Approximating Fault-Tolerant Group-Steiner Problems

Rohit Khandekar* Guy Kortsarz † Zeev Nutov‡

January 10, 2010

Abstract

In this paper, we initiate the study of designing approximation algorithms for Fault-Tolerant Group-Steiner (FTGS) problems. The motivation is to protect the well-studied group-Steiner networks from edge or vertex failures. In Fault-Tolerant Group-Steiner problems, we are given a graph with edge- (or vertex-) costs, a root vertex, and a collection of subsets of vertices called groups. The objective is to find a minimum-cost subgraph that has two edge- (or vertex-) disjoint paths from each group to the root. We present approximation algorithms and hardness results for several variants of this basic problem, e.g., edge-costs vs. vertex-costs, edge-connectivity vs. vertex-connectivity, and 2-connecting a single vertex vs. two distinct vertices from each group. Main contributions of our paper include the introduction of general structural lemmas on connectivity and a charging scheme that may find more applications in the future. Our algorithmic results are supplemented by inapproximability results, which are tight in some cases.

*IBM T.J.Watson Research Center. Email: rohitk@us.ibm.com

†Rutgers University, Camden. Email: guyk@camden.rutgers.edu Partially supported by NSF grant 08129959

‡The Open University of Israel. Email: nutov@openu.ac.il

1 Introduction

The fault-tolerant network design problems are well-studied in the theory of combinatorial optimization and approximation algorithms. The basic goal in these problems is to design a minimum-cost network that satisfies some prescribed connectivity requirements. Higher connectivity requirements are usually enforced for fault-tolerance — in order to protect connectivity in the solution against edge or vertex failures.

In this paper, we study fault-tolerant versions of the Group-Steiner problem. In this problem, we are given a (undirected or directed) graph with edge- or vertex- costs, a root vertex and a collection of subsets (groups) of vertices. The objective is to find a minimum cost subgraph H of G that contains two edge- or vertex-disjoint paths from each group S_i to the root. This problem models the flexibility, often arising in problems in VLSI design [27], in connecting any vertices from a given group as well as the fault-tolerance which requires the solution to be robust under edge- or vertex-failures.

1.1 Previous work on fault-tolerant problems

Fault tolerant problems have been extensively studied in Combinatorial Optimization and Approximation Algorithms. Consider for example the well known Steiner Network problem. Given an undirected graph $G = (V, E)$ with edge-costs $\{c_e \mid e \in E\}$, and requirement r_{ij} for every pair of vertices $i, j \in V$, the goal is to find a minimum cost subgraph H of G that contains at least r_{ij} edge-disjoint paths between i and j, for all i, j. The network H is fault tolerant in the sense that a pair i, j can sustain $r_{ij} - 1$ link failures and still be connected. The best approximation ratio known for this problem is 2 due to Jain [15].

The internally-disjoint path version of the Steiner Network problem is very hard to approximate [18, 22, 9, 20]. The currently best known ratio for this problem is $O(k^3 \log n)$ for edge-costs due to Chuzhoy and Khanna [8] and $O(k^4 \log^2 n)$ for vertex-costs [25]. The rooted version, where a root s is given and $r_{ij} = 0$ for all pairs i, j so that $i \neq s$ and $j \neq s$, has gotten a significant attention recently [3, 6, 9, 24, 8, 25]. This problem is at least as hard to approximate as Directed Steiner Tree [20]. The best approximation ratio known for this problem for general rooted demands is $O(k^2)$ for edge-costs and $O(k^2 \log n)$ for vertex-costs [25], where $k = \max r_{ij}$. We mention that prior to the work of [25], a randomized approximation algorithm with ratio $k^{O(k^2)} \log^4 n$ was developed by Chakraborty, Chuzhoy, & Khanna [3], then improved to $k^{O(k)} \log n$ by Chekuri & Korula [6], and finally to $O(k^2 \log n)$ by Chuzhoy & Khanna [9, 8] and [24]. Note that k can be as large as $\Omega(n)$.

A particular case of fault tolerant problems with 2 disjoint paths has also received an attention in [1, 21, 6]. Lau et al. [21] presented an $O(\log^2 n)$-approximation algorithm for the problem of finding a minimum-cost 2-edge connected subgraph with *at least* k vertices. This problem can be seen as a fault-tolerant generalization of the k-MST problem which requires to find a minimum spanning tree on k vertices. The best known approximation ratio for the k-MST problem is 2 [11]. Chekuri and Korula [5] presented an $O(\log^2 n)$-approximation for the problem of finding a minimum-cost 2-vertex connected subgraph with at least k terminals. In [1, 6], the fault-tolerant version of the Buy-at-Bulk problem was studied, where two edge-disjoint paths are required to be included from every terminal to the root.

In the same spirit, in this paper we consider a generalization of the Group-Steiner Tree problem. In the Group-Steiner Tree problem, we are given a graph $G = (V, E)$, edge-costs $\{c_e : e \in E\}$, a root $r \in V$, and a collection of subsets (groups) $\mathcal{S} = \{S_1, \ldots, S_q\}$ of $V \setminus \{r\}$. The objective is to find a minimum cost subtree T of G that contains r and at least one vertex from each group $S_i \in \mathcal{S}$. The best known approximation ratio for this problem is $O(\log^3 n)$ [12], and an $\Omega(\log^{2-\epsilon})$-approximation threshold was established in [14]. The Fault-Tolerant Group-Steiner Tree problem, on the other hand, requires obtaining two (edge- or vertex-) disjoint paths between each group and the root. We are not aware of any previous work on Fault-Tolerant Group-Steiner problems.

