Chain-Constrained Spanning Trees*

Neil Olver[1] and Rico Zenklusen[2]

[1] MIT, Cambridge, USA
olver@math.mit.edu
[2] Johns Hopkins University, Baltimore, USA
ricoz@jhu.edu

Abstract. We consider the problem of finding a spanning tree satisfying a family of additional constraints. Several settings have been considered previously, the most famous being the problem of finding a spanning tree with degree constraints. Since the problem is hard, the goal is typically to find a spanning tree that violates the constraints as little as possible.

Iterative rounding became the tool of choice for constrained spanning tree problems. However, iterative rounding approaches are very hard to adapt to settings where an edge can be part of a super-constant number of constraints. We consider a natural constrained spanning tree problem of this type, namely where upper bounds are imposed on a family of cuts forming a chain. Our approach reduces the problem to a family of independent matroid intersection problems, leading to a spanning tree that violates each constraint by a factor of at most 9.

We also present strong hardness results: among other implications, these are the first to show, in the setting of a basic constrained spanning tree problem, a qualitative difference between what can be achieved when allowing multiplicative as opposed to additive constraint violations.

1 Introduction

Spanning tree problems with additional $\{0,1\}$-packing constraints have spawned considerable interest recently. This development was motivated by a variety of applications, including VLSI design, vehicle routing, and applications in communication networks [8,4,14]. Since even finding a feasible solution of a constrained spanning tree problem is typically NP-hard, the focus is on efficient procedures that either certify that the problem has no feasible solution, or find a spanning tree that violates the additional constraints by as little as possible. Often, an objective function to be minimized is also provided; here, however, we focus just on minimizing the constraint violations.

A wide variety of constrained spanning tree problems have been studied. Unfortunately, for most settings, little is known about what violation of the constraints must be accepted in order that a solution can be efficiently attained. An exception is the most classical problem in this context, the degree-bounded spanning tree problem. Here the goal is to find a spanning tree $T \subseteq E$ in a

* This project was supported by NSF grant CCF-1115849.

M. Goemans and J. Correa (Eds.): IPCO 2013, LNCS 7801, pp. 324–335, 2013.

graph $G = (V, E)$ such that T satisfies a degree bound for each vertex $v \in V$, i.e., $|\delta(v) \cap T| \leq b_v$. For this problem, Fürer and Raghavachari [8] presented an essentially best possible algorithm that either shows that no spanning tree satisfying the degree constraints exists, or returns a spanning tree violating each degree constraint by at most 1. We call this an *additive* 1-*approximation*, by contrast to an α-approximation, where each constraint can be violated by a *factor* $\alpha > 1$.

Recently, iterative rounding/relaxation algorithms became the tool of choice for dealing with constrained spanning tree problems. A cornerstone for this development was the work of Singh and Lau [16], which extended the iterative rounding framework of Jain [10] with a relaxation step. They obtained an additive 1-approximation even for the *minimum* degree-bounded spanning tree problem, i.e., the cost of the tree they return is upper bounded by the cost of an optimal solution not violating any constraints. This result was the culmination of a long sequence of papers presenting methods with various trade-offs between constraint violation and cost (see [11,12,5,6,9] and references therein).

Singh and Lau's iterative relaxation technique was later generalized by Bansal, Khandekar and Nagarajan [3], to show that even when upper bounds are given on an arbitrary family of edge-sets $E_1, \ldots, E_k$, one can still find a (min cost) spanning tree violating each constraint by at most $\max_{e \in E} |\{i \in [k] \mid e \in E_i\}| - 1$. If each edge is only in a constant number of constraints, this leads to an additive $O(1)$-approximation. But extending the iterative rounding technique beyond such settings seems to typically be very difficult. Some progress was achieved by Bansal, Khandekar, Könemann, Nagarajan and Peis [2], who used an iterative approach that iteratively replaces constraints by weaker ones, leading to an additive $O(\log n)$-approximation if the constraints are upper bounds on a laminar family of cuts. They left open whether an additive or multiplicative $O(1)$-approximation is possible in this setting, even when the cuts form a chain. Recently, Zenklusen [17] presented an additive $O(1)$-approximation for generalized degree bounds, where for each vertex an arbitrary matroid constraint on its adjacent edges has to be satisfied. This algorithms differs from previous iterative rounding approaches in that it successively simplifies the problem to reach a matroid intersection problem, rather than attempting to eliminate constraints until only spanning tree constraints remain.

