A New 0–1 ILP Approach for the Bounded Diameter Minimum Spanning Tree Problem

Martin Gruber and Günther R. Raidl
{gruber|raidl}@ads.tuwien.ac.at

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/186-1, 1040 Vienna, Austria

January 2005

Abstract

The bounded diameter minimum spanning tree (BDMST) problem is NP-hard and appears e.g. in communication network design when considering certain quality of service requirements. Prior exact approaches for the BDMST problem mostly rely on network flow-based mixed integer linear programming and Miller-Tucker-Zemlin-based formulations. This article presents a new, compact 0–1 integer linear programming model, which is further strengthened by dynamically adding violated connection and cycle elimination constraints within a branch-and-cut environment. The proposed approach is empirically compared to two recently published formulations. It turns out to work well in particular on dense instances with tight diameter bounds.

Keywords: bounded diameter minimum spanning tree, branch-and-cut, integer linear programming

1 Introduction

In the *bounded diameter minimum spanning tree* (BDMST) problem, we are given an undirected, connected graph $G = (V, E)$ of $n = |V|$ nodes and $m = |E|$ edges. Each edge $(i, j) \in E$ has associated costs $c_{i,j} \geq 0$. The goal is to find a minimum spanning tree $T = (V, E_T)$ with $E_T \subseteq E$ whose diameter does not exceed a given upper bound $D \geq 2$. The BDMST problem is known to be NP-hard for $4 \leq D < n - 1$ [6].

The *eccentricity* of a node v is the maximum number of edges on a path from v to any other node in the tree T. The *diameter* of T is the maximum eccentricity of its nodes, thus the largest number of edges on any path. The *center* of T is the one node (if T's diameter is even) or the pair of adjacent nodes (if T's diameter is odd) of minimum eccentricity.

The computation of a BDMST is an optimization problem found in communication network design when certain aspects of quality of service are considered, e.g., when a maximum communication delay or a minimum signal-to-noise ratio must be guaranteed, and thus the number of relaying nodes on any path needs to be restricted. It is also met in the fields of data compression and distributed mutual exclusion and arises as a subproblem in other combinatorial optimization problems such as vehicle routing [2].

Prior exact approaches for determining a BDMST mostly rely on network flow-based mixed integer linear programming (MIP) and Miller-Tucker-Zemlin (MTZ) based formulations. We present a new 0–1 integer linear programming model involving only $\Theta(m + n \cdot D)$ variables and $\Theta(m \cdot D)$ constraints. It is strengthened by dynamically adding violated connection and cycle elimination constraints within a branch-and-cut environment. The proposed approach is empirically compared to two recently published models based on network flows and the MTZ formulation, respectively.

Work supported by the Austrian Science Fund (FWF) under grant P16263-N04.

2 Previous Work

Fundamental work of using MIPs in the field of network design was done by Magnanti and Wong [12], who proposed multi-source multi-destination network flow models. Achuthan and Caccetta [3] suggested a simpler multi-commodity formulation specifically for the BDMST by introducing virtual source and destination nodes. An improved version was published by Achuthan et al. in [4] together with a branch-and-bound framework.

Gouveia and Magnanti [8] investigated different extended variants of multi-commodity flow (MCF) formulations for the BDMST problem, in which they count and limit the hops on paths from a virtual root node to any other node. They achieved in particular tight lower bounds with the LP-relaxations of their models, however, at the expense of a large number of flow variables ($O(|E| \cdot |V| \cdot D)$). In [9], Gouveia et al. introduced an extended flow formulation for the odd diameter case. The BDMST is viewed as being composed of a variant of a directed spanning tree from an artificial root node together with a shortest and a longest constrained path from the root to any node in the tree. Gouveia et al. [8, 9] also applied modified MCF formulations to the related Steiner tree problem with hop constraints.

A formulation based on lifted Miller-Tucker-Zemlin inequalities responsible for avoiding cycles and ensuring the maximum diameter are presented in Santos et al. [5]. This approach is claimed to work well in particular on dense graphs. See Voß [17] for the use of MTZ constraints to compute Steiner trees.

