Breaching the 2-Approximation Barrier for Connectivity Augmentation: A Reduction to Steiner Tree*

JAROSŁAW BYRKA†, FABRIZIO GRANDONI‡, AND AFROUZ JABAL AMELI§

Abstract. The basic goal of survivable network design is to build a cheap network that maintains the connectivity between given sets of nodes despite the failure of a few edges/nodes. The *Connectivity Augmentation* Problem (CAP) is arguably one of the most basic problems in this area: given a k(-edge)-connected graph G and a set of extra edges (*links*), select a minimum cardinality subset A of links such that adding A to G increases its edge connectivity to $k+1$. Intuitively, one wants to make an *existing* network more reliable by *augmenting* it with extra edges. The best known approximation factor for this NP-hard problem is 2, and this can be achieved with multiple approaches (the first such result is in [Frederickson and Jájá'81]).

It is known [Dinitz et al.'76] that CAP can be reduced to the case $k = 1$, a.k.a. the *Tree Augmentation* Problem (TAP), for odd k, and to the case $k = 2$, a.k.a. the *Cactus Augmentation* Problem (CacAP), for even k. Prior to the conference version of this paper [Byrka et al. STOC'20], several better than 2 approximation algorithms were known for TAP, culminating with a recent 1.458 approximation [Grandoni et al.'18]. However, for CacAP the best known approximation was 2.

In this paper we breach the 2 approximation barrier for CacAP, hence for CAP, by presenting a polynomial-time $2\ln(4) - \frac{967}{1120} + \varepsilon < 1.91$ approximation. From a technical point of view, our approach deviates quite substantially from previous work. In particular, the better-than-2 approximation algorithms for TAP either exploit greedy-style algorithms or are based on rounding carefully-designed LPs. We instead use a reduction to the Steiner tree problem which was previously used in parameterized algorithms [Basavaraju et al.'14]. This reduction is not approximation preserving, and using the current best approximation factor for Steiner tree [Byrka et al.'13] as a black-box would not be good enough to improve on 2. To achieve the latter goal, we "open the box" and exploit the specific properties of the instances of Steiner tree arising from CacAP.

In our opinion this connection between approximation algorithms for survivable network design and Steiner-type problems is interesting, and might lead to other results in the area.

*A PRELIMINARY VERSION OF THIS PAPER APPEARED IN THE PROCEEDINGS OF STOC'20.

†University of Wrocław, Poland, jby@cs.uni.wroc.pl. Supported by the NCN grant number 2015/18/E/ST6/00456.

‡IDSIA, USI-SUPSI, Switzerland, fabrizio@idsia.ch. Partially supported by SNF Excellence Grant 200020B_182865/1.

§Eindhoven University of Technology, Netherlands, a.jabal.ameli@tue.nl. Partially supported by SNF Excellence Grant 200020B_182865/1.

1. Introduction. The basic goal of *Survivable Network Design* is to construct cheap networks that provide connectivity guarantees between pre-specified sets of nodes even after the failure of a few edges/nodes (in the following we will focus on the edge failure case). This has many applications, e.g., in transportation and telecommunication networks.

The *Connectivity Augmentation Problem* (CAP) is among the most basic survivable network design problems. Here we are given a k-(edge)-connected[1] undirected graph $G = (V, E)$ and a collection L of extra edges (*links*). The goal is to find a minimum *cardinality* subset $OPT \subseteq L$ such that $G' = (V, E \cup OPT)$ is $(k + 1)$-connected. Intuitively, we wish to augment an existing network to make it more resilient to edge failures. Dinitz et al. [12] (see also [9, 24]) presented an approximation-preserving reduction from this problem to the case $k = 1$ for odd k, and $k = 2$ for even k. This motivates a deeper understanding of the latter two special cases.

The case $k = 1$ is also known as the *Tree Augmentation Problem* (TAP). The reason for this name is that any 2-edge-connected component of the input graph G can be contracted, hence leading to a tree. For this problem several better than 2 approximation algorithms are known [1, 7, 8, 13, 14, 19, 26, 27, 29]. Among these results the best approximation factor prior to the conference version of this paper [4] was 1.458 due to Grandoni et al. [19] (see Section 1.3 for subsequent developments).

The case $k = 2$ is also known as the *Cactus Augmentation Problem* (CacAP), where for similar reasons we can assume that the input graph is a cactus[2]. Here the best-known approximation factor previous to our work [4] was 2, and this factor was achieved with multiple approaches [15, 17, 23, 24]. A better approximation was achieved recently for the special case where the input cactus is a cycle [20].

1.1. Our Results and Techniques. The main result of this paper is the first better than 2 approximation algorithm for CacAP, hence for CAP.

Theorem 1.1. *For any constant $\varepsilon > 0$, there is a polynomial-time $2\ln(4) - \frac{967}{1120} + \varepsilon < 1.9092 + \varepsilon$ approximation algorithm for the Cactus Augmentation problem.*

From Theorem 1.1 and the reduction to CacAP implied by [12], we get:

Corollary 1.2. *For any constant $\varepsilon > 0$, there is a polynomial-time $2\ln(4) - \frac{967}{1120} + \varepsilon < 1.9092 + \varepsilon$ approximation algorithm for the Connectivity Augmentation problem.*

Our result is based on a reduction to the (cardinality) Steiner tree problem by Basavaraju et al. [3]. The authors use this connection to design improved parameterized algorithms (see also [28] for a related result). Recall that in the *Steiner tree* problem we are given an undirected graph $G_{ST} = (T \cup S, E_{ST})$, where T is a set of t *terminals* and S a set of *Steiner nodes* (disjoint from T). Our goal is to find a tree (*Steiner tree*) $OPT_{ST} = (T \cup A, F)$ that contains all the terminals (and possibly a subset of Steiner nodes A) and has the minimum possible number of edges $|OPT_{ST}|$. Basavaraju et al. [3] observed that, given a CacAP instance $(G = (V, E), L)$, it is possible to construct (in polynomial time) an *equivalent* Steiner tree instance $G_{ST} = (T \cup L, E_{ST})$ (see also the description in [30]). Here T corresponds to the nodes of degree 2 in G, L is the set of Steiner nodes, and the edges E_{ST} are defined properly (more details in Section 2.1). Intuitively, each link node $\ell \in L$ is adjacent to the terminal nodes in T which are endpoints of ℓ (if any), and two link nodes $\ell, \ell' \in L$ are adjacent iff the respective endpoints cannot be separated by a min-cut of G. In particular, an optimal solution to G_{ST} induces an optimal solution to (G, L) and vice versa. An example of the reduction is given in Figure 2.1. Unfortunately, this reduction is not approximation-preserving. In particular, by working out the simple details (see also Section 2.1), one obtains that a ρ_{ST}-approximation for Steiner tree implies a $\rho \leq 3\rho_{ST} - 2$ approximation for CacAP. The current best value of ρ_{ST} is $\ln 4 + \varepsilon < 1.39$ due to Byrka, Grandoni, Rothvoss and Sanità [5]. Hence this is not good enough[3] to obtain $\rho < 2$.

In order to obtain our main result we use the same algorithm as in [5], but we analyze it differently. In particular, we exploit the specific structure of the instances of Steiner tree arising from CacAP instances via the above reduction to get a substantially better approximation factor.

[1] We recall that $G = (V, E)$ is k-connected if for every subset of edges $F \subseteq E$, $|F| \leq k - 1$, the graph $G' = (V, E \setminus F)$ is connected.

[2] We recall that a *cactus* G is a connected undirected graph in which every edge belongs to exactly one cycle. For technical reasons it is convenient to allow length-2 cycles consisting of 2 parallel edges.

[3] One would need $\rho_{ST} < \frac{4}{3}$ here. Notice that this is not ruled out by the current lower bounds on the approximability of Steiner tree.

In more detail (see also Section 3), in the analysis of the algorithm in [5] one considers an optimal Steiner tree solution $OPT_{ST} = (T \cup A, F)$ rooted at some arbitrary node r, marks a random subset $F_{mar} \subseteq F$ of edges so that each Steiner node is connected to some terminal via marked edges, and based on F_{mar} defines a proper (random) *witness set* $W(e)$ for each $e \in F$. The cost of the approximate solution turns out to be at most $(1+\varepsilon)\sum_{e\in F} E[H_{|W(e)|}]$, where $H_i := 1 + \frac{1}{2} + \ldots + \frac{1}{i}$ is the i-th harmonic number. In particular, the authors show that $E[H_{|W(e)|}] \leq \ln 4$ for each $e \in F$, hence the claimed approximation factor.

Our analysis of the algorithm deviates from [5] for the following critical reasons:

1. They (i.e., the authors of [5]) mark one child edge of each Steiner node chosen uniformly at random. In our case it is convenient to *favor* child edges with one terminal endpoint (if any). The fact that this helps is not obvious in our opinion.
2. As mentioned above, they provide a per-edge upper bound on $E[H_{|W(e)|}]$. We rather need to average over multiple edges in order to achieve a good bound. Finding a good way to do that is not trivial in our opinion.

We remark that, from a technical point of view, our result deviates quite substantially from prior approximation algorithms for TAP. The first improvements on a 2 approximation where achieved via greedy-style algorithms and a complex case analysis [13, 26, 27, 29]. More recent approaches are based on rounding stronger and stronger LP (or SDP) relaxations for the problem [1, 7, 8, 14, 19]. We also use an LP-based rounding algorithm, which is however defined for a generic Steiner tree instance (while the properties of TAP are used only in the analysis). In our opinion the connection that we established between the approximability of survivable network design problems and Steiner-type problems might lead to other results in the future.

1.2. Related and Previous Work. One can consider a natural weighted version $WCAP$ of CAP where each link has a positive weight and the goal is to minimize the total weight of selected links. The best-known approximation for WCAP is 2. The techniques used in this paper seem not to generalize to the weighted case. In particular, one might use a reduction to a node-weighted version of the Steiner tree problem, however the latter problem is harder and in general allows only a logarithmic approximation [25].

