Network Design for Vertex Connectivity

Tanmoy Chakraborty [*]
Dept. of CIS
University of Pennsylvania
Philadelphia PA
tanmoy@seas.upenn.edu

Julia Chuzhoy
Toyota Technological Institute
Chicago, IL 60637
cjulia@tti-c.org

Sanjeev Khanna [†]
Dept. of CIS
University of Pennsylvania
Philadelphia PA
sanjeev@cis.upenn.edu

ABSTRACT

We study the survivable network design problem (SNDP) for vertex connectivity. Given a graph $G(V, E)$ with costs on edges, the goal of SNDP is to find a minimum cost subset of edges that ensures a given set of pairwise *vertex connectivity* requirements. When all connectivity requirements are between a special vertex, called the *source*, and vertices in a subset $T \subseteq V$, called *terminals*, the problem is called the *single-source* SNDP. Our main result is a randomized $k^{O(k^2)} \log^4 n$-approximation algorithm for single-source SNDP where k denotes the largest connectivity requirement for any source-terminal pair. In particular, we get a polylogarithmic approximation for any constant k. Prior to our work, no non-trivial approximation guarantees were known for this problem for any $k \geq 3$. We also show that SNDP is $k^{\Omega(1)}$-hard to approximate and provide an elementary construction that shows that the well-studied set-pair linear programming relaxation for this problem has an $\tilde{\Omega}(k^{1/3})$ integrality gap.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory.

Keywords

Approximation Algorithms, Hardness of Approximation, Network Design, Vertex Connectivity.

[*]Supported by NSF Award CCF-0635084.

[†]Supported in part by a Guggenheim Fellowship, an IBM Faculty Award, and by NSF Award CCF-0635084.

1. INTRODUCTION

A fundamental problem in network design is to construct a minimum cost network that satisfies some specified connectivity requirements between pairs of vertices. One of the most well-studied problems in this framework is the *survivable network design problem* (SNDP) where we are given an undirected graph with costs on its edges, and a connectivity requirement r_{ij} for each pair i, j of vertices. The goal is to find a minimum cost subset of edges that ensures that there exist r_{ij} disjoint paths for each pair i, j of vertices. When the paths are required to be only edge-disjoint, we get the edge-connectivity variant of SNDP, denoted by EC-SNDP. If we require the paths to be vertex-disjoint, we obtain the vertex-connectivity variant of SNDP, denoted by VC-SNDP. When all $r_{ij} \in \{0, k\}$ for some integer k, we will refer to these problems as simply k-edge connectivity and k-vertex connectivity, respectively.

An interesting special case of the k-edge connectivity and k-vertex connectivity problems is the single-source version where we are given a special vertex s called the *source*, and a collection $T \subseteq V$ of *terminals*. The connectivity requirements are $r_{st} = k$ for all $t \in T$, and all other requirements are 0. We will refer to these versions as the single-source k-edge connectivity and single-source k-vertex connectivity. Even when $k = 1$, the single-source version of the k-edge connectivity and the k-vertex connectivity problems corresponds to the extensively studied minimum Steiner tree problem. The Steiner tree problem is known to be APX-hard [5].

It is thus not surprising that SNDP and its variants have received a considerable attention in the approximation algorithms community. In fact, a study of these problems is tied to the development of several powerful algorithmic design paradigms. Most known algorithmic results for these problems are based on rounding of a natural "cut-based" LP relaxations of these problems. In one of the first applications of the primal-dual approach to approximation algorithms design, Agarwal, Klein, and Ravi [1] gave a 2-approximation algorithm for 1-edge connectivity (and hence 1-vertex connectivity) using a primal-dual approach. Subsequent research (see, for instance, [16, 14]) successfully extended the primal-dual approach to higher connectivity values, culminating in an $O(\log k)$-approximation algorithm for EC-SNDP [15]; here k denotes the largest connectivity requirement between any pair. Shortly afterwards, a beautiful result of Jain [17] developed an iterative rounding scheme to show that EC-SNDP is 2-approximable, precisely matching a lower bound of 2 on the integrality gap of the LP relaxation. Jain's approach is based on showing that in any basic feasible solution of the cut-based LP relaxation, at least one variable is assigned a value of $\frac{1}{2}$ or larger. The algorithm then rounds up the value of such a variable to 1, and iterates until all variables become $\{0,1\}$. Jain's approach may be viewed as a generalization of the classical paradigm of showing existence of $1/p$-integral solutions for some integer p, and using this to bound the integrality gap of the relaxation to be at most p.

In a sharp contrast to edge-connectivity problems, progress has been hard to come by for vertex connectivity problems. Fleischer, Jain, and Williamson [11], building on the iterative rounding approach of [17], showed that the set-pair relaxation of Frank and Jordan [12] gives a 2-approximation for VC-SNDP when each connectivity requirement $r_{ij} \in \{0,1,2\}$. This result improves upon a previous 3-approximation algorithm due to Ravi and Williamson [23] for the same problem. For higher connectivity values, when $k \geq 3$, no non-trivial approximation ratios have been known for VC-SNDP even in the single source setting. We note however that some special cases of the problem are better understood. The iterative rounding technique has been used to give a 2-approximation for arbitrary r_{ij} values for a special case of VC-SNDP known as the *element connectivity* problem [11, 8]. A special case of k-vertex connectivity where we require all pairs of vertices in G to be k-connected, called the k-vertex-connected spanning subgraph problem, has also been widely studied. Cheriyan *et al.* [7, 8] gave an $O(\log k)$-approximation algorithm for this case when $k \leq \sqrt{n/6}$, and an $O(\sqrt{n/\epsilon})$-approximation algorithm for $k \leq (1-\epsilon)n$. The first result is based on a graph-theoretic characterization due to Mader [22], while the second algorithm is based on iterative rounding. For large k, Kortsarz and Nutov [21] improved the preceding bound to an $O(\ln k \cdot \min\{\sqrt{k}, \frac{n}{n-k}\ln k\})$-approximation. Fakcharoenphol and Laekhanukit [10] recently improved this bound to $O(\log n \log k)$, and obtained $O(\log^2 k)$-approximation for $k < n/2$. For $k \leq 7$, $\lceil \frac{k+1}{2} \rceil$-approximation are known for k-vertex-connected spanning subgraph problem ([9, 4, 20]).

Frank and Tardos [13] gave a polynomial time algorithm for finding a minimum cost k-outconnected subdigraph of a directed graph which has been used to obtain a 2-approximation algorithm for single-source k-vertex connectivity, when T contains all vertices in G except the source [18]. Better approximation ratios have also been obtained for variants of vertex-connectivity problems assuming uniform cost or metric cost on the edges (see, for example, [18, 20, 6]).

On the hardness front, Kortsarz *et al.* [19] recently showed that in a strong contrast to k-edge connectivity, the k-vertex connectivity problem is $2^{\log^{1-\epsilon} n}$-hard to approximate for any $\epsilon > 0$, when k is polynomially large in n. In addition, they showed an $\Omega(\log n)$-hardness of approximation for the single-source k-vertex connectivity problem, also for the case when k is a polynomial function of n.

Results.

Our main result is summarized in the following theorem:

Theorem 1 *There is a randomized polynomial-time algorithm for the* single-source k-vertex connectivity *problem that produces a* $k^{O(k^2)} \log^4 n$*-approximate solution.*

In particular, for $k \leq O(\sqrt{\log\log n / \log\log\log n})$, we get a poly-logarithmic approximation algorithm.

We note that an $O(|T|)$-approximation is straightforward to achieve for this problem as follows. For each $t \in T$, optimally solve the minimum cost k-connected subgraph problem to connect s to t, and take a union of all $|T|$ solutions. To our knowledge, no better approximation guarantees were known previously for any $k \geq 3$. Two immediate corollaries of our result are as follows:

- There is a $k^{O(k^2)} \log^4 n$-approximation for a generalization of the single source problem where $r_{st} \in \{1, 2 \ldots k\}$ for each terminal $t \in T$.
- The subset k-vertex connectivity problem, where the goal is to find a minimum cost subset of edges that makes a given subset of vertices T pairwise k-vertex connected, can be approximated to within a $k^{O(k^2)} \log^4 n$ factor in randomized polynomial time.

