Planar Multiway Cut with Terminals on Few Faces

Sukanya Pandey[*] Erik Jan van Leeuwen[†]

November 2, 2021

Abstract

We consider the Edge Multiway Cut problem on planar graphs. It is known that this can be solved in $n^{O(\sqrt{t})}$ time [Klein, Marx, ICALP 2012] and not in $n^{o(\sqrt{t})}$ time under the Exponential Time Hypothesis [Marx, ICALP 2012], where t is the number of terminals. A generalization of this parameter is the number k of faces of the planar graph that jointly cover all terminals. For the related Steiner Tree problem, an $n^{O(\sqrt{k})}$ time algorithm was recently shown [Kisfaludi-Bak *et al.*, SODA 2019]. By a completely different approach, we prove in this paper that Edge Multiway Cut can be solved in $n^{O(\sqrt{k})}$ time as well. Our approach employs several major concepts on planar graphs, including homotopy and sphere-cut decomposition. We also mix a global treewidth dynamic program with a Dreyfus-Wagner style dynamic program to locally deal with large numbers of terminals.

1 Introduction

A graph with weighted edges, and a subset of its vertices called *terminals*, is given. How would you pairwise disconnect the terminals by removing a subset of minimum possible weight of the edges of the graph? Widely known as the (Edge) Multiway Cut problem, this question is a natural generalisation of the popular minimum (s,t)-cut problem. A variant of the problem was first introduced by T.C. Hu in 1969 [36]. Formally, we can define the Edge Multiway Cut problem as follows:

Edge Multiway Cut
Input: A graph G, weight function $\omega : E(G) \to \mathbb{Q}^+$, $T \subseteq V(G)$
Question: What is the minimum possible weight of the cut that pairwise separates the vertices in T?

The study of the complexity of Edge Multiway Cut was pioneered by Dahlhaus *et al.* in their seminal paper [21]. The authors showed that Edge Multiway Cut is NP-hard on arbitrary graphs for any fixed $t \geq 3$, where t is the number of terminals. They also showed that the problem is Max-SNP-hard for all fixed $t \geq 3$ even when all weights are unit, while giving an $\mathcal{O}(tnm\log(n^2/m))$-time approximation algorithm, which given an arbitrary graph and an arbitrary t finds a solution within a $2-2/t$ ratio of the optimum. Following these hardness results, began a twofold quest: to find approximation algorithms with a better approximation ratio [9, 38] and to find exact algorithms that were more efficient than any brute-force approach [44, 10, 54, 40, 20, 15, 34, 17]. Starting with the seminal work of Marx [44], significant research effort was also put in understanding the parameterized complexity of Edge Multiway Cut for other parameters, such as the size of the solution [54, 10, 15, 34, 20].

Restriction to planar graphs Given the hardness of the problem on arbitrary graphs, even for small constant values of t, it was but natural to look for specific graph classes where the problem might be more tractable with respect to the number of terminals t. In fact, Dahlhaus *et al.* had considered the restriction to planar graphs, which we call Planar Multiway Cut. When t is arbitrary, the problem still is NP-hard, even if all the edges of the planar graph are of unit weight. However, they showed that Planar Multiway Cut could be solved in polynomial time for any fixed t. For $t = 3$, their algorithm runs in time $\mathcal{O}(n^3 \log n)$ and for $t \geq 4$ in time $\mathcal{O}((4t)^t \cdot n^{2t-1} \log n)$. In the parameterized complexity parlance, their result implied that Planar Multiway Cut belongs to the class XP. The obvious next step was to find out if it was also FPT. In 2012,

[*]Dept. Information and Computing Sciences, Utrecht University, s.pandey1@uu.nl.
[†]Dept. Information and Computing Sciences, Utrecht University, e.j.vanleeuwen@uu.nl.

however, Marx [45] showed that assuming ETH holds, the problem does not admit any algorithm running in time $f(t) \cdot n^{o(\sqrt{t})}$ and showed that it is $W[1]$-hard.