Prior to the conference version of this paper, 2 was the best-known approximation factor even for the weighted version WTAP of TAP. Some progress on weighted TAP was made in the case of small integer weights. In particular, when the largest weight W is upper bounded by a constant, better than 2 approximation algorithms are given in [1, 14, 19]. A technique in [32] allows one to extend these results to $W = O(\log n)$. Weighted TAP also admits a $1 + \ln 2$ approximation for arbitrary weights if the input tree has constant radius [11]. A better than 2 approximation can also be achieved if the fractional solution to a natural LP relaxation has non-zero entries bounded away from zero [22].

A problem closely related to CAP is to build a minimum cost k-edge-connected spanning subgraph of a given input graph [10, 16, 21, 34]. Here the best known approximation factor is $4/3$ for the unweighted case, and 2 for the weighted one.

1.3. Subsequent Work. After the publication of the conference version of this paper [4], there were a few breakthroughs in the area of survivable network design. Cecchetto et al. [6] developed an elegant and unified framework to approximate both TAP and CacAP, leading to a 1.393-approximation. Notice that this does not only greatly improves on our approximation factor for CacAP, but also on the approximation factor for TAP in [19]. Though it leads to weaker approximation factors, we believe that our connection between survivable network design and Steiner-type problems might be useful in future related work. Indeed, after our work, a similar connection was made by Nutov [31]. In more detail, he proved that an α approximation for Steiner tree implies a $1+\ln(4-x)+\varepsilon$ approximation for CAP, where x is the solution to $1+\ln(4-x) = \alpha + (\alpha-1)x$. This leads to an approximation factor below 2 (though worse than the one achieved in this work) using the current best approximation for Steiner tree [5]. Notice that, differently from our result, Nutov's reduction is black box, a useful feature.

Recently, Traub and Zenklusen [35] achieved a $1 + \ln 2 + \varepsilon$ approximation for WTAP , hence solving a major open problem in the area. The authors later improved their result to a $1.5 + \varepsilon$-approximation [36]. Achieving a better than 2 approximation for WCAP remains a challenging open problem.

Grandoni, Jabal Ameli and Traub [18] obtained the first better than 2 approximation for the Forest Augmentation Problem (FAP), i.e. the generalization of TAP where the input graph is a forest rather than a tree.

A variant of our approach was used by Nutov [30] to obtain, among other results, the first better than

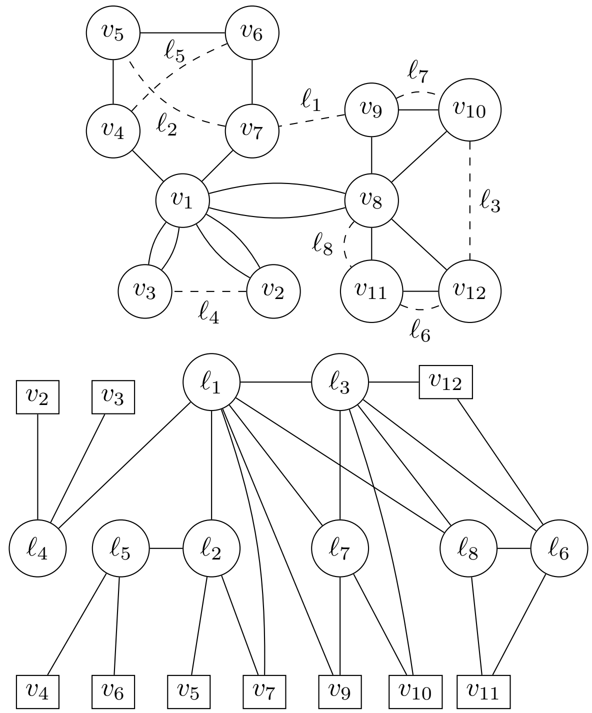

FIG. 2.1. *(top) Instance of* CacAP*, where dashed edges denote links. The projections of* ℓ_1 *are* $proj(\ell_1) = \{\{v_7, v_1\}, \{v_1, v_8\}, \{v_8, v_9\}\}$*. Link* ℓ_2 *is crossing with* ℓ_1 *and* ℓ_5*. (bottom) The Corresponding Steiner tree instance, where square nodes denote terminals.*

2 approximation for the node-connectivity version of TAP. The approximation factor for this problem was later improved by Angelidakis, Hyatt-Denesik, and Sanità [2], again using an approach similar to the one in this paper.

2. Steiner Tree and Connectivity Augmentation.

In this section we present the mentioned reduction in [3] from CacAP to Steiner tree (Section 2.1). Furthermore, we describe a specific Steiner tree approximation algorithm that we will use to solve the instance arising from the above reduction (Section 3). We analyze the resulting approximation factor in Section 4.

2.1. A Reduction to Steiner Tree.

Consider a CacAP instance $(G = (V, E), L)$. For a link $\ell = (v_0, v_{q+1})$, let $v_1, \ldots, v_q$ be the sequence of nodes of degree at least 4 that lie along every simple v_0-v_{q+1} path, excluding the endpoints (that may or may not have this property). Intuitively, any simple path from v_0 to v_{q+1} will contain edges from some cycles $C_0, \ldots, C_q$ in this order. Then v_i, $i \in \{1, \ldots, q\}$, is the (cut) node shared by the cycles C_{i-1} and C_i. Notice that each pair $\ell_i = \{v_i, v_{i+1}\}$, $i = 0, \ldots, q$, lies along a distinct cycle C_i visited by the mentioned path. We call each such ℓ_i the *projection* of ℓ on C_i, and we let $proj(\ell)$ be the set of all projections of ℓ. Consider two links $\ell = \{x, y\}$ and $\ell' = \{x', y'\}$. Then we say that ℓ and ℓ' *cross* if one of the following two conditions hold: (1) they share one endpoint or (2) there exists a cycle C such that C includes x, y, x' and y' and taking one simple x-y path P along C, P contains exactly one node in $\{x', y'\}$ as an internal node. We say that any two links ℓ and ℓ' *cross* if there exist $\ell_i \in proj(\ell)$ and $\ell'_j \in proj(\ell')$ such that ℓ_i and ℓ'_j cross. See Figure 2.1 (top) for an example.

From (G, L) we construct a Steiner tree instance $G_{ST} = (T \cup S, E_{ST})$ as follows. For each one of the t nodes v of degree 2 in G, add a terminal v to T; for each link $\ell \in L$, add a Steiner node ℓ to S (i.e., $S = L$); for each $\ell \in L$ and endpoint $v \in T$ of ℓ, add $\{\ell, v\}$ to E_{ST}; finally, for any two links ℓ and ℓ' that *cross*, add $\{\ell, \ell'\}$ to E_{ST}. See Figure 2.1 (bottom) for an example. We observe the following simple facts.

PROPERTY 1. *Each Steiner node* s *is adjacent to at most* 2 *terminals, namely the terminals corresponding to the endpoints of the link associated with* s*.*

PROPERTY 2. *The neighbors of each terminal* t *are Steiner nodes which form a clique: this clique cor-*

responds to the links that share the node corresponding to t as a common endpoint.

We will critically exploit the following lemma sketched in [3] (Lemma 1). For the sake of completeness we give a (more detailed) proof of it in Appendix B.

LEMMA 2.1. *[3] $A \subseteq L$ is a feasible solution to a* CacAP *instance (G, L) iff, in the corresponding Steiner tree instance $G_{ST} = (T \cup L, E_{ST})$, $G_{ST}[T \cup A]$ is connected.*

Notice that the above reduction is not approximation-preserving. Still, we can state the following.

COROLLARY 2.2. *Any optimum solution OPT to the input* CacAP *instance, induces a solution OPT_{ST} of cost $|OPT_{ST}| = |OPT| + t - 1$ for the associated Steiner tree instance, where t is the number of terminals in the latter instance. Vice versa, given a solution APX_{ST} to the Steiner tree instance, one can construct in polynomial time a solution APX to the input* CacAP *instance with $|APX| = |APX_{ST}| - t + 1$.*

Proof. Both claims follow directly from Lemma 2.1. For the first claim, it is sufficient to observe that a spanning tree of $G_{ST}[T \cup OPT]$ contains $t + |OPT| - 1$ edges. For the second claim, observe that the Steiner nodes in APX_{ST} induce a feasible solution to CacAP. The claim follows since $|APX_{ST}| = s + t - 1$, where s is the number of Steiner nodes in APX_{ST}. ◻

We will exploit also the following simple fact.

LEMMA 2.3. *Given some optimal solution OPT to a* CacAP *instance, there is a feasible solution OPT_{ST} to the associated Steiner tree instance with $|OPT_{ST}| = |OPT| + t - 1$ where terminals have degree exactly 1 (namely, all the terminals are leaves).*

Proof. Given any feasible solution ST to the problem, we can transform it into a solution ST' of the same cost where some terminal v of degree $d(v) \geq 2$ in ST has degree $d(v) - 1$ in ST'. In order to do that, consider any terminal v adjacent to two Steiner nodes ℓ and ℓ' in ST. By Property 2, ℓ and ℓ' are adjacent. Hence $ST' := ST \cup \{\ell, \ell'\} \setminus \{v, \ell'\}$ is a feasible Steiner tree of the same cost and with the desired property.

By iteratively applying the above process to the solution OPT_{ST} guaranteed by Corollary 2.2 one obtains the desired solution. ◻

As mentioned earlier, a ρ_{ST} approximation for Steiner tree (used as a *black box*) provides a $3\rho_{ST} - 2$ approximation for CacAP by the above construction. Indeed, the Steiner tree instance has cost at most $|OPT| + t - 1$ by Corollary 2.2, hence an approximate solution APX_{ST} would cost at most $\rho_{ST}(|OPT| + t - 1)$. By the same corollary, we can convert this into a solution APX to CacAP of cost at most $\rho_{ST}(|OPT| + t - 1) - t + 1$. Next observe that $|OPT| \geq t/2$. Indeed, any node of degree 2 in the CacAP instance needs to have at least one link incident to it in a feasible solution, and a link can be incident to at most 2 such nodes. Thus $|APX| \leq 3\rho_{ST}|OPT| - 2|OPT|$. In order to improve on this simple bound, we will have to *open the box*.

Let us remark that the size $|APX|$ of the approximate solution for CacAP that we will compute is precisely the number of Steiner nodes involved in the solution APX_{ST} that we compute for the corresponding Steiner tree instance. In our 1.91 approximation we implicitly address all the Steiner tree instances satisfying Properties 1 and 2. Therefore, we implicitly achieve the same approximation factor for the latter instances of Steiner tree, where the objective function is to minimize the number of Steiner nodes in the computed Steiner tree.

