# An almost $O(\log k)$-approximation for $k$-connected subgraphs

Zeev Nutov
The Open University of Israel
nutov@openu.ac.il

**Abstract**

We consider two cases of the Survivable Network Design (SND) problem: given a complete graph $G_n = (V, E_n)$ with costs on the edges and connectivity requirements $\{r(u, v) : u, v \in V\}$, find a minimum cost subgraph $G$ of $G_n$ that contains $r(u, v)$ internally disjoint $uv$-paths for all $u, v \in V$. Our main result is an $O\left(\log k \cdot \log \frac{n}{n-k}\right)$-approximation algorithm for the $k$-Connected Subgraph problem (the case $r(u, v) = k$ for all $u, v \in V$), for both directed and undirected graphs, where $n = |V|$. Our ratio is $O(\log k)$, unless $k = n - o(n)$. Previously, the best known approximation guarantees for this problem were $O(\log^2 k)$ for directed/undirected graphs [Kortsarz and Nutov STOC 2004, Fakcharoenphol and Laekhanukit STOC 2008], and $O(\log k)$ for undirected graphs with $k \leq \sqrt{n/2}$ [Cheriyan, Vempala, and Vetta STOC 2002]. As in previous work, we consider the $k$-Connectivity Augmentation problem of increasing at minimum cost the connectivity of a given graph $J$ from $k-1$ to $k$; a $\rho$-approximation for it is used to derive an $O(\rho \cdot \log k)$-approximation for $k$-Connected Subgraph. Fakcharoenphol and Laekhanukit showed that $k$-Connectivity Augmentation admits an $O(\log \nu)$-approximation algorithm, where $\nu$ is the number of minimal "violated" sets in $J$. However, we may have $\nu = \Theta(n)$, so this gives only an $O(\log n)$-approximation. We design a novel primal-dual algorithm that adds an edge set of cost $\leq$ opt to get $\nu \leq \frac{2n}{n-k}$. Combined with the algorithm of Fakcharoenphol and Laekhanukit, this gives the ratio $O\left(\log \frac{n}{n-k}\right)$ for $k$-Connectivity Augmentation, which is $O(1)$, unless $k = n - o(n)$.

Our additional result is for the (undirected) Rooted SND, where for a "root" $s \in V$, the connectivity requirements are $\{r(s, t) = r(t) : t \in T \subseteq V\}$, and the solution graph should contain $r(t)$ internally disjoint $st$-paths for all $t \in T$. For large values of $k = \max_{t \in T} r(t)$ Rooted SND is at least as hard to approximate as Directed Steiner Tree [Lando and Nutov APPROX 2008]. For Rooted SND [Chakraborty, Chuzhoy, Khanna STOC 08] gave recently a $k^{O(k^2)} \log^4 n$-approximation algorithm. Slightly later [Chuzhoy and Khanna FOCS 08] improved the ratio to $O(k^2 \log n)$, and also gave an $O(k^8 \log^2 n)$-approximation algorithm for the case of node-costs. Independently, we obtained a simple approximation algorithm with ratios $O(k^2 \log n)$ for edge-costs, and $O(k^4 \log^2 n)$ for node-costs.

## 1 Introduction

### 1.1 Problems considered and related work

Let $\kappa_G(u, v)$ denote the maximum number of internally-disjoint $uv$-paths in a graph $G$. We consider variants of the following fundamental problem in network design:

Survivable Network Design (SND):
*Instance:* A complete graph $G_n = (V, E_n)$ with edge-costs $\{c(e) : e \in E_n\}$, and connectivity requirements $\{r(u, v) : u, v \in V\}$.
*Objective:* Find a minimum cost subgraph $G$ of $G_n$ so that $\kappa_G(u, v) \geq r(u, v)$ for all $u, v \in V$.

For an instance of SND at hand, let opt denote the optimal solution value, let $k = \max_{u,v \in V} r(u, v)$ denote the maximum requirement, and let $n = |V|$. While the edge-connectivity case – the so called Steiner Network problem – admits a 2-approximation algorithm, nontrivial approximation algorithms for SND are known only for metric costs [7] by Cheriyan and Vetta, and for $0, 1$-costs [24, 29]. A hardness result of Kortsarz et al. [20] suggests that for general costs SND is unlikely to admit a polylogarithmic approximation; this is so even for $0, 1$-costs [30]. See also a slightly improved hardness result in [2]. It was recently shown by Y. Lando and the author [25] by a simple proof that directed SND problems can be reduced to their corresponding undirected variants with large connectivity requirements. In particular, this implies the hardness result of [20] (we will elaborate on other consequences later).

A simple directed/undirected graph is *$k$(-node) connected* if it contains $k$ internally-disjoint paths from every node to the other. When $r(u, v) = k$ for all $u, v \in V$, we have the following particular extensively studied case of SND:

$k$-Connected Subgraph

*Instance:* A complete graph $G_n = (V, E_n)$ with edge-costs $\{c(e) : e \in E_n\}$ and an integer $k$.

*Objective:* Find a minimum cost $k$-connected spanning subgraph $G$ of $G_n$.

This problem was studied extensively both in general setting [33, 5, 6, 22, 10, 25], as well as for restricted settings [12, 17, 18, 15, 16, 19, 1, 9, 21, 4, 26]. The approximability of the problem for general costs, and its complexity status for $0,1$-costs are long standing open problems. For general edge-costs, the undirected case is NP-hard for $k = 2$, and the directed one for $k = 1$. Improving the trivial ratio of $2k$, Kortsarz and the author [21] gave a $k$-approximation for undirected graphs and a $(k+1)$-approximation for directed graphs. Cheriyan, Vempala, and Vetta [6] used the iterative rounding method to improve for $k = \Omega(n^{1/2+\varepsilon})$ the ratio to $n/\sqrt{n-k}$, where $n = |V|$. In a different paper [5] they gave an $O(\log k)$-approximation algorithm for undirected graphs with $n \geq 6k^2$, based on a result of Mader [27]; this results extends also for the case $n \geq 2k^2$ using Mader's result from [28]. Kortsarz and the author [22] gave an $O(\frac{n}{n-k}\log^2 k)$-approximation algorithm for both directed and undirected graphs, which is $O(\log^2 k)$ unless $k = n - o(n)$. Recently, Fakcharoenphol and Laekhanukit [10] obtained an $O(\log^2 k)$-approximation for all $n, k$. By a recent result of Y. Lando and the author [25], for $k > n/2$ directed and undirected variants of $k$-Connected Subgraph are equivalent w.r.t. approximation, up to a constant factor.

Better approximation guarantees are known for small values of $k$ [1, 9, 21], for metric costs [19, 21], for $1,\infty$-costs, and for $0,1$-costs [12, 17, 18, 15, 16, 26]. We survey the literature on the latter case of $0,1$-costs, namely, when we are given a graph $J$ (of cost 0), and the goal is to find a minimum size edge-set $I$ (any edge is allowed by cost of 1) so that $J + I$ is $k$-connected. For undirected graphs, the complexity status of $k$-Connected Subgraph with $0,1$-costs is among the oldest open problems in network design, see [17, 18, 15, 16, 31, 26]. However, the directed case is solvable in polynomial time Frank and Jordán [12]. Jordán [17, 18] gave an algorith that computes a solution of size $\mathsf{opt} + k/2$ for the problem of increasing the connectivity by 1 (see an improved version of Jordán's algorithm in [26], and a related result in [31]). This result was generalized by Jordán and Jackson [15] that gave an algorithm that computes a solution of size roughly $\mathsf{opt} + k^2/2$ for the general $k$-Connected Subgraph with $0,1$-costs. Another very interesting result of Jordán and Jackson [16] shows that the problem can be solved exactly in time $O(2^k poly(n))$. See also a survey in [23].

