An $O(\log^2 k)$-Approximation Algorithm for the k-Vertex Connected Spanning Subgraph Problem[*]

Jittat Fakcharoenphol[†] Bundit Laekhanukit [‡]

Abstract

We present an $O(\log^2 k)$-approximation algorithm for the problem of finding a k-vertex connected spanning subgraph of minimum cost, where n is the number of vertices in an input graph, and k is a connectivity requirement. Our algorithm is the first that achieves a polylogarithmic approximation ratio for all values of k and n, and it works for both directed and undirected graphs.

As in previous works, we use the Frank-Tardos algorithm for finding k-outconnected subgraphs as a subroutine. However, with our structural lemmas, we are able to show that we need only partial solutions returned by the Frank-Tardos algorithm; thus, we can avoid paying the whole cost of an optimal solution every time the algorithm is applied.

1 Introduction

In the *minimum-cost k-vertex connected spanning subgraph* problem (k-VCSS), we are given a graph (or a directed graph) $G = (V, E)$ with a nonnegative cost c_e on each edge $e \in E$ and a connectivity requirement k. The goal is to find a min-cost spanning subgraph $G' = (V, E') \subseteq G$ such that G' is k-vertex connected. By a *k-vertex connected* graph, we mean a graph with at least $k + 1$ vertices such that removing any $k-1$ vertices leaves a connected graph. Throughout, we denote the number of vertices and edges of an input graph by $n = |V|$ and $m = |E|$.

The minimum-cost k-vertex connected spanning subgraph problem is one of the important network design problems, and a lot of work has been done to study this problem, in general settings [41, 42, 30, 31, 7, 15, 37] and in more restricted settings [30, 27, 6]. For the case $k = 1$, the problem on undirected graphs is the well-known *minimum spanning tree* problem; thus, this special case, 1-VCSS, is polynomial-time solvable. However, when $k \geq 2$, the problem becomes NP-hard since it includes the *traveling salesman* problem as a special case due to the work of Eswaran and Tarjan [14]. A similar hardness result was given by Czumaj and Lingas [12] for the case that the cost of each edge is either 1 or 2. Indeed, it can be seen that, in metric-cost graphs, 2-VCSS and the traveling salesman problem share an optimal solution. Even for the problem of augmenting the vertex-connectivity of a graph from 1 to 2, Frederickson and JáJá [21] showed that the problem is

[*]The preliminary version of this paper has appeared in the *Proceedings of the 40th ACM Symposium on the Theory of Computing (STOC)*, 2008, pp.153-158

[†]Department of Computer Engineering, Kasetsart University, 50 Ngam Wong Wan Rd, Ladyaow, Chatuchak, Bangkok, 10900 (jittat@gmail.com)

[‡]School of Computer Science, McGill University, 3480 University Street, Montreal, Quebec, Canada, H3A 2A7 (blaekh@cs.mcgill.ca). The work was done while the second author was at Kasetsart University, Kamphaeng Saen Campus.

NP-hard by a reduction from the 3-*dimensional matching* problem. Kortsarz, Krauthgamer and Lee [29] extended the result of Frederickson and JáJá to the general case of the *vertex-connectivity augmentation* problem. As a consequence, it is NP-hard to approximate k-VCSS to within a factor of $1 + \epsilon$, for some fixed $\epsilon > 0$. For the case of directed graphs, the problem is NP-hard even for $k = 1$ because this special case is indeed the *minimum-cost strongly connected spanning subdigraph* problem.

On the positive side, Frederickson and JáJá gave a 3-approximation algorithm for 2-VCSS, and the approximation ratio was subsequently improved to $2 + 1/n$ by Khuller and Raghavachari [27]. Later on, for $k = 2, 3$, Auletta, Dinitz, Nutov and Parente [1] gave a 2-approximation for k-VCSS, which is obtained by an application of the Frank-Tardos algorithm for the *min-cost k-outconnected spanning subgraph* problem. For $k = 4, 5$, Dinitz and Nutov [13] gave a 3-approximation algorithm for the problem. In higher connectivity settings, one can obtain an $O(k)$-approximation algorithm for k-VCSS by solving k instances of the min-cost k-outconnected spanning subgraph problem. An approximation ratio of $O(\log k)$ was first claimed by Ravi and Williamson [41]; however, it was found to have a very subtle flaw [42]. For the case $k \leq \sqrt{n/2}$, Cheriyan, Vempala, and Vetta [7] gave an $O(\log k)$-approximation algorithm for the case of undirected graphs. Their algorithm is based on the property of 3-*critically k-connected* graphs due to Mader [34, 35]. An approximation ratio of $O(\log k \cdot \min\{\sqrt{n}, \frac{n}{n-k} \log k\})$ was achieved by by Kortsarz and Nutov [31]; their algorithm works for both directed and undirected graphs. For metric-cost graphs, the best approximation ratio are $2 + (k - 1)/n$ for undirected graphs and $2 + k/n$ for directed graphs [30]. For unit-cost graphs, a $(1 + 1/k)$-approximation algorithm was devised by Cheriyan and Thurimella [6] for both directed and undirected graphs. For the case of Euclidean graphs, Czumaj and Lingas [12] designed PTAS for k-VCSS and gave both a randomized algorithm and its derandomized version.

The general approach for k-VCSS is to start with an empty subgraph and iteratively increase its connectivity. There are various techniques for increasing vertex-connectivity. One related notion of connectivity is outconnectivity (to be defined later). The cost of a min-cost k-vertex outconnected subgraph is a lower bound on the optimum value of k-VCSS. Therefore, the Frank-Tardos algorithm [19] for finding a min-cost k-vertex outconnected subgraph has been used, starting with Khuller and Raghavachari [27], as a subroutine for approximating k-VCSS.

In a line of research [30, 7, 31], there were attempts to minimize the number of calls to the Frank-Tardos algorithm as this would give a better performance ratio. In an important work, Cheriyan et al. [7] observed that if an undirected graph is not a ρ-critically k-connected graph, then it is enough to apply the Frank-Tardos algorithm from ρ roots to increase the connectivity of a graph by one. This leads to an $O(\log k)$-approximation algorithm for the case that $k \leq \sqrt{n/2}$ as it was shown by Mader [35] that every 3-critically k-connected graph has at most $(2k-1)k$ vertices. (In fact, Cheriyan et al. used an old upper bound of $6k^2$ [34].)

Kortsarz and Nutov [31] further explored this approach. They considered a set of ℓ-fragments which are, intuitively and informally, subsets of vertices with ℓ neighbors; these fragments are those that need to be covered in order to ensure that the connectivity of the graph is at least $\ell + 1$. With a set-cover type analysis, they obtained the bound of $O(\frac{n}{n-\ell} \cdot \log \ell)$ on the number of roots required for applying the Frank-Tardos algorithm and thus obtain an $O(\frac{n}{n-k} \log^2 k)$-approximation algorithm. They also gave another $O(\sqrt{n} \log k)$-approximation algorithm based on Ravi and Williamson's primal-dual algorithm [41, 42].