3. Steiner Tree via Iterative Randomized Rounding. As we mentioned in the introduction, the current best $(\ln 4 + \varepsilon)$-approximate Steiner tree algorithm from [5], used as a black box, is not good enough to break the 2-approximation barrier for CacAP. However, it turns out that the same algorithm achieves this goal in combination with a different analysis that exploits the properties of the specific Steiner tree instances arising from CacAP.

We first sketch the basic property of the algorithm and analysis in [5] that we need here and express it in the form of Lemma 3.1. For the sake of completeness, we include a more detailed description and a sketch of the proof of the lemma in the following Section 3.1 . The authors of [5] consider an LP relaxation DCR_k for the problem based on *directed k-components* for a proper constant parameter k depending on ε. They iteratively solve this LP, sample a directed k-component C with probability proportional to the LP values, and contract C. The process ends when all terminals are contracted into one node. This algorithm can be derandomized, and the deterministic version is good enough for our application. We do not need more details about this algorithm, other than that it runs in polynomial time.

In the analysis (more details in Sections 3.1 and 3.2) the authors of [5] consider any feasible Steiner tree $ST = (T \cup A, F)$. They interpret each full component[4] S' of ST as a tree rooted at some Steiner node r of S' (if there is no such node, it can be created by splitting the single edge in S'). Then the authors define a *marking scheme* where some child edge of each internal (Steiner) node of S' is marked. Notice that the marked edges induce a collection of disjoint paths in each full component S': such paths span the nodes of S' and each such path contains precisely one terminal (as an endpoint). A given marking scheme defines a *witness set* $W(e)$ for each edge e in S': this consists of all the pairs of terminals $\{t', t''\}$ in S' such that the t'-t'' path in S' contains e and precisely one unmarked edge. We let $w(e) = |W(e)|$. Notice that for each unmarked edge e there exists exactly one such t'-t'' path, hence $w(e) = 1$ (we will later use this property in Lemma 4.1). Then the authors prove the following, where $H_i := 1 + \frac{1}{2} + \ldots + \frac{1}{i}$ is the i-th harmonic number.

LEMMA 3.1. *[5] For any feasible Steiner tree $ST = (T \cup A, F)$ and marking scheme, for a large enough parameter $k = O_\varepsilon(1)$, the cost of the solution computed by the above algorithm is at most $(1+\varepsilon)\sum_{e\in F} E[H_{w(e)}]$.*

3.1. Some Details About the Steiner Tree Approximation Algorithm in [5]. For a complete presentation of the Steiner tree algorithm we refer to the original paper [5]. Here we sketch the main ideas. The algorithm is based on the following Directed Component Relaxation (DCR) of the Steiner tree problem.

$$\text{(3.1)} \qquad \min \sum_{C\in\mathcal{C}} c(C)x_C \qquad \text{(DCR)}$$

$$\text{(3.2)} \qquad \text{s.t.} \sum_{C\in\delta^+_{\mathcal{C}}(U)} x_C \geq 1 \qquad \forall \emptyset \neq U \subseteq T \setminus \{r\}$$

$$\text{(3.3)} \qquad x_C \geq 0 \qquad \forall\, C \in \mathcal{C}.$$

Here $\mathcal{C}$ is the set of directed components, where each directed component C is a minimum-cost Steiner tree (of cost $c(C)$) over a subset of terminals. Furthermore, the leaves of C are precisely its terminals, and C is directed towards a specific terminal: the sink of C, and the remaining terminals are the sources of C. Intuitively, our goal is to buy a minimum-cost subset of directed components so that they induce a directed path from each terminal to the root. In more detail, for any cut U that separates some non-root terminal from the root, let $\delta^+_{\mathcal{C}}(U)$ be the set of components with some source in U and the sink not in U. Then every feasible solution has to buy some component in $\delta^+_{\mathcal{C}}(U)$. The DCR relaxation follows naturally.

After restricting DCR to solutions that only use components with at most k terminals we obtain DCR_k. For constant k, DCR_k has a polynomial number of variables. Furthermore, the separation problem can be solved in polynomial time via a reduction to minimum cut. Therefore DCR_k can be solved in polynomial time. Moreover, the value of DCR_k is known to be a $(1+\epsilon)$-approximation of the value of DCR for large enough $k = O_\varepsilon(1)$.

The iterative randomised rounding algorithm from [5], until all terminals are connected to the root, in iterations $t = 1, 2, 3 \ldots$, does the following:

- solve DCR_k for the current instance of the Steiner tree problem to get x^t;
- sample a component C^t from $\mathcal{C}_k$ with probability proportional to x^t_C;
- contract the sampled component C^t.

For the ease of the analysis, by adding dummy components w.l.o.g, one may assume that the total number of components in the fractional solution remains constant across the iterations of the algorithm, i.e., $\sum_{C\in\mathcal{C}} x^t_C = M$ for a proper M for all $t = 1, 2, \ldots$. It is argued that after t iterations of the algorithm, having bought the first t sampled components, the residual instance of the problem is expected to be less costly. To this end a reference solution S^t is constructed such that $S^t \cup \bigcup_{t'=1}^{t-1} C^t$ connects all the terminals. The initial reference solution $S^1 = OPT_{ST}$ is an optimal solution to the Steiner tree instance of cost *opt*. Consecutive reference solutions $S^2, S^3, \ldots$ are obtained by gradually deleting edges that are no longer necessary due to the connectivity provided by the already sampled components.

Key to estimate the expected cost of the final solution is to bound the number of iterations until a particular edge $e \in S^1$ can be removed. Define $D(e) = \max\{t | e \in S^t\}$. In [5] (proof of Theorem

[4] Recall that a full component is a maximal subtree whose terminals are exactly its leaves.

21) it is shown that there exist a randomised process of constructing reference solutions $S^1, S^2, \ldots$ such that $E[D(e)] \leq \ln(4) \cdot M$, which allows one to bound the total expected cost of sampled components as $E\left[\sum_{t\geq 1} c(C^t)\right] \leq (\ln(4)+\epsilon)\cdot opt$. Note that the above *per-edge* guaranty allows for easily handling arbitrary costs of individual edges. In our application to (unweighted) CacAP, we need to average over multiple edges to achieve a good enough bound.

3.2. Witness Tree and Witness Sets. We next slightly abuse notation and sometimes denote in the same way a tree and its set of edges. The construction of reference solutions $S^1, S^2, \ldots$ is not trivial. It involves:

- construction of a terminal spanning tree W, called the *witness tree*, based on randomised marking (selection) of a subset of edges of S^1. Each edge e of S^1 is associated with a proper subset $W(e) \subseteq W$ (witness set of e);
- randomised deletion of a proper subset of W in response to selecting a particular component C^t in iteration t;
- removing an edge e from S^t when all edges $W(e)$ have already been deleted.

In the following we discuss the main ideas behind our approach and the key properties of each of the three above mentioned processes.

Construction of the witness tree. The high level idea behind the witness tree is that we need to always satisfy the condition that $S^t \cup \bigcup_{t'=1}^{t-1} C^t$ connects all the terminals, which is that the remaining fragments of the initial reference solution S^1 together with the already sampled components must provide sufficient connectivity. To this end a simpler object providing connectivity is constructed. It is an auxiliary tree W whose node set is the terminals of the instance (while the edges of W are not necessarily edges of the input graph). It will be easier to delete edges from W in response to sampling components rather than deleting them directly from S^t.

We will now discuss methods to construct W. Intuitively, removing edges from a Steiner tree (in response to receiving connectivity from a component) is directly possible for only a subset of edges of the Steiner tree. In particular it appears more difficult to remove a Steiner node (and hence a path connecting a Steiner node to a terminal). This is related to the concept of *Loss* and *Loss contracting algorithms* (see, e.g., [33]), where one accepts that the cost of the system of paths connecting Steiner nodes to terminals is not removable.

Consider the following procedure: For each full component S' of the Steiner tree S^1 select a single Steiner node r and interpret S' as a tree rooted at r . For every Steiner node s of S', mark one edge between s and one of its children. Note that for each Steiner node s the marked edges will form a unique path towards a leaf containing terminal $t(s)$. Note also that connected components formed by the marked edges will all contain a single terminal node. Construct $W(S')$ by adding to $E(W(S'))$ an edge $\{t(u), t(v)\}$ for each unmarked edge $\{u, v\}$ of S'.[5] Observe that the above constructed graph $W(S')$ is a tree spanning the terminals of S'. By repeating this procedure for all full components of S^1 we obtain a tree W spanning all terminals of the Steiner tree instance.

So far we did not specify how to select the edge below the Steiner node $v \in S'$ to be marked. In [5] the tree was assumed to be binary, and the edge would be selected at random by tossing a fair coin. In the current paper we use a different marking strategy as discussed in Section 4.1.

Removing edges of the witness tree. When edges of the witness tree W become unnecessary, we remove them. We keep the invariant that the (not removed) edges of W together with the already collected components are sufficient to connect all terminals. Still, given a fixed collection of the already sampled components, the choice of which edges of W to remove is not obvious. In [5] a randomised scheme was considered. It was shown (Lemma 19 in [5]) that there exists a random process removing edges from W in response to sampled components, such that for every edge $e \in W$ not removed before iteration t, the probability that it is removed in iteration t is at least $1/M$. In the current work we continue using the mentioned "uniform" witness tree edge removing process, and utilise the following lemma.

Lemma 3.2 (lemma 20 in [5]). *Let $\tilde{W} \subseteq W$. Then the expected number of iterations until all edges in $\tilde{W}$ are removed is at most $H_{|\tilde{W}|} \cdot M$.*

[5]Note that in [5] the role of marked and unmarked edges was reversed. It was irrelevant for the analysis in [5] as it was assumed that the tree S' is binary. In this paper however we will exploit the high degree of Steiner nodes in S' and hence prefer to mark the "Loss" edges.

Removing edges of the reference tree S^t. Which edges of the reference tree can be removed? Clearly it suffices if S^t provides the same terminal connectivity as the not removed edges of the witness tree W. Note that a single edge $e \in W$ corresponds to a single path $p(e)$ in S^1. It then suffices to keep the edges of S^1 that occur in a path $p(e)$ of at least one (still not removed) edge $e \in W$.