Our next result is hardness of approximation for the general version of k-vertex connectivity, where the connectivity requirements can be for arbitrary pairs of vertices. Kortsarz *et. al.* [19] have shown that for any $\epsilon > 0$, the k-vertex connectivity problem is hard to approximate within a factor of $2^{\log^{1-\epsilon} n}$, when k is polynomially large in n. We strengthen this result to show a $k^{\Omega(1)}$ hardness of approximation for any sufficiently large connectivity requirement k.

Theorem 2 *There exists an $\epsilon > 0$ and a constant $k_0 > 0$, such that the* k-vertex connectivity *problem is hard to approximate to within a factor of k^ϵ for all $k > k_0$. For constant k, the result holds under the assumption that* $\mathsf{P} \neq \mathsf{NP}$. *For super-constant k, the result holds if* $\mathsf{NP} \not\subseteq \mathsf{DTIME}\left(n^{O(\log k)}\right)$.

We also give a simple construction that shows that the integrality gap of the well-studied set-pair relaxation for this problem [12] is polynomially large in k. We note that a similar integrality gap can be shown by suitably modifying the hardness construction of Kortsarz *et al.* [19] for the k-vertex connectivity problem.

Theorem 3 *The set-pair relaxation for* k-vertex connectivity *has an integrality gap of* $\tilde{\Omega}(k^{1/3})$.

Techniques.

We start with a brief overview of the algorithmic ideas underlying Theorem 1. Our starting point is the cut-based set-pair relaxation for the problem. Using the cut-flow duality, one may view the relaxation as providing a fractional vertex-disjoint flow of value k between each terminal and the source. This flow solution is non-aggregating in that an edge e that is allocated a capacity of x_e, can be used by any number of terminals to route up to x_e units of flow each.

We view the paths as originating at the terminals and terminating at the source. The fractional solution can be well-approximated by randomized rounding when the flow paths connecting distinct terminals to the source s do not overlap extensively. But if many paths traverse an edge e (that is, the edge e has high "congestion"), then independent rounding for each terminal ends up "over-charging" against the cost of edge e in the fractional solution. However, in the latter case, we show that the support of the fractional solution provides high connectivity among the many terminals that share the edge e. This fact is exploited by our algorithm as follows. We show that there exists a subset T'

containing at most half the terminals, such that all terminals in $T \setminus T'$, connect to s by paths that either satisfy the property that every edge on the path has low congestion, or by paths that contain terminals of T'. We can then reduce the problem to that of (recursively) solving single-source k-vertex connectivity on the subset T' of terminals.

A key idea underlying our algorithm is the following combinatorial lemma that may have other applications. Let P be any collection of paths such that each path $p \in P$ originates at a *distinct* terminal, and all paths in P go through a common vertex v. Let $H(P)$ be the graph induced by the paths in P, and let $f(k) = k^{O(k^2)}$. Then one can always identify a path $p \in P$, such that if we delete any subset X of $(k-1)$ vertices in $H(P)$, none of which lies on the path p, the terminal of p remains connected to at least $|P|/f(k)$ other terminals in graph $H(P)$. Our algorithm uses this lemma to reduce congestion on the edges by *randomly rerouting* paths for some of the terminals to other terminals.

For our hardness result, we take the bipartite graph given by the Raz verifier [24], and modify it to create an instance of k-vertex connectivity. Our construction is similar to that of [19], and the essence of our result lies in reducing the connectivity requirement k. In both our construction as well as that of [19], for every source-sink pair there are $k-1$ vertex-disjoint paths of length 2 and cost 0, and the last path, of non-zero cost, corresponds to a pair of consistent answers by the provers in the Raz verifier. Restricting the structure of this last path relies on the fact that it cannot use the $k-1$ vertices belonging to the other $k-1$ paths. A careful analysis of the construction of [19] shows that in order to preserve the restricted structure of this special path, it is enough to block a smaller number of such vertices, which enables us to reduce the value of k.

Our integrality gap construction builds on similar ideas to the hardness of approximation proof, where instead of a bipartite graph given by the Raz verifier, we use a randomly generated bipartite graph.

2. PRELIMINARIES

Problem Statement.

The input to the VC-SNDP problem is an undirected n-vertex graph $G = (V, E)$, with edge costs $c_e \geq 0$ for all $e \in E$, and integer connectivity requirements r_{ij} for every pair of vertices (i, j). The goal is to choose a minimum-cost subset E' of edges, such that in the graph $G' = (V, E')$, every pair of vertices (i, j) has at least r_{ij} vertex-disjoint paths connecting them. The special case where each $r_{ij} \in \{0, k\}$ is known as the k-vertex connectivity problem.

The single-source k-vertex connectivity problem is a special case of k-vertex connectivity, in which there is one special vertex, source $s \in V$, and a collection of terminals $T \subseteq V \setminus \{s\}$. The connectivity requirements are $r_{st} = k$ for all terminals $t \in T$, and $r_{ij} = 0$ for all other pairs.

The subset k-vertex connectivity is another special case of k-vertex connectivity, in which there is a subset T of terminals, and the connectivity requirements are $r_{tt'} = k$ for all $t, t' \in T$, and all other requirements are 0.

The following simple lemma, whose proof appears in appendix, allows us to assume that edge costs are polynomially bounded, without any loss of generality.

Lemma 2.1 *With at most a constant factor loss in approximation ratio, we can reduce any instance of the* k-vertex connectivity *problem to one where all edge costs are in* $[0..n^6]$.

LP Relaxation and Flow-Paths Decomposition.

We start with a linear programming formulation for the k-vertex connectivity problem. Let $\mathcal{R}$ denote the set of all pairs (W_L, W_R) of subsets of vertices (i.e., $W_L, W_R \subseteq V$) such that $W_L \cap W_R = \emptyset$, $|V \setminus (W_L \cup W_R)| \leq k$, and for some pair (i, j) with $r_{ij} = k$, $i \in W_L$ and $j \in W_R$. For any such pair $W = (W_L, W_R)$, let $\delta(W)$ denote the set of all edges with one endpoint in W_L and the other endpoint in W_R. Notice that a feasible integral solution must contain at least $k - (n - |W_L| - |W_R|)$ of the edges in $\delta(W)$. In our linear program, for each edge $e \in E$, there is an indicator variable x_e showing whether or not e is in the solution.

$$\begin{aligned} \text{(LP)} \qquad & \min \textstyle\sum_{e \in E} c_e x_e \\ \text{s.t.} \qquad & \\ & \textstyle\sum_{e \in \delta(W)} x_e \geq k - (n - |W_L| - |W_R|) \\ & \qquad \forall\, W = (W_L, W_R) \in \mathcal{R} \qquad (1) \\ & 0 \leq x_e \leq 1 \qquad \forall e \in E \end{aligned}$$

We focus here on the single-source k-vertex connectivity problem, where $\mathcal{R}$ consists of set-pairs (W_L, W_R) such that $s \in W_L$, $W_R \cap T \neq \emptyset$.

Let x^* be an optimal solution to (LP), and let OPT denote the cost of this solution. In our next step we define, for each terminal t, a collection of k-tuples of paths, denoted by D_t. Each k-tuple $H \in D_t$ consists of k vertex disjoint paths connecting t to s and is associated with a coefficient $\rho_H \geq 0$, where $\sum_{H \in D_t} \rho_H = 1$. For a k-tuple $H \in D_t$, we say that edge $e \in H$ iff e belongs to one of the k paths in H (notice that e may belong to at most one such path). We require that for all $e \in E$ and $t \in T$, $\sum_{\substack{H \in D_t: \\ e \in H}} \rho_H \leq x_e$. We will use the following lemma whose proof follows from standard arguments, and is deferred to the full version of the paper.

Lemma 2.2 *Let* $G^* = (V^*, E^*)$ *be any graph with two special vertices* $s, t \in V$ *and values* $0 \leq x_e \leq 1$ *for edges* $e \in E$ *satisfying constraint (1) for all* $W = (W_L, W_R)$ *with* $s \in W_L, t \in W_R, W_R \cap W_L = \emptyset$ *and* $|V^* \setminus (W_R \cup W_L)| \leq k$. *Then we can find, in polynomial time, a collection* D_t *of* k*-tuples of vertex disjoint paths connecting* s *and* t, *and coefficients* $\rho_H > 0$ *for all* $H \in D_t$, *such that* $\sum_{H \in D_t} \rho_H = 1$ *and for each* $e \in E$, $\sum_{\substack{H \in D_t: \\ e \in H}} \rho_H \leq x_e$.