Simple greedy heuristics for the BDMST have been published by Abdalla et al. [1] and by Julstrom [11]. Evolutionary algorithms are described in Raidl and Julstrom [15] and Julstrom [10].

3 A Compact 0–1 ILP Formulation

As an alternative to the various existing formulations, we model the BDMST problem as a 0–1 integer linear program (ILP) using only $\Theta(m + n \cdot D)$ variables and $\Theta(m \cdot D)$ constraints.

Like most of the existing formulations, we view the task of finding a BDMST as a directed graph problem. However, we do not introduce any artificial nodes. Let A be the arc set derived from E by including for each undirected edge $(i, j) \in E$ two oppositely directed arcs (i, j) and (j, i) with the same costs $c_{i,j} = c_{j,i}$. In our formulation of the BDMST only the two center nodes in case of an odd diameter are connected by an undirected edge. All other edges are directed in such a way that there exists a directed path from the center to any other node. Let the *depth* $d(v)$ of a node $v \in V$ be the number of arcs on the path from the center to node v. The center node(s) have depth $d(v) = 0$. The height of this rooted tree is then at most $H = \lfloor \frac{D}{2} \rfloor$. The BDMST can be seen as a variation of the hop constrained minimum spanning tree problem with a root not fixed in advance [7].

We formulate the BDMST problem by assigning each node v a depth $0 \leq d(v) \leq H$. Furthermore, each node of depth $d(v) > 0$ has to have one *predecessor* denoted by $pred(v)$. In order to get a feasible spanning tree, $d(pred(v)) = d(v) - 1$ must hold for all nodes $v \in V \mid d(v) > 0$.

The ILP uses the following variables:

- $u_{i,l} \in \{0, 1\}$; $i \in V$, $l = 0, \ldots, H$: $u_{i,l} = 1 \leftrightarrow d(i) = l$.
- $p_{i,j} \in \{0, 1\}$; $(i, j) \in A$: $p_{i,j} = 1 \leftrightarrow pred(j) = i$, i.e., $(i, j) \in E_T$.
- if D is odd:
  $r_{i,j} \in \{0, 1\}$; $(i, j) \in E$: $r_{i,j} = 1 \leftrightarrow$ edge $(i, j) \in E_T$ and connects the center nodes.

3.1 The Even Diameter Case

In the simplest form, the even diameter case can be modeled as follows:

$$\text{minimize} \quad \sum_{(i,j) \in A} p_{i,j} \cdot c_{i,j} \tag{1}$$

$$\text{subject to} \quad \sum_{l=0}^{H} u_{i,l} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i \in V} u_{i,0} = 1 \tag{3}$$

$$\sum_{i|(i,j)\in A} p_{i,j} = 1 - u_{j,0} \quad \forall j \in V \tag{4}$$

$$p_{i,j} \leq 1 - u_{j,l} + u_{i,l-1} \quad \forall (i,j) \in A,\ \forall l = 1, \ldots, H. \tag{5}$$

The objective (1) is to minimize the total costs of all used arcs ($p_{i,j} = 1$). Equations (2) ensure that each node i is assigned to exactly one depth. Depth zero must be assigned to exactly one node, the center (3). Constraints (4) guarantee that each node except the center always has exactly one predecessor. Finally, we have to model the fact that node i can only be predecessor of node j if $d(i) = d(j) - 1$, or in other words $p_{i,j} = 1 \rightarrow \exists\, l \mid u_{j,l} = 1 \wedge u_{i,l-1} = 1$. This is assured by inequalities (5).

To strengthen the LP-relaxation we further add constraints which explicitly express that a node assigned to depth zero cannot have a predecessor, nor can a node assigned to depth H be predecessor of any other node:

$$u_{j,0} \leq 1 - p_{i,j} \quad \text{and} \quad u_{i,H} \leq 1 - p_{i,j} \quad \forall (i,j) \in A. \tag{6}$$

3.2 The Odd Diameter Case

In case the diameter D is odd, we can formulate the BDMST problem as follows:

$$\text{minimize} \quad \sum_{(i,j)\in A} p_{i,j} \cdot c_{i,j} + \sum_{(i,j)\in E} r_{i,j} \cdot c_{i,j} \tag{7}$$

$$\text{subject to} \quad \sum_{i \in V} u_{i,0} = 2 \tag{8}$$

$$\sum_{j|(i,j)\in E} r_{i,j} = u_{i,0} \quad \forall i \in V \tag{9}$$

Constraints (2), (4), (5) and (6) are adopted unchanged.

