On subexponential parameterized algorithms for Steiner Tree and Directed Subset TSP on planar graphs

Dániel Marx
Institute for Computer Science and Control
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary
Email: dmarx@cs.bme.hu

Marcin Pilipczuk and Michał Pilipczuk
Institute of Informatics
University of Warsaw
Warsaw, Poland
Emails: (marcin|michal).pilipczuk@mimuw.edu.pl

***Abstract*—There are numerous examples of the so-called "square root phenomenon" in the field of parameterized algorithms: many of the most fundamental graph problems, parameterized by some natural parameter k, become significantly simpler when restricted to planar graphs and in particular the best possible running time is exponential in $\mathcal{O}(\sqrt{k})$ instead of $\mathcal{O}(k)$ (modulo standard complexity assumptions). We consider two classic optimization problems parameterized by the number of terminals. The STEINER TREE problem asks for a minimum-weight tree connecting a given set of terminals T in an edge-weighted graph. In the SUBSET TRAVELING SALESMAN problem we are asked to visit all the terminals T by a minimum-weight closed walk. We investigate the parameterized complexity of these problems in planar graphs, where the number $k = |T|$ of terminals is regarded as the parameter. Our results are the following:**

- **SUBSET TSP can be solved in time $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ even on edge-weighted directed planar graphs. This improves upon the algorithm of Klein and Marx [SODA 2014] with the same running time that worked only on undirected planar graphs with polynomially large integer weights.**
- **Assuming the Exponential-Time Hypothesis, STEINER TREE on undirected planar graphs cannot be solved in time $2^{o(k)} \cdot n^{\mathcal{O}(1)}$, even in the unit-weight setting. This lower bound makes STEINER TREE the first "genuinely planar" problem (i.e., where the input is only planar graph with a set of distinguished terminals) for which we can show that the square root phenomenon does not appear.**
- **STEINER TREE can be solved in time $n^{\mathcal{O}(\sqrt{k})} \cdot W$ on undirected planar graphs with maximum edge weight W. Note that this result is incomparable to the fact that the problem is known to be solvable in time $2^k \cdot n^{\mathcal{O}(1)}$ even in general graphs.**

A direct corollary of the combination of our results for STEINER TREE is that this problem does not admit a parameter-preserving polynomial kernel on planar graphs unless ETH fails.

***Keywords*-Steiner tree; subexponential algorithms; parameterized algorithms; planar graphs;**

I. INTRODUCTION

It has been observed in the context of different algorithmic paradigms that planar graphs enjoy important structural properties that allow more efficient solutions to many of the classic hard algorithmic problems. The literature on approximation algorithms contains many examples of optimization problems that are APX-hard on general graphs, but admit polynomial-time approximation schemes (PTASes) when restricted to planar graphs (see, e.g., [1]–[10]). Moreover, even though the planar versions of most NP-hard problems remain NP-hard, a more fine-grained look reveals that significantly better running times are possible for planar graphs. As a typical example, consider the 3-COLORING problem: it can be solved in $2^{\mathcal{O}(n)}$ time in general graphs and, assuming the Exponential-Time Hypothesis (ETH), this is best possible as there is no $2^{o(n)}$-time algorithm. However, when restricted to planar graphs, 3-COLORING can be solved in time $2^{\mathcal{O}(\sqrt{n})}$, which is again best possible assuming ETH: the existence of a $2^{o(\sqrt{n})}$-time algorithm would contradict ETH. There are many other problems that behave in a similar way and this can be attributed to the combination of two important facts: (1) every planar graph on n vertices has treewidth $\mathcal{O}(\sqrt{n})$ and (2) given an n-vertex graph of treewidth t, most of the natural combinatorial problems can be solved in time $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(1)}$ (or perhaps $2^{\mathcal{O}(t \cdot \mathrm{polylog}\ t)} \cdot n^{\mathcal{O}(1)}$). On the lower bound side, to rule out $2^{o(\sqrt{n})}$-time algorithms, it is sufficient to observe that most planar NP-hardness proofs increase the size of the instance at most quadratically (because of the introduction of crossing gadgets). For example, there is a reduction that given an instance of 3SAT with n variables and m clauses produce an instance of 3-COLORING that is a planar graph with $\mathcal{O}((n+m)^2)$ vertices. Together with ETH, such a reduction rules out $2^{o(\sqrt{n})}$-time algorithms for planar 3-COLORING. Thus the existence of this "square root phenomenon" giving $2^{\mathcal{O}(\sqrt{n})}$ time complexity is well-understood both from the algorithmic and complexity viewpoints.

Our understanding of this phenomenon is much less complete for parameterized problems. A large fraction of natural fixed-parameter tractable graph problems can be solved in time $2^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ (with notable exceptions [11], [12]) and a large fraction of W[1]-hard problems can be solved in time $n^{\mathcal{O}(k)}$. There are tight or almost-tight lower bounds showing the optimality of these running times. By

now, there is a growing list of problems where the running time improves to $2^{\mathcal{O}(\sqrt{k}\cdot\text{polylog } k)}\cdot n^{\mathcal{O}(1)}$ or to $n^{\mathcal{O}(\sqrt{k}\cdot\text{polylog } k)}$ when restricted to planar graphs. For a handful of problems (e.g., INDEPENDENT SET, DOMINATING SET, FEEDBACK VERTEX SET, k-PATH) this improvement can be explained in a compact way by the elegant theory of bidimensionality [13]. However, there is no generic argument (similar to the simple argument described above for the existence of $2^{\mathcal{O}(\sqrt{n})}$ algorithms) why such an improvement should be possible for most parameterized problems. The fact that every n-vertex planar graph has treewidth $\mathcal{O}(\sqrt{n})$ does not seem to help in improving the $2^{\mathcal{O}(k)}$ factor to $2^{\mathcal{O}(\sqrt{k})}$ in the running time. The algorithmic results of this form are thus very problem-specific, exploiting nontrivial observations on the structure of the solution or invoking other tools tailored to the problem's nature. Recent results include algorithms for SUBSET TSP [14], MULTIWAY CUT [15], [16], unweighted STEINER TREE parameterized by the number of edges of the solution [17], [18], STRONGLY CONNECTED STEINER SUBGRAPH [19], SUBGRAPH ISOMORPHISM [20], facility location problems [21], ODD CYCLE TRANSVERSAL [22], and 3-COLORING parameterized by the number of vertices with degree $\geqslant 4$ [23].

It is plausible to expect that other natural problems also have significantly faster parameterized algorithms on planar graphs. The reason for this optimism is twofold. First, even though the techniques used to obtain the results listed above are highly problem-specific, they suggest that planar graphs have rich structural properties that could be exploited when solving other problems. Second, it looks almost impossible to prove lower bounds ruling out subexponential algorithms for planar problems. To prove that a parameterized algorithm with running time $2^{o(k)}\cdot n^{\mathcal{O}(1)}$ violates ETH, one needs to give a reduction from 3SAT with m clauses to a planar instance with parameter $k=\mathcal{O}(m)$. In a typical reduction, we represent each bit of information in the 3SAT instance (e.g., values of the variables in the solution) by a small "gadget" in the planar graph. In order to encode the constraints of the input instance, we need to connect these gadgets in an appropriate way. However, in a planar graph, we need to introduce some kind of "crossing gadgets" in order to realize these connections. To realize the constraints given by the $\mathcal{O}(m)$ clauses, it may be necessary to introduce up to $\mathcal{O}(m^2)$ crossings. As each crossing typically increases the parameter, we end up with an instance having parameter $k=\mathcal{O}(m^2)$, which is only sufficient to rule out $2^{o(\sqrt{k})}\cdot n^{\mathcal{O}(1)}$-time algorithms. Thus the appearance of many crossing gadgets seems to be an inherent limitation preventing stronger lower bounds. This may suggest that running times of the form $2^{o(k)}\cdot n^{\mathcal{O}(1)}$ are achievable for many problems.

