A Primal Branch-and-Cut Algorithm for the Degree-Constrained Minimum Spanning Tree Problem

Markus Behle[1], Michael Jünger[2⋆], and Frauke Liers[2⋆]

[1] Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
[2] Universität zu Köln, Institut für Informatik, Pohligstrasse 1, 50969 Köln, Germany

Abstract. The degree-constrained minimum spanning tree (DCMST) is relevant in the design of networks. It consists of finding a spanning tree whose nodes do not exceed a given maximum degree and whose total edge length is minimum. We design a primal branch-and-cut algorithm that solves instances of the problem to optimality. Primal methods have not been used extensively in the past, and their performance often could not compete with their standard 'dual' counterparts. We show that primal separation procedures yield good bounds for the DCMST problem. On several instances, the primal branch-and-cut program turns out to be competitive with other methods known in the literature. This shows the potential of the primal method.

1 Introduction

In recent years the development of networks in the area of telecommunication and computers has gained much in importance. One of the main goals in the design process is to reach total connectivity at minimum cost. Furthermore, additional constraints must be met, such as a restricted number of connections to a physical unit. Similar problems arise in the planning of road maps, where intersections can only be established among a small number of roads. In the field of integrated circuit design we are faced with the constraint that the number of wired connections to an electronic component is limited. The production of the circuit board shall be at minimum cost.

Problems of this kind can be modelled as the *degree-constrained minimum spanning tree* problem (or *DCMST problem* for short). Network devices, intersections, or electronic components are represented as nodes in a graph. Cables, roads or wires are represented by edges. A cost might be associated with an edge that models, e.g., the cable length or production cost. The technical restriction that the number of connections at a node is bounded is modeled by introducing constraints that bound the node degrees. Garey and Johnson [11] proved that the resulting degree-constrained minimum spanning tree problem is $\mathcal{NP}$-hard.

We are interested in exact solutions for instances of the DCMST problem with a 'primal' branch-and-cut method. Standard branch-and-cut methods can be viewed as 'dual' approaches in which cutting planes are added to iteratively improve the relaxation of the problem. In contrast, in a primal approach cutting planes are added to prove

⋆ Partially supported by the Marie Curie RTN Adonet 504438 funded by the EU and by the German Science Foundation under contracts Ju 204/8 and Li 1675/1 (FL).

the optimality of the best known feasible solution or to find a better solution. A primal approach has the advantage that the corresponding separation algorithms are often conceptionally easier than their dual versions. Furthermore, at any time during the run of the algorithm a feasible solution of the problem is known. Despite these advantages, primal methods have not yet received much attention in the literature. A possible reason is that the quality of the cutting planes depends on a known feasible solution. Up to now, in most cases the performance of primal approaches could not compete with the performance reachable by standard dual methods.

In this work we revise the primal cutting plane algorithm that Letchford and Lodi introduced in [20, 22, 21]. To the best of our knowledge, this algorithm has not been implemented before for the solution of practical problems. For the DCMST, our aim is to study the potential of the method by primal separation of known valid inequalities in a branch-and-cut algorithm. Some of the designed primal separation routines are asymptotically faster and much easier to implement than their dual versions. We also present a strategy for branching on a variable in the primal context.

The computational results show that the generated cutting planes are strong. On several classes of instances, the primal method outperforms the genetic algorithms that have been used by other authors. Furthermore, on some classes of instances the performance of the primal approach is comparable to a standard branch-and-cut method used by Raidl [30]. For the standard branch-and-cut algorithm in [6] no computational results on instances from the literature are reported.

In Section 2 we introduce the model. In Section 3 we explain the concept of primal separation together with a primal branch-and-cut algorithm for the DCMST problem. Finally, we show experimental results in Section 5 and discuss directions for further research.

2 Degree-constrained minimum spanning tree problem

We are given an undirected, connected graph $G = (V, E)$ with n nodes and m edges. For each edge $e \in E$ a cost $c_e \in \mathbb{R}$ is given. A minimum spanning tree of G is a connected acyclic subgraph $T = (V, E_T)$ that contains all nodes of G and has minimum cost $c(T) := \sum_{e \in E_T} c_e$ among all spanning trees. We define the neighbourhood of a node $i \in V$ as $\delta(i) := \{j \in V \mid \exists\, e = (i, j) \in E\}$. $|\delta(i)|$ is the degree of i in G. With $|\delta_T(i)|$ we denote the degree of i in the spanning tree T. To every node i we associate a capacity $b_i \in \mathbb{Z}$, where $1 \leq b_i \leq n - 1$. The *degree-constrained minimum spanning tree problem* asks for a minimum spanning tree T that satisfies the degree constraint $|\delta_T(i)| \leq b_i$ for all $i \in V$.

