Subexponential-Time Parameterized Algorithm for Steiner Tree on Planar Graphs*

Marcin Pilipczuk[1], Michał Pilipczuk[2], Piotr Sankowski[3], and Erik Jan van Leeuwen[4]

1,3 Institute of Informatics, University of Warsaw, Poland
{malcin,sank}@mimuw.edu.pl
2 Department of Informatics, University of Bergen, Norway
michal.pilipczuk@ii.uib.no
4 Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Italy
E.J.van.Leeuwen@dis.uniroma1.it

Abstract

The well-known bidimensionality theory provides a method for designing fast, subexponential-time parameterized algorithms for a vast number of NP-hard problems on sparse graph classes such as planar graphs, bounded genus graphs, or, more generally, graphs with a fixed excluded minor. However, in order to apply the bidimensionality framework the considered problem needs to fulfill a special density property. Some well-known problems do not have this property, unfortunately, with probably the most prominent and important example being the STEINER TREE problem. Hence the question whether a subexponential-time parameterized algorithm for STEINER TREE on planar graphs exists has remained open. In this paper, we answer this question positively and develop an algorithm running in $\mathcal{O}(2^{\mathcal{O}((k\log k)^{2/3})}n)$ time and polynomial space, where k is the size of the Steiner tree and n is the number of vertices of the graph. Our algorithm does not rely on tools from bidimensionality theory or graph minors theory, apart from Baker's classical approach. Instead, we introduce new tools and concepts to the study of the parameterized complexity of problems on sparse graphs.

1998 ACM Subject Classification G.2.2 Graphs Algorithms

Keywords and phrases planar graph, Steiner tree, subexponential-time algorithms

Digital Object Identifier 10.4230/LIPIcs.STACS.2013.353

1 Introduction

It is widely believed that no NP-hard problem can be solved in polynomial time. Interestingly, it turns out that NP-hard problems differ greatly in their exact exponential-time complexity. A majority of NP-complete problems admits algorithms that run in single-exponential time with respect to the natural parameters of the problem (see Section 2 for formal definitions of parameterized complexity). In many cases it is known that going below exponential time would violate the so-called *Exponential Time Hypothesis* [17]. However, a limited set of problems admits subexponential-time algorithms, which are significantly more efficient than the 'standard' exponential-time algorithms.

* The second author is supported by the ERC grant "Rigorous Theory of Preprocessing" no. 267959, whereas the remaining authors are partially supported by the ERC grant PAAl no. 259515. Additionally, the first and third authors acknowledge the support of the Foundation for Polish Science.

© Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, Erik Jan van Leeuwen;
licensed under Creative Commons License BY-ND
30th Symposium on Theoretical Aspects of Computer Science (STACS'13).
Editors: Natacha Portier and Thomas Wilke; pp. 353–364
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
SYMPOSIUM ON THEORETICAL ASPECTS OF COMPUTER SCIENCE

The main source of subexponential-time algorithms in parameterized complexity is the theory of bidimensionality. This framework applies to sparse graph classes, such as planar graphs and graphs with a fixed excluded minor. Without going into technical details, the framework provides subexponential-time parameterized algorithms [8] and linear kernels [14] for problems that fulfill a special density property. The problems that fall into this category include Dominating Set, Vertex Cover, and Feedback Vertex Set.

However, for some well-known problems the bidimensionality framework fails, with probably the most prominent and important example being the Steiner Tree problem. In this problem we are given a n-vertex graph G, an integer k, and a set of terminals $S \subseteq V(G)$. We are to find a tree T in G with at most k edges such that $S \subseteq V(T)$. The computational complexity of the Steiner Tree problem on general graphs is by now fully understood. It has a parameterized algorithm working in $\mathcal{O}(2^{|S|}n^{\mathcal{O}(1)})$ time, which is tight under Strong Exponential Time Hypothesis [21, 7], and it is unlikely to admit a polynomial kernel [10] (i.e., a polynomial-time preprocessing algorithm reducing the instance size to $|S|^{O(1)}$). It has a constant-factor approximation algorithm [6], but is APX-hard [3]. On the other hand, on planar graphs, the problem is known to be NP-hard [18, 15] and to possess an EPTAS [5]. It has remained unclear, however, whether topological assumptions on the graph (such as planarity) could speed up parameterized algorithms or provide a polynomial kernel.

In this work, we aim to bridge this gap by providing the first known subexponential-time parameterized algorithm for Steiner Tree on planar graphs. The algorithm is parameterized by the size k of the tree. We call this problem Planar Steiner Tree. Tazari [22] showed that this problem admits an $\mathcal{O}(2^{\mathcal{O}(\sqrt{k \log n})} n^{\mathcal{O}(1)})$-time algorithm by applying Baker's classical approach [1]. He explicitly posed the question whether a subexponential-time parameterized algorithm for Planar Steiner Tree exists. We answer this question positively and develop an algorithm running in $\mathcal{O}(2^{\mathcal{O}((k \log k)^{2/3})} n)$ time and polynomial space.

Our approach starts with the observation that to obtain a subexponential-time parameterized algorithm for Planar Steiner Tree it suffices to give a subexponential kernel and apply Tazari's algorithm to it. That is, we only need to develop an algorithm that reduces the size of the graph to be subexponential in k. To achieve this goal, we take a path that differs significantly from previous approaches, which were based on bidimensionality theory and which relied on some sort of grid minor theorem. Instead, we bring the strip and brick decomposition (developed by Klein and Borradaile et al. [19, 5] to obtain the EPTAS for Steiner Tree on planar graphs) to the parameterized setting. Roughly speaking, we partition the graph into a polynomial (in k) number of *bricks*, each of polynomial perimeter, such that within each brick the interaction between the solution and the perimeter of the brick is bounded *sublinearly* in k. Then the number of partial solutions within a brick is subexponential. We can restrict the graph to contain only the partial solutions, and thus obtain a graph of subexponential size. However, we were not able to obtain a true subexponential kernel for the problem. Our decomposition algorithm instead relies on branching, and in fact we obtain a subexponential number of subexponential kernels for the input instance.