Our contribution: In this paper we address two network design problems on planar graphs for which the existence of subexponential parameterized algorithms was open. Given a graph G with a subset T of vertices distinguished as terminals, the SUBSET TSP problem asks for a shortest closed walk visiting the terminals in any order. Parameterized by the number $k=|T|$ of terminals, the problem is fixed-parameter tractable in arbitrary graphs: it can be solved in time $2^k\cdot n^{\mathcal{O}(1)}$ by first computing the distance between every pair of terminals, and then solving the resulting k-terminal instance using the standard Bellman-Held-Karp dynamic programming algorithm. Klein and Marx [14] showed that if G is an undirected planar graph with polynomially bounded edge weights, then the problem can be solved significantly faster, in time $2^{\mathcal{O}(\sqrt{k}\log k)}\cdot n^{\mathcal{O}(1)}$. The limitations of polynomial weights and undirected graphs are inherent to this algorithm: it starts with computing a locally 4-step optimal solution (which requires polynomial weights to terminate in polynomial time) and relies on an elaborate subtour-replacement argument (which breaks down if the tour has an orientation). The main argument is the unexpected and somewhat mysterious claim that the union of an optimal and a locally 4-step optimal tour has treewidth $\mathcal{O}(\sqrt{k})$.

Our first result is a more robust and perhaps less mysterious algorithm that achieves the same running time, but does not suffer from these limitations.

Theorem I.1. *Given an edge-weighted directed planar graph G with terminals T,* SUBSET TSP *parameterized by $k=|T|$ can be solved in time $2^{\mathcal{O}(\sqrt{k}\log k)}n^{\mathcal{O}(1)}$.*

The proof of Theorem I.1 has the same high-level idea as the algorithm of Klein and Marx [14]: a family of $2^{\mathcal{O}(\sqrt{k}\log k)}$ subsets of terminals is computed, followed by applying a variant of the Bellman-Held-Karp dynamic programming algorithm that considers only subsets of terminals that appear in this family. However, the way we compute such a family is very different: the construction of Klein and Marx [14] crucially relies on how the optimal solution interacts with the locally 4-step optimal solution (e.g., they cross each other $\mathcal{O}(k)$ times), while our argument here does not use any such assumption. For directed graphs, we can extract much fewer properties of the structure of the solution or how it interacts with some other object. For example, we cannot require that the optimum solution is non-self-crossing and the number of self-crossings cannot be even bounded by a function of k. Thus in order to find an algorithm working on directed graphs, we need to use more robust algorithmic ideas that better explain why it is possible to have subexponential parameterized algorithms for this problem. In Section II, we highlight these new ideas in an overview of the algorithm of Theorem I.1.

Given an edge-weighted undirected graph G and a set T of terminal vertices, STEINER TREE asks for a minimum-weight tree connecting all the terminals. This problem is well known to be NP-hard, even in planar graphs [24]. Dreyfus and Wagner [25] gave a dynamic programming

algorithm that solves STEINER TREE in time $3^k \cdot n^{\mathcal{O}(1)}$ in arbitrary graphs. The running time of this algorithm was improved to $2^k \cdot n^{\mathcal{O}(1)}$ for small weights using fast subset convolution [26]. It is known that, assuming ETH, there is no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for the problem in general graphs and in fact it is conjectured that the 2^k factor cannot be improved to $(2-\epsilon)^k$ for any $\epsilon > 0$ [27].

In light of the long list of other subexponential parameterized problems on planar graphs, it is natural to expect that STEINER TREE can be solved in $2^{\mathcal{O}(\sqrt{k}\log k)} \cdot n^{\mathcal{O}(1)}$ time on planar graphs. In fact, this question has been posed as a natural open problem in various places [17], [18], [28]–[31]. As partial progress toward this goal, in the unweighted case, a subexponential algorithm parameterized by the *number of edges of the solution* was found [17], [18]. However, the number of edges can be of course much larger than the number of terminals, hence an algorithm that is subexponential in the number of edges is not necessarily subexponential in the number of terminals. We show here that there was a reason why, despite significant efforts, no such algorithm was found so far: assuming ETH, there is no subexponential parameterized algorithm for STEINER TREE on planar graphs.

Theorem I.2. *Unless the ETH fails, there is no $2^{o(k)} \cdot n^{\mathcal{O}(1)}$ time algorithm for* STEINER TREE *on an unweighted and undirected planar graph with $k = |T|$ terminals.*

Thus unlike many other problems, STEINER TREE parameterized by the number of terminals does not become dramatically simpler with the restriction of planarity. This is highly unexpected: STEINER TREE seems to be the first "genuinely planar" problem where there is no significant speedup when restricted to planar graphs, and the $2^{\mathcal{O}(k)}$ factor for arbitrary graphs cannot be improved. The informal expression "genuinely planar" emphasizes the fact that input of STEINER TREE is planar graph with a distinguished subset of vertices and there is no other extra, nonplanar information encoded in the input. For example, it was known before that DIRECTED STEINER NETWORK (given a directed graph G and requests $(s_1, t_1), \ldots, (s_k, t_k)$, find a subgraph of minimum weight that contains an $s_i \to t_i$ path for every i) can be solved in time $n^{\mathcal{O}(k)}$ on general graphs, and there is no $f(k)n^{o(k)}$ time algorithm even on planar graphs [19]. However, this problem is not genuinely planar, as the pairs (s_i, t_i) can encode arbitrary interactions that do not respect planarity.

Theorem I.2 makes the previous subexponential algorithms (including Theorem I.1 for SUBSET TSP on directed graphs) even more surprising: apparently there is no general rule why these problems should have subexponential parameterized algorithms on planar graphs, and it could have turned out for other problems as well that planarity does not allow any dramatic speedup. This negative result also changes our expectations for future work in this direction: we cannot take it for granted that most reasonable problems have subexponential parameterized algorithms on planar graphs and now it seems to be a very real possibility that other natural problems behave similarly to STEINER TREE.

We need some explanation how it was possible to prove Theorem I.2: earlier we have argued that such lower bounds seem very unlikely, because one would need $\mathcal{O}(n^2)$ crossings when reducing from a 3SAT instance with $\mathcal{O}(n)$ variables and $\mathcal{O}(n)$ clauses. In the proof of Theorem I.2, we are doing something unusual: in the created planar instance, we are not only introducing $\mathcal{O}(n)$ gadgets, each representing one bit of the 3SAT instance (as it is usually done in reductions), but we introduce also gadgets representing larger groups of bits. The crucial trick is that we can create crossings where an information flow of one bit crosses the information flow of many bits, and this crossing increases the parameter only by $\mathcal{O}(1)$. With such crossings, the total number of crossing gadgets can be limited and we can make sure that the parameter becomes $\mathcal{O}(n)$. The catch is that the reduction is no longer polynomial: the representation of large groups of bits require a planar graph that is exponentially large. However, we can argue that a subexponential parameterized algorithm on this exponentially large graph would result in a $2^{o(n)}$ algorithm for 3SAT, violating ETH.

