for each test case in the following results tables. All experiments have been performed on a dual-core AMD Opteron 2214 machine, and CPLEX 11.0 has been used as ILP solver and framework for Branch&Cut. The strategy parameters of tabu search have been $l_{\min} = 5$, $\gamma = 0.1$, and $i_{\max} = 100$.

Table 1. Success rates SR (%) in separating jump cuts for construction heuristics C^A and C^B optionally followed by local search L and tabu search T in comparison to the exact separation model on the same LP solutions.

Instance	$\|V\|$	$\|E\|$	D	#exact	SR(C^A)	SR(C^AL)	SR(C^B)	SR(C^BL)	SR(C^BLT)
TE	30	200	4	817	99.14%	99.63%	99.14%	99.39%	99.51%
			6	1038	81.61%	98.80%	97.98%	98.81%	99.29%
			8	378	90.08%	95.09%	95.77%	96.03%	97.09%
TR	30	200	4	272	100.00%	100.00%	100.00%	100.00%	100.00%
			6	152	99.34%	99.34%	100.00%	100.00%	100.00%
			8	22	100.00%	100.00%	100.00%	100.00%	100.00%
Santos	25	300	4	316	100.00%	100.00%	100.00%	100.00%	100.00%
			6	126	99.21%	99.21%	100.00%	100.00%	100.00%
			10	77	96.10%	97.40%	100.00%	100.00%	100.00%
	40	100	4	204	100.00%	100.00%	100.00%	100.00%	100.00%
			6	112	99.11%	99.11%	100.00%	100.00%	100.00%
			10	85	52.94%	55.29%	96.47%	96.47%	96.47%
TE	30	200	5	2514	93.95%	97.21%	94.18%	96.53%	97.05%
			7	2257	81.15%	95.49%	96.02%	97.14%	97.79%
TR	30	200	5	377	96.29%	97.08%	96.55%	97.35%	97.35%
			7	89	71.91%	83.71%	92.13%	94.38%	95.51%
Santos	25	300	5	794	96.60%	97.36%	97.73%	98.36%	98.61%
			7	188	87.23%	91.49%	95.21%	95.74%	96.81%
			9	118	85.59%	93.22%	98.31%	98.31%	99.15%
	40	100	5	186	100.00%	100.00%	100.00%	100.00%	100.00%
			7	453	87.70%	92.09%	95.45%	96.17%	96.31%
			9	529	77.54%	89.31%	93.66%	94.90%	95.70%

Table 2. Optimal solution values, running times t (in seconds) to find and prove these solutions when using different strategies for jump cut separation, and optimality gaps of the final LP relaxations in the root nodes of the Branch&Cut search trees when using heuristic C^B followed by local search and tabu search.

Instance	$\|V\|$	$\|E\|$	D	opt	t(exact)	t(C^AL)	t(C^BL)	t(C^BLT)	gap(C^BLT)
TE	30	200	4	599	3522.73	11.25	12.02	10.69	1.69%
			6	482	$>$ 1h	11.68	9.15	3.20	2.61%
			8	437	$>$ 1h	16.46	2.04	0.77	1.98%
TR	30	200	4	234	328.09	1.51	1.40	1.40	0.52%
			6	157	185.65	0.78	0.63	0.66	0.00%
			8	135	0.59	0.13	0.12	0.12	0.74%
Santos	25	300	4	500	809.86	2.13	2.13	2.15	0.24%
			6	378	215.30	0.84	0.72	0.73	0.53%
			10	379	419.03	11.48	0.51	0.51	0.04%
	40	100	4	755	105.34	1.17	1.20	1.21	0.06%
			6	599	41.07	0.51	0.45	0.45	0.00%
			10	574	440.55	25.22	0.40	0.38	0.13%
TE	30	200	5	534	$>$ 1h	53.53	68.46	13.00	7.29%
			7	463	$>$ 1h	22.15	29.44	1.16	6.63%
TR	30	200	5	195	831.31	2.74	2.69	1.51	10.78%
			7	144	139.08	0.35	0.30	0.31	4.57%
Santos	25	300	5	429	1122.52	7.69	10.56	2.94	8.87%
			7	408	2489.67	2.08	2.17	1.64	4.65%
			9	336	66.66	1.07	1.13	1.13	0.90%
	40	100	5	729	238.24	0.93	0.92	1.03	0.11%
			7	667	988.36	2.79	3.40	2.19	1.50%
			9	552	$>$ 1h	5.47	3.48	1.28	3.22%

The experiments were performed with modified jump cut heuristics to simultaneously identify violated directed connection cuts to avoid additional time-consuming max-flow/min-cut computations. It is easy to show that this can be achieved by not forcing the sets $S_1, \ldots, S_H$ to be nonempty.

