Ghost Value Augmentation for k-Edge-Connectivity

Ghost Value Augmentation for k-Edge-Connectivity

D. Ellis Hershkowitz
Brown University
Providence, USA
delhersh@brown.edu

Nathan Klein
Institute for Advanced Study
Princeton, USA
nklein@ias.edu

Rico Zenklusen
ETH Zurich
Zurich, Switzerland
ricoz@ethz.ch

ABSTRACT

We give a poly-time algorithm for the k-edge-connected spanning subgraph (k-ECSS) problem that returns a solution of cost no greater than the cheapest $(k+10)$-ECSS on the same graph. Our approach enhances the iterative relaxation framework with a new ingredient, which we call ghost values, that allows for high sparsity in intermediate problems.

Our guarantees improve upon the best-known approximation factor of 2 for k-ECSS whenever the optimal value of $(k+10)$-ECSS is close to that of k-ECSS. This is a property that holds for the closely related problem k-edge-connected spanning multi-subgraph (k-ECSM), which is identical to k-ECSS except edges can be selected multiple times at the same cost. As a consequence, we obtain a $(1+O(1/k))$-approximation algorithm for k-ECSM, which resolves a conjecture of Pritchard and improves upon a recent $(1+O(1/\sqrt{k}))$-approximation algorithm of Karlin, Klein, Oveis Gharan, and Zhang. Moreover, we present a matching lower bound for k-ECSM, showing that our approximation ratio is tight up to the constant factor in $O(1/k)$, unless P = NP.

CCS CONCEPTS

• **Theory of computation → Routing and network design problems.**

KEYWORDS

Approximation Algorithms, Edge Connectivity, Network Design, Iterative Rounding

ACM Reference Format:
D. Ellis Hershkowitz, Nathan Klein, and Rico Zenklusen. 2024. Ghost Value Augmentation for k-Edge-Connectivity. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3618260.3649715

1 INTRODUCTION

Computing k-edge-connected subgraphs of minimum cost is a fundamental problem in combinatorial optimization. For $k=1$, this is the famous minimum spanning tree (MST) problem. For $k \geq 2$, this problem is known as the k-edge-connected spanning subgraph (k-ECSS) problem. Formally, in k-ECSS we are given a multi-graph $G=(V,E)$ with an edge cost function $c: E \to \mathbb{R}_{\geq 0}$ and our goal is to find a set of edges $F \subseteq E$ where (V,F) is k-edge-connected while minimizing $c(F) := \sum_{e \in F} c(e)$.

For $k \geq 2$, k-ECSS is APX-Hard. More precisely, there exists an $\epsilon > 0$ such that, unless P = NP, there is no poly-time $(1+\epsilon)$-approximate algorithm for k-ECSS for any $k \geq 2$ [50]. Thus, somewhat surprisingly, the problem does not seem to get easier as k grows. The problem and its special cases have been extensively studied [1, 4, 8, 9, 12, 15–18, 24, 25, 28, 29, 32, 40, 47, 56–58], but for any $k \geq 2$ the best poly-time approximation algorithm is a four-decades-old 2-approximation due to [20, 21].[1] Designing a better-than-2 approximation algorithm for any $k \geq 2$ is a major open problem. Connectivity problems have also been considered for the notion of vertex connectivity (see, e.g., [2, 3, 14, 31, 46, 48]).

In this work, we show that one can achieve a close-to-1 approximation for k-ECSS whenever the cost of the optimal $(k+10)$-edge-connected solution is close to the cost of the optimal k-edge-connected solution. Specifically, we give a *resource augmentation* result for k-ECSS whereby we compare the quality of our algorithm's output to an adversary that has fewer resources (namely, the resource of cost budget):

We show that one can poly-time compute a k-edge-connected graph with cost no greater than that of the optimal solution which $(k+10)$-edge-connects the graph.

Similar resource augmentation results are known for other well-studied network design problems such as in the minimum cost bounded degree spanning tree problem [10, 26, 41, 42, 51]; here, the state-of-the-art is an algorithm of [54], which shows that one can find a spanning tree of maximum degree d that has cost no greater than the spanning tree of maximum degree $d-1$ in poly-time. Likewise, resource augmentation is often considered in online and scheduling algorithms [11, 33, 49, 52, 55]. However, to our knowledge, no similar results are known for k-connectivity-type problems, and standard techniques [45] would require augmenting the budget by k rather than $O(1)$.

Our result is cost-competitive with the optimal LP solution, defined as follows. For ease of presentation, we fix an arbitrary vertex $r \in V$, called the *root*, and represent each cut $(S, V \setminus S)$ by the side that does not contain r. Letting $\delta(S)$ be the set of edges crossing a cut $S \subseteq V$ and $x(F) := \sum_{e \in F} x_e$ for $F \subseteq E$, we have the following LP for k-ECSS.

$$\begin{array}{rccl} \min & c^\top x & & \\ & x(\delta(S)) & \geq & k \qquad \forall S \subseteq V \setminus \{r\}, S \neq \emptyset \\ & x & \in & [0,1]^E \end{array} \qquad (k\text{−ECSS LP})$$

[1]It also follows from the iterative rounding of [34].

STOC '24, June 24–28, 2024, Vancouver, BC, Canada
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649715

Denoting by $\mathrm{LPOPT}_{k-\mathrm{ECSS}}$ the cost of an optimal solution to k−ECSS LP and by $\mathrm{OPT}_{k-\mathrm{ECSS}}$ the cost of the optimal k-ECSS solution, our main result for k-ECSS is as follows.

Theorem 1.1. *There is a poly-time algorithm that, for any k-ECSS instance with $k \in \mathbb{Z}_{\geq 1}$, returns a k-ECSS solution of cost at most* $\mathrm{LPOPT}_{(k+10)-\mathrm{ECSS}} \leq \mathrm{OPT}_{(k+10)-\mathrm{ECSS}}$.

Equivalently, we show that one can find in poly-time a $(k-10)$-edge-connected subgraph of cost at most $\mathrm{OPT}_{k-\mathrm{ECSS}}$. Thus, in some sense, our result demonstrates that achieving the last small constant amount of connectivity is the NP-hard part of k-ECSS.

One might reasonably wonder if it is often the case that the optimal k-edge-connected solution has cost close to the optimal $(k+10)$-edge-connected solution. In fact, a well-studied problem closely related to k-ECSS called the k-edge-connected spanning *multi*-subgraph (k-ECSM) problem satisfies exactly this property. k-ECSM is the same as k-ECSS except our solution F is a multiset that can include each edge of E as many times as we want (where we pay for every copy of an edge). The canonical LP for k-ECSM is the same as that of k-ECSS but has no upper bound on how many times we choose an edge.

$$\begin{array}{lrcll} \min & c^\top x & & & \\ & x(\delta(S)) & \geq & k & \forall S \subseteq V \setminus \{r\}, S \neq \emptyset \\ & x & \in & \mathbb{R}^E_{\geq 0}. & \end{array} \tag{k−ECSM LP}$$

We notate by $\mathrm{LPOPT}_{k-\mathrm{ECSM}}$ the cost of an optimal solution to k−ECSM LP and by $\mathrm{OPT}_{k-\mathrm{ECSM}}$ the cost of an optimal k-ECSM solution. It is easy to see that scaling the optimal k-ECSM LP solution by $(k+10)/k$ results in a $(k+10)$-ECSM LP solution, so we have the claimed relation between the costs of the optimal k-edge-connected and $(k+10)$-edge-connected LP solutions:

$$\mathrm{LPOPT}_{(k+10)-\mathrm{ECSM}} \leq \left(1+\frac{10}{k}\right)\cdot \mathrm{LPOPT}_{k-\mathrm{ECSM}}. \tag{1}$$

Unlike k-ECSS, it is known that, as $k \to \infty$, approximation factors arbitrarily close to 1 are possible; however, prior to this work, the correct asymptotic dependence on k was not fully understood. [24] showed that *if the graph G is unweighted* (i.e., every edge has cost 1), then k-ECSM (and k-ECSS) admits $(1+{}^2/_k)$-approximation algorithms. This led Pritchard to pose the following conjecture for k-ECSM (with general weights).

Conjecture 1.2 ([50]). *k-ECSM admits a poly-time $(1+O({}^1/_k))$-approximation algorithm.*

Very recently, [38] made significant progress on this conjecture, showing there is a poly-time $(1+{}^{5.06}/_{\sqrt{k}})$-approximation for k-ECSM. However, their approach provably does not give better than a $(1+O({}^1/_{\sqrt{k}}))$-approximation.

The result of [38] added to a considerable body of work on k-ECSM. The first notable algorithm for k-ECSM is due to [20, 21], who gave a ${}^3/_2$-approximation algorithm for even k and a $({}^3/_2 + O({}^1/_k))$ for odd k. This algorithm essentially follows by a reduction to the well-known Christofides-Serdyukov algorithm for the traveling salesperson problem (TSP). Despite many subsequent works on k-ECSM [5–7, 13, 22, 23, 35, 39, 43, 50, 53], this algorithm of [20, 21] remained the best approximation algorithm for nearly four decades, except when the underlying graph is unweighted, $k \gg \log n$, or $k = 2$. Recently, this was very slightly improved to a $({}^3/_2 - \epsilon)$-approximation for even k where $\epsilon = 10^{-36}$ [36, 37]. Thus, for large k, the algorithm of [38] considerably improved on all known prior work for k-ECSM and gave the first algorithm with approximation ratio tending to 1 as $k \to \infty$ independent of n.

As an immediate consequence of Theorem 1.1 and Equation (1), we are able to settle Pritchard's conjecture with a poly-time $(1+O({}^1/_k))$-approximation for k-ECSM.

Theorem 1.3. *There is a poly-time algorithm for k-ECSM that, for any k-ECSM instance with $k \in \mathbb{Z}_{\geq 1}$, returns a k-ECSM solution of cost at most* $(1+\frac{10}{k})\cdot \mathrm{LPOPT}_{k-\mathrm{ECSM}} \leq (1+\frac{10}{k})\cdot \mathrm{OPT}_{k-\mathrm{ECSM}}$.

Proof of Theorem 1.3 assuming k is given in unary. We give an algorithm which runs in poly-time assuming k is represented in unary in the input. See this proof in Section 2 for a more general proof in which we assume k is represented in binary.

Suppose our instance of ECSM is on graph $G = (V, E)$ with costs c, and optimal k and $k+10$ LP costs $\mathrm{LPOPT}_{k-\mathrm{ECSM}}$ and $\mathrm{LPOPT}_{(k+10)-\mathrm{ECSM}}$, respectively. Then, consider the $(k+10)$-ECSS instance on $G' = (V, E')$ with costs c and optimal LP cost $\mathrm{LPOPT}_{(k+10)-\mathrm{ECSS}}$, where E' is E with $k+10$ copies of each $e \in E$.