To the best of our knowledge, with the exception of the setting of Zenklusen [17], no $O(1)$-approximations are known for constrained spanning tree problems where an edge can lie in a super-constant number of (linear) constraints. This seems to be an important difficulty that current techniques have trouble overcoming. Furthermore, in many settings, it is not well understood if finding an additive approximation is any harder than a multiplicative one. In particular, no constrained spanning tree problem was previously known where an $O(1)$-approximation is possible, but an additive $O(1)$-approximation is not. The goal of this paper is to address these points by studying chain-constrained spanning trees—a natural constrained spanning tree problem that evades current techniques.

1.1 Our Results

In this paper, we consider what is arguably one of the most natural constraint families for which finding $O(1)$-approximations seems beyond current techniques—chain constraints. Here we are given an undirected graph $G = (V, E)$ together with a family of cuts $\emptyset \subsetneq S_1 \subsetneq S_2, \ldots, \subsetneq S_\ell \subsetneq V$ forming a chain, and bounds $b_1, \ldots, b_\ell \in \mathbb{Z}_{>0}$. In summary, our reference problem is the following.

$$\begin{gathered}\text{Find a spanning tree } T \in \mathcal{T} \text{ satisfying:} \\ |T \cap \delta(S_i)| \leq b_i \quad \forall i \in [\ell],\end{gathered} \tag{1}$$

where $\mathcal{T} \subseteq 2^E$ is the family of all spanning trees of G, and $\delta(S_i) \subseteq E$ denotes all edges with precisely one endpoint in S_i.

Notice that chain constraints allow edges to be in a super-constant number of constraints. They are also a natural candidate problem that captures many of the difficulties faced when trying to construct $O(1)$-approximations for the laminar case. Our main algorithmic result is the following.

Theorem 1. *There is an efficient* 9*-approximation for the chain-constrained spanning tree problem.*

Contrary to previous procedures, our method is not based on iterative rounding. Instead, we reduce the problem to a family of independent matroid intersection problems. More precisely, we rely on a subprocedure that works in graphs without *rainbows*, by which we mean a pair of edges e, f such that e is in a proper superset of the chain constraints in which f is contained. To reach a rainbow-free setting, we solve a natural LP relaxation with an appropriately chosen objective function. We then show that one can consider a maximal family of linearly independent and tight spanning tree constraints to decompose the problem into rainbow-free subproblems, one for each chosen spanning tree constraint. Even though the high-level approach is quite clean, there are several difficulties we have to address. In particular, to do the accounting of how much a constraint is violated across all subproblems, we define a well-chosen requirement that the solutions of the subproblems must fulfill, and which allows us to bound the total violation of a constraint. Due to space constraints, details will appear in a long version of this paper.

Our main result on the hardness side is the following.

Theorem 2. *For the chain-constrained spanning tree problem it is NP-hard to distinguish between the cases where a spanning tree satisfying the chain constraints exists, and the case that every spanning tree violates some degree bound by* $\Omega(\log n / \log \log n)$ *units.*

This result has several interesting implications. First, it shows that even for chain constraints there is a clear qualitative difference between what can be achieved when considering additive versus multiplicative violation. Hence, Theorem 2 together with Theorem 1 show for the first time that there are constrained spanning tree problems where an additive $O(1)$-approximation would imply P = NP

whereas an $O(1)$-approximation exists. Previously, the only hardness result of a similar nature to Theorem 2 was presented by Bansal et al. [2], for a very general constrained spanning tree problem, where constraints $|T \cap E_i| \leq b_i \; \forall i \in [k]$ are given for an arbitrary family of edge sets $E_1, \ldots, E_k \subseteq E$. They showed that unless NP has quasi-polynomial time algorithms, there is no additive $(\log^c n)$-approximation for this case, for some small constant $c \in (0, 1)$. Notice that our hardness result is stronger in terms of the approximation ratio, the underlying constrained spanning tree model, and the complexity assumption. Furthermore, Theorem 2 shows that the additive $O(\log n / \log \log n)$-approximation of Bansal et al. [2] for the laminar-constrained spanning tree problem is close to optimal.

Due to space constraints, the proof of Theorem 2 and further results on hardness and integrality gaps will appear in a long version of this paper.

1.2 Related Work

The problem of finding a *thin* tree, which recently came to fame, can be interpreted as a constrained spanning tree problem. Here, upper bounds are imposed on the number of edges to be chosen in any cut of the graph. More precisely, given a point x in the spanning tree polytope of G, a spanning tree T is α-thin with respect to x if $|T \cap \delta(S)| \leq \alpha \cdot x(\delta(S)) \; \forall S \subseteq V$. For the thin spanning tree problem the currently best known procedures only lead to a thinness of $\alpha = \Theta(\log n / \log \log n)$ [1,7]. The concept of thin spanning trees gained considerably in relevance when Asadpour et al. [1] showed that an efficient algorithm for finding an α-thin spanning tree leads to an $O(\alpha)$-approximation for the Asymmetric Traveling Salesman Problem (ATSP)[1]. Using this connection they obtained the currently best approximation algorithm for ATSP with an approximation factor of $O(\log n / \log \log n)$. It is open whether $O(1)$-thin spanning trees exist, which would immediately imply an $O(1)$-factor approximation for ATSP.

