Prize-collecting Network Design on Planar Graphs

MohammadHossein Bateni*    MohammadTaghi Hajiaghayi†    Dániel Marx‡

Abstract

In this paper, we reduce Prize-Collecting Steiner TSP (PCTSP), Prize-Collecting Stroll (PCS), Prize-Collecting Steiner Tree (PCST), Prize-Collecting Steiner Forest (PCSF) and more generally Submodular Prize-Collecting Steiner Forest (SPCSF) on planar graphs (and more generally bounded-genus graphs) to the same problems on graphs of bounded treewidth. More precisely, we show any $\alpha$-approximation algorithm for these problems on graphs of bounded treewidth gives an $(\alpha + \epsilon)$-approximation algorithm for these problems on planar graphs (and more generally bounded-genus graphs), for any constant $\epsilon > 0$. Since PCS, PCTSP, and PCST can be solved exactly on graphs of bounded treewidth using dynamic programming, we obtain PTASs for these problems on planar graphs and bounded-genus graphs. In contrast, we show PCSF is APX-hard to approximate on series-parallel graphs, which are planar graphs of treewidth at most 2. This result is interesting on its own because it gives the first provable hardness separation between prize-collecting and non-prize-collecting (regular) versions of the problems: regular Steiner Forest is known to be polynomially solvable on series-parallel graphs and admits a PTAS on graphs of bounded treewidth. An analogous hardness result can be shown for Euclidean PCSF. This ends the common belief that prize-collecting variants should not add any new hardness to the problems.

*Department of Computer Science, Princeton University, Princeton, NJ 08540; Email: mbateni@cs.princeton.edu. The author is also with Center for Computational Intractability, Princeton, NJ 08540. He 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.
‡The Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel; dmarx@cs.bme.hu. He is supported by ERC Advanced Grant DMMCA.
1 Introduction

Prize-collecting problems involve situations where there are various demands that desire to be “served” by some structure and we must find the structure of lowest cost to accomplish this. However, if some of the demands are too expensive to serve, then we can refuse to serve them and instead pay a penalty. In particular, prize-collecting Steiner problems are well-known network design problems with several applications in expanding telecommunications networks (see for example [46, 52]), cost sharing, and Lagrangian relaxation techniques (see e.g. [45, 21]). A general form of these problems is the Prize-Collecting Steiner Forest (PCSF) problem\footnote{In the literature, this problem is also called Prize-Collecting Generalized Steiner Tree.}: given a network (graph) $G = (V, E)$, a set of source-sink pairs $\mathcal{D} = \{\{s_1, t_1\}, \{s_2, t_2\}, \ldots, \{s_k, t_k\}\}$, a non-negative cost function $c : E \rightarrow \mathbb{R}_+$, and a non-negative penalty function $\pi : 2^V \rightarrow \mathbb{R}_+$, our goal is a minimum-cost way of installing (buying) a set of links (edges) and paying the penalty for those pairs which are not connected via installed links. We also consider the problem with a general penalty function called Submodular Prize-Collecting Steiner Forest (SPCSF), in which the penalty function $\pi$ is a monotone non-negative submodular function\footnote{Source-sink pairs are sometimes called demands.} of all unsatisfied pairs. In PCSF when all penalties are $\infty$, the problem is the classic APX-hard Steiner Forest problem, for which the best known approximation ratio is $2 - \frac{\Delta}{2}$ ($\Delta$ is the number of nodes of the graph) due to Agrawal, Klein, and Ravi [2] (see also [35] for a more general result and a simpler analysis). The case of Prize-Collecting Steiner Forest problem in which all sinks are identical is the classic (rooted) Prize-Collecting Steiner Tree (PCST) problem. In the unrooted version of this problem, there is no specific sink (root) and the goal is to find a tree connecting some sources and pay the penalty for the rest of them. We also study two variants of (unrooted) Prize-Collecting Steiner Tree, Prize-collecting TSP (PCTSP) and Prize-collecting Stroll (PCS), in which the set of edges should form a cycle and a path (in order) instead of a tree. When in addition all penalties are $\infty$ in these prize-collecting problems, we have classic APX-hard problems Steiner Tree, TSP and Stroll (Path TSP) for which the best approximation factors in order are $1.38$ [16], $\frac{3}{2}$ [20], and $\frac{3}{2}$ [42].

In network design, planarity is a natural restriction since in practical scenarios of physical networking, with cable or fiber embedded in the ground, crossings are rare or nonexistent. Thus obtaining algorithms with better approximation factors are highly desirable in this case. In many cases, approximation algorithms for planar graphs is based on reducing the problem to bounded treewidth instances such that the optimum changes only by a small term. This idea goes back to the classical work of Baker [9] and have been applied successfully several times in various contexts. The algorithmic and graph-theoretic properties of treewidth are intensively studied and a well-understood dynamic programming technique can solve NP-hard problems on bounded treewidth graphs. Our goal is to understand how far this paradigm can be pushed: what are the most general problems that can be solved this way. In particular, we want to understand the applicability of this technique to prize-collecting variants of standard optimization problems.

TSP, Steiner Tree, and Steiner Forest all have been considered extensively on planar graphs. Indeed all these problems remain hard even on planar graphs [29]. However obtaining a PTAS
for each of these problems remained a very important open problem for several years. Grigni, Koutsoupias, and Papadimitriou [36] obtained the first PTAS for TSP on unweighted planar graphs in 1995 which later has been generalized to weighted planar graphs [6] (and improved to linear time [47]). Obtaining a PTAS for Steiner Tree on planar graphs remained elusive for almost 12 years until 2007 when Borradaile, Klein and Mathieu [15] obtained the first PTAS for Steiner Tree on planar graphs using a revolutionary technique of contraction decomposition and building spanners and posed obtaining a PTAS for Steiner Forest in planar graphs as the main open problem. Bateni, Hajiaghayi and Marx [12] very recently solved this open problem using a new primal-dual technique for building spanners and obtaining PTASs by reducing the problem to bounded treewidth graphs. Note that the Steiner Forest problem already shows signs of the reduction to bounded treewidth paradigm breaking down: surprisingly, Steiner Forest turns out to be NP-hard even on graphs of treewidth 3. However, [12] gets around this problem by using a PTAS on bounded treewidth graph instead of an exact algorithm.

Obtaining PTASs for prize-collecting versions of these problems remained a main open problem (see [12, 11]). It is not obvious how to generalize the reduction to bounded treewidth for these problems, and in particular new techniques are needed for handling penalties before building a spanner. In this paper, we resolve these open problems for all three of PCST, PCTSP, PCSF, and even more generally, for SPCSF, by reducing these problems on planar graphs to the same problems on graphs of bounded treewidth. More precisely we show any $\alpha$-approximation algorithm for these problems on graphs of bounded treewidth gives a ($\alpha + \epsilon$)-approximation algorithm for these problems on planar graphs and bounded-genus graphs, for any constant $\epsilon > 0$. Therefore, we demonstrate that the technique of reduction to bounded treewidth works even for very general version of problems involving prizes. Since PCST and PCTSP can be solved exactly on graphs of bounded treewidth using standard dynamic programming techniques (as we discuss later in the paper), we immediately obtain PTASs for PCST and PCTSP on planar graphs (the same holds for PCS as well). In contrast, we show that PCSF is APX-hard already on series-parallel graphs, which are planar graphs of treewidth at most 2, ruling out any hope for a PTAS for planar PCSF. This result is interesting on its own, since it gives the first provable hardness separation between prize-collecting and non-prize-collecting (regular) versions of the problems: regular Steiner Forest is known to be polynomially solvable on series-parallel graphs and admits a PTAS on graphs of bounded treewidth. since Steiner Forest on series-parallel graphs is polynomially solvable and more generally on graphs of bounded treewidth admits a PTAS [12]. An analogous hardness result can be given for Euclidean PCSF when the vertices of the input graph are points in the Euclidean plane and the lengths are Euclidean distances (which answers an open problem in [11]). This ends the common belief that prize-collecting variants should not add any new hardness to the problems.

**Related work.** PCST and PCTSP are two of the classic optimization problems with a large impact, both in theory and practice. At AT&T, PCST code has been used in large-scale studies in access network design, both as described in Johnson, Minkoff and Phillips [46], and another unpublished applied work by Archer at al. The impact of PCTSP within approximation algorithms is also far-reaching. In particular PCTSP is a Lagrangian relaxation of the $k$-MST problem, which asks for the minimum-cost tree spanning at least $k$ nodes, and has used in a sequence of papers ([30, 8, 22, 7]) culminating in a 2-approximation algorithm for $k$-MST by Garg [31]. PCTSP has also been used to improve the approximation ratio and running time of algorithms for the Minimum Latency problem ([5, 18]). The first approximation algorithms for the PCST and PCTSP problems were given by Bienstock et al. [13], although the PCTSP had been introduced earlier by
Balas [10]. Bienstock et al. achieved a factor of 3 for PCST and 2.5 for PCTSP by rounding the optimal solution to a linear programming (LP) relaxation. Later, Goemans and Williamson [34] constructed primal-dual algorithms using the same LP relaxation to obtain a 2-approximation for both problems, building on work of Agrawal, Klein and Ravi [2]. Chaudhuri et al. modified the Goemans-Williamson algorithm to achieve a 2-approximation algorithm for PCS [18]. Improving over the approximation factor 2 of Goemans and Williamson for PCST and PCTSP was a long-standing open problem for 17 years until recently that Archer, Bateni, Hajiaghayi, and Karloff [4] obtain constant factors strictly better than 2 (≈ 1.99) for both problems, and for PCS as well. More recently Goemans combined some ideas of [4] with others from [32] to improve the ratio for PCTSP below 1.915 [33].

The general form of the Prize-Collecting Steiner Forest problem first has been formulated by Hajiaghayi and Jain [38]. They showed how by using a primal-dual method to a novel integer programming formulation of the problem with doubly-exponential variables, we can obtain a 3-approximation algorithm for the problem. In addition, they show that the factor 3 in the analysis of their algorithm is tight. However they show how a direct randomized LP-rounding algorithm with approximation factor 2.54 can be obtained for this problem. Their approach has been generalized by Sharma, Swamy, and Williamson [53] for network design problems where violated arbitrary 0-1 connectivity constraints are allowed in exchange for a more general penalty function. Hajiaghayi and Nasri [40] show factor 3 for Prize-Collecting Steiner Forest can also be obtained via an iterative rounding approach, first introduced by Jain [44], and indeed factor 3 is the best one can hope via this approach. The work of Hajiaghayi and Jain has also motivated a game-theoretic version of the problem considered by Gupta et al. [37]. Very recently, Hajiaghayi et al. [39] obtain a 2.54 approximation algorithm for the more general problem SPCS. Aforementioned, our reduction from planar graphs to graphs of bounded treewidth works even for SPCS. It is worth mentioning optimizing a submodular function, a discrete analog of a convex function, which also demonstrates economy of scale is a central and very general problem in combinatorial optimization and has been subject of a thorough study in the literature in many important settings including cuts in graphs [43, 35, 49], plant location problems [24, 23], rank function of matroids [26], set covering problems [27], and certain restricted satisfiability problems [41, 28].