In a companion paper, Klein and Marx [40] gave a subexponential algorithm for Planar Multiway Cut with a run-time of $2^{\mathcal{O}(t)} \cdot n^{\mathcal{O}(\sqrt{t})}$. Later, Colin de Verdière [22] showed that this result extends to surfaces of any fixed genus. He showed this even holds for the more general Edge Multicut problem, where given a set of terminal pairs, we ask for the smallest set of edges, which when removed, disconnects all the given terminal pairs. This yielded an algorithm which solves the problem in time $(g+t)^{\mathcal{O}(g+t)}n^{\mathcal{O}(\sqrt{g^2+gt})}$, where g is the genus of the surface. This bound is almost tight assuming ETH holds, as was recently shown by Cohen-Added *et al.* [19].

Parameterization by terminal face cover number Due to the intractability of Planar Multiway Cut when the number of terminals is part of the input, it is worthwhile to look for other parameters that might generalize t. In particular, we could look at imposing restrictions on the location of terminals in the input plane-embedded graph. An extensively explored such restriction is when all the terminals are present on a small number of faces of the planar graph. This restriction on the input graph was studied by Ford and Fulkerson in their classic paper [30]. The minimum number of faces of the input planar graph that cover all the terminals was termed the *terminal face cover number* $\gamma(G)$ by Krauthgamer *et al.* [41]. The terminal face cover number is of broad interest as a parameter and has been studied with respect to several cut and flow problems [41, 42, 13, 47], shortest path problems [32, 14], finding non-crossing walks [28], and the minimum Steiner tree problem [39]. In particular, Planar Steiner Tree has an algorithm running in time $2^{\mathcal{O}(\gamma(G)\log\gamma(G))}n^{\mathcal{O}(\sqrt{\gamma(G)})}$.

The case when $\gamma(G) = 1$ is known as an Okamura-Seymour graph. It was shown by Chen and Wu [13] that when $\gamma(G) = 1$ and G is biconnected, the minimum multiway cut in G forms a minimum Steiner tree in its planar dual[1]. Consequently, one can find a minimum Steiner tree in the dual graph using the algorithm by Erickson *et al.* [29] (see also Bern [6]) for the case when all the terminals lie on the outer face boundary of the graph. They also gave an approximation algorithm for the case when $\gamma(G) > 1$, which runs in time $\max\{\mathcal{O}(n^2\log n\log\gamma(G)), \mathcal{O}(\gamma(G)^2 n^{1.5}\log^2 n + tn)\}$ and finds a solution within a ratio of $2 - 2/t$ of the optimum multiway cut. However, for the case when $\gamma(G) > 1$, no exact algorithm was known.

1.1 Our contribution

In this paper, we resolve the complexity of Planar Multiway Cut parameterized by the terminal face cover number, hereafter referred to as k. Given an edge-weighted planar graph $G = (V, E)$ with a fixed embedding, a set of terminals $T \subseteq V$, and a collection of faces $\mathcal{F}$ that cover all the terminals in T, the goal is to find a minimum weight cut of G that pairwise separates the terminals in T from each other. Note that $|\mathcal{F}| = k$. We show the following:

Theorem 1.1. Planar Multiway Cut *can be solved in time* $2^{\mathcal{O}(k^2\log k)}n^{\mathcal{O}(\sqrt{k})}$. *Unless ETH fails, there can be no algorithm that solves* Planar Multiway Cut *and runs in time* $n^{o(\sqrt{k})}$.

Our main contribution is the algorithm. Since $k \leq t$, the lower bound immediately follows from Marx's result [45].

We note that our algorithm requires that the set $\mathcal{F}$ is known. Fortunately, such a set of size k can be computed in time $2^{\mathcal{O}(k)}n^{\mathcal{O}(1)}$ through the algorithm of Bienstock and Monma [7]. Hence, we can run their algorithm before our own, and this does not affect the running time of the final algorithm.

We also note that while the running time of our algorithm is reminiscent of the algorithm by Kisfaludi-Bak *et al.* [39] for Steiner Tree parameterized by the terminal face cover number, our algorithm is and needs to be substantially more involved. The intuition of their algorithm is to show that a minimum Steiner tree for a set of terminals that can be covered by k faces has bounded treewidth. Then they can 'trace' this tree by a recursive algorithm that guesses the vertices of the solution in a separator implied by the tree decomposition. While one can show that the dual of a minimum multiway cut is a subgraph of the dual of bounded treewidth, tracing a solution through such a tree decomposition is not straightforward and we need significant assistance from tools from topology, particulary homotopy, which were not needed for Steiner Tree.