In the general setting, finding a DCMST is a hard task, and several, mainly heuristic, solution approaches exist in the literature. Narula and Ho [24] present a heuristic in which first a tree satisfying the degree constraints is generated that is not necessarily minimum. Then a local edge-exchange heuristic is called in order to improve on the cost of the tree while maintaining the node constraints. In a complementary approach, they start by constructing a minimum spanning tree and then repair the violation of degree constraints by exchanging edges. By the use of similar ideas, Savelsbergh and Volgenant [33] get better results than those cited in [24]. Ribeiro and Souza [32] im-

plement a variable neighbourhood search for generating good heuristic solutions for the DCMST. Recently, Andrade, Lucena and Maculan [1] presented a fast and effective heuristics for its solution. Using Lagrangian dual information, optimality of a solution can be proven in several cases. Goemans [13] designs a polynomial-time approximation algorithm for the DCMST problem with degree bound b for all vertices that finds a spanning tree of maximum degree at most $b + 2$ whose cost is at most the cost of the optimum spanning tree of maximum degree b. Volgenant [36], Knowles and Corne [17], Krishnamoorthy, Ernst and Sharaiha [18] and Raidl [31] present and compare several heuristics, simulated annealing approaches, evolutionary algorithms, Lagrangian relaxation and branch-and-bound methods on different classes of instances. Raidl [30] and Caccetta and Hill [6] also implement a standard branch-and-cut algorithm that solves the DCMST problem exactly. In this work, we are also interested in exact solutions. However, instead of solving the problem by standard branch-and-cut, we exploit the advantages of primal separation in a primal branch-and-cut approach. In the following section, we introduce the notion of primal separation and explain the primal separation algorithms.

3 Primal Separation

Let P_I be the polytope defined as the convex hull of all incidence vectors of feasible solutions of the problem. In a standard cutting plane procedure, we start by optimizing the objective function over a set of equations and inequalities that define a polytope that contains P_I and iteratively improve the relaxation of the problem by adding inequalities ("cutting planes"). To this end, we have to solve the standard separation problem that can be formulated as follows.

> **Standard separation problem** Given a point $x^* \in \mathbb{R}^m$. Return an inequality valid for P_I that is violated by x^* or prove that none exists.

Grötschel, Lovász and Schrijver [15], Karp and Papadimitriou [16] and Padberg and Rao [27] showed that an optimization problem is polynomial time equivalent to its separation problem. In contrast to an *exact* procedure, a heuristic separation algorithm not necessarily finds a violated inequality if one exists.

In the primal context, we are additionally given $\bar{x} \in \mathbb{R}^m$, a vertex of P_I. Usually, $\bar{x}$ represents the best known primal feasible solution. We formulate the primal separation problem as follows.

> **Primal separation problem** Given a point $x^* \in \mathbb{R}^m$ and a vertex $\bar{x}$ of P_I. Return an inequality valid for P_I that is violated by x^* and is tight at $\bar{x}$, or prove that none exists.

Padberg and Grötschel [25] showed that the primal separation problem is not harder than its standard version. For 0/1 polytopes, i.e., polytopes in which all vertices are vectors in $\{0, 1\}^m$, also the reverse holds true: Grötschel and Lovász [14] and Schulz, Weismantel and Ziegler [34] proved that the 0/1 optimization problem is polynomial time equivalent to the 0/1 augmentation problem. In the latter we are given a 0/1 point and either want to find a better feasible solution or prove that the given point is optimum. Finally, Eisenbrand, Rinaldi and Ventura [10] showed that the 0/1 augmentation

problem can be reduced to the primal 0/1 separation problem. For combinatorial optimization problems such as the DCMST problem, it follows that standard and primal separation are polynomial time equivalent.

Primal cutting plane algorithms have not yet received a lot of attention. In the early 1960s, Ben-Israel and Charnes used Gomory's method to develop a primal cutting plane algorithm for integer programming that was later simplified by Young [38]. Glover [12] and Arnold and Bellmore [4, 3, 2] modified the algorithm in order to reduce the number of degenerate pivots. However, only very small toy problems could be solved in practice. Sharma and Sharma [35] tried to improve the method but nevertheless could not compete with dual cutting plane algorithms. The first primal separation routines were developed in 1980 by Padberg and Hong [26] for the travelling salesman problem. In 1990, Barahona and Titan [5] and De Simone and Rinaldi [7] successfully applied primal cutting plane algorithms to the max-cut problem.

With the completion of the proof chain for the equivalence of the 0/1 primal and standard separation [10] and the work of Letchford and Lodi [20, 22, 21], the interest in primal separation increased again.

3.1 Primal Separation Algorithms for the Degree-Constrained Minimum Spanning Tree Problem

The DCMST problem can be formulated as the following 0/1 integer program.

$$
\begin{aligned}
&\min\ c^T x \\
&\text{s.t.} \sum_{e \in E(S)} x_e \leq |S| - 1 \ \forall\, S \subseteq V,\ |S| \geq 2 && (1)\\
&\quad \sum_{e \in E} x_e = |V| - 1 && (2)\\
&\quad \sum_{e \in \delta(i)} x_e \leq b_i \qquad \forall\, i \in V && (3)\\
&\quad x_e \in \{0,1\} \qquad \forall\, e \in E, && (4)
\end{aligned}
$$

where $E(S) := \{ij \in E \mid i, j \in S\}$.

Discarding the *node degree constraints* (3) leads to the minimum spanning tree problem which can be solved in polynomial time with e.g., the algorithms of Prim [29] and Kruskal [19]. Thus the node degree constraints (3) are well suited for a Lagrangian relaxation approach that Volgenant applied in [36]. Let the minimum spanning tree (MST) polytope consist of the convex hull of the incidence vectors of spanning trees. Its dimension is $m - 1$. For a complete graph, constraints (1) and $x_e \geq 0$ define facets. The *cycle elimination constraints* (1) are facets iff $|S| = 2$ and the graphs $(S, E(S))$ and $(V \setminus S, E(V \setminus S))$ are connected or $|S| \geq 3$ and the graph $(S, E(S))$ is 2-connected. Edmonds [8, 9] showed that the inequalities introduced above are sufficient to describe the MST polytope.