Remark Subsequent to, and independent of, our work, Chekuri et al. [19] obtain a subset of our results including a reduction for prize-collecting Steiner tree and prize-collecting Steiner forest from planar graphs to graphs of bounded treewidth (i.e., a weaker version of our Theorem 1, albeit with different techniques) which leads to a PTAS for planar prize-collecting Steiner tree. The hardness results though are unique to our work.

2 Contributions

We first formally define the most general problem studied in this paper. An instance of Submodular Prize-Collecting Steiner Forest SPCS is described by a triple $(G, \mathcal{D}, \pi)$ where $G$ is a undirected weighted graph, $\mathcal{D}$ is a set of $d_i = \{s_i, t_i\}$ demand pairs, and $\pi : 2^\mathcal{D} \rightarrow \mathbb{R}^+$ is a monotone nonnegative submodular penalty function. A demand $d = \{s, t\}$ is satisfied by a subgraph $F$ if and only if $s, t$ are connected in $F$. If a forest $F$ satisfies a subset $\mathcal{D}^{sat}$ of the demands, its cost is defined as $\text{cost}(F) := \text{length}(F) + \pi(\mathcal{D}^{unsat})$, where $\text{length}(F)$ is a shorthand for the total length of all edges in $F$, and $\mathcal{D}^{unsat} := \mathcal{D} \setminus \mathcal{D}^{sat}$ denotes the subset of unsatisfied demands.
We similarly define SPCTSP, SPCS and SPCST that are submodular prize-collecting variants of Travelling Salesman Problem, Stroll and Steiner Tree, respectively. The instance is represented by $(G, \mathcal{D}, \pi)$ where all the demands $d = \{s, t\} \in \mathcal{D}$ share a common root vertex $r \in V(G)$.\footnote{The problems may be more naturally defined with single-vertex demands rather demand pairs; having such a formulation, we can guess one vertex of the solution, designate it as the root and obtain the rooted formulation as defined in this paper.} A solution $F$ is a TSP (stroll or Steiner tree, respectively) for a subset of demands, say $\mathcal{D}^{\text{sat}} \subseteq \mathcal{D}$. The cost is then $\text{cost}(F) := \text{length}(F) + \pi(\mathcal{D}^{\text{unsat}})$, where $\mathcal{D}^{\text{unsat}} := \mathcal{D} \setminus \mathcal{D}^{\text{sat}}$.

We first show that Submodular Prize-Collecting Steiner Forest on planar graphs (or more generally, bounded-genus graphs) is almost equivalent to that on graphs of bounded-treewidth; refer to Appendix A for definitions regarding the treewidth and bounded-treewidth graphs as well as bounded-genus graphs. In particular, were we able to give a PTAS for SPCSF on graphs of bounded treewidth, we would readily have a PTAS for SPCSF on bounded-genus graphs. In the rest of the paper, we focus on planar graphs. All the algorithms and analyses can be extended with minor modifications to work for bounded-genus graphs.

**Theorem 1.** For any given constant $\epsilon > 0$, an $\alpha$-approximation algorithm for SPCSF on graphs of bounded treewidth gives a $(\alpha + \epsilon)$-approximation algorithm for SPCSF on planar graphs.

The core of the reduction is based on a prize-collecting clustering technique that was first implicitly used in [4] and later developed in [12]. In this work, the clustering technique is generalized as follows: First, we need to extend the ideas to work for prize-collecting variants of Steiner network problems. This can indeed make the problem provably harder; see Theorem 3. The original prize-collecting clustering associates a potential value to each node and grows the corresponding clusters consuming these potentials. However, in order to extend it to the prize-collecting setting, we consider source-sink potentials. This means that there is some interaction between the potentials of different nodes. Secondly, we consider submodular penalty functions that model even more interaction between the demands. The extended prize-collecting clustering procedure has two phases. In the first phase, we have a source-sink moat-growing algorithm, and in the second phase, we have a single-node potential moat-growing like [12].

Section 3 is devoted to the formal proof Theorem 1. The algorithm starts with a constant-approximate solution $F^1$, say, obtained using Hajiaghayi et al. [39] who prove a 3-approximation for SPCSF on general graphs. The forest $F^1$ satisfies a subset of demands, and we know the total penalty of unsatisfied demands is bounded. The algorithm then tries to satisfy more demands by constructing a forest $F^2 \supseteq F^1$ whose length is bounded; see RESTRICTDEMANDS in Section 3.2. This step heavily uses a submodular prize-collecting clustering algorithm\footnote{The algorithm bears some similarity to the primal-dual moat-growing algorithms for the Steiner network problems. One key difference is that we do not have a primal LP. We have an LP similar to the dual linear programs used in such algorithms, and we use a notion of potential as a substitute for the lack of the primal LP. The potentials, among other things, play the role of an upper bound for the value of the dual LP.} introduced in Section 3.1. At the end of this step, we can assume that the near-optimal solution does not satisfy the demands which are unsatisfied in $F^2$. Submodularity poses several difficulties in proving this property: ideally, we want to say that the cost paid by the optimal solution to satisfy these demands is significantly more than their penalty value. Surprisingly, this is not true. Nevertheless, we can prove that the marginal cost of the demands satisfied in the near-optimal solution but not in $F^2$ can be charged to the cost the near-optimal solution pays in order to satisfy them. The next step of the reduction is to build a forest $F^3 \supseteq F^2$ of bounded length that may connect several components of $F^2$.
together; see Section 3.3. This is done by assigning to each component of $F^2$ a potential proportional to its length, and then running a prize-collecting clustering similar to that of [12]. This guarantees that the near-optimal solution does not need to connect different components of $F^3$ to each other. The implication is that we can construct a spanner (see [12, 15, 47]) out of each component of $F^3$ separately from the others. In the previous work [12], we could solve each of the subinstances independently, however, the penalty interaction originating from the submodular penalty function in the current work does not allow us to solve each subinstance completely independently. Instead, we say that the forest of the near-optimal solution on each subinstance is independent of the others.

After constructing the spanner graph $F^4$, we invoke a generalization of the shifting idea of Baker [9] due to [25, 47]. Paying a cost of at most $\epsilon \cdot \text{OPT}$, we end up with a graph of bounded treewidth.

Since bounded-treewidth graphs bear some similarity to trees, several tools have been developed for solving optimization problems on them. Standard techniques, see Appendix B, allow us to obtain PTASs for several Steiner network problems on graphs of bounded treewidth.

**Theorem 2.** $\text{PCST}, \text{PCS}$ and $\text{PCTSP}$ admit PTASs on bounded-treewidth graphs.

In Section 4 we show how this results in PTASs for the above problems on planar graphs. In particular, this is simple for PCST since it is a special case of SPCS. For the other two problems, however, refer to the discussion in Section 4.

In contrast, we show Prize-Collecting Steiner Forest is APX-hard, even on planar graphs of treewidth at least two; Hajiaghayi and Jain show the problem can be solved in polynomial on tree metrics [38].

**Theorem 3.** $\text{PCSF}$ is APX-hard on (1) planar graphs of treewidth two and on (2) the two-dimensional Euclidean metric.

This is done via a reduction from Bounded-Degree Vertex Cover in Appendix 5. Indeed, the result shows that Submodular Prize-Collecting Steiner Tree (the version of the problem when the solution has to be a connected tree instead of a forest) is also APX-hard. This implies the hardness of PCSF originates from the interaction between the penalties of terminals rather than from the different components of the solution.

Surprisingly, the hardness also works for Euclidean metrics, answering an open question raised in [11]. This is a very rare instance where a natural network optimization problem is APX-hard on the two-dimensional Euclidean plane.

Theorem 3 means that planar PCSF reaches a level of complexity where even though reduction to bounded treewidth instances works, it does not give us a PTAS for the problem (in fact, no PTAS exists unless P = NP). However, the treewidth reduction approach can be still useful for obtaining constant factor approximations for planar graphs better than the factor 2.54 algorithm of [38] for general graphs. Theorem 1 show that beating the 2.54 factor on bounded treewidth graphs would immediately imply the same for planar graphs. We pose it as an open question whether this is indeed possible for PCSF.

### 3 Reduction to bounded-treewidth case

This section focuses on proving Theorem 1. In fact, we prove a stronger version of the theorem, that is necessary for obtaining PTASs for PCST, PCTSP, and PCS. We reduce an instance $(G, D, \pi)$ of SPCS to an instance $(H, D, \pi')$ where $H$ has bounded treewidth and $\pi'$ has a structure similar
to $\pi$; in particular, for some $D_{\text{unsat}} \subseteq D$ we define $\pi'(D) := \pi(D \cup D_{\text{unsat}})$ for all $D \subseteq D$. Notice that if $\pi$ is submodular, then so is $\pi'$. Moreover, if $\pi$ models a PCSF instance, i.e., $\pi$ is an additive function, then $\pi'(D) - \pi'(\emptyset)$ models a PCSF instance, too. In fact, $\pi'(D)$ is an additive function that is shifted with a fixed amount $\pi'(\emptyset)$. Same condition holds for PCST, PCTSP and PCS. Therefore, after reducing a PCST instance, we are left with a PCST instance—rather than an SPCSF one—on a bounded-treewidth graph.

The proof has three steps:

1. We start with an instance $(G, D, \pi)$ of SPCSF. We first take out a subset, say $D_{\text{unsat}}$, of demands whose cost of satisfying is too much compared to their penalties. Thus, we can focus on the remaining demands, say $D_{\text{sat}} := D \setminus D_{\text{unsat}}$.

2. Afterwards, we partition the remaining demands $D_{\text{sat}}$ into $D_1, D_2, \ldots, D_p$ such that, roughly speaking, SPCSF can be solved separately on each of the demand sets without increasing the total cost substantially.

3. Finally, we build a spanner for each demand set $D_i$, and use similar ideas as in [12] to reduce the problem to bounded-treewidth graphs.

The first step is carried out in the following theorem. The proof appears in Section 3.2, and uses a submodular prize-collecting clustering technique introduced in Section 3.1. This step allows us to focus on only a subset $D_{\text{sat}}$ of demands, and ignore the rest of the demands. The additional cost due to this is only $\epsilon \cdot \text{OPT}$.