2 The Algorithm

To simplify the exposition, we assume that we are dealing with a maximal chain of constraints imposed on the spanning tree. Hence, we can choose a numbering of the vertices $V = \{v_1, \ldots, v_n\}$ of the graph $G = (V, E)$ such that we have a constraint $|T \cap \delta(S_i)| \leq b_i$ for $S_i = \{v_1, \cdots, v_i\} \; \forall \; i \in [n-1]$. This is clearly not restrictive since by choosing a large right-hand side, any constraint can be made redundant.

Our algorithms starts by computing an optimal solution to the natural LP relaxation of Problem (1), that asks to find a point in the spanning tree polytope P_{ST} of G satisfying the chain constraints. More precisely, we do not want to start with an arbitrary feasible solution to this LP, but with one that minimizes the

[1] Strictly speaking, Asadpour et al.'s approach required the spanning tree not only to be thin, but also to be of low cost. However this second requirement is not necessary for the mentioned statement to be true (see [13]).

total length of the edges, where the length of an edge $\{v_i, v_j\} \in E$ is $|i-j|$, i.e., the number of chain constraints to which the edge contributes. This leads to the LP (2) shown below. Let x^* be an optimal solution to (2), which can be computed by standard techniques (see [15]). Notice that the objective function of (2) is the same as the total load on all cuts: $\sum_{i=1}^{n-1} x(\delta(S_i))$.

$$\begin{aligned} \min \sum_{\{v_i,v_j\}\in E} |i-j| \cdot x(\{v_i, v_j\}) \\ x \in P_{ST} \\ x(\delta(S_i)) \leq b_i \qquad \forall\, i \in [n-1] \end{aligned} \tag{2}$$

The above objective function is motivated by a subprocedure we use to find a spanning tree in an instance that does not contain what we call a *rainbow*. A rainbow consists of two edges $\{v_i, v_j\}, \{v_p, v_q\}$ with $i \leq p < q \leq j$ and either $i < p$ or $q < j$, i.e., the first edge is in a proper superset of the chain constraints in which the second edge is in. Even though the above objective function does not necessarily lead to an LP solution x^* whose support $\text{supp}(x^*) = \{e \in E \mid x^*(e) > 0\}$ does not contain rainbows—a feasible rainbow-free solution may not even exist—it eliminates rainbows in subproblems we are interested in, as we will see later. Clearly, if LP (2) is not feasible, we know that the reference problem has no feasible solution.

In all what follows, we only work on edges in $\text{supp}(x^*)$. Therefore, to simplify the exposition, we assume from now on that $E = \text{supp}(x^*)$. This can easily be achieved by deleting all edges $e \in E$ with $x^*(e) = 0$ from G.

Our algorithm decomposes the problem of finding an $O(1)$-approximate spanning tree $T \subseteq E$ into an independent family of a special type of spanning tree problem on rainbow-free graphs. To decompose the problem, we consider tight spanning tree constraints. More precisely, let $\mathcal{L} \subseteq 2^V$ be any maximal laminar family of vertex-sets corresponding to spanning tree constraints that are tight with respect to x^*. In other words, $\mathcal{L}$ is maximal laminar family chosen from the sets $L \subseteq V$ satisfying $x^*(E[L]) = |L| - 1$, where, $E[L] \subseteq E$ denotes the set of edges with both endpoints in L. In particular, $\mathcal{L}$ contains all singletons. We say that $L_2 \in \mathcal{L}$ is a child of $L_1 \in \mathcal{L}$ if $L_2 \subsetneq L_1$ and there is no set $L_3 \in \mathcal{L}$ with $L_2 \subsetneq L_3 \subsetneq L_1$. For $L \in \mathcal{L}$, we denote by $\mathcal{C}(L) \subset \mathcal{L}$ the set of all children of L. Notice that $\mathcal{C}(L)$ forms a partition of L.

To construct a spanning tree T in G we will determine for each $L \in \mathcal{L}$ a set of edges T_L in

$$E_L := E[L] \setminus (\cup_{C \in \mathcal{C}(L)} E[C]),$$

that form a spanning tree in the graph G_L obtained from the graph (L, E_L) by contracting all children of L. Hence, the vertex set of G_L is $\mathcal{C}(L)$, and an original edge $\{u, v\} \in E_L$ is simply interpreted as an edge between the two children $C_u, C_v \in \mathcal{C}(L)$ that contain u and v, respectively. For singletons $L \in \mathcal{L}$, we set $T_L = \emptyset$. One can easily observe that a family $\{T_L\}_{L \in \mathcal{L}}$ of spanning trees in $\{G_L\}_{L \in \mathcal{L}}$ leads to a spanning tree $T = \cup_{L \in \mathcal{L}} T_L$ in G. Constructing "good" spanning trees T_L in G_L, for each $L \in \mathcal{L}$, will be our independent subproblems.

