Algorithms for Single-Source Vertex Connectivity

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

Sanjeev Khanna*
University of Pennsylvania
Philadelphia PA 19103
sanjeev@cis.upenn.edu

August 27, 2008

Abstract

In the Survivable Network Design Problem (SNDP) the goal is to find a minimum cost subset of edges that satisfies a given set of pairwise connectivity requirements among the vertices. This general network design framework has been studied extensively and is tied to the development of major algorithmic techniques. For the edge-connectivity version of the problem, a 2-approximation algorithm is known for arbitrary pairwise connectivity requirements. However, no non-trivial algorithms are known for its vertex connectivity counterpart. In fact, even highly restricted special cases of the vertex connectivity version remain poorly understood.

We study the single-source k-vertex connectivity version of SNDP. We are given a graph $G(V, E)$ with a subset T of terminals and a source vertex s, and the goal is to find a minimum cost subset of edges ensuring that every terminal is k-vertex connected to s. Our main result is an $O(k \log n)$-approximation algorithm for this problem; this improves upon the recent $2^{O(k^2)} \log^4 n$-approximation. Our algorithm is based on an intuitive rerouting scheme. The analysis relies on a structural result that may be of independent interest: we show that any solution can be decomposed into a disjoint collection of multiple-legged spiders, which are then used to re-route flow from terminals to the source via other terminals.

We also obtain the first non-trivial approximation algorithm for the vertex-cost version of the same problem, achieving an $O(k^7 \log^2 n)$-approximation.

1 Introduction

In the Survivable Network Design problem (SNDP) we are given a graph $G = (V, E)$ with costs on edges and integral connectivity requirements $r_{u,v} \geq 0$ for all $u, v \in V$. The goal is to find a minimum-cost subset $E' \subseteq E$ of edges, such that every pair u, v of vertices has at least $r_{u,v}$ disjoint paths connecting them in the graph induced by E'. There are two basic versions of SNDP: in the edge connectivity version (EC-SNDP), the $r_{u,v}$ paths connecting u and v are required to be edge disjoint, while in the vertex connectivity version (VC-SNDP) they are required to be vertex disjoint. We denote by n the number of vertices in the graph, and by k the largest connectivity requirement, $k = \max_{u,v} \{r_{u,v}\}$.

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

This general framework is very versatile, and several classical optimization problems, such as minimum spanning tree and minimum Steiner tree are captured as special cases of SNDP. Consequently, the problem has received considerable attention, and in fact the study of SNDP is linked to the development of several fundamental paradigms in algorithm design. Agrawal, Klein and Ravi [1] showed a 2-approximation algorithm for the restricted version of SNDP where $r_{u,v} \in \{0, 1\}$ for all u, v (notice that for this case VC-SNDP and EC-SNDP are equivalent). This result is among the first applications of the *primal-dual* paradigm to approximation algorithms, and this approach was later successfully used to design algorithms for higher connectivity values for EC-SNDP [18, 16], eventually leading to an $O(\log k)$-approximation [17]. The best current approximation ratio of 2 is achieved by an *iterative* LP-rounding algorithm due to Jain [19]. Both the primal-dual schema and the iterative rounding technique have been extensively used in approximation algorithm design.

It is not hard to see that EC-SNDP can be cast as a special case of VC-SNDP. Since we have not found any references in the literature, for sake of completeness, we provide the proof of this fact in the Appendix. (We note that Galil and Italiano [15] showed this relationship between edge and vertex versions of the minimum k-connected subgraph problem, a special case of SNDP where for each $u, v \in V$, $r_{u,v} = k$). Despite the fact that the edge version has been extensively studied and is well understood today, little is known about VC-SNDP. To the best of our knowledge, no approximation algorithm is known for this problem, outside for the trivial algorithm, that finds, for each $u, v \in V$, the cheapest collection of $r_{u,v}$ vertex disjoint paths and outputs the union of these paths. On the hardness side, Kortsarz *et. al.* [22] showed that, in sharp contrast to the edge connectivity version, VC-SNDP is hard to approximate up to $2^{\log^{1-\epsilon} n}$ factor for any $\epsilon > 0$, when k is polynomially large in n. This has been recently improved by Chakraborty *et. al.* [7] to a k^{ϵ}-hardness of approximation for all $k > k_0$, where ϵ, k_0 are fixed constants. The restricted special case where all connectivity requirements $r_{u,v} \in \{0, 1, 2\}$ was shown to have a 2-approximation algorithm by Fleischer *et. al.* [12]. Their algorithm is based on an application of the iterative rounding technique of Jain [19] to the set-pair LP-relaxation of Frank and Jordan [13]. Our current lack of understanding of VC-SNDP is perhaps best highlighted by the following state of affairs. When each connectivity requirement $r_{u,v} \in \{0, 3\}$, the best known approximation factor is polynomially large while nothing more than APX-hardness is known on the hardness side.

In this paper we focus on the *single-source version* of VC-SNDP, where we are given a set $T \subseteq V$ of terminals and a special vertex $s \in V \setminus T$ called the *source*. The only non-zero connectivity requirements are between the terminals and the source, i.e., $r_{s,t} > 0$ for all $t \in T$ and all other requirements are 0. When $r_{s,t} = k$ for all $t \in T$, we refer the problem as the *single-source k-vertex connectivity* problem.

Even for this restricted case of VC-SNDP, little has been known until recently. A trivial $O(n)$ approximation can be achieved by connecting each terminal $t \in T$ to the source by cheapest collection of k vertex disjoint paths, that can be found via min-cost flow computations. Until recently, even for constant values of $k \geq 3$, only an $O(n)$-approximation was known with no super-constant hardness bounds. On the other hand, the current best inapproximability factor of $\Omega(\log n)$, due to Kortsarz *et. al.* [22], only holds when k is polynomially large in n. Only recently, Chakraborty *et. al.* [7] obtained the first non-trivial approximation ratios for $k \geq 3$; they give an $2^{O(k^2)} \log^4 n$-approximation algorithm for single-source k-vertex connectivity. Concurrently and independently of this work, Chekuri and Korula [8], building on the ideas in [7], have recently given an $O(k^{O(k)} \log n)$-approximation algorithm for single-source k-vertex connectivity. Their approach is based on analyzing the dual of the natural LP relaxation for the problem. They also obtain a similar approximation guarantee for the more general single-source rent-or-buy network design problem.

We also study the vertex-cost variation of VC-SNDP where the costs are on vertices instead of edges.

Specifically, the goal is to find a minimum-cost subset $V' \subseteq V$ of vertices, such that in the graph induced by V', for every pair $u, v \in V$ of vertices there are $r_{u,v}$ vertex disjoint paths connecting u to v. We focus again on the single-source version. When $k = 1$ the problem becomes equivalent to the node-weighted Steiner tree problem, for which $O(\log n)$-approximation is known [21]. On the other hand, even this special case can be shown to be $\Omega(\log n)$-hard by a reduction from Set Cover problem [25, 11]. It is interesting that even for $k = 1$ the edge and the vertex weighted versions exhibit such different behavior.

Related Work We note that for some special cases of VC-SNDP better algorithms are known. The k-vertex connected spanning subgraph problem, a special case of VC-SNDP where for all $u, v \in V$ $r_{u,v} = k$, has been studied extensively. Cheriyan *et al.* [5, 6] 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 $k \leq (1-\epsilon)n$. For large k, Kortsarz and Nutov [24] improved the preceding bound to an $O(\ln k \cdot \min\{\sqrt{k}, \frac{n}{n-k} \ln k\})$-approximation. Fakcharoenphol and Laekhanukit [10] improved it to an $O(\log n \log k)$-approximation, and further obtained an $O(\log^2 k)$-approximation for $k < n/2$. The iterative rounding technique has been used to give a 2-approximation for arbitrary r_{ij} values for a variant of VC-SNDP known as the *element connectivity* problem [12, 6]. Frank and Tardos [14] gave a polynomial time algorithm for finding a minimum cost k-outconnected subdigraph of a directed graph. This result 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, and the costs form a metric [20]. 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, [20, 23, 4]).

Our Results Our main result is stated below.

Theorem 1 *There is a randomized polynomial time $O(k \log n)$-approximation algorithm for the edge cost version of single-source k-vertex connectivity.*

This result improves upon the $2^{O(k^2)} \log^4 n$-approximation by Chakraborty *et. al.* [7]. Our algorithm and its analysis rely on a new result about the structure of solutions for single-source k-vertex connectivity.

For the vertex cost version we obtain a similar result, albeit with somewhat weaker ratios.

Theorem 2 *There is a randomized polynomial time $O(k^7 \log^2 n)$-approximation algorithm for the vertex cost version of single-source k-vertex connectivity.*

To the best of our knowledge, no non-trivial approximation algorithm was previously known for the vertex cost version.

The subset k-connectivity problem is a special case of VC-SNDP, where we are given a subset $T \subseteq V$ of terminals, and connectivity requirements are $r_{t,t'} = k$ for all $t, t' \in T$, with all other requirements being 0. Any α-approximation algorithm for the single source k-vertex connectivity problem can be converted into a $k\alpha$-approximation algorithm for subset k-connectivity problem, in both edge and vertex cost scenarios (see e.g. Theorem 7 in [7]). It is also easy to see that an α-approximation algorithm for the single-source k-vertex connectivity implies a $k\alpha$-approximation for single-source VC-SNDP, where $r_{t,s} \in \{0, 1, ..., k\}$ can be arbitrary, in both edge cost and vertex cost scenarios. We thus obtain the following results:

Theorem 3 *In the edge cost model, there is a randomized polynomial time $O(k^2 \log n)$-approximation algorithm for both single-source VC-SNDP and the subset k-connectivity problem. In the vertex cost model, there is a randomized polynomial time $O(k^8 \log^2 n)$-approximation algorithm for both single-source VC-SNDP and the subset k-connectivity problem.*

Techniques It is easy to obtain an $O(|T|)$-approximation for both edge and vertex cost versions of single-source k-vertex connectivity: for each terminal $t \in T$, find minimum cost k vertex disjoint paths connecting t to s and output the union of all such paths for all $t \in T$. The worst case scenario for such an algorithm is when the same edge is used by many terminals when connecting to the source in the optimal solution. Since the algorithm connects each one of the terminals to the source separately, without considering possible sharing of edges by the terminals, we may end up paying $|T|$ times the cost of the optimal solution. However, if edges are heavily shared by the terminals in the optimal solution (when connecting to the source), the solution induces high connectivity among the terminals. This observation motivates our algorithm. We compute, for each terminal $t \in T$, the cheapest collection of k vertex disjoint paths connecting t to vertices in $(T \cup \{s\}) \setminus \{t\}$. Let $E(t)$ be the set of edges participating in these paths. We identify a large subset T' of terminals, such that every $t \in T'$ is k-vertex connected to $(T \cup \{s\}) \setminus T'$ by edges in $E(t)$, and solve the problem recursively on $T \setminus T'$.

The heart of the analysis lies in proving that total cost of edges in sets $E(t), t \in T$ is bounded by $O(k \cdot \mathrm{OPT})$. The main ingredient of our proof is a *prefix decomposition theorem*, which may be of independent interest. Consider an optimal solution OPT, and for each terminal $t \in T$, let $B(t)$ be a k-tuple of vertex disjoint paths connecting t to s. A path p is said to be a prefix of $f \in B(t)$ iff p is a sub-path of f containing t. We show that we can define, for each $t \in T$ and $f \in B(t)$ a prefix $p(f)$ of f, such that the resulting set of prefixes forms a collection of "spiders". The internal vertices of different spiders are disjoint, and each spider has at least two legs, while every leg of a spider originates at a distinct terminal. Notice that in general it is not hard to define prefix $p(f)$ for each path f so that the set of prefixes forms a collection of internally disjoint spiders, as long as we allow one-legged spiders. However it is crucial for us that the spiders have at least two legs, since we use them to connect terminals to one another. This requirement makes the task of prefix decomposition more challenging, and it is not even clear a priori why such a decomposition should exist. The resulting spiders provide a convenient way of connecting the terminals to one another, which can be thought of as re-routing the flow from a terminal $t \in T$ to s via other terminals. Since the spiders do not share any edges, the cost of this re-routing is close to the cost of the optimal solution.