We introduce the following notation: for an edge f in S^1 let $W(f) = \{e \in W | f \in p(e)\}$, we call $W(f)$ the *witness set* of f. Therefore, at iteration t, the reference solution S^t contains the edges from S^1 whose witness sets are not fully removed until iteration $t-1$.

Observe that the expected value of the number $D(f)$ of iterations an edge f from the reference solution survives (until being removed) can be expressed using only the size of its witness set $W(f)$.

Corollary 3.3. *Let $f \in S^1$, then $E[D(f)] \le H_{|W(f)|} \cdot M$.*

Following the argument from the proof of Theorem 21 in [5], we also get

Corollary 3.4. *For $k = O_\varepsilon(1)$ large enough, the total expected cost of the components bought by the algorithm is at most*

$$\frac{1+\varepsilon}{M} \sum_{f \in S^1} E[D(f)] \cdot c(f) \le (1+\epsilon) \cdot \sum_{f \in S^1} H_{|W(f)|} \cdot c(f)$$

Therefore, it suffices to analyse how the marking scheme used in the construction of the witness tree affects the distributions of the sizes of the witness sets for the individual edges of S^1. To this end we will exploit two properties of our instances: the high degree of the Steiner nodes in the initial optimal solution S^1, and the fact that all the edges of S^1 have the same cost.

4. An Improved CacAP Approximation Algorithm.

In this section we present our improved approximation for CacAP. The algorithm is rather simple: we just build the Steiner tree instance $G_{ST} = (T \cup L, E_{ST})$ associated with the input CacAP instance (G, L) and compute an approximate solution APX_{ST} to G via the algorithm in [5] sketched in Section 3. Then we derive from APX_{ST} a feasible solution APX to the input CacAP instance as described in Corollary 2.2. We let *apx* denote the approximation ratio of this algorithm.

In Section 4.1 we describe our alternative marking scheme and prove some of its properties. In Section 4.2 we complete the analysis of the approximation factor.

4.1. An Alternative Marking Scheme.

Recall that in the analysis of the Steiner tree approximation algorithm in [5], one can focus on a specific feasible Steiner tree ST and on a specific marking scheme (so that Steiner nodes are connected to some terminal via paths of marked edges). Le OPT be some optimal solution to the considered CacAP instance. As a feasible solution ST we consider the solution $OPT_{ST} = (T \cup OPT, F)$, of cost $|OPT| + t - 1$ and with terminals being leaves, guaranteed by Lemma 2.3.

We mark edges in the following way. Consider each full component S' of OPT_{ST}. W.l.o.g, S' contains at least one Steiner node (otherwise, we can create it by splitting one edge). Let us root S' at some Steiner node r which is adjacent to at least one terminal (notice that such r must exist). For a Steiner node ℓ, we let $d(\ell)$, $s(\ell)$ and $t(\ell)$ be the number of its children, Steiner children, and terminal children, resp. In particular $d(\ell) = s(\ell) + t(\ell)$ and (by Property 1) $t(\ell) \le 2$.

For each link node ℓ, there are two options. If ℓ has at least one terminal child, we select one such child t uniformly at random, and mark edge $\{\ell, t\}$. Otherwise, we choose a child ℓ' of ℓ (ℓ' being a Steiner node) uniformly at random, and mark edge $\{\ell, \ell'\}$. Notice that this is a feasible marking scheme, namely for each Steiner node we mark exactly one child edge. Observe also that in our marking we *favor* edges connecting Steiner nodes to terminals: this will be critical in our analysis[6]. See Figure 4.1 for a possible marking of this type.

Let APX_{ST} be the Steiner tree computed by the algorithm. Let F_{mar} and F_{unm} be the (random) sets of marked and unmarked edges, resp., that partition F. Recall that for each $e \in F$, there exists a (random) witness set $W(e)$ of size $w(e) = |W(e)|$. Observe that each Steiner node ℓ has precisely one marked child edge $m(\ell)$. We let the *cost* $c(\ell)$ of ℓ be $E[H_{w(m(\ell))}]$. The following bound on the approximation ratio holds.

Lemma 4.1. *$apx \le 2\varepsilon + \frac{1+\varepsilon}{|OPT|} \sum_{\ell \in OPT} c(\ell)$.*

[6] While we are able to show that our marking scheme leads to a better than 2 approximation, we are not able to show that the same cannot be achieved with the original marking scheme in [5].

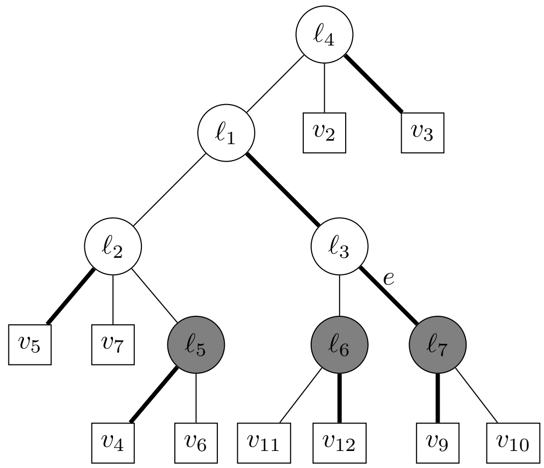

FIG. 4.1. *A feasible Steiner tree for the instance of Figure 2.1, which happens to be well-structured. Bold edges denote a possible marking. One has* $m(\ell_3) = e := \{\ell_3, \ell_7\}$*, and* $W(e)$ *contains* $\{v_9, v_{12}\}$*,* $\{v_9, v_5\}$ *and* $\{v_9, v_3\}$*. Notice that* $w(e) = |W(e)| = d(\ell_3)+d(\ell_1)-1$*. Leaf-Steiner nodes are drawn in grey. Here* ℓ_2 *(resp.,* ℓ_3*) is a good (resp., bad) father. Consequently* ℓ_5 *(resp.,* ℓ_6*) is good (resp., bad). A feasible grouping is* $g(\ell_2) = \{\ell_2\}$*,* $g(\ell_3) = \{\ell_3, \ell_7\}$*,* $g(\ell_1) = \{\ell_1, \ell_6\}$*,* $g(\ell_4) = \{\ell_4\}$*, and* $g(\ell_5) = \{\ell_5\}$*.*

Proof. Recall that by Lemma 3.1 the expected cost of the computed Steiner tree APX_{ST} is, modulo a factor $(1+\varepsilon)$, at most

$$E[\sum_{e\in F} H_{w(e)}] = E[\sum_{e\in F_{mar}} H_{w(e)} + \sum_{e\in F_{unm}} H_{w(e)}]$$
$$=E[\sum_{e\in F_{mar}} H_{w(e)} + |F_{unm}|] = E[\sum_{e\in F_{mar}} H_{w(e)}] + t - 1.$$

In the second-last equality above we used the fact that $w(e) = 1$ deterministically for an unmarked edge, and in the last equality above the fact that there are precisely $|OPT|$ marked edges and consequently exactly $t-1$ unmarked ones. From APX_{ST} we derive a feasible solution APX to the input instance of cost $|APX| = |APX_{ST}| - t + 1$ by Corollary 2.2. Hence

$$|APX| \le (1+\varepsilon)(E[\sum_{e\in F_{mar}} H_{w(e)}] + t - 1) - t + 1$$
$$\le (1+\varepsilon)E[\sum_{e\in F_{mar}} H_{w(e)}] + 2\varepsilon|OPT|.$$

In the last inequality above we used the trivial lower bound $|OPT| \ge t/2$ that we mentioned earlier. The claim follows since by definition $\sum_{e\in F_{mar}} E[H_{w(e)}] = \sum_{\ell\in OPT} c(\ell)$. ∎

From the above lemma, modulo factors $(1+\varepsilon)$, the approximation ratio of our algorithm is given by the *average cost* of Steiner nodes. The following lemma gives a generic upper bound on the cost for each non-root Steiner node based on the degree sequence of its ancestors[7].

LEMMA 4.2. *Given a non-root Steiner node* ℓ*, let* ℓ_q *be the lowest proper ancestor*[8] *of* ℓ *with* $t(\ell_q) > 0$*. Let* $\ell = \ell_1, \ell_2, \dots, \ell_q$*,* $q \ge 2$*, be the simple path between* ℓ *and* ℓ_q*, and let* $d_i = d(\ell_i)$*. Then*[9]

$$c(\ell) = \sum_{h=1}^{q-2} \frac{(d_{h+1}-1)H_{d_1+\ldots+d_h-h+1}}{d_2\cdot\ldots\cdot d_{h+1}} + \frac{H_{d_1+\ldots+d_{q-1}-q+2}}{d_2\cdot\ldots\cdot d_{q-1}}.$$

[7] Observe that for the root r, $c(r) = H_{d(r)-1}$ deterministically.
[8] Observe that this ancestor exists since the root has this property by assumption.
[9] The value of a product of type $a_i \cdot a_{i+1} \cdot \ldots \cdot a_j$ for $j < i$ is assumed to be 1 by definition.