For an edge-set or a graph $J$ on a node set $V$ and disjoint $X, Y \subseteq V$, let $\delta_J(X, Y)$ denote the set of edges in $J$ going from $X$ to $Y$. By Menger's Theorem, there are $k$ internally disjoint $st$-paths in $J$ if, and only if, $|\delta_J(S,T)| \geq k - (n - |S \cup T|)$ for all disjoint $S, T \subset V$ with $s \in S$ and $t \in T$. We will compare the cost of our solutions to the optima $\mathsf{opt}_k$ of the following LP-relaxation for the minimum cost $k$-node connected spanning subgraph that has been introduced in [12]:

$$\begin{aligned} \min & \sum_{e \in \mathcal{E}} c_e x_e \\ \text{s.t.} & \sum_{e \in \delta_{\mathcal{E}}(S,T)} x_e \geq k - (n - |S \cup T|) \quad S, T \subset V, S \cap T = \emptyset \\ & 0 \leq x_e \leq 1 \qquad\qquad\qquad\qquad e \in E \end{aligned}$$

Prior to all mentioned work, Ravi and Williamson [33] designed a primal-dual algorithm for $k$-Connected Subgraph on undirected graphs, and gave an analysis that this algorithm has ratio $O(\log k)$. However, the proof was found to contain an error, see [34]; in [34], Ravi and Williamson gave an example showing that the approximation ratio of their algorithm in [33] is $\Omega(k)$. Still, many ideas from the Ravi-Williamson algorithm were used in the subsequent papers [5, 22, 10], including using small violated sets, and paying only an $O(\log k)$ factor for considering the corresponding "augmentation" problem. The $k$-Connectivity Augmentation problem is the restriction of $k$-Connected Subgraph to instances in which $G_n$ contains a $(k-1)$-connected spanning subgraph $J$ of cost zero; namely, we seek to increase at minimum cost the connectivity of $J$ from $\ell = k - 1$ to $\ell + 1 = k$. The following statement was implicitly proved by Ravi and Williamson in [33].

PROPOSITION 1.1. ([33]) *Suppose that* $k$-Connectivity Augmentation *admits a polynomial time algorithm that computes a solution of cost* $\leq \alpha(n,k) \cdot \mathsf{opt}_k$. *Then* $k$-Connected Subgraph *admits a polynomial time algorithm that computes a solution of cost* $\leq \mathsf{opt}_k \cdot \sum_{\ell=1}^{k} \frac{\alpha(n,\ell)}{k-\ell+1}$. *In particular, if* $\alpha(n,\ell)$ *is increasing in* $\ell$ *then the cost of the solution computed* $\leq \alpha(n,k)H(k)$ *(*$H(k)$ *denotes the* $k$*th Harmonic number).*

The approach in [5, 22, 10] for $k$-Connected Subgraph was to design an approximation algorithm for $k$-Connectivity Augmentation. This was done as follows. A graph $G = (V, E)$ is $k$-outconnected from $s$ if it contains $k$ internally-disjoint paths from $s$ to every $v \in V - s$. Frank and Tardos [13] showed that the problem of finding a minimum-cost $k$-outconnected spanning subgraph can be solved in polynomial time for directed graphs; this implies a 2-approximation algorithm for undirected graphs. A $k$-connected graph is

$k$-outconnected from every node, hence the minimum-cost of a $k$-outconnected subgraph is a lower bound on opt. The inverse is not true, as a $k$-outconnected graph is not $k$-connected if the root $s$ belongs to a node-cut of cardinality $\leq k-1$. However, if $R$ is a node set so that no node-cut of $J$ of size $\leq k-1$ contains $R$, then by executing the Frank-Tardos algorithm from every root $s \in R$, and taking the union of $J$ and the $|R|$ subgraphs computed, we obtain a $k$-connected graph. This approach was used in [5] and [22]. Mader [27] showed that for undirected graphs with $n \geq 6k^2$ there exists such $R$ with $|R| = 3$; this was used by Cheriyan, Vempala, and Vetta [5] to obtain an $O(1)$ ratio for undirected $k$-Connectivity Augmentation with $n \geq 6k^2$. Kortsarz and the author [22] proved that for any $n, k$ one can find such $R$ with $|R| = O(\frac{n}{n-k} \cdot \log k)$ in both directed and undirected graphs. However, for $k = n - O(1)$ one cannot achieve a better bound than $O(n)$ on $|R|$. Using decomposition of feasible solutions, Fakcharoenphol and Laekhanukit [10] observed that the Frank-Tardos algorithm can be used to find a partial augmenting edge set that decreases the number $\nu$ of minimal violated sets by 1 and has cost $O(1/\nu) \cdot \mathsf{opt}$. Using standard set-cover analysis, this gives an $O(\log \nu)$ ratio for $k$-Connectivity Augmentation. In fact, the same decomposition can be used to compute an edge set of cost $\leq \mathsf{opt}$ that decreases the number of minimal violated sets from $\nu$ to $\lfloor \nu/2 \rfloor$, which gives another $O(\log \nu)$-approximation.

We note that the algorithms in [5, 22, 10] require only a procedure that increases the outconnectivity by 1 at minimum cost. Hence, instead of using the Frank-Tardos algorithm, they can use for this particular case the simple and elegant primal-dual algorithm of Frank [11], which is also faster. Consequently, the algorithms of [5, 22, 10] can be casted as a combination of the primal-dual approach with the greedy method.

Our additional result is for a particular case of undirected SND, where for a "root" $s \in V$ the connectivity requirements are $\{r(s,t) = r(t) : t \in T \subseteq V\}$:

Rooted SND:
*Instance:* A complete undirected graph $G_n = (V, E_n)$ with edge-costs $\{c(e) : e \in \mathcal{E}\}$, a root $s \in V$, a set $T$ of *terminals*, and connectivity requirements $\{r(t) > 0 : t \in T\}$.
*Objective:* Find a minimum cost subgraph $G$ of $G_n$ so that $\kappa_G(s,t) \geq r(t)$ for all $t \in T$.

While the $k$-Connected Subgraph problem was studied extensively, there is almost no literature on (undirected) Rooted SND, except for the case of $0,1$-costs [24, 31, 30, 29]. Recently, Rooted SND (with general costs) received some attention because Chakraborty, Chuzhoy, and Khanna [2] obtained a $k^{O(k^2)} \log^4 n$-approximation for it. Slightly later, Chuzhoy and Khanna [8] improved the ratio to $O(k^2 \log n)$. As for hardness of approximation, one of the consequences from a recent result of Y. Lando and the author [25] is that Rooted SND with $k > n/2$ is at least as hard to approximate as the notorious Directed Steiner Tree problem; the currently best known algorithm for the latter is an $O(|T|^{\varepsilon}/\varepsilon^3)$-approximation scheme in $O(|T|^{4/\varepsilon} n^{2/\varepsilon})$ time [3].