This approach allows us to obtain an $O(k \log n)$-approximation for the edge cost version. In order to obtain an approximation algorithm for the vertex cost version, we start with the prefix decomposition theorem again. However, since the costs are now on vertices, we cannot use the spiders to construct the k-tuples of paths connecting each terminal t to $(T \cup \{s\}) \setminus \{t\}$. The reason is that a spider may have many legs, and each path re-routed through the spider will use the vertex that serves as the spider head. Therefore, if we find the k-tuple of paths connecting each terminal t to $(T \cup \{s\}) \setminus \{t\}$ separately, we may end up paying very high cost. Instead, the spider decomposition theorem allows us to write a strong linear program. We then round a fractional solution to this LP to obtain the desired collection of vertices that k-vertex connects each terminal t to $(T \cup \{s\}) \setminus \{t\}$, and whose cost is close to OPT.

Organization: Section 2 formally defines the single-source k-vertex connectivity problem, and introduces some notation that we use throughout the paper. In Sections 3 and 5, we study the problem in the edge cost model and vertex cost model, establishing Theorems 1 and 2, respectively. The proof of both these results relies on a structural result that we refer to as the Prefix Decomposition theorem, and establish in Section 4. We conclude with some future directions in Section 6.

2 Preliminaries

In the *single-source k-vertex connectivity* problem, we are given a graph $G(V, E)$, a source s, a subset $T \subseteq V$ of terminals, and an integer k. In the *edge-cost version*, we are given a cost function c on edges, and the goal is to find a minimum cost subset E' of edges such that each terminal t is k-vertex connected to s in the graph induced by the edges in E'. In the *vertex-cost version*, we are given a cost function c on vertices, and the goal is to find a minimum cost subset V' of vertices, such that in the graph induced by V' every terminal t is k-vertex connected to source s. For any subset T' of terminals, we denote by $T'^{+} = T' \cup \{s\}$.

Let $T' \subseteq T$ be any subset of terminals. We say that a terminal t is *weakly k-connected* to set $T'^{+} \setminus \{t\}$ if there exist k *internally* vertex-disjoint paths from t to $T'^{+} \setminus \{t\}$. We say that terminal t is *strongly k-connected* to set $T'^{+} \setminus \{t\}$ if there exist k *internally* vertex-disjoint paths from t to $T'^{+} \setminus \{t\}$ such that each terminal in $(T' \setminus \{t\})$ is an end-point of at most one such path. Note that if t is strongly k-connected to $(T' \setminus \{t\}) \cup \{s\}$, then if we delete any subset $X \subseteq V \setminus \{s, t\}$ of size at most $(k-1)$, terminal t remains connected by a path to some vertex in $T' \cup \{s\}$.

The cost function c can be naturally extended to subsets of edges (vertices) in the edge cost (vertex cost) model. Moreover, given a collection of paths P, slightly abusing the notation, we will use $\mathrm{c}(P)$ to denote the sum of the costs of edges (vertices) on P in edge cost (vertex cost) model. Finally, we will use OPT to denote both an optimal solution as well as its cost.

3 Single-Source VC-SNDP with Edge Costs

In this section we sketch the proof of Theorem 1. Our algorithm and its analysis are based on establishing the following property. For each terminal t, let E_t be *any* minimum cost subset of edges, such that in the graph induced by E_t, t is strongly k-connected to the set $T^{+} \setminus \{t\}$. Then $\sum_{t \in T} \mathrm{c}(E_t) = O(k \cdot \mathrm{OPT})$. Note that the bound on $\sum_{t \in T} \mathrm{c}(E_t)$ holds without accounting for any sharing of edges among the solutions that strongly k-connect the terminals. We will show that this *separability property* naturally lends itself to a recursive algorithm for solving the single-source k-vertex connectivity problem.

3.1 Algorithm Description

The algorithm consists of the following three steps:

1. If $|T| \leq 10k$: for each terminal $t \in T$, find a minimum cost subset $E_t \subseteq E$ of edges, such that t is k-vertex connected to the source s in the graph $G_t = (V, E_t)$. Stop and output $\cup_{t \in T} E_t$. Otherwise: for each terminal $t \in T$, find a minimum cost subset $E_t \subseteq E$ of edges, such that t is strongly k-connected to the set $T^{+} \setminus \{t\}$ in the graph $G_t = (V, E_t)$. Let $B(t)$ denote the set of k vertex disjoint paths strongly connecting t to $T^{+} \setminus \{t\}$ realized by the edges in E_t, and let $\Gamma = \sum_{t \in T} \mathrm{c}(E_t)$.
2. Identify a subset $T' \subseteq T$ of size $\left\lceil \frac{|T|}{4(k+1)} \right\rceil$ such that (a) for each $t \in T'$, the paths in $B(t)$ terminate only on vertices in $T^{+} \setminus T'$, and (b) $\sum_{t \in T'} \mathrm{c}(E_t) \leq \Gamma / 2k$. Define $E' = \cup_{t \in T'} E_t$.
3. Recursively solve the problem on the set $T'' = T \setminus T'$ of terminals. Let E'' be the set of edges in the recursive solution. Output $E' \cup E''$ as the final solution.

Step (1) can be implemented by performing a standard min-cost max flow computation for each terminal t. Step (3) is a straightforward recursive computation. We now describe the implementation of step (2).

We construct a graph $H(V_H, E_H)$ with a vertex v_t for each terminal t. A vertex v_t is connected in H to all vertices $v_{t'}$ such that a path in $B(t)$ ends at terminal t'. It is readily seen that for any subset $U \subseteq V_H$, the graph induced by vertices in U contains a vertex of degree at most $2k$. This allows us to compute in polynomial-time a coloring of H with $2k+1$ colors [1] We now mark all vertices v_t such that $\mathrm{c}(E_t) \leq \frac{2\Gamma}{|T|}$. By the pigeonhole principle, there exists a color class that contains at least $\frac{|T|}{4(k+1)}$ marked vertices. Let T' be an arbitrary subset of $\left\lceil \frac{|T|}{4(k+1)} \right\rceil$ terminals corresponding to the marked vertices in this color class. Clearly,

$$\sum_{t \in T'} \mathrm{c}(E_t) \leq \left(\frac{2\Gamma}{|T|}\right) \cdot \left\lceil \frac{|T|}{4(k+1)} \right\rceil \leq \frac{\Gamma}{2k}.$$

3.2 Cost and Feasibility Analysis

A straightforward induction on the recursion depth allows us to prove the following lemma.

Lemma 3.1 *The algorithm outputs a feasible solution for the single-source k-vertex connectivity.*

Proof: We prove the lemma by induction on the recursion depth. Clearly, when $|T| \leq 10k$, the output of the algorithm is a feasible solution for the set T of terminal. Assume now that $|T| > 10k$, and that E'' is a feasible solution for the instance defined by the set T'' of terminals. We show that $E' \cup E''$ is a feasible solution for T. Consider some terminal $t \in T$. If $t \in T''$, then by the induction hypothesis, t is k-vertex connected to s in the graph induced by E''. Assume now that $t \in T'$. If t is not k-vertex connected to s in the graph $G^* = (V, E' \cup E'')$, there exists a subset $X \subseteq V \setminus \{s, t\}$ of $(k-1)$ vertices whose removal from G^* separates t from s. However, since t is strongly k-vertex connected to $T'' \cup \{s\}$ in graph induced by E', it remains connected to some $t' \in T'' \cup \{s\}$ even after X is removed from G^*. If $t' = s$ we are done. Otherwise, $t' \in T''$, and by induction hypothesis t' is k-vertex connected to s in G^*. Therefore, even when X is removed from G^*, t remains connected to t' and t' remains connected to s in G^*. A contradiction. □

We will show that the cost Γ in Step (1) of the algorithm is $O(k)\cdot$OPT. Assuming this property, we can bound the cost of the solution produced by the algorithm using the recurrence: $\alpha(p) \leq \alpha\left(p - \frac{p}{4(k+1)}\right) + O(\mathrm{OPT})$, where $\alpha(p)$ denotes the worst-case cost of a solution output by our algorithm when given as input a subset $T' \subseteq T$ of p terminals. It is easy to verify that $\alpha(|T|) = O(k \log |T|) \cdot \mathrm{OPT}$. The remainder of this section is devoted to proving that $\Gamma = O(k) \cdot \mathrm{OPT}$. The main ingredient of our proof is a Prefix Decomposition Theorem that may be of independent interest. We need the following definitions.

Definition 1 (Canonical Spider) *Let $\mathcal{M}$ be any collection of simple paths, such that each path $p \in \mathcal{M}$ has a distinguished endpoint $t(p)$, and the other endpoint is denoted by $v(p)$. We say that paths in $\mathcal{M}$ form a* canonical spider *iff $|\mathcal{M}| > 1$ and there is a vertex v, such that for all $p \in \mathcal{M}$, $v(p) = v$. Moreover, the only vertex that appears on more than one path of $\mathcal{M}$ is v. We refer to v as the* head *of the spider, and the paths of $\mathcal{M}$ are called the* legs *of the spider.*

Definition 2 (Canonical Cycle) *Let $\mathcal{M} = \{g_1, \ldots, g_h\}$ be any collection of simple paths, where each path g_i has a distinguished endpoint $t(g_i)$ that does not appear on any other path in $\mathcal{M}$, and the other endpoint is denoted by $v(g_i)$. We say that paths of $\mathcal{M}$ form a* canonical cycle*, iff (a) h is an odd integer, (b) for every path g_i, $1 \leq i \leq h$, there is a vertex $v'(g_i)$ such that $v'(g_i) = v(g_{i-1})$ (here we use the convention that*

[1] At each recursive step, delete a vertex v_t of degree at most $2k$, recursively color the remaining graph with $(2k+1)$ colors, and then insert back v_t by assigning it one of the $(2k+1)$ colors that is absent in its neighborhood.

$g_0 = g_h$), and (c) no vertex of g_i appears on any other path of $\mathcal{M}$, except for $v'(g_i)$ that belongs to g_{i-1} only and $v(g_i)$ that belongs to g_{i+1} only (see Figure 1).

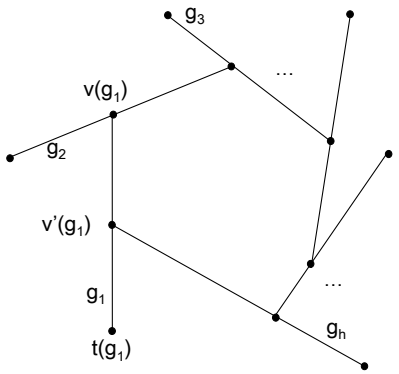

Figure 1: A canonical cycle

Assume now that we are given any collection $\mathcal{P}$ of paths, where every path $f \in \mathcal{P}$ has a distinguished endpoint $t(f)$. For $f \in \mathcal{P}$, we say that a sub-path p of f is a prefix of f iff $t(f) \in p$.

Theorem 4 (Prefix Decomposition) *Given any collection $\mathcal{P}$ of paths, where every path $f \in \mathcal{P}$ has a distinguished endpoint $t(f)$ that does not appear on any other path of $\mathcal{P}$, we can find, in polynomial time, for each path $f \in \mathcal{P}$, a prefix $p(f)$, such that in the graph induced by $\{p(f) \mid f \in \mathcal{P}\}$, the prefixes appearing in each connected component either form a canonical spider, a canonical cycle, or the connected component contains exactly one prefix $p(f)$, where $p(f) = f$ for some $f \in \mathcal{P}$.*

We defer the proof of the Prefix Decomposition Section to next section, and we proceed to show that this theorem implies that $\Gamma = O(k \cdot \text{OPT})$.

