# Euclidean Prize-collecting Steiner Forest[*]

MohammadHossein Bateni[†] MohammadTaghi Hajiaghayi[‡]

**Abstract**

In this paper, we consider *Steiner forest* and its generalizations, *prize-collecting Steiner forest* and *k-Steiner forest*, when the vertices of the input graph are points in the Euclidean plane and the lengths are Euclidean distances. First, we present a simpler analysis of the polynomial-time approximation scheme (PTAS) of Borradaile et al. [12] for the *Euclidean Steiner forest* problem. This is done by proving a new structural property and modifying the dynamic programming by adding a new piece of information to each dynamic programming state. Next we develop a PTAS for a well-motivated case, i.e., the multiplicative case, of prize-collecting and budgeted Steiner forest. The ideas used in the algorithm may have applications in design of a broad class of bicriteria PTASs. At the end, we demonstrate why PTASs for these problems can be hard in the general Euclidean case (and thus for PTASs we cannot go beyond the multiplicative case).

## 1 Introduction

Prize-collecting Steiner problems are well-known network design problems with several applications in expanding telecommunications networks (see e.g. [25, 32]), cost sharing, and Lagrangian relaxation techniques (see e.g. [24, 15]). The most general version of these problems is called the *prize-collecting Steiner forest (PCSF)* problem[1], in which, given a graph $G = (V, E)$, a set of (commodity) pairs $\mathcal{D} = \{(s_1, t_1), (s_2, t_2), \dots\}$, a non-negative cost function $c : E \to \mathbf{Q}^{\geq 0}$, and finally a non-negative penalty function $\pi : \mathcal{D} \to \mathbf{Q}^{\geq 0}$, our goal is a minimum-cost way of buying a set of edges and paying the penalty for those pairs which are not connected via bought edges. When all penalties are $\infty$, the problem is the classic APX-hard *Steiner forest* problem for which the best approximation factor is $2 - \frac{2}{n}$ ($n$ is the number of vertices of the graph) due to Goemans and Williamson [19]. When all sinks are identical in the PCSF problem, it is the classic prize-collecting Steiner tree problem. Bienstock, Goemans, Simchi-Levi, and Williamson [8] first considered this problem (based on a problem earlier proposed by Balas [3]) for which they gave a 3-approximation algorithm. The current best approximation algorithm for this problem is a recent 1.992-approximation algorithm of Archer, Bateni, Hajiaghayi, and Karloff [1] improving upon a primal-dual $\left(2 - \frac{1}{n-1}\right)$-approximation algorithm of Goemans and Williamson [19]. When in addition all penalties are $\infty$, the problem is the classic *Steiner tree* problem, which is known to be APX-hard [7] and for which the best known approximation factor is 1.55 [31].

[*]A short version of this paper appears in Proceedings of LATIN 2010 [5].

[†]Department of Computer Science, Princeton University, Princeton, NJ 08540; Email: mbateni@cs.princeton.edu. The author was supported by a Gordon Wu fellowship as well as NSF ITR grants CCF-0205594, CCF-0426582 and NSF CCF 0832797, NSF CAREER award CCF-0237113, MSPA-MCS award 0528414, NSF expeditions award 0832797.

[‡]AT&T Labs—Research, Florham Park, NJ 07932; Email: hajiagha@research.att.com.

[1]It is sometimes called *prize-collecting generalized Steiner tree (PCGST)* in the literature.

There are several 3-approximation algorithms for the *prize-collecting Steiner forest* problem using LP rounding, primal-dual, or iterative rounding methods which are first initiated by Hajiaghayi and Jain [22] (see [8, 23]). Currently the best approximation factor for this problem is a randomized 2.54-approximation algorithm [22]. The approach of Hajiaghayi and Jain has been generalized by Sharma, Swamy, and Williamson [33] for network design problems where violating arbitrary 0-1 connectivity constraints are allowed in exchange for a very general penalty function.

Lots of attention has been paid to budgeted versions of Steiner problems as well. In the *$k$-Steiner forest* (or just $k$-forest for abbreviation), given a graph $G = (V, E)$ and a set of (commodity) pairs $\mathcal{D}$, the goal is to find a minimum-cost forest that connects at least $k$ pairs of $\mathcal{D}$. The best current approximation factor for this problem is in $O(\min\{\sqrt{k}, \sqrt{n}\})$ [21]. On the other hand, Hajiaghayi and Jain [22] could transform notorious *dense $k$-subgraph* to this problem, for which the current best approximation factor is $O(n^{1/3-\epsilon})$ [16]. The special case in which we have a root $r$ and $\mathcal{D}$ consists of all pairs $(r, v)$ for $v \in V(G) - \{r\}$ is the well-known NP-hard $k$-MST problem. The first non-trivial approximation algorithm for the $k$-MST problem was given by Ravi et al. [30], who achieved an approximation ratio of $O(\sqrt{k})$. Later this approximation ratio is improved to a constant by Blum et al. [9]. Currently the best approximation factor for this problem is 2 due to Garg [17].

In this paper, we consider *Euclidean prize-collecting Steiner forest* and *Euclidean $k$-forest* in which the vertices of the input graph are points in the Euclidean plane (or low-dimensional Euclidean space) and the lengths are Euclidean distances. For the *Euclidean Steiner tree* problem, Arora [2] and Mitchell [29] gave polynomial-time approximation schemes (PTASs). Recently Borradaile, Klein and Kenyon-Mathieu [12] claim a PTAS for the more general problem of *Euclidean Steiner forest* .

## 1.1 Problem definition

Motivated by the settings in which the demand of each pair is the product of the weight of the origin vertex and the weight of the destination vertex in the pair and thus in a sense contributions of each vertex to all adjacent pairs are the same (e.g., see *product multi-commodity flow* in Leighton and Rao [27] or [10, 26], and its applications in wireless networks [28] or routing [13, 14]), we consider the following multiplicative version of prize-collecting Steiner forest for the Euclidean case.

In the *Multiplicative prize-collecting Steiner forest (MPCSF)* problem, given an undirected graph $G(V, E)$ with non-negative edge lengths $c_e$ for each edge $e \in E$, and also given weights $\phi(v)$ for each vertex $v \in V$, our goal is to find a forest $F$ which minimizes the cost

$$\sum_{e\in F} c_e + \sum_{u,v\in V:\ u \text{ and } v \text{ are not connected via } F} \phi(u)\phi(v).$$

Indeed, this is an instance of PCSF in which each ordered vertex pair $(u, v)$ forms a request with penalty $\phi(u)\phi(v)$.[2] We may be asked to *collect a certain prize $S$*, in which case the goal is to find the forest $F$ of minimum cost for which

$$\sum_{u,v\in V:\ u \text{ and } v \text{ are connected via } F} \phi(u)\phi(v) \geq S.$$

[2]We can change the definition to unordered pairs whose treatment requires only a slight modifications of the algorithms. Currently, each unordered pair $(u, v)$ has a prize of $2\phi(u)\phi(v)$ if $u \neq v$.

Let us call this problem $S$-MPCSF. We show that this is a generalization of the $k$-MST problem (see Appendix A.2) and thus currently there is no approximation better than 2 for this problem either. When working on the Euclidean case, the input does not include any Steiner vertices, as all the points of the plane are potential Steiner points.

A bicriteria $(\alpha, \beta)$-approximate solution for the the $S$-MPCSF problem is one whose cost is at most $\alpha$OPT, yet collects a prize of at least $\beta S$. Our main contribution in this paper is a bicriteria $(1 + \epsilon, 1 - \epsilon')$-approximation algorithm that runs in time exponential in $1/\epsilon$ but polynomial in $n$ and $1/\epsilon'$. We then use this algorithm to obtain a PTAS for MPCSF.

## 1.2 Our contribution

First of all, we present a simpler analysis for the algorithm of Borradaile et al. [12] for the *Euclidean Steiner forest* problem and reprove the following theorem.

**Theorem 1.** For any constant $\epsilon > 0$, there is an algorithm that runs in polynomial time and approximates the *Euclidean Steiner forest* problem within $1 + \epsilon$ of the optimal solution.

This is done by modifying the dynamic programming (DP) algorithm so that instead of storing paths enclosing the *zones* in the algorithm by Borradaile et al., we use a bitmap to identify a zone. The modification results in simplification of the structural property required for the proof of correctness (See Section 3). We prove this structural property in Theorem 6. The proof has some ideas similar to [12], but we present a simpler charging scheme that has a universal treatment throughout. Next we give an overview of the dynamic programming algorithm in Section 4. We have recently come to know that similar simplifications have been independently discovered by the authors of [12], too.

Next we extend the algorithm for Euclidean $S$-MPCSF and MPCSF problems in Section 5.

**Theorem 2.** For any $\epsilon, \epsilon' > 0$, there is a bicriteria $(1 + \epsilon, 1 - \epsilon')$-approximation algorithm for the *Euclidean $S$-MPCSF* problem, that runs in time polynomial in $n, 1/\epsilon'$ and exponential in $1/\epsilon$.

Notice that $\epsilon'$ need not be a constant. In particular, if all weights are polynomially bounded integers, we can find in polynomial time a $(1 + \epsilon)$-approximate solution that collects a prize of at least $S$; this can be done by picking $\epsilon'$ to be sufficiently small ($\epsilon'^{-1}$ is still polynomial). Next we present a PTAS for *Euclidean MPCSF*.

