A PTAS for planar group Steiner tree via spanner bootstrapping and prize collecting

Citation: Bateni, MohammadHossein, et al. "A PTAS for Planar Group Steiner Tree via Spanner Bootstrapping and Prize Collecting." STOC '16 Proceedings of the forty-eighth annual ACM symposium on Theory of Computing, 19-21 June, 2016, Cambridge, Massachusetts, ACM Press, 2016, pp. 570–83.

As Published: http://dx.doi.org/10.1145/2897518.2897549

Publisher: Association for Computing Machinery (ACM)

Persistent URL: http://hdl.handle.net/1721.1/115309

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

A PTAS for Planar Group Steiner Tree via Spanner Bootstrapping and Prize Collecting

MohammadHossein Bateni
Google Research
76 Ninth Avenue
New York, NY, USA
bateni@google.com

Erik D. Demaine*
MIT
32 Vassar Street
Cambridge, MA, USA
edemaine@mit.edu

MohammadTaghi Hajiaghayi†
University of Maryland
A.V. Williams Bldg.
College Park, MD, USA
hajiagha@cs.umd.edu

Dániel Marx‡
MTA SZTAKI
Kende utca 17
Budapest, Hungary
dmarx@cs.bme.edu

ABSTRACT

We present the first polynomial-time approximation scheme (PTAS), i.e., $(1+\varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$, for the planar group Steiner tree problem (in which each group lies on a boundary of a face). This result improves on the best previous approximation factor of $O(\log n(\log\log n)^{O(1)})$. We achieve this result via a novel and powerful technique called *spanner bootstrapping*, which allows one to bootstrap from a superconstant approximation factor (even superpolynomial in the input size) all the way down to a PTAS. This is in contrast with the popular existing approach for planar PTASs of constructing lightweight spanners in one iteration, which notably requires a constant-factor approximate solution to start from. Spanner bootstrapping removes one of the main barriers for designing PTASs for problems which have no known constant-factor approximation (even on planar graphs), and thus can be used to obtain PTASs for several difficult-to-approximate problems.

Our second major contribution required for the planar group Steiner tree PTAS is a spanner construction, which reduces the graph to have total weight within a factor of the optimal solution while approximately preserving the optimal solution. This is particularly challenging because group Steiner tree requires deciding which terminal in each group to connect by the tree, making it much harder than recent previous approaches to construct spanners for planar TSP by Klein [*SIAM J. Computing* 2008], subset TSP by Klein [STOC 2006], Steiner tree by Borradaile, Klein, and Mathieu [*ACM Trans. Algorithms* 2009], and Steiner forest by Bateni, Hajiaghayi, and Marx [*J. ACM* 2011] (and its improvement to an efficient PTAS by Eisenstat, Klein, and Mathieu [SODA 2012]. The main conceptual contribution here is realizing that selecting which terminals may be relevant is essentially a complicated prize-collecting process: we have to carefully weigh the cost and benefits of reaching or avoiding certain terminals in the spanner. Via a sequence of involved prize-collecting procedures, we can construct a spanner that reaches a set of terminals that is sufficient for an almost-optimal solution.

Our PTAS for planar group Steiner tree implies the first PTAS for geometric Euclidean group Steiner tree with obstacles, as well as a $(2+\varepsilon)$-approximation algorithm for group TSP with obstacles, improving over the best previous constant-factor approximation algorithms. By contrast, we show that planar group Steiner forest, a slight generalization of planar group Steiner tree, is APX-hard on planar graphs of treewidth 3, even if the groups are pairwise disjoint and every group is a vertex or an edge.

*Supported in part by NSF Grant CCF-1161626, NSF grant IIS-1546108, and DARPA/AFOSR GRAPHS grant FA9550-12-1-0423.

†Supported in part by NSF CAREER award 1053605, NSF Grant CCF-1161626, NSF grant IIS-1546108, DARPA/AFOSR GRAPHS grant FA9550-12-1-0423, and a Google Faculty Research award.

‡Research supported by the European Research Council (ERC) grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph algorithms; F.2.2 [**Nonnumerical Algorithms and Problems**]: Routing and layout

General Terms

Algorithms, Design, Theory

Keywords

Approximation algorithm, PTAS, group Steiner tree, planar graphs

1. INTRODUCTION

The *Steiner tree* problem is one of the most fundamental problems in combinatorial optimization and network design with both practical and theoretical significance. In this classical problem which is one of the first problems shown NP-hard by Karp [23], given a weighted graph $G = (V, E)$ and a set of terminals T, the goal is to find a minimum-length tree such that all terminals are connected in the tree. The problem remains hard even on planar graphs [18]. There is a long sequence of papers giving approximation factors better than 2 (2 is simple via a reduction to minimum spanning tree) for this problem [6, 9, 22, 24, 30, 32, 34–36]; the current best approximation ratio is 1.386 [9].

Reich and Widmayer [31] introduced a natural and by now classic generalization of Steiner tree, namely the *group Steiner tree* problem: given are a graph G with edge weights, and a collection $g_1, g_2, \ldots, g_k$ of node (terminal) sets called *groups*. The goal is to find a minimum-weight connected subgraph of G that contains at least one node from each group. Reich and Widmayer [31] are especially motivated by wire-routing phase of VLSI design, in which a *net* is a set of pins on the boundaries of various components that must be connected and for each component, there is flexibility as to the location of the pin used that we should exploit. As Demaine, Hajiaghayi, and Klein [15] observe, in the original VLSI design application of Reich and Widmayer [31], the elements of a single group are all located on the boundary of a component which occupies a (connected) region on the plane. Thus in this real-world application we need to solve an instance of the *planar group Steiner tree* problem. In planar group Steiner tree given a planar embedded graph G with edge weights, and a collection of groups $g_1, g_2, \ldots, g_k$ and corresponding distinct faces $f_1, f_2, \ldots, f_k$ of G, such that the nodes belonging to each group g_i lie on the boundary of the corresponding face f_i, the goal is to find a minimum-weight connected subgraph of G that contains at least one node from each group. Equivalently, we can assume that g_i, for $1 \leq i \leq k$, consists of the nodes on the boundary of f_i.

Much research has gone into finding good approximation algorithms for (planar) group Seiner, which itself is a very important special case of directed Steiner tree, i.e., Steiner tree on directed graphs. For general graphs, the best approximation ratio known to be achievable in polynomial time [19] is $O(\log^3 n)$, and for trees, the best known is $O(\log^2 n)$. Even when the host graph is a tree and hence planar, but the groups are not necessarily faces, the problem cannot be approximated better than $\Omega(\log^{2-\varepsilon} n)$ unless NP admits quasipolynomial-time Las Vegas algorithms [21]. It would thus appear that restricting the input to planar graphs cannot lead to substantially improved approximation. The best approximation factor for planar group Steiner tree is $O(\log n (\log \log n)^{O(1)})$ due to Demaine, Hajiaghayi, and Klein [15].

In this paper, we present the first polynomial-time approximation scheme (PTAS), i.e., $(1 + \varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$, for the planar group Steiner tree problem via a novel and powerful approach called *spanner bootstrapping*. In addition to our novel bootstrapping approach, we also need to first construct a spanner for planar group Steiner tree. In particular, deciding which terminal in a group is the one to participate in an optimal solution makes this task much harder than previous and recent approaches to construct spanners and thus obtain PTASs for planar TSP by Klein [26], subset TSP by Klein [25], Steiner tree by Borradaile, Klein, and Mathieu [8], and Steiner forest by Bateni, Hajiaghayi, and Marx [5] (and its improvement to an efficient PTAS by Eisenstat, Klein, and Mathieu [16]). Last but not least, we show planar group Steiner forest, a slight generalization of planar group Steiner tree in which the goal is to find a forest of minimum length that connects pairs of given group terminals is APX-hard on planar graphs of treewidth 3, even if the groups are pairwise disjoint and every group is a vertex or an edge. This is in sharp contrast with planar Steiner forest, an immediate generalization of Steiner tree, which has a PTAS as well [5].

1.1 Improvements for Geometric Group Steiner Tree and Group TSP

Also motivated in part by the VLSI application of Reich and Widmayer [31], Mata and Mitchell [27] consider the following problem: given a set of n polygonal regions in the plane, find a tour that visits at least one point from each region. They describe this problem as a special case of the problem *TSP with neighborhoods* (also called *group TSP*). They give a polynomial-time $O(\log n)$-approximation algorithm. Because the tour contains a spanning tree, and doubling each edge of a tree yields a tour, it is also an approximation algorithm for group Steiner tree where the groups are the polygonal regions. Gudmundsson and Levcopoulos [20] gave a faster algorithm for this problem. On the lower-bound side, unless P = NP, no constant-factor approximation is possible for disjoint disconnected regions, and no $(2 - \varepsilon)$-approximation is possible for (nondisjoint) connected regions [33].

Arkin and Hassin [1] gave constant-factor approximation algorithms for the special cases of parallel unit-length line segments, translated copies of a polygonal region, and circles. Mitchell [28] gave a PTAS for group TSP when the groups are disjoint and "fat." Most recently, Mitchell [29] gave a constant-factor approximation for group TSP when the groups are disjoint and connected.