Proof. By definition $c(\ell) = c(\ell_1) = E[H_{w(e)}]$, where $e = m(\ell_1) = \{\ell_1, \ell_0\}$ is the marked child edge of ℓ_1. Recall that $W(e)$ contains one entry for each path in the tree between terminals that contains e and precisely one unmarked edge. In our specific case, condition on $\{\ell_0, \ell_1\}, \{\ell_1, \ell_2\}, \ldots, \{\ell_{h-1}, \ell_h\}$ being a maximal sequence of consecutive marked edges. Notice that by construction $\{\ell_{q-1}, \ell_q\}$ is unmarked (since ℓ_q has a terminal child by definition), hence $h \le q-1$. In this case $w(e) = d_1 + \ldots + d_h - (h-1)$. For $h < q-1$, the mentioned event happens with probability $\frac{1}{d_2} \cdot \ldots \cdot \frac{1}{d_h} \cdot \frac{d_{h+1}-1}{d_{h+1}}$. For $h = q-1$, this probability is $\frac{1}{d_2} \cdot \ldots \cdot \frac{1}{d_h}$. The claim follows by computing the expectation of $H_{w(e)}$. ∎

We next provide an upper bound on $c(\ell)$ as a function of $d(\ell)$ only. Let us define the following variant of H_i:

$$\hat{H}_i := \frac{1}{2}H_i + \frac{1}{4}H_{i+1} + \ldots = \sum_{j \ge 0} \frac{1}{2^{j+1}} H_{i+j}.$$

One has that $\hat{H}_1 = \ln(4)$ and $\hat{H}_{j+1} = 2\hat{H}_j - H_j$. Notice that, modulo an additive ε, $\hat{H}_1$ is precisely the approximation factor for Steiner tree achieved in [5]. The first few approximate values of $\hat{H}_i$ are $\hat{H}_1 < 1.3863$, $\hat{H}_2 < 1.7726$, $\hat{H}_3 < 2.0452$, $\hat{H}_4 < 2.2571$, $\hat{H}_5 < 2.4308$, $\hat{H}_6 < 2.5781$, $\hat{H}_7 < 2.7062$, and $\hat{H}_8 < 2.8195$.

The proof of the following lemma, though not entirely trivial, is mostly based on algebraic manipulations and therefore we postpone it to Appendix A.

Lemma 4.3. *For any $\ell \in OPT$, $c(\ell) \le \hat{H}_{d(\ell)}$.*

In next subsection we will see that for a carefully defined subset of Steiner nodes ℓ it is possible to obtain a better upper bound on $c(\ell)$ than the one provided by Lemma 4.3. This will be critical in our analysis since the latter bound is not strong enough.

4.2. Analysis of the Approximation Factor. In this section we upper bound the approximation factor *apx* as given by Lemmas 4.1 and 4.2. In order to simplify our analysis, it is convenient to focus our attention on a specific class of well-structured Steiner trees OPT_{ST} (see also Figure 4.1). The following lemma shows that this is (essentially) w.l.o.g.

Definition 4.4. *A rooted Steiner tree is well-structured if, for every Steiner node ℓ, the following two conditions hold:*

1. *ℓ has at least 2 children and*
2. *ℓ has 0 or 2 terminal children.*

Lemma 4.5. *Let ρ be the supremum of $\rho(OPT_{ST}) = \frac{1}{|OPT|}\sum_{\ell \in OPT} c(\ell)$ over Steiner trees $OPT_{ST} = (T \cup OPT, F)$, and ρ_{ws} be the same quantity computed over the subset of well-structured Steiner trees OPT_{ST} of the mentioned type. Then $\rho \le \max\{\hat{H}_1, \rho_{ws}\}$.*

Proof. Recall that by Property 1 in OPT_{ST} each Steiner node ℓ has at most 2 terminal children. Consider any such tree where some Steiner node ℓ' has precisely one terminal child t. Consider the tree OPT'_{ST} which is obtained from OPT_{ST} by appending to ℓ' a second terminal child t'. Observe that the value of $c(\ell)$ does not decrease for any ℓ, and it increases for $\ell = \ell'$. Thus $\rho(OPT'_{ST}) > \rho(OPT_{ST})$. Hence ρ is equal to the supremum of $\rho(OPT_{ST})$ over the subfamily of trees that satisfies (2) in Definition 4.4.

Now consider any tree OPT_{ST} that satisfies (2), and let $oc(OPT_{ST})$ be the number of its Steiner nodes with precisely one child. We prove by induction on $oc(OPT_{ST})$ that $\rho(OPT_{ST}) \le \max\{\hat{H}_1, \rho_{ws}\}$. The claim is trivially true for $oc(OPT_{ST}) = 0$ since in this case OPT_{ST} is well-structured. Assume the claim is true up to $q-1 \ge 0$, and consider $OPT_{ST} = (T \cup OPT, F)$ with $oc(OPT_{ST}) = q$. Let ℓ' be any Steiner node with precisely one child ℓ''. Observe that ℓ'' has to be a Steiner node as well by (2), and that $c(\ell') \le \hat{H}_1$ by Lemma 4.3. Consider the tree $OPT'_{ST} = (T \cup OPT', F')$ obtained by contracting the edge (ℓ', ℓ''). We observe that OPT'_{ST} satisfies (2), $oc(OPT'_{ST}) = q-1$ and $|OPT'| = |OPT| - 1$. Note also that for any Steiner node ℓ different from ℓ' and ℓ'' the value of $c(\ell)$ does not change, while for the new node $\tilde{\ell}$ resulting from the contraction one has $c(\tilde{\ell}) = c(\ell'')$. We can conclude that

$$\frac{1}{|OPT|}\sum_{\ell\in OPT} c(\ell)$$

$$\leq \frac{1}{|OPT|}(\hat{H}_1 + \sum_{\ell\in OPT\setminus\{\ell'\}} c(\ell))$$

$$= \frac{1}{|OPT|}(\hat{H}_1 + \sum_{\ell\in OPT'} c(\ell)) \leq \max\{\hat{H}_1, \frac{1}{|OPT'|}\sum_{\ell\in OPT'} c(\ell)\}$$

$$\leq \max\{\hat{H}_1, \rho_{ws}\},$$

where in the last inequality we used the inductive hypothesis. ∎

We next show an upper bound on ρ_{ws} which is strictly greater than $\hat{H}_1$. It then follows from Lemma 4.5 that the same upper bound holds on ρ. For this goal, we next assume that OPT_{ST} is well-structured.

The upper bound on $c(\ell)$ from Lemma 4.3 is not sufficient to achieve a good approximation factor. In order to achieve a tighter bound, we consider the following classification of the Steiner nodes (see also Figure 4.1).

Definition 4.6. *A Steiner node ℓ' is a* good father *if it has at least one terminal child (hence precisely 2 such children by the above assumptions and Property 1), and a* bad father *otherwise. Each Steiner child ℓ of a good father ℓ' is* good*, and all other Steiner nodes are* bad*. Let OPT_{gf}, OPT_{bf}, OPT_g and OPT_b denote the sets of good fathers, bad fathers, good nodes and bad nodes, resp.*

Notice that the above classification is not affected by the random choices in the marking scheme. For good nodes, the analysis of the cost can be refined as follows.

Lemma 4.7. *For any $\ell \in OPT_g$, $c(\ell) \leq H_{d(\ell)}$.*

Proof. Suppose ℓ has a parent ℓ', which is a good father by definition. This implies that the edge (ℓ', ℓ) is deterministically unmarked, hence $w(m(\ell)) = d(\ell)$ deterministically. If ℓ has no parent (i.e., it is the root r), then $w(m(\ell)) = d(\ell) - 1$. The claim follows. ∎

Putting everything together, we obtain the following.

Lemma 4.8. *$apx \leq 2\varepsilon + \frac{1+\varepsilon}{|OPT|}\sum_{\ell\in OPT} c'(\ell)$ where*

$$c'(\ell) = \begin{cases} H_{d(\ell)} & \text{if } \ell \in OPT_g; \\ \hat{H}_{d(\ell)} & \text{if } \ell \in OPT_b. \end{cases}$$

Proof. It follows from Lemma 4.1, by replacing $c(\ell)$ as in Lemma 4.2 with the upper bounds given by Lemmas 4.3 and 4.7. ∎

We rewrite the upper bound from Lemma 4.8 as follows. Let $p \in [0, \hat{H}_2 - H_2]$ be a parameter to be fixed later. Intuitively, each good Steiner node $\ell \in OPT_g$ pays a *present* p to its (good) father $\ell' \in OPT_{gf}$ to thank ℓ' for making itself good. This increases the cost of ℓ by p. Symmetrically, each good father $\ell' \in OPT_{gf}$ collects presents from its (good) Steiner children and uses them to lower its own cost. Clearly by definition the total modification of the cost is zero. Let us call $c''(\ell)$ the modified costs. Then one obtains the following equality:

$$\frac{1}{|OPT|}\sum_{\ell\in OPT} c'(\ell) = \frac{1}{|OPT|}\sum_{\ell\in OPT} c''(\ell) \tag{4.1}$$

where

$$c''(\ell) = \begin{cases} H_{d(\ell)} + p - s(\ell)p & \text{if } \ell \in OPT_g \cap OPT_{gf}; \\ H_{d(\ell)} + p & \text{if } \ell \in OPT_g \cap OPT_{bf}; \\ \hat{H}_{d(\ell)} - s(\ell)p & \text{if } \ell \in OPT_b \cap OPT_{gf}; \\ \hat{H}_{d(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{bf}. \end{cases}$$

In order to upper bound (4.1), we partition OPT into groups of nodes as follows (see also Figure 4.1).

Definition 4.9. *A Steiner node ℓ is* leaf-Steiner *if it has no Steiner children (i.e., $d(\ell) = t(\ell) = 2$) and* internal-Steiner *otherwise (i.e., $s(\ell) > 0$). We let OPT_{lf} and OPT_{in} be the set of leaf-Steiner and internal-Steiner nodes, resp.*

We associate to each $\ell \in OPT_{in}$ a distinct subset $OPT_{lf}(\ell)$ of precisely $s(\ell) - 1$ leaf-Steiner nodes, and let $g(\ell) = \{\ell\} \cup OPT_{lf}(\ell)$ be the *group* of ℓ. The mapping is constructed iteratively in a bottom-up fashion as follows. Initially all Steiner nodes are unprocessed. We maintain the invariant that the subtree rooted at an unprocessed leaf-Steiner node or at a processed node with unprocessed parent contains precisely one unprocessed leaf-Steiner node. Clearly the invariant holds at the beginning of the process. We consider any unprocessed internal-Steiner node ℓ whose Steiner descendants are either processed or leaf-Steiner nodes. By the invariant, each subtree rooted at a Steiner child of ℓ (which is either an unprocessed leaf-Steiner node or a processed internal-Steiner node) contains one unprocessed leaf-Steiner node. Among this set of $s(\ell)$ unprocessed leaf-Steiner nodes, we select arbitrarily a set $OPT_{lf}(\ell)$ of size $s(\ell) - 1$ and set $g(\ell) = \{\ell\} \cup OPT_{lf}(\ell)$. All nodes in $g(\ell)$ are marked as processed. Observe that the subtree rooted at ℓ still contains an unprocessed leaf-Steiner node, hence the invariant is preserved in the following steps. At the end of the process (i.e., after processing the root r) there will be precisely one leaf-Steiner node ℓ^* which is still unprocessed, which forms a special group $g(\ell^*) = \{\ell^*\}$ on its own. Notice that the groups define a partition of OPT. In particular, $OPT = \{\ell^*\} \cup \bigcup_{\ell \in OPT_{in}} g(\ell)$. Notice also that $|g(\ell)| = s(\ell)$ for all $\ell \in OPT_{in}$ (while $|g(\ell^*)| = 1$).