**Theorem 4.** Given an instance $(G, D, \pi)$ of SPCSF (or SPCTSP or SPCS) and a parameter $\epsilon > 0$, we can construct in polynomial time a subgraph $F$ of $G$, satisfying only a subset $D_{\text{sat}} \subseteq D$ of demands, in effect leaving $D_{\text{unsat}} := D \setminus D_{\text{sat}}$ unsatisfied, such that

1. $\text{length}(F) \leq (6\epsilon^{-1} + 3) \cdot \text{OPT}$, and
2. the optimum of $(G, D_{\text{sat}}, \pi')$ is at most $(1 + \epsilon) \cdot \text{OPT}$ where $\pi'(D) := \pi(D \cup D_{\text{unsat}})$ is defined for $D \subseteq D_{\text{sat}}$.

At this point, we have a constant-approximate solution satisfying all the (remaining) demands. The second step is a generalization and extension of the work in [12]. We are trying to break the instance into smaller pieces. The solution to each piece is almost independent of the others, i.e., there is little interaction between them. The following theorem is proved in Section 3.3.

**Theorem 5.** Given are an instance $(G, D, \pi)$ of SPCSF, a forest $F$ satisfying all the demands, and a parameter $\epsilon > 0$. We can compute in polynomial time a set of trees $\{\hat{T}_1, \ldots, \hat{T}_k\}$, and a partition of demands $\{D_1, \ldots, D_k\}$, with the following properties.

1. All the demands are covered, i.e., $D = \bigcup_{i=1}^k D_i$.
2. The tree $\hat{T}_i$ spans all the terminals in $D_i$.
3. The total length of the trees $\hat{T}_i$ is within a constant factor of the length of $F$, i.e., $\sum_{i=1}^k \text{length}(\hat{T}_i) \leq (\frac{2}{\epsilon} + 1) \cdot \text{length}(F)$.
4. Let \( D^* \) be the subset of demands satisfied by \( \text{OPT} \). Define \( D^*_i := D^* \cap D_i \), and denote by \( \text{SteinerForest}(G, D) \) the length of a minimum Steiner forest of \( G \) satisfying the demands \( D \). We have \( \sum_i \text{SteinerForest}(G, D^*_i) \leq (1 + \epsilon) \text{SteinerForest}(G, D^*) \).

The final step is very similar to the spanner construction of [12, 15]. Since it has been extensively covered in those works, we defer the details to the full version of the paper.

Now we show how the above theorems imply the main theorem of the paper.

Proof of Theorem 1. Start with an instance \((G, D, \pi)\) of SPCSF. Without loss of generality we present an approximation guarantee of \( \alpha + O(1)\epsilon \). Find \( F, D_{\text{sat}} \) and \( D_{\text{unsat}} \) from applying Theorem 4 on \((G, D, \pi)\). We know that \( F \) satisfies \( D_{\text{sat}} \) and \( \text{length}(F) = O(\text{OPT}) \). Moreover, \( \text{OPT}_{D_{\text{sat}}}(G) \leq \text{OPT} \). Define \( \pi^+(D) := \pi(D \cup D_{\text{unsat}}) \) for all \( D \subseteq D \). Clearly the optimal solution of \((G, D_{\text{sat}}, \pi^+)\) costs no more than \((1 + \epsilon)\text{OPT}\). Pick \( \epsilon' < \epsilon \cdot \text{length}(F)/\text{OPT} \) and feed \((G, D_{\text{sat}}, \pi^+)\) along with \( F \) and \( \epsilon' \) to Theorem 5, in order to obtain \( D_i \)’s and \( T_i \)’s for \( i = 1, \ldots, k \). We have \( \sum_i \text{length}(T_i) = O(\text{length}(F)) = O(\text{OPT}) \) since \( \epsilon' \) is a constant. In addition, the theorem guarantees a near-optimal solution \( \text{OPT}^+ \) of cost at most \((1 + 2\epsilon)\text{OPT}\) that does not use the connectivity of different components \( D_i \) and \( D_i' \) for \( i, i' \in \{1, \ldots, k\} : i \neq i' \). This ensures that the spanner construction gives us a graph \( G^+ \) (of total length \( O(\text{OPT}) \)) that approximate the forest of the solution within a \( 1 + \epsilon \) factor. Thus, the optimal solution of \((G^+, D_{\text{sat}}, \pi^+)\) costs at most \((1 + \epsilon)(1 + 2\epsilon)\text{OPT} = 1 + O(1)\epsilon)\text{OPT} \). Since the total length of the graph \( G^+ \) is within \( O(\text{OPT}) \), we can use the decomposition theorem of [25] to reduce the problem to bounded-treewidth graphs with an increase of \( \epsilon \text{OPT} \) in the solution cost. The reduced instance is solved via the \( \alpha \)-approximation algorithm, and we finally get an approximation ratio of \( \alpha + O(\epsilon) \).

\[ \square \]

3.1 Submodular prize-collecting clustering

First we present and analyze a primal-dual algorithm for SPCSF, and later we see how this algorithm can be used to achieve the goal of identifying and removing certain demands from the optimal solution such that the additional penalty is negligible.

Consider an instance \((G(V, E), D, \pi)\) of the SPCSF. A set \( S \subseteq V \) is said to cut a demand \( d = \{s, t\} \) if and only if \( |S \cap d| = 1 \). We denote this by the short-hand \( d \odot S \), and say the demand \( d \) crosses the set \( S \). In the linear program (1)–(3), there is a variable \( y_{S, d} \) for any \( S \subseteq V, d \in D \) such that \( d \odot S \). Conveniently, we use the short-hands \( y_S := \sum_{d \in D} y_{S, d} \) and \( y_d := \sum_{S \subseteq V} y_{S, d} \).

\begin{align*}
\sum_{S \in \delta(S)} y_S & \leq c_e & \forall e \in E \quad (1) \\
\sum_{d \in D} y_d & \leq \pi(D) & \forall D \subseteq D \quad (2) \\
y_{S, d} & \geq 0 & \forall d \in D, S \subseteq V, d \odot S. \quad (3)
\end{align*}

We produce a solution to the above LP. Theorem 4 is proved via some properties of this solution. These constraints look like the dual of a natural linear program for SPCSF. For the sake of convenience, we use the notation \( y(D) := \sum_{d \in D} y_d \) for any \( D \subseteq D \).

Lemma 6. Given an instance \((G, D, \pi)\) of SPCSF, we produce in polynomial time a forest \( F \) and a subset \( D_{\text{unsat}} \subseteq D \) of demands, along with a feasible vector \( y \) for the above LP such that
1. \(y(D^{\text{unsat}}) = \pi(D^{\text{unsat}});\)

2. \(F\) satisfies any demand in \(D^{\text{sat}} := D \setminus D^{\text{unsat}}\); and

3. length\((F) \leq 2y(D)\).

The solution is built up in two stages. First we perform an submodular growth to find a forest \(F_1\) and a corresponding \(y\) vector. This is different from the usual growth phase of \([35, 1]\) in that the penalty function may go tight for a set of vertices that are not currently connected. In the second stage, we prune some edges of \(F_1\) to obtain another forest \(F_2\). Below we describe the two phases of Algorithm 1 (Submodular-PC-Clustering).

Growth We begin with a zero vector \(y\), and an empty set \(F_1\). A demand \(d \in D\) is said to be live if and only if \(x(D) < \pi(D)\) for any \(D \subseteq D\) that \(d \in D\). If a demand is not live, it is dead. During the execution of the algorithm Submodular-PC-Clustering, we maintain a partition \(C\) of vertices \(V\) into clusters; it initially consists of singleton sets. Each cluster is either active or inactive; the cluster \(C \in C\) is active if and only if there is a live demand \(d : d \odot C\). We simultaneously grow all the active clusters by \(\eta\). In particular, if there are \(\kappa(C) > 0\) live demands crossing an active cluster \(C\), we increase \(y_C, d\) by \(\eta/\kappa(C)\) for each live demand \(d : d \odot C\). Hence, \(y_C\) is increased by \(\eta\) for every active cluster \(C\). We pick the largest value for \(\eta\) that does not violate any of the constraints in (1) or (2). Obviously, \(\eta\) is finite in each iteration because the values of these variables cannot be larger than \(\pi(D)\). Hence, at least one such constraint goes tight after each growth step. If this happens for an edge constraint for \(e = (u, v)\), then there are two clusters \(C_u \supseteq u\) and \(C_v \supseteq v\) in \(C\), at least one of which is growing. We merge the two clusters into \(C = C_u \cup C_v\) by adding the edge \(e\) to \(F_1\), remove the old clusters and add the new one to \(C\). Nothing needs to be done if a constraint (2) becomes tight. The number of iterations is at most \(2|V|\) because at each event either a demand dies, or the size of \(C\) decreases.

Computing \(\eta\) is nontrivial here. In particular, we have to solve an auxiliary linear program to find its value. New variables \(y^*_S, d\) denote the value of vector \(y\) after a growth of size \(\eta\). All the constraints are written for the new variables. There are exponentially many constraints in this LP, however, it admits a separation oracle and thus can be optimized.\(^6\)

\[
\begin{align*}
\text{maximize} & \quad \eta \\
\text{subject to} & \quad y^*_S, d = y_S, d + \frac{\eta}{\kappa(S)} & \forall d \in D, S \subseteq V, d \odot S, \kappa(S) > 0 \\
& \quad y_S, d = y^*_S, d & \forall d \in D, S \subseteq V, d \odot S, \kappa(S) = 0 \\
& \quad \sum_{S : e \in \delta(S)} y^*_S \leq c_e & \forall e \in E \\
& \quad \sum_{d \in D} y^*_d \leq \pi(D) & \forall D \subseteq D \\
& \quad y^*_S, d \geq 0 & \forall d \in D, S \subseteq V, d \odot S.
\end{align*}
\]

\(^6\) Notice that there are only a polynomial number of non-zero variables at each step since \(y_S, d\) may be non-zero only for clusters \(S\), and these clusters form a laminar family in our algorithm. Verifying constraints (5)-(7) and (9) is very simple. Verifying constraints (8) is equivalent to finding \(\min_{D \subseteq D} \pi(D) - y^*(D)\) and checking that it is non-negative. The function to minimize is submodular and thus can be minimized in polynomial time \([43]\). A standard argument shows that the values of these variables have polynomial size. We defer to the full version of the paper the detailed discussion of how the LP can be approximated.
**Pruning**  Let $S$ denote the set of all clusters formed during the execution of the growth step. It can be easily observed that the clusters $S$ are laminar and the maximal clusters are the clusters of $C$. In addition, notice that $F_1[C]$ is connected for each $C \in S$.

Let $B \subseteq S$ be the set of all clusters $C$ that do not cut any live demand. Notice that a demand $d$ may still be live at the end of the growth stage if it is satisfied; roughly speaking, the demand is satisfied before it exhausts its potential. In the pruning stage, we iteratively remove edges from $F_1$ to obtain $F_2$. More specifically, we first initialize $F_2$ with $F_1$. Then, as long as there is a cluster $S \in B$ such that $F_2 \cap \delta(S) = \{e\}$, we remove the edge $e$ from $F_2$.