An important difference in our problem is that we cannot route through groups, because faces serve as obstacles, whereas the geometric problem allows routing through groups. This difference seems to make the problem harder to approximate according to the literature. Nevertheless since planar graphs can capture plane metrics by a standard mapping of each point in the plane to a point of a grid with small size cells (in terms of ε), our PTAS for planar group Steiner tree gives a PTAS for plane group Steiner tree with obstacles as well and thus a $(2+\varepsilon)$-approximation algorithm for group TSP with obstacles (by simply doubling each edge and take an Eulerian tour). This is in contrast with the result of Mitchell [29] whose constant approximation factor seems so large that according to him no attempt was made to compute it exactly. Indeed he conjectures that one should be able to obtain a $2 + \varepsilon$-approximation for group TSP and our results gives the same conjectured constant even for the problem of group TSP with obstacles. In addition, our graph approach has the advantage that planar graphs can capture metrics that are not captured by the Euclidean metric, useful, e.g., in the VLSI problem where the routing of a net must avoid obstacles and previously routed nets.

1.2 Results and Techniques

Our main result in this paper is a PTAS for the planar group Steiner tree problem.

Theorem 1. *For any constant $\bar{\varepsilon} > 0$, there is a polynomial-time $(1+\bar{\varepsilon})$-approximation algorithm for the planar group Steiner problem.*

By a standard mapping of each point in the plane to a point of a grid with small size cells (in terms of ε), we obtain the following corollary:

Corollary 2. *For any constant $\bar{\varepsilon} > 0$, there is a polynomial-time $(1+\bar{\varepsilon})$-approximation algorithm for geometric Euclidean group Steiner tree with obstacles as well as a $(2+\bar{\varepsilon})$-approximation algorithm for group TSP with obstacles.*

The proof of Theorem 1 builds on the (by now) standard steps of designing planar approximation schemes using spanners. However, we contribute two major new conceptual ideas to this framework: spanner bootstrapping and constructing a planar group Steiner spanner via prize collecting.

Approximation Schemes via Spanners: Old and New Steps.

Previous work (e.g., [4, 5, 8, 13, 25]) formulated a general recipe for designing approximation schemes for planar graph problems. Briefly, this framework consists of the following steps:

- **Step 1: Constant-factor approximation.** The first step is to compute a constant-factor approximation. In many cases, such an approximation is known to be obtainable in polynomial time even on general graphs.
- **Step 2: Spanner construction.** Next the initial constant-factor approximation is extended into a subgraph satisfying two properties: (1) there is a $(1+\varepsilon)$-approximate solution using only the edges of the subgraph and (2) the total weight of the edges in the subgraph is at most $f(1/\varepsilon)$ times the weight of an optimum solution, that is, it is still a constant-factor approximation. Subgraphs satisfying these properties are usually called *spanners* in the PTAS literature (the name comes from the apparent similarity to distance spanners). If our goal is to find a $(1+\varepsilon)$-approximate solution, then it is sufficient to solve the problem restricted to the spanner.
- **Step 3: Treewidth reduction via shifting.** A simple shifting argument, going back to the work of Baker [2], allows us to reduce the treewidth of the spanner to a constant depending on δ, at the cost of increasing the optimum cost by at most an δ-fraction of the weight of the spanner. As the spanner is a constant-factor approximation of the optimum, it is possible to perform this step with a sufficiently small δ so that the optimum changes only by at most a factor of $1+\varepsilon$.
- **Step 4: Solving bounded-treewidth instances.** Finally, after reducing the treewidth of the spanner to a constant depending only in $1/\varepsilon$, we can solve the instance restricted to the spanner using standard dynamic programming techniques.

We make two important contribution to this framework. When trying to apply these steps to give a PTAS for planar group Steiner tree, we run into a difficulty already at the first step: unlike for the ordinary Steiner tree problem, there is no constant-factor approximation known for group Steiner tree. We get around this difficulty by using the spanner construction itself to obtain better and better approximation, alleviating need for an initial constant-factor approximation.

Spanner bootstrapping. Given an arbitrary initial solution, we can use the spanner construction to obtain a solution whose cost is higher than the optimum by at most ε times the cost of the initial solution. If this results in a solution that is better than the initial solution, we can repeat the process. Otherwise, we can conclude that the initial solution is a constant-factor approximation, and hence proceed with Steps 2–4 above.

As we discuss below, this technique is very general and could potentially be applied to a large number of problems. It can be expected that it will become the natural opening step for the design of planar approximation schemes.

Having avoided the need for a constant-factor approximation using spanner bootstrapping, our goal is to generalize the planar Steiner tree spanner construction to planar group Steiner tree. That is, we want to extend the initial solution into a spanner such that there is an almost-optimal solution using only the edges of the spanner and the total cost of the spanner is at most a constant factor higher than the cost of the initial solution. However, this task is fundamentally different and more difficult for the group Steiner tree problem. The difficulty stems from the fact that we do not know which terminals of the groups are reached by an optimum solution. Thus it is not sufficient that the spanner contains at least one terminal from each group, we have to make sure that the spanner contains the set of terminals reached by some almost-optimal solution. On the one hand, the spanner cannot afford to reach every terminal of every group, as the cost of such a subgraph may be prohibitively high. On the other hand, omitting even one terminal from the spanner may have dire consequences on the cost of the optimum when the instance is restricted to the spanner. Therefore, we have to carefully weigh the costs and benefits of reaching certain (sets of) terminals. Our second main contribution is demonstrating that this task of choosing which terminals should be reached is essentially a prize-collecting problem in its nature.

Discovering and reaching relevant terminals with prize collecting. We define a potential function on (sets of) terminals, giving an upper bound on the cost of missing that terminal in the solution. We say that a tree is *cheap* if its cost is at most $1/\varepsilon$ times the potential of the terminals it reaches. We extend the spanner with a collection of cheap trees in a systematic way. We argue that any solution can be modified such that it reaches only terminals on the spanner and the cost increases only at most by a factor of $1+\varepsilon$. The modifications can be charged on certain subtrees of the solution if these trees are not cheap. Otherwise, if they are cheap, then we would have added them to the spanner.

There are several technical difficulties that need to be addressed in the implementation of this idea. In fact, a large part of the paper is devoted to working out various versions of this scheme: defining appropriate potential functions, analyzing how the solution can be modified to avoid terminals,

defining the appropriate notion of cheap trees, and so on. Prize-collecting is a recurring theme of the proofs: we make a decision on whether or not to extend the spanner with a tree based on comparing the cost of the tree with the total prize (potential) collected by the tree.

Spanner Bootstrapping: More Details and Formal Definition.

Let us discuss spanner bootstrapping in more detail. The idea of using a constant-factor approximation to the problem as a "backbone" and then taking an ε-fraction of this backbone in our final solution has been initially introduced by Demaine and Hajiaghayi (SODA'05) [11] to obtain a PTAS for planar connected dominating set. Later Borradaile, Klein, and Mathieu (SODA'07 & TALG'09) [8] not only used a constant-factor approximate solution as a backbone but also managed to construct a spanner whose total size is still linear in optimum to obtain a PTAS for planar Steiner tree. However, in both cases above after constructing a backbone/spanner, we take an ε-fraction of the backbone/spanner along with an optimum solution in a "reduced" graph instance of bounded treewidth as our final solution. This introduces an error that is ε times the weight of the spanner. Since the bound for treewidth is linear in $O(1/\varepsilon)$, this approach works only when the starting backbone is a constant-factor approximation of optimum solution (or in some special cases a logarithmic approximate solution) to obtain a (quasi-)polynomial time approximation scheme. Our new technique of spanner bootstrapping removes this main barrier of having a constant approximation factor (or at most logarithmic in very special cases) to begin with.

We now state the theorem on the provable guarantee of our bootstrapping approximation method more formally. But first we need some definitions.

Consider an optimization problem P on weighted (undirected) graphs[1] where a *solution* is a set S of edges, and the *cost* or *length* of a solution S is the sum of the edge weights: $\mathrm{len}(S) = \sum_{e \in S} w(e)$. Also let $\mathrm{len}(G)$ denote the sum of all edge weights in a graph G. Let $\mathsf{OPT}(G)$ denote the minimum cost over all solutions on the graph G.

Problem P is *closed under deletion* if deleting an edge from the graph never increases OPT: i.e., $\mathsf{OPT}(G-e) \le \mathsf{OPT}(G)$ for any edge e of G. Problem P is *closed under contraction* if contracting an edge from the graph never increases OPT: i.e., $\mathsf{OPT}(G/e) \le \mathsf{OPT}(G)$ for any edge e of G. Problem P is *ω-undoable under deletion* if there is a polynomial-time algorithm that, given a subset X of edges in G and given a solution S' for $G - X$, finds a solution to G with length at most $\mathrm{len}(S') + \omega\,\mathrm{len}(X)$ for some constant $\omega \ge 0$. Problem P is *ω-undoable under contraction* if there is a polynomial-time algorithm that, given a subset X of edges in G and given a solution S' for G/X, finds a solution to G with length at most $\mathrm{len}(S') + \omega\,\mathrm{len}(X)$ for some constant $\omega \ge 0$.

Define a *relative $\beta(\delta)$-spanner construction* for a problem P to be an algorithm that, given an edge-weighted graph G in a minor-closed family $\mathcal{G}$, given a solution S to P on G, and given a constant $\delta > 0$, constructs an edge-weighted graph $G' \in \mathcal{G}$ such that the following two properties hold.

Spanning property: $\mathsf{OPT}(G) \le \mathsf{OPT}(G') \le \mathsf{OPT}(G) + \delta\mathrm{len}(S)$. Furthermore, any solution S' to G' can be converted in polynomial time into a solution S'' of G of length no more than $\mathrm{len}(S') + \delta\mathrm{len}(S)$.

Shortness property: $\mathrm{len}(G') \le \beta(\delta) \cdot \mathrm{len}(S)$.