We believe that our approach, which brings significantly new techniques to the parameterized complexity community, is of independent interest and will inspire further research on subexponential-time algorithms on sparse graph classes.

2 Preliminaries

For the standard graph notation used throughout this paper, we refer to [9]. We additionally need the following notation. If C is a simple cycle in a planar graph with fixed embedding, then $C[x,y]$ denotes the unique counter-clockwise path on C between x and y. When we refer to an order of vertices on a cycle, we mean their counter-clockwise order along the cycle. For a path or a cycle X, $|X|$ denotes the length of X in terms of the number of its edges.

2.1 Parameterized complexity

Parameterized complexity aims at explaining time and space complexities of various, usually NP-hard problems with the help of multivariate analysis. Instead of looking at the instance only from the side of the classical input size measure, we seek relevant parameters that are responsible for the exponential blow-up in the complexity of an algorithm. These parameters generally correspond to natural aspects of the input instance, such as the solution size or some structural parameter of the input graph. In other words, we try to distinguish easy and hard instances of an NP-hard problem by introducing an appropriate measure.

Formally, an instance comes with an integer parameter k. A *parameterized problem* Q then is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. We say that the problem is *fixed-parameter tractable* (*FPT*) if there exists an algorithm solving any instance (x,k) in time $f(k)\,\mathrm{poly}(|x|)$, for some computable (usually exponential) function f. It is known that a problem is FPT if and only if it is *kernelizable*: a *kernelization algorithm* for a problem Q takes an instance (x,k) and in time polynomial in $|x|+k$ produces an equivalent instance (x',k') (i.e., $(x,k) \in Q$ if and only if $(x',k') \in Q$) such that $|x'|+k' \leq g(k)$ for some computable function g. The function g is the *size of the kernel*, and if it is polynomial (linear), then we say that Q admits a *polynomial (linear) kernel*. For more detailed expositions, see e.g. [11, 13].

3 Bricks and strip decompositions

In this section, we recall the decomposition framework of Klein and Borradaile et al. [19, 5] and modify it to suit our purposes. The original framework operates on weighted graphs and uses approximate distances, as it was designed to be the main tool in an approximation algorithm (the EPTAS for Planar Steiner Tree). However, for our parameterized algorithm it is crucial to use exact distances between vertices. At the same time, we may use the fact that our graphs are unweighted. With this in mind, we develop a modified framework.

Let (G,S,k) be a Planar Steiner Tree instance. We start by manipulating the graph in such a way that all terminals lie on the outer face, in a similar manner as in the work of Borradaile et al. [5]. Intuitively, we find an approximate Steiner tree along which we cut the graph open and then make the resulting face the outer face (see Figure 1). Formally, we first compute an approximate Steiner tree T_{apx} connecting S. We could use an $O(n\log n)$-time 2-approximation algorithm [20, 23, 24] for this. However, since we aim for a linear dependency on n in the running time, we take a different approach. We compute a breadth-first search tree from a fixed terminal, and iteratively remove any nonterminal leafs from this tree. This takes linear time. The resulting tree is a (minimal) Steiner tree connecting S, which we use as T_{apx}. Note that the distance in T_{apx} between any two terminals must be at most k, or we may return NO. Hence we verify that T_{apx} has at most k^2 edges; otherwise, we return NO.

Given the tree T_{apx}, we cut the plane open along the tree, duplicating every edge of T_{apx}. Let $\hat{G}$ be the resulting graph. From here on, we fix a plane embedding of $\hat{G}$ that has the interior of the tree T_{apx} as the outer face (see Figure 1). Observe that now the terminals S lie only on the outer face. Moreover, each vertex or edge of $\hat{G}$ can be mapped to a vertex or edge in G; we denote this map by $\pi : V(\hat{G}) \cup E(\hat{G}) \to V(G) \cup E(G)$. Note that only vertices and edges of T_{apx} possibly appear more than once in the domain of this map.

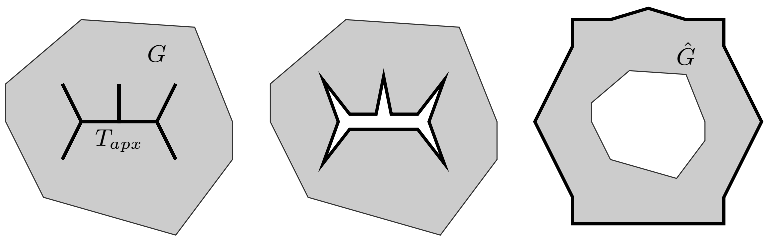

Figure 1 Construction of the cut-open graph $\hat{G}$.

We can now proceed with the basic notion of a *brick*. We call a vertex or edge of a plane graph *enclosed by a closed walk* of that graph if it is contained in a bounded Jordan region defined by the Jordan curve which the walk induces in the plane embedding.

▶ **Definition 1.** A *brick* H is the subgraph of $\hat{G}$ enclosed by a simple cycle of $\hat{G}$. This cycle is called the *perimeter* of the brick and denoted by ∂H.

We will also use the term perimeter to refer to the length of the perimeter of a brick H. By int$H$ we denote the set of edges enclosed by ∂H, i.e., int$H = E(H) \setminus E(\partial H)$. Observe that although $\pi(\partial H)$ may not always be a simple cycle in G, it is still a closed walk without self-crossings. Moreover, int$H$ is isomorphic to the part of G on one of the sides of $\pi(\partial H)$. Finally, observe that $\hat{G}$ itself is a brick with the Eulerian tour of T_{apx} as its perimeter.