In this paper, we show that it is not necessary to minimize the number of calls to the Frank-Tardos algorithm; instead, it suffices to make sure that the cost for each call is small. Our notion of progress is the number of *cores*, inclusionwise minimal ℓ-fragments, which has been used by Kortsarz and Nutov [31]. Similar to their algorithm, we decrease the number of cores in a subgraph via the Frank-Tardos algorithm[1], but we do not take the whole solution. We show that we can use only a partial solution from the Frank-Tardos algorithm, thus paying only $O(1/t)$ fraction of the usual cost[2], where t is the number of cores in a current graph. Hence, we pay a factor of $O(\log t)$ for the augmentation problem. As we have to apply the algorithm k times to increase the connectivity of a graph to k, we pay an addition factor of $O(\log k)$. This is due to an analysis based on the LP-scaling technique, which has been used in edge-connectivity problems; see [43] and [24] for more detail. Thus, the approximation guarantee of our algorithm is $O(\log n \log k)$ because we may have n cores at the start. To get an approximation guarantee of $O(\log^2 k)$, the final blow is to apply an algorithm that works well for the case of small k. In particular, Cheriyan et al.'s algorithm and Kortsarz and Nutov's give approximation guarantees of $O(\log k)$ and $O(\log^2 k)$, respectively, for the case that $k < \mathrm{poly}(n)$. Alternatively, we give a preprocessing step that reduces the number of cores to k; this technique was used in Laekhanukit's algorithm [33] for the *min-cost subset k-connected subgraph* problem, a generalization of k-VCSS. Therefore, we have an approximation guarantee of $O(\log^2 k)$ for all values of k and n.

As a side product, our procedure can be used to approximate a min-cost augmenting set (a set of edges whose addition increases the vertex-connectivity of a graph by at least one) to within a factor of $O(\log k)$ of an optimal. In particular, our algorithm implies an $O(\log k)$-approximation algorithm for the problem of increasing the vertex-connectivity of a graph from k to $k + 1$.

We remark that, at the time this paper was written, Nutov [37] already improved the approximation ratio of k-VCSS to $O(\log \frac{n}{n-k} \log k)$. Nutov's algorithm was built upon our algorithm with an addition of a preprocessing step that reduces the number of cores to $O(\frac{n}{n-k})$.

1.1 Organization

In Section 2, we give formal definitions and state some basic lemmas. In Section 3, we present the algorithm and its analysis. In Section 4, we present and analyze the procedure PartialAugment, which is our main ingredient. In Section 5, we present the subroutine required by the procedure PartialAugment. In the last section, Section 6, we present a preprocessing step that reduces the number of cores to k.

1.2 More related work

While we focus on the uniform vertex-connectivity problem, many works have been done to study the more general problems and edge-connectivity problems. The subset k-connectivity problem is the generalization of k-VCSS where Steiner vertices are allowed; that is, we are given a set of terminals $T \subseteq V$, and the goal is to find a min-cost subgraph G' such that T is k-connected in G'. This problem is strictly harder than k-VCSS because Kortsarz, Krauthgamer and Lee showed that it cannot be approximated to within a factor of $O(2^{\log^{1-\epsilon} n})$, for all fixed $0 < \epsilon < 1$, unless $\mathrm{NP} \subseteq \mathrm{DTIME}(n^{\mathrm{polylog}(n)})$. This problem has been studied in [3, 38, 33, 39]. Most approximation

[1]Since we consider the problem of increasing the connectivity of a graph by one, we may replace the Frank-Tardos algorithm by the Frank algorithm [17] for increasing the outconnectivity of a graph by one; also, Frank [18] recently gave a combinatorial algorithm for the k-outconnected spanning subgraph problem.

[2]By an LP-scaling based analysis, it can be shown that the *fractional* cost of increasing the vertex-connectivity from ℓ to $\ell + 1$ is at most $1/(k - \ell)$ times the optimal cost of k-VCSS.

algorithms for this problem are done by solving the rooted problem k times. Laekhanukit [33] showed that if $k \leq 2|T|$, then we only need to solve $O(\log k)$ instances of the rooted problem. Specifically, generalized Kortsarz and Nutov's algorithm for k-VCSS to the subset k-connectivity problem. Recently, Nutov gave an approximation ratio of $O(k \log k)$ for this problem, where $|T| \geq k + \Omega(k)$. However, for $|T| \leq k$, the best known approximation ratio is $|T|^2$ as no algorithms can beat a trivial approximation algorithm. For metric-cost graphs, the problem is easier than the general case because there is an $O(1)$-approximation algorithm due to Cheriyan and Vetta [9]. The generalization of the k-outconnected subgraph problem is the *rooted subset k-connectivity* problem, where we want to connect a given root vertex to a set of terminals by k-internally disjoint paths. A lot of attentions have been drawn to this problem [3, 10, 4, 5, 11, 38], culminating in an $O(k \log k)$-approximation algorithm by Nutov [38]. However, similar to the subset k-connectivity problem, for $|T| \leq k$, the best known is a trivial $|T|$-approximation algorithm. The most general vertex-connectivity problem is the vertex-connectivity survivable network design problem, where we are allowed that have an arbitrary requirement for each pair of vertices. Although it is the hardest problem among vertex-connectivity problems, an $O(k^3 \log |T|)$-approximation algorithm due to Chuzhoy and Khanna [11] is surprisingly simple.

The *k-edge connected spanning subgraph* problem (k-ECSS) is a variant of k-VCSS, where we are asked to find a min-cost k-edge connected subgraph. A 2-approximation algorithm for k-ECSS was proposed by Khuller and Vishkin [28]. For the case that a given graph has unit-costs, Cheriyan and Thurimella [6] gave a $(1 + 2/(k+1))$-approximation algorithm, thus showing that the problem is easier as k increases. This special case has been subsequently studied in [23, 22], and the best approximation ratio is $1 + 1/(2k) + O(1/k^2)$ due to the work of Gabow and Gallagher [22]. This ratio is almost tight as Goemans, Tardos and Williamson [23] showed that k-ECSS on graphs with unit-costs cannot be approximated to within a factor of $1 + O(1)/k$ unless P=NP. The gap between unit-cost and general-cost versions of k-ECSS was found by Pritchard [40]; he showed that it is NP-hard to approximate the general case of k-ECSS to within a factor of $1 + \epsilon$, for some fixed $\epsilon > 0$. For the general version of edge-connectivity problems, a 2-approximation algorithm was achieved by a novel iterated rounding algorithm proposed by Jain [25]. Jain's algorithm was extended to the element-connectivity problem by Fleischer, Jain and Williamson [16] and also by Cheriyan, Vempala and Vetta [8]. For a more restricted version, the *Steiner tree* problem, an approximation ratio better than 2 is known. The first algorithm that breaks a factor two barrier is due to Zelikovsky [44], and the best known approximation ratio of 1.39 was recently given by Byrka, Grandoni, Rothvoß and Sanità [2].