A cluster $C$ is called a pruned cluster if it is pruned in the second stage in which case, $\delta(C) \cap F_2 = \emptyset$. Hence, a pruned cluster cannot have non-empty and proper intersection with a connected component of $F_2$.

**Algorithm 1** 
**SUBMODULAR-PC-CLUSTERING**

**Input:** Instance $(G(V,E),D,\pi)$ of Generalized prize-collecting Steiner forest

**Output:** Forest $F$, subset of demands $D_{\text{unsat}}$ and fractional solution $y$.

1: Let $F_1 \leftarrow \emptyset$.
2: Let $y_{S,d} \leftarrow 0$ for any $d \in D$, $S \subseteq V, d \odot S$.
3: Let $S \leftarrow C \leftarrow \{\{v\} : v \in V^*\}$.
4: while there is a live demand do
5:   Compute $\eta$ via LP (4): the largest possible value such that simultaneously increasing $y_C$ by $\eta$ for all active clusters $C \in C$ does not violate Constraints (1)-(3).
6:   Let $y_{C,d} \leftarrow y_{C,d} + \eta \frac{\eta}{\kappa(C)}$ for all live demands $d$ crossing clusters $C \in C$, i.e., $d \odot C$.
7:   if $\exists e \in E$ that is tight and connects two clusters $C_1$ and $C_2$ then
8:      Pick one such edge $e = (u,v)$.
9:      Let $F_1 \leftarrow F_1 \cup \{e\}$.
10:     Let $C \leftarrow C_1 \cup C_2$.
11:     Let $S \leftarrow S \cup \{C\}$.
12:    end if
13:   end while
14:   Let $B$ be the set of all clusters $S \in S$ that do not cut any live demands.
15: while $\exists S \in B$ such that $F_2 \cap \delta(S) = \{e\}$ for an edge $e$ do
16:  Let $F_2 \leftarrow F_2 \setminus \{e\}$.
17: end while
18: Output $F := F_2$, $D_{\text{unsat}}$ and $y$.

We first bound the length of the forest $F$. The following lemma is similar to the analysis of the algorithm in [35]. However, we do not have a primal LP to give a bound on the dual. Rather, the upper bound for the length is $\pi(D)$. In addition, we bound the cost of a forest $F$ that may have more than one connected component, whereas the prize-collecting Steiner tree algorithm of [35] finds a connected graph at the end.

**Lemma 7.** The cost of $F_2$ is at most $2y(D)$.

**Proof.** Recall that the growth phase has several events corresponding to an edge or set constraint going tight. We first break apart $y$ variables by epoch. Let $t_j$ be the time at which the $j^{th}$ event point occurs in the growth phase ($0 = t_0 \leq t_1 \leq t_2 \leq \cdots$), so the $j^{th}$ epoch is the interval of
time from $t_{j-1}$ to $t_j$. For each cluster $C$, let $y_C^{(j)}$ be the amount by which $y_C$ grew during epoch $j$, which is $t_j - t_{j-1}$ if it was active during this epoch, and zero otherwise. Thus, $y_C = \sum_j y_C^{(j)}$. Because each edge $e$ of $F_2$ was added at some point by the growth stage when its edge packing constraint (1) became tight, we can exactly apportion the cost $c_e$ amongst the collection of clusters $\{C : e \in \delta(C)\}$ whose variables “pay for” the edge, and can divide this up further by epoch. In other words, $c_e = \sum_j \sum_{C \in \delta(C)} y_C^{(j)}$. We will now prove that the total edge cost from $F_2$ that is apportioned to epoch $j$ is at most $2 \sum_C y_C^{(j)}$. In other words, during each epoch, the total rate at which edges of $F_2$ are paid for by all active clusters is at most twice the number of active clusters. Summing over the epochs yields the desired conclusion.

We now analyze an arbitrary epoch $j$. Let $C_j$ denote the set of clusters that existed during epoch $j$. Consider the graph $F_2$, and then collapse each cluster $C \in C_j$ into a supernode. Call the resulting graph $H$. Although the nodes of $H$ are identified with clusters in $C_j$, we will continue to refer to them as clusters, in order to to avoid confusion with the nodes of the original graph. Some of the clusters are active and some may be inactive. Let us denote the active and inactive clusters in $C_j$ by $C_{\text{act}}$ and $C_{\text{dead}}$, respectively. The edges of $F_2$ that are being partially paid for during epoch $j$ are exactly those edges of $H$ that are incident to an active cluster, and the total amount of these edges that is paid off during epoch $j$ is $(t_j - t_{j-1}) \sum_{C \in C_{\text{act}}} \deg_H(C)$. Since every active cluster grows by exactly $t_j - t_{j-1}$ in epoch $j$, we have $\sum_C y_C^{(j)} \geq \sum_{C \in C_{\text{act}}} y_C^{(j)} = (t_j - t_{j-1})|C_{\text{act}}|$. Thus, it suffices to show that $\sum_{C \in C_{\text{act}}} \deg_H(C) \leq 2|C_{\text{act}}|$.

First we must make some simple observations about $H$. Since $F_2$ is a subset of the edges in $F_1$, and each cluster represents a disjoint induced connected subtree of $F_1$, the contraction to $H$ introduces no cycles. Thus, $H$ is a forest. All the leaves of $H$ must be live clusters because otherwise the corresponding cluster $C$ would be in $B$ and hence would have been pruned away.

With this information about $H$, it is easy to bound $\sum_{C \in C_{\text{act}}} \deg_H(C)$. The total degree in $H$ is at most $2(|C_{\text{act}}| + |C_{\text{dead}}|)$. Noticing that the degree of dead clusters is at least two, we get $\sum_{C \in C_{\text{act}}} \deg_H(C) \leq 2(|C_{\text{act}}| + |C_{\text{dead}}|) - 2|C_{\text{dead}}| = 2|C_{\text{act}}|$ as desired.

Now we can prove Lemma 6 that characterizes the output of Submodular-PC-Clustering.

**Proof of Lemma 6.** For every demand $d \in D_{\text{unsat}}$ we have a set $D \supseteq d$ such that $y(D) = \pi(D)$. The definition of $D_{\text{unsat}}$ guarantees $D \subseteq D_{\text{unsat}}$. Therefore, we have sets $D_1, D_2, \ldots, D_t$ that are all tight (i.e., $y(D_i) = \pi(D_i)$) and they span $D_{\text{unsat}}$ (i.e., $D_{\text{unsat}} = \cup_i D_i$). To prove $y(D_{\text{unsat}}) = \pi(D_{\text{unsat}})$, we use induction and combine $D_i$’s two at a time. For any two tight sets $A$ and $B$ we have $y(A \cup B) = y(A) + y(B) - y(A \cap B) = \pi(A) + \pi(B) - y(A \cap B) \geq \pi(A) + \pi(B) - \pi(A \cap B) \geq \pi(A \cup B)$, where the second equation follows from tightness of $A$ and $B$, the third step is a result of Constraint (2), and the last step follows from submodularity. Constraint (2) has it that $\pi(A \cup B) \geq y(A \cup B)$, therefore, it has to hold with equality.

Clearly, at the end of execution of Submodular-PC-Clustering, any live demand is already satisfied. Notice that such demands are not affected in the pruning stage. Hence, only dead demands may be not satisfied. This guarantees the second condition. The third condition follows from Lemma 7. 

\[\Box\]
3.2 Restricting the demands

We prove Theorem 4 in this section. First, we obtain a constant-factor approximate solution $F^+$—this can be done, e.g., via the 3-approximation algorithm for general graphs \[39\]. Let $D^+$ denote the demands satisfied by $F^+$. We denote by $T^+_j$ the connected components of $F^+$. For each demand $d = \{s, t\} \in D^+$ we clearly have $\{s, t\} \subseteq V(T^+_j)$ for some $j$. However, for an unsatisfied demand $d' = \{s', t'\} \in D \setminus D^+$, the vertices $s'$ and $t'$ belong to two different components of $F^+$. Construct $G^*$ from $G$ by reducing the length of edges of $F^+$ to zero. The new penalty function $\pi^*$ is defined as follows:

$$\pi^*(D) := \epsilon^{-1} \pi(D) \quad \text{for } D \subseteq D. \quad (10)$$

Finally we run Submodular-PC-Clustering on $(G^*, D, \pi^*)$; see Algorithm 2.

\begin{algorithm}
\caption{Restrict-Demands}
\textbf{Input}: Instance $(G, D, \pi)$ of Submodular Prize-Collecting Steiner Forest
\textbf{Output}: Forest $F$ and $D^\text{unsat}$.
1: Use the algorithm of Hajiaghayi et al. \[39\] to find a 3-approximate solution: a forest $F^+$ satisfying subset $D^+$ of demands.
2: Construct $G^*(V, E^*)$ in which $E^*$ is the same as $E$ except that the edges of $F^+$ have length zero in $E^*$.
3: Define $\pi^*$ as Equation (10).
4: Call Submodular-PC-Clustering on $(G^*, D, \pi^*)$ to obtain the result $F$, $D^\text{unsat}$ and $y$.
5: Output $F$ and $D^\text{unsat}$.
\end{algorithm}

Now we show that the algorithm Restrict-Demands outlined above satisfies the requirements of Theorem 4. Before doing so, we show how the cost of a forest can be compared to the values of the output vector $y$.

**Lemma 8.** If a graph $F$ satisfies a set $D^\text{sat}$ of demands, then $\text{length}(F) \geq \sum_{d \in D^\text{sat}} y_d$.