The algorithm of this paper continuously decomposes bricks into smaller bricks. Here, a *decomposition* of a brick H is a collection of bricks inside H such that every face inside H belongs to exactly one of these bricks. Any brick produced by the algorithm will be immediately decomposed further into a particular type of bricks, called *strips*.

▶ **Definition 2.** A brick H is called a *strip* if ∂H can be partitioned into two paths R and B (called the *red* and *blue* paths below), such that

- B is the shortest path (in H) between its endpoints,
- every proper subpath of R is the shortest path (in H) between its endpoints.

The second condition is equivalent to saying that if $R = r_1, \ldots, r_{|R|}$, then $r_2, \ldots, r_{|R|}$ is a shortest path between r_2 and $r_{|R|}$ and $r_1, \ldots, r_{|R|-1}$ is a shortest path between r_1 and $r_{|R|-1}$. Observe that R and B share no edges, but do share two vertices of ∂H. All algorithms in the paper will construct and maintain R and B for each strip.

Klein [19] showed that, given a n-vertex brick H, one can decompose it into strips in time polynomial in n. We need a slightly different result for our unweighted, parameterized case.

▶ **Lemma 3.** *There exists an $\mathcal{O}(|\partial H|^2\, n)$-time algorithm that, given a brick H, decomposes it into at most $|\partial H|$ strips of perimeter at most $|\partial H|$ each.*

Proof. To prove the lemma it is sufficient to show an algorithm that in $\mathcal{O}(|\partial H|\, n)$ time decomposes the brick H into a strip H_0 and a number of bricks $H_1, H_2, \ldots, H_r$, such that the sum of the perimeters of all bricks $H_1, H_2, \ldots, H_r$ is strictly smaller than $|\partial H|$. We may then recursively decompose the bricks $H_1, H_2, \ldots, H_r$. The decrease in the total length of the perimeters yields the bound on the total number of strips in the obtained decomposition via a trivial induction, and also proves the claimed running time bound.

We say that an ordered pair of vertices $(x, y) \in V(\partial H) \times V(\partial H)$ is *cuttable* if $\partial H[x, y]$ is not a shortest path between x and y in H. A cuttable pair (x, y) is *minimal* if there does not

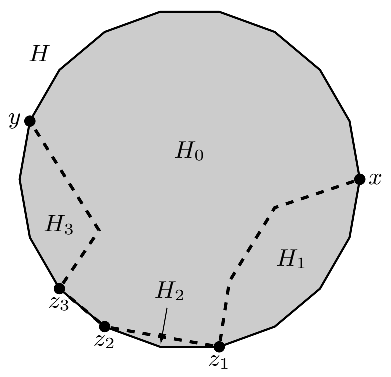

Figure 2 Step of the decomposition algorithm of Lemma 3. A brick H is decomposed into a strip H_0 and bricks H_1, H_2 and H_3. The dashed path is the path P.

exist another cuttable pair (x', y') with $\partial H[x', y']$ being a proper subpath of $\partial H[x, y]$. Note that a cuttable pair always exists in a brick H: since ∂H is a simple cycle, $|\partial H| \geq 3$ and a pair (x, y) is cuttable whenever $\partial H[y, x]$ consists of a single edge. Therefore a minimal cuttable pair always exists. Moreover, we can find such a pair in $\mathcal{O}(|\partial H|\, n)$ time using a breadth-first search for each vertex of ∂H.

Let (x, y) be a minimal cuttable pair in H and let P be a shortest path from x to y in H. We claim that P does not contain any vertex from the interior of the path $\partial H[x, y]$. Indeed, if z would be such a vertex, then a subpath of P from x to z or a subpath of P from z to y would witness that (x, z) or (z, y) is a cuttable pair, contradicting the minimality of (x, y).

We infer that $\partial H[x, y] \cup P$ is a simple cycle in H, and let H_0 be the part of H enclosed by $\partial H[x, y] \cup P$ (see Figure 2). Since P is a shortest path between x and y, $|P| \leq |\partial H[y, x]|$ and we infer that $|\partial H_0| \leq |\partial H|$. Moreover, the choice of P and the pair (x, y) implies that H_0 is a strip, with the blue path being the path P and the red path being $\partial H[x, y]$.

Let $x = z_0, z_1, z_2, \ldots, z_q = y$ be vertices of $V(P) \cap V(\partial H)$ in the order in which they appear on the path P (see Figure 2). We claim that they appear in the reverse order on the path $\partial H[y, x]$, that is, on the path $\partial H[y, x]$ the vertex z_i is closer than z_j is to x whenever $i < j$. Assume otherwise, and let i be the smallest index such that z_{i+1} is closer to x than z_i on the path $\partial H[y, x]$. Clearly $i \geq 1$. As P is a shortest path between x and y, P is a simple path, the vertices z_j are distinct, and the subpath of P from x to z_i separates z_{i+1} from y in the graph H. Therefore the subpath of P from x to z_i intersects the subpath from z_{i+1} to y, a contradiction to the fact that P is a simple path.

For $1 \leq i < q$, consider a closed walk P_i in H that consists of a subpath of P from z_{i-1} to z_i and of $\partial H[z_i, z_{i-1}]$. Note that P_i is a simple cycle unless it is of length 2. Let $H_1, H_2, \ldots, H_r$ be the set of all bricks enclosed by the paths P_i that are simple cycles (see Figure 2). From the previous claim we infer that $H_0, H_1, H_2, \ldots, H_r$ is a decomposition of H into $r + 1$ bricks. Moreover, $\sum_{i=1}^{r} |\partial H_i| \leq |P| + |\partial H[y, x]| < |\partial H[x, y]| + |\partial H[y, x]| = |\partial H|$. This finishes the proof of the lemma. ◀

If the algorithm of Lemma 3 is applied to a brick H, then a strip of the decomposition containing a vertex or edge of ∂H has that vertex or edge on its perimeter. In particular, if we apply the algorithm to the brick $\hat{G}$, then no terminal will lie in the interior of a strip.

