On the Tree Augmentation Problem

Zeev Nutov

The Open University of Israel, Ra'anana, Israel
nutov@openu.ac.il

Abstract

In the Tree Augmentation problem we are given a tree $T = (V, F)$ and a set $E \subseteq V \times V$ of edges with positive integer costs $\{c_e : e \in E\}$. The goal is to augment T by a minimum cost edge set $J \subseteq E$ such that $T \cup J$ is 2-edge-connected. We obtain the following results.

- Recently, Adjiashvili [SODA 17] introduced a novel LP for the problem and used it to break the 2-approximation barrier for instances when the maximum cost M of an edge in E is bounded by a constant; his algorithm computes a $1.96418 + \epsilon$ approximate solution in time $n^{(M/\epsilon^2)^{O(1)}}$. Using a simpler LP, we achieve ratio $\frac{12}{7} + \epsilon$ in time $2^{O(M/\epsilon^2)}$. This also gives ratio better than 2 for logarithmic costs, and not only for constant costs. In addition, we will show that (for arbitrary costs) the problem admits ratio 3/2 for trees of diameter ≤ 7.
- One of the oldest open questions for the problem is whether for unit costs (when $M = 1$) the standard LP-relaxation, so called Cut-LP, has integrality gap less than 2. We resolve this open question by proving that for unit costs the integrality gap of the Cut-LP is at most $28/15 = 2 - 2/15$. In addition, we will suggest another natural LP-relaxation that is much simpler than the ones in previous work, and prove that it has integrality gap at most 7/4.

1998 ACM Subject Classification G.2.2 [Graph Theory] Graph Algorithms

Keywords and phrases Tree augmentation, Logarithmic costs, Approximation algorithm, Half-integral extreme points, Integrality gap

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.61

1 Introduction

We consider the following problem:

Tree Augmentation
Input: A tree $T = (V, F)$ and an additional set $E \subseteq V \times V$ of edges with positive integer costs $c = \{c_e : e \in E\}$.
Output: A minimum cost edge set $J \subseteq E$ such that $T \cup J$ is 2-edge-connected.

The problem was studied extensively, cf. [11, 15, 3, 21, 8, 9, 4, 19, 5, 17, 2, 16]. For a long time the best known ratio for the problem was 2 for arbitrary costs [11] and 1.5 for unit costs [8, 17]; see also [9] for a simple 1.8-approximation algorithm. It is also known that the integrality gap of a standard LP-relaxation for the problem, so called Cut-LP, is at most 2 [11] and at least 1.5 [4]. Several other LP and SDP relaxations were introduced to show that the algorithm in [8, 9, 17] achieves ratio better than 2 w.r.t. these relaxations, cf. [2, 16]. For additional algorithms with ratio better than 2 for restricted versions see [5, 19].

Let M denote the maximum cost of an edge in E. Recently, Adjiashvili [1] introduced a novel LP for the problem – so called the k-Bundle-LP, and used it to break the natural 2-approximation barrier for instances when M is bounded by a constant. To introduce this result we need some definitions.

The edges of T will be called **T-edges** to distinguish them from the edges in E. Tree Augmentation can be formulated as a problem of covering the T-edges by paths. Let T_{uv}

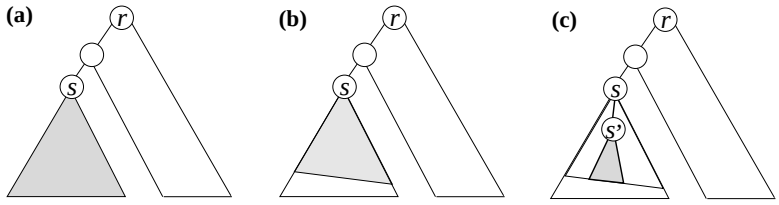

Figure 1 (a) complete rooted subtree; (b) full rooted subtree; (c) branch of a full rooted subtree.

denote the unique uv-path in T. We say that an **edge** uv **covers a** T**-edge** f if $f \in T_{uv}$. Then $T \cup J$ is 2-edge-connected if and only if J covers T. For a set $B \subseteq F$ of T-edges let $\psi(B)$ denote the set of edges in E that cover some $f \in B$, and $\tau(B)$ the minimum cost of an edge set in E that covers B. For $J \subseteq E$ let $x(J) = \sum_{e \in J} x_e$. The standard LP for the problem which we call the Cut-LP seeks to minimize $c^{\mathsf{T}}x = \sum_{e \in E} c_e x_e$ over the Cut-Polyhedron

$$\Pi^{Cut} = \left\{ x \in \mathbb{R}^E : x(\psi(f)) \geq 1 \ \forall f \in F, x \geq 0 \right\} .$$

The k-Bundle-LP of Adjiashvili [1] adds over the standard Cut-LP the constraints $\sum_{e \in \psi(B)} c_e x_e \geq \tau(B)$ for any forest B in T that has at most k leaves, where $k = \Theta(M/\epsilon^2)$. The algorithm of [1] computes a $1.96418 + \epsilon$ approximate solution w.r.t. the k-Bundle-LP in time $n^{k^{O(1)}}$. For unit costs, a modification of the algorithm achieves ratio $5/3 + \epsilon$.

Here we observe that it is sufficient to consider just certain subtrees of T instead of forests. Root T at some node r. The choice of r defines an ancestor/descendant relation on V. The **leaves of** T are the nodes in $V \setminus \{r\}$ that have no descendants. For any subtree S of T, the node s of S closest to r is the root of S, and the pair S, s is called a **rooted subtree** of T, r; we will not mention the roots of trees if they are clear from the context. We say that S is a **complete rooted subtree** if it contains all descendants of s in T, and a **full rooted subtree** if for any non-leaf node v of S the children of v in S and T coincide; see Fig. 1(a,b). A **branch of** S, or a **branch hanging on** s, is a rooted subtree B of S induced by the root s of S and the descendants in S of some child s' of s; see Fig. 1 (c). We say that a subtree B of T is a **branch** if it is a branch of a full rooted subtree, or if it is a full rooted subtree with root r. Equivalently, a branch is a union of a full rooted subtree and its parent T-edge.