3. Single-Source k-Vertex Connectivity Algorithm

A straightforward way to round the LP-solution, using Lemma 2.2, is as follows. For each terminal $t \in T$, choose a k-tuple $H \in D_t$ with probability ρ_H. Output the union of all the paths in the chosen k-tuples. The main problem with this approach is that there might be "congested" edges: informally, an edge is congested if it participates in k-tuples $H \in D_t$ for many terminals $t \in T$. For a congested edge e, the probability that it belongs to the solution may be much higher than x_e. However, the advantage of such congested edges is that many of the terminals fractionally connect to them. Assume that e is congested and belongs to some k-tuple $H \in D_t$ for some terminal t. Let $p \in H$ be the specific path in H on which e lies. If k-tuple H is chosen for t, then, instead of buying all edges on path p, we can buy the portion of the path that lies between t and e, and then buy more edges to connect to other terminals $t' \in T$ which are also fractionally connected to e. We can then randomly sample a sufficiently large subset $T' \subseteq T$ of terminals and solve the problem recursively for T'. Each terminal $t \notin T'$ will either connect directly to s by edges we have bought in the current iteration, or will connect via terminals in T' (or do a combination of both). In order for this approach to work, we need to ensure that the cost paid in each iteration is not much higher than OPT, and that the recursive solution to the problem with set T' of terminals, together with the edges chosen in the current iteration indeed define a feasible solution to the entire problem.

3.1 Algorithm Description

We now present the algorithm formally. Throughout, we will use a parameter $\alpha = 16k^{10k^2}\log n$. The algorithm works in iterations. The input to iteration i is a subset $T_i \subseteq T$ of terminals, and the output is a subset $E_i \subseteq E$ of edges, together with a subset $T_{i+1} \subseteq T_i$ of terminals that become the input for the next iteration. We set $T_1 = T$. An execution of iteration i is successful iff the following conditions hold:

C1. $|T_{i+1}| \leq |T_i|/2$.

C2. $\sum_{e\in E_i} c_e \leq O(\alpha^2 \log n)\mathsf{OPT}$.

C3. For every $t \in T_i \setminus T_{i+1}$, for every set $X \subseteq V \setminus \{s,t\}$ of $(k-1)$ vertices, there is a path p connecting t to some vertex in $T_{i+1} \cup \{s\}$ such that p only contains edges in E_i, and p does not contain any vertex from X.

The algorithm stops when $|T_i| \leq O(\alpha^2 \log n)$ for some iteration i. At this point, for each terminal $t \in T_i$, we choose the cheapest possible collection of k vertex disjoint paths from t to s, which can be found by a simple min-cost flow computation. Let E_i denote the union of edges participating in all such paths. Notice that since $|T_i| \leq O(\alpha^2 \log n)$, the cost of E_i is at most $O(\alpha^2 \log n)\mathsf{OPT}$. The output of the algorithm is $\cup_{j\leq i} E_j$.

Theorem 4 *If every iteration is successful, then the algorithm outputs a feasible solution whose cost is bounded by* $k^{O(k^2)} \log^4 n \cdot \mathsf{OPT}$.

Proof. The cost of every successful iteration is at most $O(\alpha^2 \log n)\mathsf{OPT} = k^{O(k^2)} \log^3 n \cdot \mathsf{OPT}$. If every iteration is successful, then we will have at most $\log n$ iterations in total. Thus, the solution cost is at most $k^{O(k^2)} \log^4 n \cdot \mathsf{OPT}$.

It now only remains to show that the algorithm outputs a feasible solution. We do so by induction on the recursion depth. Consider the iteration i and set T_i of corresponding terminals. If i is the last iteration, then clearly the solution is feasible with respect to T_i. Assume now that i is not the last iteration. It is enough to show that if E^* is any feasible solution for the problem defined by T_{i+1}, then $E_i \cup E^*$ is a feasible solution for the problem defined by T_i.

Let G^* be the graph defined on the set V of vertices, whose edge set is $E_i \cup E^*$. Clearly, if $t \in T_{i+1}$, then by assumption, there are k vertex disjoint paths connecting t to s. Suppose now that for some $t \in T_i \setminus T_{i+1}$, there do not exist k vertex disjoint paths connecting t to s. Then there is a collection $X \subseteq V \setminus \{s,t\}$ of $k-1$ vertices, such that when X is removed from the graph, t and s do not belong to the same connected component. But since iteration i is successful, there must be at least one path p, which consists of edges in E_i and does not contain vertices in X connecting t to some $t' \in T_{i+1} \cup \{s\}$. Since t and s lie in different components, $t' \neq s$. Moreover, since E^* is a feasible solution to the problem defined by T_{i+1}, there is at least one path p', which consists of edges in E^* and does not contain vertices of X, connecting t' to s. Therefore, t must be connected to s even after X is removed from G^*. A contradiction. $\square$

3.2 Iteration Description

Fix some i, and let $T_i \subseteq T$ be a subset of terminals, which is the input to the current iteration. We now describe an algorithm for iteration i and we show that our algorithm produces a successful iteration with probability at least 1/2. Notice that given a set E_i of edges and a set T_{i+1} of terminals, conditions C1–C3 can be checked in polynomial time for $k \leq n$. Therefore, if an execution of an iteration is not successful, we can repeat the algorithm until we have a successful execution. Overall, our algorithm always returns an $k^{O(k^2)} \log^4 n$-approximate solution, and it runs in expected polynomial time. For any path p in the graph, let $\mathsf{cost}(p) = \sum_{e\in p} c_e$ be the cost of p. Each iteration consists of three phases: *path truncation, path sampling,* and *terminal sampling.*

Path Truncation: Let P denote the set of all the paths that appear in k-tuples $H \in D_t$ for all $t \in T_i$. Notice that each path $f \in P$ connects some terminal t to the source s. Let $t(f)$ denote the terminal that is connected to s by path f. In the path-truncation step, for each path $f \in P$, we define a prefix $p(f)$. This is a portion of path between $t(f)$ and some vertex $v(f)$ that lies on f (possibly $v(f) = s$). Additionally, for each $f \in P$, we define a subset $R(f)$ of paths, such that:

P1. If $v(f) \neq s$ then $|R(f)| = \alpha$. Otherwise, $R(f) = \emptyset$

P2. For every path $p \in R(f)$, one endpoint of p is a terminal $t(p)$ and the other endpoint lies on the prefix $p(f)$. All terminals $t(p)$ for $p \in R(f)$ are distinct.

P3. For each $p \in R(f)$, $\mathsf{cost}(p) \leq 2\mathsf{cost}(p(f))$.

P4. If $v(f) \neq s$, and X is any collection of $k-1$ vertices that do not lie on f, then at least α/k^{10k^2-1} of the paths in $R(f)$ do not contain any vertex of X.

Consider some edge e in the graph and terminal t. We say that edge e *belongs to* t iff there is $H \in D_t$, such that for some path $f \in H$, the prefix $p(f)$ contains e. Additional condition that the path-truncation step guarantees:

P5. Each edge $e \in E$ belongs to $O(\alpha \log n)$ terminals $t \in T_i$.

The path-truncation step is the heart of the algorithm and we defer its description to the next section. We now complete the description of an iteration assuming we can guarantee properties P1–P5.

Path Sampling: The second stage, namely path-sampling, is performed as follows. For each terminal $t \in T_i$, we select a k-tuple $H \in D_t$ of paths with probability ρ_H. For each $f \in H$ thus selected, we add the edges on prefix $p(f)$, and edges on all paths in $R(f)$ to E_i. Let P' be the set of all the prefixes $p(f)$ that we selected in this step. Thus, every terminal t contributes exactly k prefixes to P': the k prefixes of the paths in $H \in D_t$ that has been chosen by the algorithm. The lemma below bounds the cost of this stage.

Lemma 3.1 *The expected cost of E_i is $O(\alpha^2 \log n)\mathsf{OPT}$.*

Proof. Notice first that since property P3 holds, the cost of edges in E_i is at most 2α times the cost of P'. We now bound the expected cost of P'.