In addition to the cycle elimination constraints (1) we consider two more classes of valid inequalities. The *connectivity constraints*

$$\sum_{e \in \delta(S)} x_e \geq 1 \quad \forall\, S \subset V\colon\ 2 \leq |S| < |V|, \tag{5}$$

where $\delta(S) := \{ij \in E \mid i \in S, j \notin S\}$, assure that the graph is connected. The restriction of the sets S in (1) to cycles $(C, E(C))$ leads to the *specialized cycle elimination constraints*

$$\sum_{e \in E(C)} x_e \leq |C| - 1 \quad \forall\, C \subset V\colon\ |C| < |V| \text{ and } \big(C, E(C)\big) \text{ is a cycle} \tag{6}$$

We start with the relaxation consisting of equation (2), inequalities (3) and the relaxed integrality constraints $0 \leq x_e \leq 1\ \forall\, e \in E$. In the following we present exact primal separation algorithms for the constraint classes (1), (5) and (6) and compare them with their standard versions.

Connectivity constraints W.l.o.g. we only consider sets S for which the induced subgraph $(S, E(S))$ is connected. For the standard separation of (5), we temporarily set the edge costs according to x^*. If the value of the minimum cut in the corresponding graph is smaller than 1, a violated inequality is found. The runtime of this separation procedure depends on the chosen min-cut algorithm, e.g. is $O(mn + n^2 \log n)$ for the Nagamochi-Ibaraki [23] algorithm.

For the primal separation, we are additionally given a feasible solution $\bar{x}$ that corresponds to a spanning tree T in G. Candidates are only those sets $S \subset V$ such that $\sum_{e \in \delta(S)} \bar{x}_e = 1$ holds. Every edge e of the spanning tree T induces one such set S. Let $ij = e$ be an edge of T. Temporary deletion of e yields two trees, say T_i with root i and T_j with root j. Let $V_i \subset V$ resp. $V_j \subset V$ be the subset of all nodes that are incident to an edge of T_i resp. T_j. Since T is a spanning tree we have $V_i \,\dot\cup\, V_j = V$. W.l.o.g. set $S = V_i$. The primal separation for the connectivity constraints (5) is then the following.

For each of the $n-1$ edges ij of T compute the induced set V_i via depth first search in T. If the value $\sum_{e \in \delta(S_i)} x^*_e$ of the cut $\delta(S_i)$ is less than 1, a violated inequality is found. The runtime of the primal separation of connectivity constraints is $O(n(m+n))$, which is asymptotically faster than its standard counterpart.

Specialized cycle elimination constraints (6) can be written $\sum_{e \in E(C)} (1 - x_e) \geq 1$. In the standard separation, we temporarily set the edge cost of an edge e to $c'(e) = 1 - x^*_e$. A cycle with cost less than 1 in the corresponding graph determines a violated inequality in the standard sense. Thus, the separation routine amounts to computing for every edge $e = ij$ a shortest path from i to j. If we use Dijkstra's algorithm, the standard separation runs in time $O(mn^2)$.

For the primal separation of (6), we search for cycles C in G that additionally satisfy $\sum_{e \in E(C)} \bar{x}_e = |C| - 1$. Therefore, these cycles contain exactly one edge that is not part of the spanning tree given by $\bar{x}$. This observation immediately gives us the primal separation routine:

For every edge $e = ij$ in $E \setminus T$ find the path R from i to j in T by simultaneously going up level by level in the tree, starting from i and j, until a common predecessor is found. If the cost $c'(e) + c'(R)$ of the fundamental cycle $R \cup e$ is smaller than 1, a violated inequality is found. Primal separation for the specialized cycle elimination constraints runs in $O((m-n)n)$, which is again faster than the standard separation.

Cycle elimination constraints In the case of the traveling salesman problem in which all node degrees are forced to two, the cycle elimination constraints correspond to the subtour elimination constraints for which primal separation is asymptotically faster than its dual version, [26]. In the more general context of degree constrained spanning trees, Padberg and Wolsey presented in [28] a polynomial separation routine for constraints (1). An appropriate network for minimizing the submodular function $g(S) = |S| - \sum_{e \in E(S)} x^*_e$ is defined. In this network $n-2$ min-cuts are computed via max-flow calculations. Using the maximum-distance version of the Goldberg-Tarjan algorithm for the max-flow computations leads to an $O(n^4)$ runtime of the standard separation procedure.

For the primal separation routine, we modify the standard separation. We define a new submodular function $f : S \longrightarrow \mathbb{R}$ for $\emptyset \neq S \subseteq V$ which penalizes sets S that do not satisfy the condition $\sum_{e \in E(S)} \bar{x}_e = |S| - 1$. Let

$$f(S) = \begin{cases} g(S) & \text{if } \sum_{e \in E(S)} \bar{x}_e = |S| - 1 \\ > 1 & \text{otherwise} \end{cases}$$

Primal separation then amounts to finding a set S that minimizes $f(S)$. If the minimum is less than 1, we found a violated inequality in the primal sense.