The reduction in the proof of Theorem I.2 is a "hybrid" reduction in the sense that it combines different proof strategies. In typical NP-hardness proofs, one constructs small gadgets that represent one bit or information or have a constant number of different states. In typical W[1]-hard proofs, the challenge is to construct large gadgets that can have many different states (corresponding to, say, the choice of a vertex in a clique). The proof of Theorem I.2 combines these two ideas: we need both small gadgets representing single bits and large gadgets having many different states. Additionally, we use the idea of splitting the variables into groups and allowing a blowup of the size of the instance by a factor that is exponential in the size of the groups (as it is done in, e.g., [12], [32]). Thus our reduction combines in a novel way many of the insights that we have learned in the past decades about proving lower bounds on the exact complexity of hard algorithmic problems.

Our final result shows that there is still some way in which subexponentiality appears for planar STEINER TREE. On a high level, the proof of this theorem follows the same approach as a corresponding result for rectilinear Steiner tree [33].

Theorem I.3. *Given an edge-weighted planar undirected graph G with n vertices and a set $T \subseteq V(G)$ of terminals, one can find a minimum-cost Steiner tree for T in time $n^{\mathcal{O}(\sqrt{k})} \cdot W$, where W is the maximum weight of an edge and $k = |T|$.*

Note that this running time is incomparable to the $2^k \cdot n^{\mathcal{O}(1)}$ time, available for general graphs.

It is known that unweighted STEINER TREE in planar graphs admits a polynomial kernel when parameterized by the number of edges in the solution [17] and the number of nonterminal vertices in the solution [34]. A natural question is whether this can be improved to a polynomial kernel parameterized by the number of terminals. While we cannot answer this question here, a simple combination of Theorems I.2 and I.3 excludes, under ETH, a kernelization algorithm that produces a polynomial kernel which does not increase the number of terminals more than by a constant factor.

Corollary I.4. *Suppose there is a polynomial-time algorithm that, given an unweighted planar* STEINER TREE *instance* (G, T) *and an integer* p*, computes another unweighted planar* STEINER TREE *instance* (G', T') *and an integer* p'*, such that* $|T'| = \mathcal{O}(|T|)$*,* $|G'|$ *bounded polynomially in* $|T|$*, and* (G, T) *admits a Steiner tree of size at most* p *if and only if* (G', T') *admits a Steiner tree of size at most* p'*. Then the ETH fails.*

In this extended abstract we give only an overview of the approach leading to the subexponential parameterized algorithm for DIRECTED SUBSET TSP, that is, the proof of Theorem I.1. Complete proofs of all the abovementioned results (Theorems I.1, I.2, and I.3) can be found in the full version of this work, which is available on arXiv [35].

II. DIRECTED TRAVELING SALESMAN: OVERVIEW

We first describe the high-level strategy of restricting the standard dynamic programming algorithm to a smaller family of candidate states. Then we explain the main idea of how such a family of candidate states can be obtained; however, we introduce multiple simplifying assumptions and hide most of the technical problems. Finally, we briefly review the issues encountered when making the approach work in full generality, and explain how we cope with them. We strongly encourage the reader to read this section before proceeding to the formal description, as in the formal layer many of the key ideas become somehow obfuscated by the technical details surrounding them.

A. Restricted dynamic programming

Restricting dynamic programming to a small family of candidates states is by now a commonly used technique in parameterized complexity. The idea is as follows. Suppose that we search for a minimum-cost solution to a combinatorial problem, and this search can be expressed as solving a number of subproblems in a dynamic programming fashion, where each subproblem corresponds to a *state* from a finite state space $\mathcal{S}$. Usually, subproblems correspond to partial solutions, and transitions between states correspond to extending one partial solution to a larger partial solution at some cost, or combining two or more partial solutions to a larger one. For simplicity, assume for now that we only extend single partial solutions to larger ones, rather than combine multiple partial solutions. Then the process of assembling the final solution from partial solutions may be described as a nondeterministic algorithm that guesses consecutive extensions, leading from a solution to the most basic subproblem to the final solution for the whole instance. The sequence of these extensions is a path (called also a *computation path*) in a directed graph on $\mathcal{S}$ where the transitions between the states are the arcs. Then the goal is to find a minimum-weight path from the initial state to any final state, which can be done in time linear in the size of this state graph, provided it is acyclic.

In order to improve the running time of such an algorithm one may try the following strategy. Compute a subset of states $\mathcal{S}' \subseteq \mathcal{S}$ with the following guarantee: there is a computation path leading to the discovery of a minimum-weight solution that uses only states from $\mathcal{S}'$. Then we may restrict the search only to states from $\mathcal{S}'$. So the goal is to find a subset of states $\mathcal{S}'$ that is rich enough to "capture" some optimum solution, while at the same time being as small as possible so that the algorithm is efficent.

Let us apply this principle to DIRECTED SUBSET TSP. Consider first the most standard dynamic programming algorithm for this problem, working on general graphs in time $2^k \cdot n^{\mathcal{O}(1)}$, where we denote $k = |T|$ by convention. Each subproblem is described by a subset of terminals $S \subseteq T$ and two terminals $s_1, s_2 \in S$. The goal in the subproblem is to find the shortest tour that starts in s_1, ends in s_2, and visits all terminals of S along the way. The transitions are modelled by a possibility of extending a solution for the state (S, s_1, s_2) to a solution for the state $(S \cup \{s'\}, s_1, s')$ for any $s' \notin S$ at the cost of adding the shortest path from s_2 to s'. The minimum-weight tour can be obtained by taking the best among solutions obtained as follows: for any $s_1, s_2 \in T$, take the solution for the subproblem (T, s_1, s_2) and augment it by adding the shortest path from s_2 to s_1.

This is not the dynamic programming algorithm we will be improving upon. The reason is that restricting ourselves to constructing one interval on the tour at a time makes it difficult to enumerate a small subfamily of states capturing an optimum solution.

Instead, we consider a more involved variant of the above dynamic programming routine, which intuitively keeps track of $\mathcal{O}(\sqrt{k})$ intervals on the tour at a time. More precisely, each subproblem is described by a state defined as a pair $(S, \mathcal{M})$, where $S \subseteq T$ is a subset of terminals to be visited, and $\mathcal{M}$ (also called *connectivity pattern*) is a set of pairwise disjoint pairs of terminals from S, where $|\mathcal{M}| \leqslant C\sqrt{k}$ for some universal constant C. The goal in the subproblem is to compute a family of paths $\mathcal{P}_{(S,\mathcal{M})}$ of minimum possible weight having the following properties: for each $(s_1, s_2) \in \mathcal{M}$ there is a path in $\mathcal{P}_{(S,\mathcal{M})}$ that leads from s_1 to s_2, and each terminal from S lies on some path in $\mathcal{P}_{(S,\mathcal{M})}$. Note, however, that we do not specify, for

each terminal from S, on which of the paths it has to lie. Solutions to such subproblems may be extended by single terminals as in the standard dynamic programming, but they can be also combined in pairs. Precisely, given solutions $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively for $(S_1, \mathcal{M}_1)$ and $(S_2, \mathcal{M}_2)$, where $S_1 \cap S_2 = \emptyset$, we may merge these solutions into a solution for $(S_1 \cup S_2, \mathcal{M})$ by connecting paths from $\mathcal{P}_1$ and $\mathcal{P}_2$ using shortest paths between respective start and end vertices, so that the connectivity pattern $\mathcal{M}$ is obtained at the end. Since we assume that $|\mathcal{M}_1|, |\mathcal{M}_2|, |\mathcal{M}| \leqslant C\sqrt{k}$, there are only $k^{\mathcal{O}(\sqrt{k})}$ ways to perform such a merge. While this dynamic programming formally does not conform to the "linear view" described in the paragraphs above, as it may merge partial solutions for two simpler states into a larger partial solution, it straightforward to translate the concept of restricting the state space to preserve the existence of a computation path (here, rather a computation tree) leading to a minimum-cost solution.