**1.2 Results and techniques** The best ratio for $k$-Connectivity Augmentation was $O(\log k)$ [10] for both directed and undirected graphs, and $O(1)$ for undirected graphs with $n \geq 2k^2$ [5]. In this paper we design a novel primal-dual algorithm, which we combine with the greedy method used in [22] and [10], to obtain the following result:

THEOREM 1.1. *$k$-Connectivity Augmentation admits a polynomial time algorithm that computes a solution of cost $O\left(\log \frac{n}{n-k}\right) \cdot \mathsf{opt}_k$; the approximation ratio is $O(1)$, unless $k = n - o(n)$.*

Thus we obtain for $k$-Connectivity Augmentation a constant ratio for almost all values of $n, k$, for both directed and undirected graphs, a result that was known only for undirected graphs with $n \geq 2k^2$ [5]. Note that even for $k = n - n/\log n$, our ratio is $O(\log \log k)$, which is better than the $O(\log k)$ ratio of [10]. It remains an open question whether $k$-Connectivity Augmentation admits a constant approximation ratio for all $n, k$, but we believe that the ratio in Theorem 1.1 characterizes the problem. Combining Theorem 1.1 with Proposition 1.1 we obtain:

THEOREM 1.2. *$k$-Connected Subgraph admits an $O\left(\log k \cdot \log \frac{n}{n-k}\right)$-approximation algorithm; the ratio is $O(\log k)$, unless $k = n - o(n)$.*

Our main new technique is a novel primal-dual algorithm that by cost $O(\mathsf{opt}_k)$ reduces the number of minimal violated sets from $n$ to $\frac{2n}{n-k}$. The algorithm partly uses the setting of the Ravi-Williamson algorithm [33], but is applied on directed graphs, with the following two crucial changes:

- During the algorithm some minimal violated sets – so called min-cores – are declared as "passive". The algorithm tries to cover only active (non-passive) min-cores. We will show that during the algorithm at most $\frac{2n}{n-k}$ min-cores are declared as passive.
- When trying to cover the active min-cores, the dual variables corresponding to them are not raised uniformly. Instead, we choose an active core $C$ and raise the dual variable corresponding to $C$ only.

After the number of cores is reduced to $\nu = \frac{2n}{n-k}$ we apply the $O(\log \nu)$-approximation algorithm of [10] to obtain the ratio $O\left(\log \frac{n}{n-k}\right) \cdot \mathsf{opt}_k$ for $k$-Connectivity Augmentation.

We now describe our results for Rooted SND. As was mentioned, Chakraborty, Chuzhoy, and Khanna [2] obtained a $k^{O(k^2)} \log^4 n$-approximation for Rooted SND, and recently Chuzhoy and Khanna [8] improved the ratio to $O(k^2 \log n)$; they also gave an $O(k^8 \log^2 n)$-approximation for the variant with node-cost. Independently, using different techniques from the ones in [2, 8], we obtained a simple proof of the following result:

THEOREM 1.3. Rooted SND *admits approximation algorithms with ratios* $O(k^2 \log |T|)$ *for edge-costs and* $O(k^4 \log^2 |T|)$ *for node-costs, where* $k = \max_{t \in T} r(t)$.

*Remark:* For the case of Rooted SND with edge-costs and requirements in $\{0, k\}$, namely, when $r(t) = k$ for all $t \in T$, our algorithm has ratio $O(k \log k \cdot \log |T|)$. This can be proved using standard scaling techniques, similar to the ones given in Proposition 1.1 for $k$-Connected Subgraph. We note that for this restricted version, Chuzhoy and Khanna [8] obtained the slightly better ratio of $O(k \log |T|)$. For other variants, our ratio is the same as in [8] for edge-costs, while for node costs our ratio is better.

As for $k$-Connected Subgraph, we consider an intermediate augmentation problem. The Rooted SND Augmentation problem is the restriction of Rooted SND to instances in which $G_n$ contains a subgraph $J$ of costs 0 with $\kappa_J(s,t) = k-1$, and the requirements are $r(t) = k$ for all $t \in T$; namely, we seek to increase at minimum cost the connectivity between $s$ and nodes in $T$ from $\ell = k-1$ to $\ell + 1 = k$.

THEOREM 1.4. Rooted SND Augmentation *admits approximation algorithms with ratios* $O(k \log |T|)$ *for edge-costs and* $O(k^3 \log^2 |T|)$ *for node-costs.*

It is easy to see that a $\rho$-approximation algorithm for Rooted SND Augmentation implies a $k\rho$-approximation algorithm for Rooted SND; thus Theorem 2.1 follows from Theorem 1.4, so we only need to prove Theorem 1.4. To see this, consider the following algorithm for Rooted SND. Start with $J = (V, \emptyset)$ and continue with iterations. Iteration $\ell$ starts with a graph $J$ with $\kappa_J(s,t) = \min\{\ell, r(t)\}$ for all $t \in T$, and seeks to increase the $st$-connectivity from $\ell$ to $\ell + 1$ for every $t \in T$ with $\kappa_J(s,t) = \ell$ and $r(t) \geq \ell + 1$; this is an instance of Rooted SND Augmentation. We find an edge set $I_\ell$ of cost $\rho \cdot \mathsf{opt}$ using the $\rho$-approximation algorithm for Rooted SND Augmentation. After at most $k$ iterations, $J$ satisfies the requirements, and its cost is $\leq k\rho \cdot \mathsf{opt}$.

## 2 Algorithm for $k$-Connectivity Augmentation (Proof of Theorem 1.1)

We prove Theorem 1.1 for directed graphs; for undirected graphs the result follows from the known statement (c.f., [23]) that for $k$-Connected Subgraph and for $k$-Connectivity Augmentation a $\rho$-approximation algorithm for directed graphs implies a $2\rho$-approximation algorithm for undirected graphs. Hence in this section all graphs are assumed to be directed, unless stated otherwise. Our goal is to augment an $\ell$-connected (directed) graph $J$ by a min-cost edge-set $I \subseteq \mathcal{E} - E(J)$ so that $J + I$ is $(\ell+1)$-connected.

DEFINITION 2.1. *For* $S \subseteq V$ *let* $\Gamma_J(S)$ *denote the set of neighbors of* $S$ *in* $J$*, and let* $X^* = V - X - \Gamma_J(X)$*. Let* $\mathcal{F}_J = \{X \subseteq V : |\Gamma_J(X)| = \ell \text{ and } X^* \neq \emptyset\}$*, and let* $\mathcal{S}_J = \{X \in \mathcal{F}_J : |X| \leq (n-\ell)/2\}$.

An edge $e$ *covers* a set $X \in \mathcal{F}_J$ if it goes from $X$ to $X^*$. The following statement, that was implicitly proved in [33], summarizes the properties of sets in $\mathcal{F}_J$ and $\mathcal{S}_J$ that we use.

LEMMA 2.1. ([33]) *Let* $X, Y \in \mathcal{F}_J$*. If* $X \cap Y \neq \emptyset$ *and* $X^* \cap Y^* \neq \emptyset$ *then:*

(i) $X \cap Y, X \cup Y \in \mathcal{F}_J$*, and* $(X \cap Y)^* = X^* \cup Y^*$*,* $(X \cup Y)^* = X^* \cap Y^*$.

(ii) *If an edge* $e$ *covers* $X \cap Y$ *or* $X \cup Y$ *then* $e$ *covers* $X$ *or* $Y$.

(iii) *If and edge* $e$ *covers both* $X \cap Y, X \cup Y$ *then* $e$ *covers both* $X, Y$.