4 A subexponential-time parameterized algorithm

We now show how to use strip decompositions to give a subexponential-time parameterized algorithm for Planar Steiner Tree.

4.1 Light strip decompositions

In this section, we formally define what we mean by the interaction of a Steiner tree with a decomposition of $\hat{G}$ into strips. Moreover, we show that if we know that this interaction is bounded sublinearly in k, then we can indeed find a Steiner tree of size at most k in subexponential time. Throughout, let (G, S, k) be an instance of Planar Steiner Tree and assume that it is a YES-instance. We will also assume that we have constructed the cut-open graph $\hat{G}$ as described before. Furthermore, we call a Steiner tree $T \subseteq E(G)$ *optimal* if the terminals in S are connected by T, $|T| \leq k$, and $|T|$ is minimum.

As a first step, we project trees in G onto $\hat{G}$. This can be done in a natural way using the mapping π. Somewhat abusing notation, given a tree T in G, we say that an edge e of $\hat{G}$ belongs to T if $\pi(e) \in T$. Note that this projection of T onto $\hat{G}$ may no longer be connected, and some edges of the tree may be duplicated if they coincide with the edges of T_{apx}. However, inside a brick the tree T should behave similarly in G and in $\hat{G}$, because the interiors of bricks are isomorphic in G and $\hat{G}$.

Using this projection, we can formally define the interaction between an optimal Steiner tree and a decomposition of $\hat{G}$ into strips.

▶ **Definition 4.** Let T be an optimal Steiner tree, let H be a strip in some strip decomposition of $\hat{G}$, and let $v \in V(\partial H)$. We say that v is a *portal for* T if v is incident to an edge e that belongs both to the interior of H and to T (formally, $e \in \text{int}H$ and $\pi(e) \in T$).

The number of portals of strips allows a distinction between *light* and *heavy* strips of a strip decomposition of $\hat{G}$. The threshold is based on a number $p = p(k) = k^{2/3}/\log^{1/3} k$. We choose to represent it symbolically in order to expose the nature of the trade-offs made in the design of the algorithm. We implicitly use that $p = o(k)$ and that $p, k/p = \omega(1)$.

▶ **Definition 5.** Let T be an optimal Steiner tree and let H be a strip in some strip decomposition of $\hat{G}$. A strip H is called *light with respect to* T if ∂H contains at most k/p portals for T. Otherwise, H is called *heavy with respect to* T.

We call a strip decomposition of $\hat{G}$ *light* if all its strips are light with respect to some optimal Steiner tree T. The next lemma shows that knowing a light strip decomposition of a particular type allows the instance to be solved.

▶ **Lemma 6.** *There exists an algorithm which takes as input an instance (G, S, k) of the* Planar Steiner Tree *problem together with the tree T_{apx}, the graph $\hat{G}$, and a strip decomposition $\mathcal{D}$ of $\hat{G}$ such that (i) $\mathcal{D}$ consists of at most $\mathcal{O}(k^2 p)$ strips of perimeter at most $\mathcal{O}(k^2)$ each, and (ii) no terminal is in the interior of a strip of $\mathcal{D}$, and which outputs either nothing or a solution of size at most k. Moreover, if (G, S, k) is a YES instance and $\mathcal{D}$ is light, then it returns a solution of size at most k. The algorithm runs in $\mathcal{O}(2^{\mathcal{O}(\sqrt{(k^2 \log k)/p})} n)$ time and polynomial space.*

Proof. The algorithm starts by marking the set of edges that it will be allowed to use. First, we mark all the perimeters of all the strips in the decomposition $\mathcal{D}$. As there are at most $\mathcal{O}(k^2 p)$ strips of perimeter at most $\mathcal{O}(k^2)$ each, this procedure marks at most $\mathcal{O}(k^4 p)$ edges in total. Second, we iteratively consider all strips $H \in \mathcal{D}$. For every strip H, consider each set $X \subseteq V(\partial H)$ of cardinality at most k/p. We then compute an optimal Steiner tree in H connecting X. This is an instance of the PLANAR STEINER TREE problem where all terminals lie on the outer face. Erickson et al. [12, p. 661] proved that such instances can be solved in $O(\ell^3 n + \ell^2 n \log n)$ time, where ℓ is the number of terminals. However, as noted by Borradaile et al. [5, p. 3], this running time can be trimmed to $\mathcal{O}(\ell^3 n)$. Hence in $\mathcal{O}(k^{\mathcal{O}(1)} |H|)$ time we can compute an optimal Steiner tree in H connecting X. If the cost of this tree does not exceed k, then we mark its edges. Note that the number of edges marked in this manner is at most $\mathcal{O}(k^2 p) \cdot k/p \cdot \binom{\mathcal{O}(k^2)}{k/p} \cdot k = \mathcal{O}(k^{\mathcal{O}(k/p)})$. Hence, the total number of edges marked is $\mathcal{O}(k^{\mathcal{O}(k/p)})$.

Observe that this marking of $\hat{G}$ immediately implies a marking of G using the mapping π. We now delete all the unmarked edges of G as well as any nonterminals that become isolated, and apply the algorithm of Tazari [22] to the remaining instance. The graph of this instance has $\mathcal{O}(k^{\mathcal{O}(k/p)})$ vertices. Hence Tazari's algorithm runs in $\mathcal{O}(2^{\mathcal{O}(\sqrt{k \log k^{\mathcal{O}(k/p)}})}) = \mathcal{O}(2^{\mathcal{O}(\sqrt{(k^2 \log k)/p})})$ time and polynomial space. If Tazari's algorithm finds a solution, then it is a solution of size at most k and we output it; otherwise, we output nothing. The marking procedure takes $\mathcal{O}(k^{\mathcal{O}(k/p)}\, n) \leq \mathcal{O}(2^{\mathcal{O}(\sqrt{(k^2 \log k)/p})}\, n)$ time and polynomial space as well.

