Improved Approximating Algorithms for Directed Steiner Forest

Moran Feldman* Guy Kortsarz † Zeev Nutov ‡

December 23, 2007

Abstract

We consider the k-Directed Steiner Forest (k-DSF) problem: given a directed graph $G = (V, E)$ with edge costs, a collection $D \subseteq V \times V$ of ordered node pairs, and an integer $k \leq |D|$, find a minimum cost subgraph H of G that contains an st-path for (at least) k pairs $(s, t) \in D$. When $k = |D|$, we get the Directed Steiner Forest (DSF) problem. The best known approximation ratios for these problems are: $\tilde{O}(k^{2/3})$ for k-DSF by Charikar et al. [3], and $O(k^{1/2+\varepsilon})$ for DSF by Chekuri et al. [4]. We improve these approximation ratios as follows.

For DSF we give an $\tilde{O}(n^{4/5+\varepsilon})$-approximation scheme using a novel LP-relaxation that seeks to connect pairs with "cheap" paths. This is the first sub-linear (in terms of $n = |V|$) approximation ratio for the problem; all previous algorithm had ratio $\Omega(n^{1+\varepsilon})$.

For k-DSF we give a simple greedy $O(k^{1/2+\varepsilon})$-approximation algorithm. This improves the best known ratio $\tilde{O}(k^{2/3})$ by Charikar et al. [3], and (almost) matches in terms of k the best ratio known for the undirected variant [2]. Even when used for the particular case of DSF, our algorithm favorably compares to the one of [4], which repeatedly solves linear programs and uses complex space and time consuming transformations. Our algorithm is much simpler and faster, since it essentially reduces k-DSF to a variant of the Directed Steiner Tree problem. The simplification is due to a new notion of "junction star-tree" – a union of an in-star and an out-branching having the same root, which is of independent interest.

1 Introduction

Network design problems seek to find a minimum cost subgraph of a given (directed or undirected) graph, that satisfies prescribed requirements. These problems are among the most studied problems in the fields of Combinatorial Optimization and Approximation Algorithms. We hereby list some classic network design problems on *undirected* graphs. One of the most basic network design problems is the Steiner Tree problem: given a graph $G = (V, E)$ with edge costs, and a set $T \subseteq V$ of terminals, find a min-cost subtree of G that spans U. Unlike the Minimum Spanning Tree problem (the case $U = V$), the Steiner Tree problem is NP-complete (cf. [13]). This classic problem was extensively studied with respect to approximation (see [34] and the references therein). A classic generalization is the Steiner Forest problem: given a graph $G = (V, E)$ with edge costs and a collection $D \subseteq V \times V$ of

*The Open University of Israel. feldsh@netvision.net.il

†IBM research center Yorktown heights. At sabbatical from Rutgers University. guyk@camden.rutgers.edu

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

(unordered) node pairs, find a minimum cost subgraph H of G that connects all pairs in D (namely, contains an st-path for every $\{s,t\} \in D$). The best approximation ratio known for this problem is 2 [1] (see also [16] for a more general algorithm and a simpler proof). In the more general k-Steiner Forest problem, we are also given an integer $k \leq |D|$, and the goal is to connect at least k (arbitrary) pairs from D. Here a significant obstacle lies in the way of achieving a good (say, even polylogarithmic) approximation ratio even for undirected graphs. It was observed in [17] that this problem is harder than the minimization variant of the Densest k-Subgraph problem which is commonly believed not to admit a polylogarithmic approximation. See [11] for an $O(n^{1/3-\varepsilon})$ approximation ratio for the maximization variant, with $\varepsilon \approx 1/60$. Despite several attempts, this ratio was not improved for 11 years. The best known approximation ratio for the k-Steiner Forest problem is $O(\min\{\sqrt{n}, \sqrt{k}\})$, see a recent paper by Gupta et al. [2]. In [2] the k-Steiner Forest problem is shown to have further significance due to its relation to the Dial a Ride problem.

In this paper we consider the *directed* variant of the k-Steiner Forest problem, namely:

k-Directed Steiner Forest (k-DSF)
Instance: A directed graph $G = (V, E)$, costs $\{c(e) : e \in E\}$, a set $D \subseteq V \times V$ of ordered pairs, and an integer $k \leq |D|$.
Objective: Find a min-cost subgraph H of G that contains an st-path for (at least) k pairs $(s,t) \in D$.

When $k = |D|$ we get the **Directed Steiner Forest** (DSF) problem. Another particular case of k-DSF is the ***k*-Directed Steiner Tree** (k-DST) problem, where $D = \{s\} \times T$ for some $s \in V$ and a terminal set $T \subseteq V - \{s\}$. We note that the name "Directed Steiner Forest" is a bit misleading and used just to relate the problem to the undirected version. Indeed, in the undirected version, any minimal feasible solution is a forest. However, in the directed case, the structure of a solution may be complicated (e.g., it may contain cycles). For example, if all costs are 1 and $D = V \times V$, then a directed Hamiltonian cycle is the best solution one can expect.

1.1 Directed and undirected Steiner forests and related problems

Usually, directed variants of network design problems are much harder to approximate that the undirected ones (we shall later see that for k-DSF this is not the case!). For example, while the undirected Steiner Tree and the k-Steiner Tree problems admit a constant approximation ratio (see [34, 8]), even a very special case of DST – the Group Steiner Tree problem on trees, is unlikely to admit a $\log^{2-\varepsilon} n$ ratio for any $\varepsilon > 0$ [18]. In fact, the best known ratio for DST is much worse than its proved lower bound. Extending and simplifying the recursive greedy method introduced by Zelikovsky [36] and Kortsarz and Peleg [28], Charikar et al. [3] gave a combinatorial $O(\ell^3 k^{2/\ell})$-approximation algorithm for k-DST that runs in $O(k^{2\ell} n^{\ell})$ time (where $k = |T|$). Substituting $\ell = 2/\varepsilon$ gives an $O(k^{\varepsilon})$-approximation scheme, namely, an $O(k^{\varepsilon}/\varepsilon^3)$-approximation algorithm that runs in $O(k^{4/\varepsilon} n^{2/\varepsilon})$ time for any fixed $\varepsilon > 0$. Substituting $\ell = \log k$ gives an $O(\log^3 k)$-approximation in quasi-polynomial time.

For the Steiner Forest problem, the comparison between directed and undirected graphs becomes even worse. The problem admits a constant approximation for undirected graphs [1, 16]. However, for DSF strong lower bounds are known [9]. Dodis and Khanna [9] showed that DSF is at least as hard as the LABEL-COVER$_{\max}$ problem [33]. This implies that DSF cannot be approximated within $O(2^{\log^{1-\varepsilon} n})$ for any fixed $\varepsilon > 0$, unless NP-hard problems can be solved in quasi-polynomial time [33].

The situation for DSF (and thus also for the more general k-DSF) is much worse than the above in the current state of the art. The best known ratio for LABEL-COVER$_{\max}$ is $O(\sqrt{n})$ [32]. This ratio seems hard to improve and better ratio algorithms for LABEL-COVER$_{\max}$ are not known even for very simple versions of the problem (e.g., when the structure of the graph obeys the rules of the Unique Game Conjecture [22], when the admissible pairs of answers of the two provers on any fixed query induces a matching). If LABEL-COVER$_{\max}$ is indeed $\Omega(\sqrt{n})$ hard to approximate, then so is DSF.

Still, there is no evidence yet discarding the possibility that DSF admits an $O(\sqrt{n})$ approximation ratio. Indeed, as already mentioned, we provide the first step in this direction by giving a sublinear approximation scheme, in terms of n.

Perhaps the most extreme example of the difference between undirected and directed network design problems is the Steiner Network problem. In this problem each pair (s,t) has connectivity requirement $r(s,t)$, namely, $r(s,t)$ edge disjoint st-paths are required for every $(s,t) \in D$. On undirected graphs, this problem admits a 2-approximation algorithm due to Jain [21]. As far as we know no non-trivial ratio is known for the Steiner Network problem on directed graphs.