**Theorem 3.** For any constant $\epsilon$, there is a $(1 + \epsilon)$-approximation algorithm for the *Euclidean MPCSF* problem, that runs in polynomial time.

We also study the case of asymmetric prizes for vertices in which each vertex $v$ has two types of weights (type one and type two) and the prize for an ordered pair $(u, v)$ is the product of the first type weight of $u$, i.e., $\phi^s(u)$, and the second type weight of $v$, i.e., $\phi^t(v)$. This case is especially interesting because it generalizes the multiplicative prize-collecting problem when we have two disjoint sets $S_1$ and $S_2$ and we pay the multiplicative penalty only when two vertices, one in $S_1$ and the other one in $S_2$, are not connected (by letting for each vertex in $S_1$ the first type weight be its actual weight and the second type weight be zero and for each vertex in $S_2$ the first type weight be zero and the second type weight be its actual weight.) After hinting on the arising complications, we show how we can extend our algorithms for this case as well.

**Theorem 4.** For any $\epsilon, \epsilon' > 0$, there is a bicriteria $(1+\epsilon, 1-\epsilon')$-approximation algorithm for the *Euclidean Asymmetric $S$-MPCSF* problem, that runs in time polynomial in $n, 1/\epsilon'$ and exponential in $1/\epsilon$. In addition, for any constant $\epsilon$, there is a $(1+\epsilon)$-approximation algorithm for the *Euclidean Asymmetric MPCSF* problem, that runs in polynomial time.

Indeed, the algorithms in Theorem 4 can be extended to the case in which there are a constant number of different types of weights for each vertex generalizing the case in which we have a constant number of disjoint sets and we pay the multiplicative penalty when two vertices from two different sets are not connected. Notice that the case of two disjoint sets already generalizes the *prize-collecting Steiner tree* problem (by considering $S_1 = \{r\}$ and $S_2 = V - \{r\}$) whose best approximation guarantee is currently 1.992.

At the end, we present in Section 6 why PCSF and $k$-forest problems can be APX-hard in the general case (and thus for PTASs we cannot go beyond the multiplicative case). We conclude with some open problems in Section 7. All the omitted proofs appear in the appendix.

### 1.3 Our techniques for the prize-collecting version

Here, we summarize our techniques for the multiplicative prize collecting Steiner forest algorithms; see Section 5. In all those algorithms, we store in each DP state extra parameters, including the sum of the weights, as well as the multiplicative prize already collected in each component. These parameters enable us to carry out the DP update procedure. Interestingly, the sum and collected prize parameters have their own precision units.

In the asymmetric version, a major issue is that no fixed unit is good for all sum parameters. Some may be small, yet have significant effect when multiplied by others. To remedy this, we use variable units, reminiscent of the floating-point storage formats (mantissa and exponent). To the best of our knowledge, Bateni and Hajiaghayi [4] were the first to take advantage of this idea in the context of (polynomial time) approximation schemes. The basic idea is that a certain parameter in the description of DP states has a large (not polynomial) range, however, as the value grows, we can afford to sacrifice more on the precision. Thus, we store two (polynomial) integer numbers, say $(i, x)$, where $i$ denotes a variable unit, and $x$ is the coefficient: the actual number is then recovered by $x \cdot u_i$. The conversion between these representations is not lossless, but the aggregate error can be bounded satisfactorily.

In Section 5.3 we consider the problem where the objective is a linear function of penalties paid and the cost of the forest built. The challenging case is when the cost of the optimal forest is very small compared to the penalties paid. In this case, we identify a set of vertices with large penalties and argue they have to be connected in the optimal solution. Then, with a novel trick we show how to ignore them in the beginning, and take them into account only after the DP is carried out.

## 2 Preliminaries

Let $n = |V|$ be the total number of terminals and let OPT be the total length of the optimal solution. A *bitmap* is a matrix with 0-1 entries. Two bitmaps of the same dimensions are called *disjoint* if and only if they do not have value one at the same entry. Consider two partitions $\mathcal{P} = \{P_1, P_2, \ldots, P_{|\mathcal{P}|}\}$ and $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_{|\mathcal{P}'|}\}$ over the same ground set. Then, $\mathcal{P}$ is said to be a *refinement* of $\mathcal{P}'$ if and only if any set of $\mathcal{P}$ is a subset of a set in $\mathcal{P}'$, namely $\forall P \in \mathcal{P}, \exists P' \in \mathcal{P}' : P \subseteq P'$.

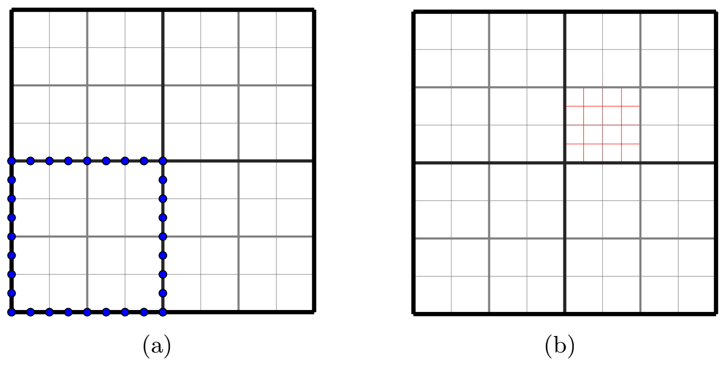

Figure 1: (a) An example of a dissection square with depth 3, and depiction of portals for a sample dissection square with $m = 8$; (b) the $\gamma \times \gamma$ grid of cells inside a sample dissection square with $\gamma = 4$.

By standard perturbation and scaling techniques, we can assume the following conditions hold incurring a cost increase of $O(\epsilon \text{OPT})$; see [2, 12] for example.

(I) The diameter of the set $V$ is at most $d' = n^2\epsilon^{-1}\text{OPT}$.

(II) All the vertices of $V$ and the Steiner points have coordinates $(2i + 1, 2j + 1)$ where $i$ and $j$ are integers.

For simplicity of exposition, we ignore the above increase in cost. As we are going to obtain a PTAS, this increase will be absorbed in the future cost increases. We have a grid consisting of vertical and horizontal lines with equations $x = 2i$ and $y = 2j$ where $i$ and $j$ are integers. Let $\mathcal{L}$ denote the set of lines in the grid. We let $L$ be the smallest power of two greater than or equal to $2d'$ and perform a dissection on the randomly shifted bounding box of size $L \times L$; see Figure 1(a).

For each dissection square $R$ and each side $S$ of $R$, designate $m + 1$ equally spaced points along $S$ (including the corners) as *portals* of $R$ where $m$ is the smallest power of 2 greater than $4\epsilon^{-1} \log L$. So the square $R$ has $4m$ portals.

There is a notion of *level* associated with each dissection square, line, or side of a square. The bounding box has level zero, and level of each other dissection square is one more than the level of its parent dissection square. The level of a line $\ell$ is the minimum level of a square $R$ a side of which falls on the line $\ell$. Thus, the first two lines dividing the bounding box have level one. If a side $S$ of a square $R$ falls on a line $\ell$, we define $\text{level}(S) = \text{level}(\ell)$. So $\text{level}(S) \leq \text{level}(R)$. The thickness of the lines in Figure 1 denotes their level: the thicker the line, the lower is its level.

For a (possibly infinite) set of geometric points $X$, let $\text{comp}(X)$ denote the number of connected components of $X$; we will use the shorthand "component" in this paper. With slight abuse of notation, $\ell \in \mathcal{L}$ is used to refer to the set of points[3] on $\ell$. In addition, we use $\mathcal{L}$ to denote the union of points on the lines in $\mathcal{L}$. Similarly, we use $R$ to denote the set of all points on or inside the square $R$. The set of points on (the boundary of) the square $R$ is referred to by $\partial R$. The total length of all line segments in $F$ is denoted by $\text{length}(F)$.

The following theorem is mentioned in [12] in a stronger form. We only need its first half whose proof follows from [2].

[3]not necessarily terminals

**Theorem 5.** [12] *There is a solution $F$ having expected length at most $(1+\frac{1}{4}\epsilon)\mathrm{OPT}$ such that each dissection square $R$ satisfies the following two properties: for each side $S$ of $R$, $F \cap S$ has at most $\rho = O(\epsilon^{-1})$ non-corner components*[4] *(boundary components property); and each component of $F \cap \partial R$ contains a portal of $R$ (portal property).*

## 3 Structural theorem

Let $R$ be a dissection square. Divide $R$ into a regular $\gamma \times \gamma$ grid of *cells*, where $\gamma$ is a constant power of two determined later; see Figure 1(b). We say $R$ is the *owner* of these cells. The level of these cells, as well as the new lines they introduce, is defined in accordance with the dissection. That is, we assign them levels as if they are normal dissection squares and we have continued the dissection procedure for $\log \gamma$ more levels. There are several lemmas in the work of [12] to prove the structural property they require (this is the main contribution of that work). We modify the dynamic programming definition such that its proof of correctness needs a simpler structural property. The proof of this property is simpler than that in the aforementioned paper.

**Theorem 6.** *There is a solution $F$ having expected length at most $(1+\frac{1}{2}\epsilon)\mathrm{OPT}$ such that each dissection square $R$ satisfies the locality property: if the terminals $t_1$ and $t_2$ are inside a cell $C$ of $R$ and are connected to $\partial R$ via $F$, then they are connected in $F \cap R$.*

The proof has ideas similar to [12, Theorem 3.2, and Lemmas 3.3, 3.4, 3.5 and 3.9]. We first mention and prove a lemma we need in order to prove Theorem 6. The lemma more or less appears in [2, 12].