Apply Theorem 1.1 to our $(k+10)$-ECSS instance to compute a multiset $F \subseteq E'$ which k-edge-connects V. By construction, F is a feasible solution to our k-ECSM instance, when interpreting the selection of parallel edges by one multi-selection of the edge in E they correspond to (and any k-ECSM solution on G can be interpreted as a solution to k-ECSS on G'), and by Equation (1) it has cost at most

$$\begin{aligned} \mathrm{LPOPT}_{(k+10)-\mathrm{ECSS}} &\leq \mathrm{LPOPT}_{(k+10)-\mathrm{ECSM}} \\ &\leq \left(1+\frac{10}{k}\right)\cdot \mathrm{LPOPT}_{k-\mathrm{ECSM}}. \end{aligned}$$ □

Our algorithm improves upon the $1+{}^{5.06}/_{\sqrt{k}}$ algorithm of [38] for all relevant k (i.e., those in which that algorithm was better than ${}^3/_2-\epsilon$).

As previously noted [27, 50], a poly-time algorithm for k-ECSM whose solutions are approximate with respect to $\mathrm{LPOPT}_{k-\mathrm{ECSM}}$ gives an approximation algorithm with the same approximation bounds for subset (a.k.a. Steiner) k-ECSM. Here, we only need to k-edge-connect a subset of the nodes. Thus, as a consequence of Theorem 1.3, we get a $1+{}^{10}/_k$ for this more general subset k-ECSM problem. See the full version for a formal statement of this problem and the result.

Complementing this solution to Pritchard's conjecture, we show that the dependence on k in our algorithm is essentially optimal. Specifically, we show that Theorem 1.3 is almost tight by showing $(1+\Omega({}^1/_k))$-hardness-of-approximation for k-ECSM. (The tightness is up to the constant in front of the term ${}^1/_k$.)

Theorem 1.4. *There exists a constant $\epsilon > 0$ such that there does not exist a poly-time algorithm which, given an instance of (unweighted) k-ECSM where k is part of the input, always returns a $\left(1+\frac{\epsilon}{k}\right)$-approximate solution, unless* P = NP.

Such a hardness result was also identified as an open question by [50].

We first give a proof sketch in Section 1.1. In Section 2, we then describe our main technical rounding theorem and how it implies our results on k-ECSS and k-ECSM. In particular, we show it allows us to compute a k-ECSS solution with cost at most that of the optimal $(k+10)$-ECSS solution, and we also show it implies a $1+O(1/k)$-approximate k-ECSM algorithm—settling Pritchard's k-ECSM conjecture. Later, in Section 5, we further discuss our $1+\Omega(1/k)$ hardness of approximation result for k-ECSM, although we defer the proof to the full version. In Section 3, we describe our algorithm and give further intuition on why it gives strong guarantees before formally analyzing its performance in Section 4.

1.1 Iterative Relaxation Barriers and Ghost Value Augmentations to Overcome Them

First note that for the rest of the paper, to slightly simplify notation, we will work with a solution to the k-ECSS LP and round it to a $(k-10)$-edge-connected graph. This is of course equivalent to working with k-edge-connectivity and the $(k+10)$-ECSS LP.

Our approach to showing Theorem 1.1 is to apply iterative LP relaxation methods to the k-ECSS LP and obtain a $(k-10)$-edge-connected graph. Iterative LP methods are a well-studied approach first pioneered by [34]. See [44] for a comprehensive overview. Generally, iterative relaxation algorithms repeatedly compute an optimal solution to a suitable LP and then make progress towards producing a desired output by either finding an edge with LP value 0 that can be deleted, a newly integral edge that can be frozen at its integral value, or some LP constraint which is *nearly* satisfied and can be dropped from future LP recomputations while approximately preserving feasibility.

Applying such an iterative relaxation approach to the k-ECSS LP is naturally suited to proving Theorem 1.1 since we are allowed $O(1)$ slack in connectivity. Specifically, one might hope to argue that if no edge can be frozen then there is some constraint of the k-ECSS LP corresponding to a cut which has at least $k-O(1)$ frozen edges crossing it. Such a constraint can be safely dropped from future recomputations since our ultimate goal allows for slack $O(1)$ in connectivity. However, it is not too hard to see that such a natural approach faces a significant barrier and so a non-standard idea is required.

In what follows, we describe this approach and barrier in more detail and how our key non-standard idea of "ghost value augmentations" allows us to overcome this barrier.

A Standard Iterative Relaxation Approach. A first attempt to prove Theorem 1.1 is as follows, where we replace the constant 10 with an arbitrary constant $c \in \mathbb{Z}_{\geq 1}$. Repeat the following until there are no remaining variables.

(i) Let y be an extreme point solution to k-ECSS LP. For each edge e with $y_e = 0$, delete e from the LP. For each edge e with $y_e = 1$, add the constraint $x_e = 1$ to the LP and call e "frozen." We let F be all edges we have frozen so far.
(ii) Now suppose that whenever we cannot delete or freeze a new edge, there is always a cut $S \subseteq V \setminus \{r\}$ such that $|\delta(S) \cap F| \geq k-c$. We call this the *light cut property.* In this case, we can delete the constraint $x(\delta(S)) \geq k$ from the LP. This is a safe operation since $\delta(S)$ already has at least $k-c$ frozen edges.

Once there are no remaining variables, we have an integral solution and can return the set of frozen edges. Provided one can still efficiently solve the LP even after dropping constraints and so long as the light cut property always holds when we cannot delete or freeze a new edge, standard arguments would demonstrate that the above algorithm always returns an integral $(k-c)$-ECSS solution of cost no more than the cost of $\text{LPOPT}_{k-\text{ECSS}}$.

A Barrier to the Standard Approach. We do not know if the light cut property is true or not, and proving or disproving it is a very interesting open problem. However, there is a major barrier to proving it using known techniques for iterative relaxation, which we detail here.

To prove results like the light cut property, one generally first demonstrates that the set of tight constraints at any stage of the algorithm's execution can be "uncrossed" to obtain a laminar family.[2] In our case, a set $S \subseteq V$ is tight if $y(\delta(S)) = k$ and one wants to show that there is a laminar family $\mathcal{L} \subseteq 2^V$ such that every constraint $x(\delta(S)) \geq k$ where S is tight is spanned by the constraints $\{x(\delta(S)) \geq k \colon S \in \mathcal{L}\}$.[3]

The major barrier to the standard iterative relaxation approach is that it does not appear that uncrossing is possible. One sufficient criteria for uncrossing is if the constraints can be expressed as $x(\delta(S)) \geq f(S)$ where the "requirement function" $f: V \longrightarrow \mathbb{Z}_{\geq 0}$ is skew supermodular [34].

A function $f: V \longrightarrow \mathbb{Z}_{\geq 0}$ is called skew supermodular if for all $S, T \subseteq V$ we have either

$$f(S) + f(T) \leq f(S \cap T) + f(S \cup T), \text{ or}$$
$$f(S) + f(T) \leq f(S \smallsetminus T) + f(T \smallsetminus S).$$

At the beginning of our process, $f(S) = k$ for all $S \subseteq V$, and thus both of these inequalities trivially hold with equality. However, once we drop a constraint S for which we had $|F \cap \delta(S)| \geq k-c$, this property fails to hold. In particular, we may have dropped the constraint for $S \cap T$ and $S \smallsetminus T$ so that $f(S \cap T) = f(S \smallsetminus T) = 0$. Thus, in this situation, the left-hand side of each equation would be $2k$ and the right-hand side k. For example, the situation in Figure 1 could occur, where we dropped cuts with at least $k-c$ frozen edges.

Figure 1 shows that we cannot apply the result of Jain as a black box. However, one can still successfully uncross in this situation. The reason is that the constraints $S \smallsetminus T, T \smallsetminus S, S \cap T, S \cup T$ are all still tight. Therefore, even though the constraints are not present in the LP, one can still replace the constraints S and T with the constraints $S \smallsetminus T$ and $T \smallsetminus S$ (or $S \cap T$ and $S \cup T$).

The true issue arises when the connectivity of some sets drop below k, as in Figure 2. In Figure 2, *none of* $S \smallsetminus T, T \smallsetminus S, S \cap T$, and $S \cup T$ are tight constraints. One could still consider adding them to the family. However, if we did this, we would face two major issues.

(1) The constraints may not be integers, as is the case for all of the sets here. This leads to problems in the next phase of

[2]A family of sets $\mathcal{L}$ is called *laminar* if it does not contain any pair of intersecting sets S, T, which are sets such that $S \cap T \notin \{S, T, \emptyset\}$. Moreover, two sets $S, T \subseteq V$ are *crossing* if all of $S \cap T$, $V \setminus (S \cup T)$, $S \setminus T$, and $T \setminus S$ are non-empty. We typically use these notions for vertex sets that correspond to cuts. Because we assume that cuts do not contain r, the notions of crossing and intersecting sets coincide for cuts.
[3]Here, and throughout this work, we treat x as a variable and y as a fixed solution to our LP.

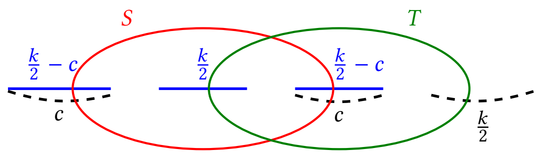


Figure 1: An example where skew supermodularity of the requirement function fails. The solid blue edges represent collections of frozen edges between sets. The dashed edges represent collections of non-frozen edges. Here, $S \setminus T$ and $S \cap T$ have been dropped as they have at least $k-c$ frozen edges. However, $S, T, T \setminus S$, and $S \cup T$ have fewer than $k-c$ frozen edges and thus remain. So, $f(S) = f(T) = k$ but $f(S \setminus T) = f(S \cap T) = 0$.

the iterative relaxation argument which uses integrality of the constraints to argue that if S, T are two tight sets in the family then the symmetric difference $\delta(S) \Delta \delta(T)$ has size at least 2.

(2) Unlike in the case in which S and T are minimum cuts of the graph (in which we can apply standard uncrossing) it is not necessarily the case that

$$\chi^{\delta(S)} + \chi^{\delta(T)} = \chi^{\delta(S \setminus T)} + \chi^{\delta(T \setminus S)},$$

where χ^F for $F \subseteq E$ is the vector in $\{0,1\}^E$ that is 1 at the edges in F and 0 elsewhere. In this example this equality does not hold due to the edge with value a. Similarly, it is not necessarily the case that

$$\chi^{\delta(S)} + \chi^{\delta(T)} = \chi^{\delta(S \cap T)} + \chi^{\delta(S \cup T)}.$$

To see this, one can extend the example by adding a small fractional edge between $S \setminus T$ to $T \setminus S$ and adjusting other edges accordingly (since only S, T are tight, this is not difficult).

However, relations as the two highlighted above are central in classical uncrossing arguments.

Therefore, to prove the light cut property, one would likely have to deal with a fairly complicated family of tight sets.

Overcoming the Barrier with Ghost Values. Instead of working with this uncrossable family, we introduce a relaxation approach we call "ghost value augmentation." We consider the LP solution y together with a ghost vector g that augments y, so that the LP constraints are now of the form $x(\delta(S)) + g(\delta(S)) = (x+g)(\delta(S)) \geq k$. We say such a constraint is tight with respect to solution y if $(y+g)(\delta(S)) = k$. This ghost vector will help us achieve uncrossing of tight sets. However, crucially, it will never be used in the final solution. This ensures that we never increase the cost of the solution compared to the LP.