Typically, G' is a subgraph of G for problems closed under deletion, and this property explains why $\mathsf{OPT}(G) \le \mathsf{OPT}(G')$. On the other hand, for problems closed under contraction, G' is usually the result of contracting certain edges in G, leading naturally to the same inequality.

The function $\beta(\delta)$ is typically a fixed but exponential function. However, in rare cases it can rely on n as well. In fact, our warm-up construction gives a $\beta(\delta)$ that has an $O(\log n)$ factor.

Also note that the above definition is a generalization of ideas used in previous work, e.g., [4, 5, 8, 13, 25]. In particular, the upper bound on $\mathsf{OPT}(G')$ is at least $(1+\delta)\ \mathsf{OPT}(G)$; in case the starting solution S were an $O(1)$-approximate solution, G' could be constructed to have a solution of length at most $(1+\delta)\ \mathsf{OPT}(G)$.

We call this construction a *relative* spanner since unlike previous work its spanning property can charge to the initial solution. Therefore, if the starting solution is not constant-approximate, the relative spanner does not guarantee the existence of a $1+\varepsilon$ approximate solution. Nevertheless, we may drop the qualifier "relative" when it is clear from the context.

In what follows we show how the relaxed version of the spanning property is sufficient for obtaining PTASs. Intuitively, spanner-based PTAS techniques are used to construct better solutions in each iteration, and finally reach at the desired solution.

Metatheorem 3 (Bootstrapping Approximation)**.** *Any problem P with the following properties admits a PTAS on graphs excluding any fixed minor.*

1. *P has a $\beta(\delta)$-spanner construction.*
2. *P is closed and ω-undoable under either deletion or contraction.*
3. *P has an α_0-approximation algorithm, where $\alpha_0 \le 2^{n^{O(1)}}$. (Typically $\alpha_0 = O(n)$.)*
4. *P has a polynomial-time algorithm, or even a PTAS, on graphs of constant treewidth.*

The resulting PTAS makes $O(\log \alpha_0)$ calls to the relative spanner construction routine with $\delta = O(\varepsilon)$. In particular, if $\alpha_0 = n^{O(1)}$, then this is $O(\log n)$ calls. The PTAS calls the bounded-treewidth algorithm with graphs of treewidth $O(\omega\beta(\delta)/\varepsilon)$. In particular, if the bounded-treewidth algorithm is fixed-parameter tractable with respect to treewidth, then the PTAS is efficient, running in $f(1/\varepsilon)n^{O(1)}$ time.

We prove Metatheorem 3 in Section 2.

Planar Group Steiner Spanner.

In conjunction with our novel bootstrapping approximation approach, we also need to construct a (relative) spanner for planar group Steiner tree. In particular, deciding which terminal in a group is the one to participate in an optimal solution makes this task much harder than previous approaches to construct spanners.

[1] In this section, we use the term "graph" for simplicity, but the framework applies equally well to graphs with additional structure (in particular, a set of terminal vertices) provided we can define how this structure is maintained under edge deletion or contraction (which for terminals is straightforward).

Before diving into the discussion for the full relative spanner construction, we go over a special case to illustrate some of the ideas. This warm-up exercise focuses on the case where each group consists of at most two vertices. Surprisingly, this seemingly benign special case, either, did not have any PTAS prior to our work. Beside illustrating some of the ideas in the general algorithm, the treatment of the special case serves to show why the main algorithm is fairly complicated.

Theorem 4. *There is a relative $2^{o(\delta^{-7.5})} \log n$ spanner construction for planar group Steiner tree if no group in the given instance has more than two vertices.*

The spanner construction above, though much simpler than the general case, is still not trivial. We present this special case partly because it is the cleanest example of the prize-collecting nature of the problem. After defining an appropriate submodular potential function, we can express the problem in the framework of submodular prize-collecting clustering and take advantage of a submodular prize-collecting clustering algorithm developed by Bateni et al. [4]. Unfortunately, such a compact presentation of the algorithm does not seem to be possible in the general case. Therefore, in the general spanner construction, instead of reducing the problem to submodular prize-collecting clustering, we employ more problem-specific (and more technical) arguments.

The $O(\log n)$ factor in the length of the spanner affects the running time of the whole algorithm. Due to this factor, we need to solve instances of planar group Steiner tree on instances with treewidth bounded by $f(1/\varepsilon)\log n$ in the last step of the algorithm. Solving group Steiner tree *exactly* can be reduced to solving ordinary Steiner tree *exactly* by introducing an artificial new terminal for each group and connecting it to the original terminals with edges of very large cost (note that this is not an approximation-preserving reduction). It is not difficult to observe that this transformation increases treewidth at most by a constant factor if the terminals of each group lie on the boundary of a face. Therefore, the last step of the algorithm can be done by solving instances of Steiner tree on graphs of treewidth $f(1/\varepsilon)\log n$. Steiner tree on graphs of treewidth w can be exactly solved in time $2^{O(w)} \cdot n^{O(1)}$ [10, 17], which is $n^{f(1/\varepsilon)}$ in our case. This means that the resulting PTAS is not an efficient PTAS, i.e., the running time is not of the form $f(1/\varepsilon)n^{O(1)}$. (We emphasize, however, that our algorithm for the general is an EPTAS.)

The main result of the current paper is based on the following theorem.

Theorem 5. *For some function $f(\delta)$, there is a relative $f(\delta)$ spanner construction for planar group Steiner tree if groups $g_1, g_2, \ldots, g_k$ correspond to distinct faces $f_1, f_2, \ldots, f_k$ of the input graph.*

Starting with an initial solution, we extend it to through a series of steps into a spanner. We call these intermediate extensions "prespanners," as they do not have the spanning property yet, but hopefully they get closer and closer to it. To make the proof modular, we formalize a notion of "spanner extension step," describing a procedure that takes a prespanner having certain properties and improves it into a prespanner having certain other properties. The proof of Theorem 5 is divided into some number of spanner extension steps that are independent from each other and relate to each other only via a well-defined interface.

Hardness for group Steiner forest. Steiner forest is a generalization of Steiner tree where, instead of connecting every terminal with a tree, the task is to connect given pairs of terminals with a forest. This generalization of the problem can be significantly harder: the solution is not necessary connected, hence we have to decide in some way which pairs of terminals are served by which connected component of the solution. Nevertheless, it was possible to generalize the PTASs for planar Euclidean Steiner tree to Steiner forest [5, 7, 16]. This raises the obvious question whether our PTAS for planar group Steiner tree can be generalized to planar group Steiner forest, that is, where a given list of pairs of groups have to connected with a forest of minimum cost. We show that, unlike for the original Steiner tree problem, this is not the case if groups are involved. With an approximation-preserving reduction from vertex cover on 3-regular graphs, we prove that planar group Steiner forest is APX-hard. The hardness result holds even if the groups are very simple: each of them consists of a single vertex or the endpoints of an edge.

Theorem 6. *The planar group Steiner forest problem is APX-hard on planar graphs of treewidth 3, even if the groups are pairwise disjoint and every group is a vertex or an edge.*

2. SPANNER BOOTSTRAPPING: THE PROOF

In this section, we prove Metatheorem 3.

We start with an α_0-approximate solution S_0, and iteratively produce solutions $S_1, S_2, \ldots$ with approximation ratios $\alpha_1 \geq \alpha_2 \geq \ldots$, where the final solution is $(1+\varepsilon)$-approximate as desired. The first solution S_0 is simply the output of the α_0-approximation algorithm. Let us for simplicity assume that we have an exact algorithm for the bounded-treewidth case. At the end, we explain how the same argument works with a bounded-treewidth PTAS, too.

Given S_i, we apply the spanner construction on G with solution S_i, to obtain a graph G'; the parameter δ will be fixed later. Thus $\mathrm{len}(G') \leq \beta(\delta)\,\mathrm{len}(S_i)$. Let $\circ$ denote the operation (deletion or contraction) under which the problem is closed and ω-undoable. Then we apply deletion decomposition [14] or contraction decomposition [12] accordingly: for any parameter $k \geq 2$, in polynomial time we obtain disjoint edge sets $X_1, X_2, \ldots, X_k$ such that $G' \circ X_j$ has bounded treewidth for all $j = 1, 2, \ldots, k$. Then we apply the bounded-treewidth algorithm to solve the problem P exactly on $G' \circ X_j$ for all $j = 1, 2, \ldots, k$. For each such solution T'_j, we use undoability to construct a solution S'_j to P on G', and let S' be the best solution among them. Next we apply the spanning property of the spanner construction to S' to obtain a solution S_{i+1} for the original graph G.

The algorithm iterates this process in two phases. The first phase consists of $\log_2\lceil\alpha_0/8\rceil$ iterations, and sets $\delta = 1$ and $k = \lceil 4\,\omega\beta(\delta)\rceil$. The second phase consists of one iteration, and sets $\delta = \frac{1}{2}\varepsilon$ and $k = \lceil 16\,\omega\beta(\delta)/\varepsilon\rceil$. The total number of iterations is thus $t = \log_2\lceil\alpha_0/8\rceil + 1$. We claim that the resulting solution S_t is a $(1+\varepsilon)$-approximation.