Let $\mathcal{B}_k$ denote the set of all branches in T with less than k leaves. The k-Branch-LP seeks to minimize $c^{\mathsf{T}}x = \sum_{e \in E} c_e x_e$ over the k-Branch-Polyhedron $\Pi_k^{Br} \subseteq \mathbb{R}^E$ defined by the constraints:

$$\begin{aligned}
\sum_{e \in \psi(f)} x_e &\geq 1 && \forall f \in F , \\
\sum_{e \in \psi(B)} c_e x_e &\geq \tau(B) && \forall B \in \mathcal{B}_k , \\
x_e &\geq 0 && \forall e \in E .
\end{aligned}$$

The set of constrains of the k-Branch-LP is a subset of constraints of the k-Bundle-LP of Adjiashvili [1], hence the k-Branch-LP is both more compact and its optimal value is no larger than that of the k-Bundle-LP. The first main result in this paper is:

▶ **Theorem 1.** *For any* $1 \leq \lambda \leq k-1$, Tree Augmentation *admits a* $4^k \cdot poly(n)$ *time algorithm that computes a solution of cost at most* $\rho + \frac{8}{3}\frac{\lambda M}{k - \lambda M} + \frac{2}{\lambda}$ *times the optimal value of the* k-Branch-LP*, where* $\rho = \frac{12}{7}$ *for arbitrary costs and* $\rho = 1.6$ *for unit costs.*

For a given ϵ, choosing properly $\lambda = \Theta(1/\epsilon)$ and $k = \Theta(M/\epsilon^2)$ gives ratio $\rho + \epsilon$ in time $2^{O(M/\epsilon^2)} \cdot poly(n)$.

We note that in parallel to our work Fiorini, Groß, Könemann, and Sanitá [10] augmented the k-Bundle LP of [1] by additional constraints – $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts, to achieve ratio $1.5 + \epsilon$ in $n^{(M/\epsilon^2)^{O(1)}}$ time, thus almost matching the best known ratio for unit costs [17]. Our result in Theorem 1, done independently, shows that already the k-Bundle LP has integrality gap closer to 1.5 than to 2. Our version of the algorithm of [1] is also simpler than the one in [10]. In fact, combining our approach with [10] enables to achieve ratio $1.5 + \epsilon$ in $2^{O(M/\epsilon^2)} \cdot poly(n)$ time. Note that this allows to achieve this ratio for logarithmic costs, and not only for constant costs.

Let $\mathsf{diam}(T)$ denote the diameter of T. Tree Augmentation admits a polynomial time algorithm when $\mathsf{diam}(T) \leq 3$. If $\mathsf{diam}(T) = 2$ then T is a star and we get the Edge-Cover problem, while the case $\mathsf{diam}(T) = 3$ is reduced to the case $\mathsf{diam}(T) = 2$ by "guessing" some optimal solution edge that covers the central T-edge. The problem is NP-hard when $\mathsf{diam}(T) = 4$ even for unit costs [11]. We prove that for arbitrary costs Tree Augmentation with trees of diameter ≤ 7 admits ratio 3/2.

Our second main result resolves one of the oldest open questions concerning the problem – whether for unit costs the integrality gap of the Cut-LP is less than 2. This was conjectured in the 90's by Cheriyan, Jordán & Ravi [3] for arbitrary costs, but so far there was no real evidence for this even for unit costs. Our second main result resolves this old open question.

▶ **Theorem 2.** *For unit costs, the integrality gap of the* Cut-LP *is at most* $28/15 = 2 - 2/15$.

In addition, we will suggest another natural LP-relaxation that is much simpler than the ones in previous work, and prove that for unit costs it has integrality gap at most 7/4.

2 The algorithm (Theorem 1)

The Theorem 1 algorithm is a modification of the algorithm of Adjiashvili [1]. We emphasize some differences. We use the k-Branch-LP instead of the k-Bundle-LP of [1]. But, unlike [1], we do not solve our LP at the beginning. Instead, we combine binary search with the ellipsoid algorithm as follows. We start with lower and upper bounds p and q on the value of the k-Branch-LP, e.g., $p = 0$ and q is the cost of some feasible solution to the problem. Given a "candidate" x with $q \leq c^{\mathsf{T}}x \leq p$, the outer iteration (see Algorithm 1) of the entire algorithm either returns a solution of cost at most $(\rho + \frac{8}{3}\frac{\lambda M}{k-\lambda M} + \frac{2}{\lambda})c^{\mathsf{T}}x$ or a constraint of the k-Branch-LP violated by x; we show that this can be done in time $4^k \cdot poly(n)$, rather than in time $n^{k^{O(1)}}$ as in [1]. We set $p \leftarrow \frac{p+q}{2}$ in the former case and $q \leftarrow \frac{p+q}{2}$ in the latter case and continue to the next iteration, terminating when $p - q$ is small enough. This essentially gives a $4^k \cdot poly(n)$ time separation oracle for the k-Branch-LP (if a violated k-branch constraint is found). Since the ellipsoid algorithm uses a polynomial number of calls to the separation oracle, the running time is $4^k \cdot poly(n)$. Note that checking whether $x \in \Pi^{Cut}$ is trivial, hence for simplicity of exposition we will assume that the "candidate" x is in Π^{Cut}.

For a set S of T-edges we denote by T/S the tree obtained from T by contracting every T-edge of S. This defines a new Tree Augmentation instance, where contraction of a T-edge uv leads to shrinking u, v into a single node in the graph (V, E) of edges. In the algorithm, we repeatedly take a certain complete rooted subtree $\hat{S}$, and either find a k-branch-constraint violated by some branch in $\hat{S}$, or a "cheap" cover J_S of a subset S of the T-edges of $\hat{S}$; in the latter case, we add J_S to our partial solution J, contract $\hat{S}$, and iterate on the instance $T \leftarrow T/\hat{S}$. At the end of the loop, the edges that are still not covered by the partial solution J are covered by a different procedure, by a total cost $\frac{2}{\lambda} \cdot c^{\mathsf{T}}x$, as follows.

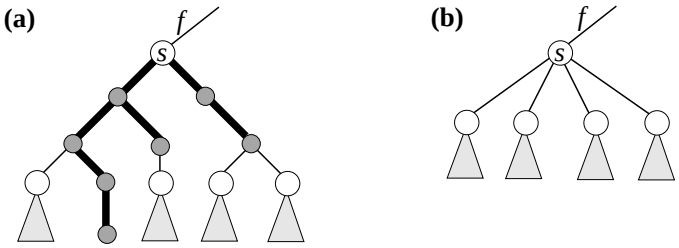

Figure 2 Branches hanging on s after contracting S'; λ-thick T-edges are shown by thick lines.

We call a T-edge $f \in F$ **λ-thin** if $x(\psi(f)) \leq \lambda$, and f is **λ-thick** otherwise. We need the following lemma from [1], for which we provide a proof for completeness of exposition.