Starting from the ILP (1) to (6) we have to adapt the objective function (1) to include the costs of the edge connecting the two center nodes (7) as well as equation (3) to reflect the fact that now there are two nodes assigned to depth zero (8). In addition, constraints (9) are required in order to get the edge connecting the two center nodes at depth zero.

Equations (9) together with (8) imply the following constraint that does not actually strengthen the LP-relaxation, but nevertheless speeds up the integer optimization in practice as our experiments indicate:

$$\sum_{e \in E} r_e = 1. \tag{10}$$

Finally, Gouveia and Magnanti [8] suggested to exploit the fact that in a BDMST with odd D, the nodes of depth one are always connected to the nearer one of the two center nodes. We consider this aspect in the following way.

For each arc $(i,j) \in A$, let $L(i,j)$ be the set of all potential center edges when i is one of the center nodes, j appears at depth one, and i is j's predecessor:

$$L(i,j) = \{(i,i') \in E \mid j \neq i' \wedge (c_{i,j} < c_{i',j} \vee c_{i,j} = c_{i',j} \wedge i < i')\}. \tag{11}$$

If $u_{j,1} = 1$, $p_{i,j}$ may only be set to one if the center edge appears in $L(i,j)$:

$$p_{i,j} \leq 1 - u_{j,1} + \sum_{e \in L(i,j)} r_e \quad \forall (i,j) \in A. \tag{12}$$

4 Branch-and-Cut

Achuthan et al. [4] already described the possibility to include constraints for explicitly avoiding cycles in order to strengthen the LP-relaxation. Unfortunately, considering each possible cycle and adding an appropriate inequality from the very beginning – as Achuthan et al. did – leads to a huge number of constraints which increases exponentially with D. This limits the applicability to very small diameters and/or instances.

We suggest a branch-and-cut approach, in which connectivity and cycle elimination constraints violated in the solution of the LP-relaxation are iteratively determined and added as cuts throughout the optimization process. These sorts of cuts are often used in cutting plane algorithms for various network design problems, such as the traveling salesman problem [13].

4.1 Connection Cuts

A spanning tree always has to be connected, i.e., each non-empty subset S of all nodes V has to induce a cut $\delta(S)$ of size greater than or equal to one. More formally,

$$\sum_{(i,j)\in\delta(S)} (p_{i,j} + r_{i,j}) \geq 1 \quad \forall S \subset V,\, S \neq \emptyset, \tag{13}$$

where variables $r_{i,j}$ are only required in case the diameter is odd. Using a minimum cut algorithm, it is easy to identify a subset of nodes for which the above constraint is violated in the solution of the LP-relaxation or to prove that no such set exists.

4.2 Cycle Elimination Cuts

In a tree no cycle may appear, and therefore we can consider the constraints

$$\sum_{(i,j)\in C} p_{i,j} + p_{j,i} + r_{i,j} \leq |C| - 1 \quad \forall \text{ cycles } C \subset E, \tag{14}$$

where again variables $r_{i,j}$ only have to be considered in the odd diameter case. We separate such cuts by constructing an undirected graph from the solution of the LP-relaxation with node set V, edge set $E'_T = \{(i,j) \in E \mid p_{i,j} + p_{j,i} + r_{i,j} > 0\}$, and with associated costs $\max(0, 1 - p_{i,j} - p_{j,i} - r_{i,j})$. A cycle violating the corresponding constraint has total costs less than 1 and can be identified by calculating the shortest path for each pair of nodes i and j with $(i,j) \in E'_T$.

From a theoretical point of view, connection and cycle elimination cuts are both covered by the more general subtour elimination cuts, where C is replaced by the set of all edges induced by a subset of vertices from V [14]. However, finding general subtour elimination cuts is much more time demanding than identifying violated connection and cycle constraints. Therefore, it is usually a good idea to first separate connection and cycle elimination constraints, before considering general subtour elimination cuts. We have not yet implemented the latter and leave them for future research. Since the separation of connection cuts is faster than looking for cycle elimination cuts, we first separate all violated connection constraints and only then calculate cycle elimination cuts when using both.