Let $a(\ell)$ be the average value of $c''(\cdot)$ over the elements of $g(\ell)$. Then obviously the maximum value of $a(\ell)$ over the groups upper bounds the average value of $c''(\cdot)$:

$$\frac{1}{|OPT|} \sum_{\ell \in OPT} c''(\ell) \leq \max_{\ell \in OPT_{in} \cup \{\ell^*\}} \{a(\ell)\}. \tag{4.2}$$

For $\ell = \ell^*$ one has that $a(\ell^*) = c''(\ell^*) = \hat{H}_2$ if ℓ^* is bad, and $a(\ell^*) = c''(\ell^*) = H_2 + p \leq \hat{H}_2$ otherwise. For the other groups $g(\ell)$, there is always a subset of $s(\ell) - 1$ leaves whose contribution to the cost is at most $\hat{H}_2$ each by the same argument as above. Furthermore, we have to add the cost $c''(\ell)$. We can conclude that:

$$a(\ell) \leq \begin{cases} a_1(s(\ell)) := \frac{H_{s(\ell)+2} + p - s(\ell)p + (s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_g \cap OPT_{gf}; \\ a_2(s(\ell)) := \frac{H_{s(\ell)} + p + (s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_g \cap OPT_{bf}; \\ a_3(s(\ell)) := \frac{\hat{H}_{s(\ell)+2} - s(\ell)p + (s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{gf}; \\ a_4(s(\ell)) := \frac{\hat{H}_{s(\ell)} + (s(\ell)-1)\hat{H}_2}{s(\ell)} & \text{if } \ell \in OPT_b \cap OPT_{bf}; \\ \hat{H}_2 & \text{if } \ell = \ell^*. \end{cases}$$

In the first and third case above we used the fact that $d(\ell) = s(\ell) + 2$ (ℓ is a good father, hence has 2 terminal children), while in the second and fourth case the fact that $d(\ell) = s(\ell)$ (ℓ is a bad father, hence has no terminal child).

We are now ready to prove the main result of this paper.

Proof of Theorem 1.1. Consider the above algorithm. Combining Lemma 4.8 with (4.1) and (4.2) one gets

$$apx \leq 2\varepsilon + (1+\varepsilon) \max_{i \geq 1} \{\hat{H}_2, a_1(i), a_2(i), a_3(i), a_4(i)\}. \tag{4.3}$$

Notice that the above approximation factor is a function of the parameter $p \in [0, \hat{H}_2 - H_2]$ which still needs to be fixed. In order to choose a convenient p, we need the following technical result (proof in Appendix A).

Claim 1. *For any $p \in [0, \hat{H}_2 - H_2]$, the maximum of $a_1(i)$, $a_2(i)$, $a_3(i)$, and $a_4(i)$ is achieved for i at most 6, 8, 6 and 8, resp.*

From (4.3) and Claim 1, for any $p \in [0, \hat{H}_2 - H_2]$, one has

$$\begin{aligned} apx \leq 2\varepsilon + (1+\varepsilon) \max\{\hat{H}_2, \max_{1 \leq i \leq 6}\{a_1(i)\}, \max_{1 \leq i \leq 8}\{a_2(i)\}, \\ \max_{1 \leq i \leq 6}\{a_3(i)\}, \max_{1 \leq i \leq 8}\{a_4(i)\}\}. \end{aligned} \tag{4.4}$$

Numerically the minimum of the right-hand side of (4.4) is achieved for $p \simeq 0.135$, and the two largest values inside the maximum turn out to be $a_2(7)$ and $a_3(1)$. By imposing $\frac{H_7+6\hat{H}_2+p}{7} = a_2(7) = a_3(1) = \hat{H}_3 - p$ one gets $p = \frac{7\hat{H}_3 - H_7 - 6\hat{H}_2}{8}$. For that value of p the value of the maximum is precisely $\frac{H_7+6\hat{H}_2+\hat{H}_3}{8} = 2\ln 4 - \frac{967}{1120}$. The claim follows by scaling ε properly. ∎

Appendix A. Omitted Proofs from Section 4.

Claim 2. $H_{d_1} + \sum_{j=d_1+1}^{\infty} \frac{1}{j \cdot 2^{j-d_1}} = \hat{H}_{d_1}$.

Proof. Note that

$$\hat{H}_{d_1} = \sum_{i=0}^{\infty} \frac{H_{d_1+i}}{2^{i+1}} = \sum_{i=0}^{\infty} \frac{H_{d_1} + \sum_{j=1}^{i} \frac{1}{d_1+j}}{2^{i+1}}$$

$$= H_{d_1} + \sum_{j=1}^{\infty} \frac{\sum_{i=j}^{\infty} \frac{1}{2^{i+1}}}{d_1+j} = H_{d_1} + \sum_{j=1}^{\infty} \frac{1}{(d_1+j)2^j}.$$ ∎

Proof of Lemma 4.3. The claim is trivially true if ℓ is the root since in that case $c(\ell) = H_{d(\ell)-1} < \hat{H}_{d(\ell)}$. So we next assume that ℓ is not the root. For a generic sequence $S = (d_1, \ldots, d_k)$ of positive integers, let us define

$$f(S) = \sum_{j=1}^{k-1} \frac{(d_{j+1}-1) \cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2 \cdot d_3 \ldots d_{j+1}} + \frac{H_{d_1+d_2+\cdots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k}.$$

By Lemma 4.2, in order to prove the claim it is sufficient to show that $f(S) \le \hat{H}_{d_1}$. For an infinite sequence $S' = (d_1, d_2, \ldots)$ of positive integers, we analogously define

$$f(S') = \sum_{j=1}^{\infty} \frac{(d_{j+1}-1) \cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2 \cdot d_3 \ldots d_{j+1}}$$

Given a finite sequence $S = (d_1, \ldots, d_k)$ of the above type, let $\bar{S} = (d_1, \ldots, d_k, 2, 2, \ldots)$ be its infinite extension where we add an infinite sequence of 2 at the end.

Claim 3. $f(S) \le f(\bar{S})$.

Proof. By definition

$$\begin{aligned}
&f(\bar{S}) - f(S) \\
&= \sum_{j=k}^{\infty} \frac{(d_{j+1}-1) \cdot H_{d_1+d_2+\ldots+d_j-j+1}}{d_2 \cdot d_3 \ldots d_{j+1}} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} \\
&\ge \sum_{j=k}^{\infty} \frac{(d_{j+1}-1) \cdot H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_{j+1}} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} \\
&= \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} \sum_{j=1}^{\infty} \frac{1}{2^j} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} \\
&= \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} - \frac{H_{d_1+d_2+\ldots+d_k-k+1}}{d_2 \cdot d_3 \ldots d_k} = 0.
\end{aligned}$$

∎

By Claim 3 it is sufficient to consider infinite sequences of type $\bar{S}$. We can also assume w.l.o.g. that all d_i, $i \ge 2$, in such sequences are at least 2 by the following claim.

Claim 4. *Let $\bar{S} = (d_1, \ldots, d_k, 2, 2, \ldots)$ and assume there exists $d_i = 1$ in the sequence for some $i \ge 2$. Let $\bar{S}_i = (d_1, \ldots, d_{i-1}, d_{i+1}, \ldots, d_k, 2, 2, \ldots)$ be the subsequence where the i-th entry is removed. Then $f(\bar{S}) = f(\bar{S}_i)$.*

Proof. Consider the entries in the sum defining $f(\bar{S})$ and $f(\bar{S}_i)$. The entry $j = i-1$ in $f(\bar{S})$ has value 0. For $j < i-1$, the j-th entries in $f(\bar{S})$ and $f(\bar{S}_i)$ are identical. For $j > i-1$, the j-th entry in $f(\bar{S})$ is equal to the $j-1$-th entry in $f(\bar{S}_i)$. ◻

By the above claims we can focus on infinite sequences $\bar{S} = (d_1, \ldots, d_k, 2, 2, \ldots)$, where $d_i \geq 2$ for $i \geq 2$. Let us prove by induction on $k \geq 1$ that $f(\bar{S}) \leq \hat{H}_{d_1}$. The claim is trivially true for $k = 1$ since in that case $f(\bar{S}) = \hat{H}_{d_1}$. Next consider any $k \geq 2$ and assume the claim is true for all values up to $k-1$. Define $\bar{S}' = (d_1 + d_2 - 1, d_3, \ldots, d_k, 2, 2, \ldots)$. By definition and inductive hypothesis:

$$f(\bar{S}) = H_{d_1}\frac{d_2-1}{d_2} + \frac{f(\bar{S}')}{d_2} \leq H_{d_1}\frac{d_2-1}{d_2} + \frac{\hat{H}_{d_1+d_2-1}}{d_2}.$$

By Claim 2,

$$\begin{aligned}
&H_{d_1}\frac{d_2-1}{d_2} + \frac{\hat{H}_{d_1+d_2-1}}{d_2}\\
&= H_{d_1}\frac{d_2-1}{d_2} + \frac{1}{d_2}\left(H_{d_1+d_2-1} + \sum_{j \geq d_1+d_2} \frac{1}{j \cdot 2^{j-d_1-d_2+1}}\right)\\
&= H_{d_1} + \sum_{j=d_1+1}^{d_1+d_2-1} \frac{1}{j \cdot d_2} + \sum_{j \geq d_1+d_2} \frac{1}{j \cdot d_2 \cdot 2^{j-d_1-d_2+1}}\\
&= H_{d_1} + \sum_{j \geq d_1+1} \frac{\alpha_j}{j},
\end{aligned}$$

where

$$\alpha_j := \begin{cases} \frac{1}{d_2} & \text{for } d_1+1 \leq j \leq d_1+d_2-1;\\ \frac{1}{j \cdot 2^{i-d_1-d_2+1}} & \text{for } j \geq d_1+d_2. \end{cases}$$

We observe the following simple facts about the coefficients α_j.

