On the Tree Augmentation Problem[1]

Zeev Nutov
The Open University of Israel

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 gives ratio better than 2 for logarithmic costs, and not only for constant costs. We also 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 prove that another natural LP-relaxation, that is much simpler than the ones in previous work, has integrality gap at most 7/4.

Categories and Subject Descriptors: F.2.2 [**Nonnumerical Algorithms and Problems**]: Computations on discrete structures; G.2.2 [**Discrete Mathematics**]: Graph Algorithms

General Terms: Tree augmentation; Logarithmic costs; Approximation algorithm; Half-integral extreme points; Integrality gap

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, c.f. [Frederickson and Jájá 1981; Khuller and Thurimella 1993; Cheriyan et al. 1999; Nagamochi 2003; Even et al. 2001; 2009; Cheriyan et al. 2008; Maduel and Nutov 2010; Cohen and Nutov 2013; Kortsarz and Nutov 2016b; Cheriyan and Gao 2015; Kortsarz and Nutov 2016a]. For a long time

[1]To appear in ESA 2017

Authors' address: Zeev Nutov, Department of Computer Science, The Open University of Israel, Raanana, Israel. E-mail: nutov@openu.ac.il.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20YY ACM 0000-0000/20YY/0000-0001 $5.00

the best known ratio for the problem was 2 for arbitrary costs [Frederickson and Jájá 1981] and 1.5 for unit costs [Even et al. 2001; Kortsarz and Nutov 2016b]; see also [Even et al. 2009] 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 [Frederickson and Jájá 1981] and at least 1.5 [Cheriyan et al. 2008]. Several other LP and SDP relaxations were introduced to show that the algorithm in [Even et al. 2001; 2009; Kortsarz and Nutov 2016b] achieves ratio better than 2 w.r.t. to these relaxations, c.f. [Cheriyan and Gao 2015; Kortsarz and Nutov 2016a]. For additional algorithms with ratio better than 2 for restricted versions see [Cohen and Nutov 2013; Maduel and Nutov 2010].

Let M denote the maximum cost of an edge in E. Recently [Adjiashvili 2017] 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} 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 2017] 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 [Adjiashvili 2017] 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 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 $\Pi_k^{Br} \subseteq \mathbb{R}^E$

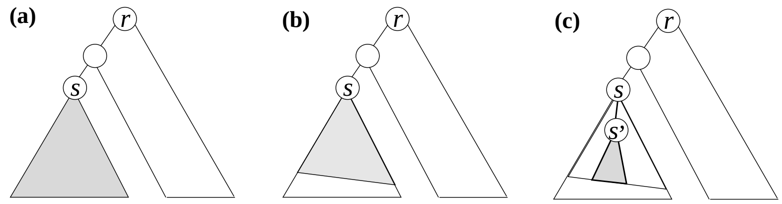

Fig. 1. (a) complete rooted subtree; (b) full rooted subtree; (c) branch of a full rooted subtree.

defined by the constraints:

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

The set of constrains of the k-Branch-LP is a subset of constraints of the k-Bundle-LP of [Adjiashvili 2017], 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 1.1. *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 $2^{O(M/\epsilon^2)} \cdot poly(n)$.

In parallel to our work Fiorini, Groß, Könemann, and Sanitá [Fiorini et al. 2017] augmented the k-Bundle LP of [Adjiashvili 2017] 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 [Even et al. 2001; Kortsarz and Nutov 2016b]. Our result in Theorem 1.1, 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 [Adjiashvili 2017] is also simpler than the one in [Fiorini et al. 2017]. In fact, combining our approach with [Fiorini et al. 2017] 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. We will provide an additional comparison of our results and those in [Fiorini et al. 2017] in Section 2.3.

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 becomes NP-hard when $\mathsf{diam}(T) = 4$ even for unit costs [Frederickson and Jájá 1981]. We prove that (without solving any LP) 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 [Cheriyan et al. 1999] 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 1.2. *For unit costs, the integrality gap of the* Cut-LP *is at most* $28/15 = 2 - 2/15$.

In addition, we will show that for unit costs, another natural simple LP-relaxation, has integrality gap at most $7/4$.

2. ALGORITHM FOR BOUNDED COSTS (THEOREM 1.1)

The Theorem 1.1 algorithm is a modification of the algorithm of [Adjiashvili 2017]. We emphasize some differences. We use the k-Branch-LP instead of the k-Bundle-LP of [Adjiashvili 2017]. But, unlike [Adjiashvili 2017], 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 [Adjiashvili 2017]. 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 (that may have loops and parallel edges), 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.

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 [Adjiashvili 2017], for which we provide a proof for completeness of exposition.

Lemma 2.1 [Adjiashvili 2017]. *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

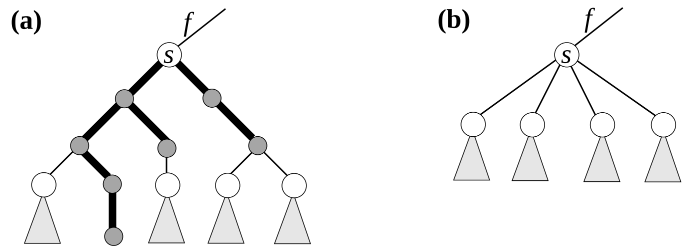

Fig. 2. Branches hanging on s after contracting S'; λ-thick T-edges are shown by thick lines.