We will still follow the general framework of iterative relaxation. Given an extreme point solution y, we will delete edges with $y_e = 0$ and freeze edges with $y_e = 1$. And, as before, we will drop constraints corresponding to tight sets S with the property that $|\delta(S) \cap F| \geq k - O(1)$ (where F is the set of frozen edges). However, we will only drop such a set S if:

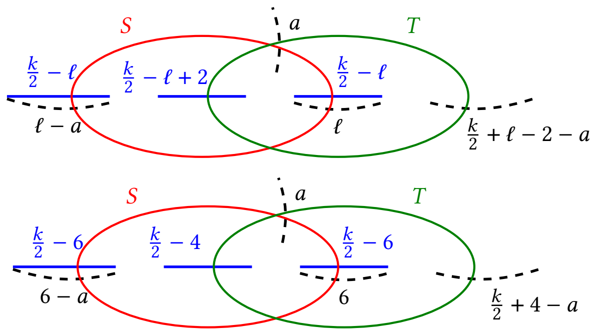


Figure 2: An example of where uncrossing breaks down, where in the top figure we let $\ell = \lceil \frac{c+1}{2} \rceil$ and in the bottom figure we assume $c = 10$ for concreteness. The blue edges represent collections of frozen edges and the dotted edges represent collections of fractional edges. To ensure feasibility we let $0 < a < 1$. In this example, we drop cuts when there are $k - c$ frozen edges (or $k - 10$ in the bottom picture). Then, in both examples we have dropped $S \setminus T$ and $S \cap T$, but we have not yet dropped S and T.

(i) **S is a minimal tight set corresponding to an LP constraint**. We restrict ourselves to tight sets for which there is no $T \subsetneq S$ such that the cut constraint corresponding to T is tight and still in the LP. Such sets S are desirable because they have the additional property that they are either vertices or all edges with both endpoints inside them are frozen.

(ii) **$\delta(S)$ has only $O(1)$ fractional edges**. This allows us to ensure that the cut $\delta(S)$ does not change much over the course of the remainder of the algorithm. In particular, this will let us derive *upper bounds* on the number of edges crossing a dropped set, which in turn helps to lower bound the value of other cuts.

Of course, point (ii) implies that there are $k - O(1)$ frozen edges, so it is sufficient to check (i) and (ii) for all sets. These points together allow us to argue that after dropping such a constraint S with $|S| \geq 2$, it is safe to contract S to a vertex.[4] This is because cuts contained in S will not change by more than $O(1)$ throughout the execution of the entire algorithm, as they only contain edges inside S, which are all frozen by (i), and edges in $\delta(S)$, of which all but $O(1)$ are frozen by (ii). See Figure 3.

Contraction now gives us the crucial property that at every iteration of the algorithm, operating on a graph $G = (V, E)$ resulting from contracting some number of sets, we have $y(\delta(S)) \geq k$ for all $S \subseteq V \setminus \{r\}$ for which $2 \leq |S|$. For all vertices $v \in V$, we only have the weaker guarantee that $y(\delta(v)) \geq k - O(1)$ where we use the notation $\delta(v) = \delta(\{v\})$. Even though the connectivity of the graph

[4]Contracting a vertex set $S \subseteq V$ in $G = (V, E)$ means that we remove S from V and add a single new vertex S. An edge $(u, v) \in E$ before contraction with $u \in S$ and $v \in V \setminus S$ will become an edge between S and v; moreover, edges with both endpoints in S will be removed in the contracted graph. As is common, when having an edge $e = (u, v) \in E$ before contraction that gets transformed into an edge (S, v) after contraction, we still consider this to be the same edge. In particular, any LP value or constraint on e before contraction will be interpreted as a value or constraint, respectively, on the new edge after contraction.

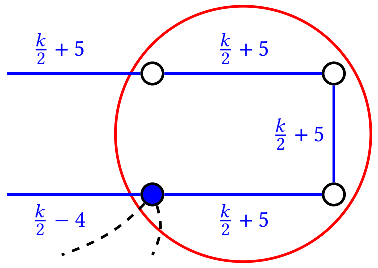


Figure 3: The blue edges are frozen, and the dotted edges are fractional. Consider the red set S with 2 incident fractional edges. Since $\delta(S)$ has only 2 fractional edges, and the edges inside S are all frozen, any set contained in S is already safe, as is the case with the blue vertex.

is not uniform, the fact that the only cuts below k are the vertices allows us to show that either:

(a) There is an edge $e = (u,v)$ with $(y+g)(E(u,v)) \in [\frac{k}{2} - O(1), \frac{k}{2})$ (where $E(u,v)$ is the set of edges between u and v). In this case, we perform a *ghost value augmentation*: we artificially increase the value of $(y+g)(E(u,v))$ to at least $\frac{k}{2}$ by increasing g_e by $O(1)$ for an edge in $E(u,v)$.

(b) Otherwise, the set of tight constraints can be successfully uncrossed. In this case, we argue that there must be a set S that is safe to drop that has no tight children and at most 3 fractional edges. If S is not a singleton, then we additionally contract it.

At first, (a) may look like a strange property to expect for k-ECSS solutions. Indeed, the underlying input graph may not have any multi-edges, so that at the first iteration $y(E(u,v)) \leq 1$ for all vertices u,v. However, as sets are contracted, these structures may begin to emerge as barriers for uncrossing. For example, in Fig. 2, since $S \setminus T$ and $S \cap T$ have been dropped, they are now singletons. This allows us to study $y(E(u,v))$ for $\{u\} = S \setminus T$ and $\{v\} = S \cap T$. And one can see that $y(E(u,v)) = \frac{k}{2} - O(1)$, giving us a candidate for ghost value augmentation.[5] For more intuition about why these structures should appear, one can study the cactus representation of minimum cuts (see [19] for a nice overview of this representation).

In both cases (a) and (b) we make progress: we either drop a constraint that is safe to drop (and possibly contract its corresponding set), or we fix an edge in our current graph to at least $\frac{k}{2}$. We describe this process in more detail in Section 3. In Section 4 we show that one of these two cases must occur. We defer the proof that at the end of the algorithm the frozen edges $(k - O(1))$-connect the graph to the full version.

2 MAIN ROUNDING THEOREM

Our main result will follow immediately from a general rounding theorem.

[5]Note that Fig. 2 is not quite representative of the situations we will arrive at over the course of the algorithm since it was designed to handle the situation in which we drop all constraints with $k - O(1)$ frozen edges and not $O(1)$ fractional edges. However, it is quite similar to situations we study in the following sections. See Fig. 4 for an instance tailored to our algorithm which shows the importance of ghost value augmentation.

Theorem 2.1 (Main Rounding Theorem). *There is a poly-time algorithm that, given $y \in \mathbb{R}^E_{\geq 0}$ where $y(\delta(S)) \geq k$ for all non-empty $S \subsetneq V$, returns a $z \in \mathbb{Z}^E_{\geq 0}$ that is:*

*(1) **Cost-Preserved:** $c^\top z \leq c^\top y$;*
*(2) **Integrally-Rounded:** $z_e \in \{\lfloor y_e \rfloor, \lceil y_e \rceil\}$;*
*(3) **Highly-Connected:** $z(\delta(S)) \geq k - 9 - \mathbb{1}[k \text{ odd}]$ for all non-empty $S \subsetneq V$.*[6]

We now observe that both our main result for k-ECSS (Theorem 1.1) and our main result for k-ECSM in polynomial time (Theorem 1.3) are immediate from this rounding theorem.

Theorem 1.1. *There is a poly-time algorithm that, for any k-ECSS instance with $k \in \mathbb{Z}_{\geq 1}$, returns a k-ECSS solution of cost at most* $\text{LPOPT}_{(k+10)-\text{ECSS}} \leq \text{OPT}_{(k+10)-\text{ECSS}}$.

Proof. The result is immediate from applying Theorem 2.1 to any optimal solution y to $(k+10)$-ECSS LP. y is poly-time computable by the Ellipsoid Method [30] and standard separation oracles. ◻

We note that in fact the same proof shows we can get the same guarantee for a slightly more general problem than k-ECSS. In particular, edges can be given arbitrary lower and upper bounds, and we can still obtain a solution in polynomial time (this includes the case in which k and possibly the lower and upper bounds are exponentially large compared to the input size).

Theorem 1.3. *There is a poly-time algorithm for k-ECSM that, for any k-ECSM instance with $k \in \mathbb{Z}_{\geq 1}$, returns a k-ECSM solution of cost at most* $(1 + \frac{10}{k}) \cdot \text{LPOPT}_{k-\text{ECSM}} \leq (1 + \frac{10}{k}) \cdot \text{OPT}_{k-\text{ECSM}}$.

Proof of Theorem 1.3. We now give the proof without assuming that k is given in unary. Let y be an optimal solution to $(k+10)$-ECSS LP of cost at most $\text{LPOPT}_{(k+10)-\text{ECSM}}$. Next, apply Theorem 2.1 to y to compute a solution z. Return (the edge multiset naturally corresponding to) z as our solution. y is computable by the Ellipsoid Method [30] and standard separation oracles. Furthermore, by the guarantees of Theorem 2.1 we have that our solution is feasible for k-ECSM and computable in poly-time. Lastly, by Theorem 2.1 and Equation (1) its cost is upper bounded by

$$\text{LPOPT}_{(k+10)-\text{ECSM}} \leq \left(1 + \frac{10}{k}\right) \cdot \text{LPOPT}_{k-\text{ECSM}}. \quad \square$$

Thus, in what follows, we focus on showing Theorem 2.1. Furthermore, observe that in order to do so it suffices to show the result for even k since, if we are given an odd k, applying the result to (the even number) $k-1$ immediately gives the result for (the odd number) k. Thus, in what follows we assume that k is even.

3 ALGORITHM FOR THE ROUNDING THEOREM

Having reduced our problem to showing Theorem 2.1, we proceed to describe the algorithm for Theorem 2.1.

As discussed earlier, our algorithm for Theorem 2.1 uses iterative relaxation techniques. Informally our algorithm is as follows. We repeatedly solve an LP that tries to round y. Each time we solve our LP, either our solution is integral in which case we return our

[6]$\mathbb{1}[k \text{ odd}]$ is 1 if k is odd and 0 otherwise.

solution, it has a newly integral edge which we freeze at its current value, it has an edge with value 0 which we delete, or we perform one of the following two relaxations of our LP:

(1) **Ghost Value Augmentation:** we costlessly increase the LP value between two carefully chosen vertices. Specifically, if there are two vertices u and v that have nearly $k/2$ total LP value on edges between them (namely total edge value in $[k/2 - 2, k/2)$) then we add "ghost values" by increasing the LP value between u and v by 2 at cost 0.
(2) **Drop/Contract:** we drop a carefully chosen set's constraint and contract it. Specifically, if there is a tight set S (i.e., a set with exactly k total LP edge mass leaving it) corresponding to a constraint in the LP, such that S is (i) minimal in the sense that it contains no strict subset corresponding to a tight LP constraint, and (ii) has at most 3 fractional edges incident to it, then we remove the constraint corresponding to S from our LP and (if the set contains 2 or more vertices) we contract it into a single vertex.