We define f as

$$f(S) := |S| - \sum_{e \in E(S)} x^*_e + D\Big(|S| - 1 - \sum_{e \in E(S)} \bar{x}_e\Big)$$

with an appropriate constant $D > 0$. First we determine a large enough value for D such that $D > \sum_{e \in E(S)} x^*_e$ holds. $D := \frac{1}{2}(n^2 - n) + 1$ suffices.

It is not hard to see that f is submodular. Following the idea of Padberg and Wolsey we define a network (G^*, c') to find a set S minimizing f. G^* consists of the nodes V, a source node 0 and a sink node $n+1$. For all nodes $i \in V$ let $l_i := \sum_{e \in \delta(i)} (x^*_e + D\bar{x}_e)$. The edges of G^* are constructed as follows.

- For every edge $e = ij$ of E we add the directed edges (i, j) and (j, i) in G^* with costs $c'_{(i,j)} = c'_{(j,i)} = \frac{1}{2}(x^*_e + D\bar{x}_e)$.
- From the source 0 we add for every node $i \in V$ a directed edge $(0, i)$ with cost $c'_{(0,i)} = \max\{\frac{1}{2} l_i - (D+1), 0\}$.
- From every node $i \in V$ we add a directed edge $(i, n+1)$ to the sink $n+1$ with cost $c'_{(i,n+1)} = \max\{(D+1) - \frac{1}{2} l_i, 0\}$.

In the network G^* we consider a $(0, n+1)$-cut given by a set $S \subseteq V$ and calculate its costs $c'\big(S \cup \{0\} : \ (V \setminus S) \cup \{n+1\}\big)$. Let $L := \sum_{i \in V} \max\{\frac{1}{2} l_i - (D+1), 0\}$. The value of a $(0, n+1)$-cut in G^* is

$$
\begin{aligned}
& c'\big(S \cup \{0\} :\ (V \setminus S) \cup \{n+1\}\big) \\
&\quad = \sum_{i \in V \setminus S} \max\{\tfrac{1}{2} l_i - (D+1), 0\} + \sum_{i \in S} \max\{(D+1) - \tfrac{1}{2} l_i, 0\} + c'(S :\ V \setminus S) \\
&\quad = (D+1)|S| - \tfrac{1}{2} \sum_{i \in S} l_i + \tfrac{1}{2} \sum_{\substack{e=ij \\ i \in S, j \in V \setminus S}} (x^*_e + D\bar{x}_e) + L \\
&\quad = f(S) + D + L
\end{aligned}
$$

Thus, determining a set S which minimizes f amounts to minimizing the $(0, n+1)$-cut in (G^*, c'). Since S may not be the empty set, following Padberg and Wolsey, we need $n-2$ max-flow calculations to compute a minimum $(0, n+1)$-cut. The runtime of the primal separation asymptotically equals that of the standard separation. Padberg and Wolsey already noted the fact that the algorithm for eliminating the cycles is quite expensive, given that determining minimum spanning trees is an easy task. Furthermore, the primal separation routine for the cycles is neither asymptotically faster nor conceptionally easier than its standard version.

4 Primal Branch-and-Cut Algorithm

In this section, we describe a primal branch-and-cut algorithm for the DCMST problem. We use a modified version of the algorithm introduced by Letchford and Lodi in [21]. The algorithm starts with a feasible solution $\bar{x}$ generated with Narula and Ho's primal heuristic [24] which is a modified version of Prim's algorithm for MST. Subsequently, we improve the solution by the variable neighbourhood search [32].

For DCMST, the initial relaxation consists of the spanning tree equation (2), the node degree constraints (3) and the relaxed integrality constraints, see Section 3.1. Generated violated inequalities are added to the current relaxation of the problem. In case the separation procedures fail to determine a violated inequality, the algorithm branches. The following section introduces a primal branching rule.

4.1 Primal Branching Rule

In a primal branch-and-cut algorithm, we need in each subproblem a solution $\bar{x}$ that is feasible for it. Applying the standard rules for branching on a binary variable usually destroys feasibility of $\bar{x}$ in one of the two branches. Letchford and Lodi [21] presented a primal branching rule maintaining feasibility for the special case that $\bar{x}$ is identically **0** and all constraints in the relaxation are tight at $\bar{x}$. The former is no restriction for 0/1 integer linear programs. Relaxing the above mentioned restrictions, the branching rule in the spirit of Letchford and Lodi reads as follows. Let N_0 and N_1 be the set of indices of variables that are set to 0 resp. 1. Let a relaxation P be given as $P = \{x : Ax \leq b, 0 \leq x \leq 1\}$, and $\bar{x} \in P \cap \{0,1\}^n$. Branching according to N_0 and N_1 leads to P^1 which is the intersection of P with the subspaces of the variables x_j fixed to 0 resp. 1 for $j \in N_0, N_1$

$$P^1 = \left\{ x : \begin{array}{rl} Ax \leq b & \\ 0 \leq x_j \leq 1 & \forall j \notin N_0 \cup N_1 \\ x_j = 0 & \forall j \in N_0 \\ x_j = 1 & \forall j \in N_1 \end{array} \right\}$$

We maintain feasibility of $\bar{x}$ by optimizing in each subproblem over the convex hull of $P^1 \cup P^2$, where $P^2 := \{x : x = \bar{x}\}$. Let $\bar{a}^1_k := \sum_{j:\bar{x}_j=1} a_{kj}$. In the primal branching rule, we distinguish three cases.