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 [Frederickson and Jájá 1981; Goemans et al. 1994; Jain 2001]. $\square$

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 2.2. *Suppose that we are given a* TREE AUGMENTATION *instance 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 1.1 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 2.2 implies:

COROLLARY 2.3. *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 1.1 and* $\gamma(S)$ *denotes the set of edges with both endnodes in* S*, and* f *is the parent* T*-edge of* 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** T has at least 2 nodes **do**
3 let $\hat{S}$ be an inclusionwise minimal (k,λ)-subtree of T
4 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
5 apply the algorithm from **Corollary 2.3** on $S \leftarrow \hat{S}/S'$
6 if **Corollary 2.3** algorithm returns a cover J_S of S then do: $F' \leftarrow F' \cup S'$, $J \leftarrow J \cup J_S$, $T \leftarrow T/\hat{S}$
7 else, return a k-branch constraint violated by x and STOP
8 compute a cover J' of F' of cost $c(J') \leq \frac{2}{\lambda} \cdot c^{\mathsf{T}} x$ using **Lemma 2.1** algorithm
9 **return** $J \cup J'$

Note that at step 7 the T-edges in F' are all λ-thick and thus Lemma 2.1 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 2.2, the partial solution cost increases by at most

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

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 1.1 follows. It only remains to prove Lemma 2.2, which we will do in the subsequent sections.

2.1 Proof of Lemma 2.2

Assume that we are given an instance $T = (V, F), E, c$ of Tree Augmentation with root r and x as in Lemma 2.2. 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 2.2, 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_x^{in}, C_x^{cr}, and C_x^r 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 2.4 and 2.5. The rounding procedure in Lemma 2.4 is similar to that of [Adjiashvili 2017], but we show that it can be implemented in time $4^k \cdot poly(n)$ instead of $n^{k^{O(1)}}$.

Lemma 2.4. *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

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

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 (e.g. see [Marx and Végh 2015]). 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 [Cygan et al. 2016, Sect. 6.1] for unit costs, but the proof extends to arbitrary costs [Cygan 2016]. Thus our reduced Tree Augmentation instance can be solved in $2^{2k-1} \cdot poly(n) \leq 4^k \cdot poly(n)$ time. $\square$

For the second rounding procedure [Adjiashvili 2017] 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 2.5. *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 2.4 and the rounding procedure from Lemma 2.5 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 2.4, and the procedure from Lemma 2.5 otherwise. In both cases we get $c(J) \leq 1.6(C^{\mathsf{in}} + C^{\mathsf{cr}}) + C^r$.

Lemma 2.5 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 2.5)

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, c.f. [Lau et al. 2011].

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 2.6. *Let (V,E) be a graph, $\mathcal{L}$ a laminar family on V, and $b \in \mathbb{N}^{\mathcal{L}}$. Suppose that for every $S \in \mathcal{L}$ 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 $\sum_{S\in\mathcal{L}} t(S) \le 2|E|$. 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 $|E_S| + |E_{\mathcal{C}(S)}| \ge 2$. Since no $e \in E$ goes between children of S, $t(S) \ge |E_S| + |E_{\mathcal{C}(S)}|$. Consequently, since $\sum_{S\in\mathcal{L}} t(S) \le 2|E|$, 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 $\sum_{S\in\mathcal{L}} t(S) \ge 2|E| + 1$. Now we replace our equation system by an equivalent one $\left\{x(E_S) - x(E_{\mathcal{C}(S)}) = b'_S : S \in \mathcal{L}\right\}$ 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 [Hochbaum et al. 1993], 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 (c.f. [Lau et al. 2011]), and using the correspondence between rooted trees and laminar families, we get from Lemma 2.6:

COROLLARY 2.7. *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 $\{x \in \mathbb{R}^E : x(\psi(f)) \ge b_f \ \forall f \in F, 0 \le x \le 1\}$. 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 2.7, 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 [Cheriyan et al. 1999], 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 $c(J) \le \frac{4}{3} c^{\mathsf{T}} x \le \frac{4}{3} c^{\mathsf{T}} x^*$. We claim that $c^{\mathsf{T}} x \le 2C^{\mathsf{in}} + C^{\mathsf{cr}} + C^r$. 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

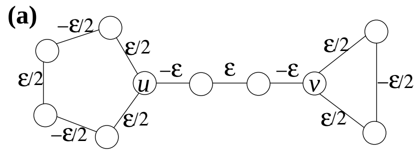

Fig. 3. Illustration to the proof that no two cycles in (V, E^{cr}) are connected by a path.

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 2.5 for the case of arbitrary costs.

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

LEMMA 2.8. *Let $a, b \geq 0$ and let x be an extreme point of the polytope*

$$\Pi = \{x \in \Pi^{Cut} : C^{\mathsf{in}}_x = a, C^{\mathsf{cr}}_x = b\}$$

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.

PROOF. Let Q be a cycle in E^{cr} and let $\epsilon = \min_{e \in Q} x_e$. Since $x_e > 0$ for all $e \in E^{\mathsf{cr}}$, $\epsilon > 0$. If $|Q|$ is even, let Q', Q'' be a partition of Q into two perfect matchings. Let z be a vector defined by $z_e = \epsilon$ if $e \in Q'$, $z_e = -\epsilon$ if $e \in Q''$, and $z_e = 0$ otherwise. By the choice of ϵ, $x + z, x - z$ are non-negative, and it is not hard to verify that $x+z, x-z \in \Pi$. However, $x = \frac{1}{2}(x+z) + \frac{1}{2}(x-z)$, contradicting that x is an extreme point.

Suppose that $|Q|$ is odd. Let u, v be nodes on Q, possibly $u = v$. We claim that $(V, E^{\mathsf{cr}} \setminus Q)$ has no uv-path; this also implies that any two odd cycles in (V, E^{cr}) are node disjoint. Suppose to the contrary that $(V, E^{\mathsf{cr}} \setminus Q)$ has a uv-path P. Let P' and P'' be the two internally disjoint uv-paths in Q where $|P'|$ is odd and $|P''|$ is even. Then one of $P \cup P'$ and $P \cup P''$ is an even cycle, contradicting that (V, E^{cr}) has no even cycles.