For more detail, we refer readers to a comprehensive survey by Kortsarz and Nutov [32].

2 Preliminaries

Let $G = (V, E)$ denote an input graph with a nonnegative cost c_e on each edge $e \in E$. A *k-separator* is a subset of vertices $S \subseteq V$ such that $|S| = k$ and $G - S$ is disconnected. We say that G is *k-vertex connected* or *k-connected* if G is a complete graph on $k+1$ vertices, or G has at least $k+1$ vertices and has no $(k-1)$-separator; thus, $G - X$ is connected, for all subsets of vertices $X \subset V$ with $|X| < k$. The *connectivity* of G, denoted by $\kappa(G)$, is the maximum integer ℓ such that G is ℓ-connected. We can check the connectivity of a graph in polynomial time. Thus, we may assume that the input graph G is k-vertex connected; otherwise, there would be no feasible solution.

For any subset of vertices $X \subseteq V$, we denote by $N_G(X)$ the set of neighbors of X in G; that is, $N_G(X) = \{v \in V : u \in X \text{ and } (u,v) \in E\}$. The *vertex complement of* X is denoted by $X^* = V - (X \cup N_G(X))$. If G is clear from the context, then we will omit the subscript G. We say that X is an *ℓ-fragment* in G if $|N_G(X)| = \ell$ and $X, X^* \neq \emptyset$. An ℓ-fragment is a certificate that the graph G is not $(\ell+1)$-connected. It can be seen that the vertex complement X^* of an ℓ-fragment X is also an ℓ-fragment; we call this ℓ-fragment X^* the *complementary ℓ-fragment* of X.

Later on, we will work on an ℓ-connected subgraph H of G; thus, we will use a fragment to mean an ℓ-fragment. A fragment X is called a *small* fragment if $|X| \leq |X^*|$. Observe that for any pair of fragments X, X^*, one of them must be small. A *core* C of H is an inclusionwise minimal small fragment; that is, C does not properly contain any small fragment. We denote by $\mathbb{C}(H)$ the set of all cores in H and denote by $t(H)$ the size of $\mathbb{C}(H)$. For any core C, let A_C be a union of all small fragments that contain only one core C. Thus,

$$A_C = \bigcup \{X \subseteq V : X \text{ is a small fragment }, C \subseteq X, D \not\subseteq X \text{ for all cores } D \neq C\}.$$

The set of all sets A_C of G is denoted by $\mathbb{A}(G) = \{A_C : C \in \mathbb{C}(G)\}$.

The definitions for directed graphs are similar. However, each subset of vertices has both *out-neighbors* and *in-neighbors*. Thus, we have both *out-fragments* and *in-fragments*. To be precise, consider two disjoint non-empty subsets of vertices $X^+, X^- \subseteq V$. We say that X^+ is an *ℓ-out-fragment* and X^- is an *ℓ-in-fragment* if $|V - (X^+ \cup X^-)| = \ell$ and there is no arc going from X^+ to X^-. The terms *out-cores* and *in-cores* are defined analogously.

We now list a few basic properties, most of which are proved by Jordan [26] and Kortsarz and Nutov [31].

Proposition 1 ([26],[31]). *Consider an ℓ-connected (directed or undirected) graph G on n vertices. Let X and Y be ℓ-fragments such that $X \cap Y \neq \emptyset$. If $n - |X \cup Y| \geq \ell$, then $X \cap Y$ is an ℓ-fragment, and if a strict inequality holds, then $X \cup Y$ is also an ℓ-fragment. In particular, the non-empty intersection of two small ℓ-fragments is also a small ℓ-fragment.*

Proof. We prove the proposition for the case of undirected graphs, which follows from the submodularity of the function $|N_G(.)|$; that is,

$$|N_G(U)| + |N_G(W)| \geq |N_G(U \cup W)| + |N_G(U \cap W)| \quad \text{for every } U, W \subseteq V.$$

Since $X \cap Y \neq \emptyset$ and G is ℓ-connected, we have $|N_G(X \cap Y)| \geq \ell$. Since X and Y are ℓ-fragments, $|N_G(X)| = |N_G(Y)| = \ell$. Thus, $|N_G(X \cup Y)| \leq \ell$ by the submodularity of $|N_G(.)|$.

Next, we claim that if $(X \cup Y)^*$ is not empty, then $|N_G(X \cup Y)| = |N_G(X \cap Y)| = \ell$; that is, both $X \cup Y$ and $X \cap Y$ are ℓ-fragments. To see this, suppose $(X \cup Y)^*$ is not empty. Since G is ℓ-connected, we have $|N_G(X \cup Y)| \geq \ell$ and $|N_G(X \cap Y)| \geq \ell$; otherwise, we would have a separator of size less than ℓ separating $X \cap Y$ and $(X \cup Y)^*$. It then follows that

$$|N_G(X)| + |N_G(Y)| = 2\ell \geq |N_G(X \cup Y)| + |N_G(X \cap Y)| \geq 2\ell$$

Thus, $|N_G(X \cup Y)| = |N_G(X \cap Y)| = \ell$.

If $n - |X \cup Y| > \ell$, then $(X \cup Y)^*$ is not empty because $|N_G(X \cup Y)| \leq \ell$. Thus, $|N_G(X \cup Y)| = |N_G(X \cap Y)| = \ell$. If $n - |X \cap Y| = \ell$, then $|N_G(X \cup Y)| = \ell$; otherwise, $(X \cup Y)^*$ is not empty, and we would have a separator of size less than ℓ. Thus, $|N_G(X \cup Y)| = |N_G(X \cap Y)| = \ell$ by the submodularity of $|N_G(.)|$. Therefore, $X \cap Y$ is an ℓ-fragment by definition, and if $n - |X \cup Y| > \ell$, then $X \cup Y$ is also an ℓ-fragment, proving the first two statements of the proposition. The last statement then follows immediately since $X \cap Y$ has size at most $\min\{|X|, |Y|\}$.

The proof for the case of directed graphs is the same but with $N_G(U)$ redefined as the set of out-neighbors of $U \subseteq V$. □

This proposition implies that no two cores intersect; thus, the set of all cores is pairwise disjoint. By the definition of a core together with Proposition 1, we have the following proposition.

Proposition 2. *Any small ℓ-fragment F in an ℓ-connected graph G contains at least one core.*

Proof. This proposition follows directly from the definition of a core. If F is a core itself, then we are done. Otherwise, F must contain another small fragment. Take a smallest fragment X contained in F. Then X does not properly contain other small fragments and thus is a core by definition. □

The following proposition, which is a consequence of the above propositions, was proved by Kortsarz and Nutov [31].

Proposition 3 ([31])**.** *For an ℓ-connected graph G, the sets in $\mathbb{A}(G)$ are pairwise disjoint.*

3 The Algorithm

In this section, we describe and analyze the performance of our algorithm.