▶ **Lemma 3** ([1]). *There exists a polynomial time algorithm that given $x \in \Pi^{Cut}$, $\lambda > 1$, and a set $F' \subseteq F$ of λ-thick T-edges computes a cover J' of F' of cost $\leq \frac{2}{\lambda} \cdot c^{\mathsf{T}} x$.*

Proof. Since all T-edges in F' are λ-thick, x/λ is a feasible solution to the Cut-LP for covering F'. Thus any polynomial time algorithm that computes a solution J' of cost at most 2 times the optimal value of the Cut-LP for covering F' has the desired property. There are several such algorithms, see [11, 12, 14]. ◀

We say that a complete rooted subtree S of T is a **(k, λ)-subtree** if S has at least k leaves and if either the parent T-edge f of S is λ-thin or $s = r$. For $\lambda = \Theta(1/\epsilon)$ and $k = \Theta(M/\epsilon^2)$ we choose $\hat{S}$ to be *an inclusionwise minimal (k, λ)-subtree*. Let us focus on the problem of covering such $\hat{S}$. Let S' be the set of T-edges of the inclusionwise maximal subtree of $\hat{S}$ that contains the root s of $\hat{S}$ and has only λ-thick T-edges (possibly $S = \emptyset$); see Fig. 2(a). We postpone covering the T-edges in S' to the end of the algorithm, so we contract S' into s and consider the tree $S \leftarrow \hat{S}/S'$; see Fig. 2(b). In S, every branch B hanging on s has less than k leaves, by the minimality of S, hence it has a corresponding constraint in the k-Branch-LP. We will show that for a k-branch B an optimal set of edges that covers B can be computed in time $4^k \cdot poly(n)$. If $\sum_{e \in \psi(B)} c_e x_e < \tau(B)$ for some branch B hanging on s in S, then we return the corresponding k-branch constraint violated by x; otherwise, we will show how to compute a "cheap" cover of S. More formally, in the next section we will prove:

▶ **Lemma 4.** *Suppose that we are given an instance of* Tree Augmentation *and $x \in \Pi^{Cut}$ such that any complete rooted proper subtree of the input tree has less than k leaves. Then there exists a $4^k \cdot poly(n)$ time algorithm that either finds a k-branch constraint violated by x, or computes a solution of cost $\leq \rho \sum_{e \in E \setminus R} c_e x_e + \frac{4}{3} \sum_{e \in R} c_e x_e$, where ρ is as in Theorem 1 and R is the set of edges in E incident to the root.*

To find a cheap covers of S, we consider the Tree Augmentation instance obtained from T/S' by contacting into s all nodes not in S. Note that every edge that was in $\psi(S) \cap \psi(f)$ is now incident to the root. Thus since $\rho \geq \frac{4}{3}$, Lemma 4 implies:

▶ **Corollary 5.** *There exists a $4^k \cdot poly(n)$ time algorithm that either finds a k-branch-constraint violated by x, or a cover J_S of S of cost $c(J_S) \leq \rho \sum_{e \in \gamma(S)} c_e x_e + \frac{4}{3} \sum_{e \in \psi(f)} c_e x_e$, where ρ is as in Theorem 1 and $\gamma(S)$ denotes the set of edges with both endnodes in S.*

The outer iteration of the algorithm is as follows:

Algorithm 1: OUTER-ITERATION$(T=(V,F),E,x,c,k,r,\lambda)$

1 $J \leftarrow \emptyset$, $F' \leftarrow \emptyset$
2 **while do**
3 $\quad$ T has at least 2 nodes
4 let $\hat{S}$ be an inclusionwise minimal (k,λ)-subtree of T
5 let S' be the edge-set of the inclusionwise maximal subtree of $\hat{S}$ that contains the root s of $\hat{S}$ and has only λ-thick edges
6 apply the algorithm from **Corollary 5** on $S \leftarrow \hat{S}/S'$
7 if the algorithm returns a cover J_S of S do: $F' \leftarrow F' \cup S'$, $J \leftarrow J \cup J_S$, $T \leftarrow T/\hat{S}$
8 else, return a k-branch constraint violated by x and STOP compute a cover J' of F' of cost $c(J') \leq \frac{2}{\lambda} \cdot c^{\mathsf{T}}x$ using the algorithm from **Lemma 3**
9 **return** $J \cup J'$

Note that at step 7 the T-edges in F' are all λ-thick and thus Lemma 3 applies. We will now analyze the performance of the algorithm assuming than no k-branch-constraint violated by x was found. Let $\delta(S)$ denote the set of edges with exactly one endnode in S and $\gamma(S)$ the set of edges with both endnodes in S. Let f be the parent T-edge of S. Since f is λ-thin

$$\sum_{e\in\psi(f)} c_e x_e \leq \sum_{e\in\psi(f)} M x_e \leq M \cdot x(\psi(f)) \leq M\lambda \ .$$

Since $x(\delta(v)) \geq 1$ for every leaf v of S, $c_e \geq 1$ for every $e \in E$, and since S is a (k,λ)-subtree

$$2\sum_{e\in\gamma(\hat{S})} c_e x_e = \sum_{v\in\hat{S}} \sum_{e\in\delta(v)} c_e x_e - \sum_{e\in\psi(f)} c_e x_e \geq \sum_{v\in\hat{S}\setminus\{s\}} x(\delta(v)) - \lambda M \geq k - \lambda M \ .$$

Consider a single iteration in the while-loop. Let $\Delta(c^{\mathsf{T}}x)$ denote the decrease in the LP-solution value as a result of contracting $\hat{S}$. Then

$$\Delta(c^{\mathsf{T}}x) = \sum_{e\in\gamma(\hat{S})} c_e x_e \geq \frac{k-\lambda M}{2} \ .$$

On the other hand, by Lemma 4, the partial solution cost increase is bounded by

$$c(J_S) \leq \rho \sum_{e\in\gamma(S)} c_e x_e + \frac{4}{3} \sum_{e\in\psi(f)} c_e x_e \leq \rho \sum_{e\in\gamma(\hat{S})} c_e x_e + \frac{4}{3}\lambda M \ .$$

Thus

$$\frac{c(J_S)}{\Delta(c^{\mathsf{T}}x)} \leq \rho + \frac{8}{3}\frac{\lambda M}{k-\lambda M} \ .$$

The while-loop terminates when the LP-solution value becomes 0, hence by a standard local-ratio/induction argument we get that at the end of the while-loop $c(J) \leq \left(\rho + \frac{8}{3}\frac{\lambda M}{k-\lambda M}\right) c^{\mathsf{T}}x$. At step 7 we add an edge set of cost $\leq \frac{2}{\lambda}c^{\mathsf{T}}x$, and Theorem 1 follows. It only remains to prove Lemma 4, which we will do in the subsequent sections.

2.1 Proof of Lemma 4

Assume that we are given an instance $T=(V,F),E,c,r$ of TREE AUGMENTATION and x as in Lemma 4. It is known that TREE AUGMENTATION instances when T is a path can be solved in polynomial time. This allows us to assume that the graph (V,E) is a complete graph and that $c_{uv} = \tau(T_{uv})$ for all $u,v \in V$. Note that we use this assumption only in the proof of Lemma 4, where the running time does not depend on the maximum cost M of an edge in E. Let us say that an edge $uv \in E$ is:

- a **cross-edge** if r is an internal node of T_{uv};
- an **in-edge** if r does not belong to T_{uv};
- an **r-edge** if $r = u$ or $r = v$;
- an **up-edge** if one of u, v is an ancestor of the other.