**Lemma 7.** *For the forest $F$ output by Theorem 5, $\mathrm{comp}(F \cap \mathcal{L}) \leq \mathrm{length}(F)$.*

We can now prove the main structural result. A side $S$ of a square $R$ is called *private* if it does not lie on a side of the parent square $R'$ of $R$. Observe that out of any two opposite sides of a dissection square, exactly one is private.

**Proof of Theorem 6.** We start with a solution $F$ satisfying Theorem 5. The final solution is produced by iteratively finding the smallest cell $C$ owned by a square $R$ that violates the locality property, and adding $\sigma(C, F)$ to $F$, where $\sigma(C, F)$ is defined as the union of the private sides of $C$ and any side of $C$ having non-empty intersection with $F$. We claim the locality property is realized after finitely many such additions. If after adding $\sigma(C, F)$ to $F$, the cell $C$ still violates the locality property, there has to be exactly two opposite sides of the cell having non-empty intersection with $F$; otherwise, the $\sigma(C, F)$ is clearly connected. However, in case of the opposite sides, one middle side will be a private side of $C$ and hence included as well.

Next, we argue that the conditions of Theorem 5 still hold. Take a side $S$ of any square $R$. If the conditions are to be affected for $S$, it has to be due to an addition involving a cell $C$ that has a side $S'$ such that (1) $S'$ has non-empty intersection with $S$, and (2) $S'$ is added to $F$ as part of $\sigma(C, F)$. The condition will be trivial if $S'$ contains $S$. Thus, we assume that $C$ is a smaller square than $R$. So $S'$ cannot be a private side of $C$. However, the number of components on $S$ cannot increase if $S'$ has already an intersection with $F$.

[4] Non-corner components are those not including any corners of squares. Note that each square can have at most four corner components.

Finally we show that the additional length is not large. Let $F^* = F \cap \mathcal{L}$, and let $\mathcal{G} = \{(x, y) : x = 2i, y = 2j\}$ be the set of all grid points. We will charge the additions to the connected components of $F^* - \mathcal{G}$. Notice that

$$\begin{aligned}
\operatorname{comp}(F^* - \mathcal{G}) &\leq \operatorname{comp}(F^*) + 3|F^* \cap \mathcal{G}| && (1)\\
&\leq \operatorname{comp}(F^*) + 3 \cdot (\operatorname{length}(F^*) + \operatorname{comp}(F^*)) && (2)\\
&= 4 \operatorname{comp}(F^*) + 3 \operatorname{length}(F) &&\\
&\leq 7 \operatorname{length}(F), \qquad\qquad \text{by Lemma 7.} && (3)
\end{aligned}$$

Inequality (1) holds because removal of each grid point on $F^*$ increases the number of components by at most three. To obtain (2), notice that in any connected component of $F^*$, the distance between any two points of $F^* \cap \mathcal{G}$ is at least 2. Hence, if there are more than one such points, there cannot be more than length$(F^*)$ ones.

We charge this addition to a connected component of $(\partial R \cap F) - \mathcal{G}$, in such a way that each connected component is charged to at most twice: once from each side. For simplicity, we duplicate each connected component of $(F \cap \ell) - \mathcal{G}$: they correspond to squares from either side of $\ell$. For any dissection square $R$, let $\mathcal{C}_R$ refer to the connected components of $F \cap R$ that reach $\partial R$. Further, let $\mathcal{K}_R$ be the set of connected components of $(F \cap \partial R) - \mathcal{G}$. When $\sigma(C, F)$ is added where $R$ is the owner of $C$, there are $k \geq 2$ components $c_1, \ldots, c_k \in \mathcal{C}_R$ that become connected. Any element of $\mathcal{K}_R$ connected via $F \cap R$ to a component $c \in \mathcal{C}_R$ is said to be an *interface* of $c$. The addition will be charged to a *free* interface of some $c \in \mathcal{C}_R$ with maximum level. This element will no longer be free for the rest of the procedure. We argue this procedure successfully charges all the additions to appropriate border components. To this end, we shortly prove the following stronger claim via induction on the number of additions performed. We call a dissection square $R$ *violated* if the locality property does not hold for a cell $C$ owned by $R$.

**Claim 8.** At all times during the execution of this procedure, any component $c \in \mathcal{C}_R$ has a free interface, for each violated square $R$. As a result, any addition can be charged to a free component.

The second statement of the claim follows from the first part. The first part is proved as follows. The claim clearly holds at the beginning, since all interfaces are free, and each component has an interface. Suppose the addition $\sigma(C, F)$ is performed and let $R$ be the owner of $C$. We show any dissection square $R'$ will stay fine. Notice that the size of the squares $R$ for which the addition is performed is increasing in time. Hence, any dissection square $R'$ smaller than $R$ is irrelevant in the statement of the claim, since they cannot be violated. For $R$ itself, each $c_i$ has at least one free interface. One of the interfaces is used, and thus the new component formed by their union has a free interface. Suppose for the sake of reaching a contradiction that a component $c' \in \mathcal{C}_{R'}$ has no free interface after the addition. Thus $R'$ contains $R$, and the charging was not done to a private side of $R$. Recall that prior to the addition, $c'$ is connected to some components of $\mathcal{C}_R$ with at least two free interfaces in $R$. One of them still remains free. We charged to the interface of maximum level and it was in $\partial R'$. Hence, the free interface is also in $\partial R$, leading to a contradiction.

Let (the random variable) $c_{\ell,j}$ denote the number of charges to components on $\ell \in \mathcal{L}$ due to cells $C$ owned by squares $R$ of level $j$. Independently of the randomness $\sum_\ell \sum_j c_{\ell,j} \leq 2 \operatorname{comp}(F^* - \mathcal{G})$ by the above discussion and Claim 8. Note the cost of adding $\sigma(C, F)$ (charged to a component on $R$) is at most $4L'/\gamma$ where $L'$ is the side length of $R$. The total increase due to charges to $\ell$ is at most $\sum_{j \geq \operatorname{depth}(\ell)} c_{\ell,j} \frac{4L}{\gamma 2^j}$ where $L$ is the side length of the bounding box. Due to the randomization in the dissection, we have $\mathbf{Pr}[\mathrm{depth}(\ell) = i] = 2^i/L$; see [2] for instance. The expected increase in length is thus

$$\begin{aligned}
\sum_{\ell}\sum_{i}\frac{2^i}{L}\sum_{j\geq i}c_{\ell,j}\frac{4L}{\gamma 2^j} &\leq \frac{4}{\gamma}\sum_{\ell}\sum_{j}\frac{c_{\ell,j}}{2^j}\sum_{i\leq j}2^i \\
&\leq \frac{8}{\gamma}\sum_{\ell}\sum_{j}c_{\ell,j} \\
&\leq \frac{16}{\gamma}\mathrm{comp}(F^* - \mathcal{G}) && \text{by Claim 8} \\
&\leq \frac{112}{\gamma}\mathrm{length}(F) && \text{by (3).}
\end{aligned}$$

We pick $\gamma$ to be the smallest power of two larger than $112(1+\epsilon)\cdot 2\epsilon^{-1}$ to finish the proof. □

Therefore, with probability 1/2, we have $\mathrm{length}(F) \leq (1+\epsilon)\mathrm{OPT}$. In the entire argument, no attempt was made to optimize the parameters.

## 4 The algorithm

A *subsolution* for $R$ is a finite set of line segments $F \subset R$ satisfying conditions of Theorems 5 and 6, with the extra property that any terminal $t$ in $R$ is connected via $F$ either to its mate or to $\partial R$. A *configuration* $\chi = (\mathcal{K}, \mathcal{P})$ for $R$ has two portions: a set $\mathcal{K}$ of pairs $\kappa_i = (P_i, M_i)$ and a partition $\mathcal{P}$ whose ground set is $\mathcal{K}$, such that

- $P_i$ is a subset of portals of $R$;
- $M_i$ is a bitmap of size $\gamma \times \gamma$;
- $P_i$ and $P_j$ are disjoint if $i \neq j$;
- the total number of portals, namely $\sum_i |P_i|$, is at most $4(\rho+1)$; and
- bitmaps $M_i$ and $M_j$ are disjoint if $i \neq j$.

The configuration captures sufficient information about $F$ so as to make it possible to take care of the interaction between $R$ and the outside. In particular, each pair $(P, M)$ describes a connected component of $F$, by specifying the set of portals on its boundary and the set of cells connected to these portals. Roughly speaking, the partition $\mathcal{P}$ tells us which components $\kappa_i$ and $\kappa_j$ need to be connected from outside $R$: this implies the existence of a pair of terminals that are in $\kappa_i$ and $\kappa_j$, respectively, but they are not connected in $R$. We will see below why this restrictive abstraction does not lose any crucial subsolutions.

We say a subsolution $F$ is *compatible* with a configuration $\chi = (\mathcal{K}, \mathcal{P})$ if

1. for any connected component $\kappa$ of $F$ that intersects $\partial R$, there exists a pair $\kappa' = (P, M) \in \mathcal{K}$ such that
    - $\kappa$ spans $P$;
    - each connected component of $\kappa \cap \partial R$ contains a portal of $P$;

- the bitmap $M$ has value one in the positions corresponding to any cell $C$ containing a terminal $t$ of $\kappa$; and

2. any terminal pair located in different components $\kappa_1$ and $\kappa_2$ of $\mathcal{K}$ are either connected via $F \cap R$, or $\kappa_1$ and $\kappa_2$ are in the same set of $\mathcal{P}$.