Intuitively, we describe the high-level topology that must be respected by the dual of some optimum solution. Part of this topology is a planar graph, of which we can find a sphere-cut decomposition of width $O(\sqrt{k})$. We then

[1]They consider an augmented planar dual, which differs from the standard dual graph in that the outer face is represented by several vertices, one per interval of the boundary vertices between two consecutive terminals.

apply a dynamic programming routine on this decomposition to find the optimum solution. While this dynamic programming routine is sufficient to find the high-level structure, the number of terminals is too large for this to effectively deal with the local structure of the solution. We then merge the popular algorithm by Dreyfus-Wagner [25] for finding minimum Steiner trees, which runs in polynomial time in cases relevant for us [29, 6], with the algorithm of Frank-Schrijver [31] to find shortest paths homotopic to a given prescription. We argue that this prescription can be efficiently guessed. This leads to a neat dynamic program used to find the local structure, and then finally, an optimum solution.

In this sense, our algorithm has more in common with the approaches of Klein and Marx [40] and Colin de Verdière [22], which extensively rely on topological arguments, and particularly homotopy. However, significant effort is needed to generalize from the parameter number of terminals employed in those works to the parameter number of faces covering terminals. This not only affects the structural results that employ homotopy, but also makes the final algorithms more involved.

We provide a more detailed overview of our algorithm in Section 2. The full paper then follows.

1.2 Related Work We discuss related work in some more detail, particularly on the parameterized complexity of Multiway Cut. Until 2006, not much was known with regard to the parameterized complexity of Multiway Cut. Marx [44] got the ball rolling with his result that Node Multiway Cut, where one must remove a subset of vertices instead of edges of the input graph, is FPT parameterized by the size of the solution (hereby denoted by s). His algorithm ran in time $\mathcal{O}^*(4^{s^3})$. Since then, the run time has been considerably improved [54, 10, 15, 34, 20], with the current best being $\mathcal{O}^*(1.84^s)$ for Edge Multiway Cut [10], and $\mathcal{O}^*(2^s)$ for Node Multiway Cut [20]. For directed graphs, Chitnis *et al.* [17] showed that Node Multiway Cut and Edge Multiway Cut are equivalent and admit an FPT algorithm running in time $\mathcal{O}^*(2^{2^{\mathcal{O}(s)}})$.

For planar graphs, we also mention some other exact algorithms. In particular, Hartvigsen [35] gave an $O(t4^t n^{2t-4}\log n)$ time algorithm, which improved on the original exact algorithm by Dahlhaus *et al.* [21]. The simple algorithm by Yeh [55], unfortunately, seems incorrect [16]. The mentioned algorithm of Klein and Marx [40] is faster than all of them. Benz also considered the generalization to Edge Multicut, first with terminals only on the outer face [5] and for the general case [4]. Unfortunately, the latter general result seems to have several flaws [22]. The algorithm by Colin de Verdière [22] for this problem that we already mentioned is also faster and more general. Finally, for planar graphs and parameter s, Pilipczuk *et al.* [49] showed that even a polynomial kernel in s exists for Edge Multiway Cut, leading to an algorithm with running time $\mathcal{O}^*(2^{\mathcal{O}(\sqrt{s}\log s)})$. This kernel was recently extended to Node Multiway Cut by Jansen *et al.* [37].

We also discuss further works on approximation algorithms. We already mentioned several constant-factor approximations for general graphs [21, 9, 38] and planar graphs [13]. Bateni *et al.* [2] presented a PTAS for Planar Multiway Cut which combined the technique of brick-decomposition from [8], the clustering technique from [3], and a technique to find short cycles enclosing prescribed amounts of weight from [48]. The more general Edge Multicut problem is known to be APX-hard, even in trees. However, Cohen-Addad *et al.* [18] recently gave a $(1+\varepsilon)$-approximation scheme for Edge Multicut on graphs embedded on surfaces, with a fixed number of terminals. Their approximation scheme runs in time $(g+t)^{\mathcal{O}((g+t)^3)} \cdot (1/\varepsilon)^{\mathcal{O}(g+t)} \cdot n\log n$. It also contains uses of the Dreyfus-Wagner and Frank-Schrijver algorithm in a combination not dissimilar from our own, but in different circumstances.