5 Computational Results

We compare our approach to the MTZ-based formulation of Santos et al. [5] (variant (B) and (C) for even and odd diameters, respectively) and the MCF formulation of Gouveia and Magnanti [8] (HopDMCF and Enh-HopMCF, respectively). The same test instances as in these previous works are used. Our experiments were performed on a Pentium®4 2.8 GHz system with 2 GB of RAM using Linux 2.4.21 and CPLEX 8.1 under default parameters as MIP solver.

Table 1 lists CPU times for finding optimal solutions and proving optimality on complete and sparse instances for our reimplementation of Santos et al. and our ILP formulation. Variants marked by '+' apply connection cuts, whereas those marked by '$*$' use cycle elimination cuts. Enclosed in parentheses we list the percentage gap between the optimal solution (opt) and the LP-relaxation (lb) at the first node within the branch-and-cut tree after performing all separations of our own and the standard cuts of CPLEX: $gap = (opt - lb)/opt \cdot 100\%$. In case there are two values listed the first one depicts the gap at the very beginning only depending on the model used and so invariant in terms of applying cuts.

Table 1: CPU times (in seconds) and LP-relaxation gaps on Euclidean instances from Santos et al. [5].

$\|V\|$	$\|E\|$	D	Santos		Santos+		Santos∗		Santos+∗		ILP		ILP+		ILP∗		ILP+∗	
15	105	4	4.7	(37.4/23.9)	9.8	(22.4)	18.88	(21.0)	29.9	(20.1)	**0.7**	(36.4/22.0)	0.9	(25.1)	1.1	(19.7)	1.7	(20.1)
15	105	5	22.8	(33.8/28.9)	36.9	(25.5)	44.1	(20.8)	78.7	(18.2)	**3.0**	(33.6/29.2)	3.2	(25.3)	5.8	(19.2)	7.0	(18.1)
15	105	6	21.1	(31.5/19.7)	18.6	(16.4)	52.6	(16.9)	48.6	(13.7)	**8.1**	(38.9/32.8)	24.1	(25.0)	15.6	(16.3)	33.6	(13.7)
15	105	7	44.8	(28.1/23.7)	29.3	(18.0)	38.8	(13.9)	26.9	(10.6)	26.4	(33.6/28.3)	37.2	(28.3)	20.5	(14.0)	**20.0**	(10.6)
15	105	9	18.4	(24.8/21.1)	16.5	(15.1)	15.8	(10.4)	**6.2**	(6.6)	150.6	(32.5/26.8)	47.0	(21.1)	20.7	(10.3)	10.7	(6.6)
15	105	10	1.5	(24.8/12.1)	**1.0**	(8.3)	5.1	(9.2)	2.4	(5.3)	65.8	(36.1/29.1)	43.3	(20.6)	15.8	(9.2)	4.3	(5.3)
20	190	4	562.9	(29.7/24.4)	1,063.6	(21.6)	1,232.5	(24.1)	4,315.1	(21.4)	**2.5**	(28.8/25.0)	3.3	(21.3)	3.7	(22.3)	5.0	(20.2)
20	190	5	436.7	(25.2/20.9)	662.5	(20.3)	572.9	(17.4)	1,260.6	(17.3)	**8.1**	(27.4/17.8)	10.2	(17.8)	9.3	(15.2)	12.8	(15.6)
20	190	6	577.0	(20.3/13.8)	489.3	(13.3)	455.2	(12.5)	531.9	(12.1)	**95.0**	(28.2/21.7)	210.5	(19.7)	396.4	(12.5)	382.8	(12.5)