It remains to prove that if (G, S, k) is a YES instance and every strip of $\mathcal{D}$ is light with respect to an optimal Steiner tree T, then there exists an optimal solution T' that uses only the marked edges. Consider a strip $H \in \mathcal{D}$ and let $C_1, C_2, \ldots, C_\ell$ be the components of the forest $T \cap \mathrm{int} H$. Each component C_i is a Steiner tree connecting incident portals. As the number of portals incident to C_i is bounded by k/p, $|C_i| \leq k$, and the interiors of the bricks are isomorphic in G and $\hat{G}$, we have marked some optimal Steiner tree C'_i connecting the same portals at non-greater cost. Replace each component C_i with C'_i, and perform such replacements in all the strips of $\mathcal{D}$, thus obtaining a tree T'. Clearly, T' is still a solution, it is not more expensive than T, and it only uses marked edges. ◀

It remains to find a light strip decomposition that is required by the lemma.

4.2 A branching algorithm to obtain a light strip decomposition

In this section, we develop a branching algorithm that outputs many strip decompositions, one of which is light. It was observed before that $\hat{G}$ is a brick, which can thus be decomposed into strips using Lemma 3. This gives an initial strip decomposition where no terminal is in the interior of a strip. However, we cannot guarantee that this decomposition is light. The idea is therefore to guess a strip which is heavy with respect to some optimal Steiner tree, and then to partition it into two simpler bricks, which are subsequently decomposed again into strips using Lemma 3. Since we have no way of knowing which strips might be heavy or what a good partition is, we apply branching and try all possibilities. Our analysis shows that after branching a certain amount of times, we find a light strip decomposition in one of the branches. We now present a formal description of this algorithmic intuition.

The main analytical tool to measure the progress of the algorithm is the following potential function. For a strip decomposition $\mathcal{D}$ of $\hat{G}$ and an optimal solution T, let

$$\Phi(\mathcal{D}, T) = \left(4 \cdot \sum_{H \in \mathcal{H}(\mathcal{D},T)} \frac{|\mathrm{int} H \cap \pi^{-1}(T)|}{k/p}\right) - |\mathcal{H}(\mathcal{D}, T)|.$$

Here $\mathcal{H}(\mathcal{D}, T)$ is the set of strips from $\mathcal{D}$ that are heavy with respect to T. Note that the potential is always nonnegative, as heavy strips each contain more than k/p portals for T

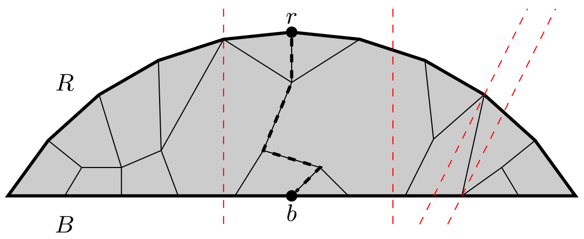

Figure 3 The split asserted by the statement of Lemma 7. The red dashed lines indicate the partition into portal components. The dashed path represents a path of length at most k between r and b, which must exist because r and b belong to the same portal component.

and thus their interiors each contain more than $k/(2p)$ edges of T. Moreover, $\Phi(\mathcal{D}, T) \leq 4p$ for any decomposition $\mathcal{D}$ and any solution T. Finally, if $\Phi(\mathcal{D}, T) = 0$ for some decomposition $\mathcal{D}$ and some solution T, then all the strips of $\mathcal{D}$ are light with respect to T. The potential function enables the definition of an *extremal* solution for a strip decomposition $\mathcal{D}$: let $T(\mathcal{D})$ denote an optimal solution that minimizes $\Phi(\mathcal{D}, T(\mathcal{D}))$.

We are now ready formalize the partition of a heavy strip (see Figure 3 for an illustration).

▶ **Lemma 7.** *Let (G, S, k) be an instance of* Planar Steiner Tree *and let H be a strip in some strip decomposition $\mathcal{D}$ of $\hat{G}$ not containing terminals in the interior, with R and B being the red and blue paths of ∂H, respectively. Assume that (G, S, k) is a YES-instance and assume furthermore that H is heavy with respect to $T(\mathcal{D})$. Let $\ell \geq k/p$ be the number of portals in H for solution $T(\mathcal{D})$. Then there exist vertices $r \in V(R)$ and $b \in V(B)$ such that*

(i) there exists a path in H between r and b that avoids ∂H apart from the endpoints and has length at most k,

(ii) for every such path P, the bricks with perimeters $\partial H[r, b] \cup P$ and $\partial H[b, r] \cup P$ both contain at least $\ell/2 - 2$ portals for $T(\mathcal{D})$.

Proof. Let F be the forest induced in $T(\mathcal{D})$ by the internal edges of the strip, i.e., $F = T(\mathcal{D}) \cap \text{int}H$. We say that two edges $e_1, e_2 \in F$ are in the same *portal component* of F if some endpoint of e_1 can be connected to some endpoint of e_2 by a path in F traversing only vertices not from $V(\partial H)$. Observe that this path can have length 0, but then its only vertex cannot belong to $V(\partial H)$. We observe that this relation is an equivalence relation. Note that the partition of the edges of F into portal components can be more refined than a partition into connected components, as edges from F incident to the same portal are all in different portal components. We say that a portal v is incident to a portal component C if v is incident to an edge from C.

Claim 1. Every portal component C is incident to a portal belonging to $V(R)$ and to a portal belonging to $V(B)$.