As we later argue, we will always be in one of the above cases. The final solution we return will ignore the ghost values, and we will show that even after ignoring the ghost values our solution is well-connected.

We now more precisely define our LP given an input vector $y \in \mathbb{R}^E_{\geq 0}$ which we would like to round. We denote by $\lfloor y \rfloor$ and $\lceil y \rceil$ the integral vector obtained from y by taking the floor and ceiling of each component respectively. We let $x \in [\lfloor y \rfloor, \lceil y \rceil]$ denote that $\lfloor y_e \rfloor \leq x_e \leq \lceil y_e \rceil$ for each $e \in E$. To achieve a clean and elegant bookkeeping of our ghost values, we will save them in a separate vector $g \in \mathbb{Z}^E_{\geq 0}$. Thus, for a given y and g, the LP we will solve is the following (potentially with extra constraints for frozen coordinates).

$$\begin{array}{llll} \min & c^\top x & & \\ & x(\delta(S)) & \geq & k - g(\delta(S)) \quad \forall S \subseteq V \setminus \{r\}, S \neq \emptyset \\ & x & \in & [\lfloor y \rfloor, \lceil y \rceil]. \end{array} \tag{k–EC Ghost LP}$$

For a ghost value vector $g \in \mathbb{Z}^E_{\geq 0}$, we call the constraint $x(\delta(S)) \geq k - g(\delta(S))$ the *g-cut constraint for* S. We say that such a constraint is *y-tight* if $y(\delta(S)) = k - g(\delta(S))$. Analogously, we will say that a constraint $x_e = p_e$ for $p_e \in \mathbb{Z}_{\geq 1}$ on a single edge e is *y-tight* if $y_e = p_e$.

Our algorithm is formally described in Algorithm 1 and uses the following notation. For any vector $y \in \mathbb{R}^E$, we denote by $\text{frac}(y) := \{e \in E : y_e \notin \mathbb{Z}\}$ all edges with a fractional y-value. To clearly distinguish between the original input graph and the graph at each iteration of our algorithm, we will denote the original input graph by $\overline{G} = (\overline{V}, \overline{E})$ and the graph used in each iteration of the algorithm (i.e., $\overline{G}$ after some vertex contractions and edge deletions) by $G = (V, E)$. We will be explicit about the specific graph considered. For any two vertex sets $S, T \subseteq V$, we denote by $E(S, T) \subseteq E$ all edges with one endpoint in S and one in T. For vertices u, v and $S \subseteq V$, we also use the shorthand $E(u, v) := E(\{u\}, \{v\})$ and $E(u, S) := E(\{u\}, S)$. For $S \subseteq V$, we let $E[S] := \{\{u, v\} \in E : u, v \in S\}$ be all edges with both endpoints in S. We note that even though one could perform multiple ghost value augmentations or multiple cut relaxations in a single iteration of the while loop, Line 9 only performs a single such operation per iteration for simplicity.

Algorithm 1: Main Algorithm. (k is even.)

```
1  Initialization
2      $z(e) = 0$ for all $e \in \overline{E}$.   // $z \in \mathbb{Z}^{\overline{E}}_{\geq 0}$ will be the returned solution.
3      $g(e) = 0$ for all $e \in \overline{E}$.   // Start with all ghost values being set to 0.
4      $G = (V, E) := (\overline{V}, \overline{E})$.
5      Let LP_Alg be $k$–EC Ghost LP using $y$ and update $y$ to an optimal vertex solution.
6      Delete from $G$ all edges $e \in E$ with $y_e = 0$.
7  while $y \notin \mathbb{Z}^E$ do
8      if there are distinct vertices $u, v \in V$ with $(y + g)(E(u, v)) \in \left[\frac{k}{2} - 2, \frac{k}{2}\right)$ then
9          Ghost Value Augmentation: set $g_e = g_e + 2$ for an arbitrary edge $e \in E(u, v)$.
10     else if there is a $y$-tight $g$-cut constraint $x(\delta(S)) \geq k - g(\delta(S))$ in LP_Alg such that
             (i) LP_Alg does not contain a $y$-tight $g$-cut constraint $x(\delta(T)) \geq k - g(\delta(T))$ for $T \subsetneq S$, and
             (ii) $|\text{frac}(y) \cap \delta(S)| \leq 3$,
11      then
12         $z_e = y_e$ for all $e \in E[S]$.
13         Drop/Contract: Contract set $S$ in $G$ and remove $g$-cut constraint for $S$ from LP_Alg.
14     Compute an optimal vertex solution $y$ to LP_Alg (with ghost values g).
15     Delete from $G$ (and from LP_Alg) all edges $e \in E$ with $y_e + g_e = 0$.
16     For all $e \in E$ with $y_e \in \mathbb{Z}$, we add the constraint $x_e = y_e$ to LP_Alg.
17 $z_e = y_e$ for all $e \in E$.
18 Return $z$.
```

Algorithm Intuition. We summarize the intuition for our algorithm. As discussed earlier, a natural approach to rounding our vector y would be to argue that whenever we re-solve our LP and it does not have a newly integral edge, there must be a constraint of our LP corresponding to a set that has at most $O(1)$ fractional edges crossing it. Such a constraint can be safely dropped from our LP and doing so corresponds to our drop/contract relaxation. However, standard arguments that the non-existence of such a set implies a newly integral edge would require showing that the relevant cut constraints can be uncrossed into a laminar family. This uncrossing is not necessarily possible if we have already dropped some of our cut constraints. The somewhat unusual operation of ghost value augmentations will rescue us from this situation. In particular, we will see that the cases where uncrossing fails are exactly those when we are able to perform a ghost value augmentation. Indeed, a ghost value augmentation can be thought of as its own sort of cut relaxation: putting a ghost value of 2 units on some edge $e \in E$, simply corresponds to replacing the cut constraints $x(\delta(S)) \geq k$ by $x(\delta(S)) \geq k-2$ for all $S \subseteq V$ with $e \in \delta(S)$.

4 ANALYSIS OF OUR ALGORITHM

We proceed to analyze our algorithm. Doing so will require addressing several non-trivial issues. First, it is not a priori clear that our algorithm terminates and, in particular, that we can always make progress by some relaxation or edge deletion or freezing. Second, even if the algorithm terminates, it is not clear that the returned vector z is integral, much less that it has value at least $k - O(1)$ on every cut. For integrality, we will need to argue that, whenever we contract a set, all of its internal edges are integral. For near-k-edge-connectivity, it is particularly not clear that we do not end up with a single cut across which we have performed many ghost value augmentations, and so we will need to argue that this does not happen. Lastly, there are several efficiency concerns to address, including how to find the set $S \subseteq V$ of Line 13 if there is one satisfying the stated criteria.

Before addressing these challenges, we introduce some notation we use throughout our analysis. For brevity, when saying that a property holds at *any iteration of the algorithm*, we mean that it holds at the beginning of any iteration of the while loop in Algorithm 1. As in Algorithm 1, we will let $\mathrm{LP}_{\mathrm{Alg}}$ be the LP used by our algorithm at the beginning of an iteration. Note that although we compute our vertex solution y at the end of an iteration and then possibly delete edges from G and $\mathrm{LP}_{\mathrm{Alg}}$, even after deleting said edges y remains a vertex to the resulting $\mathrm{LP}_{\mathrm{Alg}}$.[7] In other words, y is a vertex solution to $\mathrm{LP}_{\mathrm{Alg}}$ at the beginning of each iteration. Likewise, we denote by $G=(V,E)$, $y \in \mathbb{R}^E_{\geq 0}$, and $g \in \mathbb{Z}^E_{\geq 0}$ the current graph, the current vertex solution to $\mathrm{LP}_{\mathrm{Alg}}$, and the current ghost values, respectively, at that iteration. Throughout our analysis, we will represent by $\mathcal{R} \subseteq 2^{\overline{V} \setminus \{r\}}$ all sets that are contracted throughout the algorithm. More precisely, for some set $R \subseteq \overline{V} \setminus \{r\}$, we have $R \in \mathcal{R}$ if R is not a singleton and during some point in the algorithm we had a vertex that, when undoing the contractions, corresponds to the vertex set R. Because we perform contractions consecutively, the family $\mathcal{R}$ is laminar. At any iteration of the algorithm, each vertex $v \in V$ of the current graph $G=(V,E)$ is either an original vertex, i.e., $v \in \overline{V}$, or it corresponds to a set $R \in \mathcal{R}$ that was contracted in a prior iteration of the algorithm.

4.1 Termination of Algorithm

We start by showing that Algorithm 1 terminates.

To show that Algorithm 1 terminates, we show that, at any iteration of the while loop, if y is not yet integral, and we cannot perform a ghost value augmentation, then we can perform a cut relaxation. Because cut relaxations require cut constraints with high integrality, i.e., the number of fractional edges must be at most 3, we will derive sparsity results in the following. These results provide upper bounds on the number of y-fractional edges, or show that certain edges must have integral y-values.

[7] This can be seen by, e.g., observing that y is a vertex solution if and only if it is feasible and it is the unique solution to $\bar{A}x = b$ (i.e., the columns of $\bar{A}$ are independent) where $\bar{A}$ is the subsystem of constraints of A tight for y. For $\mathrm{LP}_{\mathrm{Alg}}$, the columns of $\bar{A}$ remain independent even after deleting the column and row corresponding to an e with $y_e = 0$ so y without this coordinate remains a vertex solution.

We start with a basic property on the $(y+g)$-load on each cut. A consequence is that at any iteration, we have $(y+g)(\delta(v)) \geq k-2$ for any vertex v, and for every $S \subseteq V$ with $2 \leq |S| \leq n-2$ we have $(y+g)(\delta(v)) \geq k$. Thus our graph is close to being fractionally k-edge-connected.

Lemma 4.1. *At any iteration of Algorithm 1, we have*

$$(y+g)(\delta(S)) \geq k-2 \qquad \forall S \subseteq V \setminus \{r\}, S \neq \emptyset.$$

Also, if for any non-empty set $S \subseteq V \setminus \{r\}$, we have $(y+g)(\delta(S)) = k-2$, then $y_e \in \mathbb{Z}_{\geq 0}$ $\forall e \in \delta(S)$.