As we will argue more formally later, the main benefit of this division is that the edge set E_L used in the subproblem to find T_L does not contain any rainbows. Our goal is to define constraints that the spanning trees T_L have to satisfy, that allow us to conclude that the resulting spanning tree $T = \cup_{L\in\mathcal{L}} T_L$ does not violate the chain constraints by more than a constant factor.

One of the arguably most natural constraint families to impose would be to require that the contribution of T_L to any cut S_i is within a constant factor of the contribution of x^* on S_i when only considering edges in E_L, i.e.,

$$|T_L \cap \delta(S_i)| \leq O(x^*(\delta(S_i) \cap E_L)). \tag{3}$$

If the above inequality holds for each $L \in \mathcal{L}$ and $i \in [n-1]$, then the final spanning tree T will indeed not violate any chain constraint by more than a constant factor: it suffices to sum up the inequalities for a fixed i over all sets L and observe that $\{T_L\}_{L\in\mathcal{L}}$ partitions T, and $\{E_L\}_{L\in\mathcal{L}}$ is a partition of E_L:

$$\begin{aligned} |T \cap \delta(S_i)| &= \sum_{L\in\mathcal{L}} |T_L \cap \delta(S_i)| \leq O\left(\sum_{L\in\mathcal{L}} x^*(\delta(S_i) \cap E_L)\right) \\ &= O(x^*(\delta(S_i))) = O(1)b_i. \end{aligned} \tag{4}$$

Unfortunately, it turns out that it is in general impossible to find spanning trees T_L that satisfy (3). This is because there can be many constraints S_i for which $x^*(\delta(S_i \cap E[L])) = o(1)$, in a setting where one has to include at least one edge in T_L that crosses one of these constraints[2].

We therefore introduce a weaker condition on T_L. For $L \in \mathcal{L}$ and $i \in [n-1]$, let $\mathcal{C}_i(L) \subseteq \mathcal{C}(L)$ be the family of all children $C \in \mathcal{C}(L)$ of L that cross the cut S_i, i.e., $S_i \cap L \neq \emptyset$ and $L \setminus S_i \neq \emptyset$. We want the sets T_L to satisfy the following:

$$|T_L \cap \delta(S_i)| \leq 7 \cdot x^*(\delta(S_i) \cap E_L) + 2 \cdot \mathbf{1}_{\{|\mathcal{C}_i(L)|\geq 2\}} \quad \forall i \in [n-1]. \tag{5}$$

Here, $\mathbf{1}_{\{|\mathcal{C}_i(L)|\geq 2\}}$ is the indicator that is equal to 1 if $|\mathcal{C}_i(L)| \geq 2$ and 0 otherwise.

We first show in Section 2.1 that satisfying the above condition indeed leads to a good spanning tree T.

Theorem 3. *For $L \in \mathcal{L}$, let T_L be a spanning tree in G_L that satisfies* (5)*. Then $T = \cup_{L\in\mathcal{L}} T_L$ is a spanning tree in G satisfying*

$$|T \cap \delta(S_i)| \leq 9x^*(\delta(S_i)) \leq 9b_i \quad i \in [n-1].$$

We then show in Section 2.2 that such spanning trees can indeed be found efficiently.

Theorem 4. *For each $L \in \mathcal{L}$, we can efficiently find a spanning tree T_L in G_L satisfying* (5)*.*

[2] Details will be provided in the full version of this paper.

Combining the above two theorems immediately leads to an efficient algorithm to find a spanning tree in G that violates each chain constraint by at most a factor of 9 whenever LP (2) is feasible, and thus proves Theorem 1. For convenience, a summary of our algorithm is provided below.

Algorithm to find $T \in \mathcal{T}$ that violates chain constraints by a factor of at most 9.

1. Compute optimal solution x^* to the linear program (2).
2. Independently for each $L \in \mathcal{L}$, invoke Theorem 4 to obtain a spanning tree T_L in G_L satisfying (5).
3. Return $T = \cup_{L \in \mathcal{L}} T_L$.