*Furthermore, if* $X, Y \in \mathcal{S}_J$ *and if* $X \cap Y \neq \emptyset$*, then* $X^* \cap Y^* \neq \emptyset$*; thus* (i)–(iii) *holds in this case. In particular, the intersection of two intersecting sets in* $\mathcal{S}_J$ *is also a set in* $\mathcal{S}_J$.

DEFINITION 2.2. *A set in* $\mathcal{S}_J$ *is a* core *if it does not contain two disjoint members of* $\mathcal{S}_J$*; an inclusion minimal core is a* min-core*. Let* $\mathcal{C}_J$ *denote the family of min-cores in* $J$*. For* $C \in \mathcal{C}_J$ *the* halo-family $\mathcal{H}(C)$ *of* $C$ *is the set of cores that contain* $C$*. Let* $H_C$ *be the union of the sets in* $\mathcal{H}(C)$.

LEMMA 2.2. ([22]) *Let* $C \in \mathcal{C}_J$ *and let* $X \in \mathcal{S}_J$*. If* $X \cap H_C \neq \emptyset$ *then* $C \subseteq X$*. Thus the members of each one of the families* $\{H_C : C \in \mathcal{C}_J\}$ *and* $\mathcal{C}_J$ *are pairwise disjoint.*

LEMMA 2.3. ([10]) *Each one of the families* $\mathcal{C}_J$ *and* $\{H_C : C \in \mathcal{C}_J\}$ *can be computed in polynomial time.*

### 2.1 The algorithm (Proof of Theorem 1.1)

An edge set $I$ covers a subfamily $\mathcal{F} \subseteq \mathcal{F}_J$, or $I$ is an $\mathcal{F}$-cover, if for every $X \in \mathcal{F}$ there is an edge in $I$ from $X$ to $X^*$. It well known and easy is to see that $J+I$ is $(\ell+1)$-connected if, and only if, $I$ covers $\mathcal{F}_J$ (c.f., [22]). For $S \in \mathcal{F}_J$, let $\delta_I(S) = \delta_I(S, S^*)$; for brevity, $\delta(S) = \delta_{E_n}(S)$. Let $\tau(\mathcal{F})$ denote the optimal value of the following LP-relaxation for covering a subfamily $\mathcal{F} \subseteq \mathcal{F}_J$:

$$\begin{array}{lllll} \textbf{(P)} & \tau(\mathcal{F}) = & \min & \sum_{e \in E} c_e x_e & \\ & & \text{s.t.} & \sum_{e \in \delta(S)} x_e \geq 1 & \forall\, S \in \mathcal{F} \\ & & & x_e \geq 0 & \forall\, e \in \mathcal{E} \end{array}$$

THEOREM 2.1. $k$-Connectivity Augmentation *admits a polynomial time algorithm that computes an $\mathcal{F}_J$-cover of cost $\leq 2\left(1 + H\left(\left\lfloor \frac{2n}{n-\ell} \right\rfloor\right)\right)\tau(\mathcal{F}_J) = O\left(\log \frac{n}{n-\ell}\right)\tau(\mathcal{F}_J)$.*

The following two statements summarize the main ideas of our algorithm. The first is due to Fakcharoenphol and Laekhanukit [10], while the second will be proved in the next section.

THEOREM 2.2. ([10])
*There exists a polynomial time algorithm that computes an $\mathcal{S}_J$-cover of cost $H(|\mathcal{C}_J|) \cdot \tau(\mathcal{F}_J)$.*

THEOREM 2.3. *There exists a polynomial time algorithm that computes a subfamily $\mathcal{U}$ of pairwise disjoint members of $\mathcal{S}_J$ and an edge-set $I \subseteq E$ such that the following two properties hold:*

**Property 1:** *$|\mathcal{C}_{J+I}| \leq \lfloor 2n/(n-\ell) \rfloor$.*

**Property 2:** *$I$ is an optimal cover of the family $\mathcal{I} = \{X \in \mathcal{S}_J : X \subseteq U \text{ for some } U \in \mathcal{U}\}$, and $c(I) = \tau(\mathcal{I})$.*

Theorems 2.3 and 2.2 easily imply Theorem 2.1. Let us show an algorithm that covers the family $\mathcal{S}_J$. We first compute an edge set $I$ as in Theorem 2.3; note that $c(I) \leq \tau(\mathcal{I}) \leq \tau(\mathcal{S}_J) \leq \tau(\mathcal{F}_J)$. Then we compute a cover $F$ of the residual family $\mathcal{F}_{J+I}$ using the algorithm from Theorem 2.2. The cost of the cover of $\mathcal{S}_J$ computed is bounded by:

$$\begin{aligned} c(I+F) &= c(I) + c(F) \\ &\leq \tau(\mathcal{S}_J) + H(\lfloor 2n/(n-\ell) \rfloor) \cdot \tau(\mathcal{F}_{J+I}) \\ &\leq (1 + H(\lfloor 2n/(n-\ell) \rfloor)) \cdot \tau(\mathcal{F}_J)\ . \end{aligned}$$

In order to cover the family $\mathcal{F}_J$, we execute this algorithm twice: once on $J$ and once on the reverse graph of $J$; the union of the two partial solutions computed covers the entire family $\mathcal{F}_J$.

### 2.2 Proof of Theorem 2.3

The proof of Theorem 2.3 follows. The dual LP of **(P)** is:

$$\begin{array}{lllll} \textbf{(D)} & \max & \sum_{S \in \mathcal{F}} y_S & & \\ & \text{s.t.} & \sum_{S \in \mathcal{F}: e \in \delta(S)} y_S \leq c_e & & \forall e \in \mathcal{E} \\ & & y_S \geq 0 & & \forall S \in \mathcal{F} \end{array}$$

For an edge set $I \subseteq \mathcal{E} - E(J)$ let $x^I$ denote the characteristic vector of $I$. Given a solution $y$ to **(D)**, an edge $e \in \mathcal{E} - E(J)$ is *tight* if the inequality in **(D)** that corresponds to $e$ holds with equality. Equivalently, setting $\bar{c}(e) \leftarrow c(e) - \sum\{S \in \mathcal{F} : e \in \delta(S)\} y_S$ to be the *residual costs*, we have that an edge $e$ is tight if, and only if, $\bar{c}(e) = 0$. The algorithm produces a subfamily $\mathcal{U}$ of pairwise disjoint members of $\mathcal{S}_J$, an edge set $I \subseteq E$, and a dual solution $y$ so that for $\mathcal{I} = \{X \in \mathcal{S}_J : X \subseteq U \text{ for some } U \in \mathcal{U}\}$ the following holds: $I$ covers $\mathcal{I}$ (so $x^I$ is a feasible solution to **(P)** with $\mathcal{F} = \mathcal{I}$), $y$ is a feasible solution to **(D)**, and $x^I$ and $y$ satisfy the complementary slackness conditions:

*Primal Complementary Slackness Conditions:*
$$e \in I \Longrightarrow e \text{ is tight.}$$
*Dual Complementary Slackness Conditions:*
$$y_S > 0 \Longrightarrow |\delta_I(S)| = 1.$$

Before describing the main algorithm, we need to describe a certain procedure that we use. A family $\mathcal{F}$ of non-empty sets is a *ring-family* if $X \cap Y, X \cup Y \in \mathcal{F}$ for any $X, Y \in \mathcal{F}$ and $\mathcal{F}$ has a unique non-empty minimal set (the intersection of all sets in $\mathcal{F}$); note that $\mathcal{F}$ also has a unique maximal set (the union of all sets in $\mathcal{F}$). If $|H_C| \leq (n-\ell)/2$ for some min-core $C$, then $\mathcal{H}(C)$ is a ring-family, with minimal set $C$ and maximal set $H_C$. The procedure that we use is the first phase of a standard primal-dual algorithm for covering such a ring family $\mathcal{H}(C)$ (the second phase is so called "reverse-delete", which we do not apply in the procedure, but only in the main algorithm). Given a partial solution $I_C$, we raise the dual variable of the (unique) minimal not yet covered set in $\mathcal{H}(C)$, until some edge covering it becomes tight, and add it to $I_C$. We repeat this step until the maximal set $H_C$ is covered. It is known, and easy to see, that the produced edge set indeed covers $\mathcal{H}(C)$. Note that at the end, there is a *unique* edge that covers the maximal set $H_C$, namely

$$|\delta_{I_C}(H_C)| = 1\ . \tag{2.1}$$

We now describe the main algorithm. The algorithm has two phases.