Problem	**Undirected**		**Directed**	
	In terms of n	*In terms of k*	*In terms of n*	*In terms of k*
Steiner Tree	1.55 [34]	1.55 [34]	$O(n^{\varepsilon})$ [3]	$O(k^{\varepsilon})$ [3]
k-Steiner Tree	2 [14]	2 [14]	$O(n^{\varepsilon})$ [3]	$O(k^{\varepsilon})$ [3]
Steiner Forest	2 [1]	2 [1]	$O(n^{1+\varepsilon})$	$O(k^{1/2+\varepsilon})$ [4]
k-Steiner Forest	$O(\sqrt{n})$ [2]	$O(\sqrt{k})$ [2]	$\tilde{O}(n^{4/3})$ [3]	$\tilde{O}(k^{2/3})$ [3]
Steiner Network	2 [21]	2 [21]	n^2	k

Table 1: Currently best known approximation ratios for Steiner Network problems, prior to our work. We improve the ratios in terms of n for Steiner Forest and k-Steiner Forest on directed graphs.

For other related problems, including the closely related Group Steiner Tree problem see [5, 15, 24, 18, 20, 12, 26] and surveys in [10] and [23, 27].

1.2 Our results

For DSF, the best known approximation ratio, in terms of n, is $O(n^{1+\varepsilon})$, which can be easily derived from the algorithm of [3] for DST. For k-DSF, Charikar et al. [3] gave an $\tilde{O}(k^{2/3})$-approximation algorithm, thus, the best ratio for k-DSF in terms of n was $\tilde{O}(n^{4/3})$, since $k = O(n^2)$.

A natural question is whether DSF admits an $O(n^{1-\varepsilon})$ approximation ratio. In particular, is there an $O(\sqrt{n})$-approximation algorithm? Our first result makes a progress toward answering this question, by giving the first sublinear, in terms of n, approximation algorithm for the problem.

Theorem 1.1 DSF *admits an $O(n^{4/5+\varepsilon})$-approximation scheme.*

The algorithm of [4] for DSF does not extend to k-DSF; see the reasons for that in Section 1.3.1. Thus another natural question is: What is the best ratio possible for k-DSF in terms of k? We prove:

Theorem 1.2 k-DSF *admits an $O(k^{1/2+\varepsilon})$-approximation scheme.*

This improves the $\tilde{O}(k^{2/3})$ ratio of [3]. It almost matches the best approximation $O(\sqrt{k})$ known (in terms of k) for undirected graphs [2]. A striking feature of the state of the art of the k-Steiner Forest problem is that the ratios known for the directed and undirected cases are not that different in terms of k: $O(\sqrt{k})$ for undirected graphs [2] versus $O(k^{1/2+\varepsilon})$ in our paper. However, in terms of n, the difference $\sqrt{n}$ versus $n^{4/5+\varepsilon}$ is still quite large.

A *setpair* is a pair (S,T) of disjoint nonempty subsets of V. [4] gives an $O(\log^2 n \log^2 |\mathcal{D}|)$-approximation algorithm for the following generalization of the Group Steiner Tree problem:

Group Steiner Forest (GSF)
Instance: An (undirected) graph $G=(V,E)$, costs $\{c(e) : e \in E\}$, and a set $\mathcal{D}$ of setpairs in V.
Objective: Find a min-cost subgraph H of G that contains an (S,T)-path for every setpair $(S,T) \in \mathcal{D}$.

In the more general k-Group Steiner Forest (k-GSF) problem, we are also given an integer $k \leq |\mathcal{D}|$, and only require H to contain an (S,T)-path for (at least) k setpairs $(S,T) \in \mathcal{D}$. Note that k-GSF also generalizes the (undirected) k-Steiner Forest problem, which is the particular case when every setpair in $\mathcal{D}$ is just a pair of nodes. The polylogarithmic approximation of [4] does not extend to k-GSF. Furthermore, k-GSF is unlikely to admit a polylogarithmic approximation ratio, otherwise, we would obtain a polylogarithmic approximation for (undirected) k-Steiner Forest. Recall that the best known ratio for the latter is $O(\min\{\sqrt{n}, \sqrt{k}\})$, and that a polylogarithmic ratio for it implies a polylogarithmic ratio for the Densest k-Subgraph problem [2].

k-GSF admits an easy (up to constants) approximation ratio preserving reduction to k-DSF. Thus by Theorem 1.2 we obtain the following extension of the $O(\sqrt{k})$-approximation for the (undirected) k-Steiner Forest problem:

Corollary 1.3 *k-GSF admits an $O(k^{1/2+\varepsilon})$-approximation scheme.*

In fact, our results can be used to show that if the k-Group Steiner Tree problem admits an α-approximation algorithm, then k-GSF admits an $O(\alpha\sqrt{k})$-approximation algorithm, implying an $\tilde{O}(\sqrt{k})$-approximation. We also note that in terms of n, k-GSF admits an $\tilde{O}(n^{2/3})$-approximation [35].

The running time of our algorithm for k-DSF: Suppose that k-DST admits an α-approximation in $T(n,k)$ time. We show that then k-DSF admits an $O(\alpha\sqrt{k})$-approximation in $O(nk^2T(2n+k,k))$ time (assuming $T(n,k)$ is increasing in k). In particular, k-DSF admits an $O(k^{1/2+\varepsilon}/\varepsilon^3)$-approximation algorithm that runs in $O(nk^{2+4/\varepsilon}(2n+k)^{2/\varepsilon})$ time for any fixed $\varepsilon > 0$. Using the specific properties of the k-DST algorithm of [3], the time complexity can be reduced to $O(nk^{1+4/\varepsilon}(2n+k)^{2/\varepsilon})$.

Comparing our algorithm and the algorithm of [4] on DSF: In addition to the above improvements, our $O(k^{1/2+\varepsilon})$-approximation algorithm for k-DSF, when restricted to the special case of DSF, achieves the same ratio as [4] but is much simpler and much faster. The algorithm of [4] repeatedly solves linear programs while our algorithm can be seen as a reduction to k'-DST and is purely combinatorial; this is the reason our running time is much lower. We achieve this by introducing a new notion of *junction star-trees* that simplifies matters.

1.3 Main new techniques: junction star-trees and a novel LP for DSF

1.3.1 Junction star-trees and their advantages

All known algorithms for k-DST accumulate *good density* trees until enough pairs are connected. The density of a solution is its cost over the number of new pairs it connects.

The idea of low-density *junction trees* was first invented for approximating Buy-at-Bulk problems [6]. For the sake of Buy-at-Bulk, it was enough to define a junction tree as a collection of paths, all going via the same node, and sending a (possibly fractional) unit of s_it_i-flow for "many" s_i, t_i pairs at a "low" cost. The reason this definition suffices is that there are known methods to round such fractional solutions into trees of low cost with only polylogarithmic loss compared to the fractional value (see [30, 7]).

For problems on *directed* graphs, it does not seem that we can easily use fractional flow methods to achieve a low ratio approximation algorithm. Even for DST, it is not clear if it is possible to round a fractional flow solution into a low cost out-branching. The only tool we have for tasks of finding low cost out- and in-branchings is the recursive greedy algorithm of [3].

Hence, in the directed setting a more careful definition of junction tree is used [4]. A junction tree is an in-branching T_I into r plus an out-branching T_O out of r connecting several pairs from D [4]. In the definition of [4], T_I and T_O are allowed to intersect. [1]

We present a few details of the [4] algorithm. Setting the goal of achieving a roughly $\sqrt{k}$-approximation, a simple averaging argument shows that there is a node r so that at least $\sqrt{k}$ s_it_i-paths go via it (otherwise, a $\sqrt{k}$ ratio is easily derived [4]) giving a low density junction tree. [2]

The question is how to find such a low density junction tree. This is a non-trivial challenge. The goal is to "match" the "source" s_i with the proper terminal t_i. A "naive" application of [3] is guessing the root r and the number k' of sources in T_I (which equals the number of terminals in T_O), and finding an in-branching with k' sources and an out-branching with k' terminals using [3]. This method fails because the sources in the in-branching and the terminals in the out-branching found may not match.