1. (a) $\exists i : \bar{x}_i = 0 \wedge i \in N_1$. Set $y := x_i$.
   (b) $\exists i : \bar{x}_i = 1 \wedge i \in N_0$. Set $y := 1 - x_i$.
   Then

$$\text{conv}(P^1 \cup P^2) = \left\{ x : \begin{array}{rl} \sum_j a_{kj} x_j \leq \bar{a}^1_k - (\bar{a}^1_k - b_k) y & \forall k \\ 0 \leq x_j \leq y & \forall j : \bar{x}_j = 0 \wedge j \notin N_0 \cup N_1 \\ x_j = 0 & \forall j : \bar{x}_j = 0 \wedge j \in N_0 \\ x_j = y & \forall j \neq i : \bar{x}_j = 0 \wedge j \in N_1 \\ 1 - y \leq x_j \leq 1 & \forall j : \bar{x}_j = 1 \wedge j \notin N_0 \cup N_1 \\ x_j = 1 - y & \forall j \neq i : \bar{x}_j = 1 \wedge j \in N_0 \\ x_j = 1 & \forall j : \bar{x}_j = 1 \wedge j \in N_1 \\ 0 \leq y \leq 1 & \end{array} \right\}$$

2. $\nexists i : \bar{x}_i = 0 \wedge i \in N_1$ and $\nexists i : \bar{x}_i = 1 \wedge i \in N_0$. Then $\text{conv}(P^1 \cup P^2) = P^1$.

Whenever a new branch-and-bound node is created, the subproblem relaxation is modified according to the corresponding case above.

4.2 Outline of the Primal Branch-and-Cut Algorithm

In every node of the branch-and-bound tree valid violated cutting planes are generated. The used cutting plane procedure is a modification of the one developed by Letchford and Lodi in [20]. We describe it in the following.

Let the result of a primal simplex iteration be x^*. In case x^* is feasible for the DCMST problem and improves upon the formerly best known solution, a new degree-constrained spanning tree $\bar{x}$ is found. If $\bar{x}$ is dual feasible, the current subproblem is optimized and we can fathom the node. In case x^* is not binary or contains a cycle, the primal separation routines are called. In contrast to a standard cutting plane algorithm it is possible that no primal separating hyperplane for a binary but infeasible point x^* can be generated. This may happen e.g. if we pivot from $\bar{x}$ to a 0/1 vertex of the relaxation which is feasible for all inequalities that can be generated by primal separation. In this case we aim at cutting off the infeasible point by standard separation routines. In case of failure, we separate a fractional point x^* by a Chvátal-Gomory cut that is tight at $\bar{x}$ and only contains integral coefficients. So at all times our matrix A is integral.

We generate Chvátal-Gomory cuts in the following way. Without loss of generality, we consider a linear relaxation of the form $P = \{x : Ax \leq b, x \geq 0\}$ with $A \in \mathbb{Q}^{m \times n}$. Let $A_{\cdot k}$ denote the k-th column of A. For $u \in \mathbb{Q}^m_{\geq 0}$ the inequality $\sum_{k=1}^{n} \lfloor u^T A_{\cdot k} \rfloor x_k \leq \lfloor u^T b \rfloor$ is valid for P_I, see e.g. [37]. The coefficients of this Chvátal-Gomory cut are integer. For each row i of the inverse of the basis matrix A_B we define u_i as the fractional part of that row, i.e. $u_i := e_i^T A_B^{-1} - \lfloor e_i^T A_B^{-1} \rfloor$. Now for each u_i we generate the Chvátal-Gomory cut and check if it is tight at $\bar{x}$.

The primal cutting plane procedure for the solution of a node in the branch-and-bound tree for the DCMST problem works as follows. Its generalization to other problems is obvious.

- Step 1: Compute a primal feasible basis for the basic solution $\bar{x}$.
- Step 2: Perform a primal simplex pivot. Let x^* be the new solution.
- Step 3: If x^* is binary and represents a DCMST, set $\bar{x} = x^*$. If $\bar{x}$ is optimal, fathom the node. Otherwise go to Step 2.
- Step 4: Call the primal separation. If violated inequalities are generated, pivot back to $\bar{x}$, add them to the LP and go to Step 2.
- Step 5: If x^* is binary but contains a cycle, generate a separating hyperplane that is not tight at $\bar{x}$, add it to the LP and go to Step 1.
- Step 6: Determine a Chvátal-Gomory cut that is tight at $\bar{x}$ and only contains integral coefficients. If such a cut is found, pivot back to $\bar{x}$, add it to the LP and go to Step 2. Otherwise branch according to the primal branching rule outlined in section 4.1.

The difference between this algorithm and the one introduced by Letchford and Lodi mainly lies in Step 5. It is called Step 5b in their enhanced algorithm. In case a binary but infeasible point x^* is found, Letchford and Lodi generate a cutting plane that is not tight at $\bar{x}$. Then the dual simplex algorithm is used to compute a new fractional point $\hat{x}$ that is a convex combination of $\bar{x}$ and x^*. Then they set $x^* := \hat{x}$ and remove the just inserted cutting plane as the mixed-integer cut generated in the next step will cut off $\hat{x}$ and thus also the old binary solution.