For smaller instances where the exact jump cut separation can also be applied, Table 1 lists success rates $\mathrm{SR}(\cdot)$ for finding existing violated jump inequalities for the two construction heuristics (C^A and C^B), optionally followed by local search (L) and tabu search (T). The number of cuts identified by the exact model is given in column "#exact". As can be seen, for even diameter already the simple construction heuristic C^A gives excellent results, in most cases further improved by local search. The significantly better heuristic C^B leaves not much room for local and tabu search to enhance the success rate. A more differentiated situation can be observed for odd diameter bounds. The number of jump cuts identified directly by C^B is significantly higher in contrast to C^A, whereas local search flattens the differences in the construction phase to a greater or lesser extent. On all test instances, tabu search further improves the success rate to more than 95%. In total, heuristic C^B followed by local search and tabu search was able to separate all jump inequalities for 9 out of 22 instances.

The consequences of the success to reliably identify violated jump inequalities can be seen in Table 2, where for the various approaches CPU-times $\mathrm{t}(\cdot)$ to identify proven optimal integer solutions are listed. It can clearly be seen that the excessive running times of the exact jump cut separation prohibit its usage on larger instances. Times of the overall optimization process are in general magnitudes higher as when using our heuristics for jump cut separation, sometimes even the given CPU-time limit of one hour is exceeded. The best performance can be observed for C^B with local and tabu search. Since tabu search is only executed in case the construction heuristic followed by local search fails to identify a violated jump inequality, running times of C^BL and C^BLT considerably differ only on few instances, especially when D is odd. Table 2 also lists optimal solution values ("opt") as well as optimality gaps of the LP relaxations at the root nodes of the Branch&Cut search trees for C^BLT. Whereas our model is quite tight in the even diameter case, the gaps for odd diameters reveal potential for further investigations to strengthen the formulation.

Finally, Table 3 compares our approach to the so far leading hop-indexed multi-commodity flow formulations from [7] (even diameter cases) and [8] (odd diameter cases) on larger instances. The several columns list for each instance the optimal objective value if known, otherwise an upper bound (opt/UB*), the LP relaxation value for construction heuristic C^B with local search (LP(C^BL)), the gap for this approach and for the best model from [7] and [8] whenever the optimum is available resp. the corresponding values were published (gap(C^BL), gap(GMR)), as well as the running time to proven optimality (t(C^BL)); a time limit of 10 hours was used for these experiments.

We were able to discover and prove previously unknown optima (bold) and could show that instance TE 80/800/4 is infeasible. Concerning the LP gaps the results are comparable on even diameter instances, while for odd diameters the

Table 3. Optimal values resp. upper bounds, LP relaxation values, LP gaps (for C^BL and GMR, the tightest models from [7] and [8]), and running times on Euclidean and random instances with 40, 60, and 80 nodes.

Instance	$\|V\|$	$\|E\|$	D	opt/UB*	LP(C^BL)	gap(C^BL)	gap(GMR)	t(C^BL)
TE	40	400	4	672	670.98	0.15%	0.04%	27.66
			6	555	544.49	1.89%	0.60%	125.92
			8	507	500.06	1.37%	0.50%	79.80
	60	600	4	1180	1178.50	0.13%	0.10%	1140.70
			6	837	816.85	2.41%	0.50%	10453.38
			8	**755**	736.28	2.48%		34167.90
	80	800	4	**infeasible**		infeasible		2202.09
			6	**1066**	1044.77	1.99%		> 10h
			8	968*	924.81	*4.46%		> 10h
TR	40	400	4	309	307.92	0.35%	0.00%	23.29
			6	189	189.00	0.00%	0.00%	2.86
			8	161	161.00	0.00%	0.00%	0.77
	60	600	4	326	323.36	0.81%	0.70%	1846.71
			6	175	171.36	2.08%	1.30%	582.42
			8	127	127.00	0.00%	0.00%	4.81
	80	800	4	424	399.67	5.74%	5.70%	> 10h
			6	**210**	206.42	1.70%		2806.13
			8	**166**	164.33	1.01%		27.18
TE	40	400	5	612	578.39	5.49%	0.00%	345.59
			7	527	494.91	6.09%	0.30%	398.94
			9	495	468.07	5.44%	0.30%	179.83
	60	600	5	965	899.54	6.78%	0.00%	> 10h
			7	789	742.03	5.95%	0.00%	> 10h
			9	738	690.87	6.39%	0.50%	> 10h
	80	800	5	**1313**	1200.69	8.55%		> 10h
			7	1010*	942.41	*6.69%		> 10h
			9	964*	871.47	*9.60%		> 10h
TR	40	400	5	253	224.90	11.11%	1.00%	17.95
			7	171	169.11	1.11%	0.00%	2.17
			9	154	154.00	0.00%	0.00%	1.09
	60	600	5	256	217.14	15.18%	3.20%	1091.02
			7	150	138.50	7.67%	0.30%	35.04
			9	124	119.84	3.35%	0.00%	5.94
	80	800	5	**323**	272.29	15.79%		> 10h
			7	**185**	176.44	4.63%		189.40
			9	**158**	154.57	2.17%		17.89