## 4.1 The dynamic programming

In the dynamic program, we build a table $T_R[\chi]$, indexed by configurations for each dissection square $R$. The goal is to populate this table so that $T_R[\chi]$ is the minimum length of a subsolution for $R$ that is compatible with $\chi$. First of all, we show that for each $R$, the number of configurations is small. Consider $\chi = (\mathcal{K}, \mathcal{P})$. There are at most $\lambda = 4(\rho + 1)$ pairs in $\mathcal{K}$. For a particular $\kappa = (P, M)$, there are $\sum_{i=0}^{\lambda} \binom{m+1}{i} = O(m^{\lambda+1})$ options for the set of portals $P$. The bitmap $M$ has $2^{\gamma^2}$ possibilities. A crude upper bound of $2^{\lambda^2}$ is trivial for possibilities of $\mathcal{P}$. Thus, the total number will be at most

$$\Phi = \left[O\left(m^{\lambda+1}\right) \cdot 2^{\gamma^2}\right]^{\lambda} \cdot 2^{\lambda^2} = O(\text{poly}(m)) = O(\text{poly}\log(n)).$$

Theorems 5 and 6 guarantee the existence of a near-optimal solution all whose subproblems are compatible with a configuration: The connected components of $F$ reaching $\partial R$ can be decomposed into disjoint bitmaps because of Theorem 6. Theorem 5 on the other hand ensures each connected component on $\partial R$ contains a portal, and the total number of such components is small. The details of the DP update, as well as its correctness proof, appears below.

The final solution of the problem is obtained from the minimum $T_R[\chi]$ where $R$ is the bounding box, and $\mathcal{P}$ of $\chi$ does not require any connections: i.e., all sets of the partition are singletons. This would imply all the necessary connections have been made inside $R$. To actually construct the solution, we need to store additional information in each dynamic programming state indicating which configurations it was last updated from. It is then straightforward to recursively construct the solution, by taking the union of the pertinent configurations.

Here we show how the dynamic programming table for Euclidean prize-collecting Steiner forest is updated from the already-computed values. And finally we show why the update routine is sound and complete. The table $T_R[\chi]$ is populated in the order of increasing size for $R$. For a base dissection square $R$, finding the value of $T_R[\chi]$ is straightforward. Notice that there is at most one point (possibly with several terminals collocated) inside $R$. Depending on whether the mates of those terminals are collocated with them or not, we may need to connect some of them to the boundary $\partial R$. There are only a constant number of portals in $\chi$, hence we can go over all the ways to connect them up and find the smallest value. Note that there cannot be any Steiner point inside $R$.

Now we get to the update rule. Consider a dissection square $R$ and a corresponding configuration $\chi = (\mathcal{K}, \mathcal{P})$. Let $R_i$ for $i = 1, 2, 3, 4$, be the children of $R$ in the dissection. Take corresponding configurations $\chi_i = (\mathcal{K}_i, \mathcal{P}_i)$. Notice that each cell of $R$ consists of exactly four cells of one $R_i$. We can expand a bitmap $M$ of $R_i$ to a bitmap $M'$ of dimensions $2\gamma \times 2\gamma$ for $R$, by placing three all-zero bitmaps of dimensions $\gamma \times \gamma$ at appropriate locations around $M$. We do this in such a way that the portion corresponding to $M$ still points to $R_i$ inside $R$. Consider all the components $\kappa = (P, M)$ corresponding to the four subsquares, expand their bitmap, and collect them in $\mathcal{K}^1$. Merge the partitions $\mathcal{P}_i$ to get $\mathcal{P}'$. If there is a terminal pair $(s, t)$ where $s$ is in $R_i$ and $t$ is in a different $R_j$,

```
Algorithm EuclideanSteinerForest
Input: Set of terminals $V$ in the plane, and set $\mathcal{D}$ of pairs of terminals
Output: A forest $F$ connecting pairs in $\mathcal{D}$

  1. Carry out the perturbation and scaling.
  2. Let $L$ be smallest power of two larger than $2n^2\epsilon^{-1}d$, where $d$ is the maximum distance
     of a pair.
  3. Perform a random dissection in the bounding box of side $L$.
  4. Place $m+1$ portals on each side of a dissection square, where $m$ is the smallest power
     of two larger than $4\epsilon^{-1}\log L$.
  5. Solve the base cases $T_R[\chi]$ for leaf dissection squares $R$:
     Go over all possible ways of connecting the portals and the center point.
  6. Populate the table $T_R[\chi]$ in increasing order of size for $R$:
     For any $\chi = (\mathcal{K}, \mathcal{P})$ corresponding to $R$ consisting of $R_1, \ldots, R_4$:
       (a) Go over all configurations $\chi_i = (\mathcal{K}_i, \mathcal{P}_i)$ corresponding to $R_i$.
       (b) Build $\mathcal{K}^1$ from the union of all components of $\mathcal{K}_i$ with expanded bitmaps.
       (c) Build $\mathcal{P}'$ from the union of $\mathcal{P}_i$.
       (d) If there is a terminal pair $(t_1, t_2)$ where $t_1 \in R_{i_1}$ and $t_2 \in R_{i_2}$ for $i_1 \neq i_2$,
             • If there is no bitmap in $\chi_{i_1}$ (or $\chi_{i_2}$) containing the cell containing $t_1$ (or $t_2$
               respectively), the configuration is bad.
             • Otherwise, merge the sets corresponding to the appropriate components in $\mathcal{P}'$.
       (e) Build $\mathcal{K}^2$ by merging components having the same portals, and make appropriate
           changes to $\mathcal{P}'$.
       (f) Build $\mathcal{K}^3$ by removing portals not on $\partial R$.
       (g) If any component with empty portal set has unsatisfied connectivity requirement
           in $\mathcal{P}'$, the current configurations are not consistent.
       (h) Build $\mathcal{K}^4$ by eliminating components with empty portal set.
       (i) If any bitmap contradicts the locality property, these configurations are not con-
           sistent.
       (j) If the configurations are consistent, update $T_R[\chi]$ with .
               $\min\left\{ T_R[\chi] + \sum_{i=1}^{4} T_{R_i}[\chi_i] \right\}.$
  7. Find the final solution among $T_R[\chi]$ where $R$ is the bounding box and $\chi$ has no unsat-
     isfied requirement.
  8. Construct the solution $F$ by recursively following the values from $T_R[\chi]$.
```

Figure 2: The algorithm for *Euclidean Steiner forest* problem.

there should be a component corresponding to each of these in $R_i$ and $R_j$, respectively. Otherwise, these configurations do not correspond to any (valid) subsolution. Merge the sets corresponding to these components in $\mathcal{P}'$: i.e., they have to be connected. Next merge any two components of $\mathcal{K}^1$ if they share a portal, and build $\mathcal{K}^2$. Further, make appropriate changes in $\mathcal{P}'$. Build $\mathcal{K}^3$ by removing from $\mathcal{K}^2$ all portals not on $\partial R$. Some of these components reach $\partial R$ and some do not, namely those with an empty portal set $P$. If there is any component with empty portal set that is not one partition set, we deem the configurations $\chi_i$ as *inconsistent*: in this case, some components that are required to be connected together do not reach the boundary. Otherwise, remove all the pairs in $\mathcal{K}^3$ with empty portal set to obtain $\mathcal{K}^4$. Now, if there is a cell of $R$ whose four constituent cells reach the boundary as more than one connected component, the configurations are not consistent either: this contradicts the property of Theorem 6. Finally, reduce the dimensions of the bitmaps to $\gamma \times \gamma$ such that a cell of the new bitmap acquires value one if and only if there is a one in one of the positions corresponding to the constituent cells in the original bitmap. Now, $\chi = (\mathcal{K}, \mathcal{P})$ is said to be consistent with the four configurations $\chi_1, \ldots, \chi_4$ if and only if $\mathcal{P}$ contains all the requirements of $\mathcal{P}'$, i.e., $\mathcal{P}'$ is a refinement of $\mathcal{P}$, and in addition, there exists a $\kappa = (P, M) \in \mathcal{K}$ for any $\kappa' = (P', M') \in \mathcal{K}'$ such that $P \subseteq P'$ and $M = M'$. In case these configurations are consistent, $T_R[\chi]$ will take the minimum of its current value and $\sum_i T_{R_i}[\chi_i]$. You can refer to Figure 2 for a summary.

## 4.2 Proof of correctness

Correctness follows from induction on the size of the square $R$ that all dynamic programming states have their intended value. In particular, we know that there is a near-optimal solution all whose subsolutions are compatible with one configuration. Hence, these will be computed correctly and give the final solution. More specifically the following claim holds for all DP states.

**Lemma 9.** A dynamic programming state $T_R[\chi]$ ends up having the minimum value corresponding to a solution $F$ of $R$, such that for any dissection square $R'$ which is a descendant of $R$ in the dissection tree, the subsolution $F \cap R'$ of $R'$ is compatible with a configuration $\chi'$ for $R'$.

Now, we are at the position to prove the main Theorem regarding the *Euclidean Steiner forest* problem.