In contrast, for esthetical reasons we decided not to switch to the dual simplex but to always stick to its primal version throughout the computations. We also cut off the infeasible point x^* with a standard cutting plane and thus lose primal feasibility of the tableau. As we need to stay at $\bar{x}$, we compute in Step 1 an appropriate primal feasible basis. To this end, we temporarily introduce a new objective function c' with entries $c'_i := 1 - 2\bar{x}_i$. Minimizing the new objective function by the primal simplex algorithm yields the optimum solution $\bar{x}$ and a primal feasible tableau. Replacing the objective function by c again keeps its primal feasibility.

5 Experimental Results

We implemented the primal branch-and-bound algorithm from Section 4.2 and the primal separation routines outlined in Section 3.1 in a straightforward way. When primal separation is called in Step 4, we first try to find a violated specialized cycle elimination constraint (6). If we do not find one we try to find a violated connectivity constraint (5)

and if we still do not succeed, we try to separate a cycle elimination constraint (1). For performance reasons we do not separate cycle elimination constraints in the way we describe in 3.1 but set up a pool of cycle elimination constraints tight at $\bar{x}$ whenever we find a new $\bar{x}$. We do a pool separation. The size of this pool depends on the problem structure and varies from 10000 up to 200000 cuts.

In Step 5 we use the standard separation for the specialized cycle elimination constraints (6). If we arrive at Step 5, the binary solution contains a cycle that the primal separation was not able to separate. So we will always find a violated cut in Step 5. In case that none of the former separation routines found a cut we invoke the Chvátal-Gomory separation. If still no cut can be found or if the number of Chvátal-Gomory cuts in the actual node of the branch-and-bound tree exceeds 10, we branch. This prevents primal degeneracy.

We took instances from Knowles and Corne [17] and Krishnamoorthy, Ernst and Sharaiha [18] which are grouped into the 6 classes R, M, CRD, SYM, STR and SHRD. From these classes, we took all instances with at most 100 nodes resulting in 234 instances in total. All computations were done on a Pentium 4 processor under Linux. We use CPLEX 9.0 as linear program solver. The quality of the solution is measured by the gap, i.e. the difference of the primal and the dual bounds divided by the primal one. For each instance, we set an upper limit of four hours of cpu time. Within this time interval, 228 out of the 234 instances could be solved to optimality. For the remaining 6 instances, the gaps are always smaller than 6.3%. In the tables n is the number of nodes.

				PBC							SBC	GA
	n	b	Type	B&B nodes	height	Pcuts	Scuts	CGcuts	time	Q	Q	Q
R	50	5	n1	1	0	8	0	0	1.62	1.61	-	1.74
R	50	5	n2	1	0	26	6	1	1.61	1.69	-	1.83
R	50	5	n3	1	0	11	3	1	3.59	1.62	-	1.78
R	100	5	n1	21	5	45	178	10	94.16	1.61	-	1.73
R	100	5	n2	1543	24	80	4144	182	5525.55	1.50	-	1.60
R	100	5	n3	2	1	22	16	1	14.08	1.55	-	1.69
M *	50	5	n1	172	34	1690	105	141	14400	2.45	2.45	3.15
M	50	5	n2	3	1	707	17	0	591.82	2.21	2.21	2.99
M	50	5	n3	12	4	615	168	22	130.07	2.36	2.36	2.58
M	100	5	n1	14	7	251	223	11	1471.3	1.98	1.98	2.44
M	100	5	n2	1580	32	1299	10705	676	10950.1	2.08	2.08	2.84
M	100	5	n3	160	29	710	1246	111	2041.48	1.98	1.98	2.86

Table 1. R and M instances

All nodes have the same upper bound b on the node degree. Our algorithm can also handle individual node degree constraints. The time spent in the branch-and-cut process is given in seconds. The time spent in the starting heuristics that generate a good feasible solution $\bar{x}$ is usually less than a second. The primal branch-and-cut algorithm is named *PBC*. *B&B nodes* shows the number of branch-and-bound nodes and *height* the height of the branch-and-bound tree. The number of generated violated primal, standard resp. Chvátal-Gomory cuts are given in the columns *Pcuts*, *Scuts* resp. *CGcuts*.

The R and M instances are taken from Knowles and Corne [17]. Their *GA* heuristic performs well on these instances. Results are averaged over 20 passes and are given as

a quotient Q which is the cost of the found degree-constrained spanning tree divided by the cost of the minimum spanning tree. Running times are not published. The primal PBC algorithm solves the R instances (see table 1) with 50 nodes in less than 4 seconds to optimality. For the larger instances the program has to branch. The class of the M instances is constructed by Knowles and Corne such that the minimum spanning tree and the DCMST are very different which causes easy greedy heuristics to fail. The M instances (see table 1) were also tackled by Raidl [30] with a standard branch-and-cut approach which we name *SBC*. However, a direct comparison between the PBC and SBC algorithms is difficult as Lagrangian relaxation was used in the latter. Using the PBC algorithm, 5 out of the 6 instances could be solved to optimality within the given time interval. For the instance marked with an asterisk the gap at the time limit was 4.1%. With their heuristics, Andrade, Lucena and Maculan [1] can solve the same R and M instances in less than 30 seconds each and furthermore prove optimality of their solution.