Finally, we show that no two cycles in (V, E^{cr}) are connected by a path. Suppose to the contrary that (V, E^{cr}) has a uv-path P that connects two distinct cycles Q_u and Q_v, see Fig. 3. Let z be defined as in Fig. 3. By the choice of ϵ, each one of the vectors $x + z$ and $x - z$ is non-negative, and they are both in Π. However, $x = \frac{1}{2}(x + z) + \frac{1}{2}(x - z)$, contradicting that x is an extreme point. □

Note that Lemma 2.8 implies that extreme points of Π^{Cut} have the property given in the lemma. From Lemma 2.8 we also get:

COROLLARY 2.9. *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 2.8 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 not hard to verify that x remains a feasible solution. In this way we can eliminate all cycles, ending with $x \in \Pi$ as required. □

Remark. Corollary 2.9 holds also for arbitrary costs, but in this case the proof is much more involved. Specifically, we use the following statement, which we do not prove here, since it currently has no application:
Let $q \geq 3$ and let $c_i, x_i \geq 0$ be reals, $i = 0, \ldots, q-1$. Denote $a_i = c_{i-1} - c_i + c_{i+1}$ where the indices are modulo k. Then $\sum_{i=0}^{k-1} c_i x_i \geq 3 \cdot \min_{0 \leq i \leq k-1} a_i x_i$.

Let x be as in Corollary 2.9 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 2.5.

2.3 Comparison to the results of Fiorini et al.

We need some definitions to compare Corollary 2.7 to a result of Fiorini, Groß, Könemann, and Sanitá [Fiorini et al. 2017], that showed that spider-shaped Tree Augmentation instances can be solved in polynomial time. Consider the polyhedron $\Pi(b) = \{x : Ax \geq b, x \geq 0\}$ where A is a given integral matrix, and let $\Pi_I(b)$ be convex hull of the integral points in $\Pi(b)$. The **$\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts** (see [Gomory 1958; Chvátal 1973; Caprara and Fischetti 1996]) are inequalities of the form $(\lambda^\mathsf{T} A + \mu^\mathsf{T})x \geq \lceil \lambda^\mathsf{T} b \rceil$, for vectors λ, μ with entries in $\{0, \frac{1}{2}\}$ such that $\lambda^\mathsf{T} A + \mu^\mathsf{T}$ is an integral vector.

A matrix A is **2-regular** if each of its non-singular square submatrices is half-integral. It is known that A is 2-regular if and only if the extreme points of $\Pi(b)$ are half-integral for any integral vector b, and that if A is 2-regular then $P_I(b)$ is described by the $\{0, 1/2\}$-Chvátal-Gomory cuts [Appa and Kotnyek 2004]. Thus in matrix terms our Corollary 2.7 implies the following:

Corollary 2.10. *In spider-shaped* Tree Augmentation *instances, the incidence matrix A of the T-edges and the paths $\{T_e : e \in E\}$ is 2-regular.*

Note that 2-regularity of A does not imply that the corresponding integer program $\min\{c^\mathsf{T} x : x \in \Pi_I(b)\}$ is in P, since we have no guarantee that the separation problem for $\{0, 1/2\}$-Chvátal-Gomory cuts is in P. However, a particular class of 2-regular matrices has this nice property. A matrix A is a **binet matrix** if there exists a square non-singular integer matrix R such that $\|z\|_1 \leq 2$ for any column z of R or of RA, where $\|z\|_1 = \sum_i |z_i|$ is the L^1-norm of z. It is known that any binet matrix is 2-regular, but binet matrices have the advantage that the separation problem for $\{0,1/2\}$-Chvátal-Gomory cuts is in P [Appa et al. 2007]. All in all, we have that if A is binet then the integer program $\min\{c^{\mathsf{T}}x : x \in \Pi_I(b)\}$ can be solved efficiently, by a combinatorial algorithm [Appa et al. 2007]. The following result, that is stronger than our Corollary 2.10, was proved by Fiorini, Groß, Könemann & Sanitá [Fiorini et al. 2017] in parallel to our work; for completeness of exposition we provide a proof-sketch.

LEMMA 2.11 [FIORINI ET AL. 2017]. *In spider-shaped* TREE AUGMENTATION *instances, the incidence matrix A of the T-edges and the paths $\{T_e : e \in E\}$ is binet.*

PROOF. For $f \in F$ let $\mathsf{ch}(f)$ denote the set of child T-edges of f in T. Define a square matrix $R \in \{-1,0,1\}^{F \times F}$ as follows: $R_{f,f} = 1$, $R_{f,g} = -1$ if $g \in \mathsf{ch}(f)$, and the other entries of R are 0. Let z be the column in R of $g \in F$. Then $z_g = 1$ and if g has a parent T-edge f then $z_f = -1$; other entries of z are 0. Thus $\|z\|_1 \le 2$. We prove by induction on $|F|$ that R is non-singular. The case $|F| = 1$ is trivial. If $|F| \ge 2$, let f be a leaf T-edge. The row of f in R has a unique non-zero entry $R_{f,f} = 1$. Let T' be obtained from T by removing f and the leaf of f. The matrix R' that corresponds to T' is obtained from R by removing the row of f and the column of f. By the induction hypothesis, $\det(R') \ne 0$. Thus $|\det(R)| = |\det(R')| \ne 0$, implying that R is non-singular.

We now describe the entries of the matrix RA. Let y be the row in R of $f \in F$. Then $y_f = 1$ and $y_g = -1$ for $g \in \mathsf{ch}(f)$; other entries of y are 0. Column e in A encodes the path T_e, namely, has 1 for each T_e-edge; other entries are 0. Thus

$$(RA)_{f,e} = |f \cap T_e| - |\mathsf{ch}(f) \cap T_e| \ .$$

In particular, if z is the column in RA of $e \in E$ then:

—If $f \in T_e$ then $z_f = 1$ if $|\mathsf{ch}(f) \cap T_e| = 0$ and $z_f = 0$ otherwise.

—If $f \notin T_e$ then $z_f = -|\mathsf{ch}(f) \cap T_e|$.