Proof. If the g-cut constraint corresponding to S is still in $\mathrm{LP}_{\mathrm{Alg}}$, then we even have $(y+g)(\delta(S)) \geq k$. Otherwise, let $\widetilde{y}$ and $\widetilde{g}$ be the $\mathrm{LP}_{\mathrm{Alg}}$ solution and ghost values, respectively, at the iteration when the g-cut constraint corresponding to S got relaxed. Hence, $(\widetilde{y}+\widetilde{g})(\delta(S)) \geq k$, because, as before, the g-cut constraint corresponding to S is part of $\mathrm{LP}_{\mathrm{Alg}}$ at the beginning of the iteration when S gets relaxed. Moreover, $|\operatorname{frac}(\widetilde{y}) \cap \delta(S)| \leq 3$, which implies

$$(\lfloor \widetilde{y} \rfloor + \widetilde{g})(\delta(S)) > k-3.$$

Because the left-hand side is integral, we get

$$(\lfloor \widetilde{y} \rfloor + \widetilde{g})(\delta(S)) \geq k-2.$$

The first statement now follows by observing that integral y-values get fixed, and hence $y \geq \lfloor \widetilde{y} \rfloor$, and that $g \geq \widetilde{g}$, because ghost values are non-decreasing. Moreover, this reasoning shows that to get $(y+g)(\delta(S)) = k-2$, we need $y(\delta(S)) = \lfloor \widetilde{y} \rfloor(\delta(S))$. Because $y \geq \lfloor \widetilde{y} \rfloor$, this implies as desired $y_e = \lfloor \widetilde{y}_e \rfloor \in \mathbb{Z}_{\geq 0}$ $\forall e \in \delta(S)$. □

The following lemma formalizes that y has low fractionality on any set of parallel edges. Below, recall that y is a vertex solution to the relevant LP in each iteration.

Lemma 4.2. *At any iteration of the algorithm, we have*

$$|\operatorname{frac}(y) \cap E(u,v)| \leq 1 \qquad \forall u,v \in V, u \neq v.$$

Proof. Assume for the sake of deriving a contradiction that there is a pair of vertices $u,v \in V, u \neq v$ with $|\operatorname{frac}(y) \cap E(u,v)| \geq 2$. Let $e_1, e_2 \in \operatorname{frac}(y) \cap E(u,v)$ be two distinct edges. Observe that for

$$\epsilon = \min\left\{y_{e_1} - \lfloor y_{e_1} \rfloor, \lceil y_{e_1} \rceil - y_{e_1}, y_{e_2} - \lfloor y_{e_2} \rfloor, \lceil y_{e_2} \rceil - y_{e_2}\right\} > 0,$$

we have that both $y + \epsilon(\chi^{\{e_1\}} - \chi^{\{e_2\}})$ and $y - \epsilon(\chi^{\{e_1\}} - \chi^{\{e_2\}})$ are feasible for the LP. This holds because there is no constraint that contains e_1 and not e_2 or vice-versa, except for integral bounds (lower/upper bounds, or equality constraints) on the values of y_{e_1} and y_{e_2}. This contradicts that y is a vertex solution to the LP. □

To obtain strong enough sparsity to show that a cut relaxation is possible, we use a classic strategy. Namely, we first show that, at any iteration of Algorithm 1, a vertex solution can be described as the unique solution to a very structured system of y-tight LP constraints. More precisely, the y-tight g-cut constraints in this system correspond to sets that form a laminar family. This is where we crucially exploit the use of ghost values, without which a statement as below would be wrong, as demonstrated in Fig. 4.

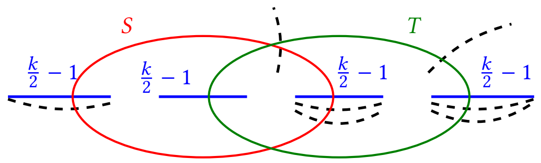


Figure 4: A situation in which ghost value augmentation is necessary. All dotted edges have value 1/2, so S and T are tight but not dropped. $S \cap T$ and $S \setminus T$, however, have been dropped and thus are allowed to drop below connectivity k. Therefore it is not possible to uncross S and T. Note that the integrality of the rightmost blue edge leaving T is not necessary. In particular, this rightmost blue edge could be any number of fractional edges and the instance would have the same behavior.

Lemma 4.3. *Consider any iteration of Algorithm 1 where no ghost value augmentation can be performed. Let $\mathcal{L} \subseteq \{S \subseteq V \setminus \{r\} : y(\delta(S)) = k - g(\delta(S))\}$ be a maximal laminar family corresponding to y-tight g-cut constraints in $\mathrm{LP}_{\mathrm{Alg}}$. Then, the linear equation system consisting of*

$$x(\delta(L)) = k - g(\delta(L)) \quad \forall L \in \mathcal{L},$$

together with all y-tight constraints on single edges in $\mathrm{LP}_{\mathrm{Alg}}$, i.e., constraints of type $x_e = p_e$ for some $e \in E$ and $p_e \in \mathbb{Z}$, form a full column rank system[8] of y-tight constraints of $\mathrm{LP}_{\mathrm{Alg}}$. Thus, because y is a vertex solution to $\mathrm{LP}_{\mathrm{Alg}}$, it is the unique solution to this system.

Proof. Assume for the sake of contradiction that there is an iteration of the algorithm where no ghost value augmentation can be performed, and such that there is a maximal laminar family $\mathcal{L} \subseteq 2^{V\setminus\{r\}}$ corresponding to y-tight g-cut constraints in $\mathrm{LP}_{\mathrm{Alg}}$ fulfilling the following: The linear equation system consisting of all equations corresponding to y-tight g-cut constraints of sets in $\mathcal{L}$, together with equations corresponding to all y-tight constraints on single edges in $\mathrm{LP}_{\mathrm{Alg}}$, does not have full column rank. We call this linear equation system the *reference system.*

The reference system not having full column rank implies that there must be a y-tight g-cut constraint $x(\delta(S)) \geq k - g(\delta(S))$ not implied by it. Among all such sets $S \subseteq V \setminus \{r\}$, we choose one where

$$\mathcal{L}_S := \{L \in \mathcal{L} : L \text{ and } S \text{ are crossing}\}$$

has smallest cardinality. Because the constraint $x(\delta(S)) \geq k - g(\delta(S))$ is not implied by the reference system, and we did not add that constraint to it, it must be that S crosses some set in $\mathcal{L}$. Hence, $|\mathcal{L}_S| \geq 1$. Let $L \in \mathcal{L}_S$. We continue by showing that the g-cut constraints corresponding to certain sets must have been dropped, allowing us to reduce to a setting similar to Fig. 4.

Claim 4.4. *We have*

(i)
- *The g-cut constraint corresponding to $S \cap L$ has been dropped from $\mathrm{LP}_{\mathrm{Alg}}$ and the equation $x(\delta(S \cap L)) = k - g(\delta(S \cap L))$ is not implied by the reference system,*
- *$(y+g)(\delta(S \cap L)) \leq k$, and*
- *if $(y+g)(\delta(S \cap L)) = k$, then $(y+g)(\delta(S \cup L)) = k$ and $E(S \setminus L, L \setminus S) = \emptyset$.*

(ii) There is a set Q_1 among $\{S \setminus L, L \setminus S\}$ (let Q_2 be the other set) such that
- *the g-cut constraint corresponding to Q_1 has been dropped from $\mathrm{LP}_{\mathrm{Alg}}$,*
- *$(y+g)(\delta(Q_1)) \leq k$, and*
- *if $(y+g)(\delta(Q_1)) = k$, then also $(y+g)(\delta(Q_2)) = k$ and $E(S \cap L, V \setminus (S \cup L)) = \emptyset$.*

Proof. We start by proving (i). If $(y+g)(\delta(S \cap L)) < k$, then the g-constraint corresponding to $S \cap L$ must have been dropped, and all conditions of (i) are fulfilled. Hence, assume from now on $(y+g)(\delta(S \cap L)) \geq k$.

We use the following well-known basic relation, which can be verified by checking that the shown equation holds for each coordinate (note that every coordinate corresponds to an edge):

$$\chi^{\delta(S)} + \chi^{\delta(L)} = \chi^{\delta(S\cup L)} + \chi^{\delta(S\cap L)} + 2\chi^{E(S\setminus L, L\setminus S)}. \tag{2}$$

By taking the scalar product of the above equation with $y+g$, we establish that $(y+g)(\delta(S)) + (y+g)(\delta(L))$ is equal to:

$$(y+g)(\delta(S \cup L)) + (y+g)(\delta(S \cap L)) + 2(y+g)(E(S \setminus L, L \setminus S)). \tag{3}$$

Note that the g-cut constraint corresponding to $S \cup L$ has not been dropped yet because $|S \cup L| \geq 2$. Hence, $(x+g)(\delta(S \cup L)) \geq k$. Together with $(y+g)(\delta(S)) = (y+g)(\delta(L)) = k$ and $(y+g)(\delta(S \cap L)) \geq k$, Equation (3) thus implies $(y+g)(\delta(S \cap L)) = (y+g)(\delta(S \cup L)) = k$ and $(y+g)(E(S \setminus L, S \setminus L)) = 0$. The last equation implies $E(S \setminus L, L \setminus S) = \emptyset$, because $\mathrm{supp}(y+g) = E$, which holds because we deleted all edges with $(y+g)$-value zero. Hence, (3) simplifies to

$$\chi^{\delta(S)} + \chi^{\delta(L)} = \chi^{\delta(S\cup L)} + \chi^{\delta(S\cap L)}.$$

Because $\chi^{\delta(L)}$ is a row of our reference system and $\chi^{\delta(S)}$ is not spanned by the rows of our reference system, we have that either the equation $x(\delta(S\cup L)) = k - g(\delta(S\cup L))$ or $x(\delta(S\cap L)) = k - g(\delta(S\cap L))$ is not implied by our reference system. Note that both equations correspond to y-tight g-cut constraints as shown above. Because the g-cut constraint corresponding to $S \cup L$ is still part of $\mathrm{LP}_{\mathrm{Alg}}$ as $|S \cup L| \geq 2$, and $\mathcal{L}_{S\cup L} \subsetneq \mathcal{L}_S$,[9] this g-cut constraint must be implied by the reference system. For otherwise, we could have chosen $S \cup L$ instead of S, which violates our choice of S. Hence, the g-cut constraint corresponding to $S \cap L$ is not implied by our reference system. Because also $\mathcal{L}_{S\cap L} \subsetneq \mathcal{L}_S$, the g-cut constraint corresponding to $S \cap L$ cannot be part of $\mathrm{LP}_{\mathrm{Alg}}$ anymore, as this would again violate our choice of S.

For point (ii) we can follow an analogous approach as for (i). If there is a set $Q \in \{S \setminus L, L \setminus S\}$ with $(y+g)(\delta(Q)) < k$, then we choose $Q_1 := Q$. Indeed, the g-cut constraint corresponding to Q_1 must have been dropped from $\mathrm{LP}_{\mathrm{Alg}}$ because $(y+g)(\delta(Q_1)) < k$. Thus, Q_1 fulfills all conditions of point (ii). Hence, from now on assume $(y+g)(\delta(S \setminus L)) \geq k$ and $(y+g)(\delta(L \setminus S)) \geq k$.

We use the following well-known basic relation:

$$\chi^{\delta(S)} + \chi^{\delta(L)} = \chi^{\delta(S\setminus L)} + \chi^{\delta(L\setminus S)} + 2\chi^{E(S\cap L, V\setminus(S\cup L))}.$$

[8] A system of linear equations $Ax = b$ has full column rank if the rank of A is equal to the number of columns of A.