Consider some edge $e \in E$. Since property P5 holds, edge e belongs to at most $O(\alpha \log n)$ terminals in T_i. For each such terminal t, we have that $\sum_{H \in D_t: e \in H} \rho_H \le x_e$. Therefore, if e belongs to t, the probability that t chooses a k-tuple H that contains e is at most x_e. In total, the probability that e belongs to some prefix in P' is $O(\alpha \log n)x_e$. Thus the expected cost of P' is $O(\alpha \log n)\mathsf{OPT}$, and the expected cost of E_i can thus be bounded by $O(\alpha^2 \log n)\mathsf{OPT}$. ☐

Therefore, condition C2 holds with high probability.

Terminal Sampling: The third, and the final stage of an iteration is terminal sampling. Each terminal $t \in T_i$ is selected to be in T_{i+1} independently with probability 1/8. By Markov's inequality, the probability that more than $|T_i|/2$ terminals are selected is at most 1/4. Therefore, condition C1 holds with probability 3/4. Finally, we need to show that C3 holds with high probability. We conclude the analysis in the following theorem.

Theorem 5 *Assume the path truncation step can be executed in polynomial time, guaranteeing properties P1–P5. Then the above algorithm produces a successful iteration with probability at least* 1/2.

Proof. As observed above, condition C1 holds with probability at least 3/4 and condition C2 holds with high probability. It is enough to show that condition C3 holds with probability at least 9/10. We need the following lemma:

Lemma 3.2 *Let $X \subseteq V \setminus \{s\}$ be any set of $(k-1)$ vertices in the graph, and let $G_i = (V, E_i)$. Then after X is removed from G_i, each connected component that does not contains s, contains either no terminals or more than α/k^{10k^2-1} terminals.*

Proof. Assume otherwise. Let X be such a set of $(k-1)$ vertices, and let t be any terminal in a connected component that does not contain s. We show that this component must contain more than α/k^{10k^2-1} terminals. Let $H \in D_t$ be the k-tuple of vertex disjoint paths that has been chosen for t. At least one of these paths does not contain vertices in X. Let f be this path. Then $v(f) \ne s$ (otherwise s belongs to the same connected component as t). Consider the set $R(f)$ of paths. All edges that lie on these paths belong to E_i. According to property P4, at least α/k^{10k^2-1} of the paths in $R(f)$ do not contain any vertex of X. Recall that each such path connects some vertex on $p(f)$ to a distinct terminal $t' \in T_i$. Therefore, the connected component of t must contain more than α/k^{10k^2-1} terminals. ☐

Consider now any set of $(k-1)$ vertices $X \subseteq V\setminus\{s\}$. When vertices in X are removed from G_i, we obtain a collection of connected components. Let $\mathcal{C}$ be the collection of resulting components that do not contain s but contain one or more terminals. We say that a *bad event* $B(X, C)$ happens for $C \in \mathcal{C}$, iff no terminal of C is chosen to T_{i+1}. Since C contains at least α/k^{10k^2-1} terminals, the probability that $B(X, C)$ happens is at most

$$\left(1 - \frac{1}{8}\right)^{\frac{\alpha}{k^{10k^2-1}}} \le e^{-\frac{\alpha}{8k^{10k^2-1}}} \le e^{-2k \log n},$$

since $\alpha = 16k^{10k^2} \log n$. For each possible set X of $(k-1)$ vertices, we have at most n such components. Therefore, using the union bound, the probability that $B(X, C)$ happens for any X, C is at most $e^{-2k \log n} n^k < 1/10$.

It is easy to see that if the bad event $B(X, C)$ does not happen for any X and C, then condition C3 holds: fix some terminal $t \in T_i \setminus T_{i+1}$ and a set $X \subseteq V\setminus\{s, t\}$ of $k-1$ vertices. Consider the connected component C to which t belongs if we remove X from the graph $G_i = (V, E_i)$. If this connected component contains s, then we are done. Otherwise, since $B(X, C)$ does not happen, the connected component must contain at least one terminal $t' \in T_{i+1}$.

Putting it all together, we get a randomized $k^{O(k^2)} \log^4 n$-approximation for single-source vertex k-connectivity. This completes the proof of Theorem 1, except for the description of the path truncation stage.

3.3 Path Truncation Stage

We now focus on the path truncation step, and show an efficient algorithm that, given the set P of paths containing the union of paths in $H \in D_t$ for all $t \in T_i$, produces, for each path $f \in P$, prefix $p(f)$ and set $R(f)$ of paths, such that properties P1–P5 hold.

Recall that for each $f \in P$, one endpoint of the prefix $p(f)$ is fixed to be the terminal $t(f)$, and the other endpoint is denoted by $v(f)$. Throughout the algorithm, for each path f, we maintain the vertex $v(f)$. At the beginning, $v(f) = s$, but as algorithm proceeds, $v(f)$ can become any vertex on path f, and then we will think of f as being trimmed at $v(f)$.

A collection F of paths is called *independent* iff for all $f \in F$, the terminals $t(f)$ are distinct. Throughout the algorithm, for each $f \in P$, if $v(f) \ne s$, we will have a set $R(f)$ of α independent paths that connect α distinct terminals to vertices on $p(f)$, where the cost of every path in $R(f)$ is at most $2\mathsf{cost}(p(f))$. Moreover, for any $(k-1)$-tuple X of vertices none of which lie on f, at least α/k^{10k^2-1} of the paths in $R(f)$ do not contain any vertex of X. Hence properties P1–P4 are maintained throughout the execution of the algorithm. The problem is with property P5: some edges may have too many terminals using them to connect to s.

At each step of the algorithm, we will identify one "heavy" vertex v, which is used by many of the terminals, and then we will trim at least one of the paths f that goes through v (by setting $v(f) = v$). This is how we make progress. In the end, when no more heavy vertices exist in the graph, we have a solution obeying properties P1–P5. We now describe the path truncation stage more formally.

At the beginning of the algorithm, for each $f \in P$, $v(f) = s$ and $R(f) = \emptyset$. Consider some vertex v, and let $F(v)$ be the set of all prefixes $p(f)$ for $f \in P$ that go through v (i.e., $v(f) \neq v$ and v lies between $t(f)$ and $v(f)$ on the path f). For each prefix $p(f) \in F(v)$, let $c_v(f)$ be the cost of the portion of $p(f)$ starting from $t(f)$ and ending at v. For each $j : 0 \leq j \leq O(\log n)$, let $F_j(v) \subseteq F(v)$ be the subset of prefixes $p(f)$ with $2^j \leq c_v(f) < 2^{j+1}$. We say that v is j-*heavy* iff the set T'_j of all terminals from which paths in F_j originate has size $|T'_j| > \alpha$. Our goal is to show that if v is j-heavy, then at least one path in F_j can be trimmed at v (and a corresponding set $R(f)$ of α paths can be defined). In each iteration, we select one j-heavy vertex (for some j), and perform the trimming. The algorithm ends when no j-heavy vertices remain for any j.

Claim 1 *If no j-heavy vertices v remain for any j, then the solution has property P5.*

Proof. We will show that if no j-heavy vertices exist for any j, then every edge belongs to at most $16\alpha \log n$ terminals, thus satisfying the property P5. Assume by way of contradiction that some edge $e = (u, v)$ belongs to more than $16\alpha \log n$ terminals $t \in T_i$. Let F be the set of $16\alpha \log n$ prefixes $p(f)$ containing e, that originate from distinct terminals. Let $F' \subseteq F$ be the set of prefixes $p(f)$ that contain e and where v lies between u and $t(f)$. We can assume w.l.o.g. that the set of terminals T' where paths F' originate contains at least $8\alpha \log n$ terminals. As before, for each prefix $p(f) \in F'$, we define $c_v(f)$ to be the cost of the portion of $p(f)$ between $t(f)$ and v. Note that Lemma 2.1 implies that $c_v(f) < n^8$ for each $f \in F$. For each $j : 1 \leq j \leq 8 \log n$, let $F_j \subseteq F$ be the subset of prefixes $p(f)$ for which $2^{j-1} \leq c_v(f) < 2^j$. Let T'_j be the subset of terminals where paths F_j originate. Then for at least one j, $|T'_j| > \alpha$ must hold, and thus v is j-heavy. $\square$

We now focus on one j-heavy vertex v and the corresponding set F_j of paths. Let $P' \subseteq F_j$ be a collection of exactly $\alpha + 1$ independent paths. The following lemma shows that one of the paths in P' can be trimmed at v.