Assume first that C is incident only to portals from $V(R)$. Let r_1 and r_2 be the first and the last portal on R that are incident to C, in counter-clockwise direction on ∂H. As C is not incident to portals from $V(B)$ and the endpoints of R belong to $V(B)$, we have that r_1 and r_2 lie in the interior of R. From the definition of a strip, we know that $\partial H[r_1, r_2]$ is a shortest path between r_1 and r_2. We infer that in T we can substitute the portal component C with the path $\partial H[r_1, r_2]$, thus creating a solution with non-greater cost (as C contains some path from r_1 to r_2) and with strictly smaller potential. This contradicts the properties of $T(\mathcal{D})$. The argument that C must be incident to a portal from $V(B)$ is analogous. This settles Claim 1.

Using Claim 1, we prove the claimed existence of the vertices r and b. Let $r_1, r_2, \ldots, r_t$ be the portals on R, in clockwise direction on ∂H, and let $b_1, b_2, \ldots, b_s$ be the portals on B, in counter-clockwise direction on ∂H. Note that possibly $r_1 = b_1$ or $r_t = b_s$. For indices $i \in \{1, 2, \ldots, t\}$ and $j \in \{1, 2, \ldots, s\}$, let $f(i,j)$ be the number of portals in the interior of the path $\partial H[r_i, b_j]$, i.e., $f(i,j) = |\{r_1, \ldots, r_{i-1}\} \cup \{b_1, \ldots, b_{j-1}\}|$. Let (i_0, j_0) be such a pair for which (a) r_{i_0} and b_{j_0} are incident to the same portal component C, and (b) $f(i_0, j_0) \leq \ell/2 - 1$ but is maximum. By Claim 1, we infer that r_1 and b_1 are always incident to the same portal component and that $f(1,1) = 0$. Hence, such a pair (i_0, j_0) is well defined. Let $r = r_{i_0}$ and $b = b_{j_0}$. As r and b are incident to the same portal component, property (i) is satisfied. To prove property (ii), we need the following claim:

Claim 2. $f(i_0, j_0) \geq \ell/2 - 2$.

Assume otherwise, i.e. $f(i_0, j_0) \leq \ell/2 - 3$. If $i_0 = t$ and $j_0 = s$, then $f(i_0, j_0) \geq \ell - 2 > \ell/2 - 3$, a contradiction. If $i_0 < t$ and r_{i_0+1} is incident to the same portal component as r and b, then we have that $f(i_0 + 1, j_0) \leq \ell/2 - 1$, which is a contradiction with the choice of (i_0, j_0). An analogous contradiction occurs if $j_0 < s$ and b_{j_0+1} is incident to the same portal component as r and b. Now note that if $i_0 = t$ and $j_0 < s$, then by Claim 1 portal b_{j_0+1} is incident to the same portal component as r and b, which, as observed, gives a contradiction. Similarly we exclude the case when $j_0 = s$ and $i_0 < t$. Therefore $i_0 < t$, $j_0 < s$, and both r_{i_0+1} and b_{j_0+1} do not belong to the same portal component as r and b. By Claim 1, r_{i_0+1} and b_{j_0+1} are incident to the same portal component. As $f(i_0+1, j_0+1) \leq f(i_0, j_0) + 2 \leq \ell/2 - 1$, this contradicts the choice of (i_0, j_0) and settles Claim 2.

We now prove that property (ii) is satisfied. Let P be any path in H between r and b that avoids ∂H apart from the endpoints. Consider the brick with perimeter $\partial H[r,b] \cup P$ and observe that all the portals in the interior of $\partial H[r,b]$ are still portals in this brick. As $f(i_0, j_0) \geq \ell/2 - 2$, this brick has at least $\ell/2 - 2$ portals. Similarly, all the portals in the interior of $\partial H[b,r]$ are still portals in the brick with perimeter $\partial H[b,r] \cup P$. Hence, this brick also has at least $\ell - f(i_0, j_0) - 2 \geq \ell - (\ell/2 - 1) - 2 \geq \ell/2 - 2$ portals. ◀

Note that $\ell/2 - 2 = \omega(1)$, so we can assume that vertices r, b given by Lemma 7 are distinct from each other. Hence, the obtained decomposition is non-degenerate.

Using Lemma 7, we can give the branching strategy.

▶ **Lemma 8.** *There exists an algorithm that, given an instance (G, S, k) of* Planar Steiner Tree*, the tree T_{apx}, and the graph $\hat{G}$, in time $\mathcal{O}(k^{\mathcal{O}(p)} n)$ outputs a sequence $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_t$ of strip decompositions of $\hat{G}$ such that the following properties hold:*

(i) $t = k^{\mathcal{O}(p)}$*;*
(ii) each decomposition consists of at most $\mathcal{O}(k^2 p)$ strips of perimeter at most $\mathcal{O}(k^2)$ each, and no terminal lies in the interior of any strip;
(iii) if (G, S, k) is a YES-instance, then there is a decomposition $\mathcal{D}_i$ such that all the strips of $\mathcal{D}_i$ are light with respect to $T(\mathcal{D}_i)$.

The working space of the algorithm, excluding the output decompositions, is polynomial.

Proof. We follow a recursive branching procedure. At depth d of the recursive procedure we maintain a strip decomposition $\mathcal{D}$ with following properties: $\mathcal{D}$ consists of at most $\mathcal{O}(dk(d+k))$ strips of perimeter at most $2k^2 + dk$ each, and no terminal lies in the interior of any strip. Moreover, at depth d we branch into at most $\mathcal{O}(dk^3(d+k)^3)$ branches. Finally, we show that it suffices to run the branching to a depth of at most $8p$. Hence, as $p = o(k)$, the total number of produced decompositions is bounded by $t = k^{\mathcal{O}(p)}$.