[9] $\mathcal{L}_{S\cup L} \subsetneq \mathcal{L}_S$ (and analogously $\mathcal{L}_{S\cap L}, \mathcal{L}_{S\setminus L}, \mathcal{L}_{L\setminus S} \subsetneq \mathcal{L}_S$) follows from the following observation on laminar families. Let $\mathcal{L}$ be a laminar family over some finite ground set N, and let $S \subseteq N$ and $L \in \mathcal{L}$ be such that S crosses L. Then any set in $\mathcal{L}$ that crosses $S \cup L$ also crosses S. However, the set L, which crosses S, does not cross $S \cup L$.

Because $(y+g)(\delta(S)) = k$ and $(y+g)(\delta(L)) = k$, we must have $(y+g)(\delta(S\setminus L)) = k$, $(y+g)(\delta(L\setminus S)) = k$, and $E(S\cap L, V\setminus(S\cup L)) = \emptyset$. This implies

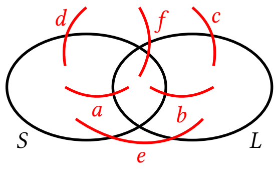
$$\chi^{\delta(S)} + \chi^{\delta(L)} = \chi^{\delta(S\setminus L)} + \chi^{\delta(L\setminus S)}.$$

Because $\chi^{\delta(L)}$ is a row in our reference system and $\chi^{\delta(S)}$ is not spanned by the rows of your reference system, we have that either the equation $x(\delta(S\setminus L)) = k - g(\delta(S\setminus L))$ or $x(\delta(L\setminus S)) = k - g(\delta(L\setminus S))$ (or both) is not implied by the reference system. Note that both equations correspond to y-tight g-cut constraints as shown above. Let $Q_1 \in \{S\setminus L, L\setminus S\}$ be such that $x(\delta(Q_1)) = k - g(\delta(Q_1))$ is not implied by the reference system. Because $\mathcal{L}_{Q_1} \subsetneq \mathcal{L}_S$, the g-cut constraint corresponding to Q_1 must have been dropped from $\mathrm{LP}_{\mathrm{Alg}}$. For otherwise, we could have chosen Q_1 instead of S, which violates our choice of S. Hence, Q_1 fulfills all conditions of point (ii). □ (Claim 4.4)

By Claim 4.4 (i), the g-cut constraint corresponding to $S\cap L$ got relaxed/dropped from $\mathrm{LP}_{\mathrm{Alg}}$ and $x(\delta(S\cap L)) = k - g(\delta(S\cap L))$ is not implied by the reference system. Moreover, Claim 4.4 (ii) implies that the constraint corresponding to at least one of the sets $S\setminus L$ or $L\setminus S$ got dropped. Recall that dropped sets got contracted and therefore correspond to singletons. We finish the proof by showing that a ghost value augmentation could have been performed with respect to the singleton $S\cap L$ and either $S\setminus L$ or $L\setminus S$.

To this end consider the different $(y+g)$-loads between the four sets $S\setminus L$, $S\cap L$, $L\setminus S$, and $V\setminus(S\cup L)$. For brevity let

$$\begin{aligned}
a &:= (y+g)(E(S\cap L, S\setminus L))\\
b &:= (y+g)(E(S\cap L, L\setminus S))\\
c &:= (y+g)(E(L\setminus S, V\setminus(S\cup L)))\\
d &:= (y+g)(E(S\setminus L, V\setminus(S\cup L)))\\
e &:= (y+g)(E(S\setminus L, L\setminus S))\\
f &:= (y+g)(E(S\cap L, V\setminus(S\cup L))).
\end{aligned}$$



The above definitions immediately lead to the following basic relations

$$\begin{array}{lll}
k & = (y+g)(\delta(S)) & = b+d+e+f\\
k & = (y+g)(\delta(L)) & = a+c+e+f\\
k & \geq (y+g)(\delta(S\cap L)) & = a+b+f\\
k-2 & \leq (y+g)(\delta(S\cap L)) & = a+b+f\\
k-2 & \leq (y+g)(\delta(S\setminus L)) & = a+d+e\\
k-2 & \leq (y+g)(\delta(L\setminus S)) & = b+c+e,
\end{array} \tag{4}$$

where the equalities at the start of the first two lines hold because S and L correspond to y-tight g-cut constraints, the inequality in the third line follows from Claim 4.4 (i), and the inequalities at the start of the last three lines hold by Lemma 4.1.

By subtracting the first relation above from the sum of the forth and fifth one, we get

$$a \geq \frac{k}{2} - 2. \tag{5}$$

Analogously, by subtracting the second relation in (4) from the sum of the forth and sixth one, we get

$$b \geq \frac{k}{2} - 2. \tag{6}$$

Let $Q_1 \in \{S\setminus L, L\setminus S\}$ be the set as described in Claim 4.4 (ii). We will show that a ghost value augmentation could have been performed between Q_1 and $S\cap L$. Note that these two sets correspond to singletons, and (5)/(6) show that the (parallel) edges between them fulfill the lower bound condition necessary to apply a ghost value augmentation. It remains to show that, if $Q_1 = S\setminus L$, we have $a < \frac{k}{2}$, and analogously, if $Q_1 = L\setminus S$, then $b < \frac{k}{2}$. Then all conditions are fulfilled to apply a ghost value augmentation, which is the desired contradiction.

The proof for the two cases is identical; hence, we assume from now on $Q_1 = S\setminus L$. We have

$$\begin{aligned}
a &= (y+g)(E(S\setminus L, S\cap L))\\
&= \frac{1}{2}\left[(y+g)(\delta(S\cap L)) + (y+g)(\delta(S\setminus L)) - (y+g)(\delta(S))\right]\\
&\leq \frac{k}{2},
\end{aligned}$$

where the inequality follows from the first and third relation in (4), and from $(y+g)(\delta(S\setminus L)) \leq k$, which holds by Claim 4.4 (ii) and the assumption $Q_1 = S\setminus L$. It remains to show that $a \neq \frac{k}{2}$. Assume for the sake of deriving a contradiction that $a = \frac{k}{2}$. We therefore must have $(y+g)(\delta(S\cap L)) = k$ and $(y+g)(\delta(S\setminus L)) = k$, which implies by Claim 4.4 (ii) in particular $(y+g)(\delta(S\cup L)) = k$, $(y+g)(\delta(L\setminus S)) = k$, and $E(S\cap L, V\setminus(S\cup L)) = \emptyset$. The contradiction we derive will be that the equation $x(\delta(S\cap L)) = k - g(\delta(S\cap L))$ is implied by $\mathrm{LP}_{\mathrm{Alg}}$, which violates Claim 4.4 (i). Note that because $E(S\cap L, V\setminus(S\cup L)) = \emptyset$, we get

$$\chi^{\delta(S\cap L)} = \chi^{E(S\setminus L, S\cap L)} + \chi^{E(L\setminus S, S\cap L)}. \tag{7}$$

Moreover, $a = \frac{k}{2} \in \mathbb{Z}$ (recall we are assuming k is even) implies $y(e) \in \mathbb{Z}$ for $e \in E(S\setminus L, S\cap L)$ because of Lemma 4.2. Hence, the y-values on the edge $E(S\setminus L, S\cap L)$ are fixed by the integrality constraints, which are part of the reference system. Moreover, we also have $b = (x+y)(\delta(S\cap L)) - a = \frac{k}{2}$, where the first equality holds because $E(S\cap L, V\setminus(S\cup L)) = \emptyset$. Hence, if also the g-cut constraint corresponding to $L\setminus S$ got dropped from $\mathrm{LP}_{\mathrm{Alg}}$, then we have analogously $y(e) \in \mathbb{Z}$ for $e \in E(L\setminus S, S\cap L)$. However, then (7) implies that $x(\delta(S\cap L)) = k - g(\delta(S\cap L))$ is implied by $\mathrm{LP}_{\mathrm{Alg}}$, which is a contradiction. Thus, it remains to consider the case where the g-cut constraint corresponding to $L\setminus S$ is still part of $\mathrm{LP}_{\mathrm{Alg}}$. Note that

$$\chi^{E(L\setminus S, S\cap L)} = \frac{1}{2}\left[\chi^{\delta(L\setminus S)} + \chi^{\delta(S)} - \chi^{\delta(S\cup L)}\right].$$

Observe that $\chi^{\delta(L\setminus S)}$, $\chi^{\delta(S\cup L)}$, and $\chi^{\delta(S)}$ are all row vectors of our reference system. Hence, also the row vector $\chi^{E(L\setminus S, S\cap L)}$ is implied by our reference system. This in turn implies by (7) that the equation $x(\delta(S\cap L)) = k - g(\delta(S\cap L))$ is implied by our reference system, which leads to the desired contradiction. □ (Lemma 4.3)

For completeness, we now present a classic reasoning, adjusted to our context, showing that the sparsity of $\mathrm{LP}_{\mathrm{Alg}}$ at any iteration of the algorithm can be bounded by the number of linearly independent y-tight g-cut constraints in a full column rank equation system of tight LP constraints. We state the result for an equation system defining the $\mathrm{LP}_{\mathrm{Alg}}$ vertex y with an arbitrary family $\mathcal{L}$ corresponding to y-tight g-cut constraints. We later apply the statement with an equation system of y-tight constraints coming from Lemma 4.3, where the family $\mathcal{L}$ is laminar.

Lemma 4.5. *Consider an iteration of Algorithm 1 where y is not yet integral, and no ghost value augmentation can be performed. Consider a full column rank system of equations corresponding to y-tight constraints of $\mathrm{LP}_{\mathrm{Alg}}$, and let $\mathcal{L} \subseteq 2^{V\setminus\{r\}}$ be all cuts of y-tight g-cut constraints that correspond to an equation in the equation system. Then $|\operatorname{frac}(y)| \leq |\mathcal{L}|$.*

Proof. First, we can assume that the considered equation system, for simplicity we call it the *reference system*, is a square system. Indeed, if the reference system is not square, then we can successively remove equations from the system that are implied by the other equations of the system until we get a square system. Moreover, the implication of the statement for the square system implies the statement for the original system.

The square reference system has two types of equations:

- equations corresponding to y-tight g-cut constraints, i.e., $x(\delta(S)) = k - g(\delta(S))\ \forall S \in \mathcal{L}$, and
- equations corresponding to y-tight constraints on single edges, i.e., these are of the form $x_e = p_e$, where $e \in E$ and $p_e \in \mathbb{Z}_{\geq 1}$.

Let $F \subseteq E$ be all edges for which an equation of the second type is in the system. Because the reference system is square, we have $|E| = |\mathcal{L}| + |F|$. Moreover, all edges in F have integral y-values, which implies the result because of

$$|\operatorname{frac}(y)| \leq |E| - |F| = |\mathcal{L}|. \qquad \square$$

Finally, the following lemma shows that we make progress in each iteration of Algorithm 1.