**Phase I** starts with $I = \emptyset$ and applies a sequence of iterations. At the beginning of an iteration, we compute the families $\mathcal{C}_{J+I}$ and $\mathcal{H}_{J+I}$, and declare some cores in $\mathcal{C}_{J+I}$ *passive*. If a core is declared passive at some iteration, it remains passive and uncovered in all the future iterations; moreover, no member of its halo-family will be ever covered. Then we choose some active (that is, non-passive) core $C$; $|H_C| \leq (n-\ell)/2$ if $C$ is active, hence $\mathcal{H}(C)$ is a ring-family. Thus we can apply a standard primal-dual algorithm to produce an edge $I_C$ that covers the family $\mathcal{H}(C)$, as described above, and note that (2.1) holds for $I_C$ and $H_C$. The edge-set $I_C$ is added to $I$, the set $H_C$ is added to $\mathcal{U}$, and the sets in $\mathcal{U}$ properly contained in $H_C$ are excluded from $\mathcal{U}$; our definition of active cores will not allow that $H_C$ will cross some $U \in \mathcal{U}$, see Lemma 2.4(iii). Phase I terminates when no active cores remain.

**Phase 2** applies on $I$ "reverse delete", which means the following. Let $I = \{e_1, \ldots, e_j\}$, where $e_{i+1}$ was added after $e_i$. For $i = j$ downto 1, we delete $e_i$ from $I$ if $I - \{e_i\}$ still covers the set-family $\mathcal{I} = \{X \in \mathcal{S}_J : X \subseteq U \text{ for some } U \in \mathcal{U}\}$. This can be implemented in polynomial time as follows. When an edge $e \in I$ is checked for deletion, $\mathcal{I}$ is covered by $I - \{e\}$ if, and only if, no core of $J + (I - e)$ is contained in a set $U \in \mathcal{U}$. At the end of the algorithm, $I$ is output.

Now we describe formally the update of the family $\mathcal{U}$ and the criteria for declaring cores passive.

**The family $\mathcal{U}$:** Initially, $\mathcal{U} = \emptyset$. We will prove later that each time we cover the family $\mathcal{H}(C)$ of some current core $C$, $H_C$ does not cross any $U \in \mathcal{U}$, see Lemma 2.4(ii). We set

$$\mathcal{U} \leftarrow \mathcal{U} - \{U \in U : U \subseteq H_C\} + H_C \ ,$$

namely, we replace all the members of $\mathcal{U}$ contained in $H_C$, if any, by $H_C$.

**Passive cores:** At the beginning of an iteration, a core $C$ is declared passive if the union

$$B(C) = H_C + \bigcup \{U \in \mathcal{U} : H_C \cap U \neq \emptyset\}$$

of $H_C$ and the sets in $\mathcal{U}$ intersecting $H_C$ is larger than $(n-\ell)/2$, namely, if $|B(C)| > (n-\ell)/2$. If a core is declared passive at some iteration, it remains passive in all the future iterations.

It is easy to see that the algorithm can be implemented in polynomial time, by Lemma 2.3. For proving **Property 1**, the key point is:

LEMMA 2.4. *At any iteration of the algorithm, if $I$ is the edge set added so far, the following holds:*

(i) *The members of $\mathcal{U}$ are pairwise disjoint and every $U \in \mathcal{U}$ is covered by a unique edge $e_U \in I$.*

(ii) *If some $X \in \mathcal{S}_{J+I}$ and $U \in \mathcal{U}$ intersect, then $\delta_I(X \cap U) = \{e_U\}$ and $\delta_I(X \cup U) = \emptyset$.*

(iii) *For any $U \in \mathcal{U}$ and any active core $C$, either $U \subset H_C$ or $U \cap H_C = \emptyset$.*

*Proof.* By induction on the number of iterations of the algorithms. Initially, $\mathcal{U} = \emptyset$ and the statement is valid. Suppose that the statement is true at the beginning of some iteration. We will show that it remains true also at the end of the iteration.

*Proof of* (i)*:* At the current iteration, we choose an active core $C$, cover the family $\mathcal{H}(C)$ by an edge-set $I_C$ satisfying (2.1), and add $H_C$ to $U$. Thus (i) follows from (ii) and (iii) in the induction hypothesis.

*Proof of* (ii)*:* Note that $X, U \in \mathcal{S}_J$, hence $X \cap U \in \mathcal{S}_J$, by Lemma 2.1. Since all the members of $\mathcal{S}_J$ contained in $U$ are already covered by $I$, there is an edge $e \in I$ covering $X \cap U$. $X$ is not covered by $I$, hence by Lemma 2.1(ii), $e$ covers $U$. But, by the induction hypothesis, $e_U$ is the only edge in $I$ covering $U$, hence $e = e_U$; $e_U$ cannot cover $X \cup U$, as then $e_U$ covers $X$, by Lemma 2.1(iii).

*Proof of* (iii)*:* Clearly, we cannot have $H_C \subseteq U$, as for every $U \in \mathcal{U}$ all the members of $\mathcal{S}_J$ contained in $U$ are already covered. Thus if the statement does not hold for $H_C$ and $U$, then $H_C$ and $U$ cross. As $C$ is active, $|H_C \cup U| \leq |B(C)| \leq (n-\ell)/2$. By (i) in the induction hypothesis, $H_C \cup U$ is still uncovered. Note that $H_C \cup U$ contains no core distinct from $C$ since all the members of $\mathcal{S}_J$ contained in $U$ are covered, and since, by Lemma 2.2, no min-core except $C$ intersects $H_C$. But then, $H_C \cup U \subseteq H_C$, by the definition of $H_C$, which gives a contradiction.

The proof of Lemma 2.4 is complete.

COROLLARY 2.1. *At any iteration of the algorithm, the members of $\mathcal{U}$ are pairwise disjoint, and for any $U \in \mathcal{U}$ there is at most one min-core $C$ so that $H_C \cap U \neq \emptyset$.*

*Proof.* Suppose to the contrary that $H_C \cap U \neq \emptyset$ and $H_{C'} \cap U \neq \emptyset$ for distinct min-cores $C, C'$. Let $u$ be the tail of $e_U$. By Lemma 2.4(i), $u \in H_C \cap H_{C'}$. This contradicts Lemma 2.2.