flow models are significantly better. A fair runtime comparison to [7] and [8] is not possible since the used hardware is too different. A rough estimation indicates that the flow formulations have their strengths on small diameter bounds, whereas Branch&Cut dominates when the diameter bound is looser.

5 Conclusions and Future Work

In this work we presented a new ILP formulation for the BDMST problem utilizing jump inequalities to ensure the diameter constraint and solve it with Branch&Cut. For the separation of jump inequalities we considered an exact ILP approach and two greedy construction heuristics followed by local and tabu search. While our exact separation prohibits its use in practice due to its excessive computation times, the heuristic methods are substantially faster and achieve convincing success rates in identifying violated jump inequalities; they lead to an excellent overall performance of the Branch&Cut.

Results on smaller instances demonstrate the benefit of applying tabu search to further increase the success rate in identifying violated cuts. Experiments and more detailed statistical analysis are required for larger instances with 40 nodes and more. Finally, improvements of our model seem to be possible especially for the odd diameter case.

References

1. Bala, K., Petropoulos, K., Stern, T.E.: Multicasting in a linear lightwave network. In: Proc. of the 12th IEEE Conference on Computer Communications, IEEE Press (1993) 1350–1358
2. Raymond, K.: A tree-based algorithm for distributed mutual exclusion. ACM Transactions on Computer Systems **7**(1) (1989) 61–77
3. Bookstein, A., Klein, S.T.: Compression of correlated bit-vectors. Information Systems **16**(4) (1991) 387–400
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
5. dos Santos, A.C., Lucena, A., Ribeiro, C.C.: Solving diameter constrained minimum spanning tree problems in dense graphs. In: Proc. of the Int. Workshop on Experimental Algorithms. Volume 3059 of LNCS., Springer (2004) 458–467
6. Gruber, M., Raidl, G.: A new 0–1 ILP approach for the bounded diameter minimum spanning tree problem. In Gouveia, L., Mourão, C., eds.: Proc. of the Int. Network Optimization Conference. Volume 1., Lisbon, Portugal (2005) 178–185
7. Gouveia, L., Magnanti, T.L.: Network flow models for designing diameter-constrained minimum spanning and Steiner trees. Networks **41**(3) (2003) 159–173
8. Gouveia, L., Magnanti, T.L., Requejo, C.: A 2-path approach for odd-diameter-constrained minimum spanning and Steiner trees. Networks **44**(4) (2004) 254–265
9. Raidl, G.R., Julstrom, B.A.: Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. In Lamont, G., et al., eds.: Proc. of the ACM Symposium on Applied Computing, ACM Press (2003) 747–752
10. Gruber, M., Raidl, G.R.: Variable neighborhood search for the bounded diameter minimum spanning tree problem. In Hansen, P., et al., eds.: Proc. of the 18th Mini Euro Conference on Variable Neighborhood Search, Tenerife, Spain (2005)
11. Gruber, M., van Hemert, J., Raidl, G.R.: Neighborhood searches for the bounded diameter minimum spanning tree problem embedded in a VNS, EA, and ACO. In Keijzer, M., et al., eds.: Proc. of the Genetic and Evolutionary Computation Conference 2006. Volume 2. (2006) 1187–1194
12. Dahl, G., Gouveia, L., Requejo, C.: On formulations and methods for the hop-constrained minimum spanning tree problem. In: Handbook of Optimization in Telecommunications. Springer Science + Business Media (2006) 493–515
13. Gouveia, L., Simonetti, L., Uchoa, E.: Modelling the hop-constrained minimum spanning tree problem over a layered graph. In: Proc. of the Int. Network Optimization Conference, Spa, Belgium (2007)
14. Dahl, G., Flatberg, T., Foldnes, N., Gouveia, L.: 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)