In [4], some clever manipulations are performed to overcome that difficulty. This includes a phase called *paths splitting* and a use of a "density type" *linear program* that in effect "forces" the sources to match the terminals. See a simpler application of a density type LP in [6]. The situation in [6] is simpler then here as the graph in [6] is undirected.

We overcome some of the above difficulties using junction star-trees. A junction star-tree is a star with leaves s_i entering a root r joined to an out-branching covering the respective t_i. If we can find a junction star-tree of low density, then the difficulties of matching s_i and t_i no longer apply. There is an easy way to force the s_i, t_i to match by "attaching" each s_i as a child of t_i with a directed edge t_is_i of cost $c(s_ir)$. Thus, the s_i become the terminals, and s_i belongs to the solution if and only if t_i does (see a more formal proof of this in Section 4). We show that the metric completion of the input

[1] It seems more natural to require that $T_I \cap T_O = \emptyset$. However, this makes no difference. It is easy to handle the extra requirement $T_I \cap T_0 = \emptyset$ by taking shortest s_it_i-paths, after joining via the junction node two copies of the input graph.

[2] Technically speaking, a union of s_it_i-paths all going via r is not a junction tree as defined in [4]. However, is easy to see that it contains a junction tree.

graph G always contains a junction star-tree of good density. Hence the problem is reduced to k'-DST problem (we still need to guess the root r and the number k' of pairs in the junction star-tree), and we do not need to use LP methods. Obviously, a drawback is that it is harder to prove the existence of a low density junction star-tree (see the Junction Star-Tree Theorem 4.5 and its proof in Section 4.1), than just the existence of a low density junction tree.

Another disadvantage of the LP method used by [4] is that it is unable to deal with the k-DSF problem. The LP may connect an arbitrary number of pairs (albeit, it returns a solution of good density). Hence, the number of pairs the LP connects may be much larger than k. The use of junction star-trees allows us to use the algorithm of [3] for k-DST (by solving the k'-DST problem, $k' \leq k$) instead of the LP methods, which in turn allows us to control the number of pairs connected.

1.3.2 A novel LP for cheap paths in the sublinear algorithm for DSF

Intuitively, a pair $st \in D$ is "good" if there are "many" nodes r so that a "cheap" st-path via r exists; otherwise, the pair is "bad". There are three main procedures in our sublinear algorithm for DSF:

1. The first procedure uses randomization to find a relatively small "junction subset" $R \subset V$ through which all good pairs can be connected. As R is small, and the paths are cheap, we can show that the cost incurred in connecting all good pairs via R is $\tilde{O}(n^{4/5}) \cdot \mathsf{opt}$. After all good pairs are connected, they are excluded from D, and we remain with bad pairs only.

2. If in some optimal solution at least half of bad pairs are connected by "long path" then we prove, by standard averaging, the existence of a low density junction-tree (as defined in [4]). A sub-graph with density close to the density of such a tree is found using the procedure of [4].

3. The difficult case is when optimal solution connect most of the bad pairs by "cheap" paths. To handle this case, we formulate a novel LP-relaxation which asks to connect pairs by cheap paths only. This LP assigns capacity x_e to every edge e so that it will be possible to send a unit of $s_i t_i$-flow (separately for every i) along cheap paths, and so that $\sum_{e \in E} c(e) x_e$ is minimized. We show how to find approximate solutions for this LP in polynomial time, and that rounding up entries x_e of large enough value gives a low density augmentation.

2 Preliminaries

2.1 The ρ-Greedy Algorithm

We use a known result about the performance of a *Greedy Algorithm* for the following type of problems:

Covering Problem
Instance: A groundset E and non-negative functions ν, c on 2^E, given by an evaluation oracle.
Objective: Find $F \subseteq E$ with $\nu(F) = 0$ and $c(F)$ minimized.

We call ν the *deficiency function* (it measures how far is F from being a feasible solution) and c the *cost function*.

Definition 2.1 *Let $F \subseteq E$ be a partial solution (partial cover) for an instance of* Covering Problem *and let $J \subseteq E$. Let $\rho(x)$ be a positive function, and let* opt *be the optimal solution value for* Covering Problem. *We say that $J \subseteq E$ obeys the $\rho(x)$-*Density Condition *if:*

$$\sigma_F(J) = \frac{c(J)}{\nu(F) - \nu(F \cup J)} \leq \mathsf{opt} \cdot \frac{\rho(\nu(F))}{\nu(F)} \tag{1}$$

The quantity $\sigma_F(J)$ in (1) is the *density of* J (w.r.t. F). The $\rho(x)$-*Greedy Algorithm* starts with $F = \emptyset$ and iteratively adds to F a subset $J \subseteq E$ obeying (1). A set-function f on 2^E is *decreasing* if $f(F_2) \leq f(F_1)$ for any $F_1 \subseteq F_2 \subseteq E$, and *subadditive* if $f(F_1 \cup F_2) \leq f(F_1) + f(F_2)$ for any $F_1, F_2 \subseteq E$. The following statement is well known (e.g., see a slightly weaker version in [3]).

Theorem 2.1 *If ν is decreasing, c is subadditive, and $\rho(x)/x$ is a decreasing function, then the $\rho(x)$-Greedy Algorithm computes a solution F with:*

$$c(F) \leq \mathsf{opt} \cdot \int_0^{\nu(\emptyset)} \frac{\rho(x)}{x} dx \ . \tag{2}$$

In our setting, the ground-set is the set E of edges of the graph. For every partial solution $F \subseteq E$, the deficiency $\nu(F)$ of F is the number of ordered pairs not connected by F. Formally, $\nu(F) = \max\{k - |D(F)|, 0\}$, where $D(F)$ denotes the set of pairs from D connected by F. Clearly, ν is decreasing, and c is subadditive.

2.2 Some simple reductions

We briefly describe some well known reductions to be used later that we can apply with negligible loss (in time complexity or approximation ratio) on a given k-DSF instance.

Reduction 1 *We may assume that $S \cap T = \emptyset$ and that no edge enters S or leaves T.*
This can be achieved by adding for every node v two new nodes s_v, t_v with edges $s_v v, v t_v$ of cost 0 each, and replacing every ordered pair $(u, v) \in D$ by the pair (u_s, v_t).

Reduction 2 *We may assume that G is transitively closed and that the costs are metric.*
This is achieved by applying metric completion.

Given an instance of k-DSF, Reductions 1 and 2 can be implemented in $O(n^3)$ time, hence negligible in our context, and result in an instance with $3n = O(n)$ nodes.

Reduction 3 *We may assume that we know τ such that* $\mathsf{opt} \leq \tau \leq 2 \cdot \mathsf{opt}$.
This can be done by checking all values $\tau \in \{1, 2, 4, \ldots, 2^{\lceil \log_2 c(E) \rceil}\}$; we omit the (well known) details, and for simplicity of exposition assume that $\tau = \mathsf{opt}$.

3 A sublinear algorithm for DSF (Proof of Theorem 1.1)

Given an instance of DSF assume that all reductions from Subsection 2.2 are implemented. In what follows, let $p,\ \alpha,\ \ell$ be parameters eventually set to:

$$p = 2 \ln k / n^{2/5}, \quad \alpha = n^{2/5}, \quad \ell = \tau / \alpha^2 \ .$$

For a graph H, let $\mathsf{dist}_H(u,v)$ denote the minimum cost of a uv-path in H.