2.1 Analysis of Algorithm (Proof of Theorem 3)

For each $L \in \mathcal{L}$, let T_L be a spanning tree in G_L that satisfies (5), let $T = \cup_{L \in \mathcal{L}} T_L$, and let $i \in [n-1]$. Using the same reasoning as in (4) we can bound the load on chain constraint i as follows:

$$\begin{aligned} |T \cap \delta(S_i)| &= \sum_{L \in \mathcal{L}} |T_L \cap \delta(S_i)| \overset{(5)}{\leq} 7 \sum_{L \in \mathcal{L}} x^*(\delta(S_i) \cap E_L) + 2 \sum_{L \in \mathcal{L}} \mathbf{1}_{\{|\mathcal{C}_i(L)| \geq 2\}} \\ &= 7x^*(\delta(S_i)) + 2 \sum_{L \in \mathcal{L}} \mathbf{1}_{\{|\mathcal{C}_i(L)| \geq 2\}}, \end{aligned}$$

using the fact that $\{E_L\}_{L \in \mathcal{L}}$ partitions E. To prove Theorem 3, it thus suffices to show

$$\sum_{L \in \mathcal{L}} \mathbf{1}_{\{|\mathcal{C}_i(L)| \geq 2\}} \leq x^*(\delta(S_i)), \tag{6}$$

which then implies

$$|T \cap \delta(S_i)| \leq 9x^*(\delta(S_i)) \leq 9b_i,$$

where the last inequality follows from x^* being feasible for (2). We complete the analysis by showing the following result, which is a stronger version of (6).

Lemma 1.

$$\sum_{L \in \mathcal{L}} (|\mathcal{C}_i(L)| - 1)^+ \leq x^*(\delta(S_i)),$$

where $(\cdot)^+ = \max(0, \cdot)$.

Proof. Let $\mathcal{L}_i \subseteq \mathcal{L}$ be the family of all sets in $\mathcal{L}$ that cross S_i, and let $\mathcal{L}_i^{\min} \subseteq \mathcal{L}_i$ be all minimal sets of $\mathcal{L}_i$. We will show the following:

$$\sum_{L \in \mathcal{L}} (|\mathcal{C}_i(L)| - 1)^+ = |\mathcal{L}_i^{\min}| - 1. \tag{7}$$

We start by observing how the statement of the lemma follows from (7). Since all sets $W \in \mathcal{L}_i$ correspond to tight spanning tree constraints with respect to

x^*, we have that the restriction $x^*|_{E[W]}$ of x^* to the edges in the graph $G[W]$ is a point in the spanning tree polytope of $G[W]$. In particular, at least one unit of $x^*|_{E[W]}$ crosses any cut in $G[W]$. Since $W \in \mathcal{L}_i$, the set S_i induces a cut $(S_i \cap W, W \setminus S_i)$ in $G[W]$. Hence

$$x^*(\delta(S_i) \cap E[W]) \geq 1 \quad \forall\, W \in \mathcal{L}_i.$$

Now observe that due to minimality of the sets in $\mathcal{L}_i^{\min}$, all sets in $\mathcal{L}_i^{\min}$ are disjoint. Thus

$$x^*(\delta(S_i)) \geq \sum_{W \in \mathcal{L}_i^{\min}} x^*(\delta(S_i) \cap E[W]) \geq |\mathcal{L}_i^{\min}|,$$

which, together with (7), implies Lemma 1. Hence, it remains to show (7).

Let $\mathcal{L}_i^{\mathrm{nm}} = \mathcal{L}_i \setminus \mathcal{L}_i^{\min}$ be all sets in $\mathcal{L}_i$ that are not minimal. Notice that only sets $L \in \mathcal{L}^{\mathrm{nm}}$ can have a strictly positive contribution to the left-hand side of (7) since these are precisely the sets $L \in \mathcal{L}$ with $|\mathcal{C}_i(L)| \geq 1$: for any other set $L \in \mathcal{L}$, either (i) $L \notin \mathcal{L}_i$, in which case non of its children can cross S_i since not even L crosses S_i, or (ii) $L \in \mathcal{L}_i^{\min}$, in which case we again get $|\mathcal{C}_i(L)| = 0$ since L has no children in $\mathcal{L}_i$ due to minimality. We thus obtain

$$\sum_{L \in \mathcal{L}} (|\mathcal{C}_i(L)| - 1)^+ = \sum_{L \in \mathcal{L}_i^{\mathrm{nm}}} (|\mathcal{C}_i(L)| - 1). \tag{8}$$

Observe that $\sum_{L \in \mathcal{L}_i^{\mathrm{nm}}} |\mathcal{C}_i(L)|$ counts each set in $\mathcal{L}_i$ precisely once, except for the set $V \in \mathcal{L}_i$ which is the only set in $\mathcal{L}_i$ that is not a child of some other set in $\mathcal{L}_i$. Hence

$$\sum_{L \in \mathcal{L}_i^{\mathrm{nm}}} |\mathcal{C}_i(L)| = |\mathcal{L}_i| - 1. \tag{9}$$