Observe that since in a state $(S, \mathcal{M})$ we stipulate the size of $\mathcal{M}$ to be $\mathcal{O}(\sqrt{k})$, the total number of states with a fixed subset $S \subseteq T$ is $k^{\mathcal{O}(\sqrt{k})}$. Thus, from the discussion above we may infer the following lemma, stated here informally.

Lemma II.1 (informal statement)**.** *Let (G, T) be an instance of* DIRECTED SUBSET TSP. *Suppose we are also given a family $\mathcal{B}$ of subsets of T with the following guarantee: there is a computation path of the above dynamic programming leading to an optimum solution that uses only states of the form $(S, \mathcal{M})$ where $S \in \mathcal{B}$. Then we can find an optimum solution for the instance (G, T) in time $k^{\mathcal{O}(\sqrt{k})} \cdot (|\mathcal{B}| \cdot |G|)^{\mathcal{O}(1)}$.*

Concluding, we are left with constructing a family $\mathcal{B}$ of subsets of T that satisfies the prerequisites of Lemma II.1 and has size $k^{\mathcal{O}(\sqrt{k})}$, provided the underlying graph G is planar. For this, we will crucially use topological properties of G given by its planar embedding.

B. Enumerating candidate states

Suppose (G, T) is the input instance of DIRECTED SUBSET TSP where G is planar. Without loss of generality we may assume that G is strongly connected. Fix some optimum solution W, which is a closed walk in the input graph G that visits every terminal.

Simplifying assumptions: We now introduce a number of simplifying assumptions. These assumptions are made with loss of generality, and we introduce them in order to present our main ideas in a setting that is less obfuscated by technical details.

- (A1) Walk W is in fact a simple directed cycle, without any self-intersections. In particular, the embedding of W in the plane is a closed curve without self-crossings; denote this curve by δ.
- (A2) The walk W visits every terminal exactly once, so that we may speak about the (cyclic) order of visiting terminals on W.

Note that Assumption A2 follows from A1, but we prefer to state them separatedly as we first obtain Assumption A2 and then discuss Assumption A1.

We will also assume that shortest paths are unique in G, but this can be easily achieved by perturbing the weights of edges of G slightly.

Suppose now that we have another closed curve γ in the plane, without self-crossings, that crosses δ in $p = \mathcal{O}(\sqrt{k})$ points, none of which is a terminal. Curve γ divides the plane into two regions—say R_1, R_2—and thus δ is divided into p intervals which are alternately contained in R_1 and R_2. Let S be the set of terminals visited on the intervals contained in R_1. Then it is easy to see that S is a good candidate for a subset of terminals that we are looking: S forms at most $\mathcal{O}(\sqrt{k})$ contiguous intervals in the order of visiting terminals by W, and hence for the connectivity pattern $\mathcal{M}$ consisting of the first and last terminals on these intervals, the state $(S, \mathcal{M})$ would describe a subproblem useful for discovering W.

However, we are not really interested in capturing one potentially useful state, but in enumerating a family of candidate states that contains a complete computation path leading to the discovery of an optimum solution. Hence, we rather need to capture a hierarchical decomposition of T using curves γ as above, so that terminal subsets S induce the sought computation path. For this, we will use the notion of *sphere-cut decompositions* of planar graphs, and the well-known fact that every k-vertex planar graph admits a sphere-cut decomposition of width $\mathcal{O}(\sqrt{k})$.

Sphere-cut decompositions: A *branch decomposition* of a graph G is a tree $\mathcal{T}$ with every internal node of degree 3, together with a bijection η from the edges of G to leaves of $\mathcal{T}$. For every edge e of $\mathcal{T}$, the removal of e from $\mathcal{T}$ splits $\mathcal{T}$ into two subtrees, say $\mathcal{T}_e^1$ and $\mathcal{T}_e^2$. This naturally induces a partition $\{F_e^1, F_e^2\}$ of the edge set of G as follows: F_e^1 comprises edges mapped by η to leaves residing in $\mathcal{T}_e^1$, and symmetrically for F_e^2 and $\mathcal{T}_e^2$. The *width* of the edge e is the number of vertices of G incident to both an edge of F_e^1 and to an edge of F_e^2, and the *width* of a branch decomposition $(\mathcal{T}, \eta)$ is the maximum width among its edges. The *branchwidth* of a graph G is the minimum possible width of a branch decomposition of G. It is well-known that a planar graph on k vertices has branchwidth $\mathcal{O}(\sqrt{k})$ (see e.g. [36]).

After rooting a branch decomposition $(\mathcal{T}, \eta)$ in any node, it can be viewed as a hierarchical decomposition of the edge set of G using vertex cuts of size bounded by the width of the decomposition. Seymour and Thomas [37] proved that in plane graphs we can always find an optimum-width branch decomposition that somehow respects the topology of the plane embedding of a graph. Precisely, having fixed a plane embedding of a connected graph G, call a closed curve γ in the plane a *noose* if γ has no self-crossings and it crosses the embedding of G only at vertices; in particular it does

not intersect any edge of G[1]. Such a curve γ divides the plane into two regions, which naturally induces a partition of the edge set of G into edges that are embedded in the first, respectively the second region. A *sphere-cut decomposition* of G is a branch decomposition $(\mathcal{T}, \eta)$ where in addition every edge e of $\mathcal{T}$ is assigned its noose $\gamma(e)$ that induces exactly the partition $\{F_e^1, F_e^2\}$ of the edge set of G in the sense above. Then the result of Seymour and Thomas [37] may be stated as follows: every connected planar graph has a sphere-cut decomposition of width equal to its branchwidth[2]. Together with the square-root behavior of the branchwidth of a planar graph, this implies the following.

Theorem II.2 (see e.g. [36]). *Every connected plane graph on k vertices has a sphere-cut decomposition of width at most $\alpha\sqrt{k}$, for some constant α.*

Turning back to our DIRECTED SUBSET TSP instance (G, T) and its optimum solution W, our goal is to enumerate a possibly small family of subsets of T that contains some complete computation path leading to the discovery of W. The remainder of the construction is depicted in Figure 1 and we encourage the reader to analyze it while reading the description. The description is divided into "concepts", which are not steps of the algorithm, but of the analysis leading to its formulation.

Concept 1: adding a tree: Take any minimal tree H_0 in the underlying undirected graph of G spanning all terminals of T. We may assume that H_0 contains at most k leaves that are all terminals, at most $k-2$ vertices of degree at least 3, and otherwise it consists of at most $2k-3$ simple paths connecting these leaves and vertices of degree at least 3 (further called *special* vertices of H_0). To avoid technical issues and simplify the picture, we introduce another assumption.

(A2) Walk W and tree H_0 do not share any edges.

Let H be the graph formed by the union of W and H_0. Even though both W and H consists of at most $2k$ simple paths in G, the graph H may have many vertices of degree more than 3. This is because a subpath Q between two consecutive terminals on W and a path P in H_0 that connects two special vertices of H_0 may cross many times. The intuition is, however, that the planar structure of H roughly resembles a structure of a planar graph on $\mathcal{O}(k)$ vertices, and a sphere-cut decomposition of this planar graph of width $\mathcal{O}(\sqrt{k})$ should give rise to the sought hierarchical partition of terminals leading to the discovery of W by the dynamic programming algorithm.