To prove this claim, we compare the approximation factor α_{i+1} of the solution S_{i+1} to the approximation factor α_i of the solution S_i. We can rewrite the algorithmic part of the

spanning property algebraically as

$$\text{len}(S_{i+1}) \le \text{len}(S') + \delta\text{len}(S_i). \quad (1)$$

Thus

$$\text{len}(S_{i+1}) \le \text{len}(S') + \delta\text{len}(S_i)$$

by (1) of the spanning property,

$$= \min_j \text{len}(S'_j) + \delta\text{len}(S_i)$$

from definition of S',

$$= \min_j [\text{len}(T'_j) + \omega\text{len}(X_j)] + \delta\text{len}(S_i)$$

by ω-undoability,

$$= \min_j [\mathsf{OPT}(G' \circ X_j) + \omega\text{len}(X_j)] + \delta\text{len}(S_i)$$

since bounded-treewidth algorithm is optimal,

$$\le \min_j [\mathsf{OPT}(G') + \omega\text{len}(X_j)] + \delta\text{len}(S_i)$$

due to closure under $\circ$,

$$= \mathsf{OPT}(G') + \omega \min_j [\text{len}(X_j)] + \delta\text{len}(S_i)$$
$$\le \mathsf{OPT}(G) + \omega \min_j [\text{len}(X_j)] + 2\delta\text{len}(S_i)$$

by spanning property,

$$\le \mathsf{OPT}(G) + \omega \frac{1}{k} \sum_j [\text{len}(X_j)] + 2\delta\text{len}(S_i)$$

because minimum is less than average,

$$= \mathsf{OPT}(G) + \omega \frac{1}{k} \text{len}(G') + 2\delta\text{len}(S_i)$$

because decomposition partitions edges into X_j's,

$$\le \mathsf{OPT}(G) + \omega \frac{1}{k} \beta(\delta)\text{len}(S_i) + 2\delta\text{len}(S_i)$$

by shortness property of spanner construction,

$$\le \mathsf{OPT}(G) + \Big[\omega \frac{1}{k} \beta(\delta) + 2\delta\Big] \text{len}(S_i)$$
$$= \mathsf{OPT}(G) + \Big[\omega \frac{1}{k} \beta(\delta) + 2\delta\Big] \alpha_i \, \mathsf{OPT}(G)$$

since S_i is an α_i-approximation,

$$= \Big[1 + \big(\omega \frac{1}{k} \beta(\delta) + 2\delta\big)\alpha_i\Big] \, \mathsf{OPT}(G).$$

Therefore, step i reduces the approximation factor from α_i to $1 + \big(\omega \frac{1}{k} \beta(\delta) + 2\delta\big)\alpha_i$.

For simplicity, suppose we know the value of α_i at each iteration. Then we define the first phase of the algorithm to be when $\alpha_i \ge 8$, and set $\delta = \frac{1}{8}$ and $k \ge 8\,\omega\,\beta(\delta)$, to obtain $\alpha_{i+1} \le 1 + \frac{3}{8}\alpha_i \le \frac{1}{2}\alpha_i$. Hence the algorithm exits this phase after $t - 1 = \log_2\lceil \alpha_0/8 \rceil$ rounds. In the second phase, $\alpha_{t-1} \le 8$, and we set $\delta = \frac{1}{32}\varepsilon$ and $k \ge 16\,\omega\,\beta(\delta)/\varepsilon$. Thus $\alpha_t \le 1 + \big(\frac{1}{16}\varepsilon + \frac{1}{16}\varepsilon\big)\alpha_{t-1} \le 1 + \frac{1}{8}\varepsilon\alpha_{t-1} \le 1 + \varepsilon$. Therefore S_t is the desired $(1+\varepsilon)$-approximation.

In reality, though, we do not know the value of α_i during the algorithm to determine the parameters. Therefore we set $\delta = \frac{1}{32}\varepsilon$ and $k \ge 16\,\omega\,\beta(\delta)/\varepsilon$, which work for both phases (assuming $\varepsilon \le 1$).

If the bounded-treewidth algorithm is only a $(1+\gamma)$-approximation, then the term $(1+\delta)\,\mathsf{OPT}(G)$ grows to a factor not exceeding $(1+\delta)(1+\gamma)\,\mathsf{OPT}(G)$. Thus we can simply set $\delta = \gamma$ to one quarter the previous value, and still obtain a $(1+\varepsilon)$-approximation.

3. SPANNER CONSTRUCTION

We build on a procedure for constructing spanners, due to Klein [25] for TSP and Borradaile et al. [8] for Steiner tree, that is now well-known and has been employed to obtain several planar PTASs. The Steiner tree spanner construction crucially depends on having a *short* tree connecting up all the terminals. This can be found by any of the known $O(1)$-approximation algorithms.

Here for group Steiner tree (or TSP), we need to make a decision as to which terminal in each group should participate in the optimal solution. Notice that, had this been given to us, the problem would have reduced to Steiner tree. It suffices to identify and connect up the "correct" terminal of every group. We can select more than one terminal from a group as long as the total connection cost is not too large. On the other hand, we require the guarantee that the chosen terminals do lead us to a near-optimal solution.

The following is a simple corollary of the said constructions [8, 25] coupled with the above discussion. It essentially says that we can construct a polynomial-time $\beta(\delta)$-spanner for group Steiner tree or group TSP given a tree S that spans the "correct" terminal in each group. In particular, we say that S *reaches all relevant terminals* if a $(1+\delta)$-approximate group Steiner tree solution exists that only employs the terminals in S.

Theorem 7. *We can construct in polynomial time a $2^{o(\delta^{-7.5})}$ spanner given a promising starting point S.*

In what follows we demonstrate how to extend an initial solution in each iteration so that it reaches all relevant terminals. As mentioned before, we first delve into discussing a special case (where each group has at most two terminals) to illustrate many of the ideas and challenges. In this case, we can directly use a technique called PC-Clustering. The more general case is discussed next and in the appendix, and it requires several more involved techniques, in particular many other prize collecting procedures (PC-Clustering no longer being directly useful).

3.1 Groups with at Most Two Terminals

We present a method to extend a solution so that it reaches all relevant terminals in the the case when each group may consist of at most two vertices (but not necessarily on one face of the graph). We emphasize that we do not even require to have a planar input graph at this point, though the next steps of the PTAS (namely the reduction to the bounded-treewidth case and solving the latter) work only with extra assumptions on planarity and lying of group vertices on a single face.

Lemma 8. *Given an initial solution S to group Steiner tree (or group TSP), we can find in polynomial time a solution S' that reaches all relevant terminals.*

Based on a given solution S satisfying all the groups, we define a submodular function π on the groups. Then we use this as a penalty (or potential) function to run a (submodular) PC-Clustering [4]. The result, among other things, is a tree S' that connects certain terminals in addition to those in S. The guarantees of PC-Clustering along with the properties of our submodular penalty function π allows us to argue that S' reaches all relevant terminals.

Clearly all terminals for one-terminal groups fall on S. As for other groups, let us for simplicity assume that every other group has exactly one vertex on S—which we call the "anchor" (vertex or terminal) of the group—and another one that is not on S—which we denote the "tip" (vertex or terminal) of the group.

Fix a (not necessarily simple) path μ spanning anchors in S. Define a binary tree I on the anchors as follows where each node of the tree corresponding to a subpath of μ and contains a consecutive subset of anchors. The top-level node $r(I)$ consists of one interval corresponds to the entire μ, and contains all anchors. Each of the two nodes at the next level contains almost half the anchors, and corresponds to the subpath of μ from the first anchor to the last. Each node has its own subtree defined recursively in a similar fashion. Clearly the depth of the tree is logarithmic in the number of anchors.

Given a subset Y of groups, $I(Y)$ denotes the nodes in I, except $r(I)$, that contain the anchor of at least one group in Y. The width of a node i in $I(Y)$, denoted $w(i)$ is the length of the subpath corresponding to the parent of the node. We define $\pi(Y) = \sum_{i \in I(Y)} w(i)$.

Lemma 9. *The function π is a nonnegative, monotone, submodular function with an upper bound of $O(\log m)\text{len}(S)$ where m is the number of groups. Furthermore, $\pi(A \cup B) - \pi(B)$ for disjoint subsets of groups A, B suffices to connect the anchors of A to those in B. More specifically, there exists a forest F, spanning anchors of A, of length at most $\pi(A \cup B) - \pi(B)$ such that each component of F contains at least one anchor from B.*

Proof. Nonnegativity and monotonicity are trivial from the definition. That the function is submodular is derived easily from the diminishing returns property.

Note that at each level of $I(Y)$, all the subpaths are disjoint. Thus, the total width of nodes at one level is at most $\text{len}(\mu)$, hence the upper bound given the logarithmic bound on the depth of I.

It remains to prove the last part of the lemma. Let $\mu(i)$ for a non-root node i of tree I denote the subpath of μ corresponding to the parent of i in I. We define $F = \bigcup_{i \in I(A) \setminus I(B)} \mu(i)$, and claim that F satisfies the conditions set forth in the statement of the lemma. First note that the length of the forest is no more than $\pi(A \cup B) - \pi(B)$. Now consider an anchor a in A. Let $I_B(a)$ denote the node in I that contains a but no anchor from B; i.e., $I_B(a)$ is the topmost node of $I(\{a\}) \setminus I(B)$. By definition, the parent of $I_B(a)$ contains not only a but also at least one anchor from B. Clearly, the subpath of μ corresponding to the parent of $I_B(a)$ is part of the construction of F, hence F satisfies a. □

We invoke the submodular PC-Clustering algorithm due to [4] where there is a single demand corresponding to each group—that of connecting its tip to S. Let $\mathcal{D}$ denote the set of these demands. The potential function $\phi(D_Y)$ for a subset $D_Y \subseteq \mathcal{D}$ of demands corresponding to groups Y is set to $\phi(D_Y) = \varepsilon^{-1}\pi(Y)$. Below we provide the theorem summarizing the properties of the procedure as stated in [3].