Fix an optimal solution OPT. For each $t \in T$, fix a k-tuple of internally vertex-disjoint paths $B(t)$ that connects t to s in OPT. We now define k-tuple $B'(t)$ of paths as follows. For each $f \in B(t)$, if f contains any terminal as intermediate vertex, let $t'(f)$ be the first such terminal, and otherwise let $t'(f) = s$. In order to obtain $B'(t)$, we replace each path $f \in B(t)$ by its sub-path that starts at t and ends at $t'(f)$. Let $\varphi^*(t) = \{t'(f) \mid f \in B'(t), t'(f) \neq s\}$ (it is possible that $\varphi^*(t) = \emptyset$). Notice that now no terminals appear as intermediate vertices on paths $\bigcup_{t \in T} B'(t)$, and each path in $B'(t)$ terminates at a vertex in $\varphi^*(t) \cup \{s\}$. For any terminal $t \in T$ and path $f \in B'(t)$, we say that p is a *prefix of f* iff p is a sub-path of f containing t. We say that all paths $f \in B'(t)$ belong to t. If f belongs to terminal t and p is a prefix of f, we say that p belongs to t as well.

Using the Prefix Decomposition Theorem, we can obtain the following theorem and its corollary that shows $\Gamma = O(k \cdot \text{OPT})$.

Theorem 5 (Weak k-connectivity) *Let $T' \subseteq T$ be any subset of terminals, such that for each $t \in T'$, $T' \cap \varphi^*(t) = \emptyset$. For each $t \in T'$, let H_t be a minimum-cost k-tuple of internally vertex disjoint paths that do not contain terminals as intermediate vertices, where each path $f \in H_t$ connects t to a vertex in $(T' \cup \varphi^*(t) \cup \{s\}) \setminus \{t\}$, and where each terminal in $\varphi^*(t)$ is an end-point of at most one path of H_t. Let E_t be the set of edges appearing on paths of H_t. Then $\sum_{t \in T} c(E_t) \leq 2$OPT.*

Proof: Consider some terminal $t \in T'$ and the corresponding set $B'(t)$ of paths. Recall that t does not appear on any path in $B'(t')$ for $t' \in T' \setminus \{t\}$. We create k copies of t and use a distinct copy for each path in $B'(t)$. Let $B''(t)$ be the resulting set of paths, and for each $f \in B'(t)$, let $\gamma(f)$ denote the corresponding path. If $t(f) = t$, then we say that $\gamma(f)$ belongs to t, and denote by $t(\gamma(f))$ the corresponding copy of t. If p is a prefix of $\gamma(f)$, then we also say that p belongs to t.

Set $\mathcal{P} = \bigcup_{t \in T'} B''(t)$. We apply the Prefix Decomposition theorem to $\mathcal{P}$, and obtain, for each $t \in T$ and $f \in B''(t)$ a prefix $p(f)$. Let $\mathcal{P}_1, \ldots, \mathcal{P}_s$ be the partition of the prefixes guaranteed by the Prefix Decomposition Theorem. Fix some terminal $t \in T'$, and let $B''(t) = \{f_1(t), \ldots, f_k(t)\}$. We construct a collection $H_t = \{f'_1(t), \ldots, f'_k(t)\}$ of internally vertex disjoint paths connecting t to $(T' \cup \varphi^*(t) \cup \{s\}) \setminus \{t\}$, such that each terminal in $\varphi^*(t)$ serves as endpoint of at most one such path, as follows. Consider some $i : 1 \leq i \leq k$, and assume that $p(f_i(t)) \in \mathcal{P}_j$. We consider now three cases:

- Case 1: If $p(f_i(t)) = f_i(t)$, then we set $f'_i(t) = f_i(t)$. Notice that in this case $f'_i(t)$ terminates at a vertex in $\{s\} \cup \varphi^*(t)$.
- Case 2: Otherwise, if $\mathcal{P}_j$ is a canonical cycle, $(g_1, \ldots, g_h)$, where $p(f_i(t)) = g_{i'}$, then $f'_i(t)$ is the concatenation of $g_{i'}$ and the portion of $g_{i'+1}$ lying between $v'(g_{i'+1})$ and $t(g_{i'+1})$, where copies of terminals are replaced by the corresponding terminals. Notice that $f'_i(t)$ terminates at some terminal $t' \in T' \setminus \{t\}$.
- Case 3: Otherwise, $\mathcal{P}_j$ is a canonical spider whose head $v \notin T^+$. Let $L = p(f_i(t))$ be the leg of the spider corresponding to the prefix of $f_i(t)$, and let L' be the next leg in the circular order (notice that the spider must contain at least two legs in this case). We define $f'_i(t)$ to be the concatenation of L and L', replacing copies of terminals by the corresponding terminals. Notice that $f'_i(t)$ terminates at some terminal $t' \in T' \setminus \{t\}$.

We have thus defined a k-tuple H_t of paths connecting t to $(T' \cup \varphi^*(t) \cup \{s\}) \setminus \{t\}$. We claim that these paths do not share internal vertices. Consider two paths $f'_i(t), f'_{i'}(t) \in H_t$, and assume that $f'_i(t) \in \mathcal{P}_j, f'_{i'}(t) \in \mathcal{P}_{j'}$. Since $\mathcal{P}_j, \mathcal{P}_{j'}$ are disjoint whenever $j \neq j'$, assume that $j = j'$. Then $\mathcal{P}_j$ must contain more than one prefix, and therefore it is either a canonical spider or a canonical cycle. Assume first that $\mathcal{P}_j$ is a canonical spider. Then the head of the spider, v belongs to both $f_i(t)$ and $f_{i'}(t)$. Since $f_i(t)$ and $f_{i'}(t)$ are vertex disjoint, this implies that $v = s$ and therefore Case 1 has been applied to both $f_i(t)$ and $f_{i'}(t)$. It follows that $f'_i(t)$ and $f'_{i'}(t)$ do not share internal vertices. Assume now that $\mathcal{P}_j$ is a canonical cycle. Prefixes of paths $f_i(t), f_i(t')$ may appear on the same canonical cycle, but since they do not share any vertices, they do not appear on the cycle consecutively. Therefore, $f'_i(t)$ and $f'_{i'}(t)$ are vertex disjoint. Finally, observe that each terminal $t' \in \varphi^*(t)$ may appear as an endpoint of $f'_i(t)$ only if $f_i(t)$ contains t'. Since paths $f_1(t), \ldots, f_k(t)$ do not share any vertices outside of s and t, terminal t' may serve as an endpoint of at most path in $f'_1(t), \ldots, f'_k(t)$.

Let E_t be the union of edges on paths $f'_1(t), \cdots, f'_k(t)$. We now show that $\sum_{t \in T} \mathsf{c}(E_t) \leq 2\mathrm{OPT}$. Consider any edge e that is used by the optimal solution. Since the vertices appearing on the paths in $\mathcal{P}_1, \ldots, \mathcal{P}_s$ are disjoint, e belongs to at most one such set $\mathcal{P}_j$. If $\mathcal{P}_j$ is a canonical spider, let t be the terminal to which the leg L of the spider, on which e lies, belongs. If the head of $\mathcal{P}_j$ is in T^+, then e only belongs to E_t. Otherwise, assume that the head of $\mathcal{P}_j$ is not in T^+. Let L' be the leg of spider that appears before L in the fixed circular order, and let t' be the terminal to which the corresponding prefix belongs. Then e only

belongs to E_t and $E_{t'}$. Assume now that $\mathcal{P}_j$ is a canonical cycle $(g_1, \ldots, g_h)$, and assume that e lies on g_x. Let t be the terminal to which g_x belongs and let t' be the terminal to which g_{x-1} belongs. Then e only belongs to E_t and $E_{t'}$. Finally, in the case that $\mathcal{P}_j$ contains a single path $f_i(t) = p(f_i(t))$, edge e only belongs to E_t. Therefore, the contribution of e to $\sum_{t \in T} |E_t|$ is at most 2.

□

Corollary 1 (Strong k-connectivity) *For each $t \in T$, let E_t be a minimum cost subset of edges in G that ensures that t is* **strongly** *k-connected to the set $T^+ \setminus \{t\}$. Then $\sum_{t \in T} c(E_t) \leq O(k \cdot OPT)$.*

Proof: We give an iterative procedure to prove this claim. We perform $2k$ iterations, and at the end of each iteration i, we produce for each terminal t, a k-tuple of internally vertex-disjoint paths, denoted by $P_i(t)$, such that

- each path in $P_i(t)$ connects t to a vertex in the set $(T^+ \setminus \{t\})$ and it does not contain any terminals as internal vertices,
- no terminal in $t' \in (T \setminus \{t\})$ is an end-point of more than $2k$ paths in $P_1(t) \cup P_2(t), \ldots, \cup P_{2k}(t)$, and
- $\sum_{t \in T} \mathrm{c}(P_i(t)) \leq (18k + 3) \cdot (\mathrm{OPT})$.

We show how to construct such path sets $P_i(t)$ below; first we show why the theorem follows from the existence of these path sets.

Aggregating over the $2k$ iterations, it is easy to see that each terminal t is able to send a total of $2k^2$ units of flow *integrally* to the set $(T^+ \setminus \{t\})$, with no terminal in $(T \setminus \{t\})$ receiving more than $2k$ units of flow, and no non-terminal appearing as an intermediate vertex on more than $2k$ paths. The total cost of this solution is $O(k^2 \mathrm{OPT})$, even when each terminal pays *separately* for the set of edges it uses.

Now fix any terminal t. If we scale down the integral flow solution generated above by a factor of $2k$, then we obtain a fractional solution whereby each terminal t is able to send k units of flow to the set $(T^+ \setminus \{t\})$, with at most a unit of flow passing through or terminating at any vertex in $V \setminus \{s, t\}$. By the integrality of the minimum cost flow, it follows that there is an integral solution of cost at most that of this fractional solution such that each terminal t strongly k-connects to the set $(T^+ \setminus \{t\})$. The theorem then follows since total cost of the scaled fractional solutions for all terminals is $O(k \cdot \mathrm{OPT})$. We now show how to implement each iteration.

For each terminal $t \in T$, we will create a set $\varphi_i(t)$ of size at most $3k$ at the end of each iteration i. We start by initializing $\varphi_0(t)$ to be $\varphi^*(t)$ for each terminal $t \in T$. Now at the beginning of iteration i, we create a graph $H(V_H, E_H)$ which contains a vertex v_t for each terminal t. There is an edge between two vertices v_t and $v_{t'}$ in H iff either $t' \in \varphi_{i-1}(t)$ or $t \in \varphi_{i-1}(t')$. Since we will ensure that $|\varphi_{i-1}(t)| \leq 3k$, it is easy to see that any $U \subseteq V_H$, the subgraph H' of H induced by the vertices in U has at most $3k|U|$ edges. Hence H' must contain a vertex of degree at most $6k$. This property is sufficient to recursively color vertices of H with $6k + 1$ colors. Let $T_1, T_2, \ldots, T_p$ be the resulting color classes, i.e., T_j contains all terminals t such that v_t is assigned color j in H.