Lemma 4.6. *At any iteration of Algorithm 1, if y is not integral and no ghost value augmentation can be applied (i.e., the algorithm is at an iteration where it reaches Line 10), then there is a g-cut constraint of $\mathrm{LP}_{\mathrm{Alg}}$ that can be relaxed.*

Proof. Let $\mathcal{L} \subseteq \{S \subseteq V \setminus \{r\} : x(\delta(S)) = k - g(\delta(S))\}$ be a maximal laminar family of sets corresponding to y-tight g-cut constraints of $\mathrm{LP}_{\mathrm{Alg}}$. By Lemma 4.3, these constraints, together with all y-tight constraints of $\mathrm{LP}_{\mathrm{Alg}}$ on single edges, correspond to a full column rank equation system with y being its unique solution. We successively remove from $\mathcal{L}$ constraints that are redundant in that system, until no g-cut constraint corresponding to a set in $\mathcal{L}$ is redundant.

Assume for the sake of deriving a contradiction that no cut relaxation can be applied. We will derive a contradiction by showing that this would imply $|\mathcal{L}| > |\operatorname{frac}(y)|$, which contradicts Lemma 4.5. To this end we use a token counting argument, where we assign two tokens to each edge in $\operatorname{frac}(y)$, and assign those tokens to the sets $\mathcal{L}$ such that each set in $\mathcal{L}$ gets at least 2 tokens, and at least one set gets strictly more than 2 tokens.

First, each edge $\{u, v\} \in \operatorname{frac}(y)$ assigns one token to the smallest set in $\mathcal{L}$ that contains u (if there is such a set) and one to the smallest set in $\mathcal{L}$ that contains v (if there is such a set). We then consider the sets in $\mathcal{L}$ in any smallest-to-largest order, and reassign excess tokens from children to their parent.[10] With a *smallest-to-largest* order, we mean any order such that when considering $L \in \mathcal{L}$, then all descendants of L in $\mathcal{L}$ have already been considered. We maintain the following invariant: After considering a set $L \in \mathcal{L}$, we have reassigned the tokens of L and its descendants in a way that L has at least 4 tokens, and each of its descendants has 2 tokens. By showing that this invariant can be maintained, the results follows because at the end of the procedure, the maximal sets in $\mathcal{L}$ will have obtained at least 4 tokens, and all other sets in $\mathcal{L}$ obtained 2 tokens.

We start by showing that the invariant holds for each minimal set L in $\mathcal{L}$. If the edges $\operatorname{frac}(y)$ have at least 4 endpoints in L, then the invariant holds. Otherwise, we have $|\operatorname{frac}(y) \cap E(L, V)| \leq 3$, and because we assumed that we cannot apply a cut relaxation to L, there must be a smaller set $S \subsetneq L$ corresponding to a y-tight g-cut constraint. We choose S to be a minimal such set. However, because $\delta(S) \subseteq E(L, V)$, we have $|\operatorname{frac}(y) \cap \delta(S)| \leq |\operatorname{frac}(y) \cap E(L, V)| \leq 3$, i.e., the set $\delta(S)$ contains at most 3 edges with fractional y-values. This implies that we could have applied a cut relaxation to S, which contradicts our assumption that no cut relaxation was possible.

Consider now a non-minimal set L in $\mathcal{L}$, and assume that the invariant holds for all of its children. If L has at least two children in $\mathcal{L}$, then it can get 2 tokens from each of them, and the invariant holds for L. Hence, assume that L has only one child $C \in \mathcal{L}$ in $\mathcal{L}$. In this case, L can get two tokens from C, which has 4 tokens. We complete the proof by showing that there are at least two edges of $\operatorname{frac}(y)$ with one endpoint in $L \setminus C$, which will give an additional 2 tokens to L, to obtain the 4 tokens required by the invariant. First observe that we must have

$$\operatorname{frac}(y) \cap \delta(C) \neq \operatorname{frac}(y) \cap \delta(L);$$

since otherwise the y-tight g-cut constraint corresponding to L is implied by the y-tight g-cut constraint that corresponds to C and the y-tight equality constraints on single edges. This implies

$$\operatorname{frac}(y) \cap \delta(L \setminus C) \neq \emptyset,$$

and already shows that L obtains at least one more token due to a fractional edge with an endpoint in $L \setminus C$. Because both L and C correspond to y-tight g-cut constraints, we have

$$y(\delta(L)) = k - g(\delta(L)), \text{ and} \tag{8}$$
$$y(\delta(C)) = k - g(\delta(C)). \tag{9}$$

Due to integrality of the ghost values g, this implies $y(\delta(L)) \in \mathbb{Z}$ and $y(\delta(C)) \in \mathbb{Z}$. Thus,

$$y(\delta(L)) - y(\delta(C)) = y(E(L \setminus C, V \setminus L)) - y(E(L \setminus C, C)) \in \mathbb{Z}. \tag{10}$$

Note that

$$\delta(L \setminus C) = E(L \setminus C, V \setminus L) \cup E(L \setminus C, C),$$

and (10) thus implies that $\delta(L \setminus C)$ cannot have a single edge with fractional y-value. Because $\operatorname{frac}(y) \cap \delta(L \setminus C) \neq \emptyset$, we have $|\operatorname{frac}(y) \cap \delta(L \setminus C)| \geq 2$, showing as desired that L gets at least two tokens from edges of $\operatorname{frac}(y)$ with one endpoint in $L \setminus C$. $\square$

Note that Lemma 4.6 readily implies that Algorithm 1 terminates. Indeed, the number of ghost value augmentations one can perform on an edge e is no more than $\lceil k/4 \rceil$, because then just the ghost values

[10]We use the usual notions like *children*, *parents*, and *descendants* for the laminar family $\mathcal{L}$. More precisely, for $L_1, L_2 \in \mathcal{L}$ with $L_2 \subsetneq L_1$, we call L_1 an *ancestor* of L_2, and L_2 is a *descendant* of L_1. If L_2 is a descendant of L_1 and there is no set $L \in \mathcal{L}$ with $L_2 \subsetneq L \subsetneq L_1$, then L_2 is a *child* of L_1, and L_1 is called the *parent* of L_2.

alone already provide a load of at least $k/2$ on e, and e therefore does not qualify anymore for a ghost value augmentation. Moreover, the sets we relax correspond to the laminar family $\mathcal{R}$, which can have size at most $O(|\overline{V}|)$.

However, bounding the number of ghost value augmentations per edge by $\lceil k/4 \rceil$ turns out to be very loose. Actually, as we will show next, we can perform at most one ghost value augmentation per edge, which is a result that is also helpful later on. This holds because whenever a ghost value augmentation is performed on an edge with endpoints u and v, then a large y-value on $E(u,v)$ is integral and, because we fix/freeze integral values, we will have a load of at least $\lfloor y \rfloor (E(u,v))$ between the vertices u and v in any future iteration where u and v did not yet get contracted through a cut relaxation. This load will be too high for another ghost value augmentation to be performed on an edge with endpoints u and v.

The following lemma implies in particular that at most one ghost value augmentation can be applied per edge. We defer the proof to the full version.

Lemma 4.7. *At any iteration of the algorithm when a ghost value augmentation is performed between two vertices u and v, then no edge in $E(u,v)$ has strictly positive ghost value, i.e., $g_f = 0$ for $f \in E(u,v)$.*

This is a desirable property when proving that the output is highly connected after removing the ghost values.

4.2 Guarantees on Cut Constraints

Our goal now is to show that the solution z returned by Algorithm 1 satisfies $z(\delta(S)) \geq k - 9$ for all $S \subseteq \overline{V} \setminus \{r\}$ with $S \neq \emptyset$ (again, recall we are assuming that k is even; hence the 9 instead of 10). To achieve guarantees on z, we crucially and repeatedly exploit that integral y-values get frozen/fixed. This guarantees that at any iteration of Algorithm 1, the solution y to $\mathrm{LP}_{\mathrm{Alg}}$ provides the following lower bound on entries of z, i.e., the solution returned by the algorithm.

$$\lfloor y \rfloor_e \leq z_e \qquad \forall e \in E.$$

We recall that z in the above inequality is the final solution returned by our algorithm (not an intermediate value of z).

Due to space constraints, we defer the proof that the output is $(k-9)$-edge-connected to the full version.

4.3 Cost of Returned Solution and Running Time

The required cost bound is easily proven.

Observation 4.8. *The solution z returned by Algorithm 1 has cost bounded by $c^\top y$ where y is the input vector of Theorem 2.1.*

Proof. We denote by $\mathrm{LPOPT}_{k-\mathrm{Ghost}}$ the optimal value of the $\mathrm{LP}_{\mathrm{Alg}}$ solution computed in Line 5 of Algorithm 1. Observe that

$$\mathrm{LPOPT}_{k-\mathrm{Ghost}} \leq c^\top y$$

since y is feasible for this LP (where again, y is the input to Theorem 2.1). Whenever we change $\mathrm{LP}_{\mathrm{Alg}}$ in Algorithm 1, we do so by either increasing ghost values in Line 9, performing a cut relaxation in Line 13, or by adding equality constraints in Line 16. All of these operations have the property that the previous LP solution is still valid for the new LP. (In case of a cut relaxation, we can use the values of the previous LP solution on the non-contracted edges.) This implies that the returned solution z fulfills $c^\top z \leq \mathrm{LPOPT}_{k-\mathrm{Ghost}} \leq c^\top y$, as desired. □

We defer the proof that the algorithm can be implemented in polynomial time to the full version.

5 HARDNESS OF APPROXIMATION OF k-ECSM

[50] showed APX-hardness for 2-ECSM by showing it is essentially the same problem as the metric version of 2-ECSS but left hardness of k-ECSM for $k > 2$ and as a function of k open, stating

What [APX-hardness of 2-ECSM] leaves to be desired is hardness for k-ECSM, $k > 2$ and asymptotic dependence on k. Why is it hard to show these problems are hard?…A new trick seems to be needed to get a good hardness result for k-ECSM.

We provide such a trick to show $1 + \Omega(1/k)$ hardness for k-ECSM. Specifically, we reduce from the unweighted tree augmentation problem (TAP) problem, which is known to be $(1+\epsilon_{\mathrm{TAP}})$-hard-to-approximate for some constant $\epsilon_{\mathrm{TAP}} > 0$. We recall below the hardness statement we show.

Theorem 1.4. *There exists a constant $\epsilon > 0$ such that there does not exist a poly-time algorithm which, given an instance of (unweighted) k-ECSM where k is part of the input, always returns a $\left(1 + \frac{\epsilon}{k}\right)$-approximate solution, unless* P = NP.

The above hardness shows the tightness of Theorem 1.1 up to constants. We prove Theorem 1.4 by showing hardness for all odd k. The basic idea is to reduce from unweighted TAP by using the following trick: if a k-ECSM solution on graph G' has value exactly k on a node w with two incident edges in G' and k is odd, then one can always make one edge incident to w have value $\lceil k/2 \rceil$ and one edge have value $\lfloor k/2 \rfloor$ while preserving the feasibility of the k-ECSM solution.

We defer the proof to the full version.

ACKNOWLEDGMENTS

This project received funding from Swiss National Science Foundation grant 200021_184622 and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 817750). The second author was supported by the National Science Foundation under Grants No. DMS-1926686, DGE-1762114, and CCF-1813135.