Theorem 10. *Given an instance $(G, \mathcal{D}, \pi)$, Submodular PC-Clustering produces in polynomial time a forest F and a subset $\mathcal{D}^{\mathsf{unsat}} \subseteq \mathcal{D}$ of demands, along with a feasible vector y for Equations (2)–(4), such that*

1. *$y(\mathcal{D}^{\mathsf{unsat}}) = \pi(\mathcal{D}^{\mathsf{unsat}})$;*
2. *F satisfies any demand in $\mathcal{D}^{\mathsf{sat}} := \mathcal{D} \setminus \mathcal{D}^{\mathsf{unsat}}$; and*
3. *$\text{len}(F) \le 2y(\mathcal{D})$.*

$$\sum_{S : e \in \delta(S)} y_S \le c_e \qquad \forall e \in E \qquad (2)$$

$$\sum_{d \in D} y_d \le \pi(D) \qquad \forall D \subseteq \mathcal{D} \qquad (3)$$

$$y_{S,d} \ge 0 \qquad \forall d \in \mathcal{D}, S \subseteq V, |S \cap d| = 1. \qquad (4)$$

Let F' be the union of F and S, and let S' be the connected component of F' containing S. Next we prove the desired properties of S'.

Lemma 11. *The length of S' is $O(\varepsilon^{-1} \log m)\text{len}(S)$ where m is the number of groups.*

Proof. The increase in the length going from S to S' is at most $\text{len}(F)$. The latter according to the theorem above is at most twice the sum of the y variables, which is in turn no more than $\phi(R)$, where R is the set of groups. Lemma 9 guarantees this to be bounded by $O(\varepsilon^{-1} \log m)\text{len}(S)$. □

Lemma 12. *There exists a near-optimal solution using only the terminals in S'.*

We need the following claim in the proof of the lemma. The lemma appears in previous work, e.g., [3, Lemma 10.2.3] restated with the definition of π.

Claim 13. *The length of a tree T connecting the tips of a subset Z of groups to S is at least $y(Z)$.*

Proof of Lemma 12. We take the optimal solution and modify it so that it only uses the terminals on S' without increasing its length significantly. Let Z be the subset of groups whose tips do not lie in S'. Let $X \subseteq Z$ be the subset of such groups whose anchors do not lie on the optimal solution, hence the optimum uses their tips to satisfy them. Let $Y = Z \setminus X$. Take $\mathcal{D}^{\mathsf{unsat}}$ from Theorem 10. By definition $X \cup Y \subseteq \mathcal{D}^{\mathsf{unsat}}$. We have $y(\mathcal{D}^{\mathsf{unsat}}) = \phi(\mathcal{D}^{\mathsf{unsat}})$. Let $U = \mathcal{D}^{\mathsf{unsat}} \setminus Y$. We have $y(U) = y(\mathcal{D}^{\mathsf{unsat}} \setminus Y) = y(\mathcal{D}^{\mathsf{unsat}}) - y(Y) \le \phi(\mathcal{D}^{\mathsf{unsat}}) - \phi(Y)$, where the inequality follows from (3). The last part of Lemma 9 argues that the right-hand side term suffices for connecting the anchors of $\mathcal{D}^{\mathsf{unsat}} \setminus Y$ to the anchors of Y. Note that the anchors of Y are necessarily part of the optimum, hence this transformation produces a valid solution using only terminals in S'. To see the increase in the length is not significant, note that the left-hand side term, which is an upper bound on the additional length, is no more than the optimum. □

Proof of Lemma 8. The two lemmas above prove that S' reaches all relevant terminals. □

3.2 Groups with Any Number of Terminals

In the general case, the algorithm to construct such a relative spanner consists of several involved steps. Following earlier work, we find it convenient to "cut open" the initial solution and assume that the initial solution is a path. Then each group has one or more terminals on the prespanner (we call those terminals the *anchors*) and possibly some other terminals not on the prespanner (we call those terminals the *tips*). The anchors of a group span a subpath on the prespanner. We divide the groups in a solution into *nonminimal* and *minimal* groups according to whether or not the subpath spanned by the group contains the subpath of some other group. Somewhat counterintuitively, we prefer nonminimal groups, as there is a simple but surprisingly powerful way of assigning potentials to such groups. Thus a general goal in the spanner construction is to extend the prespanner in a way that (after cutting open again the extended prespanner) more and more groups become nonminimal. After some number of prize collecting procedures, we can define a potential function on every group, not only on the nonminimal ones. The main argument is that there should be a vertex of the solution close to the anchor of each minimal group, otherwise the solution contains a path connecting a tip of the group to a vertex of the spanner far away from the anchor of the group. But adding such a path would make the group nonmimimal, and we would have added such a path to the prespanner in one of the prize-collecting procedures.

Having obtained a potential function for every group, we try to make the task of satisfying the different groups more independent. Removing the edges of the prespanner from the solution breaks the solution into a set of trees, which we call the *fragments* of the solution. The first difficulty that we want to overcome is that a fragment of the solution may reach the tips of more than one nonminimal group. First we show how to ensure that each fragment of the solution reaches the tips of at most a constant number of terminals, and then we reduce this constant to one. In fact, every remaining tip is *weakly isolated,* meaning that there is no way of connecting the tips of two groups without crossing the prespanner. We further strengthen this to *strongly isolated,* where (roughly speaking) from each tip we can reach only a consecutive subpath of the spanner. Then the problem we have to overcome becomes very similar to a situation that is handled in previous work [8]: given two paths, we have to find a small number of portals on each of them such that any tree connecting them can be massaged to use only these portals. Finally, we invoke the known spanner construction for Steiner tree as a black box on top of our already constructed prespanner to construct an actual relative spanner. Our metatheorem applies these steps a polynomial number of times to obtain the desired PTAS for planar group Steiner tree.

To make our proof for the case of groups with any number of terminals modular, we split the construction of the spanner into several independent steps. We formalize the notion of "spanner extension step," and describe each step in a separate section. Informally, a spanner extension step takes a prespanner subgraph satisfying a certain list of properties, and it produces a new instance with a prespanner subgraph satisfying another (hopefully more useful) list of properties. The construction has to satisfy two important properties: the value of the optimum solution in the original and the constructed new instance differ only by at most ε times the total weight of the prespanner and the weight of the prespanner increases only by a constant factor depending on ε. Thus if we construct a spanner through a sequence of spanner extension steps such that the first prespanner that we started with was a constant-factor approximation of the optimum solution, then the error introduced during the spanner extension steps can be made arbitrary small compared to the value of the optimum. This means that a $1+\varepsilon$ approximation of the new instance is sufficient to obtain a $(1+O(\varepsilon))$-approximation of the original instance.

Sometimes we need to state that not only the spanner has certain properties, but there are (almost) optimal solutions satisfying certain properties. Therefore, we define spanner extension steps in a way that if the original instance has a solution satisfying a certain list of properties, then the new instance has a solution satisfying a certain other list of properties and its weight is not much larger.

Formally, let $\mathcal{P}$ be an optimization problem, where every instance I contains an edge-weighted graph G (and perhaps other information, such as list of terminals etc.). We also assume a weight function λ on the edges (possibly having parallel edges). Let $\mathcal{G}$ be the set of all groups.

Definition 14. *A* $(\mathsf{P1},\mathsf{P2}) \to (\mathsf{P1}',\mathsf{P2}')$ spanner extension step for $\mathcal{P}$ *is an algorithm* $\mathbb{A}$ *satisfying the following.*

1. *The input is a value* $0<\varepsilon<1$*, an instance* I *of* $\mathcal{P}$*, and a subgraph* L *of the graph* G *of* I *such that* (I,L) *satisfies property* $\mathsf{P1}$*.*
2. *The running time is* $f(1/\varepsilon)n^{O(1)}$ *for some computable function* f*, where* n *is the size of the input instance* I*.*
3. *The output is an instance* I' *of* $\mathcal{P}$ *and a subgraph* L' *of the graph* G' *of* I' *such that* (I',L') *satisfies property* $\mathsf{P1}'$*, and the following hold:*
   (a) $\lambda'(L') \le h(1/\varepsilon)\lambda(L)$ *for some computable function* h*.*[2]
   (b) *Given a solution* X' *of* I'*, one can find in time* $g(1/\varepsilon)n^{O(1)}$ *a solution* X *of* I *with* $\lambda(X)\le\lambda(X')$ *for some computable function* g*.*
   (c) *There exists a* $c\ge 0$ *such that if* I *has a solution* X *satisfying* $\mathsf{P2}$*, then* I' *has a solution* X' *satisfying* $\mathsf{P2}'$ *and having* $\lambda'(X')\le\lambda(X)+c\varepsilon\lambda(L)$*.*

As the running time is $f(1/\varepsilon)n^{O(1)}$, the size of the instance I' can be also bounded by $f(1/\varepsilon)n^{O(1)}$. We may refer to function f as the *runtime for the transformation,* to function h as the *blowup of the spanner,* to function g as the *recovery time,* and to constant c as the *recovery increase.*

Lemma 15. *If problem* $\mathcal{P}$ *has a* $(\mathsf{P1},\mathsf{P2})\to(\mathsf{P1}',\mathsf{P2}')$ *spanner extension step and a* $(\mathsf{P1}',\mathsf{P2}')\to(\mathsf{P1}'',\mathsf{P2}'')$ *spanner extension step, then* $\mathcal{P}$ *also has a* $(\mathsf{P1},\mathsf{P2})\to(\mathsf{P1}'',\mathsf{P2}'')$ *spanner extension step.*

Our main goal is to construct an instance (via several successive spanner extension steps) where the groups have no tips: every terminal is on the spanner. In this case, we can invoke the previous work of Borradaile, Klein, and Mathieu [8], which implies the existence of the following spanner extension step.