1.2 Problem variants studied in this paper

One can define several variants of the Fault-Tolerant Group-Steiner problem, based on whether we desire 2-edge- or 2-vertex-connectivity, whether we have edge- or vertex-costs, whether we wish to 2-connect to the root a single vertex from each group or two distinct vertices from each group, etc. Below we formally define all the variants studied in this paper. Two paths are said to be *internally-disjoint* if they are vertex-disjoint except for their end-points. Each of the following problems takes a graph $G = (V, E)$ on n vertices with edge-costs $\{c_e \mid e \in E\}$ (or with vertex-costs $\{c_v \mid v \in V\}$), a root $r \in V$, and groups $\mathcal{S} = \{S_1, \ldots, S_q\}$ as input, and is required to compute a min-cost subgraph H of G with at least two edge/vertex-disjoint paths from each group S_i to the root, so that the end vertices of these paths are distinct. Unless stated otherwise, we consider the edge-cost version and assume that G is undirected. We also assume that the groups $S_1, \ldots, S_q$ are pairwise disjoint.[1] The vertices in the groups S_i are called *terminals*. We add the prefix EC- for edge-connectivity and the prefix VC- for vertex-connectivity. We add the suffix "-k" after the name of the problem if the instances are restricted to satisfy $|S_i| \leq k$ for all $i = 1, \ldots, q$.

- EC-FTGS: Here for every $i = 1, \ldots, q$, H should contain at least two edge-disjoint $S_i r$-paths; the end-points in S_i of these paths should be distinct.
- VC-FTGS: The same as EC-FTGS, except that the paths should be internally-disjoint.
- EC-FTGS-k and VC-FTGS-k: These are EC-FTGS and VC-FTGS, respectively, restricted to instances with $|S_i| \leq k$ for all $i = 1, \ldots, q$.

In the edge-connectivity case, for both edge and vertex-costs, the version in which the end-points in S_i of the two $S_i r$-paths may or may not be distinct, is easily reduced to EC-FTGS as follows. For every terminal s, add a new vertex s' of cost 0 connected to s with an edge of cost 0, and add s' to every group $S \in \mathcal{S}$ that contains s. After this transformation, we can assume, without loss of generality, that the two edge-disjoint paths from each group start from *distinct* terminals in that group. This may double the number of vertices, and cause a constant loss in approximation ratios that depend on n.

[1] This can be typically assumed by making multiple copies of the vertices in multiple groups and adding zero-cost edges connecting the different copies. This reduction, however, increases the number of vertices in the graph, thus possibly affecting the approximation ratio.

In the vertex-connectivity case, the reduction to VC-FTGS is carried out as follows. We make a copy of every vertex s' for every vertex $s \in S$. We connect s' to the neighbors of s. Clearly there are two vertex disjoint paths from s if there are two vertex disjoint paths from s, s' in the new graph.

We also consider the version when we insist that a single vertex from each group must be 2-edge-connected to the root. Namely, for every $i = 1, \ldots, q$ there should exist a vertex $v_i \in S_i$ such that H contains 2 edge-disjoint rv_i-paths. We call this version Single EC-FTGS.

1.3 Difficulties in approximationg Fault-Tolerant Group-Steiner problems

When two disjoint paths from every group to the root are required, we cannot use the standard transformation to Bartal trees [2, 10] as done in the approximation of the Group Steiner problem [12]. This is so because disjoint paths from a group to the root in a Bartal tree do not necessarily correspond to disjoint paths in the original graph.

We now give an evidence that an approximation that is polylogarithmic in n for either EC-FTGS-k and VC-FTGS-k may be very hard to obtain as it implies solving a long standing open problem. Note that we can reduce the Group Steiner problem to EC-FTGS-k or VC-FTGS-k problems by adding a new vertex, which is connected to the root by a zero-cost edge, to each group. Since we get one path for "free", any solution to EC-FTGS-k and VC-FTGS-k corresponds to a solution of the Group Steiner problem and vice-versa.

Now, since an algorithm for EC-FTGS-k or VC-FTGS-k cannot use Bartal trees, it must solve the Group Steiner problem as well **without** using Bartal trees. Designing a "natural" approximation algorithm with a polylogarithmic ratio for the Group Steiner problem without first reducing the graph into trees is a long standing open question, and seemingly a very hard one.

To the best of our knowledge, the best known approximation ratio for Group Steiner problem without using Bartal trees is $O(n^{\epsilon})$ for any constant $\epsilon > 0$, with running time $n^{f(1/\epsilon)}$. The *recursive greedy* technique [29, 19, 4], used in this algorithm, is a complex greedy approach with quite delicate analysis that seems inappropriate for the requirement of two disjoint paths. Thus even an n^{ϵ} approximation for every universal constant ϵ seems to be a quite significant challenge for our problems in the current state of knowledge and techniques.

In our opinion, this is a strong evidence that it is quite a challenge to get a polylogarithmic ratio for either EC-FTGS-k or VC-FTGS-k in polynomial time.

Remark: The above simple reduction shows that the $\Omega(\log^{2-\epsilon} n)$ approximation hardness of [14] for the Group Steiner problem applies also for EC-FTGS-k and VC-FTGS-k. However, from the above evidence, EC-FTGS-k and VC-FTGS-k may in fact be much harder to approximate than $\Omega(\log^{2-\epsilon} n)$, and it may be that a polynomial ratio is the best we can hope for.

1.4 Our results

We start with some notations. For two optimization problems $\mathcal{P}_1$ and $\mathcal{P}_2$, we say that $\mathcal{P}_1$ *is* $\mathcal{P}_2$*-hard* if existence of a polynomial time ρ-approximation algorithm for $\mathcal{P}_1$ implies existence of a polynomial time ρ-approximation algorithm for $\mathcal{P}_2$. Similarly, $\mathcal{P}_1$ is $\Omega(f(n))$-hard if there exists a constant $\epsilon > 0$ so that $\mathcal{P}_1$ admits no $\epsilon f(n)$-approximation algorithm, unless P=NP.

Problem	Edge-Connectivity (EC)	Vertex-Connectivity (VC)
FTGS-2	3.55, Vertex-Cover-hard $O(\log n)$, $\Omega(\log n)$-hard (vertex-costs)	$O(\log^2 n)$
FTGS-k	$O(k \log^2 n)$	$O(k \log^2 n)$
FTGS	$O(\sqrt{n} \log n)$ Group Steiner Tree-hard	$O(\sqrt{n} \log n)$ Group Steiner Tree-hard
Single FTGS	Label Cover-hard (directed)	Label Cover-hard (directed)

Table 1: Approximation ratios and hardness results for FTGS variants. The extra assumptions, if any, are given in the parentheses.

Our main results are summarized in Theorem 1.1 and in Table 1.