Fix some color class T_j, $1 \leq j \leq p$. We apply Theorem 5 to T_j to obtain, for each terminal $t \in T_j$, a collection $P_i(t)$ of k paths $P_i(t) = \{f'_1(t), \ldots, f'_k(t)\}$, connecting t to $T_j \cup \varphi^*(t) \cup \{s\}$. These paths are internally vertex disjoint and their internal vertices are non-terminals. Moreover, every terminal $t' \in \varphi^*(t)$

may serve as an endpoint of at most one path in $P_i(t)$. Let $E^i(t)$ denote the union of edges on paths in $P_i(t)$. Then for each color class T_j, $\sum_{t \in T_j} \mathsf{c}(E^i(t)) \leq 2\text{OPT}$, and $\sum_{t \in T} \mathsf{c}(E^i(t)) \leq (6k+1) \cdot (2\text{OPT}) \leq O(k \cdot \text{OPT})$.

For each terminal t, the set $\varphi_i(t)$ is defined as follows: it contains all the terminals in $\varphi^*(t)$, and also all terminals t' that have appeared as endpoints in at least k paths in sets $P_1(t), \ldots, P_i(t)$. Since $|\varphi_0(t)| = |\varphi^*(t)| \leq k$, and $|P_h(t)| = k$ for each $1 \leq h \leq i$, $|\varphi_i(t)| \leq k + i$. Any terminal in $\varphi_0(t)$ can appear as endpoint of at most one path in $P_h(t)$ for each iteration h. Any terminal in $T \setminus (\varphi_0(t) \cup \{t\})$ can appear as endpoint on at most $(2k-1)$ paths over all iterations. No terminal appears as an intermediate vertex on paths $P_i(t)$. Thus this ensures that no terminal in $T \setminus \{t\}$ receives more than $2k$ units of flow from t over all the iterations. □

4 Proof of the Prefix Decomposition Theorem

We now prove Theorem 4. Note that the Prefix Decomposition theorem is a structural result, independent of the underlying cost model (that is edge or vertex costs). We will thus also utilize this result in designing an algorithm for the vertex costs case.

We start with the set $\mathcal{P}$ of paths, where each path $f \in \mathcal{P}$ has one distinguished endpoint $t(f)$ that does not appear on any other paths in $\mathcal{P}$, and the other endpoint is denoted by $z^*(f)$. We will view path f as starting at $t(f)$ and ending at $z^*(f)$. Throughout the algorithm we will define prefixes of paths in $\mathcal{P}$. Prefix of a path f is denoted by $p(f)$, and it is defined to be the portion of f between $t(f)$ and some other vertex $z(f) \in f$. At the beginning of the algorithm, $z(f) = z^*(f)$ for all f. Throughout the algorithm, path f is trimmed by moving $z(f)$ closer to $t(f)$. We can trim a path several times, and during the execution of the algorithm the prefix may only become shorter.

Definition 3 (Canonical Set) *Let $\mathcal{P}' \subseteq \mathcal{P}$ be any set of paths. A set of prefixes $p(f)$ for $f \in \mathcal{P}'$ is said to form a* canonical set *iff in the graph induced by the prefixes $\{p(f) \mid f \in \mathcal{P}'\}$, each connected component is either a canonical spider, a canonical cycle, or it contains a single prefix $p(f) = f$ for some $f \in \mathcal{P}$.*

In order to complete the proof of the Prefix Decomposition theorem, it is enough to show that we can define, for each path $f \in \mathcal{P}$ a prefix $p(f)$, such that the set of prefixes of paths in $\mathcal{P}$ forms a canonical set.

We show an algorithm that finds such prefixes $p(f)$. We start with the set $\mathcal{P}$ of paths and prefixes $p(f) = f$ for all $f \in \mathcal{P}$. Throughout the algorithm we will maintain a collection D of *dead paths*, and all other paths in $\mathcal{P}$ are referred to as *live paths*. If f is a live path, then we refer to $p(f)$ as a *live prefix*. At the beginning, D contains a path $f \in \mathcal{P}$ iff none of the vertices of f belongs to other paths in $\mathcal{P}$. Clearly, prefixes in D form a canonical set.

Definition 4 (Special Vertices) *For any live path f, we say that u is a special vertex of $p(f)$ if u belongs to another live prefix. The i_{th} special vertex of f (counting from $t(f)$), is denoted by $u_i(f)$.*

Note that multiple special vertices on f might result due to intersections of f with another path f', so $u_i(f)$ and $u_j(f)$ may be a result of the intersection with the same path. We maintain the following invariants throughout the algorithm:

C1. The set of path prefixes in D forms a canonical set.

We define a set U of vertices as follows. For each connected component $\mathcal{P}'$ of D, if $\mathcal{P}'$ is a canonical spider, then U contains its head, and if $\mathcal{P}'$ contains only one prefix $p(f) = f$ for some $f \in \mathcal{P}$, then U contains $z^*(f)$.

C2. The only vertices that dead paths and live paths may share are vertices in U. For any live path f, its prefix $p(f)$ may contain at most one vertex in U. If $p(f)$ contains a vertex of U, then this vertex must be $z(f)$, the last vertex of $p(f)$.

C3. For a live path f, one of the following conditions must hold:

- $p(f)$ contains a vertex in U (and this is the last vertex of $p(f)$).
- $p(f)$ does not contain any vertex in U but $z(f) = z^*(f)$.
- $p(f)$ neither contains a vertex in U nor is $z(f) = z^*(f)$, but there is another live path $f' \neq f$ such that $z(f)$ is the first special vertex of f'. We refer to f' as a *witness* for f, denoted by $f' = W(f)$. Note that $W(f)$ is defined only if the first two conditions do not hold.

The algorithm works in iterations. In each iteration, we either trim prefixes of some live paths, or move some live paths to set D. At the beginning of the algorithm, it is easy to see that properties C1–C3 hold. Assume these properties hold before the current iteration. We show how to perform an iteration and maintain the properties. The algorithm ends when $D = \mathcal{P}$. It will be clear from the description given below that each iteration can be executed in polynomial time. We will use the following easy observation in our analysis:

Proposition 1 *Assume we have a collection of paths D, and prefixes of paths in $\mathcal{P}$ for which properties C1–C3 hold. Let f be any live path, $v \in p(f)$ be any special vertex of f, and f' be a live path whose first special vertex is v. Then if we trim f at vertex v by setting $z(f) = v$, and either set $W(f) = f'$ or add v to U, property C3 continues to hold.*

Proof: Note that no assertion is being made about maintaining properties C1 or C2. The only potential problem is paths f^* for which $W(f^*) = f$. In this case $p(f^*)$ must contain $u_1(f)$, the first special vertex of f. But since we trim f at its special vertex and do not trim f^*, $u_1(f)$ continues to be the first special vertex of $p(f)$, that belongs to $p(f^*)$. □

Iteration Description: We now describe how an iteration of the algorithm is executed.

At the beginning of an iteration, each live path f marks its first special vertex $u_1(f)$. If no such vertex exists, then f marks the vertex $z(f)$. Note that if $p(f)$ does not contain any special vertices then $z(f) = z^*(f)$ or $z(f) \in U$ must hold by Invariant C3. We then perform one of the next five steps. We consider the steps in the order in which they appear, and perform the earliest listed step that is applicable. After an applicable step is executed, the iteration ends and we proceed to the next iteration. As long as the set of live paths is non-empty, we will show that one of the steps below necessarily applies.

Step 1 Is performed if some path f marks its vertex $z(f)$, and either $z(f) \in U$ or $z(f) = z^*(f)$. We add f to D, where it either becomes part of an existing canonical spider or defines a new connected component.

- If $z(f) \in U$, then $p(f)$ becomes part of an existing canonical spider. Notice that $p(f)$ does not share any vertices (except for $z(f)$) with another live prefix, and therefore, properties C1 and C2 still hold. It is impossible that $f = W(f')$ for some live path f', since the only vertex that f shares with any other live prefix is $z(f)$, which belongs to U. Therefore, property C3 is still true.

- If $z(f) = z^*(f) \notin U$, then f defines a new connected component in D, and we add $z^*(f)$ to U. Since $z(f) = z^*(f)$, we have that $p(f) = f$. It is easy to see that properties C1–C3 still hold.

The case when some path f marks a vertex $z(f)$, and neither $z(f) \in U$ nor $z(f) = z^*(f)$ (and therefore, $z(f)$ is a special vertex for f), will be handled as one of the cases in Step 2. Notice that in this case, by property C3, $W(f)$ must exist, and both f and $W(f)$ mark the same vertex $z(f)$.

Step 2 Is performed if there are two or more paths that marked the same vertex v. Let $F = \{f_1, \ldots, f_h\}$ be the set of all paths that marked v, and let F' be the collection of all other live paths whose prefixes contain v (possibly $F' = \emptyset$). We perform the following actions: (i) the prefixes of all paths in F and F' are trimmed at v, (ii) the paths in F are added to set D of dead paths, and (iii) the vertex v is added to the set U. Notice that prefixes of paths of F form a canonical spider, and the only vertex they share with any live prefixes is v, while they do not share any vertex with dead prefixes. It is thus easy to see that properties C1 and C2 still hold. We can use Proposition 1 to prove that property C3 is also still true (we can think about this process as trimming paths in F and F' one by one and using Proposition 1 after each step to show that property C3 is maintained).

Step 3 Is performed if there are two live paths f, f', and f' marked some vertex v that lies strictly between $t(f)$ and $z(f)$ on $p(f)$. We then trim path f at v by setting $z(f) = v$ and $W(f) = f'$. Clearly, properties C1–C2 still hold, and we can invoke Proposition 1 to see that property C3 is still true (as v must be a special vertex of f).

Assume now that we have been unable to perform any of the Steps 1 through 3. This means that for each path f', if v is the vertex that f' has marked, then $v \neq z(f')$ and v is the last vertex on some other path f. We will say that f' gives its token to f. If there are several paths containing v, then f' gives its token to all these paths.

Proposition 2 *If none of the Steps 1 through 3 is applicable, then the token distribution results in each live path giving and receiving exactly one token.*

Proof: From the description above, it is clear that each path gives at least one token. We will argue that no path can receive more than one token. This clearly implies that every path gives and receives exactly one token. Suppose there exists a path f that receives two or more tokens. Since Step 3 is not applicable, two or more paths must have marked $z(f)$. But then Step 2 is applicable, a contradiction. □

Proposition 3 *If none of the Steps 1 through 3 is applicable, then*

- *the prefix of each live path f has at least two special vertices $u_1(f)$ and $u_2(f)$;*
- *no live prefix contains a vertex from the set U; and*
- *if f' gives token to f and $z(f) \neq z^*(f)$ then $W(f) = f'$.*

Proof: By Proposition 2, we know that each live path f gives a token by marking the vertex $u_1(f)$. Since each live path f also receives a token, some path f' must have marked a vertex u that lies on $p(f)$. If $u = u_1(f)$ then Step 2 would have been applicable. So u must be distinct from $u_1(f)$. Hence $u_2(f)$ must exist.

To see that no live prefix contains a vertex of U, note that if a path f contains a vertex of U, then it must be $z(f)$ by Property C2. Since $z(f)$ has been marked by path $f' = W(f)$, and $z(f) \in U$, it follows that $z(f) = z(f')$. We should have then executed Step 1.

By Proposition 2, for each live path f, there is a unique live path f' that gives a token to f. Hence f' is the unique live path that contains $z(f)$ as its first special vertex. It follows that $W(f) = f'$. □

Step 4 Is performed if for some f, the path f' to which f gave its token also contains $u_2(f)$. This means that f' meets f at least twice: first at $u_2(f)$ then at $u_1(f)$. We can then trim f' at $u_2(f)$ by setting $z(f') = u_2(f)$. We set $W(f') = f$ as before. It is easy to see that all properties continue to hold, since now $u_2(f)$ becomes the first special vertex of f.