[1]In standard literature, e.g. [37], a noose is moreover required to visit every face of G at most once; in this paper we do not impose this restriction.

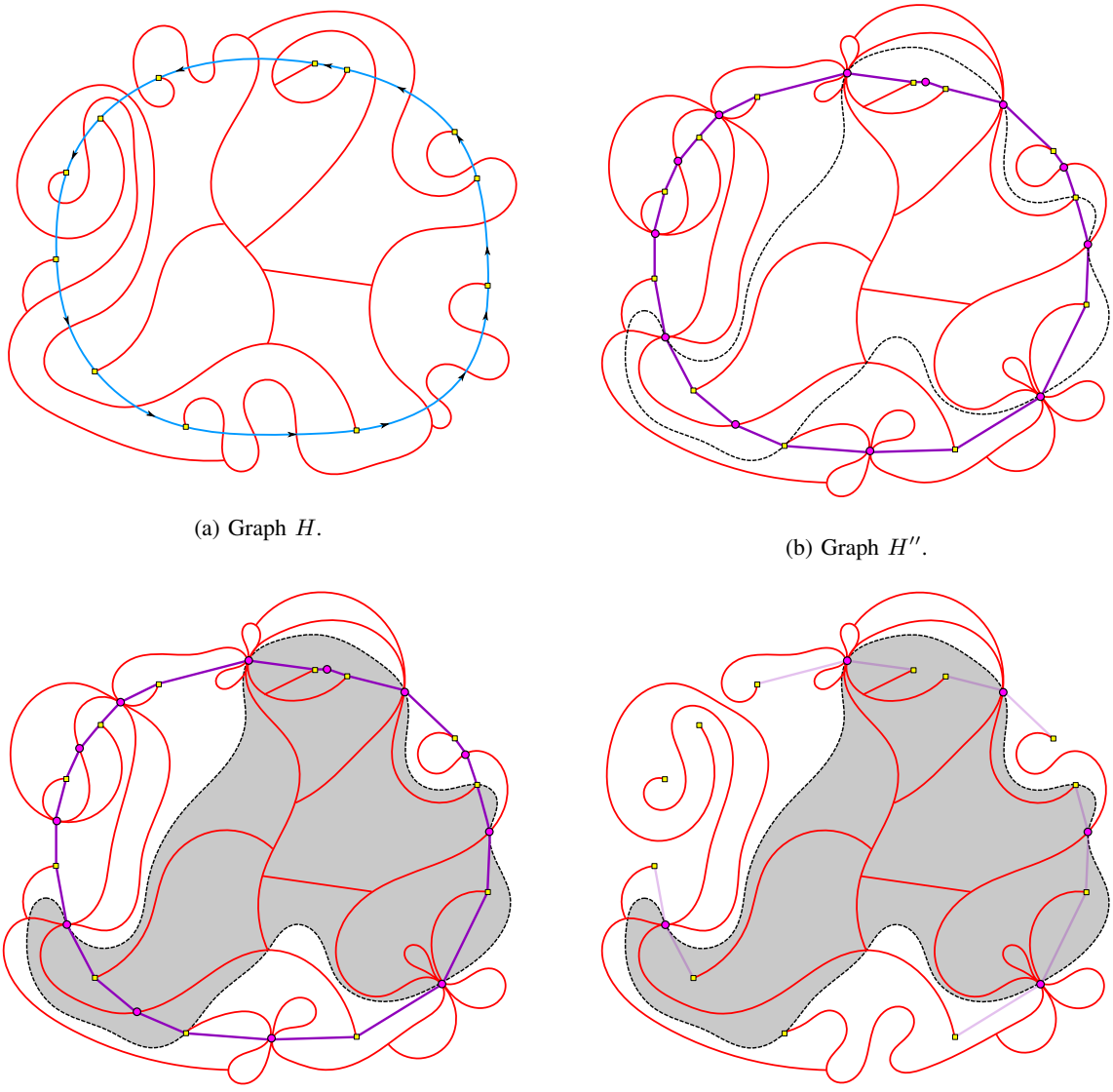
[2]In [37] it is also assumed that the graph is bridgeless, which corresponds to the requirement that every face is visited by a noose at most once. It is easy to see that in the absence of this requirement it suffices to assume the connectivity of the graph.

Let us remark that, of course, the definition of the graph H relies on the (unknown to the algorithm) solution W, though the tree H_0 can be fixed and used by the algorithm. At the end we will argue that having fixed H_0, we may enumerate a family of $k^{\mathcal{O}(\sqrt{k})}$ candidates for nooses in a sphere-cut decomposition of H. Roughly, for each such noose γ we consider the bi-partition of terminals according to the regions of the plane minus γ in which they lie, and we put all terminal subsets constructed in this manner into a family $\mathcal{B}$, which is of size $k^{\mathcal{O}(\sqrt{k})}$. Then restricting the dynamic programming algorithm to $\mathcal{B}$ as in Lemma II.1 gives us the required time complexity.

Concept 2: Contracting subpaths of W: Hence, the goal is to simplify the structure of H so that it admits a sphere-cut decomposition of width $\mathcal{O}(\sqrt{k})$. Consider any pair of terminals t_1, t_2 visited consecutively on W, and let P be the subpath of W from t_1 to t_2. Consider contracting all internal vertices on P into a single vertex, thus turning P into a path P' on 2 edges and 3 vertices. Let H' be the graph obtained from H by contracting each path between two consecutive terminals on W in the manner described above. Observe that thus, H' has less than $3k$ vertices of degree at least 3: there are at most $2k$ vertices on the contracted W in total, and there can be at most $k-2$ vertices of degree at least 3 on H_0 that do not lie on W. If we now take H' and contract every maximal path with internal vertices of degree 2 to a single edge, we turn H' into a planar graph H'' on at most $3k$ vertices. Then H'' has a sphere-cut decomposition of width $\leqslant \alpha\sqrt{3k}$, say $(\mathcal{T}, \eta, \gamma(\cdot))$.

Consider the family $\mathcal{D}$ of subsets of terminals constructed as follows. For each noose $\gamma(e)$ for $e \in \mathcal{T}$, that is, appearing in the sphere-cut decomposition $(\mathcal{T}, \eta, \gamma(\cdot))$, and each partition (X, Y) of terminals traversed by $\gamma(e)$ (there are at most $\alpha\sqrt{3k}$ such terminals, so $2^{\mathcal{O}(\sqrt{k})}$ such partitions), add to $\mathcal{D}$ the following two terminal subsets: the set of terminals enclosed by $\gamma(e)$ plus X, and the set of terminals excluded by $\gamma(e)$ plus Y. It can be now easily seen that $\mathcal{D}$ contains a complete computation path that we are looking for, as each terminal subset included in $\mathcal{D}$ forms at most $\mathcal{O}(\sqrt{k})$ contiguous intervals in the cyclic order of terminals on W, and the decomposition tree $\mathcal{T}$ shows how our dynamic programming should assemble subsets appearing in $\mathcal{D}$ in pairs up to the whole terminal set. In other words, if we manage to construct a family $\mathcal{B}$ of size $k^{\mathcal{O}(\sqrt{k})}$ with a guarantee that it contains the whole $\mathcal{D}$, then we will be done by Lemma II.1.