Now let $e = uv$ and let a be the least common ancestor of u, v. Consider two cases, in which we indicate only non-zero entries of z. If $a \in \{u,v\}$ (e is an up-edge), say $a = v$, then $z_f = 1$ if f is the parent T-edge of u and $z_f = -1$ if $a \ne r$ and f is the parent T-edge of v. If $a \notin \{u,v\}$ then $z_f = 1$ if f is the parent T-edge of u or of v, and $z_f = -2$ if f is the parent T-edge of a; however, in a spider-shaped TREE AUGMENTATION instance we cannot have $z_f = -2$, since if e is a cross edge then $a = r$ and thus a has no parent T-edge. Consequently, in both cases $\|z\|_1 \le 2$. □

By a result of [Appa et al. 2007] (an integer program $\min\{c^{\mathsf{T}}x : x \in \Pi_I(b)\}$ is in P if A is binet), Lemma 2.11 immediately implies:

COROLLARY 2.12 [FIORINI ET AL. 2017]. *Spider-shaped* TREE AUGMENTATION *instances admit a polynomial time algorithm.*

[Fiorini et al. 2017] also provided a direct simple proof that the problem of separating the $\{0,1/2\}$-Chvátal-Gomory cuts of the CUT-LP is in P. Combining this with our Corollary 2.10 and a result of [Appa and Kotnyek 2004] ($P_I(b)$ is described by the $\{0,1/2\}$-Chvátal-Gomory cuts if A is 2-regular), also enables to deduce Corollary 2.12.

[Fiorini et al. 2017] considered the Odd-Cut k-Bundle LP obtained by adding to the k-Bundle LP of [Adjiashvili 2017] the "odd-cuts" (the $\{0,1/2\}$-Chvátal-Gomory cuts). They showed that this LP is compatible with the decomposition of [Adjiashvili 2017], namely, that if x a feasible solution to this LP, then for any k-bundle B the restriction of x to $\psi(B)$ is a feasible solution to this LP on B. Since each k-branch is a k-bundle, the more compact Odd-Cut k-Branch LP is also compatible with [Adjiashvili 2017] decomposition. As was mentioned in the Introduction, combining the [Fiorini et al. 2017] and our paper techniques gives: *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 $\frac{3}{2}+\frac{2\lambda M}{k-\lambda M}+\frac{2}{\lambda}$ times the optimal value of the* Odd-Cut k-Branch LP.
Let us briefly describe the modifications needed for this combined result.

—Lemma 2.1 is used in the same way as before, namely, just to cover by cost $\frac{2}{\lambda}c^{\mathsf{T}}x$ the λ-thick edges uncovered by the main algorithm.

—Recall that [Fiorini et al. 2017] showed that separating the odd-cuts is in P. The new Lemma 2.4 would state that given $x^* \in \mathbb{R}^E$, there exists a $4^k \cdot poly(n)$ time algorithm that either finds a k-branch inequality or an odd-cut inequality violated by x^*, or returns an integral solution of cost at most $C^{\mathsf{in}}+2C^{\mathsf{cr}}+C^r$.

—Lemma 2.5 will be replaced by a result of [Fiorini et al. 2017] that a solution of cost $2C^{\mathsf{in}}+C^{\mathsf{cr}}+C^r$ can be computed in polynomial time.

—In an improved version of Corollary 2.3 one gets that if no violated inequality is found then $c(J_S) \leq \sum_{e\in\gamma(S)} c_e x_e + \sum_{e\in\psi(f)} c_e x_e$. And then, the same calculations as after Algorithm 1 give $\frac{c(J_S)}{\Delta(c^{\mathsf{T}}x)} \leq \frac{3}{2}+\frac{2\lambda M}{k-\lambda M}$.

Let us now illustrate another application of Corollary 2.12.

Lemma 2.13. 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 2.11; 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 $2C^{\mathsf{in}}+C^{\mathsf{cr}} < \frac{3}{2}(C^{\mathsf{in}}+C^{\mathsf{cr}})$. □

Lemma 2.13 can be used further to obtain ratio $9/5$ for trees of diameter ≤ 15. As before, we can reduce the case $\mathsf{diam}(T)=15$ to the case $\mathsf{diam}(T)\leq 14$ by guessing some optimal solution edge that covers the central T-edge. We compute a 3/2-approximate cover of each branch, which gives a solution of cost at most $\frac{3}{2}(C^{\mathsf{in}} + 2C^{\mathsf{cr}})$. We also compute a solution of cost at most $2C^{\mathsf{in}} + C^{\mathsf{cr}}$ as before, using Corollary 2.12. The worse case is when these two bounds are equal, namely, when $C^{\mathsf{in}} = 4C^{\mathsf{cr}}$. In this case we get that $\frac{2C^{\mathsf{in}}+C^{\mathsf{cr}}}{C^{\mathsf{in}}+C^{\mathsf{cr}}} = 1 + \frac{C^{\mathsf{in}}}{C^{\mathsf{in}}+C^{\mathsf{cr}}} = 1 + \frac{4}{4+1} = \frac{9}{5}$. In a similar way, one can further obtain ratio better than 2 when $\mathsf{diam}(T) \leq 31$, and so on, but the ratio approaches 2 when the diameter becomes higher.

We note that the effort in proving Lemma 2.11 of [Fiorini et al. 2017] and our Corollary 2.10 is roughly the same. However, the result in Lemma 2.11 of [Fiorini et al. 2017] is more general and thus enables to obtain easily results for related problems, as we illustrate below. Note however that our main result is extending the range of costs for which a ratio better than 2 can be achieved, by a simpler algorithm. Moreover, the proof idea of Corollary 2.7 might be useful for half-integral network design problems for which the corresponding matrix is not binet.

It is known that if A is binet then also the problem of minimizing $c^{\mathsf{T}}x$ over $\{x \in \Pi_I(b) : p \leq x \leq q\}$ can be solved in polynomial time for any integer vectors p and q [Appa and Kotnyek 2004; Appa et al. 2007]. Now consider the following generalization of Tree Augmentation, which we call the Generalized Tree Augmentation problem. Here we are also given demands $\{b_f : f \in F\}$ on the T-edges, and require that at least b_f edges will cover every T-edge $f \in F$; we also require that for every edge $e \in E$ at most q copies of e are selected. Then from Lemma 2.11 of [Fiorini et al. 2017] and [Appa and Kotnyek 2004; Appa et al. 2007] we immediately get:

Corollary 2.14. *Spider-shaped* Generalized Tree Augmentation *instances admit a polynomial time algorithm.*

3. BOUND ON THE INTEGRALITY GAP OF THE CUT-LP (THEOREM 1.2)

Let us write the (unit costs) Cut-LP as well as its dual LP explicitly:

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

To prove that the integrality gap of the Cut-LP is at most 28/15 we will show that a simplified version from [Kortsarz and Nutov 2016a] of the algorithm of [Even et al. 2009] 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.