Assume that none of the Steps 1 through 4 are applicable. We create a directed graph where for every live path f, there is a vertex w_f. We connect w_f to $w_{f'}$ iff the path f^* that gave its token to f has $u_2(f^*)$ belonging to f'. Notice that the out-degree of every vertex in this graph is at least 1, and since Step 4 was not applicable, no self-loops can exist. Therefore, we can find a simple cycle in this graph. Assume that the paths corresponding to the vertices on the cycle are $f_1, \ldots, f_h$ (we use the convention that $f_{h+1} = f_1$). For each $i : 1 \leq i \leq h$, let f'_i be the path that gave its token to f_i. Then $u_2(f'_i)$ belongs to f_{i+1} (see Figure 2). We will need the following simple observation.

Proposition 4 *For each $i : 1 \leq i \leq h$, $u_2(f'_i)$ lies strictly between $t(f_{i+1})$ and $z(f_{i+1})$.*

Proof: Assume for contradiction that $u_2(f'_i) = z(f_{i+1})$ (it is impossible that $u_2(f'_i) = t(f_{i+1})$). Then path f'_{i+1}, which has marked $z(f_{i+1})$ has two paths intersecting it at $z(f_{i+1})$, which is impossible since every path gives and receives exactly one token. □

We say that path f_i is bad iff for some $j : 1 \leq j \leq h$, $f_i = f'_j$, and $u_2(f'_{i-1}) = u_1(f_i)$. If no bad paths exist in our cycle, we perform Step 5.

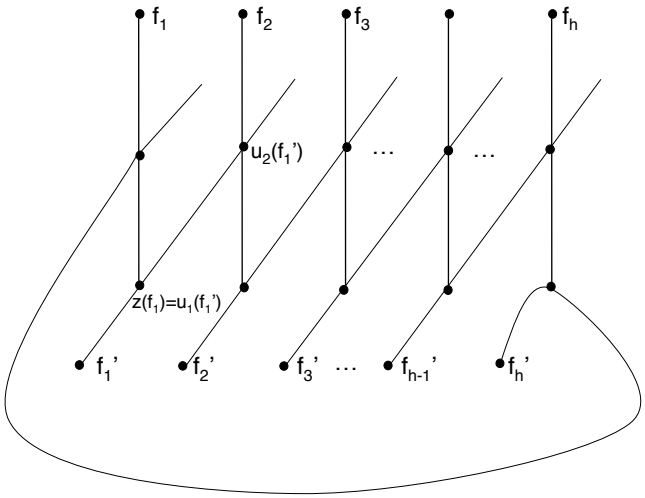

Figure 2: Before the execution of step (5).

Step 5 We trim each path f_i, $1 \leq i \leq h$, by setting $z(f_i) = u_2(f'_{i-1})$ and setting $W(f_i) = f'_{i-1}$ (See

Figure 3). It is easy to see that properties C1 and C2 continue to hold. We now focus on showing that property C3 still holds as well. We only perform trimming of paths $\{f_i\}_{i=1}^{h}$. So we only need to take care of two cases:

- For each path $f_i \in \{f_1, \ldots, f_h\}$, we need to show that after the trimming f_i contains the first special vertex of $W(f_i) = f'_{i-1}$, and this is the last vertex $z(f_i)$ of $p(f_i)$.
- If $f \notin \{f_1, \ldots, f_h\}$, and $W(f) = f_i$, then we need to show that f contains the first special vertex of f_i, and it is the last vertex of $p(f)$.

For the second case, observe that f_i has been trimmed on its special vertex, and f has not been trimmed in this iteration. Therefore, we can use arguments similar to Proposition 1 to show that f still contains the first special vertex of f_i.

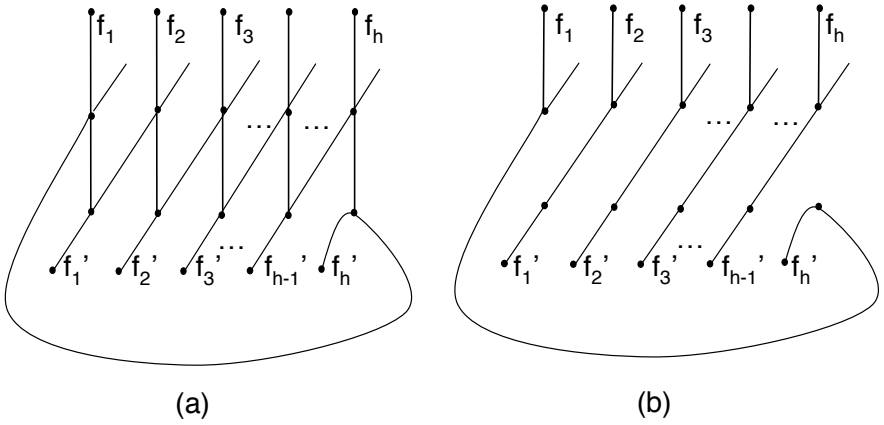

Figure 3: Illustration of step (5)

Consider now the first case. Recall that from Proposition 4, the trimming of f_{i-1} made it shorter by at least one edge. Therefore, after the trimming, f_{i-1} does not contain $u_1(f'_{i-1})$, which stops being the first special vertex of f'_{i-1}. (Recall that f_{i-1} and f'_{i-1} are the only paths that contained $u_1(f'_{i-1})$, since otherwise f'_{i-1} gives at least two tokens). If we show that $u_2(f'_{i-1})$ continues to belong to f'_{i-1} after the trimming step, then it now becomes the first special vertex of f'_{i-1}. Vertex $u_2(f'_{i-1})$ belongs to f_i even after trimming, and is the last vertex of $p(f_i)$. We only need to show that $u_2(f'_{i-1})$ still belongs to f'_{i-1} after the trimming. The only way f'_{i-1} has been trimmed is if $f'_{i-1} = f_j$ for some j. If $u_2(f'_{i-1})$ does not belong to the new prefix of $f'_{i-1} = f_j$ is then $u_2(f'_{j-1})$, at which f_j has been trimmed lies before $u_2(f_j)$ on its prefix. But since $u_2(f'_{j-1})$ is a special vertex of f_j, it must be that $u_2(f'_{j-1}) = u_1(f_j)$, and since $f_j = f'_{i-1}$, we have that f_j is a bad path, a contradiction.

Step 6 If none of the above steps has been performed, then there is a bad path $f = f_i = f'_j$ whose vertex w_f belongs to the cycle. We prove that in this case, every path f_j is bad, and a subset of prefixes of $\{f_1, \ldots, f_h\} = \{f'_1, \ldots, f'_h\}$, form a canonical cycle that can be added to the set D. We start with the following claim:

Claim 1 *If $f_i = f'_j$ is bad, then f_j is also bad and $f_j = f'_{i-1}$.*

Proof: Let $g = f'_{i-1}$. Since f_i is bad, its first special vertex is the second special vertex of g. Therefore, f_i gave its token to g, and the prefix of g currently ends at $z(g) = u_1(f)$. Thus, g has exactly two special vertices: $u_1(g)$ and $u_2(g) = z(g)$. Since $f_i = f'_j$, and f_j is the path to which f'_j gave its token, it follows that $f_j = g = f'_{i-1}$. Consider now f'_{j-1}. Its second special vertex $u = u_2(f'_{j-1})$ belongs to g and must be a special vertex of g. From Proposition 4, it cannot be the last vertex of g, $z(g) = u_2(g)$. So it must be $u_1(g)$. It follows that $g = f_j = f'_{i-1}$ is also bad. □

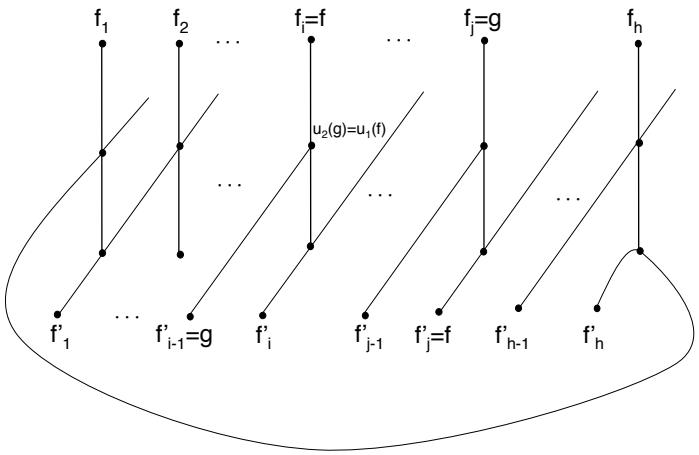

Figure 4: Proof of claim 1

Corollary 2 *If $f_i = f'_j$ is bad, then f_{i-1} is bad and $f_{i-1} = f'_{j-1}$.*

Proof: Assume that $f_i = f'_j$ is bad. Then from Claim 1, f_j is bad and $f_j = f'_{i-1}$. We now apply the same claim again to f_j serving as f_i and f'_{i-1} serving as f'_j. We get that f_{i-1} is also bad and $f_{i-1} = f'_{j-1}$. □

From the above corollary, all paths $F = \{f_1, \ldots, f_h\}$ are bad, and $F = \{f'_1, \ldots, f'_h\}$. If $f_i = f'_j$ is bad, then since f_{j+1} is also bad, the second special vertex of $f_i = f'_j$ is the first special vertex of f_i, and so it is also the last vertex of $p(f_i)$. It follows that every prefix $p(f_i)$, for $f_i \in F$, contains exactly two special vertices: $v_1(f_i)$ and $v_2(f_i) = z(f_i)$. Moreover, $v_1(f_i) = v_2(f'_{i-1})$ and no other live path contains it; $v_2(f_i) = v_1(f'_i)$ and no other live path contains it. Let $h' : 1 \leq i \leq h$ be such that $f_1 = f'_{h'}$. Considered the ordered set of prefixes $\mathcal{M} = (p(f_1), p(f'_1), p(f_2), p(f'_2), \ldots, p(f'_{h'-1}), p(f_{h'}))$. Notice that the number of prefixes in $\mathcal{M}$ is odd. From the above discussion, prefixes in $\mathcal{M}$ form a canonical cycle. Moreover, vertices appearing on the prefixes in $\mathcal{M}$ do not appear on any other live prefixes and do not belong to paths in D. We move the paths whose prefixes belong to $\mathcal{M}$ to D.

5 Single-Source VC-SNDP with Vertex Costs

We now consider the case when the costs are on vertices. We are given a graph $G = (V, E)$ with cost $c(v) \geq 0$ for each vertex $v \in V$, a subset $T \subseteq V$ of terminals, and a source $s \in V \setminus T$. The goal is to find a minimum cost subset $V' \subseteq V$ of vertices, such that in the graph induced by V' every terminal is k-vertex

connected to s. For each subset $T' \subseteq T$ of terminals, we again denote by T'^+ the set $T' \cup \{s\}$ of vertices. We assume w.l.o.g. that the cost of every vertex in T^+ is 0 since any solution must include them. The main theorem of this section is the following.

Theorem 6 *Let $G = (V, E)$ be any instance of single-source k-vertex connectivity problem with terminal set T, source s and vertex costs c. Given any subset $T' \subseteq T$ of terminals, there is a randomized polynomial time algorithm that finds, with high probability, a subset $V' \subseteq V$ of vertices of cost $O(OPT \cdot k \log n)$ with the following properties. The graph induced by V' contains, for each $t \in T'$, a k-tuple $F(t)$ of internally vertex disjoint paths. Each path in $F(t)$ connects t to some vertex in $T^+ \setminus \{t\}$, while for terminals $t' \in T \setminus T'$ at most one path in $F(t)$ terminates at t'. Moreover, paths in $F(t)$ do not contain any terminals of T as intermediate vertices.*

Before proving this theorem, we first show that it suffices to get an $O(k^7 \log^2 n)$-approximation algorithm for the vertex costs version of single-source k-vertex connectivity. We need the following analogue of Corollary 1.