Finally, our work leans on a proper specification of the homotopy of the dual of the solution. This approach was first applied explicitly to Edge Multicut (and by extension to Edge Multiway Cut) by Colin de Verdière [22] and is also evident in the work of Klein and Marx [40]. The understanding of homotopy and its uses in cut and flow problems on graphs on surfaces was built through a sequence of works, see *e.g.* [11, 12, 26, 27, 28].

1.3 Organization A high level description of the paper appears in Section 2. Later sections give most of the technical results, but proofs for results marked with ♣ are deferred to the full version.

2 Overview

At a high level, our approach is reminiscent of the one by Klein and Marx [40] as well as Colin de Verdière [22] used for the weaker parameter t, the number of terminals. Therefore, we start by giving a short overview of their work, before discussing our own algorithm for parameter k.

2.1 Previous Approaches The starting observation is that cuts in planar graphs correspond to cycles in the dual of the graph [51]. Hence, it makes sense that almost the entire algorithms and structural descriptions of Klein and Marx and Colin de Verdière, as well as ours, work with the planar dual. If the number t of terminals is bounded, this quickly leads to the intuition that the planar dual of a multiway cut should be a planar graph with t faces, one for each terminal [21]. By dissolving vertices of degree 2 of this planar graph and applying Euler's formula, one obtains a planar graph S with $O(t)$ faces, vertices, and edges. One can then guess what S looks like by exhaustive enumeration and guess the vertices of the dual corresponding to the vertices of S. Then it remains to expand the edges of the graph back into shortest paths.

However, these paths are not just any shortest paths. Instead, they must wring themselves between the terminals in a particular way, such that they perform their job in separating the terminals. This is where topological arguments come in. Intuitively, the layout of these paths is described using the sequence of crossings with a Steiner tree on the terminals. The main thrust of the work of Klein and Marx and Colin de Verdière is to argue that in some optimal solution these crossing sequences are short, in the sense that their length depends only on t. One can then guess the crossing sequences of each of the paths in the optimum and find a shortest path following a particular crossing sequence in polynomial time using an algorithm by Frank and Schrijver's algorithm [31, Section 5].

The above leads to an $f(t)n^{\mathcal{O}(t)}$ time algorithm. To improve to a $f(t)n^{\mathcal{O}(\sqrt{t})}$ time algorithm, it suffices to observe that since S is planar, it has treewidth $\mathcal{O}(\sqrt{t})$, as follows from the planar separator theorem [43]. One can then replace the guessing of the vertices of the dual corresponding to vertices of S by a dynamic program on the tree decomposition.

2.2 Warm-up: An $n^{O(k)}$-time Algorithm We design our approach along the same lines, in that we show that the topology of the dual graph of the minimum multiway cut is constrained and then enumerate all the possible topologies. For a certain topology, we find the shortest solution respecting the topology. However, since we must deal with a huge number of terminals, possibly $\mathcal{O}(n)$ many of them, this is not straightforward. Indeed, we need stronger structural properties of the solution to limit the number of diverse topologies.

Structure: Global and Local From a global perspective, however, the structure of an optimal solution looks very similar. If we think of a terminal face as a single terminal, then as in the previous works, we see that (some part of) the dual of the solution must separate the different terminal faces. We call this part the *skeleton* of the dual of the solution[2]. The paths between its branching points are called *bones*. By dissolving dual vertices of degree 2 in the skeleton, we obtain the *shrunken skeleton*; its edges are called *shrunken bones*. The shrunken skeleton has k faces and $O(k)$ vertices and edges by Euler's formula. It follows that the shrunken skeleton of the optimum can be guessed by exhaustive enumeration and has treewidth $O(\sqrt{k})$, a great starting point. This is discussed in more detail in Section 5.