Concept 3: Enumeration by partial guessing: Obviously, the graph H'' is not known to the algorithm, as its definition depends on the fixed optimum solution W. Nevertheless, we may enumerate a reasonably small family of candidates for nooses used in its sphere-cut decomposition $(\mathcal{T}, \eta, \gamma(\cdot))$. The main idea is that even though the full structure of H'' cannot be guessed at one shot within $k^{\mathcal{O}(\sqrt{k})}$



(a) Graph H.

(b) Graph H''.

(c) Graph H'' with noose γ.

(d) Graph $H_{\mathcal{R}}$ and γ as a noose in it.

Figure 1: Construction of Section II-B. In panel 1a, we see graph H consisting of the union of solution W (in blue) and tree H_0 (in red). Terminals are depicted as yellow squares. In panel 1b, we see graph H'', obtained from H by contracting the interior of every subpath of W between two consecutive terminals to one vertex (in violet). Also, paths of vertices of degree 2 in H_0 are replaced by single edges, though this is not visible. Panel 1c depcicts noose γ in the graph H''. Then γ partitions the plane into two regions: R_1 (grayed) and R_2 (non-grayed), which induces a partition of the terminals into those contained in R_1, those contained in R_2, and those traversed by γ. Note that γ traverses two terminals, five vertices obtained from contracting subpaths of W to single vertices, and two vertices of H_0 of degree 3. Finally, in panel 1d we see the graph $H_{\mathcal{R}}$ used to enumerate γ. Here, $\mathcal{R}$ consists of those pairs of terminals that are consecutive on W and moreover γ traverses the vertex of H'' obtained from contracting the shortest path between them (which is a subpath of W). These constracted shortest paths are not a part of $H_{\mathcal{R}}$, so they are depicted with reduced opacity.

possibilities, each noose we are interested in traverses only at most $\alpha\sqrt{3k}$ vertices of H'', and hence it is sufficient to guess only this small portion of H''.

More precisely, let Q be the subset of those vertices of H'' that are obtained from contracting the subpaths of W between consecutive terminals. Fix a noose γ appearing in the sphere-cut decomposition of H'', that is, $\gamma = \gamma(e)$ for some $e \in \mathcal{T}$. Then γ traverses at most $\alpha\sqrt{3k}$ vertices of Q;

say that $R \subseteq Q$ is the set of these vertices. We can now enumerate a set of $k^{\mathcal{O}(\sqrt{k})}$ candidates for γ by performing the following steps (by *guessing* we mean iterating through all options):

1) Guess a set $\mathcal{R}$ of at most $\alpha\sqrt{3k}$ pairs of distinct terminals.
2) For each $(s,t) \in \mathcal{R}$, take the shortest path $P_{(s,t)}$ from s to t and consider contracting it to a single vertex $p_{(s,t)}$.
3) Take the fixed tree H_0 that spans terminals in G, apply the above contractions in G, and let $H_{\mathcal{R}}$ be the graph to which H_0 is transformed under these contractions.
4) Enumerate all nooses γ that meet $H_{\mathcal{R}}$ only at terminals and vertices of degree at least 3, and traverse at most $\alpha\sqrt{3k}$ such vertices.

In Step 1 we have at most $k^{\mathcal{O}(\sqrt{k})}$ options for such a set $\mathcal{R}$, and the contractions in Steps 2 and 3 turn H_0 into a planar graph $H_{\mathcal{R}}$ with $\mathcal{O}(k)$ vertices. It is not hard to convince oneself that in such a graph, there are only $k^{\mathcal{O}(\sqrt{k})}$ nooses satisfying the property expressed in the Step 4, so all in all we enumerate at most $k^{\mathcal{O}(\sqrt{k})}$ curves in the plane, each traversing at most $\alpha\sqrt{3k}$ terminals. Now, to conclude the construction of $\mathcal{B}$ we do as follows. For each enumerated curve γ, and each partition (X,Y) of terminals traversed by γ, we include into $\mathcal{B}$ two terminal subsets: the set of terminals enclosed by γ plus X, and the set of terminals excluded by γ plus Y. Thus $|\mathcal{B}| = k^{\mathcal{O}(\sqrt{k})}$.

It remains to argue that $\mathcal{B}$ contains the whole family $\mathcal{D}$ that was given by the sphere-cut decomposition $(\mathcal{T}, \eta, \gamma(\cdot))$ of H'', so that Lemma II.1 may be applied. It should be quite clear that it is sufficient to show that every noose γ appearing in $(\mathcal{T}, \eta, \gamma(\cdot))$ is enumerated in Step 4 of the procedure from the previous paragraph. However, nooses with respect to $H_{\mathcal{R}}$ are formally not necessarily nooses with respect to H'', as we wanted. Nevertheless, if a noose γ appears in the sphere-cut decomposition $(\mathcal{T}, \eta, \gamma(\cdot))$ of H'', and we take $\mathcal{R}$ to be the set of pairs of consecutive terminals on W such that γ passes through the contracted vertices $p_{(s,t)}$ exactly for $(s,t) \in \mathcal{R}$, then after dropping parts of H'' not appearing in $H_{\mathcal{R}}$, γ becomes a noose enumerated for $H_{\mathcal{R}}$. Therefore, the terminal partitions raised by γ are still included in $\mathcal{B}$ as we wanted, and we are done.

C. Traps, issues, and caveats

The plan sketched in the previous section essentially leads to an algorithm with the promised time complexity, modulo Assumptions A1, A2, A3, and a number of technical details of minor relevance. Assumptions A2 and A3 are actually quite simple to achieve without loss of generality. It is Assumption A1 that was a major conceptual obstacle.

For Assumption A2, we may at the very beginning perform the following reduction. For every original terminal t, introduce a new terminal t' and edges (t,t') and (t',t) of weight 0 to the graph; t' and these edges are embedded in any face incident to t. The new terminal set consists of terminals t' for all original terminals t. In this way, any closed walk visiting any new terminal t' has to make a detour of weight 0 using arcs (t,t') and (t',t), and we may assume that an optimal solution makes only one such detour for each new terminal t'. Thus we achieve Assumption A2. Actually, the fact that we may assume that every terminal is incident to one non-trivial face and one trivial face between (t,t') and (t',t) also helps in solving technical issues later on.

For Assumption A3, observe that in the reasoning we relied only on the fact that H_0 is a tree spanning all terminals that has at most k leaves and at most $k-2$ vertices of degree at least 3. In particular, we did not use any metric properties of H_0. In fact, the reader may think of H_0 as a combinatorial object used to control the homotopy group of the plane with terminals pierced out: for any non-self-crossing curve γ on the plane, we may infer how terminals are partitioned into those enclosed by γ, excluded by γ, and lying on γ just by examining the consecutive intersections of γ with H_0. Therefore, instead of choosing H_0 arbitrarily, we may add it to the graph artificially at the very beginning, say using edges of weight $+\infty$. In this way we make sure that the optimum solution W does not use any edge of H_0.

Finally, let us examine Assumption A1: the optimum solution W is a simple directed cycle without self-intersections. Unfortunately, this assumption may not hold in general. Consider the example depicted in Figure 2, where we have a directed planar graph with two terminals s,t, and the only closed walk visiting both s and t consists of two paths, one from s to t and the second from t to s, that intersect each other an unbounded number of times. Therefore, in general the optimum solution W may have an unbounded number of self-crossings. Nevertheless, we may still develop some kind of a combinatorial understanding of the topology of W.