20	190	7	8.1	(13.4/9.7)	5.1	(7.7)	7.8	(5.9)	10.2	(5.1)	10.0	(19.3/11.3)	12.9	(8.3)	**4.5**	(5.0)	7.2	(5.0)
20	190	9	241.9	(22.2/18.5)	105.7	(14.4)	244.3	(11.2)	73.4	(8.9)	1,209.1	(30.0/24.4)	923.7	(18.6)	194.6	(11.0)	**66.7**	(8.8)
20	190	10	64.6	(21.9/15.4)	41.9	(10.6)	205.1	(11.2)	**29.7**	(8.1)	13,755.5	(34.0/26.1)	13,972.1	(18.4)	226.4	(11.3)	101.0	(8.1)
25	300	4	15,203.7	(30.3/26.6)	> 20,000.0	(25.9)	> 20,000.0	(26.1)	> 20,000.0	(25.1)	**12.0**	(31.2/23.2)	14.2	(22.9)	16.5	(22.2)	20.3	(21.7)
25	300	5	> 20,000.0	(32.4/28.7)	> 20,000.0	(25.1)	> 20,000.0	(25.5)	> 20,000.0	(22.6)	76.3	(33.5/27.4)	**64.3**	(23.1)	102.7	(23.7)	127.3	(20.9)
25	300	6	1,282.5	(18.6/12.4)	826.7	(11.4)	1,241.7	(10.9)	1,151.3	(10.1)	**26.4**	(28.4/17.0)	166.7	(13.9)	75.8	(10.6)	146.1	(10.1)
25	300	7	11,521.3	(18.7/15.9)	> 20,000.0	(14.0)	> 20,000.0	(13.3)	17,014.1	(11.4)	**770.5**	(26.5/17.6)	2,719.3	(14.9)	1,090.3	(13.0)	989.6	(11.6)
25	300	9	> 20,000.0	(22.7/18.4)	**246.0**	(8.0)	13,677.6	(15.5)	429.7	(5.3)	> 20,000.0	(31.3/22.7)	2160.1	(11.9)	4,143.3	(15.4)	295.4	(5.2)
25	300	10	278.2	(10/8.6)	**254.8**	(5.9)	401.4	(7.0)	327.0	(5.3)	3,666.1	(24.0/12.6)	2,904.2	(9.2)	1,127.9	(7.2)	404.9	(5.3)
20	50	4	1.0	(32.9/19.1)	1.0	(16.5)	4.7	(18.8)	3.8	(16.4)	**0.2**	(29.0/13.9)	**0.2**	(13.4)	0.3	(13.8)	0.4	(13.2)
20	50	5	4.6	(60.7/57.8)	9.0	(52.5)	15.6	(54.6)	23.7	(51.2)	**1.0**	(62.1/58.8)	2.5	(53.6)	3.5	(53.8)	4.9	(50.6)
20	50	6	34.6	(28.9/20.8)	**0.8**	(9.5)	45.6	(15.2)	1.5	(8.8)	8.7	(35.1/27.2)	5.1	(11.6)	19.9	(15.2)	6.0	(8.8)
20	50	7	13.3	(26.1/22.4)	**0.8**	(8.4)	13.3	(20.8)	1.2	(7.9)	1.2	(27.4/25.5)	1.7	(7.8)	2.5	(19.5)	3.0	(7.4)
20	50	9	76.2	(24.3/19.5)	**0.7**	(7.7)	108.5	(14.5)	0.8	(4.6)	42.5	(31.8/24.9)	2.8	(10.4)	25.6	(14.4)	3.7	(4.6)
20	50	10	98.5	(29.5/21.7)	**0.2**	(3.9)	187.7	(16.8)	**0.2**	(2.2)	505.3	(40.4/32.8)	1.8	(7.4)	79.1	(16.7)	1.3	(2.2)
40	100	4	43.7	(41.9/27.7)	82.4	(23.5)	516.5	(24.6)	2,352.1	(23.0)	**1.9**	(39.6/20.4)	**1.9**	(19.4)	8.5	(19.0)	10.0	(18.5)
40	100	5	471.0	(65.4/56.7)	291.7	(52.9)	1,646.5	(56.1)	893.2	(52.3)	**6.4**	(62.9/47.5)	6.8	(46.9)	13.1	(46.9)	18.2	(46.7)
40	100	6	1,991.6	(29.9/25.7)	50.9	(4.6)	13,719.8	(23.3)	100.2	(4.1)	182.9	(44.8/30.2)	**13.2**	(6.4)	449.6	(23.3)	37.4	(4.6)
40	100	7	> 20,000.0	(36.6/21.1)	459.4	(14.7)	3,731.7	(20.4)	10,224.2	(13.4)	**212.4**	(45.5/34.5)	4,463.2	(15.2)	525.2	(20.9)	9455.3	(13.2)
40	100	9	14,882.5	(27.7/23.3)	1,565.0	(15.8)	13,078.0	(21.1)	4,262.7	(14.5)	3,828.7	(39.2/28.2)	9,974.3	(16.9)	**979.8**	(21.1)	5137.2	(14.6)