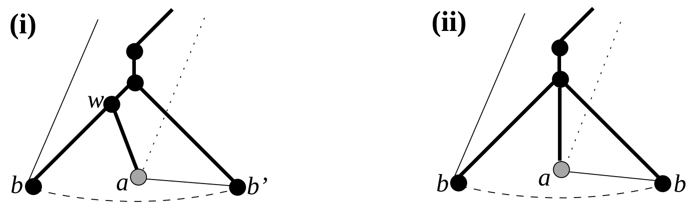

Fig. 4. Dangerous trees. Here and in subsequent figures T-edges are shown by bold lines, edges in M by dashed lines, some other existing edges by thin solid lines, and edges that cannot exist by dotted lines. Nodes that must be original are shown by black circles, while nodes that may be compound nodes 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 exactly 2 children each, and contracting the path between these two nodes results in a dangerous tree of type (ii).

The algorithm 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. Whenever T' contains the root of T, the new compound node becomes the root of the new tree.

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 all compound nodes were created by M-compatible contractions, then the following type of contractions is also M-compatible.

Definition 3.1 (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 we use the 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 3.2 (semi-closed tree). *Let T' be a complete rooted subtree of T. For a subset A of nodes of T' we say that T' of 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 parent in Fig. 4(i)) and T' is of type (ii) otherwise (see Fig. 4(ii)).

Definition 3.3 (dangerous semi-closed tree). *A semi-closed subtree T' of T is* **dangerous** *if it is as in Fig. 4. 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.*

Definition 3.4 (twin-edge, stem). *Let L denote the set of leaves of T. 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**.

In [Even et al. 2009] the following is proved:

Lemma 3.1 [Even et al. 2009]. *Suppose that M has no twin-edges and 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 $|J'| = |M'| + |U'|$.*

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

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 $L \setminus L(M)$
contract every link in J
while T has at least 2 nodes **do**
  exhaust greedy contractions
  if T has at least 2 nodes then for T', J' as in Lemma 3.1 do:
   $J \leftarrow J \cup J'$, $T \leftarrow T/T'$
return J

We now describe how to construct y satisfying Properties 1 and 2 as above. For simplicity of exposition let us use the notation y_v to denote the dual variable of the parent T-edge of v. 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. 5)
- $y_v \leftarrow 1$ if $v \in L \setminus (M \cup J)$
- $y_v \leftarrow 4/5$ 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

contract every link in J
while T has at least 2 nodes **do**
  exhaust greedy contractions
  if T has at least 2 nodes then for T', J' as in **Lemma 3.1** do:
   $J \leftarrow J \cup J'$, $T \leftarrow T/T'$
   Case 1: $|C'| = 0$ and either: $|M'| = 0$ or $|M'| = 1, |U'| \geq 2$
    • update y as shown in **Fig. 6**
   Case 2: $|C'| = 0$ and $|M'| = |U'| = 1$
    • update y as shown in **Fig. 7**
return J

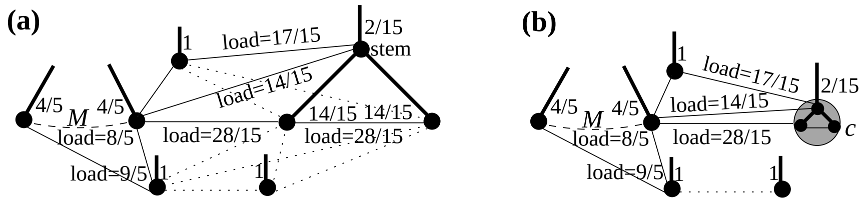

Fig. 5. T-edges are shown by bold lines, edges in M by dashed lines, some other existing edges by thin solid lines, and edges that cannot exist by dotted lines. (a) Initial duals at step 1 of Algorithm 3 and the initial loads. Here there is one stem and $|M| = 1$. (b) After contracting the twin-edge at step 2, the new compound node c has credit 1.

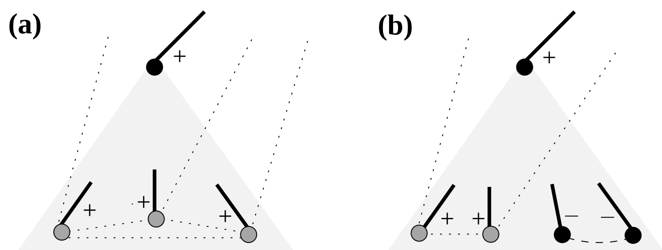

Fig. 6. Duals updates in Case 1 of Algorithm 3. (a) $|M'| = 0$; here "+" means increasing the dual variable by 1/2. (b) $|M'| = 1$, $|U'| \geq 2$; here "+" means increasing the dual variable by 2/5 and "−" means decreasing the dual variable by 2/5.