This is quite intuitive. Recall that the $y$ variables color the edges of the graph. Consider a segment on edges corresponding to cluster $S$ with color $d$. At least one edge of $F$ passes through the cut $(S, \bar{S})$. Thus, a portion of the cost of $F$ can be charged to $y_{S,d}$. Hence, the total cost of the graph $F$ is at least as large as the total amount of colors paid for by $D^\text{sat}$. We now provide a formal proof.
Proof. The length of the graph $F$ is
\[
\sum_{e \in F} c_e \geq \sum_{e \in F} \sum_{s \in \delta(S)} y_s \geq \sum_S |F \cap \delta(S)| y_S = \sum_S |F \cap \delta(S)| y_S, \tag{1}
\]
by (1)
\[
\sum_{S: F \cap \delta(S) \neq \emptyset} y_S = \sum_{S: F \cap \delta(S) \neq \emptyset} \sum_{d \subseteq S} y_{S,d} = \sum_d \sum_{S: d \subseteq S} y_{S,d} \geq \sum_{d \in \mathcal{D}^{\text{sat}}} \sum_{S: d \subseteq S} y_{S,d} \geq \sum_{d \in \mathcal{D}^{\text{sat}}} y_d.
\]
because $y_{S,d} = 0$ if $d \in \mathcal{D}^{\text{sat}}$ and $F \cap \delta(S) = \emptyset$,
\[
= \sum_{d \in \mathcal{D}^{\text{sat}}} y_d. \tag*{\square}
\]

Proof of Theorem 4. We know that $\text{length}(F^+) + \pi(\mathcal{D} \setminus \mathcal{D}^+) \leq 3\text{OPT}$ because we start with a 3-approximate solution. For any demand $d = (s, t)$, we know that $y_d$ is not more than the distance of $s, t$ in $G^*$. Since distance between endpoints of $d$ is zero if it is satisfied in $\mathcal{D}^+$, $y_d$ is non-zero only if $d \in \mathcal{D} \setminus \mathcal{D}^+$, we have $y(D) = y(\mathcal{D} \setminus \mathcal{D}^+) \leq \pi^*(\mathcal{D} \setminus \mathcal{D}^+)$ by constraint (2). Lemma 6 gives length($F$) in $G^*$, denoted by length$_{G^*}(F)$, is at most $2y(D) \leq 2\pi^*(\mathcal{D} \setminus \mathcal{D}^+) = 2\epsilon^{-1}\pi(\mathcal{D} \setminus \mathcal{D}^+) \leq 6\epsilon^{-1}\text{OPT}$. Therefore, $\text{length}(F) = \text{length}(F^+) + \text{length}_{G^*}(F) \leq (6\epsilon^{-1} + 3)\text{OPT}$.

To establish the second condition of the theorem, take an optimal forest $F'$: $F'$ satisfies demands $\mathcal{D}^{\text{OPT}}$, and we have $\text{length}(F') + \pi(\mathcal{D} \setminus \mathcal{D}^{\text{OPT}}) = \text{OPT}$. Define $A := \mathcal{D}^{\text{OPT}} \setminus \mathcal{D}^{\text{sat}}$ and $B := \mathcal{D}^{\text{unsat}} \setminus A$. The penalty of $F'$ under $\pi'$ is $\pi((\mathcal{D} \setminus \mathcal{D}^{\text{OPT}}) \cup \mathcal{D}^{\text{unsat}}) = \pi((\mathcal{D}^{\text{sat}} \setminus \mathcal{D}^{\text{OPT}}) \cup A \cup B)$. Hence, the increase in penalty of $F'$ due to changing from $\pi$ to $\pi'$ is $\pi((\mathcal{D}^{\text{sat}} \setminus \mathcal{D}^{\text{OPT}}) \cup A \cup B) - \pi((\mathcal{D}^{\text{sat}} \setminus \mathcal{D}^{\text{OPT}}) \cup B) \leq \pi(A \cup B) - \pi(B)$ due to the decreasing marginal cost property of submodular functions. We have $y(A \cup B) = \pi^*(A \cup B) = \epsilon^{-1}\pi(A \cup B)$ because $A \cup B = \mathcal{D}^{\text{unsat}}$ is the set of dead demands of SUBMODULAR-PC-CLUSTERING; see the first condition of Lemma 6. We also have $\epsilon^{-1}\pi(B) = \pi^*(B) \geq y(B)$ because of Constraint (2). Therefore, the additional penalty is at most $\epsilon(y(A \cup B) - y(B))] = \epsilon y(A)$. Since $F'$ satisfies the demands $A$, we have $y(A) \leq \text{length}(F') \leq \text{OPT}$ from Lemma 8. Therefore, the additional penalty is at most $\epsilon\text{OPT}$.

The extension to SPTSP and SPSC is straight-forward once we observe that the cost of building a tour or a stroll on a subset $S$ of vertices is at least the cost of constructing a Steiner tree on the same set. Hence, there algorithm pretends it has an SPCST instance, and restricts the demand set accordingly. However, the extra penalty due to the ignored demands $\mathcal{D}^{\text{unsat}}$ is charged to the Steiner tree cost which is no more than the TSP or stroll length. \tag*{\square}
3.3 Restricting the connectivity

We first run \textsc{Restrict-Demands} on \((G, \mathcal{D}, \pi)\). Let \(F\) and \(D_{\text{unsat}}\) be its output. The forest \(F\) satisfies all the demands in \(D_{\text{sat}} := \mathcal{D} \setminus D_{\text{unsat}}\). The length of this forest is \(O(\text{OPT})\) and the demands in \(D_{\text{unsat}}\) can be safely ignored.

The forest \(F\) consists of tree components \(T_i\). In the following, we connect some of these components to make the trees \(\hat{T}_i\). It is easy to see that this construction guarantees the first two conditions of Theorem 5. We work on a graph \(G^*(V^*, E^*)\) formed from \(G\) by contracting each tree component of \(F\). A potential \(\phi_v\) is associated with each vertex \(v\) of \(G^*\), which is \(\epsilon - 1\) times the length of the tree component corresponding to \(v\) in case \(v\) is the contraction of a tree component, and zero otherwise.

We use the algorithm \textsc{PC-Clustering} introduced in [12] to cluster the components \(T_i\) and construct a forest \(F_2\) with components \(\hat{T}_i\); the details of the algorithm can be seen in [12]. We obtain the following guarantees.

We first show the cost of the new edges is small.

\textbf{Lemma 9 ([12, Lemma 6])}. The cost of \(F_2\) is at most \(2 \sum_{v \in V^*} \phi_v\).

Recall that the trees \(T_i\) are contracted in \(F_2\). Construct \(\hat{F}\) from \(F_2\) by uncontracting all these trees. Let \(\hat{F}\) consist of tree components \(\hat{T}_i\). It is not difficult to verify that \(\hat{F}\) is indeed a forest, but we do not need this condition since we can always remove cycles to find a forest. Define \(\mathcal{D}_i := \{(s, t) \in \mathcal{D} : s, t \in V(\hat{T}_i)\}\), and let \(D^*\) be the subset of demands satisfied by \(\text{OPT}\). Define \(D_i^* := D^* \cap \mathcal{D}_i\), and denote by \(\text{SteinerForest}(G, D)\) the length of a minimum Steiner forest of \(G\) satisfying the demands \(D\).

\textbf{Lemma 10 ([12, Lemma 10])}. \(\sum_i \text{SteinerForest}(G, D_i^*) \leq (1 + \epsilon) \text{SteinerForest}(G, D^*)\).

Now, we are ready to prove the main theorem of this section.

\textit{Proof of Theorem 5.} The first condition of the lemma follows directly from our construction: we start with a solution, and never disconnect one of the tree components in the process. The construction immediately implies the second condition. By Lemma 9, the cost of \(F_2\) is at most \(2 \sum_{v \in V^*} \phi_v \leq \frac{2}{\epsilon} \text{length}(F)\). Thus, \(\hat{F}\) costs no more than \((2/\epsilon + 1)\text{length}(F)\), giving the third condition. Finally, Lemma 10 establishes the last condition.

\section{PTASs for PCST, PCTSP and PCS on planar graphs}

Since PCST is a special case of PCSF, Theorems 1 and 2 imply that PCST admits a PTAS on planar graphs. However, obtaining the same result for PCTSP and PCS is not immediate from those theorems since the latter problems are not special cases of PCSF. Here we explain how we can use these theorems to obtain the desired PTASs. Here we focus on PCTSP, however, the same arguments with minor changes apply to PCS as well.

Take an instance \(I = (G, \mathcal{D}, \pi)\) of PCTSP, and apply Theorem 4 on \(I\) to obtain \(F\) and \(D_{\text{unsat}}\). Since all the demands share a common root vertex\(^7\), all the terminals in \(D_{\text{sat}}\) are connected in \(F\). We then invoke the TSP spanner construction of Arora et al. [6] to build \(H\). Finally, we use

\(^7\)If we have a penalty for each vertex in the PCTSP formulation, we can guess a root vertex \(r\) and define the demand pairs accordingly.
the contraction decomposition theorem of Demaine et al. [25] to contract a small-weight subset of edges and reduce the problem to graphs of bounded treewidth. The total additional charge due to penalties of $D^{\text{unsat}}$ and contracted edges is at most $O(\epsilon)\text{OPT}$. Therefore, we can obtain a PTAS by solving the bounded-treewidth instance precisely.

5 Hardness of PCSF on series-parallel graphs

We first present the hardness proof for PCSF on a planar graph of treewidth two. The proof shows hardness for a very restricted class of graphs: short cycles going through a single central vertex.

Proof of Theorem 3(1). We reduce an instance $I$ of Vertex Cover on 3-regular graphs to an instance $I'$ of PCSF on a planar graphs of treewidth two. The former is known to be APX-hard [3]. The instance $I$ is defined by an undirected graph $G$. If $n$ denotes the number of vertices of $G$, the number edges is $m = 3n/2$. We will denote the $i$-th vertex of $G$ by $v_i$, the $j$-th edge by $e_j$, and the first and second endpoints of $e_j$ by $e_{j}^{(1)}$ and $e_{j}^{(2)}$, respectively.

We now specify the reduction (illustrated in Figure 1); $I'$ is represented by $(H, D, \pi)$. The graph $H$ consists of the vertices

- $a_i$ for $1 \leq i \leq n$,
- $b_j, c_j^1, c_j^2$ for $1 \leq j \leq m$,
- central vertex $w$,

and the edges

- $\{w, a_i\}$ of cost 2 ($1 \leq i \leq n$),
- $\{w, c_j^1\}, \{w, c_j^2\}, \{c_j^1, b_j\}, \{c_j^2, b_j\}$ of cost 1 ($1 \leq j \leq m$).

The instance contains the following demands:

- $\{w, b_j\}$ with penalty 3 ($1 \leq j \leq m$),
- If $v_i = e_{j}^{(\ell)}$ for some $1 \leq i \leq n$, $1 \leq j \leq m$, and $\ell \in \{1, 2\}$, then $\{a_i, e_{j}^{\ell}\}$ is a demand with penalty 1.

Thus the number of demands is exactly $m + 3n$ and each $a_i$ appears in exactly 3 demands. We claim that the cost of the optimum solution of $I'$ is exactly $2m + 2n + \tau(G)$, where $\tau(G)$ is the size of the minimum vertex cover in $G$. Note that $\tau(G) \geq n/3$ (as $G$ is 3-regular), thus $2m + 2n + \tau(G)$ is at most a constant times $\tau(G)$. In order to prove the correctness of the reduction, we prove the following two statements:

1. Given a vertex cover of size $k$ for $G$, a solution of cost $2m + 2n + k$ can be constructed.
2. Given a solution of cost at most $2m + 2n + k$, a vertex cover of size at most $k$ can be constructed.

To prove (1), suppose that $C$ is a vertex cover of size $k$ for $G$. Let $T$ be a tree of $H$ that contains

- edge $\{w, a_i\}$ if and only if $v_i \notin C$,
Figure 1: Illustrating the reduction from 3-Regular Vertex Cover to PCSF.