Theorem 1.1

(i) EC-FTGS-2 *admits the following approximation ratios:*
$(2+\gamma)$ *for edge costs, where* $\gamma < 1.55$ *is the best approximation ratio for the* Steiner Tree *problem, and* $O(\log n)$ *for vertex costs. Moreover, the edge-cost version is* Vertex-Cover*-hard and the vertex-cost version is* $\Omega(\log n)$*-hard.*

(ii) EC-FTGS-k *and* VC-FTGS-k *admit an* $O(k \log^2 n)$*-approximation algorithm.* EC-FTGS *and* VC-FTGS *admit an* $O(\sqrt{n} \log n)$*-approximation algorithm.* EC-FTGS *and* VC-FTGS *are* Group Steiner Tree*-hard and thus are* $Omega(\log^{2-\epsilon} n)$*-hard for any constant* $\epsilon > 0$*.*

(iii) *The directed version of* Single EC-FTGS *problem admits no* $2^{\log^{1-\epsilon} n}$*-approximation for any constant* $\epsilon > 0$*, unless* NP $\subseteq$ DTIME$(n^{\text{polylog}(n)})$.

The results (i), (ii) and (iii) are proved in Sections 2 and 3 and 4, respectively.

1.5 A summary of our techniques

In Part (i), we observe that *all* terminals need to be connected to the root, thus our algorithm first computes a Steiner tree connecting all the terminals to the root (not necessarily by disjoint paths). We then note that the residual problem can be reduced to the one of covering an *uncrossable* set-family by edges. In the case of edge-costs, this problem can be solved using the primal-dual method of Goemans et al. [13], or via the iterative rounding technique of Jain [15]. Overall we obtain a $(2+\gamma)$-approximation for this problem if we use a γ-approximation algorithm for Steiner Tree as a subroutine. Since $\gamma < 1.55$ is known [28], we get a 3.55-approximation. A similar approach works for vertex-costs using the $O(\log n)$-approximation algorithm of [16] for computing the Steiner tree, and using in the augmentation step the algorithm of [23] instead of that of [13].

Our main technical contribution is in our algorithms for problems in Part (ii). Our algorithm for VC-FTGS-2 also reduces the problem to an augmentation problem, by first computing a Steiner tree connecting all the terminals to the root. To solve this augmentation problem, we introduce an interesting graph theoretic lemma (Lemma 3.4). Our lemma provides an equivalence between 2-connecting two vertices to the root and 2-connecting a single carefully chosen vertex

to the root. We believe that this lemma may be of independent interest. For every vertex v, we let the profit $p(v)$ of v be the number of groups that will be satisfied if v is 2-connected to the root. The *density* version of the residual problem seeks a subgraph that minimizes the ratio of its cost over the total profit of vertices that are 2-vertex-connected to the root. We then use the $O(\log n)$-approximation algorithm by Chekuri and Korula [5] for this density problem, combined with the greedy method.

Our algorithms for VC-FTGS-k derives the intuition from that for VC-FTGS-2, and also uses other techniques. Let us call a pair of terminals from a group that are connected to the root via vertex-disjoint paths in the optimum solution a *twin pair* (see Definition 3.1). We divide the progress of our algorithm into logarithmically many phases. In each phase, our algorithm first tries to connect the twin pairs from at least half of the remaining groups to the root (not necessarily by vertex-disjoint paths). Unfortunately, we do not know the twin pairs a-priori. Nevertheless, we show that one can use several applications of the p-Steiner tree algorithm [7] to obtain a relatively low-cost tree that achieves this. The residual problem is now similar to the augmentation problem considered for the case of VC-FTGS-2. We once again use our graph theoretical lemma (Lemma 3.4), relate 2-connecting a twin pair to 2-connecting a carefully chosen vertex, define $p(v)$ of such a vertex as in the VC-FTGS-2 algorithm. It turns out that a naive application and analysis of the greedy method as in our VC-FTGS-2 algorithm gives an approximation ratio of $O(k^2 \log^2 n)$, since there could be $\Omega(k^2)$ pairs that may claim profit for a single group. An important component in our analysis is a careful counting of how the profit can be claimed to prove an overall approximation factor of $O(k \log^2 n)$.

The last statement in Part (ii) of Theorem 1.1 then easily follows. To "cover" small groups, i.e., groups of size at most $k = \sqrt{n}/\log n$, we use the algorithm above. Since the groups are disjoint, the number of large groups is at most $n/k = \sqrt{n}\log n$. We cover the large groups one at a time. Overall, the approximation ratio is $O(\min\{k \log^2 n, n/k\}) = O(\sqrt{n}\log n)$.

The proof of part (iii) follows from a reduction from the Min-Rep problem.

2 Proof of Part (i)

We start with some definitions. An edge e is said to *cover* a subset $X \subset V$ of vertices if exactly one end-point of e lies in X. Let $\mathcal{F}$ be a collection of subsets of V. We say that an edge-set E' covers $\mathcal{F}$ if for each $X \in \mathcal{F}$, there is an edge $e \in E'$ that covers X. The Set-Family Edge-Cover problem with edge-cost is to find a minimum-cost collection of edges E' that covers $\mathcal{F}$. In the vertex-cost version, we wish to minimize the total cost of vertices incident to E' that covers $\mathcal{F}$. The family $\mathcal{F}$ may not be given explicitly, but we require that certain queries related to $\mathcal{F}$ can be answered in polynomial time. Specifically, we assume that, in the edge-cost version, for any edge-set E', the inclusion minimal members of $\mathcal{F}$ that are *not* covered by E' can be computed in polynomial time; while, in the vertex-cost version, for any $s, t \in V$, one can compute in polynomial time a min-cost cover of all members of $\mathcal{F}$ that separate s and t.

We call a family $\mathcal{F}$ of sets *uncrossable* if $X \cap Y, X \cup Y \in \mathcal{F}$ or $X \setminus Y, Y \setminus X \in \mathcal{F}$ for any $X, Y \in \mathcal{F}$. Our algorithms for EC-FTGS-2 problem with edge-costs or vertex-costs use the following results, respectively.

Theorem 2.1 (Goemans et al. [13]) *The* Set-Family Edge-Cover *problem with edge-costs and*

with uncrossable set-family $\mathcal{F}$ admits a 2-approximation algorithm.