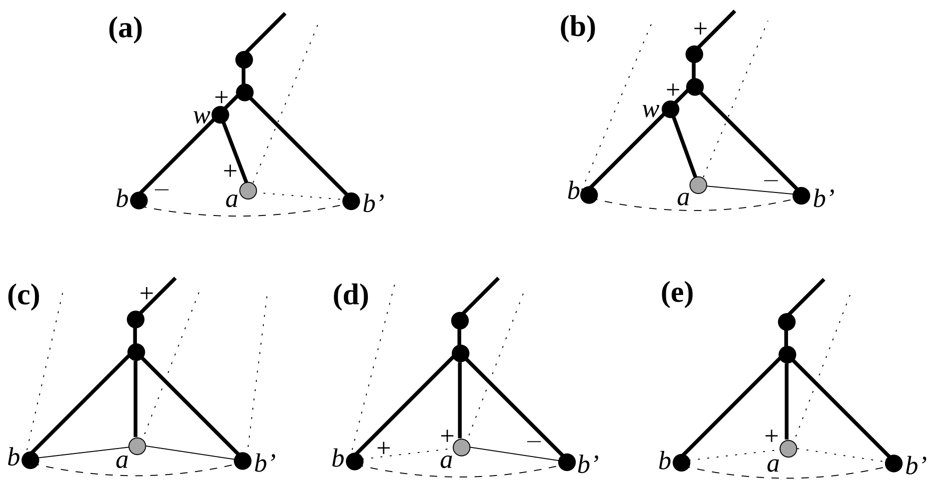

Fig. 7. Non-dangerous trees with $|M'| = |U'| = 1$ and duals updates in Case 2 of Algorithm 3. Here "+" means increasing the dual variable by 2/5 and "−" means decreasing the dual variable by 2/5. All trees are a-closed. The trees in (a,b) are non-dangerous trees of type (i), and the trees in (c,d,e) 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 edges 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.

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 3.5 (load of an edge). *Given $y \in \mathbb{R}_+^F$ and an edge $e \in E$, 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 3.6 (credit of a node). *Consider a constructed dual solution y and a node c of T during the algorithm, where c is obtained by contracting the (possibly trivial) subtree S of T. The* **credit** *$\pi(c)$ is defined as follows. Let $\pi'(c)$ be the sum of the dual variables y of the edges of S and the parent edge of c minus the number of edges used by the algorithm to contract S into c. Then $\pi(c) = \pi'(c) + 1$ if $r \in S$ and $\pi(c) = \pi'(c)$ otherwise.*

Our goal is 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 at least 1. For an edge e that connects nodes u, v of the current tree T the **level** $\ell(e)$ of e (w.r.t. the current tree T) is the number of compound nodes and original leaves (of the current tree T) in $\{u, v\}$. Clearly, $\ell(e) \in \{0, 1, 2\}$ and note that if both endnodes of e lie in the same compound node then e is a loop and $\ell(e) = 2$.

Lemma 3.2. *At the end of step 2 of Algorithm 3, and then at the end of every iteration in the "while" loop, the following holds.*

(i) $\pi(c) \geq 1$ *if c is an unmatched leaf or a compound node of T.*

(ii) *For any edge e:*
- $\sigma(e) \leq 28/15$ *if $\ell(e) = 2$.*
- $\sigma(e) \leq 16/15$ *if $\ell(e) = 1$.*
- $\sigma(e) = 0$ *if $\ell(e) = 0$.*

Proof. It is easy to see that the statement holds at the end of step 2, see Fig. 5. We will prove by induction that the statement continues to hold after each contraction step of the while-loop. Let us consider such contraction step that resulted in a new compound node c and denote by σ', ℓ', π' the new values of σ, ℓ, π after the contraction. By the induction hypothesis σ, ℓ, π satisfy properties (i) and (ii) above, and we prove that σ', ℓ', π' satisfy (i) and (ii) as well.

For (i) it is sufficient to prove that $\pi'(c) \geq 1$, as $\pi' = \pi$ for other nodes. Consider a greedy contraction with an edge e connecting two unmatched leaves u and v. By the induction hypothesis, $\pi(u), \pi(v) \geq 1$. Thus $\pi'(c) \geq \pi(u) + \pi(v) - 1 \geq 1$. Now suppose that a semi-closed tree T' was contracted into c. Let $\Delta(y)$ denote the increase in the value of y during the contraction step and note that

$$\pi'(c) \geq \left(\pi(C') + \frac{8}{5}|M'| + |U'|\right) - (|M'| + |U'|) + \Delta(y) \geq |C'| + \frac{3}{5}|M'| + \Delta(y)\ .$$

If $|C'| \geq 1$ or $|M'| \geq 2$ then $\pi'(c) \geq |C'| + \frac{3}{5}|M'| \geq 1$. If $|M'| = 0$ (Fig. 6(a)) then $\pi'(c) \geq \Delta(y) \geq \frac{1}{2}(|U'| + 1) \geq 1$. If $|M'| = 1$ then $\Delta(y) = \frac{2}{5}$, since in all cases in Figures 6(b) and 7, the number of "+" signs is larger by one than the number of "−" signs; thus $\pi'(c) \geq \frac{3}{5}|M'| + \frac{2}{5} \geq 1$. In all cases $\pi'(c) \geq 1$, as required.

We now show that property (ii) holds. Note that if $\sigma'(e) = \sigma(e)$ then (ii) continues to hold for e, since contractions can only increase the edge level and since the bounds in (ii) are increasing with the level. Thus we only need to consider the cases when we change the dual variables, namely, when a semi-closed tree T' was contracted into c; these are the cases given in Figures 6 and 7.

It is sufficient to consider edges with at least one endnode in T', as $\sigma' = \sigma$ and $\ell' = \ell$ holds for other edges. Let e be an edge that has an endnode in T'. Let $q(e)$ denote the number of "+" signs minus the number of "−" signs in Figures 6 and 7 along the path T_e; we have $\sigma'(e) - \sigma(e) = \frac{1}{2}q(e)$ in Fig. 6(a) and $\sigma'(e) - \sigma(e) = \frac{2}{5}q(e)$ in all other cases. One can verify that $q(e) \leq 0$ if e connects a leaf of T' to another leaf of T' or to a node outside T'. Thus it remains to consider the case when e is incident to a non-leaf node of T'. Then $\ell'(e) > \ell(e)$, since $|C'| = 0$. One can verify that $q(e) \leq 1$, except one case – $q(e) = 2$ if in Fig. 7(a) e connects the leaf a to a node v in T' that is an ancestor of w; this tight case is the one that determined our initial assignment of dual variables. In all cases we have $\sigma'(e) - \sigma(e) \leq \frac{4}{5}$, which equals the minimum difference $\frac{28-16}{15}$ in the bounds in (ii) due to an increase of an edge level. This concludes the proof of (ii) and of the lemma. □