Definition 3.1 *A path P is* short *if $c(P) \leq \ell$, and* long *otherwise. For $(s,t) \in D$, let $U(s,t) = \{u \in V : \mathsf{dist}_G(s,u), \mathsf{dist}_G(u,t) \leq \ell\}$. A pair $(s,t) \in D$ is* good *if $|U(s,t)| \geq \alpha$, and is* bad *otherwise.*

3.1 Connecting good pairs

Lemma 3.1 *There exists a polynomial time algorithm that given an instance of* DSF *finds an edge set F of cost $c(F) \leq 4pn^2\ell = \tilde{O}(n^{4/5}) \cdot \tau$ that connects all good pairs.*

Proof: Form a set $R \subseteq V$ by picking every node $v \in V$ into R with probability p. For a given good pair (s,t) we have:

$$\Pr[R \cap U(s,t) = \emptyset] \leq (1-p)^{\alpha} \leq \frac{1}{k^2}$$

By the union bound, the probability that $R \cap U(s,t) \neq \emptyset$ for every good pair (s,t) is at least $1-k$. $|R|$ is a random variable with binomial distribution $B(n,p)$. Thus $E(|R|) = pn$. Using the Chernoff Bound we get:

$$\Pr[|R| \leq 2pn] = \Pr[|R| \leq 2 \cdot E(|R|)] > 1 - e^{-pn/4} \ .$$

For $pn/4 \geq \ln k$ we get that with high probability both $|R| \leq 2pn$ and $R \cap U(s,t) \neq \emptyset$ for every good pair (s,t). This procedure can be derandomized using the method of conditional probabilities. We connect by a short path every node $s \in S$ to every node $v \in R$, if such path exists. Similarly, we connect by a short path every node $v \in R$ to every node $t \in T$, if such path exists. Let H be the sub-graph constructed by the above procedure. Clearly H connects all good pairs. As $|S| + |T| \leq 2n$, we get that $c(H) \leq |R| \cdot 2n \cdot \ell \leq 2pn \cdot 2n \cdot \ell = 4pn^2\tau/\alpha^2 = \tau \cdot \tilde{O}\left(n^{4/5}\right)$. □

3.2 Connecting bad pairs

After all good pairs are connected using the algorithm of Lemma 3.1, they are excluded from D, and we remain with bad pairs only.

Lemma 3.2 *There exists an algorithm that given a* DSF *instance without good pairs and a constant $\varepsilon > 0$, computes in polynomial time an edge set $J \subseteq E$ of density $\tilde{O}(n^{4/5+\varepsilon}) \cdot \tau/|D|$.*

In the rest of this section we prove Lemma 3.2. We compute two edge sets using two different algorithms, and choose among them the one with lower density. For the analysis, let us fix some optimal solution H, so $c(H) = \mathsf{opt} = \tau$. Let $L = \{(s,t) \in D : \mathsf{dist}_H(s,t) \geq \ell\}$. We will consider two cases: $|L| \geq |D|/2$ and $|D - L| > |D|/2$.

Definition 3.2 ([4]) *An edge set J in a directed graph is called* junction tree *if it is the union of an ingoing tree and an outgoing tree (not necessarily edge disjoint), both rooted at the same node r.*

Lemma 3.3 ([4]) *The problem of finding a minimum density junction tree admits an $O(k^{\varepsilon})$-approximation scheme.*

Proposition 3.4 *H contains a junction tree J of density at most*

$$\sigma(J) \leq \frac{\tau}{\ell} \cdot \frac{\tau}{|L|} = n^{4/5} \cdot \frac{\tau}{|L|}$$

Hence if $|L| \geq |D|/2$, the algorithm of [4] finds a junction tree J of density $O(n^{4/5+\varepsilon}) \cdot \tau/|D|$.

Proof: Let $\Pi(L)$ be a set of paths in H corresponding to the pairs in L. The sum of the costs of the paths in $\Pi(L)$ is at least $|L| \cdot \ell$. Since the paths of $\Pi(L)$ are in H, there must be an edge of H which belongs to at least $|L| \cdot \ell/\tau$ paths. This implies that there is a junction tree in H connecting at least $|L| \cdot \ell/\tau$ pairs from $\Pi(L)$. The density of such tree is at-most $c(H)/(|L| \cdot (\ell/\tau)) \leq (\tau/|L|) \cdot (\tau/\ell)$, as claimed. □

Now suppose that $|L| < |D|/2$, so $|D - L| > |D|/2$. Consider the following LP-relaxation (LP1) for the problem of connecting at least $k' \leq |D - L|$ pairs from $D = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. Intuitively, (LP1) decides on a capacity x_e for every $e \in E$ and an amount y_i of $s_i t_i$-flow. The sum of the y_i's is at least k'. The main restriction is that the flow has to be delivered on (simple) paths of cost $\leq \ell$. This is done as follows. Let $\Pi(i)$ be the set of (simple) $s_i t_i$-paths in G of cost $\leq \ell$, and let $\Pi = \bigcup_i \Pi(i)$. For every i, decompose the final $s_i t_i$-flow in the graph into flow paths. For every $P \in \Pi(i)$, the variable f_P is the amount of $s_i t_i$-flow through P. The total $s_i t_i$-flow equals the sum of the flows on the paths in $\Pi(i)$, namely, $y_i = \sum_{P \in \Pi(i)} f_P$. For every i and $e \in E$, the capacity constraint is $\sum_{\Pi(i) \ni P \ni e} f_P \leq x_e$; namely, the total $s_i t_i$-flow through e is at most x_e. Note that it holds for every pair separately, namely, it may not be possible to deliver simultaneously flows y_i and $y_{i'}$, $i \neq i'$.

$$
\begin{array}{lllll}
\text{(LP1)} & \min & \sum_{e \in E} c(e) x_e & & \\
& \text{s.t.} & \sum_i y_i & \geq & k' \\
& & \sum_{\Pi(i) \ni P \ni e} f_P & \leq & x_e \qquad \forall\ i,\ e \in E \\
& & \sum_{P \in \Pi(i)} f_P & = & y_i \qquad \forall\ i \\
& & y_i,\ x_e & \leq & 1 \qquad \forall\ i,\ e \in E \\
& & y_i,\ f_P,\ x_e & \geq & 0 \qquad \forall\ i,\ P \in \Pi,\ e \in E
\end{array}
$$

The corresponding dual LP is:

$$
\begin{array}{lllll}
\text{(LP2)} & \max & \sum_{e \in E} x_e + \sum_i y_i - W \cdot k' & & \\
& \text{s.t.} & \sum_i z_{i,e} + c(e) & \leq & x_e \qquad \forall\ e \in E \\
& & y_i + w_i & \geq & W \qquad \forall\ i \\
& & w_i & \leq & \sum_{e \in P} z_{i,e} \qquad \forall\ i,\ P \in \Pi(i) \\
& & W,\ x_e,\ y_i,\ z_{i,e} & \geq & 0 \qquad \forall\ i,\ e \in E
\end{array}
$$

Lemma 3.5 *For any $k' \leq |D - L|$ the optimal value of (LP1) is at most* opt*. Furthermore, a solution for (LP1) of value $\leq (1 + \varepsilon) \cdot$ opt can be found in polynomial time.*

Proof: The first statement is obvious, as (LP1) is a relaxation for the problem. We show how to find an approximate solution in polynomial time. Although the number of variables in (LP1) might be exponential, any basic feasible solution to it has $O(|D| \cdot m)$ non-zero variables. Now, if we had a polynomial time separation oracle for (LP2), we could compute an optimal solution to (LP1) (the non-zero entries) in polynomial time. The number of non-zero entries in such a computed solution is polynomial in $O(|D| \cdot m)$. The only problematic constraints are $w_i \leq \sum_{e \in P} z_{i,e}$, unfortunately, for these constraints, a polynomial time separation oracle may not exist, since the separation problem defined by a specific pair (s_i, t_i) is equivalent to the following problem, which is NP-hard [29]:

Restricted Shortest Path (RSP)

Instance: A directed graph $G = (V, E)$, transition times $\{z(e) : e \in E\}$, lengths $\{\ell(e) : e \in E\}$, a pair (s, t), and an integer Z.

Objective: Find a minimum length st-path P such that $\sum_{e \in P} z(e) \leq Z$.