				PBC				BB
type	n	b	B&B nodes	height	Pcuts	Scuts	CGcuts	time	Gap
CRD	30	3	1	0	4.5	0	0	0.11	0.70
	50	3	1	0	10.8	0	0	0.42	0.01
	70	3	3.4	1.6	42.5	13.5	24.5	54.54	0.01
	100	3	5.89	1.11	115.4	5.9	48.8	290.8	0.04
SYM	30	3	1	0	5	0	1.7	0.49	4.26
	30	4	1	0	4	0	0.7	0.17	0.16
	50	3	33.3	4.8	41.9	24.4	317	124.94	4.78
	50	4	142.4	3.6	34.5	15.7	147.8	107.91	0.95
	50	5	1	0	1.6	0	0	0.11	0.10
	70	3	36	5.2	37.8	9.4	361.8	77.35	5.60
	70	4	174.7	4.3	31.3	20.9	522.1	284.93	1.03
	70	5	153.2	2.5	15.3	12.2	160.6	219.37	0.11
STR	30	4	1	0	25.2	0	0	0.2	13.45
	30	5	1	0	28.8	0	0	0.28	10.8
	50	4	2.6	0.4	96.6	2.2	18.9	4.3	12.31
	50	5	3.4	0.6	105.2	2.5	26.4	4.8	9.88
	70	4	1.7	0.4	268	0.3	11.5	24.17	11.49
	70	5	9.2	0.4	263.3	2.9	88.2	38.75	9.21
	100	3	5.7	0.8	274.3	0.2	3.3	1251.1	12.85
	100	4	1	0	544.63	0	0	171.84	10.59
	100	5	1	0	555.25	0.13	0.75	79.45	8.48

Table 2. CRD, SYM and STR instances

The four instance classes CRD, SYM, STR and SHRD are taken from Krishnamoorthy, Ernst and Sharaiha [18]. We compare our algorithm with their branch-and-bound approach called *BB* with a time limit of 600 seconds. The values in the table 2 are the means of 10 different instances with the same number of nodes and the same b. The CRD instances are two-dimensional Euclidean graphs. The SYM instances are random multi-dimensional Euclidean graphs. Usually these instances are solved within seconds with PBC. One CRD instance with 100 nodes exceeded the time limit reaching a gap of

$3.4 \cdot 10^{-3}\%$. Instances that exceeded the time limit were excluded from the averages. The instances of the STR class usually cannot be solved to optimality by the branch-and-bound approach of Krishnamoorthy, Ernst and Sharaiha with a time limit of 600 seconds [18]. Except for 4 large instances out of the 100, we can solve all instances to optimality. The gaps for these 4 instances range from 3.2% to 6.3%. 90 instances could be solved within 600 seconds. The small instances are solved within seconds.

Krishnamoorthy, Ernst and Sharaiha claim that the instances from the SHRD class are the hardest since none of its members could be solved with their branch-and-bound approach within 600 seconds. We compare our results with their best heuristic *PSS*. In contrast to PSS, Raidl's approach [30] and our algorithm did not have any problems (see table 3). We could solve every instance to optimality in less than 21 seconds. Again, with their heuristics, Andrade, Lucena and Maculan are able to solve SHRD instances with up to 309 nodes in less than 30 seconds per instance. They can additionally prove optimality.

		PBC						PSS
n	b	B&B nodes	height	Pcuts	Scuts	CGcuts	time	Gap
15	3	1	0	3	4	0	0.06	1.72
15	4	1	0	0	0	0	0.26	2.08
15	5	1	0	0	0	0	0.18	0
20	3	2	1	18	16	0	1.53	0.45
20	4	1	0	5	1	1	2	0
20	5	1	0	0	0	0	1.58	0.47
25	3	1	0	1	0	0	5.73	0
25	4	1	0	2	1	0	4.82	0
25	5	1	0	0	0	0	1.87	1.07
30	3	4	2	302	33	2	20.25	0
30	4	2	1	15	16	1	7.55	0
30	5	2	1	47	16	0	4.02	0

Table 3. SHRD instances

As can be expected, the quality of the starting feasible solution given by a heuristic is important for the performance of the primal branch-and-cut process. Usually, the closer the objective value of the solution is to the optimal objective value, the faster the process terminates. The usage of the primal cutting planes varies from instance to instance, not only from class to class. For some instances it is useful to spend more time in the separation whereas others are solved faster if less cutting planes are generated and more branching is done. The special cycle elimination constraints and the connectivity constraints can be separated very fast but are usually weaker than the cycle elimination constraints that turned out to be the most important constraints for the tested instances. We do not perform an exact separation of this class but set up a pool of constraints that is searched for a violated one. We observed that the larger the pool is, the less standard and Chvátal-Gomory cuts have to be generated. As we do not refill this pool until a new $\bar{x}$ is found, the longer we stay at a certain $\bar{x}$ the more standard and Chvátal-Gomory cuts are needed. This effect can be observed for the large R and M instances.

6 Conclusion

In this paper we presented primal separation routines for the DCMST problem and implemented them in primal branch-and-cut procedure. We showed that primal separation can be conceptionally easier and theoretically faster than the standard dual separation. The computational results show that primal methods can compete with existing approaches. Research has been undertaken in more depth for dual than for primal methods yet, and several questions remain to be answered in the primal context such as, e.g., what are the most effective branching rules, rules for chosing the branching variables, etc. We believe that there is potential for primal branch-and-cut methods to become competitive with dual methods also for other hard problems.