Finally, combining (8) with (9) we obtain

$$\sum_{L \in \mathcal{L}} (|\mathcal{C}_i(L)| - 1)^+ = \sum_{L \in \mathcal{L}_i^{\mathrm{nm}}} (|\mathcal{C}_i(L)| - 1) = |\mathcal{L}_i| - 1 - |\mathcal{L}_i^{\mathrm{nm}}| = |\mathcal{L}_i^{\min}| - 1,$$

thus proving (7). □

2.2 Main Step of Algorithm (Proof of Theorem 4)

Let $L \in \mathcal{L}$. We now consider the problem of finding a spanning tree T_L in G_L that satisfies (5). Recall that G_L is obtained from the graph (L, E_L) by contracting all children of L. For simplicity, we again interpret an edge $\{v_i, v_j\} \in E_L$ as an edge in G_L between the two vertices corresponding to the sets $C_i, C_j \in \mathcal{L}$ that contain v_i and v_j, respectively.

We start by showing that there are no rainbows in E_L, which is a crucial assumption in the algorithm to be presented in the following.

Lemma 2. *For $L \in \mathcal{L}$, E_L does not contain any rainbows.*

Due to space constraints, the formal proof of Lemma 2 is deferred to the long version of this paper. The intuition behind the result is that when restricting x^* to G_L, a point z in the *interior* of the spanning tree polytope of G_L is obtained with components in $(0, 1)$. Two edges $e, f \in E_L$ forming a rainbow would contradict the optimality of x^* for (2), since one could move some mass from the edge that is in more constraints to the one in fewer. This decreases the objective function, does not violate any chain constraints, and is still in the spanning tree polytope since changes are only done to components represented in z, and z is in the interior of the spanning tree polytope of G_L.

We classify chain constraints S_i into two types, depending on the right-hand side of (5). More precisely, we call a cut S_i *bad* if one can include at most one edge that crosses S_i in T_L without violating (5), i.e.,

$$7x^*(\delta(S_i) \cap E_L) + 2 \cdot \mathbf{1}_{\{|\mathcal{C}_i(L)|\geq 2\}} < 2.$$

Otherwise, a cut S_i is called *good*. Notice that for a cut S_i to be bad, we need to have $|\mathcal{C}_i(L)| = 1$ because of the following. Clearly, if $|\mathcal{C}_i(L)| \geq 2$, then S_i cannot be bad due to the term $2 \cdot \mathbf{1}_{\{|\mathcal{C}_i(L)|\geq 2\}}$. If $|\mathcal{C}_i(L)| = 0$, then we use the fact that all edges in $E[L]$ that cross S_i are part of E_L, hence

$$x^*(\delta(S_i) \cap E_L) = x^*(\delta(S_i) \cap E[L]) \geq 1,$$

where the last inequality follows from the fact that $x^*|_{E[L]}$ is in the spanning tree polytope of the graph $(L, E[L])$. Hence a cut S_i is bad if and only if the following two conditions hold simultaneously:

1. $|\mathcal{C}_i(L)| = 1$,
2. $x^*(\delta(S_i) \cap E_L) < \frac{2}{7}$.

An edge $e \in E_L$ is called *bad* if e crosses at least one bad cut S_i, otherwise it is called *good*. We denote by $A_L \subseteq E_L$ the sets of all good edges.

The procedure we use to find a tree T_L satisfying (5) constructs a tree T_L that consists of only good edges, i.e., $T_L \subseteq A_L$. We determine T_L using a matroid intersection problem that asks to find a spanning tree in G_L satisfying an additional partition matroid constraint.

To define the partition matroid we first number the edges $A_L = \{e_1, \ldots, e_k\}$ as follows. For $e \in A_L$, let $\alpha(e) < \beta(e)$ be the lower and higher index of the two endpoints of e, hence, $e = \{v_{\alpha(e)}, v_{\beta(e)}\}$. (Notice that $\alpha(e) = \beta(e)$ is not possible since $x^*(e) > 0 \ \forall e \in E$ and $x^* \in P_{ST}$.) The edges $e \in A_L$ are numbered lexicographically, first according to lower value of $\alpha(e)$ and then according to lower value of $\beta(e)$, i.e., for any $p \in [k-1]$ either $\alpha(e_p) < \alpha(e_{p+1})$, or $\alpha(e_p) = \alpha(e_{p+1})$ and $\beta(e_p) \leq \beta(e_{p+1})$. Ideally, we would like to group the edges in A_L into consecutive blocks $\{e_p, e_{p+1}, \ldots, e_q\}$ each having a total weight of exactly $x^*(\{e_p, \ldots, e_q\}) = 3/7$. Since this is in general not possible, we will split some of the edges by creating two parallel copies. More precisely, to define the first set P_1 of our partition, let $p \in [k]$ the lagest index for which $x^*(\{e_1, \ldots, e_p\}) \leq 3/7$.