The algorithm starts with an empty subgraph G_0 of G. It then proceeds in k rounds. In round ℓ, starting from round 0, it augments G_ℓ resulting in $G_{\ell+1}$ whose vertex-connectivity is $\ell + 1$ with a procedure described in Subsection 4.

Lying at the heart of our augmenting procedure is the subroutine PartialAugment. A set of edges F is an *augmenting* set of H if $\kappa(H \cup F) > \kappa(H)$. We define a *partial augmenting set* to be a set of edges A such that $t(H \cup A) < t(H)$. In other words, adding the set of edges A to an ℓ-connected graph H reduces the number cores in H by at least one (or increases the connectivity of a graph by one if $t(H) = 1$).

Let $opt_k = opt_k(G)$ be the cost of an optimal k-vertex connected spanning subgraph of G. We prove in Section 4 the following lemma.

Lemma 1. *Given an ℓ-connected subgraph H of G with $t = t(H)$ cores,* PartialAugment *finds a partial augmenting set for H of cost at most*

$$O\left(\frac{1}{t} \cdot \frac{1}{k-\ell}\right) \cdot opt_k.$$

With PartialAugment, our augmenting procedure is straight-forward; we repeatedly call PartialAugment until the resulting subgraph is $(\ell + 1)$-connected.

Lemma 1 implies the following theorem.

Theorem 1. *There is an $O(\log n \log k)$-approximation algorithm for the minimum-cost k-vertex connected subgraph problem, which runs in polynomial time.*

Proof. First, we show that for each round ℓ, the augmenting cost is at most $O(\frac{\log n}{k-\ell}) \cdot opt_k$. Let $G_\ell = H_0, H_1, \ldots, H_p = G_{\ell+1}$ denote a sequence of subgraphs produced by PartialAugment. From Lemma 1, the cost of augmenting H_i to H_{i-1} is at most

$$O\left(\frac{1}{t(H_{i-1})} \cdot \frac{1}{k-\ell}\right) \cdot opt_k.$$

Summing them up, we have that the augmenting cost at round ℓ is at most

$$\begin{aligned} O\left(\left(\sum_{i=0}^{p-1} \frac{1}{t(H_i)}\right)\left(\frac{1}{k-\ell}\right)\right) opt_k &\leq \left(\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1}\right) O\left(\frac{1}{k-\ell}\right) opt_k \\ &= O\left(\frac{\log n}{k-\ell}\right) opt_k. \end{aligned}$$

The second inequality follows because the number of cores decreases by at least one in each step, and the number of cores in H_0 is at most n. Hence, the total cost is

$$\sum_{\ell=0}^{k-1} O\left(\frac{\log n}{k-\ell}\right) opt_k = O(\log n \log k) \cdot opt_k.$$

□

Our algorithm gives an approximation guarantee of $O(\log n \log k)$, which is $O(\log^2 k)$ for $k = \Omega(n)$. To obtain a guarantee of $O(\log^2 k)$ for the case $k = o(n)$, we apply an $O(\frac{n}{n-k} \log^2 k)$-approximation algorithm of Kortsarz and Nutov [31]. In particular, we apply our algorithm when $k \geq 2n$ and apply Kortsarz and Nutov algorithm otherwise. Thus, we have a final guarantee of $O(\log^2 k)$ for all values of n and k. An alternative method is to reduce the number of cores to k by a preprocessing step given in Section 6.

Regarding the cost of an augmenting set, the proof above shows that one can augment an ℓ-connected subgraph to be an $(\ell+1)$-connected subgraph by adding edges of cost $O(\frac{1}{k-\ell} \log n \cdot opt_k)$.

4 Partially augmenting a graph

In this section, we describe the procedure PartialAugment that, given an ℓ-connected subgraph H of G, finds a partial augmenting set of small cost. We rely on the fact that, for each core C in H, there exists a subroutine that finds a set A_C. (Recall that A_C is the union of all small fragments that contain only one core C.) For the sake of the presentation flow, we describe the subroutine separately in Section 5. Later in this section, we describe the algorithm. In Subsection 4.1, we prove the correctness of the procedure PartialAugment. Then we show its performance guarantee in Subsection 4.2. We will focus on the undirected case. The discussion on the directed case will be given later in Subsection 4.3.

The procedure PartialAugment finds, for each core C, a partial augmenting set F_C. Then it returns the set F_C whose cost is minimum over all F_C's. To find F_C, it applies as a subroutine the Frank-Tardos algorithm [19], which (approximately) finds a min-cost subgraph that is $(\ell+1)$-outconnected from a root vertex r in C.

We shall give a precise definition of a k-outconnected subgraph. A directed graph $G = (V, E)$ is *k-outconnected from a root vertex* r if G has *k-internally disjoint paths* from r to each vertex in

$V - \{r\}$. Similarly, G is *k-inconnected to a root vertex r* if G has k-internally disjoint paths to r from each vertex in $V - \{r\}$. Since these two notions are the same in undirected graph, we will use only the term k-outconnected when considering undirected graphs.

In the *min-cost k-outconnected spanning subgraph* (k-outconnectivity) problem, we are given an undirected or directed graph $G = (V, E)$, a root vertex r and a connectivity requirement k. The goal is to find a min-cost spanning subgraph of G that is k-outconnected from r. If an input graph is directed, Frank and Tardos [19] (and also Frank [18]) showed that a natural linear programming relaxation for the k-outconnectivity problem is *totally dual integral*; thus, the ellipsoid method gives an optimal integral solution. As this was proved by Frank and Tardos, we refer to the algorithm for the k-outconnectivity problem as the Frank-Tardos algorithm. For the case of undirected graphs, we replace each edge (u, v) by two arcs (u, v) and (v, u) with the same cost as the edge (u, v) and then solve the k-outconnectivity problem in the resulting directed graph. Thus, the Frank-Tardos algorithm gives a 2-approximate solution to the linear program of the k-outconnectivity problem in undirected graphs.

Now, we show how to find an augmenting set F_C, where C is a given core. Note that here an input graph is undirected. The case of directed graphs will be discussed later in Subsection 4.3. First, we construct a graph G_C by setting to "zero" the costs of all edges in $G \cap H$ and all edges whose endpoints are not in A_C. We pick an arbitrary vertex $u \in C$ and then apply the Frank-Tardos algorithm to find an $(\ell + 1)$-outconnected spanning subgraph I_C of G_C rooted at u. The augmenting set F_C is obtained by picking all edges in I_C with at least one endpoint in A_C.

The procedure PartialAugment is presented in Figure 1.