[2]For a function such as $\lambda\colon X\to\mathbb{Z}^+$, we define $\lambda(X')=\sum_{x\in X'}\lambda(x)$ for $X'\subseteq X$ the usual way.

Lemma 16 ([8]). *There is a* $(\mathsf{P10}, -) \to (\mathsf{P11}, -)$ *spanner extension step for planar group Steiner tree, where*

[P10]: *Every terminal is on* L.
[P11]: $L = G$.

That is, this spanner extension step creates a new instance where the spanner is actually the entire graph. Standard treewidth-based techniques can be used to solve the an instance of planar group Steiner tree withing an additive $\varepsilon\lambda(G)$ term of the optimum solution. As it remains true during the successive applications of the spanner extension steps that the spanner is a constant-factor approximation of the optimum, this results in a $(1 + O(\varepsilon))$-factor approximation.

In the rest of this extended abstract, we give a brief tour of all the spanner extension steps appearing in our spanner construction. Fist we define four basic properties P0–P3 that will be required for some of the spanner expansion steps. We argue that the instance can be easily modified to achieve these properties whenever we need them. Then we go on to define further properties representing the intermediate goals in our construction; see Figure 1 for an overview. The detailed description of the spanner extension steps and their analysis will appear in the full version of the paper.

3.2.1 Property P0: Some Basic Conditions

Property P0 formalizes some basic properties of the prespanner that we need in most of the proofs. Formally, a prespanner L is a subgraph $L \subseteq G$ given in the input such that every group $x \in \mathcal{G}$ has at least one terminal on L. Furthermore, we require the following additional properties (note that here the spanner L in general is not necessarily a path):

- If a vertex of group x is on L, then it is a terminal of x.
- If a vertex v of x is in $V(L)$, then v has no edges outside $E(L)$ besides the two infinite edges of f_x.
- If a vertex v of x is not in $V(L)$, then v has degree at most 3: it has the infinite edges of f_x plus additionally one more edge.

Given any instance with a subgraph L containing at least one terminal of each group, it is easy to modify the instance to satisfy these additional properties.

3.2.2 Property P1: The Prespanner is a Path

In a large part of the proof, we assume that the prespanner is a path. The following properties and related definitions, denoted by P1 formalize the situation that the prespanner L is a path:

- The prespanner is a path L with endpoints v_ℓ and v_r.
- We denote by $L[a, b]$ the subpath of L between a and b and by $\lambda(L[a, b])$ is its length.
- The terminals of a group on L are the *anchors* and the remaining terminals are the *tips.*
- There is a special group g_0 having v_ℓ and v_r as terminals, and the terminals of g_0 are all that distance 0 from each other.
- There is an edge of weight 0 connecting v_ℓ and v_r and every group is inside the cycle formed by L and this edge.

In previous spanner constructions [25], it was often assumed that the initial solution is a cycle. This can be achieved using the (by now standard) technique of "cutting open a tree." Using similar steps, we can modify the instance such that P1 holds (see Figure 2).

Let H be an arbitrary subgraph of G having at least one vertex on $V(L)$. We define $\mathrm{span}(H)$ to be the minimal subpath of L containing every vertex of $V(H) \cap V(L)$. If $x \in \mathcal{G}$, then by a slight abuse of notation, we let $\mathrm{span}(x)$ to be $\mathrm{span}(f_x)$, where f_x is the face whose vertices are terminals of x. Observe that the planarity of the graph and the connectedness of the groups imply that for every $x, y \in \mathcal{G}$, either $\mathrm{span}(x)$ and $\mathrm{span}(y)$ are disjoint or one is the subset of the other. We say that x is *minimal* if there is no other $y \in \mathcal{G}$ such that $\mathrm{span}(y) \subset \mathrm{span}(x)$; otherwise, we say that x in *nonminimal*. We denote by $\check{\mathcal{G}}$ the set of minimal groups (intuitively, they are local and compact) and by $\widehat{\mathcal{G}}$ the set of nonminimal groups (intuitively, they are wide).

3.2.3 Property P2: Minimal Groups Have Only One Anchor

Suppose that property P0 and P1 hold. With simple modifications, we can achieve the following without changing which groups are minimal and which are not:

Every minimal group has exactly one anchor.

Figure 4 shows that this property can be achieved by an easy modification of the instance.

3.2.4 Property P3: Nonminimal Groups Have Exactly Two Anchors

Suppose that property P0 and P1 hold. With simple modifications, we can achieve the following without changing which groups are minimal and which are not:

Every nonminimal group has exactly 2 anchors.

This property can be achieved in a way similar to how P2 was obtained, see Figure 4.

3.2.5 Goal 1: Potential Functions

The notion of potentials and potential functions will be crucial for our spanner construction. It is analogous to the function $\phi(D_Y)$ used in Section 3.1, with the important difference that here we define a potential for each group and the potential of a set of groups is simply the sum of the potentials in the set (whereas the function $\phi(D_Y)$ was a general submodular function that was not necessarily modular).

We define the *potential* of a group $x \in \mathcal{G}$ with respect to a solution X as the length of the shortest path P with $V(P) \subseteq V(L)$ that connects an anchor of x and a vertex of X. We say that $p : \mathcal{G} \to \mathbb{Z}^+$ is a *potential function compatible with* X if it is true that every $x \in \mathcal{G}$ has potential at most $p(x)$ with respect to X.

It turns out that it is not very difficult to define a potential function for the nonminimal groups in such a way that the function is compatible with every solution.

Lemma 17. *In polynomial time, we can find a function* $p\colon \widehat{\mathcal{G}} \to \mathbb{Z}^+$ *such that*

1. $p(\widehat{\mathcal{G}}) \leq 6\lambda(L)$ *and*
2. *for every solution* X *and every* $x \in \widehat{\mathcal{G}}$*, group* x *has an anchor that is at distance at most* $p(x)$ *from* X *on* L.

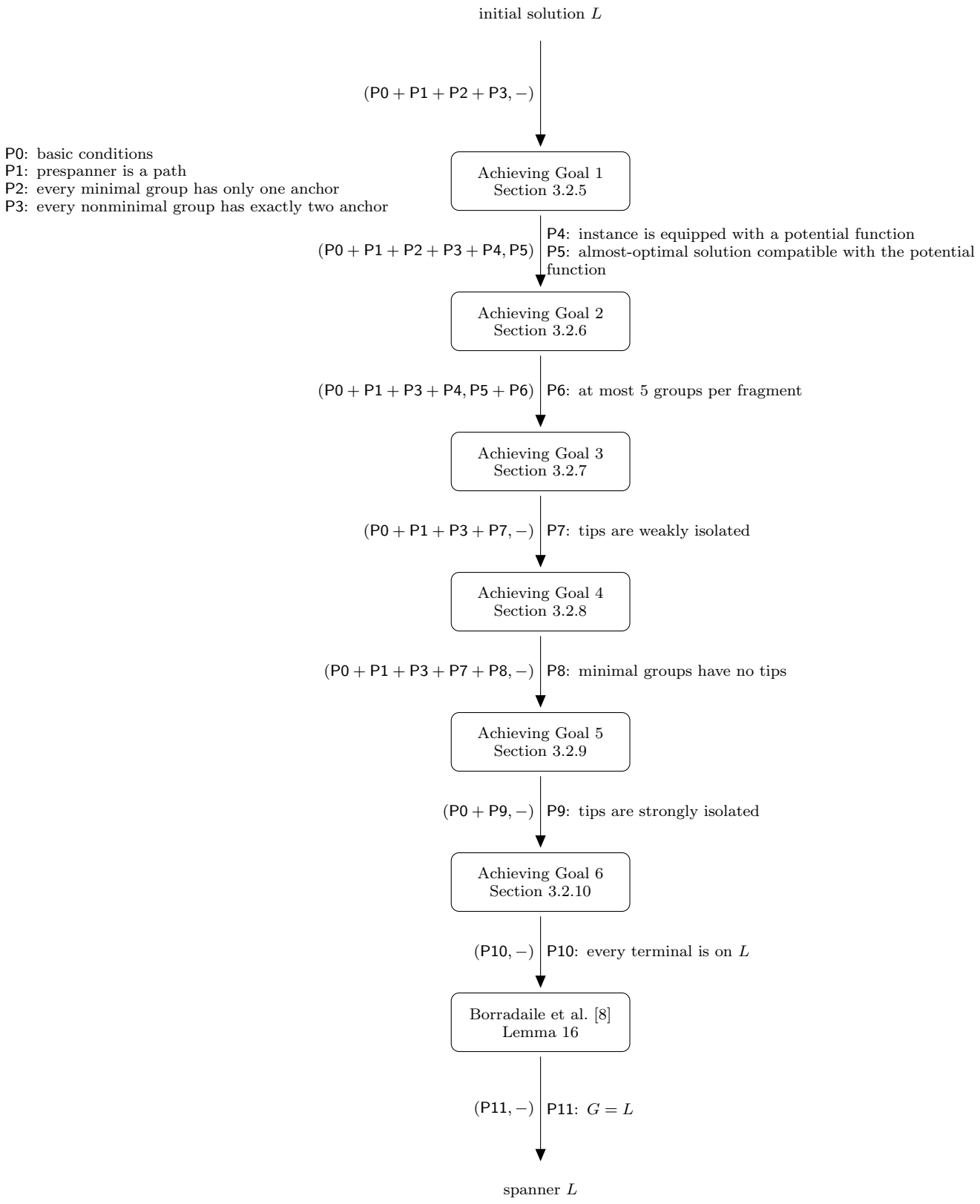

Figure 1: Overview of all spanner extension steps.