Lemma 3.3 *Let $P' = \{p_1, p_2, \ldots, p_{\alpha+1}\}$ be a set of paths where each path p_i connects a distinct terminal t_i to a fixed vertex v. Then there exists a path $p \in P'$ and a collection R of paths such that:*

- *the set R contains α paths, where for every path $p' \in P' \setminus \{p\}$, R contains the portion of the path p' that lies between the terminal of p' and the first vertex appearing on both p and p'.*
- *for any subset X of at most $k - 1$ vertices, none of which lie on the path p, at least $\frac{\alpha}{2\cdot(2k)^{4k^2+1}} \geq \frac{\alpha}{k^{10k^2-1}}$ of the paths in R do not contain vertices of X.*

Proof. Fix an integer $\beta \in \{1, \ldots, (4k^2)\}$. We say that a vertex u is *type-β congested* iff more than $\alpha/(2k)^\beta$ of the paths in P' go through it. Every path in $p' \in P'$ assigns a *type-β token* to the first $\alpha/(2k)^\beta$ distinct paths that it meets on its way to the vertex v. If path p' meets several paths simultaneously, we order them arbitrarily. The total number of type-β tokens that get distributed is at least $\alpha^2/(2k)^\beta$. We consider two cases, namely, when there exists a $\beta \in \{1, \ldots, (4k^2)\}$ such that some path receives more than $\alpha/(2k)^{\beta-1}$ type-β tokens, and when every path receives at most $\alpha/(2k)^{\beta-1}$ type-β tokens for each $\beta \in \{1, \ldots, (4k^2)\}$.

If there exists a path p that receives more than $\alpha/(2k)^{\beta-1}$ type-β tokens for some $\beta \in \{1, \ldots, (4k^2)\}$, then we claim that p is the desired path. The set R of paths contains then, for every path $p' \in P \setminus \{p\}$, the portion of p' between the terminal of p' and the first vertex of p on p'. At least $\alpha/(2k)^{\beta-1}$ paths in P' have the property that they meet the path p for the first time before passing through a type-β congested vertex (note that the vertex where these paths meet p may be a type-β congested vertex itself). Let $Q \subseteq R$ be the set of corresponding sub-paths. Thus for any vertex x that does not belong to p, at most $\alpha/(2k)^\beta$ paths in Q contain x. In particular, for any set X of up to $(k - 1)$ vertices, at most $\frac{\alpha(k-1)}{(2k)^\beta}$ paths in Q contain vertices of X. The lemma follows since $|Q| \geq \alpha/(2k)^{\beta-1}$.

We now focus on the second case. We say a path p is type-β *good* iff it receives at least $\alpha/(2\cdot(2k)^\beta)$ type-β tokens. Since we allocate at least $\alpha^2/(2k)^\beta$ type-β tokens in total, and each path receives at most $\alpha/(2k)^{\beta-1}$ tokens of type-β, there must be at least $\alpha/(4k - 1)$ type-β good paths, for each $\beta \in \{1, \ldots, (4k^2)\}$. It then follows that there exists a path p that is good for at least k distinct types. Let $1 \leq i_1 \leq i_2 \leq, \ldots, \leq i_k \leq 4k^2$ be the k distinct indices such that the path p is type-i_j good for $1 \leq j \leq k$, and let R be the set of α paths, containing, for each one of the remaining paths $p' \in P'$, the sub-path lying between its terminal and the first vertex belonging to p. Moreover, let $Q_{i_j} \subseteq R$ denote the subset whose corresponding paths gave a type-i_j token to the path p. Recall that $|Q_{i_j}| \geq \alpha/(2 \cdot (2k)^{i_j})$ for all $1 \leq j \leq k$.

We now claim that for any subset X of $(k - 1)$ vertices, none of which reside on p, at least a $(1/k)$-fraction of the paths in one of the sets Q_{i_j} does not contain vertices of X. For any $x \in X$, let $\varphi(x)$ denote the least integer ℓ such that x is a type-ℓ congested vertex. Thus, if the number of paths going through x is η, then $\alpha/(2k)^{\varphi(x)} < \eta \leq \alpha/(2k)^{\varphi(x)-1}$. Clearly, a vertex $x \in X$ cannot lie on a path in Q_{i_j} if $\varphi(x) \leq i_j$, since a path in Q_{i_j} reaches p before going through a type-$\varphi(x)$ congested vertex. Moreover, x lies on at most $\alpha/(2k)^{\varphi(x)-1}$ paths from any Q_{i_j} when $\varphi(x) > i_j$. By pigeonhole principle, there exists an index i_j such that $\varphi(x) \neq i_j + 1$ for each $x \in X$. For such Q_{i_j}, we have that $|Q_{i_j}| \geq \alpha/(2 \cdot (2k)^{i_j})$, while any vertex in X lies on at most $\alpha/(2k)^{i_j+1}$ paths in Q_{i_j}. Thus any vertex in X belongs to at most a $(1/k)$-fraction of paths in Q_{i_j}. The claim follows. $\square$

In the path truncation stage, we start by setting $p(f) = s$ and $R(f) = \emptyset$ for all $f \in P$. While there exists a j-heavy vertex v for some j, we use Lemma 3.3 to find path p and set R of paths. Note that p and R can be computed in polynomial time. Let $f \in P$ be such that p is the prefix of f. We then set $v(f) = v$ and $R(f) = R$. It is easy to see that conditions P1-P4 hold throughout the algorithm. The algorithm stops when no j-heavy vertices exist for any j. Claim 1 implies that properties P1–P5 hold at the end of the algorithm.

4. Algorithms for Generalized Single-Source and Subset k-Connectivity

We consider the following generalization of single-source k-vertex connectivity: there is a source s and a set of terminals T, and a positive integer $r_{st} \leq k$ for every $t \in T$, which is the connectivity requirement between s and t.

As a corollary to Theorem 1, we obtain approximation algorithms for the generalized single-source k-vertex connectivity, and also for subset k-vertex connectivity.

Theorem 6 *There is a randomized polynomial-time algorithm for the generalized* single-source k-vertex connectivity *problem that produces a $k^{O(k^2)} \log^4 n$-approximate solution.*

PROOF. For every integer i, $1 \le i \le k$, define $T^i = \{t \mid r_{st} = i\}$. Find a solution E^i to single-source k-vertex connectivity with $k = i$, s as the source and T^i as the set of terminals. $\cup_{1 \le i \le k} E^i$ gives a solution to the generalized problem, with the required approximation guarantee. □

Theorem 7 *There is a randomized polynomial-time algorithm for the* subset k-vertex connectivity *problem that produces a $k^{O(k^2)} \log^4 n$-approximate solution.*

PROOF. Let T be the set of terminals that are required to be k-connected to each other. If $|T| \le k$, then for every $t \in T$, run the single-source k-vertex connectivity algorithm with t as the source and $T \setminus \{t\}$ as the set of terminals. If $|T| > k$, then choose any subset $T' \subset T$ containing k terminals, and then for every $t \in T'$, run the single-source k-vertex connectivity algorithm with t as the source and $T \setminus \{t\}$ as the set of terminals. The union of these solutions gives a solution to the subset k-vertex connectivity problem with the required approximation guarantee. □

5. A k^ϵ-Hardness for k-Vertex Connectivity

In this section we prove Theorem 2. Recall that in k-vertex connectivity, we are given a graph $G = (V, E)$ with costs c_e on edges, and a collection of source-sink pairs (s_i, t_i). The goal is to find a minimum cost subset $E' \subseteq E$ of edges, such that for each source-sink pair (s_i, t_i), there are k vertex-disjoint paths from s_i to t_i in $G' = (V, E')$.

The Starting Point.

We will perform the reduction from the 3SAT(5) problem. In this problem we are given a 3SAT formula φ on n variables and $5n/3$ clauses. Each clause contains 3 distinct literals and each variable participates in exactly 5 different clauses. We say that φ is a YES-INSTANCE if it is satisfiable. We say that φ is a NO-INSTANCE with respect to some parameter δ, iff no assignment satisfies more than δ-fraction of clauses. The following well-known theorem follows from the PCP theorem [3, 2].