Algorithm 1 PartialAugment

```
for all core $X$ in $\mathbb{C}(H)$ do
    • Construct a graph $G_X$ from $G$ by setting to zero the costs of edges in $H$ and all edges whose endpoints are not in $A_X$.
    • Choose an arbitrary vertex $u$ in $X$.
    • Apply the Frank-Tardos algorithm to find an $(\ell + 1)$-outconnected spanning subgraph $I_X$ of $G_X$ rooted at $u$.
    • Let $F_X = \{(u, v) \in I_X : \{u, v\} \cap A_C \neq \emptyset\}$.
end for
return $F_C$ whose cost is minimum over all cores $X \in \mathbb{C}(H)$.
```

4.1 Correctness of PartialAugment

In this section, we prove the correctness of PartialAugment; that is, we show that the set F_C that it returns is indeed a partial augmenting set. More generally, we show that, for any core C, the set F_C computed by PartialAugment is a partial augmenting set.

The following lemma uses various facts listed in Section 2.

Lemma 2. *Let H be an ℓ-connected graph with t cores. Let C be a core in H. Then the number of cores in $H \cup F_C$ is at most $t - 1$.*

Proof. Let $H' = H \cup F_C$, $\mathbb{C} = \mathbb{C}(H)$ and $\mathbb{C}' = \mathbb{C}(H')$.

Note that every small fragment X in H' is also a small fragment in H since $|N_H(X)| \leq |N_{H'}(X)| \leq \ell$. This means that any core in H' is a fragment in H.

For the sake of contradiction, assume that $|\mathbb{C}'| \geq t$.

We first show that $|\mathbb{C}'|$ cannot be greater than t. Since any core in H' is a small fragment in H, and any small fragment contains at least one core (from Proposition 2), we have that any core $X' \in \mathbb{C}'$ contains some core $X \in \mathbb{C}$. By the Pigeon Hole Principle, if $|\mathbb{C}'| > t$, then there are $X', Y' \in \mathbb{C}'$ that contain the same core $X \in \mathbb{C}$. By proposition 1, cores are pairwise disjoint, a contradiction since $X' \cap Y' \neq \emptyset$ (because $\emptyset \neq X \subseteq X' \cap Y'$).

Next, consider the case that $|\mathbb{C}'| = t$. Using the previous arguments, we have that exactly one core $D \in \mathbb{C}'$ contains C, and D contains no other cores in $\mathbb{C}$. The Frank-Tardos algorithm outputs a subgraph I_C such that $N_{I_C}(X) \geq \ell + 1$, for any set of vertices X with $|X| < n - \ell - 1$. Thus, $N_{H \cup I_C}(D) \geq \ell + 1$. Clearly, D is contained in A_C. This means that every edge in I_C with at least one endpoint in D is also in F_C. So, we conclude that $N_{H'}(D) = N_{H \cup I_C}(D) \geq \ell + 1$. This contradicts the assumption that D is a core in H', and the lemma follows. □

Note that this lemma also implies that the number of cores in $H \cup I_C$ is at most $t - 1$ as well since $F_C \subseteq I_C$.

4.2 The cost for PartialAugment

Let $OPT_k = (V, E_k)$ denote an optimal solution to the k-vertex connected spanning subgraph problem, and let opt_k denote its cost. We compare the cost of our solution to the cost of an optimal solution of the following linear program for the min-cost k-connected spanning subgraph problem introduced in [20] and used in [7, 31].

$$LP(k) \begin{cases} \min & \sum_{e \in E} c_e x_e \\ \text{s.t.} & \sum_{e \in \delta_E(S,T)} x_e \geq k - (n - |S \cup T|) \quad \begin{array}{l} \forall \emptyset \neq S, T \subset V, \\ S \cap T = \emptyset \end{array} \\ & 0 \leq x_e \leq 1 \qquad\qquad \forall e \in E, \end{cases}$$

where $\delta_E(S,T) = \{(u,v) \in E : u \in S, v \in T\}$. We call this linear program $LP(k)$ and denote its optimal cost by Z. Since this is a relaxation, we have $Z \leq opt_k$.

As our algorithm iteratively adding edges to a solution subgraph, denoted by $H = (V, E_h)$, we may define an instance $\Pi(\ell + 1)$ of $LP(\ell + 1)$ by assigning zero costs to all edges in H. We can construct a solution $\mathbf{x}$ for $\Pi(\ell + 1)$ from OPT_k by assigning

$$\mathbf{x}_e = \begin{cases} 1 & \text{if } e \in E_h, \\ 1/(k - \ell) & \text{if } e \in E_k - E_h, \\ 0 & \text{otherwise.} \end{cases}$$

It is known (see, e.g., [7]) that $\mathbf{x}$ is feasible and has cost at most $opt_k/(k - \ell)$. We include the proof for completeness.

Lemma 3. *There is a feasible solution to $\Pi(\ell + 1)$ with cost at most $\frac{1}{k-\ell} opt_k$.*

Proof. First, we show that $\mathbf{x}$ satisfies all the constraints of $\Pi(\ell + 1)$. Let S and T be arbitrary non-empty sets of vertices such that S and T are disjoint. Consider the constraint

$$\sum_{e \in \delta_E(S,T)} \mathbf{x}_e \geq (\ell + 1) - (n - |S \cup T|)$$

Note that $\delta_E(S,T) = \delta_{E_h}(S,T) \cup \delta_{E_k - E_h}(S,T)$. Let q denote the number of edges in $\delta_{E_h}(S,T)$, and let r denote the number of edges in $\delta_{E_k - E_h}(S,T)$. Recall that $\mathbf{x}_e = 1$ for all edges $e \in H$, and $\mathbf{x}_e = 1/(k-\ell)$ for all edges $e \in E_k - E_h$. Since H is ℓ-connected, we have $r \geq \ell - (n - |S \cup T|)$. If $r > \ell - (n - |S \cup T|)$, then the constraint is satisfied because r is integer. So, let consider the case that $r = \ell - (n - |S \cup T|)$. Since OPT_k is k-connected, $|\delta_{E_k}(S,T)| \geq k - (n - |S \cup T|)$. So, we have $q \geq k - (n - |S \cup T|) - r = k - \ell$, which implies that

$$\sum_{e \in \delta_E(S,T)} \mathbf{x}_e = r + \frac{q}{k-\ell} \geq r + 1 = (\ell+1) - (n - |S \cup T|).$$

Thus, $\mathbf{x}$ is a feasible solution to $\Pi(\ell+1)$.

We now consider the cost of $\mathbf{x}$. Since the cost of each edge in H is zero, and all variables of non-zero cost edges are scaled by $\frac{1}{k-\ell}$, the cost of $\mathbf{x}$ is at most $\frac{1}{k-\ell} \cdot opt_k$. This proves the lemma. $\square$

Given an undirected graph $G = (V,E)$ and a root vertex r, the following linear program $\widehat{LP}(G,k,r)$ is a relaxation of the k-outconnected spanning subgraph problem.

$$\widehat{LP}(G,k,r) \begin{cases} \min & \sum_{e \in E} c_e x_e \\ \text{s.t.} & \sum_{e \in \delta_E(S,T)} x_e \geq k - (n - |S \cup T|) \qquad \begin{array}{l} \forall \emptyset \neq S,T \subset V \\ S \cap T = \emptyset, r \in S \end{array} \\ & 0 \leq x_e \leq 1 \qquad \forall e \in E \end{cases}$$

The linear program $\widehat{LP}(G,k,r)$ is similar to $LP(k)$ except that every set S in the constraints contains r. Thus, any feasible solution to $LP(k)$ is also feasible for $\widehat{LP}(G,k,r)$, but the converse is not true. The Frank-Tardos algorithm [19] (or the Frank algorithm [18]) gives a 2-approximate solution for $\widehat{LP}(G,k,r)$. This allows us to bound the cost of a solution obtained by our algorithm in terms of opt_k.