**Proof of Theorem 1.** By Lemma 9, the proposed dynamic programming is sound and complete. There are $\Phi = O(\text{poly}(n))$ DP states. To solve each non-base state, we go over at most $\Phi^4$ child states and then perform a polynomial consistency check. Each base case state is computed in constant time. Hence, the total algorithm runs in time $O(\text{poly}(n))$. □

## 4.3 Highlights of the new ideas

Here, we point out the differences between our work and the previous work of [12]. Borradaile et al. use closed paths to identify the connected zones of the dissection square. These paths consist of vertical and horizontal lines and all the break-points are the corners of the cells. As part of their structural property, they prove that they can guarantee a solution in which these zones can be identified via paths whose total length is at most a constant $\eta$ times the perimeter of the square $R$. Then each path is represented by a chain of $\{1, 2, 3\}$ of length at most $O(\eta\gamma)$: the three values are used to denote moving one unit forward, or turning to the left or right. This results in a storage of $3^{O(\eta\gamma)}$ which is a constant parameter. Instead, we use a bitmap of size $\gamma \times \gamma$ to address this issue.

Each zone is represented by a bitmap that has an entry one in the cells of the zone. The bound that we obtain, $2^{\gamma^2}$, may be slightly worse than the previous work, however, a simpler structural property, namely the locality property, suffices as the proof of correctness. Borradaile et al. in contrast need a bound on the total length of the zone boundaries, as noted above.

In addition to the simplification made due to this change, both to the proof and the treatment of the dynamic programming, we simplify the proof further. Borradaile et al. charge the additions of $\sigma(C, F)$ to three different structures, and the argument is described and analyzed separately for each. We manage to perform a universal treatment and charging all the additions to the simplest of the three structures in their work. But this can be done only after showing $F^* - \mathcal{G}$ has a limited number of components. The proof is simple yet elegant—a weaker claim is proved in [12], but even the statement of the claim is hard to read.

## 5 Multiplicative prizes

We first tackle the *S-multiplicative prize-collecting Steiner forest* problem. Then, we will take a look at its asymmetric generalization. Finally, we show how the *multiplicative prize-collecting Steiner forest* problem can be reduced to $S$-MPCSF.

### 5.1 Collecting a fixed prize

Suppose we are given $S$, the amount of prize we should collect. Let OPT be the minimum cost of a forest $F$ that collects a prize of at least $S$, and suppose $Q \subseteq \mathcal{D}$ is the set of terminal pairs connected via $F$. We show how to find a forest with cost at most $(1+\epsilon)$OPT that collects a prize of at least $(1-\epsilon')S$. By the structural property, we know that there is a solution $F'$ connecting the same set of terminal pairs $Q$ whose cost is at most $(1+\epsilon)$OPT, yet it satisfies the conditions of Theorems 5 and 6. Round all the vertex weights down to the next integer multiple of $\theta = \epsilon'\sqrt{S}/2n$. In a connected component of $F'$ of total weight $A_i$ that lost a weight $a_i$ due to rounding, the lost prize is $A_i^2 - (A_i - a_i)^2 \le 2a_iA_i \le 2a_i\sqrt{S}$, because the total weight of the component is at most $\sqrt{S}$. Thus, $F'$ collects at least $S - 2n\theta\sqrt{S} \le (1-\epsilon')S$ from the rounded weights.

Each dynamic programming state consists of a dissection square $R$, a set of components $\mathcal{K}$, and a new parameter $\Pi$ which denotes the total prize collected inside $R$ by connecting the terminal pairs. Each element of $\mathcal{K}$—corresponding to a connected component in the subsolution—now has the form $\kappa = (P, \Sigma)$ where $P$ denotes the portals of $\kappa$, and $\Sigma$ is the total sum of the weights in $\kappa$. The DP is carried out in a fashion similar to that of [2]. The values of $\Sigma$ and $\Pi$ are easy to determine for the base cases. It is not difficult to update them, either. Whenever two components $\kappa_1 = (P_1, \Sigma_1)$ and $\kappa_2 = (P_2, \Sigma_2)$ merge in the DP, the sum $\Sigma$ for the new component is simply $\Sigma_1 + \Sigma_2$. Besides, the merge increases the $\Pi$ value of the DP state by $2\Sigma_1\Sigma_2$.

**Proof of Theorem 2.** The soundness and completeness is simple and is along the same lines as the proof of Theorem 1. Carrying out the above operation assumes the values of $\Sigma$ and $\Pi$ could be stored accurately. However, as they describe the dynamic programming states, their size should be sufficiently small or else the algorithm will not run in polynomial time. Here does the rounding help us. All values of $\Sigma$ are stored as multiples of $\theta$ and the values of $\Pi$ are stored as multiples of $\theta^2$. Notice that as we round the vertex weights at the beginning, throughout the algorithm the values of $\Sigma$ and $\Pi$ will be multiples of their respective units. Hence, no extra precision error will occur and we find the aforementioned solution. If at any time during the execution of the algorithm, the value of $\Sigma$ goes above $\sqrt{S}$, we truncate it to $\sqrt{S}$. Similarly, the value of $\Pi$ is not allowed to surpass $S$. This does not eliminate any solution, because at the point of truncation, the subsolution has already gathered sufficient prize. Hence, the range of $\Sigma$ is from zero up to $\sqrt{S}$, and this gives $\sqrt{S}/\theta = 2n/\epsilon'$ different values. Similarly for $\Pi$, there are at most $S/\theta^2 = 4n^2/\epsilon'^2$ options. There are at most

$$\Phi_1 = \left[O\left(m^{\lambda+1}\right) \cdot 2^{\gamma^2} \cdot 2n/\epsilon'\right]^{\lambda} \cdot 2^{\lambda^2} \cdot 4n^2/\epsilon'^2 = O\left(\text{poly}\left(n, \frac{1}{\epsilon'}\right)\right)$$

DP states for each square $R$. The running time is polynomial in $\Phi_1$ and the claim follows. $\square$

To start the algorithm, we need to guarantee the instance satisfies the conditions at the beginning of Section 2. See Appendix A.1 for details of how this is achieved.

## 5.2 The asymmetric prizes

The basic idea is to store two parameters $\Sigma^s$ and $\Sigma^t$ for each component of $\mathcal{K}$. These parameters store the total weight of the first and second type in the component, namely $\sum_i \phi_i^s$ and $\sum_i \phi_i^t$, respectively. The difficulty is that to collect a prize of $A = A^s A^t$ in a component, only one of the parameters $A^s$ or $A^t$ needs to be large. In particular, we cannot do a rounding with a precision like $\epsilon'\sqrt{A}/n$. It may even happen that $A^s$ is large in one component, whereas we have a large $A^t$ in another. In fact, we cannot store the values of the $\Sigma^s$ or $\Sigma^t$ as multiples of a fixed unit. To get around the problem, $\Sigma^s$ is stored as a pair $(v, x)$, where $v$ is a vertex of the graph and $x$ is an integer. Together they show that $\Sigma^s$ is $x \cdot \epsilon_1 \phi^s(v)/n^2$; the value of $\epsilon_1$ will be chosen later, and $v$ is supposed to be the vertex of largest type-one weight present in the component. A similar provision is made for $\Sigma^t$. Finally, the value of $\Pi$ is stored as a multiple of $\epsilon_2 A/n$; we will shortly pick the value of $\epsilon_2$.

Whenever $\Sigma_1^s = (v_1, x_1)$ and $\Sigma_2^s = (v_1, x_1)$ are added to give $\Sigma^s = (v, x)$, we do the calculation as follows: let $v$ be the vertex $v_1$ or $v_2$ that has the larger $\phi^s$ value, and then

$$x = \left\lfloor \frac{x_1\phi^s(v_1)/n^2 + x_2\phi^s(v_2)/n^2}{\epsilon_1\phi^s(v)/n^2} \right\rfloor.$$

**Proof of Theorem 4.** The precision error for $\Sigma^s = (v, x)$ is at most $n \cdot \epsilon_1\phi^s(v)/n^2 = \epsilon_1\phi^s(v)/n$, because there is an accumulation of at most $n$ rounding errors each of which has been less than $\epsilon_1\phi^s(v)/n^2$. Notice that if $\Sigma^s$ is stored in terms of the vertex $v$, it has to include $v$ and thus its type one weight is at least $\phi^s(v)$. Hence, the precision error is at most a $\epsilon_1/n$ multiplicative factor. Therefore, when we do a multiplication of $\Sigma^s\Sigma^t$ to get an addition to $\Pi$, the error is at most a multiplicative $2\epsilon_1/n$: $(1 - \epsilon_1/n)A^s(1 - \epsilon_1/n)A^t \geq (1 - 2\epsilon_1/n)A^sA^t$. Next a rounding error may happen to store the value in terms of $\epsilon_2 A/n$. Each $\Pi$ on the other hand is made up of at most $n$ addition terms, so the total error is at most $n(2\epsilon_1/n + \epsilon_2/n)A$. We pick $\epsilon_1 = \epsilon_2 = \epsilon'/3$ to conclude that the total error is bounded by $\epsilon' A$.

All the discussion applies to $\Sigma^t$ as well. Due to truncation and rounding, there are at most $n/\epsilon_2$ options for $\Pi$. And each $\Sigma^s$ (or $\Sigma^t$) has at most $n^2/\epsilon_1$ possibilities. Thus, the total number of DP states for each dissection square is $\Phi_2 = \text{poly}(n, 1/\epsilon')$. Therefore, we obtain a bicriteria approximation to the asymmetric variant of the problem. $\square$

## 5.3 The prize-collecting version: trade-off between penalty and forest cost

In the prize-collecting variant, we pay for the cost of the forest, and for the prizes not collected. If the total weight is $\Delta$, the prize not collected is $\Delta^2$ minus the collected prize. One difficulty here is to determine the correct range for the collected prize so that we can use the algorithm of Section 5.1. The trivial range is zero to $\Delta^2$. However, the rounding precision we pick for the penalties should also take into account the cost of the forest. If the cost of the intended solution is much smaller than $\Delta^2$, we cannot simply go with rounding errors like $\epsilon\Delta/n$. Otherwise, the error caused due to rounding the penalties will be too large compared to the solution value.