If $x^*(\{e_1,\ldots,e_p\}) = 3/7$ then $P_1 = \{e_1,\ldots,e_p\}$. Otherwise, we replace the edge e_{p+1} by two parallel copies e'_{p+1}, e''_{p+1} of e_{p+1}, and we distribute the weight of $x^*(e_{p+1})$ on e'_{p+1}, e''_{p+1} as follows:

$$x^*(e'_{p+1}) = \frac{3}{7} - x^*(\{e_1,\ldots,e_p\}),$$
$$x^*(e''_{p+1}) = x^*(e_{p+1}) - x^*(e'_{p+1}).$$

This splitting operation does not violate any previous assumptions: the weight x^* on the new edge set $\{e_1,\ldots,e_p,e'_{p+1},e''_{p+1},e_{p+2},\ldots,e_k\}$ is still a point in the spanning tree polytope of the graph over the vertices $\mathcal{C}(L)$ with the new edge set. By applying this splitting operation whenever necessary, we can assume that $A_L = \{e_1,\ldots,e_k\}$ can be partitioned into sets $P_1 = \{e_1,\ldots,e_{p_1}\}$, $P_2 = \{e_{p_1+1},\ldots,e_{p_2}\}$, $\ldots$, $P_s = \{e_{p_{s-1}+1},\ldots,e_k\}$ satisfying:

(i) $x^*(P_h) = 3/7 \quad \forall h \in [s-1]$,
(ii) $x^*(P_s) \leq 3/7$.

Using this partition we define a unitary partition matroid $M = (A_L, \mathcal{I})$ on the good edges A_L, with independent sets

$$\mathcal{I} = \{U \subseteq A_L \mid |U \cap P_h| \leq 1 \ \forall h \in [s]\}.$$

The tree spanning T_L in G_L that our algorithm selects is any spanning tree $T_L \subseteq A_L$ in G_L that is independent in the partition matroid M. Notice that if there exists a spanning tree in G_L that is independent in M, then such a spanning tree can be found in polyonmial time by standard matroid intersection techniques (see [15] for more details about matroids in general and techniques to find common independent sets in the intersection of two matroids). Hence to complete the description and analysis of our algorithm, all that remains is to show the existence of a spanning tree in G_L that is independent in M, and that it satisfies (5). We address these two statements in the following.

The theorem below shows the feasibility of the matroid intersection problem.

Theorem 5. *There exists a spanning tree $T_L \subseteq A_L$ in G_L that is independent in M, i.e., $T_L \in \mathcal{I}$.*

We give a sketch of the proof plan; the full proof is omitted from this extended abstract. We prove that the intersection of the dominant of the spanning tree polytope and the matroid polytope corresponding to M is nonempty. The result then follows by the fact that the intersection of these two polyhedra leads to an integral polytope, a classical result on matroid intersection [15]. More precisely, we show that the point obtained from $\frac{7}{3}x^*$ by setting the values of all bad edges to zero is in both of the above-mentioned polyhedra.

The following theorem finishes the analysis of our algorithm.

Theorem 6. *Let $T_L \subseteq A_L$ be a spanning tree in G_L that is independent in M, then T_L satisfies* (5).

Proof. Consider a cut $S_i \in \mathcal{S}$ for some fixed $i \in [n-1]$. We consider the partition $P_1, \ldots, P_s$ of A_L used to define the partition matroid M. We are interested in all sets in this partition that contain edges crossing S_i. The definition of the partition $P_1, \ldots, P_s$, together with the fact that A_L has no rainbows, implies that the sets of the partition containing edges crossing S_i are consecutively numbered $P_a, P_{a+1}, \ldots, P_b$, for some $1 \leq a \leq b \leq s$. Since T_L contains at most one edge in each partition, we have

$$|T_L \cap \delta(S_i)| \leq b - a + 1. \tag{10}$$

We first consider the case $b - a \geq 2$. Notice that all edges in any set P_h for $a < h < b$ cross S_i. Hence,

$$x^*(\delta(S_i) \cap E_L) \geq \sum_{h=a+1}^{b-1} x^*(P_h) = (b - a - 1) \cdot \frac{3}{7},$$

where we used $x^*(P_h) = \frac{3}{7}$ for $1 \leq h \leq s - 1$. Combining the above inequality with (10), and using that $b - a \geq 2$ in the second inequality, we obtain that

$$|T_L \cap \delta(S_i)| \leq b - a + 1 \leq 3(b - a - 1) \leq 7x^*(\delta(S_i) \cap E_L).$$

Thus T_L satisfies (5).