REFERENCES

[1] David Adjiashvili. 2018. Beating approximation factor two for weighted tree augmentation with bounded costs. *ACM Transactions on Algorithms (TALG)* 15, 2 (2018), 1–26.
[2] H. Angelidakis, D. Hyatt-Denesik, and L.j Sanitá. 2023. Node Connectivity Augmentation via Iterative Randomized Rounding. *Mathematical Programming, Series A* 199 (2023), 995–1031.
[3] V. Auletta, Y. Dinitz, Z. Nutov, and D. Parente. 1999. A 2-Approximation Algorithm for Finding an Optimum 3-Vertex-Connected Spanning Subgraph. *Journal of Algorithms* 32 (1999), 21–30.

[4] André Berger and Michelangelo Grigni. 2007. Minimum weight 2-edge-connected spanning subgraphs in planar graphs. In *International Colloquium on Automata, Languages and Programming (ICALP)*. Springer, 90–101.

[5] Sylvia Boyd, Joseph Cheriyan, Robert Cummings, Logan Grout, Sharat Ibrahimpur, Zoltán Szigeti, and Lu Wang. 2020. A 4/3-Approximation Algorithm for the Minimum 2-Edge Connected Multisubgraph Problem in the Half-Integral Case. In *International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, Vol. 176. 61:1–61:12.

[6] Sylvia Boyd, Yao Fu, and Yu Sun. 2016. A 5/4-approximation for subcubic 2EC using circulations and obliged edges. *Discrete Applied Mathematics* 209 (2016), 48–58.

[7] Robert Carr and R. Ravi. 1998. A New Bound for the 2-Edge Connected Subgraph Problem. In *Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 112–125.

[8] Federica Cecchetto, Vera Traub, and Rico Zenklusen. 2021. *Bridging the Gap between Tree and Connectivity Augmentation: Unified and Stronger Approaches.* 370–383.

[9] Parinya Chalermsook, Chien-Chung Huang, Danupon Nanongkai, Thatchaphol Saranurak, Pattara Sukprasert, and Sorrachai Yingchareonthawornchai. 2022. Approximating k-edge-connected spanning subgraphs via a near-linear time LP solver. *arXiv preprint arXiv:2205.14978* (2022).

[10] Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. 2009. What would Edmonds do? Augmenting paths and witnesses for degree-bounded MSTs. *Algorithmica* 55, 1 (2009), 157–189.

[11] Chandra Chekuri, Ashish Goel, Sanjeev Khanna, and Amit Kumar. 2004. Multi-processor scheduling to minimize flow time with ε resource augmentation. In *Annual ACM Symposium on Theory of Computing (STOC)*. 363–372.

[12] Joseph Cheriyan and Zhihan Gao. 2018. Approximating (unweighted) tree augmentation via lift-and-project, part I: stemless TAP. *Algorithmica* 80 (2018), 530–559.

[13] J. Cheriyan and R. Thurimella. 2000. Approximating minimum- size k-connected spanning subgraphs via matching. *SIAM J. Comput.* 30 (2000), 528–560.

[14] J. Cheriyan and L. A. Végh. 2014. Approximating Minimum-Cost k-Node Connected Subgraphs via Independence-Free Graphs. *SIAM J. Comput.* 43, 4 (2014), 1342–1362.

[15] Artur Czumaj, Michelangelo Grigni, Papa A Sissokho, and Hairong Zhao. 2004. Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs.. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Vol. 4. 496–505.

[16] Artur Czumaj and Andrzej Lingas. 1999. On Approximability of the Minimum-Cost k-Connected Spanning Subgraph Problem.. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Vol. 99. 281–290.

[17] Michal Dory. 2018. Distributed approximation of minimum k-edge-connected spanning subgraphs. In *ACM Symposium on Principles of Distributed Computing (PODC)*. 149–158.

[18] Samuel Fiorini, Martin Groß, Jochen Könemann, and Laura Sanità. 2018. Approximating weighted tree augmentation via Chvátal-Gomory cuts. In *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 817–831.

[19] Tamás Fleiner and András Frank. 2009. *A quick proof for the cactus representation of mincuts.* Technical Report QP-2009-03. Egerváry Research Group, Budapest. www.cs.elte.hu/egres.

[20] G. N. Fredrickson and Joseph F. JáJá. 1981. Approximation Algorithms for Several Graph Augmentation Problems. *SIAM J. Comput.* 10, 2 (1981), 270–283.

[21] G. N. Fredrickson and Joseph F. JáJá. 1982. On the relationship between the biconnectivity augmentation and traveling salesman problem. *Theoretical Computer Science* 19 (1982), 189 – 201.

[22] H. Gabow. 2005. An improved analysis for approximating the smallest k-edge connected spanning subgraph of a multi-graph. *SIAM Journal on Discrete Math* 19 (2005), 1–18.

[23] H. Gabow and S. Gallagher. 2008. Iterated rounding algorithms for the smallest k-edge connected spanning subgraph. *SIAM J. Comput.* 41 (2008), 61–103.

[24] Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. 2009. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks* 53, 4 (2009), 345–357.

[25] Harold N Gabow, Michel X Goemans, and David P Williamson. 1998. An efficient approximation algorithm for the survivable network design problem. *Mathematical Programming* 82, 1-2 (1998), 13–40.

[26] Michel X. Goemans. 2006. Minimum Bounded Degree Spanning Trees. In *Symposium on Foundations of Computer Science (FOCS)*. 273–282.

[27] Michel X. Goemans and Dimitris Bertsimas. 1993. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming* 60 (1993), 145–166.

[28] F. Grandoni, A. Jabal Ameli, and V. Traub. 2022. Breaching the 2-Approximation Barrier for the Forest Augmentation Problem. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 1598–1611.

[29] F. Grandoni, C. Kalaitzis, and R. Zenklusen. 2018. Improved approximation for tree augmentation: Saving by rewiring. In *Annual ACM Symposium on Theory of Computing (STOC)*. 632–645.

[30] M. Grötschel, L. Lovász, and A. Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 2 (1981), 169–197.

[31] D. Hyatt-Denesik, A. Jabal Ameli, and Sanità L. 2023. Finding Almost Tight Witness Trees. In *International Colloquium on Automata, Languages and Programming (ICALP)*. 79:1–79:16.

[32] Jennifer Iglesias and R Ravi. 2017. Coloring Down: 3/2-approximation for special cases of the weighted tree augmentation problem. *arXiv preprint arXiv:1707.05240* (2017).

[33] Patrick Jaillet and Michael R Wagner. 2008. Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses. *Operations Research* 56, 3 (2008), 745–757.

[34] Kamal Jain. 2001. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. *Combinatorica* 21 (2001), 39–60.

[35] D. Karger. 1999. Random sampling in cut, flow, and network design problems. *Math OR* 24 (1999), 383–413.

[36] Anna Karlin, Nathan Klein, and Shayan Oveis Gharan. 2022. A (Slightly) Improved Bound on the Integrality Gap of the Subtour LP for TSP. In *Symposium on Foundations of Computer Science (FOCS)*. 844–855.

[37] Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. 2021. A (Slightly) Improved Approximation Algorithm for Metric TSP. In *Annual ACM Symposium on Theory of Computing (STOC)*. ACM.

[38] Anna R Karlin, Nathan Klein, Shayan Oveis Gharan, and Xinzhi Zhang. 2022. An improved approximation algorithm for the minimum k-edge connected multi-subgraph problem. In *Annual ACM Symposium on Theory of Computing (STOC)*. 1612–1620.

[39] S. Khuller and B. Raghavachari. 1996. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms* 21 (1996), 434–450.

[40] Samir Khuller and Uzi Vishkin. 1994. Biconnectivity approximations and graph carvings. *Journal of the ACM (JACM)* 41, 2 (1994), 214–235.

[41] Jochen Könemann and Ramamoorthi Ravi. 2000. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Annual ACM Symposium on Theory of Computing (STOC)*. 537–546.

[42] Jochen Könemann and Ramamoorthi Ravi. 2003. Primal-dual meets local search: approximating MST's with nonuniform degree bounds. In *Annual ACM Symposium on Theory of Computing (STOC)*. 389–395.

[43] Bundit Laekhanukit, Shayan Oveis Gharan, and Mohit Singh. 2012. A Rounding by Sampling Approach to the Minimum Size k-Arc Connected Subgraph Problem. In *International Colloquium on Automata, Languages and Programming (ICALP)*. 606–616.

[44] Lap-Chi Lau, R. Ravi, and Mohit Singh. 2011. *Iterative Methods in Combinatorial Optimization* (1st ed.). Cambridge University Press, New York, NY, USA.

[45] C. St. J. A. Nash-Williams. 1961. Edge disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society* 36 (1961), 445–45.

[46] Z. Nutov. 2014. Approximating Minimum-Cost Edge-Covers of Crossing Biset-Families. *Combinatorica* 43, 1 (2014), 95–113.

[47] Z. Nutov. 2017. On the tree augmentation problem. In *Annual European Symposium on Algorithms (ESA)*. 61:1–61:14.

[48] Z. Nutov. 2022. A $4 + \epsilon$ approximation for k-connected subgraphs. *J. Comput. System Sci.* 123 (2022), 64–75.

[49] Cynthia A Phillips, Cliff Stein, Eric Torng, and Joel Wein. 1997. Optimal time-critical scheduling via resource augmentation. In *Annual ACM Symposium on Theory of Computing (STOC)*. 140–149.

[50] David Pritchard. 2011. k-Edge-Connectivity: Approximation and LP Relaxation. In *Approximation and Online Algorithms.*

[51] Ramamoorthi Ravi and Mohit Singh. 2006. Delegate and conquer: An LP-based approximation algorithm for minimum degree MSTs. In *International Colloquium on Automata, Languages and Programming (ICALP)*. Springer, 169–180.

[52] Tim Roughgarden. 2020. Resource Augmentation.

[53] András Sebö and Jens Vygen. 2014. Shorter tours by nicer ears: 7/5-Approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica* 34, 5 (2014), 597–629.

[54] Mohit Singh and Lap Chi Lau. 2015. Approximating minimum bounded degree spanning trees to within one of optimal. *Journal of the ACM (JACM)* 62, 1 (2015), 19 pages.

[55] Daniel D. Sleator and Robert E. Tarjan. 1985. Amortized Efficiency of List Update and Paging Rules. *Commun. ACM* 28, 2 (feb 1985), 202–208.

[56] Vera Traub and Rico Zenklusen. 2021. A Better-Than-2 Approximation for Weighted Tree Augmentation. (2021).

[57] Vera Traub and Rico Zenklusen. 2022. Local search for weighted tree augmentation and Steiner tree. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 3253–3272.

[58] V. Traub and R. Zenklusen. 2023. A $(1.5 + \varepsilon)$-Approximation Algorithm for Weighted Connectivity Augmentation. In *Annual ACM Symposium on Theory of Computing (STOC)*.

Received 11-NOV-2023; accepted 2024-02-11