Corollary 3 *There is a randomized polynomial time algorithm, that finds, with high probability, a subset $V' \subseteq V$ of vertices of cost $O(OPT \cdot k^6 \cdot \log n)$, such that in the graph induced by V' every terminal $t \in T$ is strongly k-vertex connected to $T^+ \setminus \{t\}$.*

Proof: We will define an iterative process, consisting of $4k^2$ iterations. The input to iteration i contains, for each terminal $t \in T$, a set $\varphi_{i-1}(t)$ of forbidden terminals, with $|\varphi_i(t)| \leq ik$ for all i. As input to the first iteration, $\varphi_0(t) = \emptyset$ for all $t \in T$.

Iteration i proceeds as follows. First, we construct a graph H of terminals as before. There is a vertex v_t for each terminal $t \in T$, and there is an edge between t and t' iff $t' \in \varphi_{i-1}(t)$ or $t \in \varphi_{i-1}(t')$. Since $|\varphi_{i-1}(t)| \leq 4k^3$ for all i, we can color this graph with $p = 8k^3 + 1$ colors. Let $T_1, \ldots, T_p$ be the partition of terminals according to their color classes. For each color class T_j, we then apply Theorem 6 with $T' = T_j$ to obtain a subset V_j^i of vertices and a k-tuple $P_i(t)$ of internally vertex disjoint paths for each $t \in T_j$, that are contained in the graph induced by V_j^i. Recall that each path in $P_i(t)$ terminates at a vertex in $T^+ \setminus \{t\}$, with at most one path terminating at any terminal $t' \in T \setminus T_j$. Finally, we define $\varphi_i(t)$ as follows: $\varphi_i(t)$ contains all terminals t', such that at least one path in sets $P_1(t), \ldots, P_i(t)$ terminates at t'. Since $|P_j(t)| = k$ for all j, we have that $|\varphi_i(t)| \leq ik$ for all i.

Let $V^i = \bigcup_{j=1}^{p} V_j^i$ and let $V' = \bigcup_{i=1}^{4k^2} V^i$. Then for all $1 \leq i \leq 4k^2$, $\mathrm{c}(V^i) \leq O(\mathrm{OPT} \cdot k^4 \log n)$ and $\mathrm{c}(V') \leq O(\mathrm{OPT} \cdot k^6 \log n)$.

It remains to show that each terminal t is strongly k-connected to $T^+ \setminus \{t\}$ in the graph induced by V'. Fix a terminal $t \in T$ and consider the k-tuples of paths $P_1(t), \ldots, P_{4k^2}(t)$. For each $i : 1 \leq i \leq 4k^2$, let D_i be the set of terminals t' that do not belong to $\varphi_{i-1}(t)$ such that at least one path in $P_i(t)$ terminates at t', and let D_i' be the set of terminals t' that belong to $\varphi_{i-1}(t)$ such that at least one path in $P_i(t)$ terminates at t' (for $t' \in D_i'$ there is exactly one such path).

For each $j : 1 \leq j \leq 2k$, we say that a terminal t' is *j-bad* iff $t' \in D_{k(j-1)+1} \cup \cdots \cup D_{kj}$ and the total flow that t sends to t' is more than $4k^2 - k(j-1)$. Assume first that there is some $j : 1 \leq j \leq 2k$ such that no terminal is j-bad. Consider the flow-paths defined by the k-tuples $P_{k(j-1)+1}(t), \ldots, P_{4k^2}(t)$, where one unit of flow is sent along each path. The total flow that t sends via these k-tuples is $k(4k^2 - k(j-1))$. Since each k-tuple of paths is internally vertex disjoint, every non-terminal vertex has at most $4k^2 - k(j-1)$

flow units passing through it. We claim that every terminal t' receives at most $4k^2 - k(j-1)$ flow units. Assume otherwise and let t' be a terminal that receives more than $4k^2 - k(j-1)$ flow units via paths in $P_{k(j-1)+1}(t), \ldots, P_{4k^2}(t)$. Let D_h be the first set in which t' appears, $k(j-1)+1 \leq h \leq 4k^2$. Then for each $i > h$, at most one flow unit is sent to t' via P_i. Since $P_h(t)$ sends at most k flow units to t', it must be the case that $k(j-1)+1 \leq h \leq kj$ and thus t' is j-bad, a contradiction. In total we have that $k(4k^2 - k(j-1))$ flow units are sent from t, with at most $4k^2 - k(j-1)$ flow units going through any vertex and at most $4k^2 - k(j-1)$ flow units terminating at any terminal. We can scale this flow down by the factor of $4k^2 - k(j-1)$ and obtain a fractional flow that sends k units from t to $T^+ \setminus \{t\}$, such that at most one flow unit passes through every non-terminal vertex, and at most one flow unit terminates at any terminal. From the integrality of flow, t is strongly k-vertex connected to $T^+ \setminus \{t\}$ in the graph induced by V'.

Assume now that for each $j : 1 \leq j \leq 2k$ there is a j-bad terminal, denoted by t_j. This means that $t_j \in D_{k(j-1)+1} \cup \cdots \cup D_{kj}$ and the total flow that t sends to t_j is at least $4k^2 - k(j-1)$. If we consider the k-tuples $P_{2k^2+1}(t), \ldots, P_{4k^2}(t)$ of paths, then they must send at least $2k^2 - k$ flow units to each t_j, $1 \leq j \leq 2k$. So in total these k-tuples have to send $2k(2k^2 - k) = 4k^3 - 2k^2$ flow units to $t_1, \ldots, t_{2k}$, while each one of the $2k^2$ k-tuples actually only sends k flow units, which is impossible. □

We now state our algorithm for the vertex costs case.

1. If $|T| \leq 10k$: for each terminal $t \in T$, find a minimum cost subset $V_t \subseteq V$ of vertices, such that t is k-vertex connected to the source s in the graph induced by V_t. Stop and output $\cup_{t \in T} V_t$. Otherwise, using Corollary 3, find a set V' of vertices of cost $O(\text{OPT} \cdot k^6 \cdot \log n)$, such that in the graph induced by V' every terminal $t \in T$ is strongly k-vertex connected to $T^+ \setminus \{t\}$. Let $F(t)$ denote the k-tuple of paths that strongly k-vertex connect t to $T^+ \setminus \{t\}$.
2. Identify a subset $T' \subseteq T$ of size $\left\lceil \frac{|T|}{2(k+1)} \right\rceil$ such that for each $t \in T'$, the paths in $F(t)$ terminate only on vertices in $T^+ \setminus T'$.
3. Recursively solve the problem on the set $T'' = T \setminus T'$ of terminals. Let V'' be the set of vertices in the recursive solution. Output $V' \cup V''$ as the final solution.

Using similar arguments as in Lemma 3.1 it is easy to see that the algorithm outputs a feasible solution. The total depth of recursion is bounded by $O(k \log n)$ and thus the solution cost is at most $O(\text{OPT} \cdot k^7 \log^2 n)$.

The proof of Theorem 6 is based on rounding an LP relaxation for computing a minimum cost spider decomposition. We next present the details of this process.

5.1 An LP Relaxation for Minimum Cost Spider Decomposition

Fix an optimal solution OPT for the given instance of the single-source k-vertex connectivity problem. For each terminal $t \in T$, let $B(t)$ be a k-tuple of vertex disjoint paths connecting t to s in OPT. As before, for each $f \in B(t)$ we construct a path f' as follows: if f does not contain any terminals as intermediate vertices then $f' = f$. Otherwise, f' is a prefix of f that starts at t, ends at some terminal and does not contain any terminals as internal vertices. Let $B'(t)$ be the set of corresponding paths f' for $f \in B(t)$, and let $\varphi^*(t)$ denote the set of endpoints of paths in $B'(t)$, excluding s and t (note that $0 \leq |\varphi^*(t)| \leq k$). Let $\mathcal{P} = \bigcup_{t \in T'} B'(t)$. We construct a new set $\mathcal{P}'$ of paths from $\mathcal{P}$, as follows. For each terminal $t \in T'$,

we replace every appearance of t on paths $B'(t)$ by a copy of t, obtaining a k-tuple $B''(t) \subseteq \mathcal{P}'$ of paths. For each path $f \in B''(t)$, we say that f belongs to t, and we say that each prefix p of f belongs to t as well. We set $\mathcal{P}' = \bigcup_{t \in T'} B''(t)$. We can now apply the Prefix Decomposition theorem to $\mathcal{P}'$ and obtain, for each $f \in \mathcal{P}'$, a prefix $p(f)$. Let $\mathcal{R}$ be the set of the resulting prefixes, and let $\mathcal{H}_1, \ldots, \mathcal{H}_h$ be the partition of $\mathcal{R}$ according to the connected components in the graph induced by $\mathcal{R}$. Consider some component $\mathcal{H}_i$, $1 \leq i \leq h$ that forms a canonical cycle. We need to further decompose $\mathcal{H}_i$ into a collection of spiders, as follows. Let $g_1, \ldots, g_h$ denote the collection of prefixes in $\mathcal{H}_i$, and recall that h is odd. Let $f_1, \ldots, f_h$ be the collection of paths in $\mathcal{P}'$ such that $p(f_j) = g_j$ for all $1 \leq j \leq h$. Consider the paths $g_1, g_3, g_5 \ldots, g_h$. There must exist a pair of consecutive paths in this sequence, g_{2j-1}, g_{2j+1} which belong to distinct terminals: otherwise, g_1 and g_h belong to the same terminal t, and this is impossible since g_1 and g_h share a vertex. Assume that g_{2j-1} and g_{2j+1} belong to distinct terminals. For convenience, we change the indexing of the paths so that g_{2j} becomes g_1, but the circular ordering of the paths does not change.

For each $i : 1 \leq i < h/2$, we trim the prefix of g_{2i} at vertex $v'(g_{2i})$. Now the graph defined by the union of these prefixes decomposes into $\lfloor h/2 \rfloor$ 2-legged spiders, where the spiders are defined by the prefixes of g_{2i} and g_{2i-1} for $1 < i < h/2$. Since these two prefixes intersect, they must belong to distinct terminals. Additionally, we get a 3-legged spider defined by the prefixes of g_1, g_2 and g_h (see Figure 5). Notice that this is not a canonical spider, since there is no vertex appearing on all three prefixes. We have ensured that g_2 and g_h belong to distinct terminals. Since g_1 intersects both these paths, all three paths belong to distinct terminals. We perform this decomposition on every component $\mathcal{H}_i$ whose prefixes form a canonical cycle. Let $\mathcal{R}'$ denote the resulting set of prefixes.

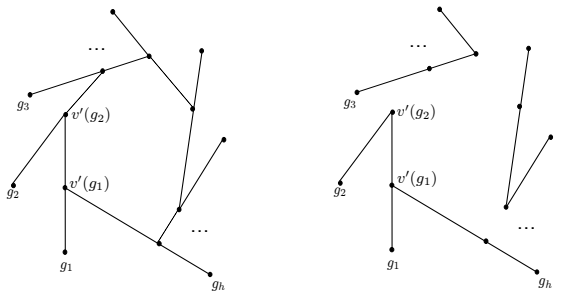

Figure 5: Decomposition of canonical cycles into spiders.

Finally, we obtain a collection $\mathcal{R}''$ of paths from $\mathcal{R}'$ by replacing each copy of every terminal $t \in T$ on prefixes of paths $B''(t)$ by terminal t itself. For each terminal $t \in T'$, let $B^*(t)$ denote the k-tuple of prefixes corresponding to the paths in $B''(t)$. We therefore obtain a collection of spiders, whose internal vertices are completely disjoint, but endpoints and heads may be shared by several spiders. We have spiders of two types: those whose head is either a terminal or the source s may have one or more legs; other spiders must have at least two legs each. Moreover, no terminal appears as an intermediate vertex on legs of any spider, and for every terminal $t \in T$, all k prefixes in $B^*(t)$ belong to distinct spiders whose heads are distinct vertices in $V \setminus \{t\}$. We will write an LP relaxation for finding such a spider decomposition of minimum cost, and round the solution. The goal of the rounding procedure is to find a subset $V' \subseteq V$ of vertices of cost $O(\mathrm{OPT} \cdot k \log n)$ and a k-tuple $F(t)$ of paths for each terminal $t \in T'$, that are contained in the graph induced by V', such that the paths in $F(t)$ are internally vertex disjoint and do not contain any

terminals as intermediate vertices. Moreover, for each $t' \in T \setminus T'$, at most one path in $F(t)$ may terminate at t'.