RSP admits a FPTAS (see [19] and an improved result in [29]), therefore we get an approximate separation oracle, which for any $\varepsilon > 0$ checks whether there exists a path $P \in \Pi(i)$ so that $w_i \leq \sum_{e \in P} z_{i,e}/(1+\varepsilon)$. This implies that we can solve the following linear program in time polynomial in $1/\varepsilon$ and the size of the original DSF problem:

$$\begin{array}{lrcll} \text{(LP3)} & \max & \sum_{e \in E} x_e + \sum_i y_i - W \cdot k' & & \\ & \text{s.t.} & \sum_i z_{i,e} + c(e) & \leq & x_e & \forall\ e \in E \\ & & y_i + w_i & \geq & W & \forall\ i \\ & & w_i & \leq & \sum_{e \in P} z_{i,e}/(1+\varepsilon) & \forall\ i,\ P \in \Pi(i) \\ & & W,\ x_e,\ y_i,\ z_{i,e} & \geq & 0 & \forall\ i,\ e \in E \end{array}$$

Thus we can also solve the dual of (LP3), which is:

$$\begin{array}{lrcll} \text{(LP4)} & \min & \sum_{e \in E} c(e) x_e & & \\ & \text{s.t.} & \sum_i y_i & \geq & k' & \\ & & \sum_{\Pi(i) \ni P \ni e} f_P & \leq & x_e \cdot (1+\varepsilon) & \forall\ i,\ e \in E \\ & & \sum_{P \in \Pi(i)} f_P & = & y_i & \forall\ i \\ & & y_i,\ x_e & \leq & 1 & \forall\ i,\ e \in E \\ & & y_i,\ f_P,\ x_e & \geq & 0 & \forall\ i,\ P \in \Pi,\ e \in E \end{array}$$

Let $\mathsf{opt}(\varepsilon)$ denote the optimal value of (LP4). Clearly, $\mathsf{opt}(\varepsilon) \leq \mathsf{opt}$. Note that if $x(\varepsilon)$ is a feasible solution to (LP4) then by replacing the value of every variable x_e in $x(\varepsilon)$ by $\min\{1, x_e \cdot (1+\varepsilon)\}$ we get a new solution x which is a feasible solution to (LP1). The value of such x is at most $(1+\varepsilon) \cdot \mathsf{opt}(\varepsilon) \leq (1+\varepsilon) \cdot \mathsf{opt}$. □

Lemma 3.6 *Let x, y be a feasible solution to (LP1) and let $0 \leq \beta < k'/|D|$ arbitrary. Then at most $(|D| - k')/(1-\beta)$ pairs in D have flow $y_i < \beta$. Thus, the number of pairs in D that have flow $y_i \geq \beta$ is at least:*

$$|\{i : y_i \geq \beta\}| \geq |D| - \frac{|D| - k'}{1-\beta} = \frac{k' - \beta|D|}{1-\beta}$$

Proof: If more than $(|D| - k')/(1-\beta)$ pairs in D have flow strictly less than β, then the sum of the flows between all pairs must be strictly less than:

$$\frac{|D| - k'}{1-\beta} \cdot \beta + \left(|D| - \frac{|D| - k'}{1-\beta}\right) \cdot 1 = |D| + (\beta - 1) \cdot \frac{|D| - k'}{1-\beta} = k'$$

This is a contradiction, since in any feasible solution of (LP1), the sum of the flows between all pairs must be at least k'. □

Lemma 3.7 *Let x, y be a feasible solution to (LP1) and let $0 \leq \beta < 1$. If $y_i \geq \beta$ for some i then $J = \{e \in E : x_e \geq 4\beta/\alpha^2\}$ contains an $s_i t_i$-path.*

Proof: We claim that $C \cap J \neq \emptyset$ for every s_it_i-cut C. Suppose to the contrary that $C \cap J = \emptyset$ for some s_it_i-cut C, namely, $x_e < 4\beta/\alpha^2$ for every $e \in C$. Thus $|C| \geq \alpha^2/4$, since $\sum_{e \in C} x_e \geq y_i \geq \beta$. Every edge $e = uv \in C$ that carries a positive amount of s_it_i-flow belongs to some short s_it_i-path, thus $u, v \in U(s_i, t_i)$. We got that $U(s_i, t_i)$ contains end nodes of at least $\alpha^2/4$ edges of the cut C, implying that $U(s_i, t_i)$ contains at least $2\sqrt{\alpha^2/4} = \alpha$ nodes. Thus (s_i, t_i) is a good pair, contradicting our assumption that all the pairs are bad. □

Corollary 3.8 *Assuming $k' \leq |D - L|$, let (x, y) be any solution to (LP1) found using Lemma 3.5. Then for any $0 \leq \beta < k'/|D|$, the edge set $J = \{e \in E : x_e \geq 4\beta/\alpha^2\}$ has density at most:*

$$\frac{\alpha^2 \mathsf{opt} \cdot (1+\varepsilon)}{4\beta} \cdot \frac{(1-\beta)}{k' - \beta|D|}$$

In particular, for $k' = |D|/2 \leq |D - L|$ and $\beta = 1/4$, the density of J is at most $3\alpha^2 \cdot \mathsf{opt} \cdot (1+\varepsilon)/D = O(n^{4/5}) \cdot \mathsf{opt}/|D|$.

Proof: Since $k' \leq |D - L|$, the value of (LP1) is at most $\mathsf{opt} \cdot (1 + \varepsilon)$, by Lemma 3.5. Thus $c(J) \leq \mathsf{opt} \cdot (1+\varepsilon)/(4\beta/\alpha^2) = \mathsf{opt} \cdot (1+\varepsilon)\alpha^2/(4\beta)$. By Lemmas 3.6 and 3.7, $|D(J)| \geq (k' - \beta|D|)/(1-\beta)$. Thus:

$$\sigma(J) = \frac{c(J)}{|D(J)|} \leq \frac{\mathsf{opt} \cdot (1+\varepsilon)\alpha^2/(4\beta)}{(k' - \beta|D|)/(1-\beta)} = \frac{\alpha^2 \mathsf{opt} \cdot (1+\varepsilon)}{4\beta} \cdot \frac{(1-\beta)}{k' - \beta|D|}$$

□

Proof of Lemma 3.2: We execute two algorithms to compute edge sets J', J'' and choose among them the one with the better density. The set J' is computed using the algorithm of Lemma 3.3. The set J'' is computed using the algorithm of Corollary 3.8 with parameters $k' = |D|/2$ and $\beta = 1/4$. If $|L| \geq |D|/2$ then the density of J' is $\tilde{O}(n^{4/5+\varepsilon}) \cdot \tau/|D|$. Otherwise, if $|D - L| \geq |D|/2$, the density of J'' is $O(n^{4/5}) \cdot \tau/|D|$. In any case, one of J', J'' has density $\tilde{O}(n^{4/5+\varepsilon}) \cdot \tau/|D|$. □

3.3 Putting everything together

Implement Reductions 1, and 2, in this order. Assuming that we know τ (Reduction 3), the entire algorithm is as follows:

1. Find an edge set F as in Lemma 3.1, and exclude all good pairs from D.
2. *While* F is not a feasible solution do:
   - Find an edge set J as in Lemma 3.2;
   - $F \leftarrow F + J$;
   - $D \leftarrow D - D(J)$.

   EndWhile
3. *Return* F.

The reductions incur only a constant loss in the approximation ratio. The total cost of the edges added at Step 1 is $\tilde{O}(n^{4/5}) \cdot \tau$, by Lemma 3.1. Step 2 is essentially a ρ-Greedy Algorithm with

$\rho = \tilde{O}(n^{4/5+\varepsilon})$, by Lemma 3.2. Thus by Theorem 2.1, the total cost of the edges added at Step 2 is $\tilde{O}(n^{4/5+\varepsilon}) \cdot \tau = \tilde{O}(n^{4/5+\varepsilon}) \cdot \mathsf{opt}/|D|$. This finishes the proof of Theorem 1.1.

Remark: Note that during the algorithm we do not zero the cost of the edges accumulated by the partial solution at Steps 1,2, but only exclude the pairs already connected from D. This is since zeroing costs of edges might make bad pairs good, while we connect the good pairs only once, at the beginning of the algorithm at Step 1. Although it does not improve the ratio we can prove, it is possible to zero the costs of the edge sets found, by postponing connecting the good pairs to the end of the algorithm. This might improve the ratio in practice, and is achieved as follows. First, connect the bad pairs only; at every iteration zero the costs of the edge sets added into the partial solution, and update accordingly the set of bad pairs. Second, when no bad pairs remain, connect all good pairs by the algorithm of Lemma 3.1.