Theorem 8 *There is a constant $\delta : 0 < \delta < 1$, such that it is NP-hard to distinguish between* YES-INSTANCES *and* NO-INSTANCES *of the 3SAT(5) problem.*

We use the Raz verifier for 3SAT(5) with ℓ parallel repetitions. This is an interactive proof system, in which two provers try to convince the verifier that the input 3SAT(5) formula φ is satisfiable. The verifier chooses, independently at random, ℓ clauses $C_1, \ldots, C_\ell$, and for each $i : 1 \le i \le \ell$, a variable x_i participating in clause C_i is chosen. The verifier then sends one query to each one of the two provers, while the query to the first prover consists of the indices of the variables $x_1 \ldots, x_\ell$, and the query to the second prover contains the indices of the clauses $C_1, \ldots, C_\ell$. The first prover returns an assignment to variables $x_1, \ldots, x_\ell$. The second prover is expected to return an assignment to all the variables in clauses $C_1, \ldots, C_\ell$, which must satisfy the clauses. Finally, the verifier checks that the answers of the two provers are consistent, i.e., for each $i : 1 \le i \le \ell$, the assignment to x_i, returned by the first prover, is identical to the assignment to x_i, obtained by projecting the assignment to the variables of C_i, returned by the second prover, onto x_i. (We assume that the answers sent by the second prover always satisfy the clauses appearing in its query). The following theorem is obtained by combining the PCP theorem with the parallel repetition theorem [24].

Theorem 9 ([2, 24]) *There exists a constant $\gamma > 0$, such that:*

- *If φ is a* YES-INSTANCE, *then there is a strategy of the provers, for which the acceptance probability is* 1.
- *If φ is a* NO-INSTANCE, *then for any strategy of the provers, the acceptance probability is at most $2^{-\gamma\ell}$.*

We denote the set of all the random strings of the verifier by R, $|R| = (5n)^\ell$, and the sets of all the possible queries of the first and the second prover by Q_1 and Q_2 respectively, $|Q_1| = n^\ell$, $|Q_2| = (5n/3)^\ell$, and set $Q = Q_1 \cup Q_2$. For each query $q \in Q$, let $A(q)$ be the collection of all the possible answers to q (if q is a query of the second prover, then $A(q)$ only contains answers that satisfy all the clauses of the query). Let $A = 2^\ell$, $A' = 7^\ell$. Then for each $q \in Q_1$, $|A(q)| = A$, and for each $q' \in Q_2$, $|A(q')| = A'$. Given a random string $r \in R$, let $q_1(r), q_2(r)$ be the queries sent to the first and the second prover respectively, when the verifier chooses r.

Construction.

Given any $k \ge k_0$ for some sufficiently large constant k_0, we now construct a gap instance for k-vertex connectivity. We will set the Raz verifier repetition parameter ℓ to be $\Theta(\log k)$, and show that the cost of an optimal solution when φ is a NO-INSTANCE is $2^{\Omega(\ell)}$ times larger than the optimal cost when φ is a YES-INSTANCE. Our construction is as follows.

The set of vertices is $V = V_1 \cup V_2$. We start by describing the set V_1, and define V_2 later on. For every query $q \in Q$, there is a vertex $u(q) \in V_1$. For each $q \in Q$, $a \in A(q)$, there is a vertex $v(q, a) \in V_1$. For each random string $r \in R$, there are two vertices (that form a source-sink pair) $s(r), t(r) \in V_1$. Thus, $V_1 = \{u(q) \mid q \in Q\} \cup \{v(q, a) \mid q \in Q, a \in A(q)\} \cup \{s(r), t(r) \mid r \in R\}$.

The set E of edges is the union of three edge sets E_0, E_1, E_2. Edges in E_0 are called *special edges*, and are defined as follows. For every $q \in Q$ and $a \in A(q)$, we add a special edge between $u(q)$ and $v(q, a)$. The cost of this edge is C_1 if $q \in Q_1$, and it is C_2 otherwise. We set $C_1 = 5^\ell$ and $C_2 = 3^\ell$. Notice that $|Q_1|C_1 = (5n)^\ell = |Q_2|C_2$.

The set E_1 of edges consists of *non-special edges of type 1*, which are defined as follows. For each random string $r \in R$, if $q_1 = q_1(r)$ and $q_2 = q_2(r)$, we add edges from $s(r)$ to $u(q_1)$, from $u(q_2)$ to $t(r)$, and for every pair a_1, a_2 of matching answers to q_1 and q_2, we add an edge between $v(q_1, a_1)$ and $v(q_2, a_2)$. Edges of this type have cost 0.

Finally, the set E_2 consists of *non-special edges of type 2*. To define set E_2 of edges, we first construct an auxiliary graph $H = (R, U)$ of random strings. Each random string

$r \in R$ is represented by a vertex in H. There is an edge between r and r' iff $q_1(r) = q_1(r')$ or $q_2(r) = q_2(r')$. Thus, the degree of every vertex in H is $2^{O(\ell)}$. The distance between $r, r' \in R$ is the length of the shortest path from r to r' in H. For each random string $r \in R$, we define a set $X(r) \subseteq V_1$ of vertices, as follows:

- If r' is within distance at most 2 from r, and $r' \neq r$, add $s(r')$ and $t(r')$ to $X(r)$.
- Let $r' \in R \setminus \{r\}$, such that $q_1(r') = q_1(r)$. Then for each $a \in A(q_2(r'))$, add $v(q_2(r'), a)$ to $X(r)$. Similarly, let $r'' \in R \setminus \{r\}$, such that $q_2(r'') = q_2(r)$. Then for each $a \in A(q_1(r''))$, add $v(q_1(r''), a)$ to $X(r)$.

The following properties of sets $X(r)$ will be used later:

A1. Let $x \in \{s(r), t(r)\}$ and $y \in \{s(r'), t(r')\}$ for some $r, r' \in R$. Then $x \in X(r')$ iff $y \in X(r)$.

A2. Let $v(q, a) \in X(r)$ for some $r \in R$, $q \in Q$, $a \in A(q)$, and let $r' \in R$, such that $q_1(r') = q$ or $q_2(r') = q$. Then $s(r'), t(r') \in X(r)$ and $s(r), t(r) \in X(r')$.

The reason why A2 is true is that if $v(q, a) \in X(r)$ then there is some $r'' \in R$ for which $q_1(r'') = q$ or $q_2(r'') = q$, and r, r'' are within distance 1 from each other. But then r', r'' share a query q, and thus r, r'' are within distance 2 from each other.

Let $\Delta_r = |X(r)|$, and let $\Delta = \max_r \{\Delta_r\}$. For each $r \in R$, we create a set $Y(r)$ of $\Delta - \Delta_r$ distinct vertices, which are also added to $X(r)$. These vertices define the set V_2: $V_2 = \{Y(r) \mid r \in R\}$. Note that at this point $|X(r)| = \Delta = 2^{O(\ell)}$ for all $r \in R$. Finally, we add for each $r \in R$, *non-special edges of type* 2 from $s(r)$ to every vertex in $X(r)$, and from every vertex in $X(r)$ to $t(r)$. The costs of these edges is set to 0.

In the final graph, the set of vertices is $V_1 \cup V_2$, and the set of edges is $E = E_0 \cup E_1 \cup E_2$, where E_0 are the special edges, and E_1, E_2 are non-special edges of type 1 and 2 respectively. We set $k = \Delta + 1$ to be the demand for every source-sink pair. Notice that the construction size is $N = n^{O(\ell)}$.

Yes-Instance Analysis.

Assume φ is the Yes-Instance, and let $g(q)$ for every $q \in Q$ denote the "correct" answer to query q, such that under the strategy defined by g the verifier accepts with probability 1. We claim that $E'_0 \cup E_1 \cup E_2$, where $E'_0 \subseteq E_0$ contains the special edge $(u(q), v(q, g(q)))$ for each $q \in Q$, is a feasible solution for the k-vertex connectivity instance. Solution cost is $C_{YI} = C_1|Q_1| + C_2|Q_2|$. Consider any random string r, with $q = q_1(r)$, $q' = q_2(r)$. The demand of $\Delta + 1$ between $s(r)$ and $t(r)$ is satisfied as follows. For every vertex $v \in X(r)$, we have a path from $s(r)$ to v to $t(r)$. All these paths use type-2 non-special edges, and they are vertex disjoint. Additionally, if $a = g(q)$ and $a' = g(q')$, then we have path $(s(r) \to u(q) \to v(q, a) \to v(q', a') \to u(q') \to t(r))$.

No-Instance Analysis.

Assume that φ is a No-Instance. Consider any solution E'. We can assume w.l.o.g. that $E' = E'_0 \cup E_1 \cup E_2$, where $E'_0 \subseteq E_0$. We now claim the following.