Let $V^* = V \setminus T^+$. We start by constructing a directed graph $G' = (V', E')$, as follows. For each vertex $v \in V^*$, we create two copies, h_v, ℓ_v. The copy h_v corresponds to vertex v being a head of a spider, and the copy ℓ_v corresponds to vertex v being an intermediate vertex. The cost of each copy is $c(v)$. We set $V' = \{h_v, \ell_v \mid v \in V^*\} \cup T \cup \{s\}$. For each edge $e = (u, v) \in E$, we add all possible edges between the copies of u, v (there are two such copies for non-terminals and one copy for terminal vertices). Now to obtain set E' of edges, we remove all edges leaving h_v for all $v \in V$, and we remove all edges leaving s. We also remove all edges leaving terminals $t \in T \setminus T'$.

For every vertex $v \in V^*$, we have two variables: x_v, y_v. Variable x_v indicates whether or not v is used as a spider-head, and variable y_v indicates whether or not v is used as an intermediate vertex.

We will think of each terminal t_i as sending k units of flow from t_i to spider-heads; we will refer to this as the *flow of type* i. The total congestion on intermediate vertices is at most 1. The total congestion allowed on spider-heads is unrestricted, but congestion with respect to each flow type is restricted to be at most 1 and no flow of type i can terminate at t_i. Recall that any terminal $t \in T$ may also serve as a spider-head. In this case again the total congestion is unrestricted but congestion with respect to each flow type is at most 1. For each vertex u we denote by $\delta^-(u)$ and $\delta^+(u)$ the set of incoming and outgoing edges in G', respectively. Also, for each edge e and flow type i, we use variable $f_i(e)$ to denote the flow of type i traversing edge e.

$$\text{min } \sum_{v \in V^*} c(v)(x_v + y_v)$$

s.t.

$x_v + y_v \le 1$	$\forall v \in V^*$	
$\sum_{e \in \delta^+(t_i)} f_i(e) = k$	$\forall t_i \in T'$	(k units of flow leave t_i)
$f_i(e) = 0$	$\forall t_i \in T', e \in \delta^-(t_i)$	(no type i flow enters t_i)
$f_j(e) = 0$	$\forall t_i, t_j \in T', j \ne i, \forall e \in \delta^+(t_i)$	(no flow of type j leaves t_i)
$\sum_{e \in \delta^+(\ell_v)} f_i(e) = \sum_{e \in \delta^-(\ell_v)} f_i(e)$	$\forall v \in V^*, t_i \in T'$	(flow conservation)
$\sum_{e \in \delta^-(\ell_v)} \sum_i f_i(e) = y_v$	$\forall v \in V^*$	(total congestion on intermediate vertices)
$\sum_{e \in \delta^-(h_v)} f_i(e) \le x_v$	$\forall v \in V^*, t_i \in T'$	(congestion due to flow of type i
$\sum_{e \in \delta^-(t_j)} f_i(e) \le 1$	$\forall t_j \in T, t_i \in T', i \ne j$	on spider-head vertices)
$\sum_{e \in \delta^-(h_v)} \sum_i f_i(e) \ge 2x_v$	$\forall v \in V^*$	(at least two spider legs)
$f_i(e), x_v, y_v \ge 0$	$\forall v \in V^*, t_i \in T', e \in E'$	

Observe that the spider-decomposition of the optimal solution defined by the set R'' of prefixes provides a feasible solution to the LP of cost at most OPT. We now fix the optimal feasible solution of the LP and round it.

5.2 The Rounding Algorithm

As a pre-processing step, we perform a suitable flow-path decomposition of a fractional solution to the above LP. Let $\epsilon > 0$ be a sufficiently small value such that each x_v and y_v value can be expressed as an integer multiple of ϵ. For each terminal $t_i \in T'$, we build k/ϵ flow-paths, each carrying ϵ units of flow of type i.

Each flow-path starts at t_i and ends either at s, or $t_j \in T \setminus \{t_i\}$, or h_v for some $v \in V \setminus (T \cup \{s\})$. All other vertices on the flow-path are intermediate vertices ℓ_v. Our LP constraints ensure that the total congestion on the inner vertices ℓ_v is at most y_v. No flow goes through the spider-head vertices h_v, and at most x_v flow units of each type terminate at h_v. For terminal $t_i \in T$, no flow of type $j \neq i$ goes through t_i, and at most one unit of that flow may terminate at t_i, but the total amount of flow terminating at t_i is unrestricted. For the source vertex s, there is no restriction on the amount of flow of each type terminating at s.

Given a flow-path decomposition as above, the rounding algorithm performs the following three steps: *flow pairing, sampling of spider-head vertices,* and *sampling of flow-paths*. The flow pairing step is performed once at the beginning of the algorithm, and the next two steps are performed $(4k \log n)$ times.

Step 1 (Flow Pairing): Consider some spider-head non-terminal vertex h_v and the set F of flow-paths terminating at h_v. Recall that the total amount of flow terminating at h_v is at least $2x_v$, and the total amount of flow of each type is at most x_v. For each flow-path f terminating at h_v, we find another flow-path $R(f)$ terminating at h_v, such that:

- f and $R(f)$ belong to different terminals (different flow-types).
- For each flow-path f', there are at most two paths f for which $R(f) = f'$.
- If $R(f') = R(f'') = f$ then f' and f'' belong to different terminals.

We find the pairing as follows: while F contains at least two flow paths f, f' that belong to different terminals, we set $R(f) = f'$ and $R(f') = f$ and remove f, f' from F. When this process can no longer be performed, set F contains paths belonging to one terminal, t_i. Let F' be the set of all flow-paths belonging to t_i that terminate at h_v. Notice that it is possible that for some $f \in F'$, $R(f)$ has already been defined. The total flow on paths in F' is at most x_v. But the LP ensures that the flow from terminals $t_j \neq t_i$ to h_v is at least x_v. So for each flow-path $f \in F$ we can find a distinct path f' that terminates at h_v and belongs to a terminal different from t_i. We set $R(f) = f'$. (Notice that for paths $f \in F'$ for which $R(f)$ has been previously defined, we re-define $R(f)$ here).

The next two steps are repeated $(4k \log n)$ *times.*

Step 2 (Sampling of Spider Head Vertices): In this step, each spider-head vertex h_v is selected into set H with probability x_v.

Step 3 (Flow Sampling): Our final step is the flow sampling step. For each terminal t_i, we divide its set of flow paths into two subsets, $F_1(t_i)$ and $F_2(t_i)$. The set $F_1(t_i)$ consists of flow paths that terminate at $T^+ \setminus \{t_i\}$. The set $F_2(t_i)$ consists of flow paths that terminate at some spider-head vertex h_v.

- For each flow path $f \in F_1(t_i)$, we sample f with probability ϵ.
- For flow paths in $F_2(t_i)$, we perform the following random experiment. Let $\delta_v(t_i)$ denote the total flow from terminal t_i to a spider-head vertex $h_v \in H$ along the flow paths in $F_2(t_i)$. With probability $\frac{\delta_v(t_i)}{x_v}$, we choose to route through the vertex h_v. If we choose to route through h_v, then we sample

uniformly at random a flow path $f \in F_2(t_i)$ connecting t_i to h_v. If f is selected then we select $R(f)$ as well.

We set V' to contain all the vertices v for which $h_v \in H$ or ℓ_v belongs to one of the flow-paths that we have selected (in addition to vertices in $\{s\} \cup T$).

5.3 Cost and Feasibility Analysis

We first show that the expected cost of vertices in set V' is at most $O(k \log n)$OPT. We will then prove that for each terminal $t_i \in T'$, the graph induced by V' contains a set $F(t_i)$ of k paths with the desired properties.

Expected Cost: We start by bounding the expected solution cost. We need to analyze only the expected cost of vertices in $V \setminus T^+$. Vertex v belongs to the solution if $h_v \in H$, or ℓ_v belongs to one of the paths we selected. The probability that $h_v \in H$ is $O(x_v \cdot k \log n)$.

Consider now vertex $\ell_v \in V'$. Recall that the total flow that goes through ℓ_v is at most y_v. Let f be any flow-path traversing v. If it terminates in s or one of the terminals, then the probability that we select it is $O(\epsilon \cdot k \log n)$. Assume now that it terminates at some spider-head $h_{v'}$. Then the probability that $h_{v'} \in H$ is $O(x_{v'} \cdot k \log n)$. The probability that either f is sampled, or one of the at most two paths f' for which $R(f') = f$ is sampled, given that $h_{v'}$ is selected is $O(\epsilon / x_{v'})$.

So overall f is added to the solution with probability at most $O(\epsilon \cdot k \log n)$. Using the union bound, the probability that any path containing ℓ_v is chosen is at most $O(y_v \cdot (k \log n))$. Overall, the expected cost of vertex v is at most $(x_v + y_v)c(v) \cdot O(k \log n)$, and the expected solution cost is $O(k \log n)$OPT.

Feasibility: We now show that the solution produced by the above random process is feasible with high probability.

Lemma 5.1 *With high probability, for each terminal $t_i \in T$, there is a k-tuple $F(t_i)$ of k internally vertex disjoint paths in the graph induced by V', where each path $f \in F(t_i)$ connects t_i to $T^+ \setminus \{t_i\}$ and does not contain terminals as intermediate vertices. Moreover, for each $t' \in T \setminus T'$, at most one path in $F(t_i)$ terminates at t'.*

Proof: Fix some terminal t_i. We construct an auxiliary graph G_i as follows. Graph G_i contains t_i as a source and S as a super-sink. Graph G_i is identical to G, except that all vertices in $T'^+ \setminus \{t_i\}$ are merged into the super-sink S. Additionally, for each $t' \in T \setminus T'$, we add an edge from t' to S. We set the capacity of each vertex $v \in V^*$ to be $x_v + y_v$, and capacity of each terminal $t \in T \setminus T'$ to be 1.

For each flow-path f of type i in the LP-solution, if f terminates at s or t_j, $j \neq i$, we add path f to the graph G_i: we replace each ℓ_v on the path by v; if $t_j \in T'^+ \setminus \{t_i\}$ then the new path terminates at S; otherwise after reaching t_j it continues directly to S. If f terminates at some vertex h_v, we add a new flow-path that consists of concatenation of f and $R(f)$ to G_i: we replace h_v by v and $\ell_{v'}$ by v' for all v' on the new path. Notice that the terminal t_j where $R(f)$ starts belongs T'^+, and so the new path terminates at S.

Observe that we now send k units of flow from t_i to S in G_i. We first argue that resulting flow obeys the capacities assigned to the vertices. Consider some vertex $v \in V^*$, and let f be a flow-path in G_i traversing v. We group such paths into three sets: $S_1(v)$, $S_2(v)$ and $S_3(v)$. If f was obtained from a path f' in G' that

terminates in h_v, then $f \in S_1(v)$. Otherwise, f was obtained from a concatenation of paths f', f'' in G'. Assume w.l.o.g. that f' is the path that belongs to t_i. If $\ell_v \in f'$ then $f \in S_2(v)$; otherwise, $\ell_v \in f''$ and we set $f \in S_3(v)$. The total flow on paths in $S_1(v)$ is bounded by x_v. The total flow on paths in S_2 and S_3 is bounded by y_v: the flow traversing vertex ℓ_v in G' is at most y_v. If f'' is any flow-path that belongs to a terminal other than t_i and traverses ℓ_v, then there is at most one path f' belonging to t_i for which $R(f') = f''$. Therefore, the total flow via vertex v in G_i is at most $x_v + y_v$. Consider now some terminal $t \in T \setminus T'$. The total flow of type i entering t is 1, and therefore at most 1 unit of flow is routed through t.