- edges $\{w, c_j^1\}, \{c_j^1, b_j\}$ if and only if $e_j^1 \not\in C$,
- edges $\{w, c_j^2\}, \{c_j^2, b_j\}$ if and only if $e_j^1 \in C$.

The total cost of $T$ is $2(n - k) + 2m$. Observe that all the demands $\{w, b_j\}$ are connected (either via $c_j^1$ or $c_j^2$). Furthermore, if $v_i \not\in C$, then all three demands where $a_i$ appears are satisfied: edge $\{w, a_i\}$ is in $T$ and if $v_i = e_j^1$, then edge $\{w, c_j^1\}$ is in $T$ as well. (Note that if $v_i = e_j^2$ and $v_i \not\in C$, then $e_j^1 \in C$ must hold, and therefore $\{w, c_j^2\}$ is in $T$.) Thus the total penalty is at most $3k$, and hence the cost of the solution is at most $2n + 2m + k$, as claimed.

To prove (2), suppose that subgraph $F$ of $G$ is a solution such that the sum of the cost of $F$ and the penalties is at most $2m + 2n + k$. We can assume that for every $1 \leq i \leq n$, vertex $b_j$ can be reached from $w$: otherwise we can decrease the penalty by 3 at the cost of adding two edges of cost 1. Furthermore, we can assume that only one of $c_j^1$ and $c_j^2$ is can be reached from $w$: otherwise we can remove an edge without disconnecting $b_j$ from $w$, thus the cost decreases by 1 and the penalty increases by at most 1. Finally, we can assume that if $\{w, a_i\} \in F$, then all 3 demands containing $a_i$ are connected: otherwise removing $\{w, a_i\}$ decreases the cost by 2 and increases the penalty by at most 2.

Let vertex $v_i$ be in $C$ if and only if $\{w, a_i\} \not\in F$. We claim that $C$ is a vertex cover of size at most $k$. To see that $C$ is a vertex cover, consider an edge $e_j$. We have observed above that one of $c_j^1$ and $c_j^2$ cannot be reached from $w$. If $e_j^1$ cannot be reached from $w$ and $e_j^{(1)} = v_i$, then the demand $\{v_i, c_j^1\}$ is not connected by $F$. Therefore, not all 3 demands containing $a_i$ are connected, which means (as observed above) that $\{w, a_i\} \not\in F$. Thus $v_i \in C$, covering the edge $e_j$.

Since every $b_j$ can be reached from $w$ and $\{w, a_i\} \in F$ if $v_i \not\in C$, the cost of $F$ is at least $2m + 2(n - |C|)$. Furthermore, if $v_i \in C$, then $\{w, a_i\} \not\in F$, which means that we have to pay the penalty for the 3 demands containing $a_i$. Therefore, the total cost of the solution is at least
The proof for the Euclidean version is very similar to the graph version. The main difference is that the central vertex $w$ is replaced by a set of points arranged along a long vertical path.

Proof of Theorem 3(2). We reduce an instance $I$ of Vertex Cover on 3-regular graphs to an instance $I'$ of PCSF on points in the Euclidean plane. If $n$ denotes the number of vertices of the 3-regular graph $G$ in $I$, then the number edges is $m = 3n/2$. We will denote the $i$-th vertex of $G$ by $v_i$, the $j$-th edge by $e_j$, and the first and second endpoints of $e_j$ by $e_j^{(1)}$ and $e_j^{(2)}$, respectively.

We now specify the reduction (illustrated in Figure 2). Let us define $U := 10000(n + m)$ (“basic unit of cost”), $H = 10U$ (“horizontal length”), and $V = 100U$ (“vertical spacing”). Instance $I'$ contains the following set $P$ of points:

- $z_{0,y} = (0, y)$ for every $-mV \leq y \leq nV$,
- $z_{x,y} = (x, y)$ and for every $0 \leq x \leq H$ and $y = iV$ for $1 \leq i \leq n$,
- $z_{x,y} = (x, y)$ and $z_{x+1,y}$ for every $0 \leq x \leq H$ and $y = -jV$ for $1 \leq j \leq m$,
- $a_i = (H + 2U, iV)$ for $1 \leq i \leq n$,
- $b_j = (H, -jV + 2U)$ for $1 \leq j \leq m$,
- $c_j = (H, -jV + U)$, and $c_j^2 = (H, -jV + 3U)$ for $1 \leq j \leq m$.

Let $Z$ be the set of all $z_{x,y}$ vertices in $P$, note that $|Z| = V(i + j) + 1 + (i + 2j)H$. For ease of notation, we define $w_i = z_{H,iV}$, $w_1 = z_{H,-jV}$, $w_2 = z_{H,-jV+4U}$.

The instance contains the following demands:

1. If $z_{x,y}$ and $z_{x+1,y}$ are both in $P$, then there is a demand $\{z_{x,y}, z_{x+1,y}\}$ with penalty 1.
2. If $z_{x,y}$ and $z_{x,y+1}$ are both in $P$, then there is a demand $\{z_{x,y}, z_{x,y+1}\}$ with penalty 1.
3. $\{(0,0), b_j\}$ with penalty $3U$ ($1 \leq j \leq n$),
4. If $v_i = e_j^{(\ell)}$ for some $1 \leq i \leq n$, $1 \leq j \leq m$, and $\ell \in \{1, 2\}$, then $\{a_i, c_j^\ell\}$ is a demand with penalty $\bar{U} − 10$.

The total number of demands is $|Z| - 1 + n + 3m$ and each $a_i$ appears in exactly 3 demands. We claim that the cost of the optimum solution of $I'$ is between $|Z| + (2m + 2n + \tau(G))U$ and $|Z| + (2m + 2n + \tau(G))U - 100n$, where $\tau(G)$ is the size of the minimum vertex cover in $G$. Note that $m = 3n/2$ and $\tau(G) \leq m/3$, thus $|Z| + (2m + 2n + \tau(G))U$ is at most a constant factor larger than $\tau(G)U$.

More precisely, in order to prove the correctness of the reduction, we prove the following two statements:

(1) Given a vertex cover of size $k$ for $G$, a solution of cost at most $|Z| + (2m + 2n + k)U$ for $I'$ can be constructed.
(2) Given a solution of cost at most \(|Z| + (2m + 2n + k)U\) for \(T^{'}\), a vertex cover of size at most \(k\) can be constructed.

To prove (1), suppose that \(C\) is a vertex cover of size \(k\) for \(G\). Let \(F\) be the forest (actually, a tree) that contains

1. edge \(\{z_{x,y}, z_{x+1,y}\}\) if both these points are in \(P\),
2. edge \(\{z_{x,y}, z_{x,y+1}\}\) if both these points are in \(P\),
3. edge \(\{w_i, a_i\}\) if \(v_i \notin C\),
4. edges \(\{w_j^1, c_j^1\}\) and \(\{c_j^1, b_j\}\) if \(e_j^{(1)} \notin C\),
5. edges \(\{w_j^2, c_j^2\}\) and \(\{c_j^2, b_j\}\) if \(e_j^{(1)} \in C\).

The total cost of \(F\) is \(|Z| - 1 + 2U(n - k) + 2Um\). Observe that all the demands \(\{(0, 0), b_j\}\) are satisfied. Furthermore, if \(v_i \notin C\), then all three demands where \(a_i\) appears are satisfied. This can be seen as follows. First, \(a_i\) is in the same component as \(w_i\) and hence as every vertex of \(Z\). If \(v_i = e_j^{(1)}\), then there is a demand \(\{a_i, c_j^1\}\) and \(c_j^1\) is connected with \(w_j^1\) (and hence with \(a_i\)). If \(v_i = e_j^{(2)}\), then \(v_i \notin C\) means that \(e_j^{(1)} \in C\) must hold, and therefore \(c_j^2\) is connected to \(w_j^2\), satisfying the demand \(\{a_i, c_j^2\}\). Thus the total penalty is at most \(3k(U - 10)\), and hence the cost of the solution is at most \(|Z| - 1 + (2m + 2n + k)U - 30k\), as claimed.

To prove (2), suppose that forest \(F\) is an optimum solution such that the sum of the cost of \(F\) and the penalties is at most \(|Z| + (2n + 2m + k)U\). First, we can assume that every demand of the first two types is satisfied: if, say, \((z_{x,y}, z_{x+1,y})\) is not satisfied, then we can extend \(F\) by adding an edge of cost 1, which decreases the penalty by at least 1. Thus all the \(z_{x,y}\) points are in the same connected component \(K\) of \(F\). We can also assume that every demand of the third type is satisfied: if \(\{(0, 0), b_j\}\) is not satisfied, then we can decrease the penalty by \(3U\) at the cost of \(2U\) by adding edges \(\{w_j^1, c_j^1\}\) and \(\{c_j^1, b_j\}\), contradicting the optimality of \(F\). Therefore, every vertex \(b_j\) is in the component \(K\).

Let \(Z' = \{z_{x,y} \in Z \mid x = 0 \lor x \geq 10\}\). Let \(R\) be the region of the plane at Manhattan distance at most 3 from \(Z'\). Note that \(R\) consists of one “vertical” and \(n + 2m\) “horizontal” components.

We claim that the cost of \(F\) inside \(R\) is at least \(|Z'|\). We have seen above that a single component \(K\) of \(F\) contains every point of \(P \cap R\). The restriction of \(K\) to \(R\) gives rise to several components. Consider such a component \(K'\) containing a subset \(S 
subseteq Z'\) of vertices. We show that the cost of \(K'\) is at least \(|S|\). The vertices of \(S\) lie on a horizontal or vertical line. This means that there are two vertices \(s_1, s_2 \in S\) at distance \(d \geq |S| - 1\). As \(K\) is not contained fully in any component of \(R\), component \(K'\) has to contain a point \(s_3\) on the boundary of \(R\). As \(s_3\) is at distance at least 3 from \(s_1\) and \(s_2\), it can be verified that any Steiner tree of \(s_1, s_2, s_3\) has cost at least \(d + 1 = |S|\). Summing for every component \(K'\) of the restriction of \(K\) to \(R\), we get that the cost of \(K\) in \(R\) is at least \(|P \cap R|\).