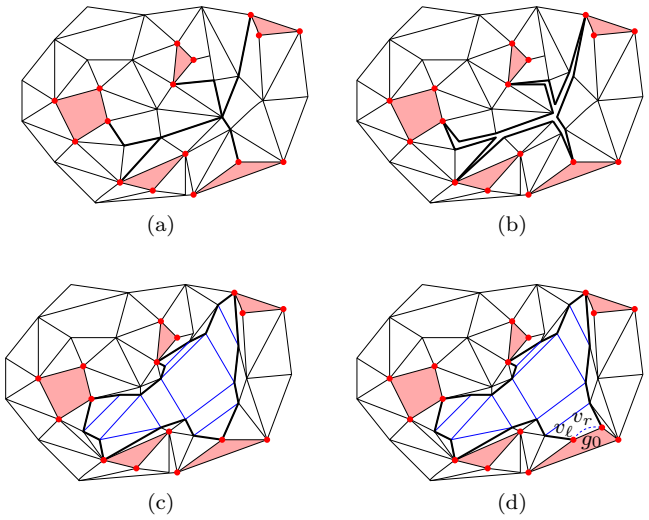

Figure 2: Cutting open a tree. (a) A tree containing one terminal from each group. (b) An Eulerian tour of the tree. (c) Cutting open the tree along the Eulerian tour by duplicating vertices and connecting them with edges of weight 0 (shown in blue). (d) Making the cycle a path by introducing an additional terminal to the group g_0, and introducing an edge of weight 0 between v_ℓ and v_r (shown by dashed lines).

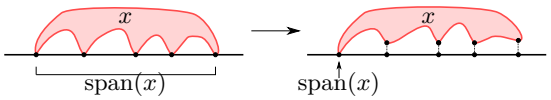

Figure 3: Decreasing the number of anchors of a minimal group to one.

The main observation in the proof is the following. Consider a nonminimal group $x \in \widehat{\mathcal{G}}$ such that span(x) is minimal, that is, span(x) does not contain the anchor of any other nonminimal group, but it does contain an anchor of a minimal group $y \in \widehat{\mathcal{G}}$ (as x is nonminimal). Now the only way for a solution X to reach the anchor or a tip of y is to enter the area enclosed by x and span(x), and this is only possible via span(x). Thus subpath span(x) of L is "safe" in the sense that every solution X contains at least one vertex of span(x). Thus to define the function p, all we need to do is to find, for every $x \in \widehat{\mathcal{G}}$ a subpath of L that includes both an anchor of x and such a "safe" subpath of P. Somewhat surprisingly, this can be done in a way that every edge of L is used only twice, which results in the bound $p(\widehat{\mathcal{G}}) = O(\lambda(L))$ (in some cases, we need a simple additional argument for those groups $x \in \widehat{\mathcal{G}}$ for which there is a unique maximal $y \in \widehat{\mathcal{G}}$ with span(y) $\subseteq$ span(x)). Figure 5 gives an example how these paths can be defined.

Defining the potential function for the minimal groups turns out to be significantly more difficult. In fact, we do not construct a potential function that is compatible with every solution: we achieve only the weaker goal of constructing a potential function that is compatible with *some* almost-optimal solution. The following lemma states this as a spanner extension step.

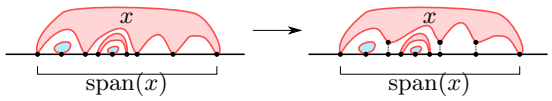

Figure 4: Decreasing the number of anchors of non-minimal groups to two without changing span(x).

Lemma 18. *There is a* $(\mathsf{P0}+\mathsf{P1}+\mathsf{P2}+\mathsf{P3},-) \to (\mathsf{P0}+\mathsf{P1}+\mathsf{P2}+\mathsf{P3}+\mathsf{P4},\mathsf{P5})$ *spanner extension step, where*

[P2]: *Every minimal group has only one anchor.*
[P3]: *Every nonminimal group has exactly two anchors.*
[P4]: *L is equipped with a potential function*
[P5]: *Solution X is compatible with the potential function*

3.2.6 Goal 2: At Most 5 Groups per Fragments

Let X be a solution. We may imagine X as a collection of trees attached to L, plus some subpaths of L itself. The following definitions are useful for this interpretation:

Definition 19. *Let G be a graph, H be a subgraph of G, and let T be a subtree of G. We say that T is* normal *with respect to H if $V(T) \cap V(H) \neq \emptyset$ and every vertex $v \in V(T) \cap V(H)$ has degree 1 in T. We say that a tree T is* singly attached *to H if T is H-normal and $V(T) \cap V(H) = 1$ holds.*

Definition 20. *Let G be a graph, H a subgraph of G, and T a subtree of G. A* fragment *of T with respect to H is a maximal normal subtree T' of T (i.e., no proper supergraph of T' is normal).*

Observe that the fragments of T with respect to H are pairwise edge disjoint and every edge of $T \setminus E(L)$ is in one of the fragments.

A particular difficulty of the proofs is that certain local parts of the solution may be important for than one group: an L-fragment may contain the tips of more than group. This has to be taken into account in charging arguments, where we are trying to charge a value related to a group on an L-fragment reaching a tip of that group. Some of these charging arguments work only if there are at most a constant number of groups per L-fragment. Our next goal is to achieve this. We modify the solution that every L-fragment contains the tips of at most 5 groups. The modification exploits the fact that the instance is equipped with a potential function.

Lemma 21. *There is a* $(\mathsf{P0}+\mathsf{P1}+\mathsf{P2}+\mathsf{P3}+\mathsf{P4},\mathsf{P5}) \to (\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P4},\mathsf{P5}+\mathsf{P6})$ *spanner extension step, where*

[P2]: *Every minimal group has only one anchor.*
[P3]: *Every nonminimal group has exactly two anchors.*
[P4]: *L is equipped with a potential function*
[P5]: *Solution X is compatible with the potential function*
[P6]: *Every L-fragment of the solution contains tips of a most 5 groups.*

3.2.7 Goal 3: Terminals are Weakly Isolated

To further simplify the instance, we would like to ensure that each L-fragment of the solution can contain the tip of at most one group. We ensure this in a very strong way: we

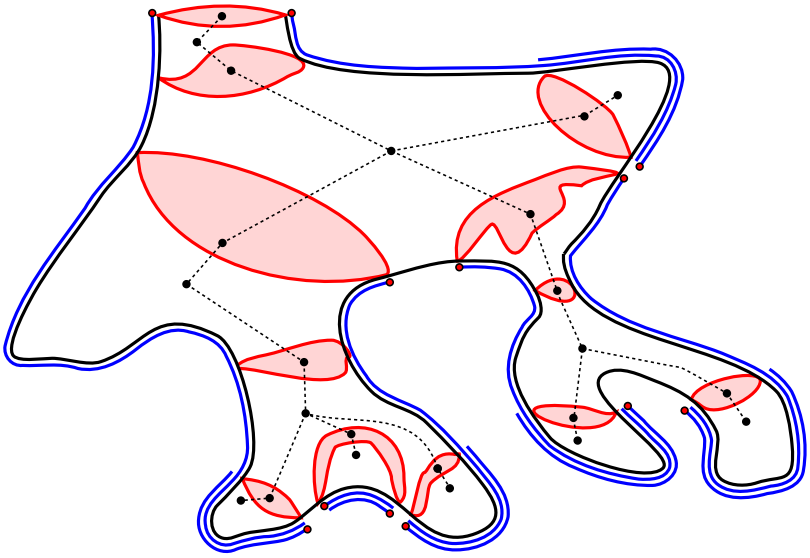

Figure 5: Lemma 17.

modify the instance such that there is a path disjoint from the prespanner between tips of two groups.

Definition 22. *We say that a terminal v of group x is* weakly isolated *in a graph H if there is no path Q with $V(H) \cap V(Q) = \emptyset$ connecting v with a terminal of a group $x' \neq x$.*

See Figure 6 for examples. Note that it is possible that some of the tips of a group are weakly isolated, and some others are not.

Using the assumptions that the instance is equipped with a potential function and that every L-fragment contains the tips of at most 5 groups makes it significantly easier to modify the instance in such a way that every group is weakly isolated.

Lemma 23. *There is a* $(\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P4}, \mathsf{P5}+\mathsf{P6}) \to (\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P7}, -)$ *spanner extension step, where*

- [P3]: *Every nonminimal group has exactly two anchors.*
- [P4]: *L is equipped with a potential function*
- [P5]: *Solution X is compatible with the potential function*
- [P6]: *Every L-fragment of the solution contains tips of a most 5 groups.*
- [P7]: *Every tip is weakly isolated in L.*

3.2.8 Goal 4: Minimal Groups Have No Tips

If the tips of the minimal groups are weakly isolated, then we can modify the instance in a way that the minimal groups have no tips. First, we extend the prespanner in a way that the tips are isolated in the following sense.

Definition 24. *We say that a terminal v of group x is* strongly isolated *in a graph H if there is a subpath P_x of the boundary of x going through v and a subpath P_H of H with the same endpoints as P_x such that P_H and P_x form a cycle with no terminal strictly in its interior.*

See Figure 6 for examples. Note that strongly isolated implies weakly isolated and every terminal in $V(H)$ is strongly isolated in H. Also, if a terminal is strongly isolated in H, then it is strongly isolated in every supergraph of H.