Table 2: CPU times (in seconds) and LP-relaxation gaps on incomplete instances from Gouveia and Magnanti [8] (gaps for G&M at the beginning of the first branch-and-cut node without CPLEX generated cuts).

	$\|V\|$	$\|E\|$	D	G&M		ILP		ILP+		ILP∗		ILP+∗	
Random	20	100	4	**0.5**	(0.0)	0.9	(35.8/23.9)	0.9	(23.9)	1.5	(24.3)	1.6	(23.4)
	20	100	5	6.3	(0.0)	2.7	(35.5/27.0)	**2.4**	(26.9)	**2.4**	(25.6)	3.2	(25.6)
	20	100	6	5.8	(1.0)	**2.9**	(34.7/21.0)	7.2	(20.6)	5.9	(16.3)	7.9	(18.0)
	20	100	7	94.0	(1.5)	10.6	(31.6/17.1)	5.7	(16.2)	**2.6**	(13.1)	15.8	(13.2)
	20	100	8	**1.3**	(0.0)	4.4	(28.1/10.0)	4.3	(9.4)	3.4	(5.8)	5.7	(5.8)
Random	30	200	4	**0.8**	(0.0)	3.5	(39.8/30.1)	6.1	(29.7)	6.3	(29.8)	8.7	(29.5)
	30	200	5	**58.6**	(0.0)	377.8	(48.1/41.9)	283.8	(41.9)	428.1	(42.1)	482.4	(41.6)
	30	200	6	**2.9**	(0.0)	5.7	(28.4/19.4)	20.4	(14.9)	11.5	(14.6)	39.6	(13.5)
	30	200	7	529.4	(0.0)	112.4	(21.3/16.6)	261.1	(16.3)	**53.9**	(14.3)	100.8	(14.2)
	30	200	8	**2.3**	(0.0)	10.6	(18.3/14.3)	13.9	(5.4)	3.67	(8.3)	4.9	(2.2)
Euclidian	20	100	4	**0.1**	(0.0)	1.1	(20.3/17.5)	1.4	(17.4)	2.3	(17.2)	2.2	(17.1)
	20	100	5	5.3	(0.0)	**1.7**	(16.8/13.2)	2.0	(13.0)	2.5	(12.8)	3.5	(12.6)
	20	100	6	**3.1**	(0.2)	7.3	(16.3/10.4)	9.5	(9.9)	15.5	(8.0)	21.2	(8.5)
	20	100	7	49.5	(0)	**10.0**	(14.1/8.3)	24.3	(8.2)	11.2	(7.0)	31.3	(7.6)
	20	100	8	**1.1**	(0.0)	10.7	(14.2/10.0)	12.9	(7.7)	18.9	(6.8)	31.2	(5.2)
Euclidian	30	200	4	130.8	(1.7)	148.6	(35.2/25.8)	84.9	(26.0)	**59.5**	(26.3)	107.0	(25.3)
	30	200	5	**25.1**	(0.1)	36.1	(33.9/25.7)	42.2	(26.1)	65.3	(24.1)	61.5	(25.6)
	30	200	6	1,381.9	(0.8)	**348.0**	(28.8/19.4)	4,022.7	(17.6)	7,224.2	(16.9)	17,144.3	(15.0)
	30	200	7	6,912.1	(1.2)	1,339.7	(23.5/17.5)	3,713.8	(16.3)	**1,014.4**	(13.6)	4,205.4	(13.4)
	30	200	8	**1,111.0**	(0.8)	2,864.7	(22.6/16.2)	9,298.8	(12.6)	2,430.6	(11.5)	$> 20{,}000.0$	(9.4)

In general, no approach dominates any other. However, it can be observed that the ILP formulation performs better on tighter constrained instances with rather small diameters, whereas the model of Santos et al. becomes faster than the ILP for looser diameter constraints.