Assume now $b - a \leq 1$. If S_i is bad, then $|T_L \cap \delta(S_i)| = 0$ since T_L only contains good edges and no good edge crosses any bad cut. Hence, T_L trivially satisfies (5). So assume that S_i is good, i.e., either $|\mathcal{C}(L)| \geq 2$ or $x^*(\delta(S_i) \cap E_L) \geq \frac{2}{7}$. If $|\mathcal{C}(L)| \geq 2$, then beginning again from (10) we have

$$|T_L \cap \delta(S_i)| \leq b - a + 1 \leq 2 = 2 \cdot \mathbf{1}_{|\mathcal{C}_i(L)| \geq 2}.$$

Otherwise, if $x^*(\delta(S_i) \cap E_L) \geq \frac{2}{7}$, then

$$|T_L \cap \delta(S_i)| \leq 2 \leq 7x^*(\delta(S_i) \cap E_L).$$

Either way, T_L satisfies (5). □

3 Conclusions

We would like to close with several interesting directions for future research. One very natural question is whether there is an $O(1)$-approximation for laminar cut constraint; we believe this to be true. Although it seems non-trivial to directly generalize our procedure for the chain-constrained case to the laminar case, we hope that they can be useful in combination with insights from $O(1)$-approximations for the degree-bounded case.

Another natural extension would be to find a weighted $O(1)$-approximation for the chain-constrained spanning tree problem, where the cost of the returned spanning tree should be no larger than the optimal cost of a spanning tree that does not violate the constraints. The main reason our approach does not generalize easily to this setting is that we use a particular objective function to eliminate rainbows in the subproblems.

References

1. Asadpour, A., Goemans, M.X., Madry, A., Oveis Gharan, S., Saberi, A.: An $O(\log n/\log\log n)$-approximation algrithm for the asymmetric traveling salesman problem. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA (2010)
2. Bansal, N., Khandekar, R., Könemann, J., Nagarajan, V., Peis, B.: On generalizations of network design problems with degree bounds. Mathematical Programming, 1–28 (April 2012)
3. Bansal, N., Khandekar, R., Nagarajan, V.: Additive guarantees for degree-bounded directed network design. SIAM Journal on Computing 39(4), 1413–1431 (2009)
4. Bauer, F., Varma, A.: Degree-constrained multicasting in point-to-point networks. In: Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies, INFOCOM, pp. 369–376 (1995)
5. Chaudhuri, K., Rao, S., Riesenfeld, S., Talwar, K.: A push-relabel approximation algorithm for approximating the minimum-degree MST problem and its generalization to matroids. Theoretical Computer Science 410, 4489–4503 (2009)
6. Chaudhuri, K., Rao, S., Riesenfeld, S., Talwar, K.: What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs. Algorithmica 55, 157–189 (2009)
7. Chekuri, C., Vondrák, J., Zenklusen, R.: Dependent randomized rounding via exchange properties of combinatorial structures. In: Proceedings of the 51st IEEE Symposium on Foundations of Computer Science, FOCS, pp. 575–584 (2010)
8. Fürer, M., Raghavachari, B.: Approximating the minimum-degree Steiner Tree to within one of optimal. Journal of Algorithms 17(3), 409–423 (1994)
9. Goemans, M.X.: Minimum bounded degree spanning trees. In: Proceedings of the 47th IEEE Symposium on Foundations of Computer Science, FOCS, pp. 273–282 (2006)
10. Jain, K.: A factor 2 approximation algorithm for the generalized Steiner Network Problem. Combinatorica 21, 39–60 (2001)
11. Könemann, J., Ravi, R.: A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. SIAM Journal on Computing 31, 1783–1793 (2002)
12. Könemann, J., Ravi, R.: Primal-dual meets local search: approximating MST's with nonuniform degree bounds. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC, pp. 389–395 (2003)
13. Oveis Gharan, S., Saberi, A.: The asymmetric traveling salesman problem on graphs with bounded genus. ArXiv (January 2011), `http://arxiv.org/abs/0909.2849`
14. Ravi, R., Marathe, M.V., Ravi, S.S., Rosenkrantz, D.J., Hunt III, H.B.: Approximation algorithms for degree-constrained minimum-cost network-design problems. Algorithmica 31(1), 58–78 (2001)
15. Schrijver, A.: Combinatorial Optimization, Polyhedra and Efficiency. Springer (2003)
16. Singh, M., Lau, L.C.: Approximating minimum bounded degree spanning trees to within one of optimal. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing, STOC, pp. 661–670 (2007)
17. Zenklusen, R.: Matroidal degree-bounded minimum spanning trees. In: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, pp. 1512–1521 (2012)