Claim 5. *One has:*

1. $\sum_{j \geq d_1+1} \alpha_j = 1$.
2. *For every $i > 1$, $\sum_{j \geq d_1+i} \alpha_j \geq \frac{1}{2^{i-1}}$.*

Proof. For (1) one has

$$\sum_{j \geq d_1+1} \alpha_j = \frac{d_2-1}{d_2} + \sum_{j=d_1+d_2}^{\infty} \frac{1}{d_2 \cdot 2^{j-d_1-d_2+1}} = 1 - \frac{1}{d_2} + \frac{1}{d_2}.$$

Let us prove (2). For $i \geq d_2$, one has

$$\sum_{j \geq d_1+i} \alpha_j = \sum_{j=d_1+i}^{\infty} \frac{1}{d_2 \cdot 2^{j-d_1+d_2-1}} = \frac{1}{d_2 \cdot 2^{i-d_2}} \geq \frac{1}{2^{i-1}},$$

where in the inequality we used the fact that $k \leq 2^{k-1}$ for any integer $k \geq 1$.

For $2 \leq i \leq d_2 - 1$, one has:

$$\sum_{j \geq d_1+i} \alpha_j = \frac{d_2-i}{d_2} + \frac{1}{d_2} = \frac{d_2-i+1}{d_2} \geq \frac{1}{i} \geq \frac{1}{2^{i-1}},$$

where in the first inequality above we used the fact that $\frac{k-j+1}{k}$ is a decreasing function of $k \geq j+1$ and $d_2 \geq i+1$, and in the second inequality again the fact that $k \leq 2^{k-1}$ for $k \geq 1$. ◻

Intuitively, the term $A = \sum_{j=d_1+1}^{\infty} \frac{\alpha_j}{j}$ is a convex combination of terms of type $1/j$ under the constraint that the sum of the tail coefficients is large enough. An obvious upper bound on A is obtained by choosing coefficients β_j that respect the constraints on α_j given by Claim 5, and at the same time are as large as possible on the smallest terms of the sum. An easy induction shows that the best choice is $\beta_j = \frac{1}{2^{j-d_1}}$ for all $j \geq d_1 + 1$. Thus we can conclude

$$f(\bar{S}) \leq H_{d_1} + \sum_{j \geq d_1+1} \frac{\alpha_j}{j} \leq H_{d_1} + \sum_{j \geq d_1+1} \frac{\beta_j}{j}$$

$$= H_{d_1} + \sum_{j=d_1+1}^{\infty} \frac{1}{j \cdot 2^{j-d_1}} = \hat{H}_{d_1},$$

where last equality comes from Claim 2.

Summarizing, given a non-root Steiner node ℓ and the associated values $S = (d_1, \ldots, d_{q-1})$, $q \geq 2$, as defined in Lemma 4.2, we have that, for $\bar{S} = (d_1, \ldots, d_{q-1}, 2, 2, \ldots)$,

$$c(\ell) \overset{\text{Lem. 4.2}}{=} f(S) \overset{\text{Claim 3}}{\leq} f(\bar{S}) \leq \hat{H}_{d_1} = \hat{H}_{d(\ell)}.$$

∎

Proof of Claim 1. Consider $a_1(i)$. Excluding a fixed additive term $\hat{H}_2 - p$, the value of this function is $a_1'(i) := \frac{H_{i+2}-x}{i}$, where $x = \hat{H}_2 - p \in (0, \hat{H}_2]$. Taking the discrete derivative

$$a_1'(i+1) - a_1'(i) = \frac{x + \frac{i+1}{i+3} - H_{i+3}}{i(i+1)}$$

one might observe that this is negative for $i \geq 6$ since $x + \frac{i+1}{i+3} \leq \hat{H}_2 + 1 < 2.7726 < H_9 > 2.8289$.

Consider now $a_2(i)$. Excluding a fixed additive term $\hat{H}_2$, the value of this function is $a_2'(i) := \frac{H_i - x}{i}$, where $x = \hat{H}_2 - p \in (0, \hat{H}_2]$. One has

$$a_2'(i+1) - a_2'(i) = \frac{x + 1 - H_{i+1}}{i(i+1)},$$

which is negative for $i \geq 8$ since $x + 1 \leq \hat{H}_2 + 1 < 2.7726 < H_9 > 2.8289$.

Consider next $a_3(i)$. Excluding a fixed additive term $\hat{H}_2 - p$, the value of this function is $a_3'(i) := \frac{\hat{H}_{i+2} - \hat{H}_2}{i}$. One has

$$a_3'(i+1) - a_3'(i) = \frac{\hat{H}_2 - \hat{H}_{i+2}}{i(i+1)} + \sum_{j \geq 1} \frac{1}{2^j (i+1)(i+j+2)}$$

$$\leq \frac{\hat{H}_2 + 1 - \hat{H}_{i+2}}{i(i+1)},$$

which is negative for $i \geq 6$ since $\hat{H}_2 + 1 < 2.7726 < \hat{H}_8 > 2.8194$.

It remains to consider $a_4(i)$. Excluding a fixed additive term $\hat{H}_2$, the value of this function is $a_4'(i) := \frac{H_i - \hat{H}_2}{i}$. One has

$$a_4'(i+1) - a_4'(i)$$

$$= \frac{\hat{H}_2 - \hat{H}_i}{i(i+1)} + \sum_{j \geq 1} \frac{1}{2^j (i+1)(i+j)} \leq \frac{\hat{H}_2 + 1 - \hat{H}_i}{i(i+1)},$$

which is negative for $i \geq 8$ since $\hat{H}_2 + 1 < 2.7726 < \hat{H}_8 > 2.8194$. ∎

Appendix B. Details on the Reduction to Steiner Tree.

DEFINITION B.1. *Let $G = (V, E)$ be a connected graph and let L be a set of extra edges. Let $E_1 \subseteq E$ be an edge-cut of G. We say that L covers the cut E_1 if E_1 is not an edge-cut of $G' = (V, E \cup L)$.*

LEMMA B.2. *Let $G = (V, E)$ be an input cactus of CacAP which consists of exactly one cycle and let A be a feasible solution for G. Then $G_{ST}[A]$ is connected.*

Proof. Assume by contradiction that $G_{ST}[A]$ is not connected. Then A can be partitioned in L_R and L_B, such that for any $l_R \in L_R$ and $l_B \in L_B$, l_R does not cross l_B. We call the links in L_R red links and the links in L_B blue links. We can also partition V in V_R and V_B, such that the endpoints of red links belong to V_R and the endpoints of blue links belongs to V_B. Therefore we call V_B and V_R, blue vertices and red vertices respectively.

Let $V_1, V_2, \ldots, V_{2k}$ be the partition of vertices of the cycle G into maximal consecutive blocks of vertices of the same color, so that $V_1 \cup V_3 \cup \cdots \cup V_{2k-1} = V_R$ and $V_2 \cup V_4 \cup \cdots \cup V_{2k} = V_B$.

We say that a link $\ell = \{u, w\} \in A$ is *nice*, if u and v belong to different blocks V_i and V_j, $i \neq j$. We say that an edge $e = \{u, v\} \in E$ is *colorful* if u is red and v is blue or vice versa. Note that G has precisely $2k$ colorful edges. If there is no nice link in A, then any pair of colorful edges of G is not covered by A, which is a contradiction.

Assume that $\ell = \{u, v\} \in A$ is a nice link, such that the distance between u and v in the cycle G is minimum. Assume w.l.o.g. that $u \in V_1$ and $v \in V_{2x+1}$ (and therefore these are red vertices) and also that the vertices of V_2 are in the shortest path from u to v. Now let e_1 and e_2 be the colorful edges such that exactly one of their endpoints is in V_2. We next show that the edge-cut $\{e_1, e_2\}$ is not covered by A.

Assume that $\{e_1, e_2\}$ is covered, then there should be a link $\ell_1 = (w, z)$ such that $w \in V_2$ and $z \notin V_2$. Then either this link is a nice link that crosses ℓ (which is a contradiction since $\ell \in L_R$ and $\ell_1 \in L_B$) or ℓ_1 is a nice link such that the distance of w and z is less than the distance of u and v (which contradicts the choice of ℓ). ◻

Proof of Lemma 2.1. $\Leftarrow$ Let $A \subseteq L$ be such that $G_{ST}[T \cup A]$ is connected. Assume by contradiction that A is not a feasible CacAP solution. Then there exists a 2-edge cut $\{e_1, e_2\}$, for two edges e_1, e_2 belonging to some cycle C of G, which is not covered by any link in A. Let $G_L = (V_L, E_L)$ and $G_R = (V_R, E_R)$ be the two (node disjoint) connected components of G identified by this cut. Let also $t_L \in V_L$ and $t_R \in V_R$ be any two nodes of degree 2 in G. (Observe that these nodes must exist.) By assumption there exists a (simple) path $P = t_L, \ell_1, \ldots, \ell_q, t_R$ between t_L and t_R in $G_{ST}[T \cup A]$, where all ℓ_i's are link nodes. Since $\{e_1, e_2\}$ is not covered, each such link has both endpoints either in V_L or in V_R. Furthermore, ℓ_1 and ℓ_q have one endpoint in V_L and V_R, resp. Hence there must be two consecutive links ℓ_i and ℓ_{i+1} where ℓ_i has both endpoints in V_L and ℓ_{i+1} both endpoints in V_R. These links cannot be crossing, therefore contradicting the fact that $\{\ell_i, \ell_{i+1}\}$ is an edge of G_{ST}.