Applying our cuts to the different formulations lead to ambivalent results, except on sparse instances where connection cuts show significantly better performance. It also turns out that the ILP model in general benefits more from cycle elimination cuts, in particular on bigger instances with larger diameters.

Concerning the listed LP-relaxation gaps we further remark that in addition to our connection and cycle elimination cuts, CPLEX sometimes adds additional general-purpose cuts, which further reduce the LP-bound. In a few cases, this leads to the effect that the LP-bound of a variant where only one type of our cuts is used is slightly smaller than when applying both of them.

Table 2 shows running times and LP-bounds for the ILP variants and Gouveia and Magnanti's (G&M) approach. The times listed for G&M are adopted from [8] and scaled by a factor of 1/8 to account for different hardware. This factor has been determined by considering the widely used floating point benchmarks published at http://www.spec.org. Nevertheless, we remark that this scaling is only a rough estimation and running time comparisons should be taken with care.

Concerning the LP-relaxation gaps it can easily be seen that the ILP model cannot compete with the flow formulations of G&M. When looking at the computation times no single variant dominates any other. The different ILP approaches outperform G&M on several occasions. However, no real pattern can be observed for conditions under which a certain method performs best; speed-up factors are in general not as large as those of the first series of experiments (Table 1).

In this context we want to point out that the above results are not sufficient to draw more general conclusions on expected running times for specific classes of instances. During our tests we experienced highly varying computation times for instances randomly generated all the same way. For example, when running ILP+ on 10 different complete Euclidean graphs with 20 randomly distributed nodes, CPU times ranged from 51 to 54,400 seconds. A similar behavior was observed for the model of Santos et al. and the flow formulations as well.

In addition to the presented results we also made first experiments on instances with 40 nodes and more. As expected due to the observed LP-relaxation gaps the size of the branch-and-cut tree grows fast and so does the computation time. Without further improvements the ILP results will not be competitive to recently published flow formulations.

We also tried to reduce the initialization overhead always involved when calculating connection or cycle elimination cuts. A number of two to three cuts generated at once (i.e. cuts added to the model before a new LP is solved) has proven to be a good choice.

6 Additional Constraints

In this section we present some additional constraints we made experiments with, some of them strengthening the LP-relaxation of the ILP model. However, exhaustive benchmarks led to the conclusion not to include them into the model in general, because over all tested instances the running time behavior was not significantly better.

$$\sum_{(i,j)\in A} p_{i,j} = n - 1 - (D \bmod 2) \tag{15}$$

A spanning tree of n nodes contains $n-1$ edges. In case of an odd diameter we have to subtract the center edge encoded separately using the variables $r_{i,j}$. When including this equation to the model the calculation times are undifferentiated; they range from three times faster and more to about two times slower.

$$p_{i,j} \leq 1 - u_{i,l-1} + u_{j,l} \quad \forall (i,j) \in A,\ \forall l = 1,\ldots,H. \tag{16}$$

These constraints are a modification of the inequalities (5). Replacing (5) by them yields a poorer LP-relaxation. Using both types of constraints strengthens the LP-relaxation noticeable, but in either case the running times are substantially higher.

$$p_{i,j} + p_{j,i} \leq 1 \quad \forall (i,j) \in A \tag{17}$$

A node i cannot be predecessor and successor of a node j at the same time. These inequalities strengthen the LP-relaxation significantly and often speed up computation for instances with larger diameter (approx. 8 and above) and sparse graphs. For other instances, however, running times are usually increased.

$$p_{i,j} \leq 1 - u_{j,l} + \sum_{l'=0}^{l-1} u_{i,l'} \quad \forall (i,j) \in A,\ \forall l = 1,\ldots,H \tag{18}$$