Claim 2 *Let $r \in R$ be any random string, and let $q_1 = q_1(r), q_2 = q_2(r)$. Then there is a pair a, a' of matching answers to q_1 and q_2 respectively such that edges $(u(q_1), v(q_1, a))$ and $(u(q_2), v(q_2, a'))$ belong to E'_0.*

Before proving the claim, we show that it suffices to claim the desired hardness gap.

Lemma 5.1 *The cost of E' is at least $2^{\gamma\ell/3} C_{YI}$.*

Proof. Assume otherwise. For each $q \in Q$, we define set $B_q \subseteq A(q)$ of answers as follows: $a \in B_q$ iff the edge $(u(q), v(q, a)) \in E'$. We say that a query q is *good* iff $|B_q| \leq 8 \cdot 2^{\gamma\ell/3}$. By averaging arguments, at least 3/4 of the queries in Q_1 and at least 3/4 of the queries in Q_2 must be good. Let R' be the set of all random strings for which both $q_1(r)$ and $q_2(r)$ are good. Then $|R'| \geq |R|/2$. We now define the strategy for the two provers as follows. For each good query $q \in Q$, the corresponding prover will choose one of the at most $8 \cdot 2^{\gamma\ell/3}$ answers in B_q. Notice that the probability that a random string $r \in R'$ is chosen is at least 1/2. From Claim 2, for each $r \in R$, there is a pair $a \in B_{q_1(r)}$, $a' \in B_{q_2(r)}$ of matching answers. Therefore, for each $r \in R'$, the probability that a matching pair of answers is chosen is at least $1/(64 \cdot 2^{2\gamma\ell/3})$. In total, the probability that the verifier accepts is at least $1/(128 \cdot 2^{2\gamma\ell/3}) > 2^{-\gamma\ell}$ when ℓ (and hence k) is sufficiently large. This is a contradiction since φ is a No-Instance. $\square$

The hardness of approximation gap that we obtain is $2^{\Omega(\ell)}$, while $k = 2^{O(\ell)}$ and the construction size is $N = n^{O(\ell)}$. Therefore, we obtain a k^ϵ-hardness of approximation for some constant $0 < \epsilon < 1$. For constant k, this holds under the assumption that $\mathsf{P} \neq \mathsf{NP}$. For super-constant k we use the assumption that $\mathsf{NP} \not\subseteq \mathsf{DTIME}\left(n^{O(\log k)}\right)$.

We now prove Claim 2.

Proof. Fix a random string r, and let $q_1 = q_1(r), q_2 = q_2(r)$. Consider the $\Delta + 1$ vertex-disjoint paths connecting $s(r)$ to $t(r)$. Our main claim is that at least one such path has to use two special edges $(u(q_1), v(q_1, a))$, $(u(q_2, a'), u(q_2))$, where a and a' are matching answers to q_1 and q_2, respectively. This will finish the proof of the claim.

Consider the $\Delta + 1$ vertex-disjoint paths connecting $s(r)$ to $t(r)$. Discard those paths that use vertices in $X(r)$. There must be at least one remaining path, P, and it is not allowed to use vertices in $X(r)$.

Let $S(r)$ be the following set of vertices:

$$S(r) = \{s(r), t(r), u(q_1), u(q_2)\} \cup \{v(q, a) \mid q \in \{q_1, q_2\}, a \in A(q)\}$$

We will use the following claim:

Claim 3 *Let e be any edge incident on a vertex in $S(r)$. Then the other end-point of e belongs to the set $S(r) \cup X(r)$.*

Proof. We consider all three types of edges. Recall that edges in E_0 connect, for each query $q \in Q$, vertex $u(q)$ to vertices $v(q, a)$ for $a \in A(q)$. Clearly, any such edge that is incident on a vertex in $S(r)$ has both endpoints in $S(r)$.

We now consider edges in E_1. Recall that such edges connect, for each $r' \in R$, $s(r')$ to $u(q_1(r'))$ and $t(r')$ to $u(q_2(r'))$. Any such edge that is incident on $s(r)$ or $t(r)$ has its other endpoint in $S(r)$. An edge in E_1 that is incident on $u(q_1)$ or $u(q_2)$ has $s(r')$ or $t(r')$ as its other endpoint, for some r' which is within distance 1 from r in the graph H of random strings. Therefore, $s(r'), t(r') \in X(r)$. Additionally, edges in E_1 connect, for each random string r' and pair a, a' of matching answers to $q_1(r')$ and $q_2(r')$, respectively, $v(q_1(r'), a)$ to $v(q_2(r'), a')$. Consider any such edge that is incident on vertices in $S(r)$. If one endpoint is $v(q_1, a)$ for some $a \in A(q_1)$, and the other endpoint is not in $S(r)$, then it must be $v(q_2', a')$ for some $q_2' \neq q_2$, $a' \in A(q_2')$, such that for some random string r', $q_1(r') = q_1$ and $q_2(r') = q_2'$. But then $v(q_2', a') \in X(r)$.

Similarly, if one endpoint of the edge is $v(q_2, a)$ for some $a \in A(q_2)$, and the other endpoint is not in $S(r)$, then it must be $v(q_1', a')$ for some $q_1' \neq q_1$, $a' \in A(q_1')$, such that for some random string r', $q_2(r') = q_2$ and $q_1(r') = q_1'$. But then $v(q_1', a') \in X(r)$.

Finally, we consider edges in E_2. Assume that such an edge is incident on some vertex $y \in S(r)$. Let y' be the other endpoint of this edge, and assume that $y' \notin S(r)$. Two causes are possible for the existence of edge (y, y'): either $y' \in X(r)$, and then we are done; or $y \in X(r')$ for some $r' \in R$. In the latter case, if $y \in \{s(r), t(r)\}$ then $y' \in \{s(r'), t(r')\}$ must hold, and by property A1, $y \in X(r)$. Otherwise, $y = v(q, a)$ for some $q \in \{q_1, q_2\}$ and $a \in A(q)$. In this case, $y' \in \{s(r'), t(r')\}$ must hold and by property A2, $y' \in X(r)$. $\square$

Since path P begins and ends in $S(r)$ and does not contain any vertex in $X(r)$, Claim 3 implies that P only contains vertices in $S(r)$, and hence it has to be of the form: $(s(r) \rightarrow u(q_1) \rightarrow v(q_1, a) \rightarrow v(q_2, a') \rightarrow u(q_2) \rightarrow t(r))$, where a and a' are matching answers to queries q_1, q_2. Therefore a pair $(u(q_1, v(q_1, a)), (u(q_2), v(q_2, a')$ of special edges where a and a' are matching answers for q_1 and q_2, respectively, must belong to the solution.

6. Integrality gap for k-Vertex Connectivity

We prove here Theorem 3, namely, an integrality gap of $\tilde{\Omega}(k^{1/3})$ for the set-pair LP relaxation given in Section 2.

We start by constructing a random bipartite graph $G(V_1 \cup V_2, E)$ where V_1 and V_2 denote the left and the right partition of vertices respectively. Each of the sets V_1, V_2 consists of b blocks of vertices with each block containing exactly L vertices. Let $A_1, A_2, ..., A_b$ denote the blocks of vertices in V_1, and similarly, let $B_1, B_2, ..., B_b$ denote the block of vertices in V_2. The edge set E is obtained by adding a *random perfect matching* between each pair of blocks A_i, B_j where $1 \leq i, j \leq b$. We now modify G to obtain our final instance $G'(V', E')$. Initially, $V' = V_1 \cup V_2$ and $E' = E$. For each block A_i of vertices, we add a new vertex u_i that is connected by an edge of cost 1 to every vertex in A_i. Similarly, for each block B_j of vertices, we add a new vertex v_j that is connected by an edge of cost 1 to every vertex in B_j. We refer to these edges of unit cost as *special edges*. In addition, for each pair of blocks A_i, B_j, we now add a source-sink pair of vertices $s(i,j)$-$t(i,j)$ to V'. We add edges of cost 0 from $s(i,j)$ to u_i and from $t(i,j)$ to v_j.