$\Rightarrow$ Let $A \subseteq L$ be a feasible CacAP solution. We will show that $G_{ST}[T \cup A]$ is connected. We first observe that, w.l.o.g., we can replace each link ℓ with its projections $proj(\ell)$. Indeed, the feasibility of A is preserved. Furthermore, the number of connected components of $G_{ST}[T \cup A]$ does not change since the links in $proj(\ell)$ induce a path in G_{ST}. With this modification, all links in A have both their endpoints in the same cycle (since projections have this property by definition). Let $C_1, \ldots, C_k$ be the cycles of G. For any cycle C_i of the cactus G let A_i be the set of links in A with both their endpoints in C_i. Lemma B.2 shows that $G_{ST}[A_i]$ is connected. For every pair of cycles C_i and C_j that share a node v, there is a link $\ell_i \in A_i$ and $\ell_j \in A_j$ which are incident to v, thus ℓ_i and ℓ_j cross. We can conclude that $G_{ST}[A]$ is connected. Finally, since A is feasible, there exists at least one link $\ell \in A$ incident to each node t of degree 2 in G, which implies that the edge $\{\ell, t\}$ belongs to E_{ST}. Thus $G_{ST}[T \cup A]$ is also connected. ◻

Acknowledgments. This work is highly in debt to Saket Saurabh. During a visit of the second author to Bergen University a few years ago, Saket mentioned the possibility of using the reduction to Steiner tree to approximate connectivity augmentation problems, possibly with an ad-hoc analysis. The result in this paper follows precisely that high-level path, however fixing the details in the analysis was highly non-trivial. The second author is also grateful to M. S. Ramanujan and L. Vegh for several helpful discussions on this topic.

REFERENCES

[1] D. Adjiashvili, *Beating approximation factor two for weighted tree augmentation with bounded costs*, ACM Trans. Algorithms, 15 (2018), https://doi.org/10.1145/3182395, https://doi.org/10.1145/3182395.

[2] H. Angelidakis, D. Hyatt-Denesik, and L. Sanità, *Node connectivity augmentation via iterative randomized rounding*, CoRR, abs/2108.02041 (2021), https://arxiv.org/abs/2108.02041, https://arxiv.org/abs/2108.02041.

[3] M. Basavaraju, F. V. Fomin, P. A. Golovach, P. Misra, M. S. Ramanujan, and S. Saurabh, *Parameterized algorithms to preserve connectivity*, in Automata, Languages, and Programming - 41st International Collo-

quium, ICALP 2014, July 8-11, 2014, Proceedings, Part I, Copenhagen, Denmark, 2014, Springer, pp. 800–811, https://doi.org/10.1007/978-3-662-43948-7_66, https://doi.org/10.1007/978-3-662-43948-7_66.

[4] J. Byrka, F. Grandoni, and A. Jabal Ameli, *Breaching the 2-approximation barrier for connectivity augmentation: a reduction to Steiner tree*, in Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020, K. Makarychev, Y. Makarychev, M. Tulsiani, G. Kamath, and J. Chuzhoy, eds., ACM, 2020, pp. 815–825.

[5] J. Byrka, F. Grandoni, T. Rothvoss, and L. Sanità, *Steiner tree approximation via iterative randomized rounding*, J. ACM, 60 (2013), pp. 6:1–6:33.

[6] F. Cecchetto, V. Traub, and R. Zenklusen, *Bridging the gap between tree and connectivity augmentation: Unified and stronger approaches*, STOC 2021, (2021), https://arxiv.org/abs/2012.00086.

[7] J. Cheriyan and Z. Gao, *Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP*, Algorithmica, 80 (2018), pp. 530–559, https://doi.org/10.1007/s00453-016-0270-4, https://doi.org/10.1007/s00453-016-0270-4.

[8] J. Cheriyan and Z. Gao, *Approximating (unweighted) tree augmentation via lift-and-project, part II*, Algorithmica, 80 (2018), pp. 608–651, https://doi.org/10.1007/s00453-017-0275-7, https://doi.org/10.1007/s00453-017-0275-7.

[9] J. Cheriyan, T. Jordán, and R. Ravi, *On 2-coverings and 2-packings of laminar families*, in Algorithms - ESA '99, 7th Annual European Symposium, July 16-18, 1999, Proceedings, Prague, Czech Republic, 1999, Springer, pp. 510–520, https://doi.org/10.1007/3-540-48481-7_44, https://doi.org/10.1007/3-540-48481-7_44.

[10] J. Cheriyan and R. Thurimella, *Approximating minimum-size k-connected spanning subgraphs via matching*, SIAM J. Comput., 30 (2000), pp. 528–560, https://doi.org/10.1137/S009753979833920X, https://doi.org/10.1137/S009753979833920X.

[11] N. Cohen and Z. Nutov, *A (1+ln2)-approximation algorithm for minimum-cost 2-edge-connectivity augmentation of trees with constant radius*, Theor. Comput. Sci., 489-490 (2013), pp. 67–74, https://doi.org/10.1016/j.tcs.2013.04.004, https://doi.org/10.1016/j.tcs.2013.04.004.

[12] E. A. Dinits, A. V. Karzanov, and M. V. Lomonosov, *On the structure of a family of minimal weighted cuts in a graph*, Studies in Discrete Optimization, (1976), pp. 290–306.

[13] 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 (2009), pp. 21:1–21:17.

[14] S. Fiorini, M. Gross, J. Könemann, and L. Sanità, *Approximating weighted tree augmentation via chvátal-gomory cuts*, in Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, January 7-10, 2018, A. Czumaj, ed., New Orleans, LA, USA, 2018, SIAM, pp. 817–831, https://doi.org/10.1137/1.9781611975031.53, https://doi.org/10.1137/1.9781611975031.53.

[15] G. N. Frederickson and J. JáJá, *Approximation algorithms for several graph augmentation problems*, SIAM Journal on Computing, 10 (1981), pp. 270–283.

[16] H. N. Gabow and S. Gallagher, *Iterated rounding algorithms for the smallest* k*-edge connected spanning subgraph*, SIAM J. Comput., 41 (2012), pp. 61–103, https://doi.org/10.1137/080732572, https://doi.org/10.1137/080732572.

[17] M. X. Goemans, A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson, *Improved approximation algorithms for network design problems*, in Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia, USA, 1994, ACM/SIAM, pp. 223–232, http://dl.acm.org/citation.cfm?id=314464.314497.

[18] F. Grandoni, A. Jabal Ameli, and V. Traub, *Breaching the 2-approximation barrier for the forest augmentation problem*, in Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2022, June 20-24, 2022.

[19] F. Grandoni, C. Kalaitzis, and R. Zenklusen, *Improved approximation for tree augmentation: saving by rewiring*, in Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, June 25-29, 2018, Los Angeles, CA, USA, 2018, ACM, pp. 632–645, https://doi.org/10.1145/3188745.3188898, https://doi.org/10.1145/3188745.3188898.

[20] W. GÃĄlvez, F. Grandoni, A. Jabal Ameli, and K. Sornat, *On the cycle augmentation problem: Hardness and approximation algorithms*, Theory of Computing Systems, 65 (2021), pp. 1–24, https://doi.org/10.1007/s00224-020-10025-6.

[21] C. Hunkenschröder, S. S. Vempala, and A. Vetta, *A 4/3-approximation algorithm for the minimum 2-edge connected subgraph problem*, ACM Trans. Algorithms, 15 (2019), pp. 55:1–55:28, https://doi.org/10.1145/3341599, https://doi.org/10.1145/3341599.

[22] J. Iglesias and R. Ravi, *Coloring down: 3/2-approximation for special cases of the weighted tree augmentation problem*, CoRR, abs/1707.05240 (2017), http://arxiv.org/abs/1707.05240, https://arxiv.org/abs/1707.05240.

[23] K. Jain, *A factor 2 approximation algorithm for the generalized steiner network problem*, Combinatorica, 21 (2001), pp. 39–60.

[24] S. Khuller and R. Thurimella, *Approximation algorithms for graph augmentation*, Journal of Algorithms, 14 (1993), pp. 214–225.

[25] P. Klein and R. Ravi, *A nearly best-possible approximation algorithm for node-weighted steiner trees*, Journal of Algorithms, 19 (1995), pp. 104–115.

[26] G. Kortsarz and Z. Nutov, *Lp-relaxations for tree augmentation*, in Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France, 2016, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 13:1–13:16, https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.13, https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2016.13.

[27] 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 (TALG), 12 (2016), pp. 23:1–23:20.

[28] D. Marx and L. A. Végh, *Fixed-parameter algorithms for minimum-cost edge-connectivity augmentation*, ACM Trans. Algorithms, 11 (2015), pp. 27:1–27:24, https://doi.org/10.1145/2700210, https://doi.org/10.1145/2700210.

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

[30] Z. Nutov, *2-node-connectivity network design*, in Approximation and Online Algorithms - 18th International Workshop, WAOA 2020, Virtual Event, September 9-10, 2020, Revised Selected Papers, C. Kaklamanis and A. Levin, eds., vol. 12806 of Lecture Notes in Computer Science, Springer, 2020, pp. 220–235, https://doi.org/10.1007/978-3-030-80879-2_15, https://doi.org/10.1007/978-3-030-80879-2_15.

[31] Z. Nutov, *Approximation algorithms for connectivity augmentation problems*, CoRR, abs/2009.13257 (2020), https://arxiv.org/abs/2009.13257, https://arxiv.org/abs/2009.13257.

[32] Z. Nutov, *On the tree augmentation problem*, Algorithmica, 83 (2021), pp. 553–575, https://doi.org/10.1007/s00453-020-00765-9, https://doi.org/10.1007/s00453-020-00765-9.

[33] G. Robins and A. Zelikovsky, *Tighter bounds for graph Steiner tree approximation*, SIAM Journal on Discrete Mathematics, 19 (2005), pp. 122–134.

[34] A. Sebö and J. Vygen, *Shorter tours by nicer ears: 7/5-approximation for graphic tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs*, Combinatorica, 34(5) (2014), pp. 597–629, http://arxiv.org/abs/1201.1870, https://arxiv.org/abs/1201.1870.

[35] V. Traub and R. Zenklusen, *A better-than-2 approximation for weighted tree augmentation*, CoRR, abs/2104.07114 (2021), https://arxiv.org/abs/2104.07114, https://arxiv.org/abs/2104.07114.

[36] V. Traub and R. Zenklusen, *Local Search for Weighted Tree Augmentation and Steiner Tree*, 2022, pp. 3253–3272, https://doi.org/10.1137/1.9781611977073.128, https://epubs.siam.org/doi/abs/10.1137/1.9781611977073.128, https://arxiv.org/abs/https://epubs.siam.org/doi/pdf/10.1137/1.9781611977073.128.