For a subset $E' \subseteq E$ of edges the **E'-up vector of** x is obtained from x as follows: for every non-up edge $e = uv \in E'$ increase x_{ua} and x_{va} by x_e and then reset x_e to 0, where a is the least common ancestor of u and v. The **fractional cost** of a set J of edges w.r.t. c and x is defined by $\sum_{e \in J} c_e x_e$. Let C^{in}_x, C^{cr}_x, and C^r_x denote the fractional cost of in-edges, cross-edges, and r-edges, respectively, w.r.t. c and x. We fix some $x^* \in \Pi^{Cut}$ and denote by C^{in}, C^{cr}, and C^r the fractional cost of in-edges, cross-edges, and r-edges, respectively, w.r.t. c and x^*. We give two rounding procedures, given in Lemmas 6 and 7. The rounding procedure in Lemma 6 is similar to that of Adjiashvili [1], for which we present an improved running time analysis.

▶ **Lemma 6.** *There exists a $4^k \cdot poly(n)$ time algorithm that either finds a k-branch inequality violated by x^*, or returns an integral solution of cost at most $C^{\mathsf{in}} + 2C^{\mathsf{cr}} + C^r$.*

Proof. Let $\mathcal{B}$ be the set of branches hanging on r. For every $B \in \mathcal{B}$ compute an optimal solution J_B. If for some $B \in \mathcal{B}$ we have $\tau(B) > \sum_{e \in \psi(B)} c_e x^*_e$ then a k-branch inequality violated by x^* is found. Else, the algorithm returns the union $J = \bigcup_{B \in \mathcal{B}} J_B$ of the computed edge sets. As every cross-edge has its endnodes in two distinct branches, while every in-edge or r-edge has its both endnodes in the same branch, we get

$$c(J) \leq \sum_{B \in \mathcal{B}} \tau(B) \leq \sum_{B \in \mathcal{B}} \sum_{e \in \psi(B)} c_e x^*_e = \sum_{B \in \mathcal{B}} \left(\sum_{e \in \delta(B)} c_e x^*_e + \sum_{e \in \gamma(B)} c_e x^*_e \right) = 2C^{\mathsf{cr}} + C^{\mathsf{in}} + C^r \ .$$

It remains to show that an optimal solution in each branch of r can be computed in time $4^k \cdot poly(n)$. More generally, we will show that Tree Augmentation instances with k leaves can be solved optimally within this time bound. Recall that we may assume that the graph (V, E) is a complete graph and that $c_{uv} = \tau(T_{uv})$ for all $u, v \in V$. We claim that then we can assume that T has no node v with $\deg_T(v) = 2$. This is a well known reduction, cf. [20]. In more details, we show that any solution J can be converted into a solution of no greater cost that has no edge incident to v, and thus v can be "shortcut". If J has edges uv, vw then it is easy to see that $(J \setminus \{uv, vw\}) \cup \{uw\}$ is also a feasible solution, of cost at most $c(J)$, since $c_{uw} \leq c_{uv} + c_{vw}$. Applying this operation repeatedly we may assume that $\deg_J(v) \leq 1$. If $\deg_J(v) = 0$, we are done. Suppose that J has a unique edge $e = vw$ incident to v. Let vu and vu' be the two T-edges incident to v, where assume that vu' is not covered by e. Then there is an edge $e' \in J$ that covers vu'. Since e' is not incident to v, it must be that e' covers vu. Replacing e by the edge wu gives a feasible solution without increasing the cost.

Consequently, we reduce our instance to an equivalent instance with at most $2k - 1$ tree edges. Now recall that Tree Augmentation is a particular case of the Min-Cost Set-Cover problem, where the set F of T-edges are the elements and $\{T_e : e \in E\}$ are the sets. The Min-Cost Set-Cover problem can be solved in $2^n \cdot poly(n)$ time via dynamic programming, where n is the number of elements; such an algorithm is described in [7, Sect. 6.1] for unit costs, but the proof extends to arbitrary costs [6]. Thus our reduced Tree Augmentation instance can be solved in $2^{2k-1} \cdot poly(n) \leq 4^k \cdot poly(n)$ time. ◀

For the second rounding procedure Adjiashvili [1] proved that for any $\lambda > 1$ one can compute in polynomial time an integral solution of cost at most $2\lambda C^{\mathsf{in}} + \frac{4}{3} \frac{\lambda}{\lambda - 1} C^{\mathsf{cr}}$. We prove:

▶ **Lemma 7.** *There exists a polynomial time algorithm that computes a solution of cost* $\frac{4}{3}(2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r)$*, and a solution of size* $2C^{\mathsf{in}} + \frac{4}{3}C^{\mathsf{cr}} + C^r$ *in the case of unit costs.*

Consider the case of arbitrary bounded costs. If $C^{\mathsf{in}} \geq \frac{2}{5}C^{\mathsf{cr}}$ we use the rounding procedure from Lemma 6 and the rounding procedure from Lemma 7 otherwise. In both cases we get $c(J) \leq \frac{12}{7}(C^{\mathsf{in}} + C^{\mathsf{cr}}) + \frac{4}{3}C^r$. In the case of unit costs, if $C^{\mathsf{in}} \geq \frac{2}{3}C^{\mathsf{cr}}$ we use the rounding procedure from Lemma 6, and the procedure from Lemma 7 otherwise. In both cases we get $c(J) \leq 1.6(C^{\mathsf{in}} + C^{\mathsf{cr}}) + C^r$.

Lemma 7 is proved in the next section. The proof relies on properties of extreme points of the Cut-Polyhedron Π^{Cut} that are of independent interest.

2.2 Properties of extreme points of the Cut-Polyhedron (Lemma 7)

W.l.o.g., we augment the Cut-LP by the constraints $x_e \leq 1$ for all $e \in E$, while using the same notation as before. Then (the modified) Cut-LP always has an optimal solution x that is an **extreme point** or a **basic feasible solution** of Π^{Cut}. Geometrically, this means that x is not a convex combination of other points in Π^{Cut}; algebraically this means that there exists a set of $|E|$ inequalities in the system defining Π^{Cut} such that x is the unique solution for the corresponding linear equations system. These definitions are known to be equivalent and we will use both of them, cf. [18].

A set family $\mathcal{L}$ is **laminar** if any two sets in the family are either disjoint or one contains the other. Note that Tree Augmentation is equivalent to the problem of covering the laminar family of the node sets of the complete rooted proper subtrees of T, where an edge covers a node set S if it has exactly one endnode in S. In particular, note that the constraint $\sum_{e\in\psi(f)} x_e \geq 1$ is equivalent to the constraint $x(\delta(S)) \geq 1$ where S is the node set of the complete rooted subtree with parent T-edge f.