Theorem 2.2 (Nutov [23]) *The* Set-Family Edge-Cover *problem with vertex-costs and with uncrossable set-family $\mathcal{F}$ admits an $O(\log n)$-approximation algorithm.*

2.1 Algorithmic results

Since the problem insists that the two edge-disjoint paths from each group must start at distinct terminals in the group, the optimum solution contains a Steiner tree containing all the terminals and the root. Our algorithm first finds a Steiner tree T that connects all the terminals to the root. If we use an α-approximation algorithm for this step ($\alpha < 1.55$ for edge-costs [28] and $\alpha = O(\log n)$ for the vertex-costs [16]), we get $\text{COST}(T) \leq \alpha \cdot \text{OPT}$, where OPT denotes the optimal cost of a solution.

For $X \subset V$, let $\mathtt{deg}_T(X)$ denote the number of edges in T from X to $V \setminus X$. Define an instance of Set-Family Edge-Cover by setting

$$\mathcal{F} = \{X \subseteq V \setminus \{r\} \mid \mathtt{deg}_T(X) = 1, S \subseteq X \text{ for some } S \in \mathcal{S}\}.$$

We now present two important observations.

Lemma 2.3 *For $I \subseteq E \setminus E(T)$, the set $T \cup I$ is a feasible solution to* EC-FTGS-2 *if, and only if, I covers $\mathcal{F}$.*

Proof: Note that T has a path from each terminal to the root. Thus by Menger's Theorem, $H = T \cup I$ is a feasible solution to EC-FTGS-2 if, and only if, $\mathtt{deg}_H(X) \geq 2$ for every set $X \subseteq V \setminus \{r\}$ that contains some group $S \in \mathcal{S}$. As $\mathtt{deg}_T(X) \geq 1$ for any $X \subseteq V \setminus \{r\}$ that contains at least one vertex from some group, we obtain that the latter condition is equivalent to $\mathtt{deg}_I(X) \geq 1$ for every $X \in \mathcal{F}$. ∎

Lemma 2.4 *The set family $\mathcal{F}$ is uncrossable.*

Proof: Note that by the definition of $\mathcal{F}$, $X \in \mathcal{F}$ if, and only if, X is a union of a rooted proper subtree of T that contains a group $S \in \mathcal{S}$ and some subset of vertices not in T. Let $X, Y \in \mathcal{F}$. Then $X \cap T, Y \cap T$ are disjoint, or one of them contains the other. In the former case, we have $X \setminus Y, Y \setminus X \in \mathcal{F}$; e.g., $X \setminus Y \in \mathcal{F}$ since $(X \setminus Y) \cap T = X \cap T$, hence $X \setminus Y$ is a union of the subtree contained in X and the vertex subset $X \setminus (T \cup Y)$ disjoint to T. In the latter case, $X \cap Y, X \cup Y \in \mathcal{F}$; e.g., if $X \subseteq Y$, then $X \cap Y$ is a union of the subtree contained in X and some vertices not in T, while $X \cup Y$ is the union of the subtree contained in Y and some vertices not in T. ∎

It is easy to check that for any edge set $I \subset E \setminus E(T)$, the minimal members of the family $\mathcal{F}$ not covered by I can be computed in polynomial time. Moreover, for any $s, t \in V$, a min-cost cover of all members in $\mathcal{F}$ that separate s and t can also be computed in polynomial time. Thus, we can use the algorithms in Theorems 2.1 and 2.2 respectively to complete the solutions for the edge- and vertex-cost versions.

2.2 Hardness of approximation results

We now show that EC-EFTGS-2 is Vertex-Cover-hard in the case of edge-costs, and that it is Hitting-Set-hard, i.e., $\Omega(\log n)$-hard, in the case of vertex-costs.

Let $J = (V_J, E_J)$ be an instance of Vertex-Cover. Define an instance $\{G = (V, E), \{c_e : e \in E\}, r, \mathcal{S}\}$ of 2-EC-FTGS-2 as follows. Set $V = V_J \cup \{a, r\}$, connect every vertex in V_J to a with an edge of cost 0 and connect a to r with an edge of cost 0. Let T denote the set of these zero-cost edges. Connect each vertex in V_J to r with an edge of cost 1 each. The set $\mathcal{S}$ of pairs is defined by edges of E_J, namely, $\mathcal{S} = \{\{u, v\} \mid (u, v) \in E_J\}$. Note that the optimum solution, without loss of generality, picks all the edges in T. It is now easy to see that $T + H$ is a feasible solution to the obtained instance of EC-FTGS-2 if and only if the set of end-points of the edges in H is a vertex-cover in J.

In the case of vertex-costs, EC-FTGS-2 is easily reduced to the Steiner Tree problem with vertex-costs which is known to be Hitting-Set-hard [16]. Given an instance $\{J = (V_J, E_J), r, S\}$ of Steiner Tree with vertex-costs, for every $s \in S$ add a copy s' of cost 0 and connect s' to r. The set of pairs is $\mathcal{S} = \{\{s, s'\} \mid s \in S\}$. It is easy to see that T is a feasible solution to Steiner Tree with vertex-costs if and only if $T \cup \{(r, s') \mid s \in S\}$ is a feasible solution to the constructed 2-EC-FTGS-2 instance.

The proof of Part (i) of Theorem 1.1 is thus complete.

3 Proof of Part (ii)

3.1 Algorithm for VC-FTGS-2

In this section, we introduce our main technical ideas. We present an $O(\log^2 n)$-approximation algorithm for VC-FTGS-2.

As in the edge-connectivity case, we first compute a Steiner tree T of cost $\textsc{cost}(T) \leq \alpha \cdot \textsc{opt}$ connecting all terminals to the root. We have $\alpha < 1.55$ for edge-costs and $\alpha = O(\log n)$ for vertex-costs. Our algorithm uses set-cover like approach in which we iteratively add partial solutions with low *density*, i.e., low cost to profit ratio, till we complete the solution. We get one logarithmic factor in the approximation from the set-cover analysis (and since the number of groups is $O(n)$) and another logarithmic factor from the fact that we can only compute $O(\log n)$ approximation to the minimum-density subproblem.