Acknowledgments

The authors thank A. Letchford and A. Lodi for fruitful discussions.

References

1. R. Andrade, A. Lucena, and N. Maculan. Using lagrangian dual information to generate degree constrained spanning trees. *Discrete Applied Mathematics*, 154(5):703–717, 2006.
2. L. R. Arnold and M. Bellmore. A bounding minimization problem for primal integer programming. *Operations Research*, 22:383–392, 1974.
3. L. R. Arnold and M. Bellmore. A generated cut for primal integer programming. *Operations Research*, 22:137–143, 1974.
4. L. R. Arnold and M. Bellmore. Iteration skipping in primal integer programming. *Operations Research*, 22:129–136, 1974.
5. F. Barahona and H. Titan. Max mean cuts and max cuts. In *Combinatorial Optimization in Science and Technology*, pages 30–45, 1991.
6. L. Caccetta and S. P. Hill. A branch and cut method for the degree-constrained minimum spanning tree problem. *Networks*, 37(2):74–83, 2001.
7. C. De Simone and G. Rinaldi. A cutting plane algorithm for the max-cut problem. *Optimization Methods and Software*, 3:195–214, 1994.
8. J. Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and their Applications*, pages 69–87, New York, 1970. Gordon and Breach.
9. J. Edmonds. Matroids and the greedy algorithm. *Math. Programming*, 1:127–136, 1971.
10. F. Eisenbrand, G. Rinaldi, and P. Ventura. 0/1 optimization and 0/1 primal separation are equivalent. In *Proceedings of the 13th annual ACM-SIAM symposium on discrete algorithms, SODA '02*, pages 920–926, 2002.
11. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
12. F. Glover. A new foundation for a simplified primal integer programming algorithm. *Operations Research*, 16:727–740, 1968.
13. M. X. Goemans. Minimum bounded-degree spanning trees. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 273–282, 2006.
14. M. Grötschel and L. Lovász. *Handbook of Combinatorics*, volume 2, chapter Combinatorial Optimization, pages 1541–1597. North Holland, 1995.
15. M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

16. R. M. Karp and C. H. Papadimitriou. On linear characterizations of combinatorial optimization problems. In *21st Annual Symposium on Foundations of Computer Science*, pages 1–9, Syracuse, New York, 1980.
17. J. D. Knowles and D. W. Corne. A new evolutionary approach to the degree-constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, 4(2):125–134, 2000.
18. M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha. Comparison of algorithms for the degree constrained minimum spanning tree. *Journal of Heuristics*, 7:587–611, 2001.
19. J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematics Society*, 7(1):48–50, 1956.
20. A. N. Letchford and A. Lodi. Primal cutting plane algorithms revisited. *Mathematical Methods of Operations Research*, 56(1):67–81, 2002.
21. A. N. Letchford and A. Lodi. An augment-and-branch-and-cut framework for mixed 0-1 programming. In M. Jünger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization: Eureka, You Shrink!*, volume 2570 of *Lecture Notes in Computer Science*. Springer, 2003.
22. A. N. Letchford and A. Lodi. Primal separation algorithms. *4OR*, 1(3):209–224, 2003.
23. H. Nagamochi and T. Ibaraki. Computing edge connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics*, 5:54–66, 1992.
24. S. C. Narula and C. A. Ho. Degree-constrained minimum spanning tree. *Computers & Operations Research*, 7:239–249, 1980.
25. M. W. Padberg and M. Grötschel. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, chapter Polyhedral computations, pages 307–360. John Wiley & Sons, 1985.
26. M. W. Padberg and S. Hong. On the symmetric travelling salesman problem: a computational study. *Mathematical Programming Study*, 12:78–107, 1980.
27. M. W. Padberg and M. R. Rao. The russian method for linear programming III: Bounded integer programming. Technical Report 81-39, Graduate School of Business and Administration, New York University, 1981.
28. M. W. Padberg and L. A. Wolsey. Trees and cuts. *Annals of Discrete Mathematics*, 17:511–517, 1983.
29. R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.
30. G. R. Raidl. personal communication.
31. G. R. Raidl. An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, volume 1, pages 104–111, 2000.
32. C. C. Ribeiro and M. C. Souza. Variable neighborhood search for the degree-constrained minimum spanning tree problem. *Discrete Applied Mathematics*, 118(1-2):43 – 54, 2002.
33. M. Savelsbergh and T. Volgenant. Edge exchanges in the degree-constrained minimum spanning tree problem. *Computers & Operations Research*, 12:341–348, 1985.
34. A. S. Schulz, R. Weismantel, and G. M. Ziegler. 0/1 integer programming: Optimization and augmentation are equivalent. In P. Spirakis, editor, *ESA '95*, volume 979 of *Lecture Notes in Computer Science*, pages 473–483. Springer, 1995.
35. S. Sharma and B. Sharma. New technique for solving primal all-integer linear programming. *Opsearch*, 34:62–68, 1997.
36. A. Volgenant. A lagrangean approach to the degree-constrained minimum spanning tree problem. *European Journal of Operational Research*, 39:325–331, 1989.
37. L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.
38. R. D. Young. A simplified primal (all-integer) integer programming algorithm. *Operations Research*, 16:750–782, 1968.