▶ **Lemma 8.** *Let* (V, E) *be a graph,* $\mathcal{L}$ *a laminar family on* V*, and* $b \in \mathbb{N}^{\mathcal{L}}$*. Suppose that for every* $A \in \mathcal{L}$ *there is no edge between two distinct children of* S *and that the equation system* $\{x(\delta(S)) = b_S : S \in \mathcal{L}\}$ *has a unique solution* $0 < x^* < 1$*. Then* $x^*_e = 1/2$ *for all* $e \in E$*. Furthermore, each endnode of every* $e \in E$ *belongs to some* $S \in \mathcal{L}$*.*

Proof. For every $uv \in E$ put one token at u and one token at v. The total number of tokens is $2|E|$. For $S \in \mathcal{L}$ let $t(S)$ be the number of tokens placed at nodes in S that belong to no child of S. Since $\mathcal{L}$ is laminar, every token is placed in at most one set in $\mathcal{L}$, and thus $\sum_{S\in\mathcal{L}} t(S) \leq 2|E|$. Let $S \in \mathcal{L}$ and let $\mathcal{C}(S)$ be the set of children of S in $\mathcal{L}$. Let E_S be the set of edges in $\delta(S)$ that cover no child of S, and $E_{\mathcal{C}(S)}$ the set of edges not in $\delta(S)$ that cover some child of S. Note that no $e \in E_{\mathcal{C}(S)}$ connects two distinct children of S. Observe that

$$x^*(E_S) - x^*(E_{\mathcal{C}(S)}) = x^*(\delta(S)) - \sum_{C\in\mathcal{C}(S)} x^*(\delta(C)) = b_S - \sum_{C\in\mathcal{C}(A)} b_C \equiv b'_S \ .$$

Thus $x^*(E_S) - x^*(E_{\mathcal{C}(S)})$ is an integer. We cannot have $|E_S| = |E_{\mathcal{C}(S)}| = 0$ by linear independence, and we cannot have $|E_S| + |E_{\mathcal{C}(S)}| = 1$ by the assumption $0 < x < 1$. Thus $|E_S| + |E_{\mathcal{C}(S)}| \geq 2$. Since no $e \in E$ goes between children of S, $t(S) \geq |E_S| + |E_{\mathcal{C}(S)}|$. Consequently, since $\sum_{S\in\mathcal{L}} t(S) \leq 2|E|$, we get: $t(S) = |E_S| + |E_{\mathcal{C}(S)}| = 2 \ \forall S \in \mathcal{L}$. Moreover, if an endnode of some $e \in E$ belongs to no $S \in \mathcal{L}$, then we get the contradiction $\sum_{S\in\mathcal{L}} t(S) \geq 2|E| + 1$. Now we replace our equation system by an equivalent one $\{x(E_S) - x(E_{\mathcal{C}(S)}) = b'_S : S \in \mathcal{L}\}$ obtained by elementary operations on the rows of the coefficients matrix. Note that x^* is also a unique solution to this new equation system. Moreover, this equation system has exactly two variables in each equation and all its coefficients are integral. By [13], the solution of such systems is always half-integral. ◀

Let us say that Tree Augmentation instance is **spider-shaped** if every in-edge in E is an up-edge. By a standard "iterative rounding" argument (cf. [18]), and using the correspondence between rooted trees and laminar families, we get from Lemma 8:

▶ **Corollary 9.** *Suppose that we are given a spider-shaped* Tree Augmentation *instance and* $b \in \mathbb{N}^F$. *Let* x *be an extreme point of the polytope* $\{x \in \mathbb{R}^E : x(\psi(f)) \geq b_f \ \forall f \in F, 0 \leq x \leq 1\}$. *Then* x *is half-integral (namely,* $x_e \in \{0, \frac{1}{2}, 1\}$ *for all* $e \in E$*) and* $x_e \in \{0, 1\}$ *for every* $e \in \delta(r)$.

The algorithm that computes an integral solution of cost $\frac{4}{3}(2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r)$ is as follows. We obtain a spider-shaped instance by removing all non-up in-edges and compute an optimal extreme point solution x to the Cut-LP. By Corollary 9, x is half-integral and $x_e \in \{0, 1\}$ for every $e \in \delta(r)$. We take into our solution every edge e with $x_e = 1$ and round the remaining $1/2$ entries using the algorithm of Cheriyan, Jordán & Ravi [3], that showed how to round a half-integral solution to the Cut-LP to integral solution within a factor of $4/3$. Thus we can compute a solution J of cost at most $c(J) \leq \frac{4}{3}c^{\mathsf{T}}x \leq \frac{4}{3}c^{\mathsf{T}}x^*$. We claim that $c^{\mathsf{T}}x \leq 2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r$. To see this let E^{in} be the set of in-edges and let x' be the E^{in}-up vector of x^*. Then x' is a feasible solution to the Cut-LP of value $2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r$, in the obtained Tree Augmentation instance with all non-up in-edges removed. But since x is an optimal solution to the same LP, we have $c^{\mathsf{T}}x \leq c^{\mathsf{T}}x' = 2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r$. This concludes the proof of Lemma 7 for the case of arbitrary costs.

Note that Corollary 9 implies a 4/3-approximation algorithm for spider-shaped Tree Augmentation instances. Parallel to our work, a stronger result was proved in [10].

▶ **Lemma 10** (Fiorini, Groß, Könemann & Sanitá [10])**.** *Spider-shaped* Tree Augmentation *instances can be solved optimally in polynomial time.*

The following theorem illustrates an application of Lemma 10.

▶ **Corollary 11.** Tree Augmentation *admits ratio* $3/2$ *for trees of diameter* ≤ 7.

Proof. The case $\mathsf{diam}(T) = 7$ is reduced to the case $\mathsf{diam}(T) \leq 6$ by "guessing" some optimal solution edge that covers the central T-edge. So assume that $\mathsf{diam}(T) \leq 6$. Let r be a center of T. Fix some optimal solution and let C^{in} and C^{cr} denote the fractional cost of in-edges and cross-edges in this solution. As before, apply the following two procedures.

1. Each branch B hanging on r is a tree of diameter ≤ 3, hence an optimal cover J_B of B can be computed in polynomial time. The union of the edge sets J_B gives a solution of cost at most $C^{\mathsf{in}} + 2C^{\mathsf{cr}}$.
2. Compute an optimal solution of the spider-shaped instance obtained by removing all non-up in-edges using Lemma 10; the cost of this solution is $2C^{\mathsf{in}} + C^{\mathsf{cr}}$.