Given a partial solution $I \subset E \setminus E(T)$, let the *deficiency* of I be the number of groups in $\mathcal{S}$ that are not 2-vertex-connected to r in $T \cup I$. Let the *density* of an edge set $F \subset E \setminus (E(T) \cup I)$ be $\textsc{cost}(F)$ divided by the decrease in the deficiency caused by adding F to $T \cup I$. The following two lemmas captures the essence of our algorithm for VC-FTGS-2.

Lemma 3.1 *Given a partial solution $T \cup I$, the problem of computing a minimum density augmenting edge set $F \subset E \setminus (E(T) \cup I)$ for* VC-FTGS-2 *admits an $O(\log n)$-approximation algorithm.*

Lemma 3.2 *The algorithm in Lemma 3.1 can be used to obtain $O(\log^2 n)$ approximation for* VC-FTGS-2.

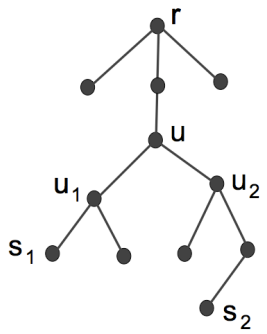

Figure 1: An example of converting the instance into one satisfying `Property P`.

As mentioned, the proof of Lemma 3.2 follows from the standard set-cover like analysis and is omitted. In the rest of this section we prove Lemma 3.1. We ignore the groups in $\mathcal{S}$ that are already connected in $T \cup I$ to the root via 2 vertex-disjoint paths starting from distinct vertices.

We start by recalling some definitions. A vertex $v \in V$ is a cut-vertex of a graph H if $H \setminus \{v\}$ has more connected components than H. A cut-vertex v of H is said to *separate* vertex r and set $S \subset V \setminus \{r, v\}$ if r and S belong to the same connected component of H but $H \setminus \{v\}$ does not contain a path from r to any vertex in S. Consider a group $\{s_1, s_2\} \in \mathcal{S}$. By Menger's Theorem we have:

Proposition 3.1 *A subgraph that contains an rs_1-path and an rs_2-path, contains such paths that are internally vertex disjoint if and only if it has no cut-vertex that separates r and $\{s_1, s_2\}$.*

Now think of the tree T as being rooted at r. For any two vertices $s_1, s_2 \in T$, we define $\text{LCA}(s_1, s_2)$ to be the least common ancestor of s_1 and s_2 in T. Consider $S = \{s_1, s_2\} \in \mathcal{S}$ with $u = \text{LCA}(s_1, s_2)$ in T. Note that $u \neq r$ since S is not 2-vertex connected to r via paths starting from s_1 and s_2. Let $\mathcal{U}(S) = \{u_1, u_2\}$ be two (possibly identical) vertices defined as follows.

- If $u \notin \{s_1, s_2\}$, then let u_1 (resp. u_2) be the child of u that lies on the rs_1- (resp. rs_2-) path in T.
- If $u \in \{s_1, s_2\}$, then let $u_1, u_2 = u$.

Define a family $\mathcal{U} = \mathcal{U}(\mathcal{S}, T)$ of pairs (groups) as $\mathcal{U} = \{\mathcal{U}(S) \mid S \in \mathcal{S}\}$. Note that the pairs in $\mathcal{U}$ may not be disjoint, and that some "pairs" in $\mathcal{U}$ are in fact singletons. The following lemma captures the "equivalence" between covering pairs in $\mathcal{S}$ and pairs in $\mathcal{U}$.

Lemma 3.3 *For any edge set $F \subset E \setminus (E(T) \cup I)$, the solution $H = T \cup I \cup F$ contains 2-vertex disjoint paths (with distinct starting points) between r and S if, and only if, it contains 2-vertex disjoint paths (with distinct starting points) between r and $\mathcal{U}(S)$.*

Proof: Let $u = \text{LCA}(s_1, s_2)$ where $S = \{s_1, s_2\}$. By Proposition 3.1, S is 2 vertex-connected to r in $H = T \cup I \cup F$ if and only if H has no cut-vertex separating S from r; namely, no vertex on

the ur-path in T is a cut-vertex of H. By the definition of $\mathcal{U}(S) = \{u_1, u_2\}$ and Proposition 3.1, this is equivalent to the property that $\mathcal{U}(S)$ is 2-connected to r in H. ∎

The above lemma implies that the densities of F w.r.t. $\mathcal{S}$ and w.r.t. $\mathcal{U}$ are equal. Furthermore, $T \cup I \cup F$ is a feasible solution w.r.t. $\mathcal{S}$ if and only if it is w.r.t. $\mathcal{U}$. Note that the groups $\mathcal{U}$ satisfy a special property, which is crucial in rest of the analysis.

> `Property P`: For all groups $\{u_1, u_2\} \in \mathcal{U}$, either $u_1 = u_2$ or u_1, u_2 have the same parent in T.

Lemma 3.4 *Let $U = \{u_1, u_2\} \in \mathcal{U}$. For $F \subset E \setminus (E(T) \cup I)$, the graph $H = T \cup I \cup F$ contains a u_1r-path and a u_2r-path that are internally vertex-disjoint if and only if H contains 2 internally-disjoint paths to r from either u_1 or u_2.*

Proof: The proof uses Menger's Theorem and `Property P` of the groups in $\mathcal{U}$.

Suppose that H contains 2 internally vertex-disjoint paths from u_1 to r. Then H has no cut-vertex separating r and u_1, by Menger's Theorem. In particular, there is no cut-vertex separating r and $\{u_1, u_2\}$. Thus H contains a u_1r-path and a u_2r-path that are internally vertex-disjoint, by Proposition 3.1.