First, we need to provide a starting decomposition at the 0-th level of the recursion. We start with a single brick $\hat{G}$ — that is, the entire graph $\hat{G}$ with its outer face (the Euler tour of T_{apx}) as the perimeter. Recall that the size of T_{apx} is bounded by k^2, and thus the perimeter of $\hat{G}$ is at most $2k^2$. We apply Lemma 3 to this brick in order to obtain some strip decomposition $\mathcal{D}_0$ of $\hat{G}$. This decomposition consists of at most $2k^2$ strips, each having perimeter at most $2k^2$. As all the terminals lie on the perimeter of $\hat{G}$, no terminal lies inside any strip of $\mathcal{D}_0$. This decomposition is the initial one for the branching procedure.

We now describe the operations performed on level d of the recursion. We do the following:

- output the current decomposition $\mathcal{D}$ as the next $\mathcal{D}_i$;
- branch on each H, r, b, where H is a strip of D with R, B being the red and the blue path of ∂H, respectively, and $r \in V(R), b \in V(B)$ are vertices that can be connected via a path P (chosen arbitrarily) of length at most k that avoids ∂H apart from the endpoints. For each such H, r, b, divide H into bricks with perimeters $\partial H[r,b] \cup P$ and $\partial H[b,r] \cup P$, and apply Lemma 3 to decompose both bricks into strips. Proceed with the in total $\mathcal{O}(dk(d+k) \cdot (k(d+k))^2)$ obtained decompositions to the next level of the recursion.

Observe that if $|\partial H| \leq 2k^2 + dk$, then both bricks created in the initial division have perimeter at most $2k^2 + (d+1)k$, as $|P| \leq k$. Applying Lemma 3 cannot create strips with longer perimeter, and increases the number of strips in the decomposition by at most $4k^2 + 2(d+1)k$. Hence, the $\mathcal{O}(dk(d+k))$ bound on the number of strips on level d of the recursion holds. As all the obtained decompositions only refine the initial one, in all the subcases no terminal will lie inside any strip.

We now analyse the running time of the algorithm. Note that finding the path P for a given strip H and vertices $r \in V(R), b \in V(B)$ boils down to finding a shortest path in an appropriate subgraph of the graph $\hat{G}$, which can be done in $O(n)$ time. As we output $\mathcal{O}(k^{\mathcal{O}(p)})$ decompositions, the bound on the running time follows.

It remains to prove that if we perform $8p$ levels of the branching procedure, then property (iii) will hold. We do this by examining the change of the potential $\Phi(\mathcal{D}, T(\mathcal{D}))$ during the branching procedure.

Claim 1. Let $\mathcal{D}$ be the decomposition at some step of the branching procedure, such that strip $H \in \mathcal{D}$, where R, B are the red and blue paths of ∂H, respectively, is heavy with respect to $T(\mathcal{D})$. Let $r \in V(R)$ and $b \in V(B)$ be the vertices whose existence is asserted by Lemma 7. Then in the branch of H, r, and b the potential decreases by at least $1/2$.

Let $\mathcal{D}'$ be the strip decomposition after performing the aforementioned branching. We prove a slightly stronger claim, namely that $\Phi(\mathcal{D}', T(\mathcal{D})) \leq \Phi(\mathcal{D}, T(\mathcal{D})) - 1/2$. As $\Phi(\mathcal{D}', T(\mathcal{D}')) \leq \Phi(\mathcal{D}', T(\mathcal{D}))$ by the definition of $T(\mathcal{D}')$, this implies Claim 1.

We consider three cases. First assume that all the strips created when constructing $\mathcal{D}'$ from $\mathcal{D}$ are in fact light with respect to $T(\mathcal{D})$. Then the first term in the potential function decreases by at least 2, as every heavy brick has more than $k/(2p)$ edges from $T(\mathcal{D})$ in the interior, while the second term increases by 1. Hence the potential decreases by at least 1.

Second, assume that exactly one created strip H_0 is heavy. Without loss of generality assume that this strip was obtained while decomposing the brick with perimeter $\partial H[r,b] \cup P$. By Lemma 7, the brick with perimeter $\partial H[b,r] \cup P$ has at least $k/(2p) - 2 \geq k/(4p)$ portals; here we use the bound $k \geq 8p$ that holds for large enough k due to $p = o(k)$. Hence, it has at least $k/(8p)$ edges from $T(\mathcal{D})$ in the interior. Therefore, the brick with perimeter $\partial H[r,b] \cup P$ has at least $k/(8p)$ edges from $T(\mathcal{D})$ fewer in the interior than H. The strip H_0 can only have fewer edges from $T(\mathcal{D})$ in the interior, so the first term of the potential function decreases by at least $1/2$. As the second term is unchanged, this settles this case.

Finally, assume that at least two created strips are heavy. Then the first term of the potential function cannot increase, as every edge of $T(\mathcal{D})$ in the interior of the created heavy strips was already in the interior of the heavy strip H, while the second term decreases by at least 1. Hence the potential decreases by at least 1 in this case. This settles Claim 1.

Since the potential initially is at most $4p$, and Claim 1 proves that the potential decreases by at least $1/2$ in one of the branches, it suffices to branch to at most $8p$ levels deep to ensure that we find a decomposition $\mathcal{D}_i$ for which $\Phi(\mathcal{D}_i, T(\mathcal{D}_i)) = 0$. As observed before, this implies that all strips of $\mathcal{D}_i$ are light with respect to $T(\mathcal{D}_i)$. This proves property (iii). ◀

We are ready to prove the main result of this paper.