It will be convenient to assume that no edge of the graph is traversed by W more than once; this can be easily achieved by copying each edge $|T|$ times, and using a different copy for each traversal. Consider two visits of the same vertex u by W; let e_1, e_2 be the edges incident to u used by W just before and just after the first visit, and define f_1, f_2 in the same way for the second visit. Examine how e_1, e_2, f_1, f_2 are arranged in the cyclic order of edges around vertex u. If they appear in the interlacing order, i.e., (e_1, f_1, e_2, f_2) or (e_1, f_2, e_2, f_1), then we say that these two visits form a *self-crossing* of W. Intuitively, if the order is not interlacing, then we may slightly pull the two parts of the embedding of W near u corresponding to the visits so that they do not intersect. So topologically we do not consider such a self-intersection as a self-crossing. For two walks W_1, W_2 in G that do not share any common edges we define their *crossing* in a similar manner, as a common visit of a vertex u such that the cyclic order of edges used by W_1 and W_2 immediately

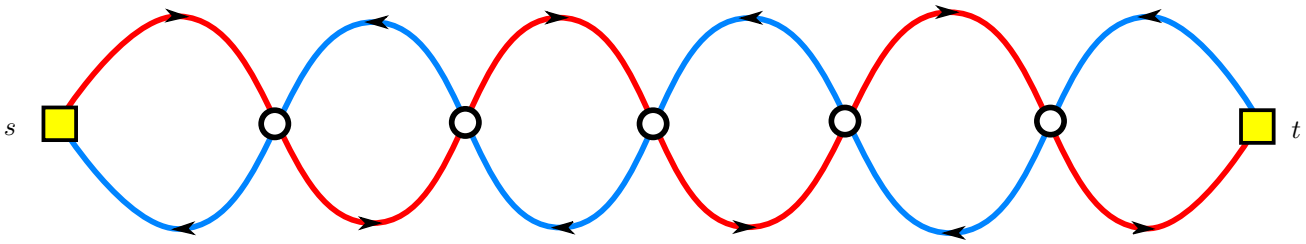


Figure 2: A planar DIRECTED SUBSET TSP instance with two terminals. The only solution consists of the union of the red path from s to t and the blue path from t to s. These two paths cross each other many times, which gives many self-crossings of the solution.

before and immediately after these visits is interlacing.

We now prove the following structural statement about self-crossings of W: we may always choose an optimal solution W so that the following holds. Consider any self-crossing of W at some vertex u (recall it consists of two visits of u) and say it divides W into two closed subwalks W_1 and W_2: W_1 is from the first visit of u to the second, and W_2 is from the second visit of u to the first. Then the subwalks W_1 and W_2 do not cross at all. This statement can be proved by iteratively "uncrossing" an optimum solution W as long as the structure of its self-crossings is too complicated. However, one needs to be careful in order not to split W into two closed curves when uncrossing.

It is not hard to observe that the statement given in the previous paragraph actually shows that the topology of W roughly resembles a cactus where each 2-connected component is a cycle (here, we assume that self-intersections that are not self-crossings are pulled slightly apart so that W does not touch itself there). See Figure 3 for reference. Then we show that W can be decomposed into $\mathcal{O}(k)$ subpaths $\mathcal{P} = \{B_1, \ldots, B_\ell\}$ such that:

- each path B_i has no terminal as an internal vertex and is the shortest path between its endpoints; and
- each path B_i may cross with at most one other path B_j.

To see this, note that the cactus structure of W may be described as a tree $\mathcal{T}$ with at most k leaves and at most $k-2$ vertices of degree at least 3. We have a pair of possibly crossing subpaths in the decomposition $\mathcal{P}$ per each maximal path with internal vertices of degree 2 in $\mathcal{T}$.

The idea now is as follows. In the previous section we essentially worked with the partition of W into subpaths between consecutive terminals, as Assumption A1 allowed us to do so. In the absence of this assumption, we work with the finer partition $\mathcal{P}$ as above. The fact that the paths of $\mathcal{P}$ interact with each other only in pairs, and in a controlled manner, makes the whole reasoning go through with the conceptual content essentially unchanged, but with a lot more technical details.

One nontrivial difference is that in the previous section we

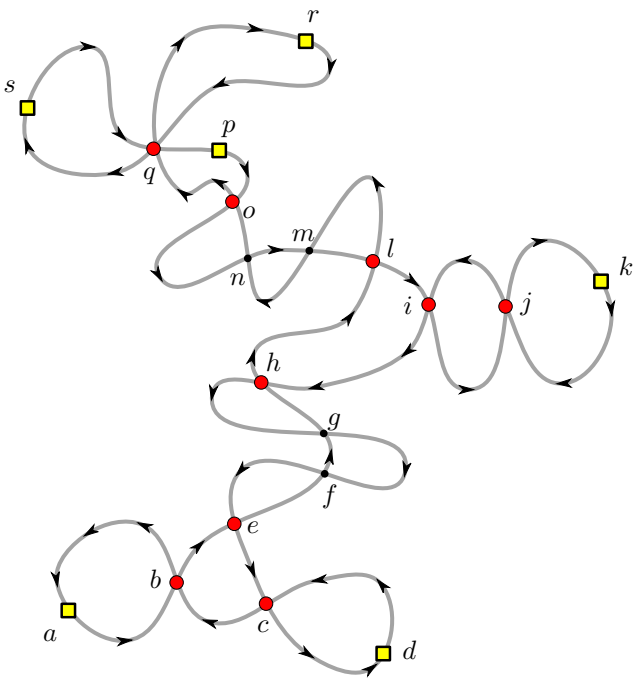


Figure 3: An exemplary solution.

were contracting shortest paths between pairs of consecutive terminals, so we had a small set of candidates for the endpoints of these paths: the terminals themselves. In the general setting, the decomposition statement above a priori does not give us any small set of candidates for endpoints of paths B_i. If we chose those endpoints as arbitrary vertices of the graph, we would end up with time complexity $n^{\mathcal{O}(\sqrt{k})}$ instead of promised $k^{\mathcal{O}(\sqrt{k})} \cdot \mathrm{poly}(n)$. Fortunately, the way we define the decomposition $\mathcal{P} = \{B_1, \ldots, B_\ell\}$ allows us to construct alongside also a set of at most k^4 *important* vertices such that each path B_i is the shortest path from one important vertex to another important vertex.

Finally, there are more technical problems regarding handling possible self-intersections of W that are not self-crossings. Recall that in our topological view of W, we would like not to regard such self-intersections as places

where W touches itself. In particular, when examining a sphere-cut decomposition of the union of W and H_0 after appropriate contractions, the nooses in this sphere-cut decomposition should not see such self-intersections as vertices through which they may or should travel. A resolution to this problem is to consider a "blow-up" of the original graph where each vertex is replaced by a large well-connected "cloud" of vertices, and each edge is replaced by a large matching of parallel edges leading from one cloud to another. Walks in the original graph naturally map to walks in the blow-up. Every original self-crossing maps to a self-crossing, and every original self-intersection that is not a self-crossing actually is "pulled apart": there is no self-intersection at this place anymore. This blow-up has to be performed at the very beginning of the proof, and hence we need to work on it throughout the whole reasoning. This technical layer is somehow conceptually simple, but contributes to the technical complexity of the argumentation.

Acknowledgements

This research is a part of projects that have received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreements No. 280152, 725978 (Dániel Marx) and 714704 (Marcin Pilipczuk). The research of Michał Pilipczuk is supported by Polish National Science Centre grant UMO-2013/11/D/ST6/03073. Michał Pilipczuk was also supported by the Foundation for Polish Science (FNP) via the START stipend programme.