Suppose now that H contains a u_1r-path and a u_2r-path that are internally vertex-disjoint. Now we use `Property P`. If $u_1 = u_2$, the proof is complete. Assume therefore that $u_1 \neq u_2$ and that they have a common parent u in T and assume to the contrary that H has no pair of internally vertex-disjoint u_ir-paths for $i = 1, 2$. Then by Menger's Theorem, there are cut-vertices v_1, v_2 in H, where v_i separates u_i from r. As u_1, u_2 have a common parent $u \neq r$, any vertex separating r from one of u_1, u_2 must lie on the ur-path in T. If $v_1 = v_2 = v$, then v separates both u_1, u_2 from r contradicting the assumption (by Proposition 3.1). Thus $v_1 \neq v_2$, so one of v_1, v_2 is distinct from u, say $v_1 \neq u$. The graph $H \setminus \{v_1\}$ contains a u_2r-path. As $T \setminus \{v_1\}$ contains a u_1u_2-path, this implies that $H \setminus \{v_1\}$ contains a u_1r-path. This contradicts that v_1 separates u_1 and r. ∎

Thus the original density problem can be reduced to the following problem. Given a collection of groups $\{u_1^i, u_2^i\}$ for $i = 1, 2, \ldots$, find a subset $F \subset E \setminus (E(T) \cup I)$ such that the ratio of $\textsc{cost}(F)$ to the number of groups i such that at least one of u_1^i or u_2^i has 2-vertex-disjoint paths to r in $E(T) \cup I \cup F$.

The problem of finding a subgraph that minimizes the ratio of its cost over the total profit of vertices that are 2-vertex-connected to the root was studied by Chekuri and Korula [5], who gave an $O(\log n)$-approximation for the problem. We use their algorithm to compute an $O(\log n)$-approximation for the density version of our problem, as follows. The input to the algorithm of Chekuri and Korula is the original graph with root r and with *profits* on vertices defined as follows. Let the *profit* $p(u)$ of a vertex u be the number of groups i such that $u \in \{u_1^i, u_2^i\}$. Thus $p(u)$ denotes the number of new groups that would get connected to the root via 2 vertex-disjoint paths provided u gets connected to the root via 2 vertex-disjoint paths. Note that since both u_1^i or u_2^i may claim profit for covering group i, we may overestimate the profit of 2-vertex connecting a subset of vertices to r by a factor of 2. This introduces another factor 2 in the ratio. Using the Chekuri-Korula algorithm, we compute a subgraph which yields an $O(\log n)$ approximation for minimizing the ratio of its cost over the total profit of its vertices that are 2-vertex-connected to the root. This subgraph yields an $O(\log n)$ approximation to the problem defined in Lemma 3.1.

Thus the proof is complete.

(i) Initialize subgraph $\mathcal{H} \leftarrow \emptyset$ and $q' \leftarrow q$ to be the number of uncovered groups.

(ii) If $q' > 0$, begin a phase:

  (a) Assign a cost of zero to all vertices in $\mathcal{H}$.

  (b) `Find Twin-pairs`: Compute a subgraph H of cost $O(\text{OPT} \cdot k \log n)$ that contains twin pairs (Definition 3.1) from at least $q'/2$ uncovered groups.

  (c) `Cover`: Compute a subgraph I of cost $O(\text{OPT} \cdot k \log n)$ that covers at least $q'/2$ uncovered groups.

  (d) Update $\mathcal{H} \leftarrow \mathcal{H} \cup H \cup I$ and update q' to be the number of uncovered groups in $\mathcal{H}$.

(iii) Output $\mathcal{H}$.

Figure 2: An outline of our algorithm for VC-FTGS-k with vertex-costs.

3.2 Algorithm for EC-FTGS-k and VC-FTGS-k with edge/vertex costs

We present an algorithm for VC-FTGS-k with vertex-costs, which is more general than the case of edge-costs. Adaptation of this algorithm to EC-FTGS-k is easy.

Fix an optimum solution OPT with cost also denoted by OPT. To simplify the presentation, the algorithm given below is assumed to know the value of OPT. In reality, the algorithm tries all possible guesses for the power of 2 that is closest to OPT and picks the cheapest solution among those computed for each of these guesses.

Our algorithm has logarithmic number of phases. In each phase, it covers at least half of the remaining groups. A high-level pseudo-code of the algorithm is given in Figure 2. At the beginning of each phase, we make the cost of the vertices that are already picked in the solution $\mathcal{H}$ *zero*. In what follows, we analyze a single phase which begins with q' uncovered groups overall. The groups in $\mathcal{S}$ that are already covered are ignored. In what follows, we explain how to implement steps `Find Twin-pairs` and `Cover` respectively.

3.2.1 How to implement step `Find Twin-pairs`

Definition 3.1 *For a group $S \in \mathcal{S}$, we say that the terminals $s_1, s_2 \in S$ form a* twin pair *if* OPT *contains two internally-disjoint paths from s_1 and s_2 to r. If there are more than one such pairs for a group, we designate exactly one of these pairs as a* twin pair *arbitrarily.*

We do not know which terminals form twin pairs a-priori. Nevertheless, we can compute a low-cost tree that connects the twin pairs from at least half of the remaining groups to the root, as shown below. We iteratively use p-Steiner tree algorithm for $p = q'$. Recall that the p-Steiner Tree problem is to compute a minimum-cost tree that connects at least p terminals to the root. Let H denote the union of the p-Steiner trees computed so far. Assume that the number of uncovered groups is at least $q'/2$.

Lemma 3.5 *Assign a cost of zero vertices in H. Now apply $(c \log n)$-approximation algorithm [7] for the p-Steiner tree problem (where c is a constant) for the instance given by H,*

root r, $p = q'$ and terminals as the vertices in the the union of uncovered groups in $\mathcal{S}$ but not in H: $\{v \in S \mid S \in \mathcal{S}$ is uncovered, $v \notin H\}$. If the cost of the computed Steiner tree is more than $(c \log n) \cdot \text{OPT}$, then H contains the twin pairs for at least $q'/2$ uncovered groups.

Proof: Assume on the contrary that H does not contain twin pairs for at least $q'/2$ uncovered groups. Thus the OPT solution connects at least $2 \cdot q'/2 = q'$ terminals in this p-Steiner tree instance to the root. Since we use a $(c \log n)$-approximation, the cost of the computed Steiner tree is at most $(c \log n) \cdot \text{OPT}$. This is a contradiction, and the lemma follows. ∎

We run the p-Steiner tree algorithm iteratively while the cost of the new tree computed is at most $c \log n \cdot \text{OPT}$. Since H contains at least $p = q'$ new terminals in each iteration, the total number of invocations of such p-Steiner tree algorithm is at most $2q' \cdot k/q'$. This holds since the size of each group is at most k. Thus the total cost of the step `Find Twin-pairs` is $\text{COST}(H) \leq O(\text{OPT} \cdot k \log n)$.