**Proof of Property 1:** The sets $B(C)$ are pairwise disjoint, by Corollary 2.1 and Lemma 2.2. At the end of the algorithm, only passive cores remain, hence we have $|B(C)| > (n-\ell)/2$ for every core $C \in \mathcal{C}_{J+I}$. Thus $|\mathcal{C}_{J+I}| \leq \lfloor 2n/(n-\ell) \rfloor$, as claimed. □

**Proof of Property 2:** As any set $H_C$ added to $\mathcal{U}$ is a member of $\mathcal{S}_J$, $\mathcal{U} \subseteq \mathcal{S}_J$. The members of $\mathcal{U}$

are pairwise disjoint, by Lemma 2.4(i). For every set $H_C$ added to $\mathcal{U}$, we covered all the members of $\mathcal{S}_J$ contained in $U$. Thus at the end of Phase 1, $I$ covers $\mathcal{I} = \{X \in \mathcal{S}_J : X \subseteq U \text{ for some } U \in \mathcal{U}\}$. It is also clear that $I$ remains a cover of $\mathcal{I}$ after Phase 2. Thus it remains to prove that $c(I) \leq \tau(\mathcal{I})$. Consequently, to finish the proof of Theorem 2.3 it is sufficient to prove:

LEMMA 2.5. *When the algorithm ends, $x^I$ is a feasible solution to* **(P)**, *$y$ is a feasible solution to* **(D)**, *and $x^I, y$ satisfy the complementary slackness conditions; hence $x^I$ and $y$ are optimal solutions.*

*Proof.* It is clear that $x^I$ is a feasible solution to **(P)** and that during the algorithm $y$ remains a feasible solution to **(D)**. Since only tight edges enter $I$, after Phase 1 the *Primal Complementary Slackness Conditions* hold for $I$. So, the only part that requires proof is that after Phase 2, the *Dual Complementary Slackness Conditions* hold for $x^I$ and $y$.

*Claim: Consider an arbitrary $S \in \mathcal{I}$ with $y_S > 0$ and an edge $e \in \delta_I(S)$. There exists $W_e \in \mathcal{I}$ such that $S \subseteq W_e$ and $\delta_I(W_e) = \{e\}$.*

*Proof.* Such $W_e$ can be chosen as any member of $\mathcal{I}$ which becomes uncovered if we delete (instead of keeping) $e$ at the reverse delete step when $e$ was considered for deletion; note that the algorithm decided to keep $e$, hence such $W_e$ exists. Moreover, since the edges were deleted in the reverse order, $W_e \in \mathcal{S}_{J+I-e}$. Obviously, $\delta_I(W_e) = \{e\}$ and $S$ and $W_e$ intersect. Finally, to see that $S \subseteq W_e$ note that: (i) at any iteration before $e$ was added, $W_e$ was uncovered; (ii) since $y_S > 0$, there was an iteration before $e$ was added at which $S$ was a min-core. Hence $S \subseteq W_e$, by Lemma 2.2.

We show that the *Dual Complementary Slackness Conditions* hold for $x^I$ and $y$. Assume to the contrary that there is $S \in \mathcal{F}$ with $y_S > 0$ such that there are $e_1, e_2 \in \delta_I(S)$, $e_1 \neq e_2$. Let $W_1 = W_{e_1}$ and $W_2 = W_{e_2}$ be as in the Claim above. Then $S \subseteq W_1 \cap W_2$; in particular $W_1$ and $W_2$ intersect, so $W_1 \cup W_2 \in \mathcal{I}$ by Lemma 2.1, since if two sets in $\mathcal{I}$ intersect then they are contained in some $U \in \mathcal{U}$, so their union has size $\leq (n - \ell)/2$. Thus, there is an edge $e \in I$ covering $W_1 \cup W_2$, namely $e \in \delta_I(W_1 \cup W_2)$. Note that the head of $e$ must be in $(W_1 \cup W_2)^* = W_1^* \cap W_2^*$ (the equality is by Lemma 2.1). This implies that $e \in \delta_I(W_1)$ or $e \in \delta_I(W_2)$, by Lemma 2.1(ii). Consequently, $e = e_1$ or $e = e_2$. Since the tail of each one of $e_1, e_2$ is in $W_1 \cap W_2$, so is the tail of $e$. As the head of $e$ is in $W_1^* \cup W_2^*$, we conclude that $e \in \delta_I(W_1) \cap \delta_I(W_2)$. This is a contradiction since $\delta_I(W_1) = \{e_1\}$, $\delta_I(W_2) = \{e_2\}$, and $e_1 \neq e_2$.

The proof of Theorem 2.3 is complete.

## 3 Algorithm for Rooted SND Augmentation (Proof of Theorem 1.4)

To avoid considering "mixed" cuts that contain both nodes and edges, we may assume that $st \notin E_J$ for all $t \in T$. One way to achieve this is to subdivide every edge $st \in E_J$ with $t \in T$ by a new node.

DEFINITION 3.1. *A node subset $X \subseteq V$ is* $t$-tight *for $t \in T$ if $t \in X$, $s \in X^*$, and $|\Gamma_J(X)| = \ell$; $X$ is* tight *if it is $t$-tight for some $t \in T$. A tight set is a* core *if it does not contain two inclusion minimal tight sets. Let $\mathcal{C}_J = \{C_1, \ldots, C_\nu\}$ denote the set of min-cores, and for $i = 1, \ldots \nu$ let $M_i$ be some max-core containing $C_i$. Let $\mathcal{M}_J = \{M_1, \ldots, M_\nu\}$.*

The families $\mathcal{C}_J$ and $\mathcal{M}_J$ can be computed in polynomial time using standard max-flow methods. By Menger's Theorem, $I$ is a feasible solution to an instance of Rooted SND Augmentation if, and only if, $I$ covers all tight sets (assuming $st \notin E_J$ for all $t \in T$). Thus our goal is to find such $I$ of low cost. Note that terminals not belonging to any min-core can be discarded, as any tight set containing such a terminal also contains a terminal that belongs to some minimal tight set. Hence from now and on we assume that every terminal belongs to some minimal tight set.

The following "sub-modular" and "posi-modular" properties of the function $\Gamma(\cdot) = \Gamma_J(\cdot)$ is well known, c.f., [17] and [30].

PROPOSITION 3.1. *For any $X, Y \subseteq V$ the following holds:*

$$|\Gamma(X)| + |\Gamma(Y)| \geq |\Gamma(X \cap Y)| + |\Gamma(X \cup Y)| \tag{3.2}$$

$$|\Gamma(X)| + |\Gamma(Y)| \geq |\Gamma(X \cap Y^*)| + |\Gamma(Y \cap X^*)| \tag{3.3}$$

$$(X \cap Y)^* = X^* \cup Y^*,\ (X \cup Y)^* = X^* \cap Y^* \tag{3.4}$$
*if equality holds in* (3.2)

$$(X \cap Y^*)^* = X^* \cup Y,\ (Y \cap X^*)^* = X \cup Y^* \tag{3.5}$$
*if equality holds in* (3.3)

LEMMA 3.1. *Let $X$ be $x$-tight and let $Y$ be $y$-tight. Then:*

(i) *If $x \in X \cap Y$ then $X \cap Y, X \cup Y$ are $x$-tight, and if $y \in X \cap Y$ then $X \cap Y, X \cup Y$ are $y$-tight. Furthermore, in both cases equality holds in* (3.2).

(ii) *If $x \in X \cap Y^*, y \in Y \cap X^*$ then $X \cap Y^*$ is $x$-tight, $Y \cap X^*$ is $y$-tight, and equality holds in* (3.3).