References

[1] G. Borradaile and P. N. Klein, "The two-edge connectivity survivable-network design problem in planar graphs," *ACM Trans. Algorithms*, vol. 12, no. 3, pp. 30:1–30:29, 2016.

[2] K. Fox, P. N. Klein, and S. Mozes, "A polynomial-time bicriteria approximation scheme for planar bisection," in *STOC 2015*. ACM, 2015, pp. 841–850.

[3] P. N. Klein, C. Mathieu, and H. Zhou, "Correlation clustering and two-edge-connected augmentation for planar graphs," in *STACS 2015*, ser. LIPIcs, vol. 30. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2015, pp. 554–567.

[4] D. Eisenstat, P. N. Klein, and C. Mathieu, "Approximating k-center in planar graphs," in *SODA 2014*. SIAM, 2014, pp. 617–627.

[5] ——, "An efficient polynomial-time approximation scheme for Steiner forest in planar graphs," in *SODA 2012*. SIAM, 2012, pp. 626–638.

[6] M. Bateni, M. Hajiaghayi, P. N. Klein, and C. Mathieu, "A polynomial-time approximation scheme for planar Multiway Cut," in *SODA 2012*. SIAM, 2012, pp. 639–655.

[7] G. Borradaile, P. N. Klein, and C. Mathieu, "An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs," *ACM Trans. Algorithms*, vol. 5, no. 3, pp. 31:1–31:31, 2009.

[8] P. N. Klein, "A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights," *SIAM J. Comput.*, vol. 37, no. 6, pp. 1926–1952, 2008.

[9] M. Bateni, E. D. Demaine, M. Hajiaghayi, and D. Marx, "A PTAS for planar Group Steiner Tree via spanner bootstrapping and prize collecting," in *STOC 2016*. ACM, 2016, pp. 570–583.

[10] M. Bateni, M. T. Hajiaghayi, and D. Marx, "Approximation schemes for Steiner Forest on planar graphs and graphs of bounded treewidth," *J. ACM*, vol. 58, no. 5, pp. 21:1–21:37, 2011.

[11] M. Cygan, M. Pilipczuk, and M. Pilipczuk, "Known algorithms for Edge Clique Cover are probably optimal," *SIAM J. Comput.*, vol. 45, no. 1, pp. 67–83, 2016.

[12] D. Lokshtanov, D. Marx, and S. Saurabh, "Slightly superexponential parameterized problems," in *SODA 2011*. SIAM, 2011, pp. 760–776.

[13] 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*, vol. 52, no. 6, pp. 866–893, 2005.

[14] P. N. Klein and D. Marx, "A subexponential parameterized algorithm for Subset TSP on planar graphs," in *SODA 2014*. SIAM, 2014, pp. 1812–1830.

[15] ——, "Solving Planar k-Terminal Cut in $O(n^{c\sqrt{k}})$ time," in *ICALP 2012*, ser. LNCS, vol. 7391. Springer, 2012, pp. 569–580.

[16] D. Marx, "A tight lower bound for Planar Multiway Cut with fixed number of terminals," in *ICALP 2012*, ser. LNCS, vol. 7391. Springer, 2012, pp. 677–688.

[17] M. Pilipczuk, M. Pilipczuk, P. Sankowski, and E. J. van Leeuwen, "Network sparsification for Steiner problems on planar and bounded-genus graphs," in *FOCS 2014*. IEEE Computer Society, 2014, pp. 276–285.

[18] ——, "Subexponential-time parameterized algorithm for Steiner tree on planar graphs," in *STACS 2013*, ser. LIPIcs, vol. 20. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2013, pp. 353–364.

[19] R. H. Chitnis, M. Hajiaghayi, and D. Marx, "Tight bounds for Planar Strongly Connected Steiner Subgraph with fixed number of terminals (and extensions)," in *SODA 2014*. SIAM, 2014, pp. 1782–1801.

[20] F. V. Fomin, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, "Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering," in *FOCS 2016*. IEEE Computer Society, 2016, pp. 515–524.

[21] D. Marx and M. Pilipczuk, "Optimal parameterized algorithms for planar facility location problems using Voronoi diagrams," in *ESA 2015*, ser. LNCS, vol. 9294. Springer, 2015, pp. 865–877.

[22] D. Lokshtanov, S. Saurabh, and M. Wahlström, "Subexponential parameterized odd cycle transversal on planar graphs," in *FSTTCS 2012*, ser. LIPIcs, vol. 18. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2012, pp. 424–434.

[23] P. Aboulker, N. Brettell, F. Havet, D. Marx, and N. Trotignon, "Coloring graphs with constraints on connectivity," *Journal of Graph Theory*, vol. 85, no. 4, pp. 814–838, 2017.

[24] M. R. Garey and D. S. Johnson, "The rectilinear Steiner tree problem in NP complete," *SIAM Journal of Applied Mathematics*, vol. 32, pp. 826–834, 1977.

[25] S. E. Dreyfus and R. A. Wagner, "The Steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.

[26] J. Nederlof, "Fast polynomial-space algorithms using inclusion-exclusion," *Algorithmica*, vol. 65, no. 4, pp. 868–884, 2013.

[27] 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," *ACM Trans. Algorithms*, vol. 12, no. 3, pp. 41:1–41:24, 2016.

[28] M. Cygan, F. V. Fomin, B. M. P. Jansen, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, "Open problems for FPT School 2014," 2014, `http://fptschool.mimuw.edu.pl/opl.pdf`.

[29] G. Borradaile, P. N. Klein, D. Marx, and C. Mathieu, "Algorithms for optimization problems in planar graphs (Dagstuhl seminar 13421)," *Dagstuhl Reports*, vol. 3, no. 10, pp. 36–57, 2013.

[30] J. Erickson, P. N. Klein, D. Marx, and C. Mathieu, "Algorithms for optimization problems in planar graphs (Dagstuhl seminar 16221)," *Dagstuhl Reports*, vol. 6, no. 5, pp. 94–116, 2016.

[31] D. Marx, "What's next? future directions in parameterized complexity," in *The Multivariate Algorithmic Revolution and Beyond*, ser. LNCS, vol. 7370. Springer, 2012, pp. 469–496.

[32] J. Chen, B. Chor, M. Fellows, X. Huang, D. W. Juedes, I. A. Kanj, and G. Xia, "Tight lower bounds for certain parameterized NP-hard problems," *Inf. Comput.*, vol. 201, no. 2, pp. 216–231, 2005.

[33] F. V. Fomin, S. Kolay, D. Lokshtanov, F. Panolan, and S. Saurabh, "Subexponential algorithms for rectilinear Steiner tree and arborescence problems," in *SoCG 2016*, ser. LIPIcs, vol. 51. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2016, pp. 39:1–39:15.

[34] O. Suchý, "Extending the kernel for Planar Steiner Tree to the number of Steiner vertices," *Algorithmica*, vol. 79, no. 1, pp. 189–210, 2017.

[35] D. Marx, M. Pilipczuk, and M. Pilipczuk, "On subexponential parameterized algorithms for Steiner Tree and Directed Subset TSP on planar graphs," *CoRR*, vol. abs/1707.02190, 2017. [Online]. Available: http://arxiv.org/abs/1707.02190

[36] F. V. Fomin and D. M. Thilikos, "New upper bounds on the decomposability of planar graphs," *Journal of Graph Theory*, vol. 51, no. 1, pp. 53–81, 2006.

[37] P. D. Seymour and R. Thomas, "Call routing and the ratcatcher," *Combinatorica*, vol. 14, no. 2, pp. 217–241, 1994.