A relaxed alternative for inequalities (5), where an edge from a node at depth l may be attached to any node at depth $\leq l-1$, not just $l-1$. This formulation leads to weaker LP-bounds as well as a poorer running time behavior.

$$\delta_{i,j} + (H+1)p_{i,j} + (H-1)p_{j,i} \leq H \qquad \forall (i,j) \in A \qquad \text{with} \tag{19}$$

$$\delta_{i,j} = \sum_{l=0}^{H} l \cdot u_{i,l} - l \cdot u_{j,l} \qquad \forall (i,j) \in A \tag{20}$$

being the difference of the depths of nodes i and j. These lifted MTZ constraints from [5] strengthen the LP-relaxation, but the computational results do not exhibit a conclusive pattern.

7 Conclusions and Future Work

A compact 0–1 ILP for the BDMST problem has been introduced. When solving this ILP with the general purpose MIP-solver CPLEX, the performance can be improved by separating violated connectivity and/or cycle constraints and adding them in a branch-and-cut manner at each node of the branch-and-bound tree. The new approach has been compared to two recently published formulations. It turns out that in particular dense instances with small to moderate diameter bounds are solved significantly faster than with the approach from Santos et al. [5]. This MTZ-base model, however, exhibits its strengths on instances where D approaches the diameter of the unconstrained minimum spanning tree.

In [9], Gouveia et al. introduced a specific improvement for the odd diameter case we did not consider so far. Further theoretical investigations and experiments on more and larger benchmark instances are necessary to get a closer insight into the specific assets and drawbacks of the different formulations.

Besides the implementation of tighter subtour elimination cuts as a substitute for the cycle elimination cuts another future objective will be to combine exact algorithms like those discussed here

with (meta-)heuristic approaches. This will be done not just in the classical sense – for example by heuristically determining a good starting solution for an exact algorithm – but also by running different algorithms in parallel and letting them exchange information relevant for the optimization in order to benefit from synergy [16].

References

[1] A. Abdalla, N. Deo, and P. Gupta. Random-tree diameter and the diameter constrained MST. *Congressus Numerantium*, 144:161–182, 2000.

[2] N. R. Achuthan and L. Caccetta. Models for vehilce routing problems. *Proceedings of the 10th National Conference of the Australian Society for Operations Research*, pages 276–294, 1990.

[3] N. R. Achuthan and L. Caccetta. Minimum weight spanning trees with bounded diameter. *Australasian Journal of Combinatorics*, 5:261–276, 1992.

[4] N. R. Achuthan, L. Caccetta, P. Caccetta, and J. F. Geelen. Computational methods for the diameter restricted minimum weight spanning tree problem. *Australasian Journal of Combinatorics*, 10:51–71, 1994.

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

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[7] L. Gouveia. Using the Miller-Tucker-Zemlin constraints to formulate a minimal spanning tree problem with hop constraints. *Computers and Operations Research*, 22(9):959–970, 1995.

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

[9] L. Gouveia, T. L. Magnanti, and C. Requejo. A 2-path approach for odd-diameter-constrained minimum spanning and Steiner trees. *Networks*. to appear 2004.

[10] B. A. Julstrom. Encoding bounded-diameter minimum spanning trees with permutations and with random keys. In K. Deb et al., editors, *Genetic and Evolutionary Computation - GECCO 2004*, volume 3102 of *LNCS*, pages 1282–1281. Springer, 2004.

[11] B. A. Julstrom. Improved greedy heuristics for the bounded-diameter minimum spanning tree problem. Technical report, St. Cloud State University, 2004. Submitted for journal publication.

[12] T. Magnanti and R. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1), 1984.

[13] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1988.

[14] M. W. Padberg and L. A. Wolsey. Trees and cuts. *Annals of Discrete Mathematics*, 17:511–517, 1983.

[15] 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, *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 747–752, New York, 2003. ACM Press.

[16] S. Talukdar, S. Murthy, and R. Akkiraju. Asynchronous teams. In *Handbook of Metaheuristics*, pages 537–556. Kluwer Academic Publishers, 2003.

[17] S. Voß. The Steiner tree problem with hop constraints. *Annals of Operations Research*, 86:321–345, 1999.