Choosing the better among the two computed solutions gives a solution of cost at most $\min\{C^{\mathsf{in}} + 2C^{\mathsf{cr}}, 2C^{\mathsf{in}} + C^{\mathsf{cr}}\}$, while the optimal solution cost is $C^{\mathsf{in}} + C^{\mathsf{cr}}$. It is easy to see that the approximation ratio is bounded by $3/2$; if $C^{\mathsf{in}} \leq C^{\mathsf{cr}}$ then $C^{\mathsf{in}} + 2C^{\mathsf{cr}} \leq \frac{3}{2}(C^{\mathsf{in}} + C^{\mathsf{cr}})$, while if $C^{\mathsf{in}} > C^{\mathsf{cr}}$ then $2C^{\mathsf{in}} + C^{\mathsf{cr}} < \frac{3}{2}(C^{\mathsf{in}} + C^{\mathsf{cr}})$. ◀

Corollary 11 can be used further to obtain ratio $9/5$ for trees of diameter ≤ 15. In a similar way, one can further obtain ratio better than 2 for trees of diameter ≤ 31, and so on, but the ratio approaches 2 when the diameter becomes higher.

In the rest of this section we consider the case of unit costs. For this case we prove:

▶ **Lemma 12.** *Let* x *be an extreme point of the polytope* $\Pi = \{x \in \Pi^{Cut} : C^{\mathsf{in}}_x = a, C^{\mathsf{cr}}_x = b\}$ *where* $a, b \geq 0$, *such that* $x_e > 0$ *for every cross-edge* e. *Then the graph* (V, E^{cr}) *of cross-edges has no even cycle and each one of its connected components has at most one cycle.*

The proof of Lemma 12 will be given in the full version. From Lemma 12 we also get:

▶ **Corollary 13.** *In the case of unit costs there exists a polynomial time algorithm that computes $x \in \Pi$ such that the graph (V, E^{cr}) of cross edges of positive x-value is a forest and such that and $C^{\mathsf{in}}_x = C^{\mathsf{in}}$, $C^r_x = C^r$, and $C^{\mathsf{cr}}_x \leq \frac{4}{3}C^{\mathsf{cr}}$.*

Proof. Let Π be as in Lemma 12 where $a = C^{\mathsf{in}}$ and $b = C^r$ and let x be an optimal extreme point solution to the LP $\min\{\sum_{e\in E} x_e : x \in \Pi\}$. Let Q be a cycle of cross-edges and e the minimum x-value edge in Q. We update x by adding x_e to each of $x_{e'}, x_{e''}$ and setting $x_e = 0$. The increase in the value of x is at most $\frac{1}{3}\sum_{e\in Q} x_e$, and it is easy to see that x remains a feasible solution. In this way we can eliminate all cycles, ending with $x \in \Pi$ as required. ◀

Let x be as in Corollary 13 and let x' be an E^{in}-up vector of x. Note that $x' \in \Pi^{Cut}$, since $x \in \Pi^{Cut}$. We will show how to compute a solution J of size $c(J) \leq x'(E) \leq 2C^{\mathsf{in}} + \frac{4}{3}C^{\mathsf{cr}} + C^r$. While there exists a pair of edges $e = uv$ and $e' = u'v'$ such that $x'_e, x'_{e'} > 0$ and $T_{u'v'} \subset T_{uv}$ we do $x'_e \leftarrow x'_e + x'_{e'}$ and $x'_{e'} \leftarrow 0$. Then x' remains a feasible solution to the Cut-LP without changing the value (since we are in the case of unit costs). Hence we may assume that there is no such pair of edges. Let E' be the support of x'. If every leaf of T has some cross-edge in E' incident to it, then by the assumption above there are no up-edges. In this case, since E' is a forest, $x_e \geq 1$ for every $e \in E'$ and E' is a solution as required.

Otherwise, there is a leaf v of T such that no cross-edge in E' is incident to v. Then there is a unique up-edge e incident to v, and $x'_e \geq 1$. We take such e into our partial solution, updating x' and E' accordingly. Note that some cross-edges may become r-edges, but no up-edge can become a cross-edge, and the set of cross-edges remains a forest. Applying this as long as such leaf v exists, we arrive at the previous case, where adding E' to the partial solution gives a solution as required. This concludes the proof of Lemma 7.

3 An upper bound on the integrality gap of the Cut-LP (Theorem 2)

Let us write the Cut-LP as well as its dual LP explicitly:

$$\begin{array}{lll} \min & \sum_{e\in E} x_e & \\ \text{s.t.} & \sum_{e\in\psi(f)} x_e \geq 1 & \forall f \in F \\ & x_e \geq 0 & \forall e \in E \end{array} \qquad \begin{array}{lll} \max & \sum_{f\in F} y_f & \\ \text{s.t.} & \sum_{\psi(f)\ni e} y_f \leq 1 & \forall e \in E \\ & y_f \geq 0 & \forall f \in F \end{array}$$

To prove that the integrality gap of the Cut-LP is at most $28/15 = 2 - 2/15$ we will show that a simplified version from [16] of the algorithm of [9] has the desired performance. For the analysis, we will use the dual fitting method. We will show how to construct a (possibly infeasible) dual solution $y \in \mathbb{R}^F_+$, that has the following two properties:

Property 1. y fully pays for the constructed solution J, namely, $|J| \leq \sum_{f\in F} y_f$.

Property 2. y may violate the dual constraints by a factor of at most $\rho = 28/15$.

From the second property we get that y/ρ is a feasible dual solution, hence by weak duality the value of y is at most ρ times the optimal value of the Cut-LP. Combining with the first property we get that $|J|$ is at most ρ times the optimal value of the Cut-LP.

The algorithm of [9, 16] iteratively finds a pair T', J' where T' is a subtree of the current tree and J' covers T', contracts T', and adds J' to J. We refer to nodes created by contractions as **compound nodes** and denote by C the set of non-leaf compound nodes of the current tree. Non-compound nodes are referred to as **original nodes**. For technical reasons, the root r is considered as a compound node, hence initially $C = \{r\}$.

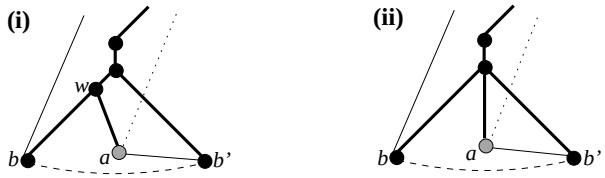

Figure 3 Dangerous trees. T-edges are shown by bold lines, edges in M by dashed lines, other existing edges by thin solid lines, and edges that cannot exist by dotted lines. Original nodes are shown by black circles, while nodes that may be compound are shown by gray circles. Some of the edges may be paths, possibly of length 0. A dangerous tree of type (i) has two nodes with 2 children each, and contracting the path between these two nodes results in a dangerous tree of type (ii).

To identify a pair T', J' as above, the algorithm maintains a matching M on the original leaves. We denote by U the leaves of the current tree unmatched by M. A subtree T' of T is **M-compatible** if for any $bb' \in M$ either both b, b' belong to T' or none of b, b' belongs to T'; in this case we will also say that a contraction of T' is M-compatible. Assuming that all compound nodes were created by M-compatible contractions, then the following type of contractions is also M-compatible.