3.2.2 How to implement step `Cover`

Even if H contains the twin pairs of at least $q'/2$ uncovered groups, we still do not know which of the terminals in H form twin pairs. We therefore need one more definition.

Definition 3.2 *Let T be a spanning tree of H. We say that a vertex u_1* can help *an uncovered group S if there exist distinct vertices $s_1, s_2 \in S \cap T$ and another vertex u_2 so that u_1, u_2 have the same parent $u = \text{LCA}(s_1, s_2)$ in the tree T. The* profit *$p(u)$ of vertex u is defined as the number of uncovered groups in $\mathcal{S}$ that u can help.*

The intuition of the above definition comes from our algorithm for VC-FTGS-2, and in particular, from Lemma 3.4. Note that the profit of a vertex u is the number of uncovered groups that will get covered if u gets connected to the root via 2 internally-disjoint paths.

Since more than one vertex can claim a profit for covering a single group, it is important to understand how many vertices can help a particular group. Since there are at most k terminals in any group $S \in \mathcal{S}$, it is obvious that there can be at most $O(k^2)$ vertices that can help S. This crude upper bound comes from the fact that there are $O(k^2)$ pairs $s_1, s_2 \in S$ that can give rise to such vertices. However the following lemma presents a careful counting of such vertices.

Lemma 3.6 *There are $O(k)$ vertices that can help any single group $S \in \mathcal{S}$.*

Proof: Consider tree T with terminals in group S marked as s (see Figure 3). Further consider a subtree T' restricted to terminals in S. The vertices $u \in T'$ that can play a role of $\text{LCA}(s_1, s_2)$ for $s_1, s_2 \in S$ (these are shown as red squares in Figure 3) have a degree of at least 3 in T', i.e., $\texttt{deg}_{T'}(u) \geq 3$. Thus if a vertex v can help group S, it must be a child of a vertex u with $\texttt{deg}_{T'}(u) \geq 3$. The number of children of a vertex u is $\texttt{deg}_{T'}(u) - 1$. Therefore, the number of vertices v that can help S is at most $\Sigma u_{\{u | deg_{T'}(u) \geq 3\}}(deg_{T'}(u) - 1)$.

Since T' is a tree induced on at most k terminals of S, the tree has at most k leaves. By a simple counting argument, it therefore follows that the desired sum $\Sigma u_{\{u | deg_{T'}(u \geq 3\}}(deg_{T'}(u) - 1)$ is $O(k)$. The lemma thus holds. ∎

Now we have all the ingredients to present the step `Cover`. We again use the $O(\log n)$-approximation algorithm of Chekuri and Korula [5] that was also used in Section 3.1, for the

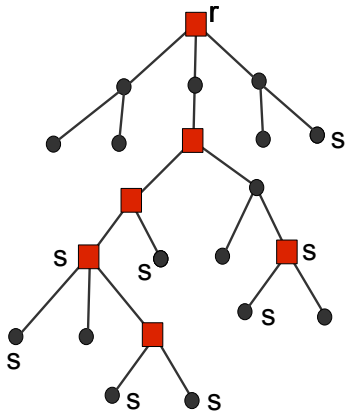

Figure 3: The vertices marked with s are terminals in a group S. The vertices marked with red squares can take a role of $\textsc{lca}(s_1, s_2)$ for a pair $\{s_1, s_2\}$ in group S. The children of red squares that lie on a path from some $s \in S$ to r can help S. There are $O(k)$ such vertices.

problem of finding a subgraph that minimizes the ratio of its cost over the total profit of vertices that are 2-vertex-connected to the root. Recall that the subgraph induced by OPT has cost at most OPT and profit at least $q'/2$ (since T contains twin pairs from at least $q'/2$ uncovered groups). Thus the cost-to-profit ratio of the solution computed by our algorithm is at most $O(\log n \cdot \textsc{opt}/q')$. From Lemma 3.6, we get that the ratio of the cost of this subgraph to the number of uncovered groups covered is at most $O(k \log n \cdot \textsc{opt}/q')$.

We apply such algorithm iteratively till end of the phase, i.e., at least $q'/2$ previously uncovered groups are covered in this phase, and call this subgraph I. From the above analysis, it is clear that the total cost of step `Cover` is $\textsc{cost}(I) = q' \cdot O(k \log n \cdot \textsc{opt}/q') = O(k \log n \cdot \textsc{opt})$. Thus the total cost of a phase is $\textsc{cost}(H)+\textsc{cost}(I) = O(\textsc{opt} \cdot k \log n)$. Since there are $O(\log q) = O(\log n)$ phases overall, the total cost of the algorithm is $O(\textsc{opt} \cdot k \log^2 n)$ as desired.

As was mentioned in the Introduction, an $O(\sqrt{n} \log n)$-approximation algorithm for VC-FTGS immediately follows from the $O(k \log^2 n)$-approximation for VC-FTGS-k, and thus the proof of Part (ii) of Theorem 1.1 is now complete.

4 Proof of Part (iii)

To present our hardness results, we introduce the Min-Rep problem [26]. The input consists of a bipartite graph $G = (V_1, V_2, E)$, and two disjoint partitions $V_1 = \bigcup_{i=1}^{q} A_i$ and $V_2 = \bigcup_{i=1}^{q} B_i$ of V_1 and V_2 respectively. Let $n = |V_1| + |V_2|$. The above instance induces a bipartite *supergraph* $\tilde{G} = (\{A_i\}, \{B_i\}, \bar{E})$. The *supervertices* of this supergraph are $\{A_i\}_{i=1}^{q} \cup \{B_i\}_{i=1}^{q}$. A *superedge* between A and B is added if there exist $a \in A$, $b \in B$ so that $(a, b) \in E$. We say that two vertices $a \in A$ and $b \in B$ *Min-Rep-cover* (A, B) if $(a, b) \in E$. A subset $C \subseteq V_1 \cup V_2$ of vertices is said to *cover* a superedge (A, B) if it contains a pair $a \in A$ and $b \in B$ that covers (A, B). A subset C is a *Min-Rep-cover* if it covers all the superedges.