Procedure PartialAugment computes partial augmentations from many roots. Recall that, for each core C, the algorithm first constructs a graph G_C by assigning zero costs to all edges in H and all edges with no endpoints in A_C. It then computes I_C using the Frank-Tardos algorithm. Let F_C denote the set of partial augmenting edges computed by PartialAugment.

We shall bound the cost of F_C to the cost of OPT_k. Let denote by $O_C = \{(u,v) \in OPT_k : \{u,v\} \cap A_c \neq \emptyset\}$ the set of edges in OPT_k with at least one endpoint in A_C. Then we have the following lemma.

Lemma 4. *A solution $\mathbf{x}$ is feasible for $\widehat{LP}(G_C, \ell+1, v)$, where v is any vertex in a core C. Therefore, the cost of F_C is at most*

$$2 \cdot \left(\frac{1}{k-\ell}\right) \sum_{e \in O_C} c_e$$

Proof. First, $\mathbf{x}$ is feasible for $\widehat{LP}(G_C, \ell+1, v)$ because every constraint of $\widehat{LP}(G_C, \ell+1, v)$ is also a constraint of $LP(\ell+1)$ (and $\mathbf{x}$ is feasible for $LP(\ell+1)$).

Next, we show that F_C has cost as claimed. Let u be a root vertex selected for computing I_C. Since $\mathbf{x}$ is feasible for $\widehat{LP}(G_C, \ell+1, u)$ and the Frank-Tardos algorithm finds a 2-approximate solution for this linear program, the cost of I_C is at most the cost of $\mathbf{x}$ in G_C. Because we assign zero costs to all edges with no endpoints in A_C, the cost of $\mathbf{x}$ is at most $2 \cdot \left(\frac{1}{k-\ell}\right) \sum_{e \in O_C} c_e$.

To see that the bound also applies to F_C, we note that the cost of all edges in F_C remains the same as that in the graph G_C. □

Applying Lemma 4, we have the following bound on the total cost of all partial augmenting sets.

Corollary 1.

$$\sum_{C \in \mathbb{C}(G_\ell)} cost(F_C) \leq 4 \cdot \frac{1}{k-\ell} opt_k.$$

Proof. The corollary follows since all sets in $\mathbb{A}(H)$ are pairwise disjoint as asserted in Proposition 3. Therefore, each edge $e \in OPT_k$ contributes at most twice to the right-hand-side. □

The above corollary implies our key lemma, which guarantees the performance of PartialAugment.

Lemma 1. *Given an ℓ-connected subgraph H of G with $t = t(H)$ cores,* PartialAugment *finds a partial augmenting set for H of cost at most*

$$O\left(\frac{1}{t} \cdot \frac{1}{k-\ell}\right) \cdot opt_k.$$

Proof. By Corollary 1, there is at least one core in H such that the cost of F_C is at most $\frac{1}{t} \cdot (\frac{4}{k-\ell} \cdot opt_k)$. The lemma thus follows from the fact that PartialAugment returns the set F_C with minimum cost. □

Note that if we replace the fractional solution $\mathbf{x}$ by the minimum augmenting set, then we have the cost of the optimal augmenting set as the lower bound instead of the term $\frac{1}{k-\ell} \cdot opt_k$. A proof similar to the main theorem shows that one can apply PartialAugment repeatedly to find an augmenting set of cost at most $O(\log n)$ times the optimal.

4.3 Discussion for the case of directed graphs

Here we discuss the case of directed graphs. In this case, even if we have no small out-fragment, we may still have some out-fragment that is *not small*, which means that the graph is not $(\ell+1)$-connected. Thus, we have to work on both small and *large* out-fragments. As we need special properties of small fragments, this would be a problem. Fortunately, the vertex-complement of a large out-fragment must be a small in-fragment by definition. Thus, it suffices to work on two families of small fragments: the family of small out-fragments and the family of small in-fragments. The algorithm is the same as that of undirected graphs with cores replaced by out-cores (in-cores) and small fragments replaced by small out-fragments (in-fragments). In the case of directed graphs, the cost of a partial augmenting set F_C is bounded by $(\frac{1}{k-\ell}) \sum_{e \in O_C} c_e$ because the Frank-Tardos algorithm gives an optimal integral solution in directed graphs. Since the tail of each arc can be in exactly one set A_C, each arc contributes exactly once to the sum $\sum_{C \in \mathbb{C}(G_\ell)} cost(F_C)$. Thus,

$\sum_{C\in\mathbb{C}(G_\ell)} cost(F_C) \le \frac{1}{k-\ell} opt_k$. It then follows from the same proof as that of Lemma 1 that the cost of the set F_C returned by PartialAugment is at most $O(\frac{1}{t}\cdot\frac{1}{k-\ell}\cdot)\cdot opt_k$. Note that we have to work on two families of fragments, but this only incurs a factor of two in the performance guarantee of the main algorithm.

5 Finding a union of all small ℓ-fragments containing only one core C

In this section we describe a subroutine that, given an ℓ-connected graph H and a core C of H, outputs a set A_C. As the procedure does the same for both undirected and directed graphs, we describe it in terms of undirected graphs.

The subroutine, instead of generating all small fragments which share the same core C, determines whether a given vertex $v \in V - C$ is in A_C using a decision procedure described shortly.

Note that, given a graph $H = (V, E)$ and a pair of vertices u and v, one can find a minimal subset X containing u such that $N(X)$ is the minimum vertex cut separating u and v using one max-flow computation (on the vertex capacitated network induced by H'). Therefore, given an ℓ-connected graph $H = (V, E)$ and a vertex $u \in V$, we can find an inclusionwise minimal fragment that contains u by applying at most n max-flow computations from u to all other vertices. In [31], it was shown that this algorithm can be implemented to run in time $O(\ell mn)$.

To determine if a given vertex v is in A_C, one has to find a fragment X that contains both C and v. Let c be an arbitrary vertex in C. Note that if we add an edge e_c from c to v, then the resulting graph will contain no fragment X' which contains C but $v \notin X' \cup N(X')$; otherwise, e_c would violate the fact that $N(X')$ is a separator. However, this does not rule out the possibility that v lies in $N(X')$; therefore, the algorithm that finds the minimal ℓ-fragment discussed above might find X' instead of the required X.