▶ **Definition 14** (greedy contraction)**.** Adding to the partial solution J an edge e with both endnodes in U and contracting T_e is called a **greedy contraction**.

Given a complete rooted M-compatible subtree T' of T let us use the following notation:

- $M' = M(T')$ is the set of edges in M with both endnodes in T'.
- $U' = U(T')$ is the set of unmatched leaves of T'.
- $C' = C(T')$ is the set of non-leaf compound nodes of T'.

▶ **Definition 15** (semi-closed tree)**.** Let T' be a complete rooted subtree of T. For a subset A of nodes of T' we say that T' is **A-closed** if there is no edge from A to a node outside T', and T' is **A-open** otherwise. Given a matching M on the leaves of T, we say that T' is **semi-closed** if it is M-compatible and U'-closed.

The following definition characterizes semi-closed subtrees that we want to avoid. We will say that T' with 3 leaves is of type (i) if it has two nodes with exactly two children each (see the node w and its father in Fig. 3(i)) and T' is of type (ii) otherwise (see Fig. 3(ii)).

▶ **Definition 16** (dangerous semi-closed tree)**.** A semi-closed subtree T' of T is **dangerous** if it is as in Fig. 3. Namely, $|M'| = 1$, $|U'| = 1$, $|C'| = 0$, and if a is the leaf of T' unmatched by M then: T' is a-closed and there exists an ordering b, b' of the matched leaves of T' such that $ab' \in E$, the contraction of ab' does not create a new leaf, and T' is b-open.

In [9, 17] the following is proved:

▶ **Lemma 17** ([9, 17])**.** *Suppose that the current tree T was obtained from the initial tree by sequentially applying a greedy contraction or a semi-closed tree contraction, and that T has no greedy contraction. Then there exists a polynomial time algorithm that finds a non-dangerous semi-closed subtree T' of T and a cover J' of T' of size $|J'| = |M'| + |U'|$.*

▶ **Definition 18** (twin-edge, stem)**.** An edge on L is a **twin-edge** if its contraction results in a new leaf. The least common ancestor of the endnodes of a twin-edge is a **stem**.

Let $L(M)$ denote the set of leaves matched by M. The algorithm is as follows:

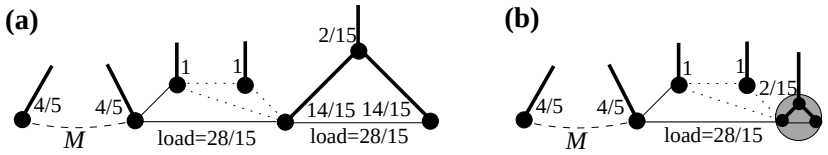

Figure 4 (a) Initial duals at step 1 of Algorithm 3 and the initial loads. Here there is one stem and $|M| = 1$. (b) After contracting the twin-edge, the new compound node has credit 1.

Algorithm 2: ITERATIVE-CONTRACTION$(T = (V, F), E)$

initialize: $M \leftarrow$ inclusionwise maximal matching on L among non-twin edges
$J \leftarrow$ inclusionwise maximal matching on $L \setminus L(M)$
while do
  T has at least 2 nodes
exhaust greedy contractions
if T has at least 2 nodes then for T, J' as in Lemma 17 do: $J \leftarrow J \cup J'$, $T \leftarrow T/T'$
return J

We now describe how to construct y satisfying Properties 1 and 2. For simplicity of exposition, we use the notation y_v and $y_{T'}$ to denote the dual variable of the parent edge of v and of T', respectively. With this notation, Algorithm 3 incorporates into Algorithm 2 the steps of the construction of the dual (possibly infeasible) solution y.

Algorithm 3: DUAL-CONSTRUCTION$(T = (V, F), E)$

Initialize: $M \leftarrow$ inclusionwise maximal matching on L among non-twin edges
$J \leftarrow$ inclusionwise maximal matching on $L \setminus L(M)$ (see Fig. 4)
- $y_v \leftarrow 1$ if $v \in L \setminus L(M \cup J)$
- $y_v \leftarrow 4/5$ if $v \in L(M)$
- $y_v \leftarrow 14/15$ if $v \in L(J)$
- $y_v \leftarrow 2/15$ if v is a stem of an edge in J

while do
  T has at least 2 nodes
exhaust greedy contractions
if T has at least 2 nodes then for T, J' as in Lemma 17 do: $J \leftarrow J \cup J'$, $T \leftarrow T/T'$
Case 1: $|C'| = 0$ and either: $|M'| = 0$ or $|M'| = 1, |U'| \geq 2$
- $y_{T'} \leftarrow 2/5$
- $y_v \leftarrow y_v + 2/5$ if $v \in U'$ and $y_v \leftarrow y_v - 2/5$ if $v \in L' \setminus U'$

Case 2: $|C'| = 0$ and $|M'| = |U'| = 1$
- update y as shown in Fig. 5 **return** J

We now define certain quantities that will help us to prove that at the end of the algorithm $|J| \leq \sum_{f \in F} y_f$ and that y violates the dual constraints by a factor of at most 28/15.

▶ **Definition 19** (load of an edge). Given $y \in \mathbb{R}^F_+$ and an edge $e = uv$, the **load** $\sigma(e)$ **of** e is the sum of the dual variables in the constraint of e in the dual LP, namely $\sigma_e = \sum_{\psi(f) \ni e} y_f$.

▶ **Definition 20** (credit of a node). Consider a constructed dual solution y and a node c of T/J during the algorithm, where c is obtained by contracting the subtree T' of T. The

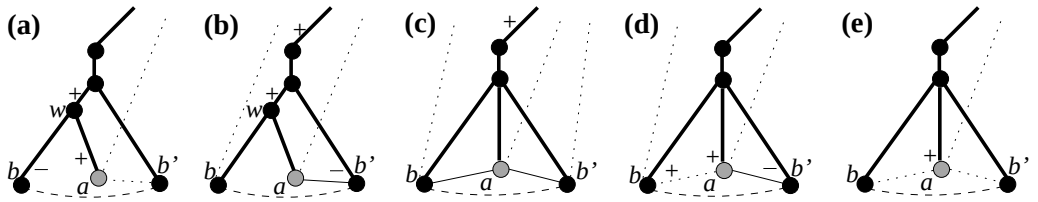

Figure 5 Non-dangerous trees with $|M'| = |U'| = 1$ and duals updates in Case 2 of Algorithm 3. Here "+" means increasing the dual variable by 2/5 and "−" means decreasing the dual variable by 2/5. All trees are a-closed. The trees in (a,b) are non-dangerous trees of type (i), and the trees in (c,d,e) are non-dangerous trees of type (ii). In (a) the edge ab' is missing and in (b) ab' is present and T' is b-closed. In (c) both ab and ab' are present, hence to be non-dangerous the tree must be both b'-closed and b-closed. In (d) ab' is present hence the tree must be b-closed; the case when ab present and the tree is b'-closed is identical. In (e) both ab and ab' are missing.

credit $\pi(c)$ is defined as follows. Let $\pi'(c)$ be the sum of the dual variables y of the edges of T' and the parent edge of v minus the number of edges used by the algorithm to contract T' into c. Then $\pi(c) = \pi'(c) + 1$ if $r \in T'$ and $\pi(c) = \pi'(c)$ otherwise.