The trick is to find an estimate of the solution value, and then consider two cases depending on how the cost compares to the total penalty. Using a 3-approximation algorithm, we obtain a solution of value $\omega$. We are guaranteed that $\mathrm{OPT} \geq \omega/3$. If $\Delta^2 \leq \omega/3$, the optimum solution is to collect no prize at all. Otherwise, assume $\Delta^2 > \omega/3$. To beat the solution of value $\omega$, we should collect a prize of at least $\Delta^2 - \omega$.

We first consider the simpler case when $\omega/\Delta^2 > 1/n^2$: For an $\epsilon' > 0$ whose precise value will be fixed below, we use the algorithm of Section 5.1 to find a bicriteria $(1+\epsilon/2, 1-\epsilon')$-approximate solution for collecting a prize $S$; this is done for any $S$ which is a multiple of $\epsilon'\Delta^2$ in range $[(1-\epsilon')\Delta^2 - \omega, \Delta^2]$. We select the best one after adding the uncollected prize to each of these solutions. Suppose the optimal solution OPT collects a prize $S'$. Let $\mathrm{OPT}_f = \mathrm{OPT} - (\Delta^2 - S')$ be the length of the forest. Round $S'$ down to the next multiple of $\epsilon'\Delta^2$, say $S$. Fed with prize value $S$, the algorithm finds a solution that collects a prize of at least $(1-\epsilon')S$ with forest cost at most $(1+\epsilon/2)\mathrm{OPT}_f$.

**Claim 10.** The total cost of this solution is at most $(1+\epsilon)\mathrm{OPT}$ if $\epsilon' = \frac{\min\left(\frac{\epsilon}{3},1\right)}{6n^2}$.

**Proof.** The total cost of this solution is

$$\begin{aligned}
\left(1+\frac{\epsilon}{2}\right)\mathrm{OPT}_f + \left[\Delta^2 - (1-\epsilon')S\right] &\leq \left(1+\frac{\epsilon}{2}\right)\mathrm{OPT}_f + \left[\Delta^2 - (1-\epsilon')(1-\epsilon')S'\right] \\
&\leq \mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT}_f + (2\epsilon' + \epsilon'^2)S' \\
&= \mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT} + (2\epsilon' + \epsilon'^2)\frac{S'}{\mathrm{OPT}}\mathrm{OPT} \\
&\leq \mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT} + (2\epsilon' + \epsilon'^2)\frac{\Delta^2}{\mathrm{OPT}}\mathrm{OPT} \\
&\leq \mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT} + (2\epsilon' + \epsilon'^2)3n^2\mathrm{OPT} \qquad (4)\\
&\leq \mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT} + \frac{\epsilon}{2}\mathrm{OPT} \qquad (5)\\
&= (1+\epsilon)\mathrm{OPT},
\end{aligned}$$

where (4) follows from $\frac{\Delta^2}{\mathrm{OPT}} \leq \frac{n^2\omega}{\omega/3} = 3n^2$, and (5) uses the definition of $\epsilon'$. ☐

The other case, i.e., $\omega/\Delta^2 \leq 1/n^2$, is more challenging. Notice that in order to carry out the same procedure in this case, $\epsilon'$ may not be bounded by $1/\operatorname{poly}(n)$ and thus the running time may not be polynomial. The solution, however, has to collect almost all the prize. Thus, one of the connected components includes almost all the vertex weights. We set aside a subset $\mathcal{B}$ of vertices of large weight. The vertices of $\mathcal{B}$ have to be connected in the solution, or else the paid penalty will be too large. Then, dynamic programming proceeds by ignoring the effect of these vertices and only keeping tabs on how many vertices from $\mathcal{B}$ exist in each component. At the end, we only take into account the solutions that gather *all* the vertices of $\mathcal{B}$ in one component and compute the actual cost of those solutions and pick the best one. In the following, we provide the details of our method and prove its correctness.

Let $\mathcal{B}$ be the set of all vertices whose weight is larger than $n\omega/\Delta$.

**Lemma 11.** All the vertices of $\mathcal{B}$ are connected in the optimal solution.

**Proof.** There are at most $n$ components, so there is a component, say $\mathcal{C}$, whose total weight is not less than $\Delta/n$. We claim all the vertices of $\mathcal{B}$ are inside this component. The penalty paid by the optimal solution is at most $\omega \leq \Delta^2/n$. If there is any vertex of $\mathcal{B}$ outside $\mathcal{C}$, the penalty of the solution is more than $\Delta/n \cdot n\omega/\Delta = \omega$, yielding a contradiction. □

Next, we round up all the weights to the next multiple of $\theta = \epsilon'\omega/\Delta$ for vertices not in $\mathcal{B}$. Define $\mathrm{OPT}'$ as the optimal solution of the resulting instance. Let $\mathrm{OPT}_f$ be the length of the forest in OPT, and define $\mathrm{OPT}'_f$ similarly. Let $\mathrm{OPT}_\pi$ and $\mathrm{OPT}'_\pi$ denote the penalty paid by OPT and $\mathrm{OPT}'$, respectively. Assume that $\epsilon' \leq 1$.

**Lemma 12.** $\mathrm{OPT}'_\pi \leq \mathrm{OPT}_\pi + 12n\epsilon'\mathrm{OPT}$.

**Proof.** We recompute the penalties paid by OPT using the rounded weights. The pair $(s,t)$ not connected in OPT is either of the two kinds: (1) one of $s$ and $t$ is in $\mathcal{B}$; or (2) none of them is in $\mathcal{B}$. The total rounding error for the penalties of the first type is bounded by $n\Delta\theta$. There are at most $n^2$ pairs of the second type. Since the weights of these terminals are at most $n\omega/\Delta$, the error is not more than $n^2[2(n\omega/\Delta)\theta + \theta^2]$. Hence, the total error is at most

$$\begin{aligned}
n^2[2(n\omega/\Delta)\theta + \theta^2] + n\Delta\theta &\leq n^2\left[(\epsilon'^2 + 2n\epsilon')\frac{\omega^2}{\Delta^2}\right] + n\epsilon'\omega \\
&\leq n^2\left[3n\epsilon'\frac{\omega^2}{\Delta^2}\right] + n\epsilon'\omega && \text{because } \epsilon' \leq 1 \\
&= \left(3n^2\frac{\omega}{\Delta^2} + 1\right)n\epsilon'\omega \\
&\leq 4n\epsilon'\omega && \text{because } \frac{\omega}{\Delta^2} \leq \frac{1}{n^2},
\end{aligned}$$

which is no more than $12n\epsilon'\mathrm{OPT}$ as desired. □

Suppose we use a dynamic programming approach similar to the previous subsections to find the approximately minimum forest length for any specified collected prize amount; in particular, we obtain a bicriteria $(1+\epsilon/2, 1-\epsilon')$-approximate solution. During this process, we ignore the weights associated with vertices in $\mathcal{B}$. Consider a DP state $\chi = (\mathcal{K}, \Pi)$ corresponding to a dissection square $R$. Each component $\kappa \in \mathcal{K}$ looks like $(P, \Sigma, \mu)$: the new piece of information, $\mu$, is an integer number denoting the number of vertices of $\mathcal{B}$ inside $\kappa$. Extending the previous algorithm to populate the new DP table is simple. Finally, we look at all the configurations $\chi$ for the bounding box such that the $\mu$ value of one component is exactly $|\mathcal{B}|$ whereas it is zero for all other components. This guarantees that all elements of $\mathcal{B}$ are inside the former component and hence we can add up the penalties involving those vertices. Let $\mathcal{K} = \{\kappa_1, \kappa_2, \ldots, \kappa_q\}$ where $\kappa_i = (P_i, \Sigma_i)$, and let $\kappa_1$ be the component containing $\mathcal{B}$. The additional cost due to vertices of $\mathcal{B}$ is

$$\left(\sum_{v\in\mathcal{B}}\phi(v)\right)\cdot\left(\sum_{i=2}^{q}\Sigma_i\right).$$

Finally, we report the best solution corresponding to these configurations.

**Proof of Theorem 3.** Let us first see that the algorithm described runs in polynomial time. It is sufficient to bound the number of configurations. The new piece of information has at most $n$ possibilities. Further, $\Sigma \le \frac{n^2}{\epsilon'}\theta$ is always a multiple of $\theta$. Similarly, $\Pi$ will not exceed $\frac{n^4}{\epsilon'^2}\theta^2$ and is always a multiple of $\theta^2$.

We pick $\epsilon' = \frac{1}{24n}$. By Lemmas 11 and 12, the rounding does not increase the penalties paid by the optimal solution by more than $\epsilon/2\mathrm{OPT}$. We then utilize the algorithm described for $S$-MPCSF to find a solution of cost at most $(1+\epsilon/2)\mathrm{OPT}_f + \mathrm{OPT}_\pi + \epsilon/2\mathrm{OPT} \le (1+\epsilon)\mathrm{OPT}$. Finally, changing the weights back to the original values clearly does not increase the cost. □

## 6 Evidence for Hardness

So far PTASs for geometric problems in Euclidean plane including ours and those of Arora [2] and Mitchell [29] can be easily generalized for Euclidean $d$-dimensional space, for any constant $d > 2$. However we can prove the following theorem on the hardness of the problem for Euclidean $d$-dimensional space.