We now define a set of vertices $X(i,j)$ for every pair i, j. $X(i,j)$ consists of all vertices in $V_1 \cup V_2$ except $A_i \cup B_j$, and all source and sink vertices except $s(i,j)$ and $t(i,j)$. We connect each $s(i,j)$ to every vertex in $X(i,j)$ with edges of cost 0. Similarly, each $t(i,j)$ is connected by an edge of cost 0 to every vertex in $X(i,j)$. Finally, all edges in E (the matching edges) are assigned a cost of 0. Total number of vertices in G' is $N = 2b(L+1) + 2b^2$. We now set a connectivity requirement for each $s(i,j)$-$t(i,j)$ pair to be $k = N + 1 - 2(b + L) = |X(i,j)| + 1$. Note that each $s(i,j)$-$t(i,j)$ pair is already $|X(i,j)| = (k-1)$-connected in the graph G' if we include all edges of cost 0 in the solution.

Fractional Solution.

Consider the fractional solution that assigns fraction 1 to all 0-cost edges, and fraction $1/L$ to all the edges of cost 1. This solution has cost $2b$. We now argue the feasibility of this solution.

Consider two disjoint sets W_L and W_R of vertices, containing $s(i,j)$ and $t(i,j)$ respectively, for some i, j. Let $Q = V \setminus (W_L \cup W_R)$, $|Q| = q$. We need to show that the fractional solution satisfies the inequality $\sum_{e \in \delta(W)} x_e \geq k - q$ where $W = (W_L, W_R)$.

There are $|X(i,j) \setminus Q|$ vertex disjoint paths from $s(i,j)$ to $t(i,j)$ using only 0-cost edges, that do not intersect Q, and hence contribute at least 1 each to $\sum_{e \in \delta(W)} x_e$. If Q is not a subset of $X(i,j)$, then $|X(i,j) \setminus Q| \geq |X(i,j)| - (q-1) = k - q$, and the inequality is satisfied.

Now suppose $Q \subseteq X(i,j)$. Then, the above paths still contribute $k - q - 1$ to $\sum_{e \in \delta(W)} x_e$. Let $S(i,j) \subseteq V$ consist of $A_i, B_j, u_i, v_j, s(i,j)$ and $t(i,j)$. Then $S(i,j) \subseteq W_L \cup W_R$. The contribution to $\sum_{e \in \delta(W)} x_e$ by the subgraph induced by $S(i,j)$ is at least the smallest edge cut that separates $s(i,j)$ from $t(i,j)$ in this subgraph, which is 1. Hence the inequality is satisfied.

Integral solution.

Consider any integral solution of cost less than $\gamma b/2$ where γ is a parameter to be specified later. It means that less than $\gamma b/2$ special edges are chosen in the integral solution. W.l.o.g., we assume that the solution contains all edges that have 0 cost. Each special edge is incident on some block A_i or B_j. Clearly, at most $b/2$ blocks can have more than γ special edges chosen in the solution. We focus on the remaining blocks, at least $b/2$ of them on each side that have fewer than γ special edges chosen. We say that a vertex in A_i or B_j is *selected* if the special edge incident on it is present in the integral solution.

It is easy to see that every edge incident on a vertex in $S(i,j)$ has its other end-point in the set $S(i,j) \cup X(i,j)$ where $S(i,j)$ is defined to be $A_i \cup B_j \cup \{u_i, v_j, s(i,j), t(i,j)\}$. Thus for each $s(i,j) - t(i,j)$ pair, the construction ensures that the connectivity requirement is satisfied if and only if there is a path from $s(i,j)$ to $t(i,j)$ of length 5 that passes through u_i, a vertex $w_1 \in A_i$, a vertex $w_2 \in B_j$, and v_j, in that order. This is equivalent to saying that the w_1 and w_2 are selected, and that w_1 and w_2 are adjacent in the random matching between A_i and B_j.

Consider any pair i, j such that both A_i and B_j have less than γ selected vertices each. There are at least $b^2/4$ such pairs. The probability that a fixed vertex $w_1 \in A_i$ is matched with a fixed vertex $w_2 \in B_j$ by the random perfect matching is $1/L$. Taking the union bound over all pairs of chosen vertices $w_1 \in A_i$ and $w_2 \in B_j$, we get that the probability that there is a pair of matched selected vertices is less than γ^2/L. For the assignment to be feasible, it should satisfy the above condition for every pair i, j. These events are independent, since the random matchings between different pairs of blocks are independent. Hence, the probability that any such integral assignment is feasible, is less than $(\gamma^2/L)^{b^2/4}$.

The number of integral assignments of cost bounded by $\gamma b/2$ and containing all zero-cost edges is at most $\binom{2bL}{\gamma b/2}$. To rule out that there always exists an integral solution of cost at most $(\gamma b)/2$, it suffices to show that $\binom{2bL}{\gamma b/2}.(\gamma^2/L)^{b^2/4} < 1$. It is easy to verify that this condition is satisfied if we choose $L = b^2$ and $\gamma \leq b/10 \log b$. Thus we have $k = \Theta(b^3)$, and the integrality gap is $\Omega(\gamma) = \Omega(k^{1/3}/\log k)$.

7. ACKNOWLEDGEMENTS

We are grateful to Chandra Chekuri for his valuable comments.

8. REFERENCES

[1] A. Agrawal, P. N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM Journal of Computing*, 24(3):440–456, 1995.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[3] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[4] 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.

[5] M. Bern and P. Plassmann. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[6] J. Cheriyan, T. Jordan, and Z. Nutov. On rooted node-connectivity problems. *Algorithmica*, 30(3):353–375, 2001.

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

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

[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. Fakcharoenphol and B. Laekhanukit. An $O(\log^2 k)$-approximation algorithm for the k-vertex connected subgraph problem. In *STOC 2008, to appear.*

[11] L. Fleischer, K. Jain, and D. P. Williamson. Iterative rounding 2-approximation algorithms for minimum cost vertex connectivity problems. *Journal of Computer and System Sciences*, 72(5):838–867, 2006.

[12] A. Frank and T. Jordan. Minimal edge-coverings of pairs of sets. *Journal of Combinatorial Theory, Series B*, 65(1):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. Goemans and D. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms, D. Hochbaum, Ed., PWS*, 1997.

[15] M. X. Goemans, A. V. Goldberg, S. A. Plotkin, D. B. Shmoys, É. Tardos, and D. P. Williamson. Improved approximation algorithms for network design problems. In *Proceedings of the fifth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 223–232, 1994.

[16] M. X. Goemans, M. Mihail, V. Vazirani, and D. P. Williamson. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995.

[17] K. Jain. Factor 2 approximation algorithm for the generalized steiner network problem. In *Proceedings of the thirty-ninth annual IEEE Foundations of Computer Science (FOCS)*, pages 448–457, 1998.

[18] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21(2):434–450, 1996.

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

[20] G. Kortsarz and Z. Nutov. Approximating node connectivity problems via set covers. *Algorithmica*, 37(2):75–92, 2003.

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

[22] W. Mader. Endlichkeitsätze für k-kritische graphen (german). *Mathematische Annalen*, 229:143–153, 1977.

[23] R. Ravi and D. P. Williamson. An approximation algorithm for minimum-cost vertex-connectivity problems. *Algorithmica*, 18(1):21–43, 1997.

[24] R. Raz. A parallel repetition theorem. *SIAM Journal of Computing*, 27(3):763–803, 1998.

APPENDIX

Proof of Lemma 2.1: Let U_0 be the set of edges whose cost is 0. We can assume w.l.o.g. that for all $e \notin U_0$, $c_e \geq 1$, by multiplying all edge costs by an appropriate large integer. Let i^* be the smallest integer such that the set of all edges with costs less than n^{i^*} forms a feasible solution to our problem. We can assume w.l.o.g. that $i^* > 0$. Therefore, the optimal solution cost OPT is bounded by: $n^{i^*-1} \leq \mathsf{OPT} < n^2 \cdot n^{i^*}$. It follows that in any optimal solution, (a) an edge with cost greater than n^{i^*+2} is never used, and (b) total contribution to OPT by edges of cost n^{i^*-4} or less does not exceed $n^{i^*-2} \leq \mathsf{OPT}/n$. We now transform our instance as follows: every edge of cost n^{i^*-4} or less is assigned a cost of 0, all edges with cost greater than n^{i^*+2} are removed from the graph, the cost of each remaining edge e is set to be $\lceil \frac{c_e}{n^{i^*-4}} \rceil$. Thus in the modified graph, all new non-zero edge costs are integers between 1 and n^6. Clearly, a β-approximate solution for the transformed instance defines an $O(\beta)$-approximate solution for the original instance.

□