Let \(R^+\) be the region of space at Manhattan distance at most 3 from \(Z\). We claim that the cost of every component of \(F \setminus R^+\) is at most \(3U\). There are two types of components of \(F \setminus R^+\): (1) those that contain a point of \(P\) and (2) those that do not contain such a point. Clearly, there are at most \(n + 3m\) components of the first type. Suppose that there is a component \(D\) of the second type having cost more than \(3U\). In this case, we modify \(F\) to obtain a better solution as follows. Consider
Figure 2: Illustrating the reduction from 3-Regular Vertex Cover to Euclidean PCSF.
\( F \setminus R^+ \) (i.e., let us remove the part of \( F \) inside \( R^+ \)) and let us remove every component of the second type. After that, let us add all the \(|Z| - 1\) edges of the form \( \{w_{x,y}, w_{x+1,y}\}, \{w_{x,y}, w_{x,y+1}\} \). Finally, for every component of the first type, if it intersects \( R^+ \), then let us choose a point of the component on the boundary of \( R^+ \) and connect this point to the nearest vertex of \( Z \). It is clear that the new forest \( F' \) satisfies every demand satisfied by \( F \): every point of \( P \) connected to \( Z \) remains connected to \( Z \). By our claim in the previous paragraph, the cost of \( F \setminus R' \) is less than the cost of \( F \) by at least \(|Z'| = |Z| - 9(n + 2m)\). Removing components of the second type decreases the cost by more than \( 3U \) (as there are at least one such component having cost more than \( 3U \)). The edges connecting \( Z \) increase the cost by \(|Z| - 1\). Adding the new connections corresponding to the components of the first type increases the cost by at most \( n + 3m \). As \( 3U \geq 9(n + 2m) - 1 + n + 3m \), forest \( F' \) is a strictly better solution, a contradiction.

Suppose now that there is a component \( D \) of the first type with cost more than \( 3U \). For \(-m \leq s \leq n\), let \( R_s \) be the region of the plane at Manhattan distance at most \( 4U \) from \((H, sV)\). Observe that for each \( s \), all the points of \( P \cap R_s \) can be connected to the nearest point of \( Z \) with a total cost of at most \( 3U \). This means that if \( D \) intersects only one of these regions, say \( R_s \), then we can substitute \( D \) at cost at most \( 3U \) in such a way that every demand satisfied by \( F \) remains satisfied, contradicting the optimality of \( F \). Suppose therefore that \( D \) intersects \( t \geq 2 \) of these regions; in this case, the cost of \( D \) is at least \((t - 1)(V - 8U) > 6tU - 6U \geq 3tU \). Let us replace \( D \) by connecting every point of \( P \cap D \) to the closest vertex of \( Z \). The new connections increase the cost by at most \( t \cdot 3U \), which is less than the cost of \( D \), a contradiction.

We have proved that for every component \( D \) of \( F \setminus R^+ \), \( D \cap P \) is either a single \( a_i \), or a subset of \( \{b_j, c_j^1, c_j^2\} \). Therefore, every such component \( D \) intersects \( R^+ \): otherwise, \( D \) could be safely removed, as it does not satisfy any demand. Next we show that it can be assumed that only one of \( c_j^1 \) and \( c_j^2 \) is in \( K \). Otherwise we can remove every component of \( F \setminus R^+ \) intersecting \( \{b_j, c_j^1, c_j^2\} \) and replace them with the edges \( \{w_j, c_j^1\} \) and \( \{c_j^1, b_j\} \). The total cost of the components we removed is at least \( 2U - 3 + U - 3 \) (which is the minimum cost of connecting \( b_j, c_j^1, c_j^2 \) to \( R^+ \)) and the new edges have cost \( 2U \). This transformation might disconnect the demand containing \( c_j^2 \), hence the penalty can increase by at most \( U - 10 \) only, contradicting the optimality of \( F \).

We can assume that if \( a_i \) is in \( K \), then all 3 demands containing \( a_i \) are connected: otherwise removing the component of \( F \setminus R^+ \) containing \( a_i \) decreases the cost by at least \( 2U - 3 \) and increases the penalty by at most \( 2(U - 10) \).

Let vertex \( v_i \) be in \( C \) if and only if \( a_i \) is not in component \( K \). We claim that \( C \) is a vertex cover of size at most \( k \). To see that \( C \) is a vertex cover, consider an edge \( e_j \). We have observed above that one of \( c_j^1 \) and \( c_j^2 \) is not in \( K \). If \( c_j^1 \not\in K \) and \( e_j^{(1)} = v_i \), then the demand \( \{a_i, c_j^1\} \) is not connected by \( F \). Therefore, not all 3 demands containing \( a_i \) are connected, which means (as observed above) that \( a_i \) is not in \( K \). Thus \( v_i \in C \), covering the edge \( e_j \). Similarly, \( c_j^2 \not\in K \), then \( e_j^{(2)} \in C \).

The cost of \( F \cap R^+ \) is at least \(|Z| - 9(n + 2m)\). Since every \( b_j \) is in \( K \) and \( a_i \) is in \( K \) if \( v_i \not\in C \), the cost of \( F \setminus R^+ \) is at least \((2U - 3)m + (2U - 3)(n - |C|)\). Furthermore, if \( v_i \in C \), then we have to pay the penalty for the 3 demands containing \( a_i \). Therefore, the total cost of the solution is at least

\[
|Z| - 9(n + 2m) + (2U - 3)m + (2U - 3)(n - |C|) + 3|C|(U - 10) \geq |Z| + (2m + 2n + |C|)U - 100n.
\]

We assumed that the cost of the solution is at most \(|Z| + (2m + 2n + k)U \). As \( U > 100n \), this is only possible if \( |C| \leq k \), what we had to prove. \( \square \)

20
References

[1] A. Agrawal, P. N. Klein, and R. Ravi, When trees collide: An approximation algorithm for the generalized Steiner problem on networks, in Proceedings of the twenty-third Annual ACM Symposium on Theory of Computing (STOC), 1991, pp. 134–144.

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

[3] P. Alimonti and V. Kann, Some APX-completeness results for cubic graphs, Theoret. Comput. Sci., 237 (2000), pp. 123–134.

[4] 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.

[5] A. Archer, A. Levin, and D. P. Williamson, A faster, better approximation algorithm for the minimum latency problem, SIAM J. Comput., 37 (2008), pp. 1472–1498.

[6] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn, A polynomial-time approximation scheme for weighted planar graph tsp, in Proceedings of the ninth annual ACM-SIAM Symposium on Discrete algorithms (SODA’98), 1998, pp. 33–41.

[7] S. Arora and G. Karakostas, A $2 + \epsilon$ approximation algorithm for the k-MST problem, Mathematical Programming, 107 (2006), pp. 491–504.

[8] S. Arya and H. Ramesh, A 2.5 factor approximation algorithm for the k-MST problem, Information Processing Letters, 65 (1998), pp. 117–118.

[9] B. S. Baker, Approximation algorithms for np-complete problems on planar graphs, Journal of the ACM, 41 (1994), pp. 153–180.

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

[11] M. Bateni and M. Hajiaghayi, Euclidean prize-collecting steiner forest, in Proceedings of the 9th Latin American Theoretical Informatics Symposium (LATIN’10), 2010. to appear.

[12] M. Bateni, M. Hajiaghayi, and D. Marx, Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth, in Proceedings of the forty-second annual ACM Symposium on Theory of computing (STOC’10), New York, NY, USA, 2010, ACM. to appear.

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

[14] H. L. Bodlaender and A. M. C. A. Koster, Combinatorial optimization on graphs of bounded treewidth, Comput. J., 51 (2008), pp. 255–269.

[15] 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.
[16] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanita, An improved LP-based approximation for steiner tree, in Proceedings of the forty-second annual ACM Symposium on Theory of computing (STOC’10), New York, NY, USA, 2010, ACM. to appear.

[17] S. Cabello and B. Mohar, Finding shortest non-separating and non-contractible cycles for topologically embedded graphs, in Proceedings of the 13th Annual European Symposium of Algorithms (ESA), 2005, pp. 131–142.

[18] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, Paths, trees, and minimum latency tours, in Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science, 2003, pp. 36–45.

[19] C. Chekuri, A. Ene, and N. Korula, Prize-collecting Steiner tree and forest in planar graphs, 2010. Personal communications.

[20] N. Christofides, Worst-case analysis of a new heuristic for the travelling-salesman problem, tech. rep., Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[21] 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’01), London, UK, 2001, Springer-Verlag, pp. 60–70.

[22] F. A. Chudak, T. Roughgarden, and D. P. Williamson, Approximate k-MSTs and k-Steiner trees via the primal-dual method and Lagrangean relaxation, Mathematical Programming, 100 (2004), pp. 411–421.

[23] G. Cornuejols, M. Fisher, and G. L. Nemhauser, On the uncapacitated location problem, in Studies in integer programming (Proc. Workshop, Bonn. 1975), North-Holland, Amsterdam, 1977, pp. 163–177. Ann. of Discrete Math., Vol. 1.

[24] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms, Management Sci., 23 (1976/77), pp. 789–810.

[25] E. D. Demaine, M. Hajiaghayi, and B. Mohar, Approximation algorithms via contraction decomposition, in Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2007, pp. 278–287.

[26] J. Edmonds, Submodular functions, matroids, and certain polyhedra, in Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969), Gordon and Breach, New York, 1970, pp. 69–87.

[27] U. Feige, A threshold of \( \ln n \) for approximating set cover, J. ACM, 45 (1998), pp. 634–652.

[28] U. Feige and M. X. Goemans, Approximating the value of two power proof systems, with applications to max 2sat and max dicut, in Proceedings of the 3rd Israel Symposium on the Theory of Computing Systems (ISTCS’95), Washington, DC, USA, 1995, IEEE Computer Society, p. 182.
[29] M. R. Garey and D. S. Johnson, *The rectilinear Steiner tree problem is NP-complete*, SIAM J. Appl. Math., 32 (1977), pp. 826–834.

[30] N. Garg, *A 3-approximation for the minimum tree spanning k vertices*, in Proceedings of the 37th Annual Symposium on Foundations of Computer Science, 1996, pp. 302–309.

[31] ———, *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, 2005, pp. 396–402.

[32] M. Goemans, *The prize-collecting TSP revisited*. Available from http://www-math.mit.edu/~goemans/prizecollect.ps. Talk slides from the 1998 SIAM Discrete Math conference.

[33] M. X. Goemans, *Combining approximation algorithms for prize-collecting TSP*. unpublished manuscript, 2009.

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

[35] M. X. Goemans and D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming*, J. Assoc. Comput. Mach., 42 (1995), pp. 1115–1145.

[36] M. Grigni, E. Koutsoupias, and C. Papadimitriou, *An approximation scheme for planar graph tsp*, in Proceedings of the 36th Annual Symposium on Foundations of Computer Science (FOCS’95), Washington, DC, USA, 1995, IEEE Computer Society, p. 640.

[37] A. Gupta, J. Könemann, S. Leonardi, R. Ravi, and G. Schäfer, *An efficient cost-sharing mechanism for the prize-collecting Steiner forest problem*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA ’07), 2007, pp. 1153–1162.

[38] 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, New York, 2006, ACM, pp. 631–640.

[39] M. Hajiaghayi, R. Khandekar, G. Kortsarz, and Z. Nutov, *Prize-collecting Steiner network problems*, in Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO), 2010. to appear.