(iii) *If none of* (i),(ii) *holds then $y \in \Gamma(X)$ or $x \in \Gamma(Y)$.*

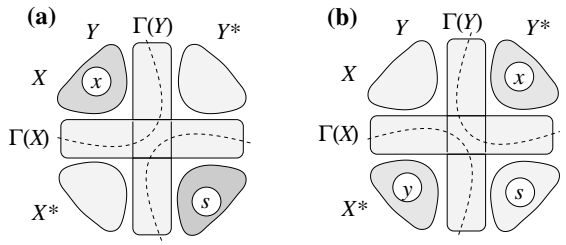


Figure 1: Illustration to the proof of Lemma 3.1.

*Proof.* Part (iii) is obvious, so we prove parts (i),(ii) using (3.2) and (3.3), see Figure 3.

If $x \in X \cap Y$ (the proof of the case $y \in X \cap Y$ is similar) then by (3.2) (see Figure 3(a)):

$$\ell+\ell = |\Gamma(X)|+|\Gamma(Y)| \geq |\Gamma(X\cap Y)|+|\Gamma(X\cup Y)| \geq \ell+\ell\ .$$

Hence equality holds everywhere, so $X \cap Y, X \cup Y$ are $x$-tight.

If $x \in X \cap Y^*$ and $y \in Y \cap X^*$ then by (3.3) (see Figure 3(b)):

$$\ell+\ell = |\Gamma(X)|+|\Gamma(Y)| \geq |\Gamma(X\cap Y^*)|+|\Gamma(Y\cap X^*)| \geq \ell+\ell\ .$$

Hence equality holds everywhere, so $X \cap Y^*$ is $x$-tight and $Y \cap X^*$ is $y$-tight.

In [14], a set-family $\mathcal{F}$ was called *uncrossable* if $X \cap Y, X \cup Y \in \mathcal{F}$ or $X - Y, Y - X \in \mathcal{F}$ for any $X, Y \in \mathcal{F}$. An edge $e$ was said to cover a set $X$ if $e$ has one endnode in $X$ and the other in $V - X$. In [14] is given a 2-approximation primal-dual algorithm that computes a cover of an uncrossable family $\mathcal{F}$. In our case, we use the following modified definition:

Definition 3.2. *We say that a subfamily $\mathcal{F}$ of tight sets is* uncrossable *if for any $X, Y \in \mathcal{F}$ at least one of the following holds: $X \cap Y, X \cup Y \in \mathcal{F}$ and* (3.4) *holds, or $X \cap Y^*, Y \cap X^* \in \mathcal{F}$ and* (3.5) *holds.*

It is known and easy to verify that the primal-dual algorithm of [14] can be adjusted to cover "setpair" families $\{\{X, X^*\} : X \in \mathcal{F}\}$, provided $\mathcal{F}$ is uncrossable in the sense of Definition 3.2. Such an algorithm can be implemented in polynomial time provided that for any edge set $I$ the minimal members of the residual family $\mathcal{F}_I$ of the members of $\mathcal{F}$ not covered by $I$ can be computed in polynomial time.

Unfortunately, the family of tight sets may not be uncrossable (in the sense of Definition 3.2), but we will show a method to decompose it into uncrossable families.

Lemma 3.2. *For any tight set $X$ and any $C_i \in \mathcal{C}_J$ either $C_i \cap X \cap T = \emptyset$ or $(C_i \cap T) \subseteq X$. Thus $C_i \cap C_j \cap T = \emptyset$ for any $i \neq j$.*

*Proof.* Otherwise, $C_i \cap X$ is tight, by Lemma 3.1 (i), contradicting the minimality of $C_i$.

Definition 3.3. *Let $T_i = T \cap C_i$ and let $\Gamma_i = \Gamma(M_i)$. We say that $M_i, M_j \in \mathcal{M}_J$ are* independent *if the sets $T_i \cap M_j^*, T_j \cap M_i^*$ are both nonempty.*

From Lemma 3.1 we have:

Corollary 3.1. *For any $i$ the set $M_i$ is unique. For any $i \neq j$, at least one of the following holds: $M_i, M_j$ are independent and thus $M_i \cap M_j^*, M_j \cap M_i^*$ are tight, or $T_i \subseteq \Gamma_j$, or $T_j \subseteq \Gamma_i$.*

Given a subfamily $\mathcal{M} \subseteq \mathcal{M}_J$, the subfamily of tight sets induced by $\mathcal{M}$ is

$$\mathcal{F}(\mathcal{M}) = \{X : X \subseteq M \in \mathcal{M}, X \text{ is tight}\}\ .$$

Lemma 3.3. *If $X, Y \subseteq M_i$ are tight then $X \cap Y, X \cup Y$ are also tight, and* (3.4) *holds for $X, Y$. If $M_i, M_j$ with $i \neq j$ are independent then for any tight $X \subseteq M_i$ and $Y \subseteq M_j$ the sets $X \cap Y^*, Y \cap X^*$ are tight and* (3.5) *holds for $X, Y$. Thus if the members of $\mathcal{M}$ are pairwise independent, then the family $\mathcal{F}(\mathcal{M})$ is uncrossable.*

*Proof.* The first statement follows from Lemma 3.1 (i). The second statement follows from Lemma 3.1 (ii) and the fact that if $T_i \cap M_j^* \neq \emptyset$ then $T_i \cap Y^* \neq \emptyset$ for any $Y \subseteq M_j$.

Lemma 3.4. *The family $\mathcal{M}_J$ can be partitioned into at most $2\ell + 1$ parts so that the members of each part are pairwise independent, and such a partition can be found in polynomial time.*

*Proof.* Construct an auxiliary directed graph $\mathcal{J}$ as follows. The node set of $\mathcal{J}$ is $\mathcal{M}_J$. Add an arc $M_iM_j$ if $T_i \subseteq \Gamma_j$. The maximum indegree of every node in $\mathcal{J}$ is $\leq \ell$. This implies that every subgraph of the underlying graph of $\mathcal{J}$ has a node if degree $\leq 2\ell$. A graph is $d$-degenerate if every its subgraph has a node of degree at most $d$. It is well known that any $d$-degenerate graph is $(d+1)$-colorable, and such coloring can be computed in polynomial time. Hence $\mathcal{J}$ is $(2\ell + 1)$-colorable, thus its node set can be partitioned into at most $2\ell + 1$ independent sets.

Lemma 3.5. *If $I$ covers $\mathcal{F}(\mathcal{M}_J)$ then the number of min-cores in $G + I$ is at most $\nu/2$.*

*Proof.* Every min-core of $J + I$ is a tight set in $J$. Thus by Lemma 3.2 and by the definition of $M_i$ every min-core $C$ of $J + I$ contains the terminals of at least 2 distinct min-cores $C_i, C_j$ of $J$; namely, $T_i, T_j \subseteq C$. As the min-cores of $J + I$ are also disjoint on the terminals, by Lemma 3.2, the statement follows.

Summarizing, we can find an edge set $I$ of cost $\leq 2(2\ell+1)\cdot \mathsf{opt}$ so that the number of min-cores in $G+I$ is $\leq \nu/2$. Such $I$ is a union of 2-approximate covers of $2\ell+1$ uncrossable families as in Lemma 3.4. We can apply this procedure iteratively, until no min-cores remain. The number of iterations is at most $1+\log\nu = O(\log|T|)$, and the overall cost over all iterations is $O(\ell\log|T|)\cdot\mathsf{opt}$.