▶ **Theorem 9.** *The* Planar Steiner Tree *problem can be solved in* $\mathcal{O}(2^{\mathcal{O}((k \log k)^{2/3})}\, n)$ *time and polynomial space.*

Proof. We start by constructing the approximate solution T_{apx} and the graph $\hat{G}$ in $\mathcal{O}(n)$ time. Then, we run the algorithm given by Lemma 8 and to each output decomposition we apply Lemma 6. The correctness of the algorithm follows from property (iii) in the statement of Lemma 8, while the claim on the running time follows from the fact that to at most $\mathcal{O}(k^{\mathcal{O}(p)}) = \mathcal{O}(2^{\mathcal{O}((k \log k)^{2/3})})$ instances we apply an algorithm running in $\mathcal{O}(2^{\mathcal{O}(\sqrt{(k^2 \log k)/p})}\, n) = \mathcal{O}(2^{\mathcal{O}((k \log k)^{2/3})}\, n)$ time. The polynomial space bound can be achieved by applying Lemma 6 to each output decomposition once it is fully constructed. ◀

5 Conclusions and open problems

Although our result positively answers the question whether a subexponential-time parameterized algorithm for Planar Steiner Tree exists, it raises many new open questions.

We first ask whether our algorithm can be improved. We can show that Planar Steiner Tree cannot have a $\mathcal{O}(2^{o(\sqrt{k})}\, n^{\mathcal{O}(1)})$-time algorithm unless the Exponential Time Hypothesis is false, using the standard NP-hardness reduction from Connected Vertex Cover. It seems reasonable to think that the right upper bound is $\mathcal{O}(2^{\mathcal{O}(\sqrt{k})}\, n^{\mathcal{O}(1)})$. Such an algorithm — up to a logarithmic factor in the exponent — follows immediately if Planar Steiner Tree would admit a polynomial kernel. We conjecture that such a kernel indeed exists.

Apart from the parameterization by the size of the tree investigated in this paper, there is a second natural parameterization of the Steiner Tree problem, namely by $|S|$, the number of terminals. Recall that on general graphs the $\mathcal{O}(2^{|S|} n^{\mathcal{O}(1)})$-time algorithm due to Nederlof [21] is probably optimal [7]. Our techniques seem to break down on this stronger parameterization. Does Planar Steiner Tree admit a subexponential-time algorithm with respect to the number of terminals in the instance?

Another interesting direction is to try to generalize our algorithm to the closely related Planar Steiner Forest problem. With a bit of simple preprocessing, we can obtain an analogue of the approximate tree T_{apx} and construct the cut-open graph $\hat{G}$. Using this graph, we may perform the same branching procedure as in Lemma 8, and obtain a subexponential number of subexponential kernels for this problem. However, the last step of the algorithm — Tazari's algorithm [22] based on Baker's approach [1] — breaks down, as Steiner Forest is NP-hard on graphs of treewidth 3 [2, 16]. Thus, one needs significantly new ideas to turn a subexponential kernel into a subexponential algorithm for this problem.

Last but not least, we mention that Borradaile et al. [4] did some work on lifting the brick and strip decomposition to graphs of bounded genus. It may therefore be interesting whether our algorithm can also be extended to such graphs.

Acknowledgments
The authors thank Fedor Fomin, M.S. Ramanujan, and Marek Cygan for helpful discussions.

References

1 B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994.

2 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.

3 M. W. Bern and P. E. Plassmann. The Steiner problem with edge lengths 1 and 2. *Inf. Process. Lett.*, 32(4):171–176, 1989.

4 G. Borradaile, E. D. Demaine, and S. Tazari. Polynomial-time approximation schemes for subset-connectivity problems in bounded-genus graphs. In *STACS*, pages 171–182, 2009.

5 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.

6 J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In L. J. Schulman, editor, *STOC*, pages 583–592. ACM, 2010.

7 M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *IEEE Conference on Computational Complexity*, pages 74–84. IEEE, 2012.

8 E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and D. M. Thilikos. Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM*, 52(6):866–893, 2005.

9 R. Diestel. *Graph Theory.* Springer, 2005.

10 M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and IDs. In *ICALP (1)*, pages 378–389, 2009.

11 R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

12 R. E. Erickson, C. L. Monma, and A. F. J. Veinott. Send-and-split method for minimum-concave-cost network flows. *Mathematics of Operations Research*, 12(4):pp. 634–664, 1987.

13 J. Flum and M. Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. Springer, 2006.

14 F. V. Fomin, D. Lokshtanov, S. Saurabh, and D. M. Thilikos. Bidimensionality and kernels. In M. Charikar, editor, *SODA*, pages 503–510. SIAM, 2010.

15 M. R. Garey and D. S. Johnson. The Rectilinear Steiner Tree problem is NP complete. *SIAM Journal of Applied Mathematics*, 32:826–834, 1977.

16 E. Gassner. The Steiner forest problem revisited. *J. Discrete Algorithms*, 8(2):154–163, 2010.

17 R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

18 R. Karp. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

19 P. N. Klein. A subset spanner for planar graphs, with application to Subset TSP. In J. M. Kleinberg, editor, *STOC*, pages 749–756. ACM, 2006.

20 K. Mehlhorn. A faster approximation algorithm for the Steiner problem in graphs. *Inf. Process. Lett.*, 27(3):125–128, 1988.

21 J. Nederlof. Fast polynomial-space algorithms using Möbius inversion: Improving on Steiner Tree and related problems. In *ICALP (1)*, pages 713–725, 2009.

22 S. Tazari. Faster approximation schemes and parameterized algorithms on H-minor-free and odd-minor-free graphs. In *MFCS*, pages 641–652, 2010.

23 P. Widmayer. On approximation algorithms for Steiner's problem in graphs. In G. Tinhofer and G. Schmidt, editors, *WG*, volume 246 of *LNCS*, pages 17–28. Springer, 1986.

24 Y.-F. Wu, P. Widmayer, and C. K. Wong. A faster approximation algorithm for the Steiner problem in graphs. *Acta Inf.*, 23(2):223–229, 1986.