4 An algorithm for k-DSF (Proof of Theorem 1.2)

This section is organized as follows: Subsection 4.1 defines the notation of "junction star-trees" and proves the "The Junction Star-Tree Theorem" which ensures the existence of a good density junction star-tree in the metric completion of any graph. Subsection 4.2 describes the algorithm for k-DSF.

4.1 Junction star-trees

Definition 4.1 *Let G be a graph with a set $D = \{(s_1,t_1),\ldots,(s_{|D|},t_{|D|})\}$ of ordered pairs; $S = \{s_1,\ldots,s_{|D|}\}$ are* sources *and $T = \{t_1,\ldots,t_{|D|}\}$ are* terminals*. A subgraph J of G is a* junction star-tree *if it is a union of an out-branching J_T rooted at r in $(G-S)\cup\{r\}$ and a star J_S ingoing to r in $(G-T)\cup\{r\}$.*

Roughly speaking, J is a junction star-tree if it can be obtained by taking a tree rooted at r (with no source nodes, except for r) that connects r to a set T' of terminals, and adding for every $s \in S'$ (where S' is a set of non-terminal nodes) the edge sr. Our main interest will be in the case where S' is the set of sources corresponding to the terminals of T'.

The main result of this section is the following statement that is used in the algorithm presented in the next section, and which we believe is of independent interest.

Theorem 4.1 (The Junction Star-Tree Theorem) *Let $H = (V,E)$ be a graph with edge costs $\{c(e) : e \in E\}$ containing a set Π of k paths connecting a set $D \subseteq S \times T$ of k node pairs, so that $S \cap T = \emptyset$ and so that no edge enters S or leaves T. If $c(P) \geq c(H)/g$ for every $P \in \Pi$ then the metric completion of H contains a junction star-tree J of density at most:*

$$\frac{c(J)}{|D(J)|} \leq c(H) \cdot \left(\frac{g}{k} + \frac{2}{g}\right) \tag{3}$$

In the rest of this section we will prove Theorem 4.1. For every st-path $P \in \Pi$, the *truncated path* $\bar{P}$ of P is the maximal sv-subpath of P so that $c(\bar{P}) < c(H)/g$. Let e_P be the edge in $P - \bar{P}$ leaving the last node of $\bar{P}$. Since $c(P) \geq c(H)/g$, then by the definition of $\bar{P}$: e_P always exists, and $c(\bar{P} + e_P) \geq c(H)/g$. Let $\bar{\Pi} = \{\bar{P} : P \in \Pi\}$.

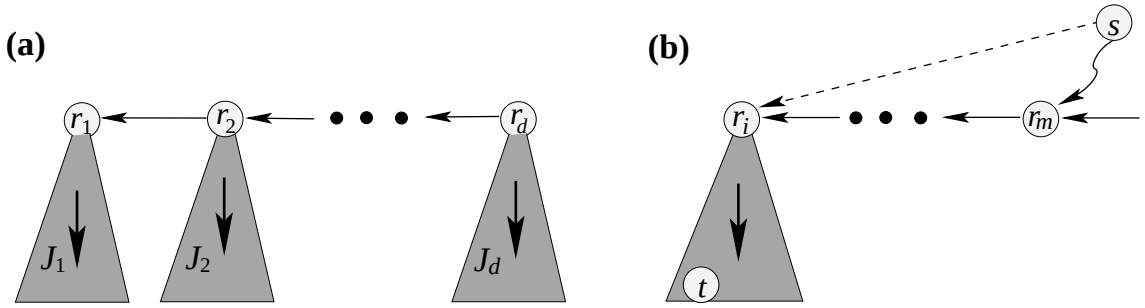

Figure 1: (a) The trees $J_1, \ldots, J_d$ hanged on the path $\bar{P}$; the trees are edge disjoint, but might not be node disjoint; some of the trees might consist of the root only. (b) Illustration of property 3 in Lemma 4.3 and the "shortcut" in the proof of Corollary 4.4.

Definition 4.2 *We say that two (not necessarily different) truncated paths in* $\bar{\Pi}$ collide *if they have a node in common.*

Lemma 4.2 *There exists a partition* $\bar{\mathcal{P}}_1, \ldots, \bar{\mathcal{P}}_q$ *of* $\bar{\Pi}$ *into* $q \leq g$ *parts, and a set of pairwise non-colliding paths* $\{\bar{P}_i \in \bar{\mathcal{P}}_i : i = 1, \ldots, q\}$*, such that* $\bar{P}_i$ *collides with every path in* $\bar{\mathcal{P}}_i$*,* $i = 1, \ldots, q$*. Thus there is a path* $\bar{P} \in \bar{\Pi}$ *colliding with at least* $\ell \geq k/g$ *paths in* $\bar{\Pi}$*.*

Proof: We will construct the partition iteratively. Assuming that at the end of iteration $i-1$ we constructed a subpartition $\{\bar{\mathcal{P}}_1, \ldots, \bar{\mathcal{P}}_{i-1}\}$ of $\bar{\Pi}$, which is not yet a partition of $\bar{\Pi}$, in iteration i perform two steps:

1. Pick a path $\bar{P}_i \in \bar{\Pi}$ which does not belong to any part yet, and place it in a new part $\bar{\mathcal{P}}_i$.
2. Add to $\bar{\mathcal{P}}_i$ every path that collides with $\bar{P}_i$ and does not belong to any other part yet.

By the construction, it is clear that eventually we will get a partition of $\bar{\Pi}$, such that $\bar{P}_i$ collides with every path in $\bar{\mathcal{P}}_i$ for every i, and that $\{\bar{P}_i\}_{i=1}^q$ are pairwise non-colliding. Hence we only need to show that the number q of parts is bounded by g. Let $e_i = e_{P_i}$, $i = 1, \ldots, q$. Note that since $\bar{P}_1, \ldots, \bar{P}_q$ are pairwise node disjoint, the paths $\bar{P}_1 + e_1, \ldots, \bar{P}_q + e_q$ are pairwise edge-disjoint. Thus their total cost is at most $c(H)$. Since $c(\bar{P}_i + e_i) \geq c(H)/g$ for every i, the statement follows. □

Focus on a path $\bar{P} \in \bar{\Pi}$ and a set of $\bar{\mathcal{P}} = \{\bar{P}_1, \ldots, \bar{P}_\ell\}$ of $\ell \geq k/g$ truncated paths colliding with $\bar{P}$ ($\bar{P} \in \bar{\mathcal{P}}$), which existence is guaranteed by Lemma 4.2. Let $\mathcal{P} = \{P_1, \ldots, P_\ell\} \subseteq \Pi$ be the set of corresponding non-truncated paths. Let $\bar{S} = \{s_1, \ldots, s_\ell\}$ and $\bar{T} = \{t_1, \ldots, t_\ell\}$ be the sets of sources and terminals of the paths in $\mathcal{P}$, respectively. Let $r_1, \ldots, r_d$ be the sequence of nodes of $\bar{P}$ arranged in reverse order; r_d is the first node of $\bar{P}$, r_{d-1} is the second, and so on; the last node of $\bar{P}$ is r_1 (see Fig. 1(a)).