4. INTEGRALITY GAP OF THE 3-BUNCH-LP

The following simple LP-relaxation was suggested by the author several years before [Adjiashvili 2017] and [Fiorini et al. 2017]. Let us call an odd size set B of edges of T a **bunch** if no 3 edges in B lie on the same path in T. Let $\mathcal{B}$ denote the set of bunches in T. For every $B \in \mathcal{B}$ at least $w_B := (|B| + 1)/2$ edges are needed to cover B. The corresponding Bunch-LP and its dual LP are:

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

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 1-Bunch-LP is just the Cut-LP, and that Theorem 1.1 says that the integrality gap of the 1-Bunch-LP is at most 28/15. We can easily prove a better bound for the 3-Bunch-LP.

Theorem 4.1. *For unit costs, the integrality gap of the* 3-Bunch-LP *is at most* 7/4.

Proof. We use the same algorithm as before, but define the dual variables differently. In the initialization step we set (see Fig. 8):

- $y_v \leftarrow 1$ 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 $v \in L(J)$
- $y_B \leftarrow 1/2$ if B is the 3-bunch of the T-edges incident to a stem of an edge in J

In the updates of the dual variables in Figures 6 and 7, "+" and "−" means increasing and decreasing the dual variable by 1/2, respectively, with one exception:

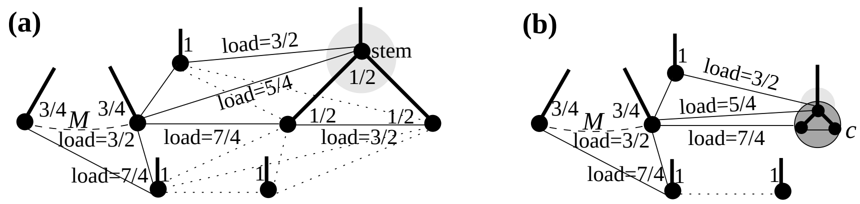

Fig. 8. Initial duals of the 3-Bunch-LP and the initial loads. The 3-bunch of the edges incident to the stem is shown by a light gray circle.

in Fig. 7(a) the updates are $y_b \leftarrow y_b - 1/2$ and $y_B \leftarrow 1/2$, where B is the 3-bunch formed by the parent T-edges of a, b, w. Similarly to Lemma 3.2 we prove that after step 2 the following holds:

(i) $\pi(c) \geq 1$ if c is an unmatched leaf or a compound node of T.

(ii) For any edge e: $\sigma(e) \leq 7/4$ if $\ell(e) = 2$, $\sigma(e) \leq 1$ if $\ell(e) = 1$, and $\sigma(e) = 0$ if $\ell(e) = 0$.

It is easy to see that the statement holds at the end of step 2, see Fig. 8; note that after step 2 the edge with load 5/4 has level 2. As in Lemma 3.2 we continue by induction while using the same notation, but focus only on the arguments that are different from the ones in Lemma 3.2.

Suppose that a semi-closed tree T' was contracted into a compound node c. Then

$$\pi'(c) \geq \left(\pi(C') + \frac{3}{2}|M'| + |U'|\right) - (|M'| + |U'|) + \Delta(y) \geq |C'| + \frac{1}{2}|M'| + \Delta(y) \ .$$

If $|C'| \geq 1$ or $|M'| \geq 2$ then $\pi'(c) \geq |C'| + \frac{1}{2}|M'| \geq 1$. If $|M'| = 0$ (Fig. 6(a)) then $\pi'(c) \geq \Delta(y) \geq \frac{1}{2}(|U'| + 1) \geq 1$. If $|M'| = 1$ then $\Delta(y) = \frac{1}{2}$ and thus $\pi'(c) \geq \frac{1}{2}|M'| + \Delta(y) \geq 1$; this is since in each one of the cases in Figures 6(b) and 7(b,c,d,e) the number of "+" signs is larger by one than the number of "−" signs, while in the case in Fig. 7(a) we gain $2 \cdot \frac{1}{2} = 1$ when increasing by $\frac{1}{2}$ the dual variable of a 3-bunch, and loose just $\frac{1}{2}$ by decreasing y_b by $\frac{1}{2}$. In all cases we have $\pi'(c) \geq 1$, as required.

We now show that property (ii) holds. Consider a semi-closed tree T' was contracted into c and an edge e with at least one endnode in T'. Note that now $\frac{3}{4}$ is the minimum difference in the bounds in (ii) due to an increase of an edge level.

Let us consider the case in Fig. 7(a). If e is incident to b or if $e = vb'$ for some $v \in T'$ then $\sigma'(e) \leq \sigma(e)$. In all the other cases we have $\ell'(e) > \ell(e)$ and $\sigma'(e) - \sigma(e) \leq \frac{1}{2} < \frac{3}{4}$. Hence the induction step holds in this case.

For the other cases, as before, let $q(e)$ denote the number of "+" signs minus the number of "−" signs in Figures 6 and 7(b,c,d,e) along the path T_e; we have $\sigma'(e) - \sigma(e) = \frac{1}{2}q(e)$ in all cases. One can verify that $q(e) \leq 0$ if e connects a leaf of T' to another leaf of T' or to a node outside T'. If e is incident to a non-leaf node of T' then $\ell'(e) > \ell(e)$ and $q(e) \leq 1$, which implies $\sigma'(e) - \sigma(e) \leq \frac{1}{2}$. This concludes the proof of (ii) and of the lemma. □