Finally, we explain how to obtain an $O(k^3\log^2|T|)$-approximation for the case of node-costs. In [32] is given an $O(\log\nu)$-approximation algorithm for the problem of covering an uncrossable (in the sense of [14]) set-family $\mathcal{F}$ by edges, so that the weight of the endnodes of the edges is minimum, where $\nu$ is the number minimal members in $\mathcal{F}$. A slight modification of this algorithm that invokes an extra term of $O(k^2)$ applies also to cover setpair families $\{\{X,X^*\} : X\in\mathcal{F}\}$, provided $\mathcal{F}$ is uncrossable in the sense of Definition 3.2. Hence in the case of node-weights, instead of using the 2-approximation algorithm of [14], we use the $O(k^2\log\nu)$-approximation algorithm of [32] to obtain the required ratio.

The proof of Theorem 1.4, and thus also the proof of Theorem 2.1 is now complete.

## 4 Open problems

With each open problem we associate a (subjective) value, according to its "age", expected "hardness" of solution, and author's interest.

- (99\$) Does $k$-Connected Subgraph Augmentation admits a constant ratio approximation algorithm? As was mentioned in the Introduction, we conjecture that the ratio $O\left(\log\frac{n}{n-k}\right)$ obtained in this paper characterizes the problem.

- (99\$) One of the main open problems is determining the complexity status of $k$-Connected Subgraph Augmentation and $k$-Outconnected Subgraph Augmentation for the case of undirected graphs and $0,1$-costs. The currently best known approximation algorithms for these problems compute a solution of size roughly $\mathsf{opt}+k/2$, see [15, 16] (or [26]) and [31], respectively.

- (25\$) Does SND admits an approximation algorithm with ratio better than the trivial $O(n^2)$? Such algorithms exist for $0,1$-costs [24, 29].

- (40\$) Does undirected Rooted SND Augmentation admits a $\rho(k)$-approximation algorithm? We conjecture that the answer is positive; for $0,1$-costs, such an algorithms exist even for SND [29].

- (50\$) What is the approximability of undirected Rooted SND Augmentation? Does the problem admits a sublinear in $n$ algorithm? By [25], the problem is harder than Directed Steiner Tree.

- (60\$) Does *directed* Rooted SND Augmentation admits an approximation scheme with ratio $O(n^{\varepsilon}/\varepsilon^3)$, similar to the one given in [3] for the Directed Steiner Tree problem? We conjecture the answer is positive.

- (10\$) As was mentioned, even for SND with unit costs (namely, when every edge is allowed) we are not aware of any explicit construction that solves the problem exactly, like the Harary graphs for $k$-Connected Subgraph.

For additional open problems see [23].

## References


[1] V. Auletta, Y. Dinitz, Z. Nutov, and D. Parente. A 2-approximation algorithm for finding an optimum 3-vertex-connected spanning subgraph. *Journal of Algorithms*, 32(1):21–30, 1999.

[2] T. Chakraborty, J. Chuzhoy, and S. Khanna. Network design for vertex connectivity. In *STOC*, pages 167–176, 2008.

[3] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. *Journal of Algorithms*, 33:73–91, 1999.

[4] J. Cheriyan and R. Thurimella. Approximating minimum size $k$-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000.

[5] J. Cheriyan, S. Vempala, and A. Vetta. An approximation algorithm for the minimum-cost $k$-vertex connected subgraph. *SIAM Journal on Computing*, 32(4):1050–1055, 2003.

[6] J. Cheriyan, S. Vempala, and A. Vetta. Network design via iterative rounding of setpair relaxations. *Combinatorica*, 26(3):255–275, 2006.

[7] J. Cheriyan and A. Vetta. Approximation algorithms for network design with metric costs. In *STOC*, pages 167–175, 2005.

[8] J. Chuzhoy and S. Khanna. Algorithms for single-source vertex-connectivity. In *FOCS*, 2008. To appear.

[9] Y. Dinitz and Z. Nutov. A 3-approximation algorithm for finding optimum $4,5$-vertex-connected spanning subgraphs. *Journal of Algorithms*, 32(1):31–40, 1999.

[10] J. Fackharoenphol and B. Laekhanukit. An $O(\log^2 k)$-approximation algorithm for the $k$-vertex connected subgraph problem. In *STOC*, pages 153–158, 2008.

[11] A. Frank. Increasing the rooted-connectivity of a digraph by one. *Mathematical Programming*, 84(3):565–576, 1999.

[12] A. Frank and T. Jordán. Minimal edge-coverings of pairs of sets. *J. of Comb. Theory B*, 65:73–110, 1995.

[13] A. Frank and E. Tardos. An application of submodular flows. *Linear Algebra and its Applications*, 114/115:329–348, 1989.

[14] M. X. Goemans, A. V. Goldberg, S. Plotkin, D. B. Shmoys, E. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *SODA*, pages 223–232, 1994.

[15] B. Jackson and T. Jordán. A near optimal algorithm for vertex connectivity augmentation. In *ISAAC*, pages 313–325, 2000.

[16] B. Jackson and T. Jordán. Independence free graphs and vertex connectivity augmentation. *J. of Comb. Theory B*, 94(1):31–77, 2005.

[17] T. Jordán. On the optimal vertex-connectivity augmentation. *J. on Comb. Theory B*, 63:8–20, 1995.

[18] T. Jordán. A note on the vertex connectivity augmentation. *J. on Comb. Theory B*, 71(2):294–301, 1997.

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

[20] G. Kortsarz, R. Krauthgamer, and J. R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM Journal on Computing*, 33(3):704–720, 2004.

[21] G. Kortsarz and Z. Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37:75–92, 2003.

[22] G. Kortsarz and Z. Nutov. Approximating $k$-node connected subgraphs via critical graphs. *SIAM Journal on Computing*, 35(1):247–257, 2005.

[23] G. Kortsarz and Z. Nutov. Approximating minimum-cost connectivity problems. In T. F. Gonzalez, editor, Chapter 58 in *Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007.

[24] G. Kortsarz and Z. Nutov. Tight approximation algorithm for connectivity augmentation problems. *J. Comput. Syst. Sci.*, 74(5):662–670, 2008.

[25] Y. Lando and Z. Nutov. Inapproximability of survivable networks. In *APPROX-RANDOM*, pages 146–152, 2008.

[26] G. Liberman and Z. Nutov. On shredders and vertex-connectivity augmentation. *Journal of Discrete Algorithms*, 5(1):91–101, 2007.

[27] W. Mader. Endlichkeitsätze für $k$-critishe graphen. *Math. Ann.*, 229:143–153, 1977.

[28] W. Mader. On $k$-con-critically $n$-connected graphs. *J. of Comb. Theory B*, 86(2):296–314, 2002.

[29] Z. Nutov. Approximating node-connectivity augmentation problems. Manuscript.

[30] Z. Nutov. Approximating connectivity augmentation problems. In *SODA*, pages 176–185, 2005.

[31] Z. Nutov. Approximating rooted connectivity augmentation problems. *Algorithmica*, 44:213–231, 2005.

[32] Z. Nutov. Approximating steiner networks with node weights. In *LATIN*, pages 411–422, 2008.

[33] R. Ravi and D. P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 18:21–43, 1997.

[34] R. Ravi and D. P. Williamson. Erratum: an approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 34(1):98–107, 2002.