[40] M. Hajiaghayi and A. A. Nasri, *Prize-collecting Steiner networks via iterative rounding*, in Proceedings of the 9th Latin American Theoretical Informatics Symposium (LATIN’10), 2010. To appear.

[41] J. Höstad, *Some optimal inapproximability results*, J. ACM, 48 (2001), pp. 798–859 (electronic).

[42] J. Hoogeveen, *Analysis of Christofides’ heuristic: Some paths are more difficult than cycles*, Operations Research Letters, 10 (1991), pp. 291–295.
A Basic graph theory definitions

Let $G(V, E)$ be a graph. As is customary, let $\delta(V')$ denote the set of edges having one endpoint in a subset $V' \subseteq V$ of vertices. For a subset of vertices $V' \subseteq V$, the subgraph of $G$ induced by $V'$ is denoted by $G[V']$. With slight abuse of notation, we sometimes use the edge set to refer to the graph itself. Hence, the above-mentioned subgraph may also be referred to by $E[V']$ for simplicity.

We denote the length of a shortest $x$-to-$y$ path in $G$ as $\text{dist}_G(x, y)$. For an edge set $E$, we denote by $\ell(E) := \sum_{e \in E} c_e$ the total length of edges in $E$.

Given an edge $e = (u, v)$ in a graph $G$, the contraction of $e$ in $G$ denoted by $G/e$ is the result of unifying vertices $u$ and $v$ in $G$, and removing all loops and multiple edges except the shortest edge. More formally, the contracted graph $G/e$ is formed by the replacement of $u$ and $v$ with a single vertex such that edges incident to the new vertex are the edges other than $e$ that were incident
with \( u \) or \( v \). To obtain a simple graph, we first remove all self-loops in the resulting graph. In case of multiple edges, we only keep the shortest edge and remove all the rest. The contraction \( G/E' \) is defined as the result of iteratively contracting all the edges of \( E' \) in \( G \), i.e., \( G/E' := G/e_1/e_2/\ldots/e_k \) if \( E' = \{e_1,e_2,\ldots,e_k\} \). Clearly, the planarity of \( G \) is preserved after the contraction. Similarly, contracting edges does not increase the cost of an optimal Steiner forest.

The boundary of a face of a planar embedded graph is the set of edges adjacent to the face; it does not always form a simple cycle. The boundary \( \partial H \) of a planar embedded graph \( H \) is the set of edges bounding the infinite face. An edge is strictly enclosed by the boundary of \( H \) if the edge belongs to \( H \) but not to \( \partial H \).

Now we define the basic notion of treewidth, as introduced by Robertson and Seymour [50]. To define this notion, we consider representing a graph by a tree structure, called a tree decomposition. More precisely, a tree decomposition of a graph \( G(V,E) \) is a pair \((T,B)\) in which \( T(I,F) \) is a tree and \( B = \{B_i \mid i \in I\} \) is a family of subsets of \( V(G) \) such that 1) \( \bigcup_{i \in I} B_i = V; \) 2) for each edge \( e = (u,v) \in E, \) there exists an \( i \in I \) such that both \( u \) and \( v \) belong to \( B_i; \) and 3) for every \( v \in V, \) the set of nodes \( \{i \in I \mid v \in B_i\} \) forms a connected subtree of \( T. \)

To distinguish between vertices of the original graph \( G \) and vertices of \( T \) in the tree decomposition, we call vertices of \( T \) nodes and their corresponding \( B_i \)'s bags. The width of the tree decomposition is the maximum size of a bag in \( B \) minus 1. The treewidth of a graph \( G \), denoted \( \text{tw}(G) \), is the minimum width over all possible tree decompositions of \( G. \)

For algorithmic purposes, it is convenient to define a restricted form of tree decomposition. We say that a tree decomposition \((T,B)\) is nice if the tree \( T \) is a rooted tree such that for every \( i \in I \) either

1. \( i \) has no children (\( i \) is a leaf node),
2. \( i \) has exactly two children \( i_1, i_2 \) and \( B_i = B_{i_1} = B_{i_2} \) holds (\( i \) is a join node),
3. \( i \) has a single child \( i' \) and \( B_i = B_{i'} \cup \{v\} \) for some \( v \in V \) (\( i \) is an introduce node), or
4. \( i \) has a single child \( i' \) and \( B_i = B_{i'} \setminus \{v\} \) for some \( v \in V \) (\( i \) is a forget node).

It is well-known that every tree decomposition can be transformed into a nice tree decomposition of the same width in polynomial time. Furthermore, we can assume that the root bag contains only a single vertex.

We also need a basic notion of embedding; see, e.g., [51, 17]. In this paper, an embedding refers to a 2-cell embedding, i.e., a drawing of the vertices and edges of the graph as points and arcs in a surface such that every face (connected component obtained after removing edges and vertices of the embedded graph) is homeomorphic to an open disk. We use basic terminology and notions about embeddings as introduced in [48]. We only consider compact surfaces without boundary. Occasionally, we refer to embeddings in the plane, when we actually mean embeddings in the 2-sphere. If \( S \) is a surface, then for a graph \( G \) that is (2-cell) embedded in \( S \) with \( f \) facial walks, the number \( g = 2 - |V(G)| + |E(G)| - f \) is independent of \( G \) and is called the Euler genus of \( S \). The Euler genus coincides with the crosscap number if \( S \) is non-orientable, and equals twice the usual genus if the surface \( S \) is orientable.
PCST, PCTSP and PCS on bounded-treewidth graphs

Treewidth is a notion of how similar a graph is to trees. Since tree structure usually lends itself to the dynamic programming approach, it is plausible that many optimization problems may be solvable in polynomial time on graphs of bounded treewidth; Bodlaender and Koster [14] have a comprehensive survey on this topic. In particular, several Steiner network problems become relatively easy when restricted to bounded-treewidth graphs. Among them are Steiner Tree, TSP and Stroll. One surprising outlier is Steiner forest that is proved to be NP-hard, yet it admits a PTAS [12]. In this section, we study the prize-collecting extensions of the above problems, and when possible, we provide a polynomial-time algorithm for them. More specifically, we present PTASs for PCST, PCTSP and PCS on bounded-treewidth graphs. We already showed in Section 5 that PCSF is APX-hard even on series-parallel graphs. The proof is extended to give APX-hardness for Euclidean plane.

We focus the discussion on PCST, however, minor modifications allow us to solve PCTSP and PCS, too. We are given a weighted graph $G(V, E)$ of treewidth $k - 1$ for a fixed parameter $k$, and a penalty function $\pi : V \to \mathbb{R}_+$. We have a nice tree decomposition $(T, B)$ for $G$. Each bag $B_i$ has size at most $k$. These are sometimes called portals for the subtree below node $B_i$. Let $I$ denote the nodes of the tree decomposition $T$, and for each $i \in I$, let $T_i$ be the subtree of $T$ below $i$. A dynamic programming entry is specified by a tuple $(i, S, P)$ where

- $i \in I$ is a node in the tree decomposition,
- $S \subseteq B_i$ is a subset of portals of the subtree $T_i$, and
- $P$ is a partition of $S$.

Let us denote by $V_i$ the vertices corresponding to the subtree $T_i$, i.e., $V_i := \cup_{T \in T_i} B_i \cup {v}$. A dynamic programming entry $DP(i, S, P)$ takes up the least cost of building a subgraph $H$ such that

- $H$ uses only the edges whose both endpoints are in $V_i$,
- $H$ connects the vertices in each set $P_j$ of the partition $P = \{P_1, P_2, \ldots, P_m\}$,
- $S$ is the subset of $B_i$ whose penalty is not paid, moreover, if a vertex $v \in V_i$ is not connected to $S$ via $H$, then its penalty $\pi(v)$ is paid in the total cost.

The final solution to the problem can be found as $\min_S DP(r, S, \{S\})$ where $r$ is the root of the tree decomposition, i.e., it does not matter which subset of the bag of the root is picked as long as they form a single component.

The DP entries are easy to compute for leaves: let $B_i = \{v\}$ for a leaf $i$. There are two possibilities: $DP(i, \emptyset, \emptyset) = \pi(v)$ and $DP(i, \{v\}, \{\{v\}\}) = 0$. The update procedure works as follows for different tree nodes:

**Introduce node** $i$ is the parent of $i'$, and we have $B_i = B_{i'} \cup \{v\}$. Then, $DP(i, S, P) = \pi(v) + DP(i', S, P)$ if $v \not\in S$. Next consider an entry $DP(i, S, P)$ such that for $v \in S$ and $P = \{P_1, P_2, \ldots, P_m\}$ where $v \in P_1$. Let $P' := \{P_1 \setminus \{v\}, P_2, \ldots, P_m\}$ and let $d$ be the distance of $v$ to the set $P_1 \setminus \{v\}$. The dynamic programming sets $DP(i, S, P) = d + DP(i', S \setminus \{v\}, P')$. 

26
**Forget node**  
$i$ is the parent of $i'$, and we have $B_{i'} = B_i \cup \{v\}$. Then,

$$\text{DP}(i, S, P) = \min \left[ \pi(v) + \text{DP}(i', S, P), \right.$$  

$$\left. \min_{P'} \{ \text{DP}(i', S \cup \{v\}, P') : P' \text{ is formed by adding } v \text{ to a set of } P \} \right].$$

The first term considers the case where we pay the penalty for $v$ and do not connect it in the final Steiner tree, whereas the second term takes into account the case where $v$ is connected to each connected component of the partition.

**Join node**  
The node $i$ has two children $i_1$ and $i_2$ with the same bags. We set $\text{DP}(i, S, P)$ to

$$\min_{P_1, P_2} \{ \text{DP}(i_1, S, P) + \text{DP}(i_2, S, P) - \pi(B_i \setminus S) \},$$

where the minimization goes over all pairs $P_1$ and $P_2$ whose connectivity implies that of $P$. The last term in the minimum operand is for canceling the double charging of the unsatisfied terminals of $B_i$.

It is not difficult to verify that the algorithm produces the correct output, and we defer the proof to the full version of the paper. The running time of the algorithm is polynomial in the number of DP entries, and the latter is at most $n \cdot 2^k \cdot k^k$. Since $k$ is a constant, the running time is a polynomial.

To extend the algorithm to PCTSP, the DP state is modified to $(i, \mathcal{P})$ where $i \in I$ is a node of the tree decomposition, and $\mathcal{P}$ is a set of pairs of vertices in bag $B_i$. A pair $s, t$ implies that there is a path between $s$ and $t$ in the subsolution, but the two nodes should be extended from outside the subtree $T_i$ to make a tour. The final solution is stored in $\text{DP}(r, \{(r, r)\})$. The algorithm for PCS works in the same way except that the final solution can be founded in $\min_{s, t \in B_r} \text{DP}(r, \{(s, t)\})$ since we do not need to have a closed tour.