**Theorem 13.** If notorious densest $k$-subgraph is hard to approximate within a factor $O(n^{\frac{1}{d}})$ for some constant $d$, then for any $d' > 2d+1$, the $k$-forest problem in Euclidean $d'$-dimensional space is hard to approximate within a factor $O(n^{\frac{1}{2d}-\frac{1}{d'-1}})$.

**Proof.** Hajiaghayi and Jain [22] show that if densest $k$-subgraph is hard to approximate within a factor $O(n^{\frac{1}{d}})$, then the $k$-forest problem on stars is hard to approximate within a factor $O(n^{\frac{1}{2d}})$. On the other hand, Gupta [20] shows that a tree metric of size $n$ can be embedded into Euclidean $d'$-dimensional space with distortion in $O(n^{\frac{1}{d'-1}})$. Thus for any $d' > 2d+1$, we cannot obtain an approximation factor $o(n^{\frac{1}{2d}-\frac{1}{d'-1}})$ for $k$-forest in Euclidean $d'$-dimensional space, since otherwise by solving the problem in Euclidean $d'$-dimensional space, finding an Eulerian tour and shortcutting it, and finally embedding it back into the star, we can obtain a better approximation than $O(n^{\frac{1}{2d}})$, a contradiction. □

Note as mentioned above that, despite extensive study, the current best approximation factor for notorious densest $k$-subgraph is $O(n^{1/3-\epsilon})$ [16] and thus we do not expect to have any PTAS for $k$-forest in 8-dimensional Euclidean space.

Unlike the general cases of these problems, as far as PTASs for the case of Euclidean spaces are concerned, it seems *k-forest* and *prize-collecting Steiner forest* problems are essentially equivalent. Indeed in Lemma 15, we prove that any PTAS for *k-forest* results in a PTAS for *prize-collecting Steiner forest*, and we believe that any DP algorithm giving a PTAS for PCSF computes along its way the optimal solution to different $k$-forest instances.

Thus based on the evidences above, we do believe *Euclidean k-forest* and *Euclidean prize-collecting Steiner forest* have no PTASs in their general forms.

## 7 Conclusion

Besides presenting a simpler and correct analysis of the PTAS for the *Euclidean Steiner forest problem*, we showed how the approach can be generalized to solve multiplicative prize-collecting problems.

Generalizing our results to planar graphs, especially obtaining a PTAS for Steiner forest, has been a long-standing open problem in this field. The question was settled very recently by Bateni, Hajiaghayi and Marx [6]. While Borradaile, Klein and Kenyon-Mathieu [11] gave a PTAS for *Steiner tree* on planar graphs, a main ingredient of their algorithm is solving *Steiner tree* on graphs of bounded-treewidth. However in a sharp contrast, Gassner [18] showed recently that *Steiner forest* is NP-hard even on graphs of treewidth at most 3. Bateni et al. [6] gives a PTAS for the problem on graphs of bounded treewidth, and uses it to obtain a PTAS for planar and bounded-genus graphs.

Last but not least, obtaining any improvement over the approximation factor 2.54 in [22] for multiplicative prize-collecting Steiner forest in general graphs seems very interesting.

## References

[1] A. Archer, M. Bateni, M. Hajiaghayi, and H. Karloff, *Improved approximation algorithms for prize-collecting steiner tree and TSP*, in Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2009.

[2] S. Arora, *Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems*, Journal of the ACM, 45 (1998), pp. 753–782.

[3] E. Balas, *The prize collecting traveling salesman problem*, Networks, 19 (1989), pp. 621–636.

[4] M. Bateni and M. Hajiaghayi, *Assignment problem in content distribution networks: unsplittable hard-capacitated facility location*, in Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2009, pp. 805–814.

[5] M. Bateni and M. Hajiaghayi, *Euclidean prize-collecting Steiner forest*, in Proceedings of The 9th Latin American Theoretical Informatics Symposium (LATIN), 2010. to appear.

[6] M. Bateni, M. Hajiaghayi, and D. Marx, *Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth*, CoRR, abs/0911.5143 (2009).

[7] M. Bern and P. Plassmann, *The Steiner problem with edge lengths 1 and 2*, Information Processing Letters, 32 (1989), pp. 171–176.

[8] D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson, *A note on the prize collecting traveling salesman problem*, Mathematical Programming, 59 (1993), pp. 413–420.

[9] A. Blum, R. Ravi, and S. Vempala, *A constant-factor approximation algorithm for the k-MST problem*, Journal of Computer and System Sciences, 58 (1999), pp. 101–108.

[10] P. Bonsma, *Sparsest cuts and concurrent flows in product graphs*, Discrete Applied Mathematics, 136 (2004), pp. 173–182.

[11] G. Borradaile, C. Kenyon-Mathieu, and P. N. Klein, *A polynomial-time approximation scheme for Steiner tree in planar graphs*, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2007, pp. 1285–1294.

[12] G. Borradaile, P. N. Klein, and C. Mathieu, *A polynomial-time approximation scheme for Euclidean Steiner forest*, in Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2008, pp. 115–124. See the updated version available at http://www.math.uwaterloo.ca/ glencora/downloads/Steiner-forest-FOCS-update.pdf.

[13] C. Chekuri, S. Khanna, and F. B. Shepherd, *Edge-disjoint paths in planar graphs*, in Proceedings of the 45th Symposium on Foundations of Computer Science (FOCS), 2004, pp. 71–80.

[14] C. Chekuri, S. Khanna, and F. B. Shepherd, *Multicommodity flow, well-linked terminals, and routing problems*, in Proceedings of the thirty-seventh annual ACM symposium on Theory of computing (STOC), 2005, pp. 183–192.

[15] F. A. Chudak, T. Roughgarden, and D. P. Williamson, *Approximate k-MSTs and k-Steiner trees via the primal-dual method and lagrangean relaxation*, in Proceedings of the 8th International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2001, pp. 60–70.

[16] U. Feige, G. Kortsarz, and D. Peleg, *The dense k-subgraph problem*, Algorithmica, 29 (2001), pp. 410–421.

[17] N. Garg, *Saving an epsilon: a 2-approximation for the k-MST problem in graphs*, in Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), 2005, pp. 396–402.

[18] E. Gassner, *The Steiner subgraph problem revisited*, Tech. Rep. 2008-17, Graz University of Technology, September 2008.

[19] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM Journal on Computing, 24 (1995), pp. 296–317.

[20] A. Gupta, *Embedding tree metrics into low dimensional euclidean spaces*, Discrete & Computational Geometry, 24 (2000), pp. 105–116.

[21] A. Gupta, M. Hajiaghayi, V. Nagarajan, and R. Ravi, *Dial a ride from k-forest*, in Proceedings of the 15th Annual European Symposium on Algorithms (ESA), 2007, pp. 241–252.

[22] M. Hajiaghayi and K. Jain, *The prize-collecting generalized Steiner tree problem via a new approach of primal-dual schema*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 631–640.

[23] M. Hajiaghayi and A. Nasri, *Prize-collecting Steiner networks via iterative rounding*, in Proceedings of The 9th Latin American Theoretical Informatics Symposium (LATIN), 2010. to appear.

[24] K. Jain and V. V. Vazirani, *Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation*, Journal of the ACM, 48 (2001), pp. 274–296.

[25] D. S. Johnson, M. Minkoff, and S. Phillips, *The prize collecting Steiner tree problem: theory and practice*, in Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2000, pp. 760–769.

[26] P. Kolman and C. Scheideler, *Improved bounds for the unsplittable flow problem*, in Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), 2002, pp. 184–193.

[27] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, Journal of the ACM, 46 (1999), pp. 787–832.

[28] R. Madan, D. Shah, and O. Leveque, *Product multicommodity flow in wireless networks*, IEEE Transactions on Information Theory, 54 (2008), pp. 1460–1476.

[29] J. C. Mitchell, *Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems*, SIAM Journal on Computing, 28 (1995), pp. 1298–1309.

[30] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, *Spanning trees - short or small*, SIAM Journal on Discrete Mathematics, 9 (1996), pp. 178–200.

[31] G. Robins and A. Zelikovsky, *Tighter bounds for graph Steiner tree approximation*, SIAM Journal on Discrete Mathematics, 19 (2005), pp. 122–134.

[32] F. S. Salman, J. Cheriyan, R. Ravi, and S. Subramanian, *Approximating the single-sink link-installation problem in network design*, SIAM Journal on Optimization, 11 (2000), pp. 595–610.

[33] Y. Sharma, C. Swamy, and D. P. Williamson, *Approximation algorithms for prize collecting forest problems with submodular penalty functions*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA), 2007, pp. 1275–1284.