Our approach for finding X is to modify the graph H further to get a graph H' so that v does not lie inside any ℓ-separators separating a small fragment containing C. To do this, we first add to H' an edge e_c from c to v (if such edge is not already in H). Recall that $N_H(v)$ denotes a set of neighbors of v in H. We also add to H' edges joining each pair of vertices in $N_H(v)$.

The decision procedure can be described as follows. First, it finds all cores in H, denoted by $\mathbb{C}(H)$. It then constructs H' and applies a max-flow algorithm to find an inclusionwise minimal fragment X containing c. If X is a small fragment, then X is a core by definition. The procedure accepts v if X exists, contains no other cores in $\mathbb{C}(H)$ and is small.

Lemma 5. *The decision procedure accepts v if and only if v is in A_C.*

Proof. First, we prove the forward direction. Suppose that v is accepted by the procedure. Then the procedure finds a small fragment X of H' that contains v and contains C as a unique core. It can be seen that X is also a small fragment of H. Thus, we have a certificate for the fact that v is in A_C.

Now, we prove the converse. Suppose v is in A_C. Then there exists a minimal small fragment X in H such that X contains both C and v, and X contains no other core distinct from C. To show that the procedure accepts v, it suffices to show that X is a core of H' and thus will be found by the procedure.

For X to be a core of H', X has to be a small fragment in H'. Clearly, both endpoints of e_c are inside X since X contains both v and C (and $c \in C$). Since v is in X, each vertex $u \in N_H(v)$ is

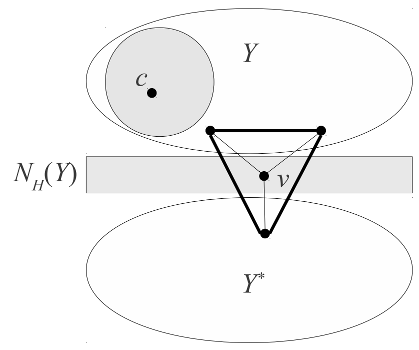

Figure 1: The vertex v has neighbors on both Y and Y^*.

either in $N_H(X)$ or in X. Thus, no edges added to H (to form H') go from X to X^*. This proves that X is a small fragment in H'.

Next, assume that X is not a core of H'. Then there is a small fragment Y properly contained in X. As H' has the edge $e_c = (c, v)$, every fragment containing c cannot have v in its complementary fragment. Thus, only two cases are possible: either (i) Y contains v or (ii) v lies in $N_{H'}(Y)$ where $N_{H'}(Y)$ is the set of neighbors of Y in H'. Case (i) is ruled out because of the minimality of X.

We now consider case (ii). Recall the fact that every fragment in H' is also a fragment in H. So, v is also a neighbor of Y in the original graph H; that is, $v \in N_H(Y)$. We use the fact that every vertex in a minimal separator has neighbors in both sides (fragments) of the separator. In particular, v has neighbors in both Y and Y^*. Let u and w be neighbors of v in Y and Y^*, respectively. Then, by the construction, H' has an edge (u, w) connecting these two neighbors of v. (See Figure 1.) Thus, Y is not a fragment in H', a contradiction. This rules out case (ii).

Since both cases (i) and (ii) are impossible, we have a contradiction, and the if-part follows.

For the completeness, we prove the claim that v has neighbors in both Y and Y^*. (Readers who are familiar with this fact may skip the rest of the proof.) Assume a contradiction that v has no neighbors in Y or Y^*. Since it is symmetric, we may assume that v has no neighbors in Y. Now, consider the graph $H - (N_H(Y) - \{v\})$. Then $Y - \{v\}$ has no edge connecting to $Y^* \cup \{v\}$. This means that $(N_H(Y) - \{v\})$ is a separator of size $\ell - 1$, contradicting to the fact that H is ℓ-connected. □

6 The Preprocessing Step

In this section, we describe a preprocessing step that reduces the number of cores to k. This technique was introduced in [27] and was used in [33, 39].

Given an input graph $G = (V, E)$, we first construct an auxiliary graph $\widehat{G}$. This auxiliary graph is obtained by adding a new vertex r and joining r to any k vertices of G by zero cost edges; we denote the set of these k vertices by R. Then we apply to $\widehat{G}$ the Frank-Tardos algorithm to find a k-outconnected spanning subgraph $\widehat{H}$ with r as a root vertex. Thus, we have a subgraph $\widehat{H}$ that is k-outconnected from r. Removing r, we have a subgraph $H = \widehat{H} - r$ of G. We claim that, for all $\ell = 1, 2, \ldots, k - 1$, every ℓ-fragment of H, if it exists, contains a vertex of R. The claim is proved in the next lemma.

Lemma 6. *Consider a subgraph H of G constructed as above. For all $\ell = 1, 2, \ldots, k-1$, every ℓ-fragment of H, if it exists, contains a vertex of R.*

Proof. Assume a contradiction that, for some $1 \leq \ell < k$, H has an ℓ-fragment X that does not contain any vertex of R. Since $\ell < k$, $R - N_H(X) \neq \emptyset$. Thus, the complementary fragment X^* contains some vertex of R. Now, consider the graph $\widehat{H}$. Since $\widehat{H}$ is k-outconnected from r, $\widehat{H}$ has no ℓ-separators separating r from a vertex $v \in V$. So, $\widehat{H} - N_H(X)$ has a path P from r to X. By the construction, any path from r has to visit some vertex of R before entering X and so does P. We conclude that $P - \{r\}$ has an edge going from X^* to X, contradicting to the fact that X is an ℓ-fragment. □

For the case of directed graphs, we construct $\widehat{G}$ by adding arcs in both direction; that is, we add arcs (r, v) and (v, r) for each vertex $v \in R$. In this case, we have to apply Frank-Tardos algorithm twice to make the resulting graph H to be both k-outconnected and k-inconnected from r. The proof for the directed case is the same as that of the undirected case.

Now, we have that every ℓ-fragment contains a vertex of R. Since cores of small ℓ-fragments are pairwise disjoint, this means that we have at most $|R| = k$ cores. Moreover, the cost of H is at most $2 \cdot opt$, where opt is the cost of an optimal solution to k-VCSS. This is because the Frank-Tardos algorithm gives a solution of cost at most $2 \cdot opt$ for undirected graphs and at most opt for directed graphs (but, we apply the algorithm twice for latter case). Thus, we can add edges of H to the graph at the start of the algorithm to reduce the number of cores to k, and adding this preprocessing step does not increase the approximation factor of the algorithm. The next theorem follows immediately from our discussion.

Theorem 2. *There is an algorithm that reduces the number of cores to k for both undirected and directed graphs, and the cost of edges added is at most twice the cost of an optimal solution to the k-vertex connected spanning subgraph problem.*

Acknowledgment

We thank anonymous referees and all the colleagues for useful comments and discussions, especially Joseph Cheriyan and Zeev Nutov.