Notice that in order to complete the proof, it is enough to show that with high probability, after the flow sampling step, there are k vertex disjoint paths connecting t_i to S in G_i. Assume otherwise. Then there exists a separator X of size $(k-1)$ that separates t_i from S in G_i after flow sampling. For each subset X of $k-1$ vertices in G_i, with $S, t_i \notin X$, let $\mathcal{E}_i(X)$ denote the event that X separates t_i from S after the flow sampling. Before the flow sampling step, the graph G_i contains a collection $D(t_i)$ of flow paths carrying at least one unit of flow from t_i to S such that no flow path contains any vertex from X. Thus if the event $\mathcal{E}_i(X)$ occurs, we did not sample any path in $D(t_i)$. We now bound the probability of the event $\mathcal{E}_i(X)$. Let $F_1'(t_i)$ denote the set of flow paths in $F_1(t_i)$ when mapped to the graph G_i. Similarly, let $F_2'(t_i)$ denote the set of flow paths in $F_2(t_i)$ when mapped to the graph G_i.

Consider first the case when the total flow on paths in $F_1'(t_i) \cap D(t_i)$ is at least $1/2$, so $|F_1'(t_i) \cap D(t_i)| \geq \frac{1}{2\epsilon}$. For each path $f \in F_1'(t_i) \cap D(t_i)$, we define a random variable $\gamma(f)$ that is equal to 1 if f is selected and equal to 0 otherwise. Let $\gamma = \sum_{f \in F_1'(t_i) \cap D(t_i)} \gamma(f)$. The probability that none of the paths in $F_1'(t_i) \cap D(t_i)$ is chosen in any given iteration of Step (3) is

$$\Pr[\gamma = 0] = \prod_{f \in F_1'(t_i) \cap D(t_i)} (1 - \epsilon) \leq e^{-\frac{1}{2}}.$$

Then the probability that $\gamma = 0$ in each one of the $(4k \log n)$ independent trials of Step (3) can be bounded by $e^{-\frac{1}{2}(4k \log n)} \leq n^{-2k}$.

Suppose now that flow paths in $F_2'(t_i) \cap D(t_i)$ carry more than $1/2$ unit of flow. We now analyze the probability that no path in $F_2'(t_i)$ is chosen by analyzing the probability of failure in a *single iteration* of the steps (2) and (3) above. For each spider-head vertex h_v, let Z_v be a $0/1$ random variable that is 1 if and only if each of the following events occurs:

- The vertex h_v is chosen in step (2).
- We chose to route through the vertex h_v in step (3) along a path in the set $F_2'(t_i) \cap D(t_i)$.

We define $Z = \sum_{h_v} Z_v$. Clearly, the set X disconnects t_i from $T \setminus \{t_i\}$ only if $Z = 0$. So we would like to bound the probability of this event. Note that the variables Z_v are independent of each other. Let $\delta_v'(t_i)$ denote the total flow from terminal t_i to a spider-head vertex $h_v \in H$ along the flow paths in $F_2'(t_i) \cap D(t_i)$. For each Z_v, we have

$$\Pr[Z_v = 1] = x_v \cdot \frac{\delta_v(t_i)}{x_v} \cdot \frac{\delta_v'(t_i)}{\delta_v(t_i)} = \delta_v'(t_i).$$

Thus using the fact that $\sum_v \delta_v'(t_i) \geq 1/2$, we get

$$\Pr[Z=0] = \prod_v \Pr[Z_v = 0] = \prod_v \left(1 - \delta'_v(t_i)\right) \leq e^{-\sum_v \delta'_v(t_i)} \leq e^{-1/2}.$$

Then the probability that $Z = 0$ in each one of the $(4k \log n)$ independent trials of Steps (2) and (3) can be bounded by $e^{-\frac{1}{2}(4k \log n)} \leq n^{-2k}$.

Hence in each case above, the probability of the event $\mathcal{E}_i(X)$ can be bounded by n^{-2k}. To complete the proof, using the union bound, we get

$$\sum_{t_i \in T} \sum_{\substack{X \subset V(G_i) \setminus \{S, t_i\}, \\ |X| = k-1}} \Pr[\mathcal{E}_i(X)] \leq n \cdot n^{k-1} \cdot \frac{1}{n^{2k}} = \frac{1}{n^k}$$

□

6 Concluding Remarks

The results in this paper represent progress towards closing the gaps in our understanding of the approximability of single-source k-vertex connectivity. While there still remains a separation between the upper and lower bounds in the single-source vertex connectivity, the gap in the upper and lower bounds for the general VC-SNDP is far more striking. For any fixed $k \geq 3$, the upper bound is a polynomial-ratio approximation algorithm while the lower bound is an APX-hardness. Perhaps both the approximability factor and the hardness factor can be much improved.

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] 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.

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

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

[5] 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.

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

[7] T. Chakraborty, J. Chuzhoy, and S. Khanna. Network Design for Vertex Connectivity. In *Proceedings of ACM Symposium on Theory of Computing* (STOC), 2008.

[8] C. Chekuri and N. Korula. Single-sink network design with vertex connectivity Requirements. *Manuscript*, April 2008.

[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 *Proceedings of ACM Symposium on Theory of Computing* (STOC), 2008.

[11] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM* 45(4) pp. 634-652, 1998.

[12] 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.

[13] A. Frank and T. Jordan. Minimal edge-coverings of pairs of sets. *Journal of Combinatorial Theory, Series B*, 65(1):73–110, 1995.

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

[15] Z. Galil and G. Italiano. Reducing edge connectivity to vertex connectivity. *ACM SIGACT News*, 22(1), pp. 57–61, 1991.

[16] 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.

[17] M. X. Goemans, A. V. Goldberg, É. Tardos, S. A. Plotkin, D. B. Shmoys, 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.

[18] 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.

[19] 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.

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

[21] P. N. Klein and R. Ravi. A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees. *Journal of Algorithms*, 19 (1): 104–115, 1995.

[22] 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.

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

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

[25] C. Lund, and M. Yannakakis. On the hardness of approximating minimization problems. *J.ACM 41,* 5, 960–981, 1994.

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

A Reducing Edge Connectivity to Vertex Connectivity

In this section we show that EC-SNDP is a special case of VC-SNDP. Our reduction also works in the single-source scenario and proves that the single source EC-SNDP is a special case of the single-source VC-SNDP.

Theorem 7 *Given any instance G of EC-SNDP, we can construct, in polynomial time, an instance G' of VC-SNDP, such that the optimal solutions of both instances have the same cost. If G is an instance of single-source EC-SNDP then G' is an instance of single-source VC-SNDP.*

Proof: Let $G = (V, E)$ be an instance of EC-SNDP, with costs $c(e)$ on edges e and connectivity requirements $r_{u,v}$ for $u, v \in V$. We construct an instance $G' = (V', E')$ of VC-SNDP as follows.

For each $v \in V$, and each $e \in E$ that is adjacent to v, we create a vertex e_v. Let $Z(v) = \{e_v \mid v \in e \in E\}$. Set V' of vertices is then $V' = V \cup \{Z(v) \mid v \in V\}$. Set E' of edges consists of two sets, $E' = E_1 \cup E_2$. The set E_1 contains *special edges* defined as follows: for each edge $e = (u, v) \in E$, there is a special edge $(e_u, e_v) \in E_1$ whose cost is $c(e)$. Set E_2 of edges contains non-special edges whose cost is 0. For each vertex $v \in V$, we add a non-special edge between each pair of vertices in set $\{v\} \cup Z(v)$. Let $E(v)$ denote the set of these edges. Notice that in the graph induced by E_2, for each $v \in V$ there is a separate connected component, which is a clique on vertices $\{v\} \cup Z(v)$.

Connectivity requirements are defined as follows: for each $u, v \in V$, the connectivity requirement is the same as in the original instance. For any other pair of vertices the connectivity requirement is 0. Notice that if G is an instance of single-source EC-SNDP then G' is an instance of single-source VC-SNDP.

Let E^* be any solution to the EC-SNDP instance G. We show a solution E^{**} to the VC-SNDP instance G' of the same cost. The solution is defined as follows. For each $e = (u, v) \in E^*$, we add (e_u, e_v) to E^{**}. We also add all non-special edges of E_2 to E^{**}. Clearly the cost of E^{**} is the same as the cost of E^*. We now show that it is a feasible solution to the VC-SNDP instance. Fix a pair $u, v \in V$ of vertices, and let $\mathcal{P}$ be the set of $r_{u,v}$ edge disjoint paths connecting u to v in OPT. For each path $p \in \mathcal{P}$, we define a path $h(p)$ in the sub-graph of G' induced by E^{**}, so that the set $\mathcal{P}' = \{h(p) \mid p \in \mathcal{P}\}$ of paths is vertex disjoint, and every path in the set connects u to v. Let $p = (v = v_1, v_2, \ldots, v_q = u)$, and let $e^i = (v_i, v_{i+1})$ for $1 \leq i < q$. We define path $h(p) = (v = v_1, e^1_{v_1}, e^1_{v_2}, e^2_{v_2}, \ldots, e^{q-1}_{v_{q-1}}, e^{q-1}_{v_q}, v_q = u)$. It is easy to see that for $p, p' \in P$ the resulting paths $h(p), h(p')$ are vertex disjoint: the only vertices appearing on $h(p)$ are of the form e_z, where e belongs to p. Therefore, if $h(p), h(p')$ share an intermediate vertex, the paths p, p' must share an edge, which is a contradiction.

We now show that the other direction also holds. Let E^* be any solution to the VC-SNDP instance. We show a solution E^{**} to the EC-SNDP instance of the same cost. Solution E^{**} is defined as follows: for each special edge $(e_u, e_v) \in E^*$, we add e to E^{**}. Clearly, the cost of E^{**} is the same as the cost of E^*. We now show that this defines a feasible solution. Let $u, v \in V$ be any pair of vertices and let $\mathcal{P}$ be the set of $r_{u,v}$

vertex disjoint paths connecting u to v in the sub-graph of G' induced by E^*. We define, for each $p \in P$, a path $h(p)$ connecting u to v in the sub-graph of G induced by E^{**}, so that for each pair $p, p' \in \mathcal{P}$, the paths $h(p), h(p')$ are edge disjoint.

Let $p \in \mathcal{P}$ be a path connecting u to v in E^*. Let (e_x, e_y), $(e_{x'}, e_{y'})$ be any pair of successive special edges on p. Recall that in the graph induced by set E_2 of non-special edges the connected components correspond to sets $Z(v')$ of vertices, for all $v' \in V$. Therefore, $y = x'$ must hold. Assume that the special edges appearing on path p are: $(e^1_{v_1}, e^1_{v_2}), (e^2_{v_2}, e^2_{v_3}), \ldots, (e^{q-1}_{v_{q-1}}, e^{q-1}_{v_q})$. Then $v_1 = v$ and $v_q = u$ must hold. We now define $h(p) = (v_1, v_2, \ldots, v_q)$. It now only remains to show that if $p, p' \in \mathcal{P}$, $p \neq p'$, then $h(p), h(p')$ are edge disjoint. Assume otherwise, and let $e = (x, y)$ be some edge appearing on both paths. Then both e_x and e_y appear on both p and p' as intermediate vertices, a contradiction. □