Lemma 4.3 *There exists in* H *a family* $J_1, \ldots, J_d$ *of pairwise edge disjoint trees so that (see Fig. 1(b)):*

1. *Every* J_i *is rooted at* r_i*,* $i = 1, \ldots, d$*.*
2. *Every* $t \in \bar{T}$ *belongs exactly one tree* J_i*,* $1 \leq i \leq d$*.*
3. *If* $t \in \bar{T} \cap J_i$ *and* $(s, t) \in D$*, then there is* $m \geq i$ *so that* r_m *belongs to a path in* $\bar{\mathcal{P}}$ *starting at* s*.*

Proof: We construct the trees iteratively. J_1 is any inclusion minimal tree in H rooted at r_1 that contains the set T_1 of all the terminals in $\bar{T}$ that are reachable in H from r_1. J_2 is any inclusion minimal

tree in H rooted at r_2 that contains the set T_2 of all the terminals in $\bar{T} - T_1$ that are reachable in H from r_2. And, in general, J_i is any inclusion minimal tree in H rooted at r_i that contains the set T_i of all the terminals in $\bar{T} - T_1 \cup \cdots \cup T_{i-1}$ that are reachable in H from r_i. By the construction, and since every path in $\bar{\mathcal{P}}$ collides with $\bar{P}$ and no edge leave the terminals, it is clear that the three properties given in the lemma hold. We explain why the trees $J_1, \ldots, J_d$ are pairwise edge disjoint. Otherwise, there are $1 \leq m < i \leq d$ so that J_m and J_i have an edge uv in common. By the minimality of J_i, there is a terminal $t \in \bar{T}_i$ reachable from v in J_i (possibly $v = t$). But then t is also reachable from r_m, hence, by the construction, t should have appeared in J_m and not in J_i, contradiction. □

Using Lemma 4.3, we show that the metric completion of H contains a low density junction star-tree as a subgraph. For a subgraph J of H let $k(J) = |V(J) \cap \bar{T}|$ denote the number of terminals from $\bar{T}$ in J .

Corollary 4.4 *There exists a junction star-tree J in the metric completion of H, such that (3) holds.*

Proof: Let $J_1, \ldots, J_d$ be the decomposition of H into trees as in Lemma 4.3. We will extend these rooted trees to junction star-trees by adding for every st path in $\mathcal{P}$ an edge sr_i from s to the root r_i of the tree J_i which includes t (see Fig. 1(b), if $s = r_i$ we need not add this edge). The cost of each new edge is at most $2c(H)/g$, since it shortcuts a path that is obtained by joining two subpaths of truncated paths (recall that each truncated path has cost less than $c(H)/g$). Let $J_1^+, \ldots, J_d^+$ denote the resulting junction star-trees. Every junction star-tree connects all its sources to the corresponding terminals, and therefore $\sum_{i=1}^{d} k(J_i^+) = \ell$. On the other hand we can bound the sum of the costs of the junction star-trees as follows:

$$\sum_{i=1}^{d} c(J_i^+) < \sum_{i=1}^{d} c(J_i) + \ell \cdot \frac{2c(H)}{g} \leq c(H) + \ell \cdot \frac{2c(H)}{g}$$

The last inequality holds because $J_1, \ldots, J_d$ are subgraphs of H that are pairwise edge disjoint. Using an averaging argument we get that there must be a junction star-tree $J = J_i^+$ whose density is bounded by:

$$\frac{c(J)}{k(J)} \leq \frac{c(H) + \ell \cdot 2c(H)/g}{\ell} = \frac{c(H)}{\ell} + \frac{2c(H)}{g} \leq c(H) \cdot \left(\frac{g}{k} + \frac{2}{g}\right)$$

The last inequality holds because $\ell \geq k/g$. □

4.2 The algorithm

Given a k-DSF instance assume that Reduction 1 and 2 are implemented.

Lemma 4.5 *For any k-DSF instance (after applying Reductions 1, 2), there exists a junction star-tree J so that $c(J)/|D(J)| \leq \mathsf{opt} \cdot \sqrt{8/k}$.*

Proof: This follows from Theorem 4.1 by choosing H as an optimal solution of a k-DSF instance (after applying Reductions 1, 2) and $g = \sqrt{2k}$. Indeed, if $c(P) \leq c(H)/g$ for some st-path P with $(s,t) \in D$, then P is a junction star-tree of density $\leq c(H)/g = c(H)/\sqrt{2k}$; otherwise, by Theorem 4.1, H contains a junction star-tree J of density $c(J)/|D(J)| \leq \sqrt{8/k} \cdot c(H)$. □

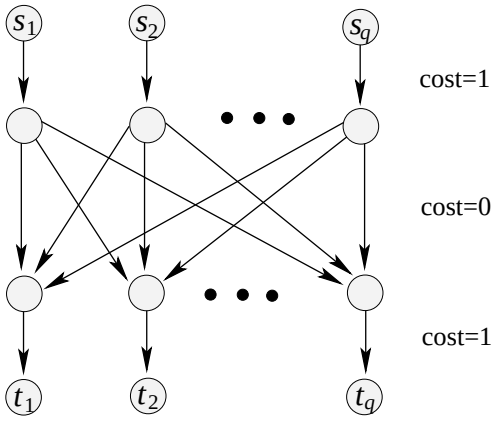

Figure 2: An example showing that the bound in Lemma 4.5 is tight.

Example: The following example shows that the bound in Lemma 4.5 is tight up to a constant factor. Consider the graph in Fig. 2, where $D = \{(s_i, t_j) : 1 \leq i, j \leq k\}$. Here $k = q^2$, and the lowest possible density of a junction star-tree is $(q+1)/q > 1$, while the density of the optimal solution (which is the entire graph) is $2q/q^2 = 2/q$.

Lemma 4.6 *Suppose that there exists an algorithm that given an instance of k-DSF finds an edge set J of density $\sigma \leq \mathsf{opt} \cdot \rho(k)/k$ and the set $D(J)$ of demand pairs that J connects in $T'(n,k)$ time. Then the $\rho(x)$-Greedy Algorithm for k-DSF can be implemented in $O(kT'(n,k))$ time.*

Proof: We need to show how to find a low density edge set J for every instance G, c, D of k-DSF and every partial cover F. For that, set $D \leftarrow D - D(F)$ to get an instance G, c, D' of $(k-|D(F)|)$-DSF. Then use the given algorithm for finding an edge set J of density at most $\mathsf{opt}' \cdot \rho(k-|D(F)|)/(k-|D(F)|) = \mathsf{opt}' \cdot \rho(\nu(F))/\nu(F) \leq \mathsf{opt} \cdot \rho(\nu(F))/\nu(F)$, where opt and opt' denote the optimum solution values of the instances G, c, D and G, c, D', respectively. The number of iterations is at most k, since in each iteration at least one more demand pair is satisfied. Hence the time complexity is $O(kT'(n,k))$. □

If we could find a low-density junction star-tree as in Lemma 4.5 in polynomial time, then we would obtain an $O(\sqrt{k})$-approximation algorithm for k-DSF, by Theorem 2.1 and Lemma 4.6. We will show how to find a junction star-tree of approximately optimal density using any approximation algorithm for k-DST; in particular, we can use the algorithm of [3].

Corollary 4.7 *If k-DST admits an α-approximation in $T(n,k)$ time then there exists an algorithm that given an instance of k-DSF finds a junction star-tree J satisfying $\sigma(J) \leq \mathsf{opt} \cdot \alpha \cdot \sqrt{8k}/k$ and $D(J)$ in $O(nkT(2n+k,k))$ time.*

Proof: We may assume that we know the root r of some optimal density star-junction tree, as we may try every $r \in V$. For every demand pair $(s,t) \in D$, add a new node t' and the edge tt' of cost $c(sr)$ (if $s = r$ let the cost of the edge be 0). Let T' be the set of nodes added. For every $1 \leq k' \leq k$ apply the α-approximation algorithm on the obtained instance of k'-DST with root r and terminal set T'. From the solutions computed, output one J' of the minimum density. The junction star-tree J is obtained from J' by replacing every terminal t' of J' by the corresponding edge sr. It is easy to see

that J' is as required, and that it is possible to calculate $D(J)$ without increasing the time complexity. The graph on which we call the algorithm for k'-DST has $n + |T| + |S| + k$ nodes due to Reduction 1 and the addition of the nodes of T'. However, $|S|$ of these nodes are sources (into which no edge enters) and can be removed before the algorithm for k'-DST is called. The time complexity follows. □

Combining Corollary 4.7 with the result of [3], Theorem 2.1, and Lemma 4.6, gives Theorem 1.2.