We need to prove that at the end of the algorithm, $\sigma(e) \leq 28/15$ for all $e \in E$ and that the unique node of T has credit ≥ 1. For an edge $e = uv$ the **level** $\ell(e)$ of e is the number of endnodes of e in the leaves and compound nodes of T. Clearly, $\ell(e) \in \{0, 1, 2\}$ and if both endnodes of e lie in the same compound node then $\ell(e) = 2$. In the full version we prove:

▶ **Lemma 21.** *At the end of step 1 of the algorithm, and then at the end of every iteration in the "while" loop, the following holds.*

- *For any edge e: $\sigma(e) \leq \frac{28}{15}$ if $\ell(e) = 2$, $\sigma(e) \leq \frac{16}{15}$ if $\ell(e) = 1$, and $\sigma(e) = 0$ if $\ell(e) = 0$.*
- *$\pi(c) \geq 1$ if c is an unmatched leaf or a compound node of T.*

The following LP-relaxation was suggested by the author several years ago. We call an odd size set B of T-edges a **bunch** if no two edges of B lie on the same path in T. Let $\mathcal{B}$ denote the set of bunches in T. For any $B \in \mathcal{B}$ at least $w_B = (|B| + 1)/2$ edges are needed to cover B. The corresponding Bunch-LP and its dual LP are:

$$\begin{array}{lll} \min & \sum_{e \in E} x_e & \\ \text{s.t.} & \sum_{e \in \psi(B)} x_e \geq w_B & \forall B \in \mathcal{B} \\ & x_e \geq 0 & \forall e \in E \end{array} \qquad \begin{array}{lll} \max & \sum_{B \in \mathcal{B}} w_B y_B & \\ \text{s.t.} & \sum_{\psi(B) \ni e} y_B \leq 1 & \forall e \in E \\ & y_B \geq 0 & \forall B \in \mathcal{B} \end{array}$$

A **k-bunch** is a bunch of size k. Let k-Bunch-LP be the restriction of the Bunch-LP to bunches of size $\leq k$. Note that Theorem 1 says that the integrality gap of the 1-Bunch-LP is at most 28/15. We can easily prove a much netter bound for the 3-Bunch-LP.

▶ **Theorem 22.** *For unit costs, the integrality gap of the* 3-Bunch-LP *is at most* 7/4.

Proof. We use the same algorithm but define the dual variables slightly differently. In the initialization step we set $y_v \leftarrow 1$ if $v \in L \setminus L(M \cup J)$, $y_v \leftarrow 3/4$ if $v \in L(M)$, $y_v \leftarrow 1/2$ if $v \in L(J)$, and $y_B \leftarrow 1/2$ if B is the set of the 3 T-edges incident to a stem of an edge in J.

In Case 1 we update $y_{T'} \leftarrow 1/2$, $y_v \leftarrow y_v + 1/2$ if $v \in U'$ and $y_v \leftarrow y_v - 1/2$ if $v \in L' \setminus U'$.

In Case 2 we update y as shown in Fig. 5, where here "−" means decreasing the dual variable by 1/2 and:

- In (a), $y_B \leftarrow y_B + 1/2$ where B is the 3-bunch formed by the parent edges of a, b, w.
- In (b,c,d,e) "+" means increasing the dual variable by $1/2$.

The rest of the proof of Theorem 22 is similar to that of Theorem 2, and will be presented in the full version of the paper. ◀

Acknowledgment. The author thanks László Végh, Shoni Gilboa, Manor Mendel, Moran Feldman, and Gil Alon for several discussions.

References

1. D. Adjiashvili. Beating approximation factor two for weighted tree augmentation with bounded costs. In *SODA*, pages 2384–2399, 2017.
2. J. Cheriyan and Z. Gao. Approximating (unweighted) tree augmentation via lift-and-project, part II. Manuscript, 2015.
3. J. Cheriyan, T. Jordán, and R. Ravi. On 2-coverings and 2-packing of laminar families. In *ESA*, pages 510–520, 1999.
4. J. Cheriyan, H. Karloff, R. Khandekar, and J. Koenemann. On the integrality ratio for tree augmentation. *Operation Research Letters*, 36(4):399–401, 2008.
5. N. Cohen and Z. Nutov. A $(1 + \ln 2)$-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius. *Theoretical Computer Science*, 489-490:67–74, 2013.
6. M. Cygan. Private communication, 2016.
7. M. Cygan, F. Fomin, F. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2016.
8. G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 3/2-approximation for augmenting a connected graph into a two-connected graph. In *APPROX*, pages 90–101, 2001.
9. G. Even, J. Feldman, G. Kortsarz, and Z. Nutov. A 1.8-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms*, 5(2), 2009.
10. S. Fiorini, M. Groß, J.Könemann, and L. Sanitá. A $\frac{3}{2}$-approximation algorithm for tree augmentation via chvátal-gomory cuts. `https://arxiv.org/abs/1702.05567`, Feb 27, 2017.
11. G. Frederickson and J. Jájá. Approximation algorithms for several graph augmentation problems. *SIAM J. Computing*, 10:270–283, 1981.
12. M. Goemans, A. Goldberg, S. Plotkin, E. Tardos D. Shmoys, and D. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.
13. D. Hochbaum, N. Megiddo, J. Naor, and A. Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Programming*, 62:69–83, 1993.
14. K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.
15. S. Khuller and R. Thurimella. Approximation algorithms for graph augmentation. *J. of Algorithms*, 14:214–225, 1993.
16. G. Kortsarz and Z. Nutov. LP-relaxations for tree augmentation. In *APPROX-RANDOM*, pages 13:1–13:16, 2016.
17. G. Kortsarz and Z. Nutov. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms*, 12(2):23, 2016.
18. L. C. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, 2011.
19. Y. Maduel and Z. Nutov. Covering a laminar family by leaf to leaf links. *Discrete Applied Mathematics*, 158(13):1424–1432, 2010.

20 D. Marx and L. Végh. Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation. *ACM Transactions on Algorithms*, 11(4):27, 2015.

21 H. Nagamochi. An approximation for finding a smallest 2-edge connected subgraph containing a specified spanning tree. *Discrete Applied Mathematics*, 126:83–113, 2003.