Our goal is to obtain an instance where every tip is strongly isolated. The following theorem invokes previous work of Borradaile et al. [8] on building spanners and allows us to mark a bounded number of strongly isolated terminals as "relevant" in the sense that there is an almost-optimal solution that reaches only these terminals. Then we can ignore the rest of the terminals and extend the prespanner in a way that all these terminals are reached. This way, we achieve our main goal: every terminal is on the prespanner.

Theorem 25. *Consider a planar embedded graph H whose outer face consists of two paths P_L and P_g (a subpath of a group face g) where the latter path is formed of infinite edges only. (The two paths have the same endpoints.) Let Z denote the set of vertices on P_g, and assume each vertex of Z may have at most one edge not on P_g. Then, for any given $\varepsilon > 0$, in polynomial time, we can find a set $Z' \subseteq Z$ of vertices such that the following hold.*

1. *$|Z'| = f(\varepsilon^{-1})$.*
2. *For any subtree T of H connecting Z to P_L, there exists a subtree T' of H such that*
   - *(a) $T' \cap Z' \neq \emptyset$,*
   - *(b) $T' \cap P_L \supseteq T \cap P_L$, and*
   - *(c) $\mathrm{len}(T') \leq \mathrm{len}(T) + \varepsilon\,\mathrm{len}(P_L)$.*

Applying Theorem 25 on each face of the graph, it is not difficult to show that the prespanner can be extended in a way that every relevant strongly isolated terminal is reached:

Lemma 26. *Suppose that* P1 *holds. Let S be the set of terminals that are strongly isolated with respect to L. Then we can compute a supergraph L' of L with $\lambda(L') = f(1/\varepsilon)\lambda(L)$ such that the following holds: if there is a solution X, then there is a solution X^* with $\lambda(X^*) \leq \lambda(X) + O(\varepsilon\lambda(L))$ such that X^* has no vertex in $S \setminus V(L')$.*

Putting together, we get a spanner extension step that first makes every tip of every minimal group strongly isolated and then uses Lemma 26 to extend the spanner in a

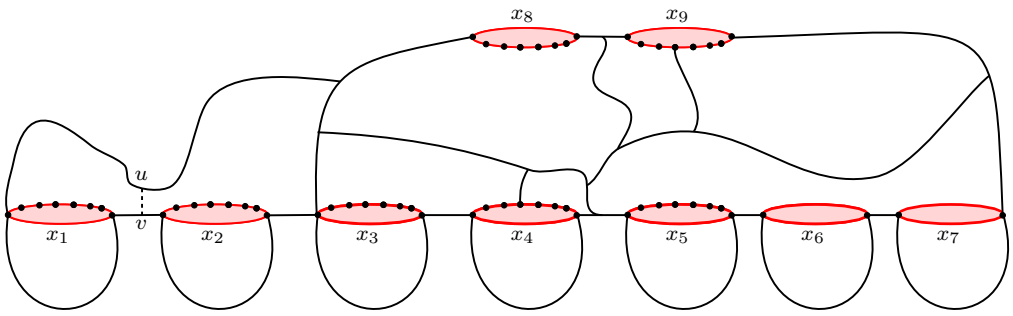

Figure 6: The black lines show the prespanner L. The tips of groups x_1, x_2, x_5 are all weakly isolated (note that groups x_6 and x_7 have no tips, and the edge uv, which is not in L, makes it impossible to connect a tip of x_1 and tip x_2 with a path internally disjoint from L. The tips of groups x_1, x_2, x_5 are not strongly isolated, but all the tips of groups x_8 and x_9, and three tips of x_4 are strongly isolated.

way that reaches all the relevant tips. Then the remaining tips of the minimal groups can be removed, and we achieve that the minimal groups have no tips.

Lemma 27. *There is a* $(\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P7},-) \to (\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P7}+\mathsf{P8},-)$ *spanner extension step, where*

[P2]: *Every minimal group has only one anchor.*
[P3]: *Every nonminimal group has exactly two anchors.*
[P7]: *Every tip is weakly isolated in L.*
[P8]: *The minimal groups have no tips.*

3.2.9 Goal 5: Every Tip is Strongly Isolated

Achieving property P7 is an important milestone, but it does not make the problem completely trivial yet. Even if an L-fragment contains a tip t of only one group, it can be attached to L at several points, and the choice of the tip t can affect what the most efficient way of connecting these points is. Our next goal is to extand the prespanner in a way that every tip of every nonminimal group becomes strongly isolated.

Lemma 28. *There is a* $(\mathsf{P0}+\mathsf{P1}+\mathsf{P3}+\mathsf{P7}+\mathsf{P8},-) \to (\mathsf{P0}+\mathsf{P9},-)$ *spanner extension step, where*

[P3]: *Every nonminimal group has exactly two anchors.*
[P7]: *Every tip is weakly isolated in L.*
[P8]: *The minimal groups have no tips.*
[P9]: *Every tip is strongly isolated in L.*

3.2.10 Goal 6: Handling Strongly Isolated Terminals

The next step of the algorithm finally reaches our main goal: the groups have no tips, that is, every terminal is on the prespanner. Formally, we prove the existence of the following spanner extension step, which is just a straightforward application of Lemma 26. Note that in this step, we do not assume P1, i.e., L is not necessarily a path.

Lemma 29. *There is a* $(\mathsf{P0}+\mathsf{P9},-) \to (\mathsf{P0}+\mathsf{P10},-)$ *spanner extension step, where*

[P9]: *Every tip is strongly isolated in L.*
[P10]: *Every terminal is on L.*

References

[1] E. M. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics*, 55(3):197–218, 1994.

[2] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[3] M. Bateni. *A Primal-Dual Clustering Technique with Applications in Network Design*. PhD thesis, Princeton University, Sept. 2011.

[4] M. Bateni, C. Chekuri, A. Ene, M. T. Hajiaghayi, N. Korula, and D. Marx. Prize-collecting Steiner problems on planar graphs. In *Symposium on Discrete Algorithms*, pages 1028–1049. SIAM, 2011.

[5] M. Bateni, M. T. Hajiaghayi, and D. Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21, 2011.

[6] P. Berman and V. Ramaiyer. Improved approximations for the Steiner tree problem. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 325–334, 1992.

[7] 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)*, pages 115–124, 2008.

[8] G. Borradaile, P. N. Klein, and C. Mathieu. An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5(3), 2009.

[9] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6, 2013.

[10] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 150–159. IEEE, 2011.

[11] E. D. Demaine and M. Hajiaghayi. Bidimensionality: New connections between fpt algorithms and ptass. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 590–601, 2005.

[12] E. D. Demaine, M. Hajiaghayi, and K.-i. Kawarabayashi. Contraction decomposition in H-minor-free graphs and algorithmic applications. In *Symposium on Theory of Computing*, pages 441–450. ACM, 2011.

[13] 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)*, pages 278–287, 2007.

[14] E. D. Demaine, M. T. Hajiaghayi, and K.-i. Kawarabayashi. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *Symposium on Foundations of Computer Science*, pages 637–646. IEEE Computer Society, 2005.

[15] E. D. Demaine, M. T. Hajiaghayi, and P. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 328–340. Springer, 2009.

[16] D. Eisenstat, P. Klein, and C. Mathieu. An efficient polynomial-time approximation scheme for Steiner forest in planar graphs. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 626–638, 2012.

[17] F. V. Fomin, D. Lokshtanov, and S. Saurabh. Efficient computation of representative sets with applications in parameterized and exact algorithms. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 142–151. SIAM, 2014.

[18] M. R. Garey and D. S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM J. Appl. Math.*, 32(4):826–834, 1977.

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

[20] J. Gudmundsson and C. Levcopoulos. A fast approximation algorithm for TSP with neighborhoods. *Nord. J. Comput.*, 6(4):469–, 1999.

[21] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *The 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 585–594, 2003.

[22] S. Hougardy and H. J. Prömel. A 1.598 approximation algorithm for the Steiner problem in graphs. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 448–453, 1999.

[23] R. M. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[24] M. Karpinski and A. Zelikovsky. New approximation algorithms for the Steiner tree problems. *Journal of Combinatorial Optimization*, 1(1):47–65, 1997.

[25] P. N. Klein. A subset spanner for planar graphs, with application to subset TSP. In *Symposium on Theory of Computing*, pages 749–756. ACM, 2006.

[26] P. N. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM Journal on Computing*, 37(6):1926–1952, 2008.

[27] C. S. Mata and J. S. B. Mitchell. Approximation algorithms for geometric tour and network design problems (extended abstract). In *Symposium on Computational Geometry*, pages 360–369, 1995.

[28] J. S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 11–18. SIAM, 2007.

[29] J. S. B. Mitchell. A constant-factor approximation algorithm for TSP with pairwise-disjoint connected neighborhoods in the plane. In *Symposium on Computational Geometry*, pages 183–191. ACM, 2010.

[30] H. J. Prömel and A. Steger. RNC-approximation algorithms for the Steiner problem. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 559–570, 1997.

[31] G. Reich and P. Widmayer. Beyond Steiner's problem: A VLSI oriented generalization. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, volume 411 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 1989.

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

[33] S. Safra and O. Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. *Computational Complexity*, 14(4):281–307, 2006.

[34] A. Zelikovsky. An 11/6-approximation for the Steiner problem on graphs. In *Proceedings of the Fourth Czechoslovakian Symposium on Combinatorics, Graphs, and Complexity (1990) in Annals of Discrete Mathematics*, volume 51, pages 351–354, 1992.

[35] A. Zelikovsky. An 11/6-approximation algorithm for the network Steiner problem. *Algorithmica*, 9(5):463–470, 1993.

[36] A. Zelikovsky. Better approximation bounds for the network and Euclidean Steiner tree problems. Technical report, University of Virginia, 1996.