## A Deferred proofs and further discussion

**Proof of Lemma 7.** The proof of Theorem 5 (although not reproduced here) does not increase $\text{comp}(F \cap \mathcal{L})$. Hence, it suffices to prove the result for the forest $F$ specified at the beginning of Section 2. Observe that by (II), $F \cap \mathcal{L}$ consists merely of singleton points, because no Steiner point lies on a line $\ell \in \mathcal{L}$. Further notice that the $\ell_1$-length of $F$ is at most $\sqrt{2}\,\text{length}(F)$. Let $F^x$ be the total absolute distance $F$ travels in the $x$ direction. Since $x$-coordinate difference of any two consecutive *break-points* of $F$ is a multiple of 2 and the intersection with vertical lines of $\mathcal{L}$ occurs at coordinates of the form $(2i, y)$, the total number of intersections with vertical lines is exactly $F^x/2$. We can similarly argue for the intersections with horizontal lines, and finally conclude that $\text{comp}(F \cap \mathcal{L}) \leq \frac{\sqrt{2}}{2}\,\text{length}(F)$. □

**Proof of Lemma 9.** This is clearly true for the base cases of the DP since we go over all the possibilities. Next, take any configuration $\chi = (\mathcal{K}, \mathcal{P})$ corresponding to a non-leaf dissection square $R$, and suppose there is a subsolution $F$ with respect to $R$ compatible with $\chi$, such that any subsolution $F'$ formed by restricting $F$ to a dissection square $R'$ which is a descendant of $R$ is compatible with some configuration $\chi'$ of $R'$. Let $F_i$ be the subsolutions restricted to the subsquares

$R_i$. Each of them is thus compatible with an appropriate $\chi_i = (\mathcal{K}_i, \mathcal{P}_i)$. By inductive hypothesis, the dynamic programming states $T_{R_i}[\chi_i]$ have been correctly computed. Each connected component of $F$ not connected to $\partial R$ has to have all its terminal pairs satisfied. This is taken care of by checking the partition $\mathcal{P}'$: the terminals in components that do not advance in the dynamic table to $T_R[\chi]$ have their demands satisfied internally. In addition, the locality property for $F$ ensures the configurations will be consistent, and hence we perform an update of $T_R[\chi]$ from $T_{R_i}[\chi_i]$. This finishes the completeness proof.

Verifying that the update rule is sound is trivial. If the four configurations $\chi_i$ update $\chi$, then there exists a subsolution $F$ formed by the union of the corresponding subsolutions $F_i$, that is compatible with $\chi$. □

## A.1 The preliminary conditions for multiplicative prizes

In Section 2, we said that standard perturbation and scaling techniques allow us to assume with a cost increase of at most $O(\epsilon\text{OPT})$ that the bounding box of the instance has side length at most $n^2\epsilon^{-1}\text{OPT}$, while restricting all vertices and Steiner points to points of the form $(2i+1, 2j+1)$ for integers $i$ and $j$. The claim is based on the following two premises:

1. If $d$ is the maximum distance of a pair in $\mathcal{D}$, then $\text{OPT} \geq d$.
2. If $u$ and $v$ are farther than $n^2 d$, they cannot be connected in the optimum solution.

Using this, the instance can be broken up into disjoint subinstances and then the perturbation can be carried out. However, the first premise is false in the case of multiplicative prizes since not all the pairs need to be connected. Next we show how similar conditions can be guaranteed in this case.

The value of OPT can be *guessed* using binary search. To begin the search, we can get crude bounds of $\omega/n \leq \text{OPT} \leq \omega$, using simple approximation algorithms for the general cases of PCSF and $k$-forest.[5] Knowing OPT, we build a graph $G'$ on the vertices: there is an edge between $u$ and $v$ if and only if their distance is at most OPT. The diameter of each connected component is at most $n\text{OPT}$. We consider each of them separately, since two vertices in different components cannot be connected in the optimal solution.

The side length of the bounding box is at most $n\text{OPT}$. Scale the instance by $8\epsilon^{-1}$ and let $\text{OPT}' = 8\epsilon^{-1}\text{OPT}$ denote the new optimal value. Build a grid in the bounding by lines with equations $x = 2i$ and $y = 2j$ for integers $i, j$. Move each vertex and Steiner point to the closest point of the form $(2i+1, 2j+1)$. Notice that there are at most $n$ Steiner points. Assuming $\text{OPT} > 0$, the change in the solution value due to the perturbation is at most $2n \cdot 4 = 8n \leq \epsilon\text{OPT}'$. Hence, we can assume that

- the side length of the bounding box is at most $n\epsilon^{-1}\text{OPT}'$, and
- the vertices and Steiner points are at coordinates $(2i+1, 2j+1)$ for integers $i, j$.

## A.2 $k$-MST as a special case of $S$-MPCSF

Here we show that (even the symmetric) $S$-MPCSF is a generalization of the rooted $k$-MST problem (for which the best approximation guarantee is 2). Suppose we are given an instance $\mathcal{I}$ of the rooted

[5]The best known approximation algorithms known for these problems are 2.54 and $\min\{\sqrt{k}, \sqrt{n}\}$, respectively.

$k$-MST problem. It consists of a graph $G(V, E)$, edge lengths $c_e$, a root vertex $r$ and a number $k$. Suppose $r$ is not to be counted among the $k$ vertices. We build the new instance $\mathcal{I}'$ of the $S$-MPCSF problem as follows. The graph $G'$ is the same as $G$. The weights of all vertices are one, except for $r$ whose weight is $n^2$. Then, the goal will be to find the cheapest forest that gathers a prize of at least $S = (n^2 + k)^2 = n^4 + 2n^2k + k^2$.

**Theorem 14.** The instance $\mathcal{I}$ of the rooted $k$-MST problem is equivalent to the instance $\mathcal{I}'$ of the $S$-MPCSF problem.

**Proof.** As we noted in Subsection 1.2, in case of polynomially bounded integer weights, we can make sure the returned solution collects a prize of at least $S$ (without any approximation factor). This can be achieved by picking $\epsilon' < 1/S$.

Obviously, any tree connecting $k$ vertices to the root is translated to a forest that collects a prize of at least $S$. Let each vertex not spanned by the tree be a singleton component in the forest.

Finally, we claim that any solution of value $S$ or higher translates to a solution of value at least $k$ for the original instance. The resulting tree is just the component of the forest containing the root vertex. Suppose for the sake of reaching a contradiction that the component spans $k' < k$ non-root vertices. The total prize collected is at most

$$\begin{aligned}(n^2 + k')^2 + (n - k' - 1)^2 &< n^4 + 2n^2k' + k'^2 + n^2 \\ &= S + 2n^2(k' - k) + k'^2 + n^2 - k^2 \\ &< S + 2n^2(k' - k + 1) \\ &\le S,\end{aligned}$$

yielding a contradiction, and proving the supposition is false. □

## A.3 PCSF vs. $k$-forest

**Lemma 15.** An $\alpha$-approximation algorithm for the *$k$-forest* problem gives an $\alpha(1 + \epsilon)$-approximation algorithm for the *prize-collecting Steiner forest* problem, for any constant $\epsilon > 0$.

**Proof.** We show how to approximate a PCSF instance $\mathcal{I}$ by invoking several (polynomially many) instances $\mathcal{I}'$ of the $k$-forest problem. Obtain an estimate $\omega$ for $\mathcal{I}$, such that $\frac{\omega}{3} \le \mathrm{OPT} \le \omega$ using a general-case 3-approximation algorithm. Let $\pi_i$ be the penalty of the pair $i$ in $\mathcal{I}$. Without loss of generality, we can assume that $\pi_i \le 2\omega$ for any pair $i$. Let $\theta = \epsilon\omega/3n$. Place $p_i = \lfloor \frac{\pi_i}{\theta} \rfloor$ copies of the pair $i$ in $\mathcal{I}'$. Find an $\alpha$-approximate solution to the resulting $k$-forest instance for every value of $0 \le k \le n'$, where $n'$ is the number of pairs in $\mathcal{I}'$. Compute the PCSF value for each of these solutions and report the best one.

We show that at least one of these candidate solutions is good. Let $\mathrm{OPT}_f$ and $\mathrm{OPT}_\pi$ be the length of the forest and the paid penalty of the optimal solution, respectively. Suppose OPT connects a subset of terminal pairs $Q$. Then, $\mathrm{OPT}_\pi = \sum_{i \notin Q} \pi_i$. Focus on the candidate solution with $k = \sum_{i \in Q} \lfloor \pi_i/\theta \rfloor$. The length of the corresponding $k$-forest instance is at most $\mathrm{OPT}_f$, because a possible solution is that of connecting the copies of $Q$. To compute the PCSF value, we add the penalty of pairs in $Q$ that are not connected using this tree. We can assume either all or no copies of each pair is connected. The number of pairs not connected is at most $n' - k$, and their penalties sum to no more than

$$\begin{aligned}\sum_{i \text{ not connected}} \pi_i &\le \sum_{i \text{ not connected}} (p_i + 1)\theta \\ &\le \sum_{i \text{ not connected}} p_i\theta + n\theta \\ &\le (n' - k)\theta + n\theta \\ &\le \mathrm{OPT}_\pi + n\theta \\ &= \mathrm{OPT}_\pi + \epsilon\omega/3 \\ &\le \mathrm{OPT}_\pi + \epsilon\mathrm{OPT}.\end{aligned}$$

Thus, the PCSF value of the best candidate solution is at most $\alpha\mathrm{OPT}_f + \mathrm{OPT}_\pi + \epsilon\mathrm{OPT} \le \alpha(1+\epsilon)\mathrm{OPT}$. It remains to show the instances $\mathcal{I}'$ have polynomial size. Since $\pi \le 2\omega$, each pair $i$ will have $p_i \le 6n\epsilon^{-1}$ copies. Hence, $\mathcal{I}'$ has polynomial size and we can use the approximation algorithm for the $k$-forest. $\square$