Remark: When using the algorithm of [3] for k-DST, the time complexity in Corollary 4.7 is in fact $O(nT(2n+k,k))$, since this algorithm approximates the minimum density augmentation tree in k-DST within the same time bound as approximating k-DST.

5 Conclusions and open problems

We presented the first sub-linear, in terms of n, approximation algorithm for the DSF problem. Due to a reduction from LABEL-COVER$_{\max}$ [9], obtaining an approximation ratio better than $O(\sqrt{n})$ (namely, $O(n^{1/2-\varepsilon})$ for some constant $\varepsilon > 0$) for DSF implies improving the best ratio for LABEL-COVER$_{\max}$ due to Peleg [32]. Still, it is an open question whether this ratio is indeed feasible. We also presented a simple combinatorial $O(k^{1/2+\varepsilon})$-approximation scheme for k-DSF, which matches the best known LP-based algorithm of Chekuri et al. [4] for the less general problem DSF. Our result also (almost) matches the best know ratio in terms of k for the undirected version of the problem by Gupta et al. [2]. It is interesting to note that the situation is completely different in terms of n, as there is no known non-trivial approximation ratio for k-DSF in terms of n, while the undirected version admits an $O(\sqrt{n})$-approximation [2]. It is an open question whether the asymmetry between the parameters n and k can be reduced.

Almost every aspect of the more general Directed Steiner Network problem is still an open problem. No non-trivial approximation ratio is known for this problem, not even for the simple case when the maximum requirement is 2. In contrast, the Undirected Steiner Network problem was studied extensively, and admits a 2-approximation algorithm due to Jain [21].

Note that the Directed Steiner Network problem can be trivially solved using min-cost flow techniques when there is only a single positive requirement pair. This fact can be used to achieve a k approximation for the Directed Steiner Network problem, where k is the number of positive requirement pairs: Simply solve independently for every positive requirement pair and combine the resulting graphs. A similar algorithm also extends to the more general problem of k-Directed Steiner Network, where we are only required to connect k positive requirement pairs. Again, we can solve the problem separately for each positive requirement pair and then combine the k cheapest resulting graphs.

We also note that on directed graphs, there is an approximation ratio preserving reduction between the edge-weighted and the node-weighted versions, but this is not so for undirected graphs. On undirected graphs, the best known ratio for the Node-Weighted Steiner Forest is $O(\log |U|)$ due to Klein and Ravi [25] and this is tight (up to a constant factor), where U is the set of nodes involved in a positive requirement pair. Recently, an $r_{\max} \cdot O(\ln |U|)$-approximation algorithm for the *undirected* Node-Weighted Steiner Network problem was presented by Nutov [31], where $r_{\max} = \max_{u,v \in V} r(u,v)$ is the largest requirement.

We believe that it should be possible to approximate the Directed Steiner Network problem to a factor of $r_{\max} \cdot \rho_{\mathrm{DSF}}$, where ρ_{DSF} is the approximation ratio of the DSF problem. A possible way to approach this ratio is through the Rooted Directed Steiner Network problem, a restricted version of the Directed Steiner Network problem in which the set of pairs with positive requirement is a subset of $\{s\} \times V$, for some node $s \in V$. We believe that an approximation ratio of $r_{\max} \cdot \rho_{\mathrm{DST}}$ should be feasible for this problem (where ρ_{DSF} is the approximation ratio of the DST problem). We also believe that using the ideas presented in this paper, the Direct Steiner Network problem can be reduced to a variant of Rooted Direct Steiner Network, in a way resembling our reduction from the DSF problem to a variant of DST.

References

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

[2] A. G. an M. T. Hajiaghayi, V. Nagarajan, and R. Ravi. Dial a ride from k-Forest. In *European Symposium on Algorithms (ESA)*, pages 241–252, 2007.

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

[4] C. Chekuri, G. Even, A. Gupta, and D. Segev. Set connectivity problems in undirected graphs and the directed Steiner network problem. To appear in Symposium on Discrete Algorithms (SODA) 2008.

[5] C. Chekuri, G. Even, and G. Kortsarz. A greedy approximation algorithms for the group Steiner problems. *Discrete applied Math.*, 154(1):15–34, 2006.

[6] C. Chekuri, M. T. Hajiaghayi, G. Kortsarz, and M. R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design. In *Symposium on the Foundations of Computer Science (FOCS)*, pages 677–686, 2006.

[7] C. Chekuri, S. Khanna, and J. Naor. A deterministic algorithm for the cost-distance problem. In *Symposium on Discrete Algorithms (SODA)*, pages 232–233, 2001.

[8] F. Chudak, T. Roughgarden, and D. Williamson. Approximate k-MSTs and k-Steiner trees via the primal-dual method and lagrangean relaxation. In *Integer Programming and Combinatorial Optimization (IPCO)*, pages 60–70, 2001.

[9] Y. Dodis and S. Khanna. Design networks with bounded pairwise distance. In *Symposium on the Theory of Computing (STOC)*, pages 750–759, 1999.

[10] G. Even. *Recursive greedy methods,* Ch. 5 in *Approximation Algorithms and Metahueristics,* T. F. Gonzales ed. CRC, 2007.

[11] U. Feige, G. Kortsarz, and D. Peleg. The dense k-Subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[12] J. Feldman and M. Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. In *Symposium on the Foundations of Computer Science (FOCS)*, page 299, 1999.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences).* W. H. Freeman, 1979.

[14] N. Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Symposium on the Theory of Computing (STOC)*, pages 396–402, 2005.

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

[16] M. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Computing*, 24(2):296–317, 1995.

[17] M. T. Hajiaghayi and K. Jain. The prize-collecting generalized steiner tree problem via a new approach of primal-dual schema. In *Symposium on Discrete Algorithms (SODA)*, pages 631–640, 2006.

[18] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Symposium on the Theory of Computing (STOC)*, pages 585–594, 2003.

[19] R. Hassin. Approximation schemes for the restricted shortest path problem. *Math. Oper. Res.*, 17(1):36–42, 1992.

[20] C. H. Helvig, G. Robins, and A. Zelikovsky. Improved approximation scheme for the group Steiner problem. *Networks*, 37(1):8–20, 2001.

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

[22] S. Khot. On the unique games conjecture. In *Symposium on the Foundations of Computer Science (FOCS)*, page 3, 2005.

[23] S. Khuller. *Approximation algorithms for for finding highly connected subgraphs,* Chapter 6 in *Approximation Algorithms for NP-hard problems,* D. S. Hochbaum Ed., pages 236-265. PWS, 1995.

[24] S. Khuller and L. Zosin. On directed Steiner trees. In *Symposium on Discrete Algorithms (SODA)*, pages 59–63, 2002.

[25] 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.

[26] G. Kortsarz and Z. Nutov. Tight approximation for connectivity augmentation problems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 443–452, 2006.

[27] G. Kortsarz and Z. Nutov. *Approximating minimum cost connectivity problems,* Ch. 58 in *Approximation Algorithms and Metahueristics,* T. F. Gonzales ed.,. CRC, 2007.

[28] G. Kortsarz and D. Peleg. Approximating the weight of shallow Steiner trees. *Discrete Applied Math.*, 93:265–285, 1999.

[29] D. Lorenz and D. Raz. A simple efficient approximation scheme for the restricted shortest paths problem, 1999.

[30] A. Meyerson, K. Munagala, and S. A. Plotkin. Cost-distance: Two metric network design. In *Symposium on the Foundations of Computer Science (FOCS)*, pages 624–630, 2000.

[31] Z. Nutov. Approximating steiner networks with node weights. To appear in Latin American Theoretical Informatics Symposium (LATIN) 2008.

[32] D. Peleg. Approximation algorithms for the label-cover$_{MAX}$ and red-blue set cover problems. *J. of Discrete Algorithms*, 5(1):55–64, 2007.

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

[34] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Symposium on Discrete Algorithms (SODA)*, pages 770–779, 2000.

[35] D. Segev and G. Segev. k-generalized connectivity via collapsing hierarchically well-separated trees. Manuscript, 2008.

[36] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18:99–110, 1997.