5. CONCLUSIONS

In this paper we presented an improved algorithm for TREE AUGMENTATION, by modifying the algorithm of [Adjiashvili 2017]. A minor improvement is that the algorithm is simpler, as it avoids a technical discussion on so called "early compound nodes", see [Adjiashvili 2017] and [Fiorini et al. 2017]. A more important improvement is in the running time – $4^k poly(n)$ instead of $n^{k^{O(1)}}$, where $k = \Theta(M/\epsilon^2)$. This allows ratio better than 2 also for logarithmic costs, and not only costs bounded by a constant. These two improvements are based, among others, on a more compact and simpler LP for the problem. Another important improvement is in the ratio – $\frac{12}{7} + \epsilon$ instead of $1.96418 + \epsilon$ in [Adjiashvili 2017]. This algorithm is based on a combinatorial result for spider-shaped TREE AUGMENTATION instances. We showed that for spider-shaped instances, the extreme points of the CUT-POLYHEDRON are half-integral, and thus TREE AUGMENTATION on such instances can be approximated within 4/3. As was mentioned, a related recent result of [Fiorini et al. 2017] shows that for spider-shaped instances, augmenting the CUT-LP by $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts gives an integral polyhedron and that such instances can be solved optimally in polynomial time. Overall we get that spider-shaped instances behave as "star-instances" – when T is a star (this is essentially the EDGE-COVER problem): the extreme points of the CUT-LP are half-integral, while augmenting it by $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts gives an integral polyhedron. The description of the $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts in [Fiorini et al. 2017] is somewhat complicated, and a natural question is whether using the simpler BUNCH-LP gives the same result. This is so when T is a star, c.f. [Schrijver 2004] where an equivalent EDGE-COVER problem is considered.

Our second main result is that in the case of unit costs the integrality gap of the CUT-LP is less than 2, which resolves a long standing open problem. Our goal here was just to present the simplest verifiable proof for this fact, and we believe that our bound $2 - 2/15$ can be improved by a slightly more complex algorithm and analysis. As was mentioned, several LP and SDP relaxations, more complex than the CUT-LP, were shown to have integrality gap less than 2 for particular cases (e.g., the k-BRANCH-LP with logarithmic costs). The hope was that this may lead to ratio better than 2 for the general case. Our result suggests that already the simplest CUT-LP, combined with the dual fitting method, may be the right one to study to achieve this goal. More complex LP's (e.g., the BUNCH-LP or the ODD-CUT LP) may be used to improve the ratio.

Acknowledgment. I thank Shoni Gilboa, Manor Mendel, Moran Feldman, and Gil Alon for several discussions. I also thank Chaitanya Swamy for a discussion on the BUNCH-LP and the ODD-CUT LP during FND 2014, and László Végh on further discussions on the BUNCH-LP during 2015.

REFERENCES

ADJIASHVILI, D. 2017. Beating approximation factor two for weighted tree augmentation with bounded costs. In *SODA*. 2384–2399.

APPA, G. AND KOTNYEK, B. 2004. Rational and integral k-regular matrices. *Discrete Mathematics 275,* 1-3, 1–15.

Appa, G., Kotnyek, B., Papalamprou, K., and Pitsoulis, L. 2007. Optimization with binet matrices. *Operations Research Letters 35,* 3, 345–352.

Caprara, A. and Fischetti, M. 1996. $\{0, 1/2\}$-Chvátal-Gomory cuts. *Mathematical Programming 74,* 3, 221–235.

Cheriyan, J. and Gao, Z. 2015. Approximating (unweighted) tree augmentation via lift-and-project, part II. Manuscript.

Cheriyan, J., Jordán, T., and Ravi, R. 1999. On 2-coverings and 2-packing of laminar families. In *ESA*. 510–520.

Cheriyan, J., Karloff, H., Khandekar, R., and Koenemann, J. 2008. On the integrality ratio for tree augmentation. *Operation Research Letters 36,* 4, 399–401.

Chvátal, V. 1973. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics 4,* 4, 305–337.

Cohen, N. and Nutov, Z. 2013. 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.

Cygan, M. 2016. Private communication.

Cygan, M., Fomin, F., Kowalik, F., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. 2016. *Parameterized Algorithms.* Springer.

Even, G., Feldman, J., Kortsarz, G., and Nutov, Z. 2001. A 3/2-approximation for augmenting a connected graph into a two-connected graph. In *APPROX*. 90–101.

Even, G., Feldman, J., Kortsarz, G., and Nutov, Z. 2009. A 1.8-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms 5,* 2.

Fiorini, S., Gross, M., J.Könemann, and Sanitá, L. Feb 27, 2017. A $\frac{3}{2}$-approximation algorithm for tree augmentation via chvátal-gomory cuts. `https://arxiv.org/abs/1702.05567`.

Frederickson, G. and Jájá, J. 1981. Approximation algorithms for several graph augmentation problems. *SIAM J. Computing 10*, 270–283.

Goemans, M., Goldberg, A., Plotkin, S., D. Shmoys, E. T., and Williamson, D. 1994. Improved approximation algorithms for network design problems. In *SODA*. 223–232.

Gomory, R. 1958. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society 64,* 5, 275–278.

Hochbaum, D., Megiddo, N., Naor, J., and Tamir, A. 1993. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Math. Programming 62*, 69–83.

Jain, K. 2001. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica 21,* 1, 39–60.

Khuller, S. and Thurimella, R. 1993. Approximation algorithms for graph augmentation. *J. of Algorithms 14*, 214–225.

Kortsarz, G. and Nutov, Z. 2016a. LP-relaxations for tree augmentation. In *APPROX-RANDOM*. 13:1–13:16.

Kortsarz, G. and Nutov, Z. 2016b. A simplified 1.5-approximation algorithm for augmenting edge-connectivity of a graph from 1 to 2. *ACM Transactions on Algorithms 12,* 2, 23.

Lau, L. C., Ravi, R., and Singh, M. 2011. *Iterative Methods in Combinatorial Optimization.* Cambridge University Press.

Maduel, Y. and Nutov, Z. 2010. Covering a laminar family by leaf to leaf links. *Discrete Applied Mathematics 158,* 13, 1424–1432.

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

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

Schrijver, A. 2004. *Combinatorial Optimization, Polyhedra and Efficiency.* Springer-Verlag Berlin, Heidelberg New York.