References

[1] Vincenzo Auletta, Yefim Dinitz, Zeev Nutov, and Domenico Parente. A 2-approximation algorithm for finding an optimum 3-vertex-connected spanning subgraph. *J. Algorithms*, 32(1):21–30, 1999. Preliminary versions in STACS'97 and CIAC'97.

[2] Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. An improved LP-based approximation for steiner tree. In *STOC*, pages 583–592, 2010.

[3] Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In *STOC*, pages 167–176, 2008.

[4] Chandra Chekuri and Nitish Korula. Single-sink network design with vertex connectivity requirements. In *FSTTCS*, pages 131–142, 2008.

[5] Chandra Chekuri and Nitish Korula. A graph reduction step preserving element-connectivity and applications. In *ICALP (1)*, pages 254–265, 2009.

[6] Joseph Cheriyan and Ramakrishna Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM J. Comput.*, 30(2):528–560, 2000. Preliminary version in FOCS'96.

[7] Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. An approximation algorithm for the minimum-cost k-vertex connected subgraph. *SIAM J. Comput.*, 32(4):1050–1055, 2003. Preliminary version in STOC'02.

[8] Joseph Cheriyan, Santosh Vempala, and Adrian Vetta. Network design via iterative rounding of setpair relaxations. *Combinatorica*, 26(3):255–275, 2006. Preliminary version in STOC'02.

[9] Joseph Cheriyan and Adrian Vetta. Approximation algorithms for network design with metric costs. *SIAM J. Discrete Math.*, 21(3):612–636, 2007. Preliminary version in STOC'05.

[10] Julia Chuzhoy and Sanjeev Khanna. Algorithms for single-source vertex connectivity. In *FOCS*, pages 105–114, 2008.

[11] Julia Chuzhoy and Sanjeev Khanna. An O(k^{3}log n)-approximation algorithm for vertex-connectivity survivable network design. In *FOCS*, pages 437–441, 2009.

[12] Artur Czumaj and Andrzej Lingas. On approximability of the minimum-cost k-connected spanning subgraph problem. In *SODA*, pages 281–290, 1999.

[13] Yefim Dinitz and Zeev Nutov. A 3-approximation algorithm for finding optimum 4, 5-vertex-connected spanning subgraphs. *J. Algorithms*, 32(1):31–40, 1999. Preliminary version in CIAC'97.

[14] Kapali P. Eswaran and Robert Endre Tarjan. Augmentation problems. *SIAM J. Comput.*, 5(4):653–665, 1976.

[15] Jittat Fakcharoenphol and Bundit Laekhanukit. An O($\log^2 k$)-approximation algorithm for the k-vertex connected spanning subgraph problem. In *STOC*, pages 153–158, 2008.

[16] Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. Syst. Sci.*, 72(5):838–867, 2006. Preliminary versions in FOCS'01 and IPCO'01.

[17] András Frank. Increasing the rooted-connectivity of a digraph by one. *Math. Program.*, 84(3, Ser. B):565–576, 1999. Connectivity augmentation of networks: structures and algorithms (Budapest, 1994).

[18] András Frank. Rooted k-connections in digraphs. *Discrete Applied Mathematics*, 157(6):1242–1254, 2009.

[19] András Frank and Éva Tardos and. An application of submodular flows. *Linear Algebra and its Applications*, 114-115:329 – 348, 1989.

[20] András Frank and Tibor Jordán. Minimal edge-coverings of pairs of sets. *J. Comb. Theory, Ser. B*, 65(1):73–110, 1995.

[21] Greg N. Frederickson and Joseph JáJá. Approximation algorithms for several graph augmentation problems. *SIAM J. Comput.*, 10(2):270–283, 1981.

[22] Harold N. Gabow and Suzanne Gallagher. Iterated rounding algorithms for the smallest k-edge connected spanning subgraph. *SIAM J. Comput.*, 41(1):61–103, 2012. Preliminary version in SODA'08.

[23] Harold N. Gabow, Michel X. Goemans, Éva Tardos, and David P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks*, 53(4):345–357, 2009. Preliminary version in SODA'05.

[24] Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.

[25] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001. Preliminary version in FOCS'98.

[26] Tibor Jordán. On the optimal vertex-connectivity augmentation. *J. Comb. Theory Ser. B*, 63(1):8–20, 1995.

[27] Samir Khuller and Balaji Raghavachari. Improved approximation algorithms for uniform connectivity problems. *J. Algorithms*, 21(2):434–450, 1996. Preliminary version in STOC'95.

[28] Samir Khuller and Uzi Vishkin. Biconnectivity approximations and graph carvings. *J. ACM*, 41(2):214–235, 1994. Preliminary version in STOC'92.

[29] Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004. Preliminary version in APPROX'02.

[30] Guy Kortsarz and Zeev Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003. Preliminary version in APPROX'00.

[31] Guy Kortsarz and Zeev Nutov. Approximating k-node connected subgraphs via critical graphs. *SIAM J. Comput.*, 35(1):247–257, 2005. Preliminary version in STOC'04.

[32] Guy Kortsarz and Zeev Nutov. *Handbook on Approximation Algorithms an Metaheuristics*, chapter Approximating min-cost connectivity problems. Chapman and Hall, 2006.

[33] Bundit Laekhanukit. An improved approximation algorithm for minimum-cost subset k-connectivity - (extended abstract). In *ICALP (1)*, pages 13–24, 2011.

[34] W. Mader. Endlichkeitsätze für k-kritische graphen (german). *Math. Ann.*, 229:143–153, 1977.

[35] Wolfgang Mader. On k-con-critically n-connected graphs. *J. Comb. Theory, Ser. B*, 86(2):296–314, 2002.

[36] Zeev Nutov. Approximating minimum cost connectivity problems via uncrossable bifamilies. Preliminary version in [38], Preprint available at `http://www.openu.ac.il/home/nutov/FOCS09.pdf`.

[37] Zeev Nutov. An almost $O(\log k)$-approximation for k-connected subgraphs. In *SODA*, pages 912–921, 2009.

[38] Zeev Nutov. Approximating minimum cost connectivity problems via uncrossable bifamilies and spider-cover decompositions. In *FOCS*, pages 417–426, 2009. Journal version in [36].

[39] Zeev Nutov. Approximating subset k-connectivity problems. In *WAOA*, pages 9–20, 2011.

[40] David Pritchard. k-edge-connectivity: Approximation and lp relaxation. In *WAOA*, pages 225–236, 2010.

[41] R. Ravi and David P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 18(1):21–43, 1997. Preliminary version in SODA'95. Erratum in [42].

[42] R. Ravi and David P. Williamson. Erratum: An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 34(1):98–107, 2002. Preliminary version in SODA'02.

[43] David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. In *STOC*, pages 708–717, 1993.

[44] Alexander Zelikovsky. An 11/6-approximation algorithm for the network steiner problem. *Algorithmica*, 9(5):463–470, 1993.