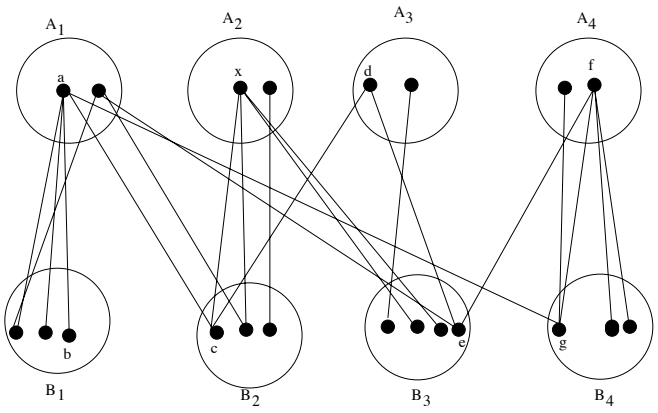

Figure 4: A yes instance. The vertices a, x, d, f, b, c, e, g are a Min-Rep-cover of the instance

Theorem 4.1 (Raz [26]) *For any constant $\epsilon > 0$, there is a reduction from any NP-complete problem instance to a* Min-Rep *instance $(G, \tilde{G})$ so that for a yes instance, there exists a set C containing one vertex out of any supervertex that is a* Min-Rep*-cover, but for a no instance, every* Min-Rep*-cover contains at least $q \cdot 2^{\log^{1-\epsilon} n}$ vertices of $V_1 \cup V_2$.*

We reduce Min-Rep instance to an instance $\hat{G}$ of directed Single EC-FTGS. Add all the vertices of the Min-Rep instance to $\hat{G}$. For every edge in $e = (a, b) \in E$, add a vertex v_e to $\hat{G}$ in the middle of e and replace e by two directed edges from v_e to each of a and b. These edges have cost 0. Add a root r and add a directed edge of cost 1 from every vertex in $V_1 \cup V_2$ to r. Every group corresponds to a superedge (A, B). Let g_{AB} be this group. For every $e = (a, b)$ where $a \in A, b \in B$, add v_e to g_{AB}. This finishes the description of the reduction.

We show that every solution to the Single EC-FTGS instance $\hat{G}$ defines a solution for the Min-Rep instance of the same cost and a solution for Min-Rep defines a solution for Simple EC-FTGS instance $\hat{G}$ of the same cost.

First consider a solution for the Single EC-FTGS instance $\hat{G}$. Consider a superedge (A, B). The only way to have two edge-disjoint paths from a single vertex in g_{AB} to r is for some v_e, $e = (a, b)$ to have two edge-disjoint paths to r: $v_e \mapsto a \mapsto r$ and $v_e \mapsto b \mapsto r$. This incurs a cost 1 for the edge ar and a cost of 1 for the edge br. From the construction, the collection of such end-points $\{a, b\}$ together induces a Min-Rep-cover of cost at most the cost of the Single EC-FTGS solution.

Conversely, given a Min-Rep solution, every $e = (a, b)$ covering the superedge (A, B) gives in an obvious way two disjoint paths to r from v_e, adding 2 to the cost of the Single EC-FTGS solution. Thus cost of the solution for Single EC-FTGS is at most the cost (the number of of vertices of $V_1 \cup V_2$ selected) in the solution for the Min-Rep instance [17]. This completes the proof of the hardness of approximation part of Part (iii) in Theorem 1.1.

Remark: We note that a similar hardness results hold for the vertex-connectivity version or vertex-costs.

References

[1] S. Antonakopoulos, C. Chekuri, B. Shepherd, and L. Zhang. Buy-at-bulk network design with protection. In *FOCS*, pages 634–644, 2007.

[2] Y. Bartal. On approximating arbitrary metrices by tree metrics. In *STOC*, pages 161–168, 1998.

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

[4] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed steiner problems. *J. of Algorithms*, 33(1):73–91, 1999.

[5] C. Chekuri and N. Korula. Pruning 2-connected graphs. In *FSTTCS*, 2008.

[6] C. Chekuri and N Korula. Single-sink network design with vertex connectivity requirements. In *FSTTCS*, 2008.

[7] F. Chudak, T. Roughgarden, and D. Williamson. Approximate k-MSTs and k-Steiner Trees via the Primal-Dual Method and Lagrangean Relaxation. In *IPCO*, pages 60–70, 2001.

[8] J. Chuzhoy and S. Khanna. An $O(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. In *FOCS*. To appear.

[9] J. Chuzhoy and S. Khanna. Algorithms for single-source vertex-connectivity. In *FOCS*, pages 105–114, 2008.

[10] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[11] N. Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *STOC*, pages 396–402, 2005.

[12] N. Garg, G. Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000.

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

[14] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *STOC*, pages 585–594, 2003.

[15] K. Jain. A factor 2 approximation algorithm for the generalized Steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[16] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. of Algorithms*, 19(1):104–115, 1995.

[17] G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001.

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

[19] G. Kortsarz and D. Peleg. Approximating the weight of shallow steiner trees. *Discrete Applied Mathematics*, 93(2-3):265–285, 1999.

[20] Y. Lando and Z. Nutov. Inapproximability of survivable networks. *Theor. Comput. Sci.*, 410:2122–2125, 2009.

[21] L. C. Lau, J. Naor, M. R. Salavatipour, and M. Singh. Survivable network design with degree or order constraints. In *STOC*, pages 651–660, 2007.

[22] Z. Nutov. Approximating connectivity augmentation problems. In *SODA*, pages 176–185, 2005. To appear in *Transactions of Algorithms*.

[23] Z. Nutov. Approximating Steiner Networks with Node Weights. In *LATIN*, pages 411–422, 2008.

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

[25] Z. Nutov. Approximating minimum cost connectivity problems via uncrossable bifamilies and spider-cover decompositions. In *FOCS*, 2009. To appear.

[26] R. Raz. A parallel repitition theorem. *SIAM J. on Computing*, 27(3):763–803, 1998.

[27] G. Reich and P. Widmayer. Beyond steiner's problem: a VLSI oriented generalization. In *WG*, pages 196–210, 1989.

[28] G. Robins and A. Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 1(19):122–134, 2005.

[29] A. Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997.