Now consider the local parts of the solution, namely the part of the dual of the solution inside each of the k faces of the skeleton. Chen and Wu [13] proved that when there is a single terminal face that is a simple cycle, then the dual of a minimum multiway cut is a minimum Steiner tree in the dual graph. To be more precise, this holds in the augmented dual graph. This graph splits the dual vertex of s of the face corresponding to the simple cycle into r parts, where r is the number of terminals of the face, such that all dual edges of each maximal subpath of the cycle without terminals belong to the same part of s. The solution is then a minimum Steiner tree on the augmented dual vertices. We generalize this argument to prove that inside each face of the skeleton, the solution is a forest of minimum Steiner trees on the augmented terminals, defined with respect to the terminal face inside the face of the skeleton. We call these *nerves*. All nerves attach to the boundary of the face of the skeleton. Crucially, the augmented terminals belonging to each of these nerves form an *interval* of the set of augmented terminals. Then, by applying the Dreyfus-Wagner algorithm [25], any nerve can be found in polynomial time [29, 6], if we know the interval. See Section 5.

Algorithm: Global and Local As a warm-up, we now discuss how to find an algorithm with running time $f(k)n^{\mathcal{O}(k)}$. First, guess the shrunken skeleton of the optimum solution by exhaustive enumeration. Then, for

[2]Our notion of skeleton should not be confused with the notion of skeleton as defined by Cohen-Addad *et al.* [18]. In particular, our notion of a (shrunken) skeleton is very reminiscent of the multicut dual of Colin de Verdière [22], provides a high-level view of the solution, and has no weight restrictions. The skeleton of [18] instead can be seen as 'orthogonal' to the solution; by controlling its weight, the skeleton can be portalized (as in Arora [1]) and the solution can be approximated by combining Steiner trees of a particular homotopy between portals. This is different from our use and definition of a skeleton.

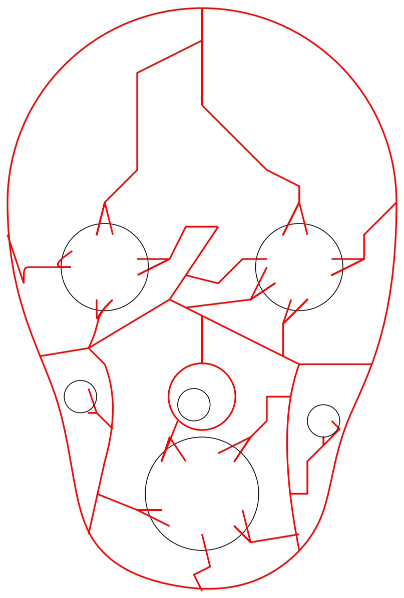

Figure 1: This figure illustrates the broken bones of the minimum multiway cut dual. In the figure, the terminal faces are drawn as black circles. In red, we depict the partial solution, which we term the broken bones. We fix these broken bones by splints to get C^+

each shrunken bone of this shrunken skeleton, we note that it needs to expand to separate two terminal faces, say $F_\alpha, F_\beta \in \mathcal{F}$, and some of the terminals on each of them. For both the terminal faces, we guess the intervals of augmented terminals I_α and I_β covered by nerves that attach to the corresponding bone. Since intervals are between two augmented terminals, there are $n^{\mathcal{O}(k)}$ intervals and we can guess the optimal intervals by exhaustive enumeration in the same time.

We now apply a dynamic program to find the bone. The crux here is to find the paths between the attachment points of different nerves. While each of those paths individually again has a short crossing sequence, as can be argued by adapting the approach of Klein and Marx [40] and Colin de Verdière [22], there are too many of these paths to efficiently guess all these crossing sequences. Instead, we argue that we can group these crossing sequences while keeping them small, and that we do not need to guess the crossing sequences for all groups.

We start by discussing the grouping. We group consecutive nerves towards the same terminal face. We then prove that the union of the paths between the nerves in such a group has a bounded crossing sequence (see Section 6). Then we can guess the optimal crossing sequence by exhaustive enumeration. Assuming we know the starting and ending nerve of the group, we can then employ a dynamic program to find all nerves in between, while ensuring that the paths between the attachment points of the nerves follows the guessed crossing sequence. This will take care of all (augmented) terminals between the intervals.

Next, we argue that we only need few groups. We say that two groups *alternate* if they are towards different terminal faces. If there are two alternating groups of nerves on the left part of the bone, and two alternating groups of nerves on the right part of the bone, then we can observe that these nerves effectively cordon off part of the plane. Inside this region, we can show that all terminals effectively lie on a cycle. This enables us to use the ideas of Chen and Wu [13] again to find an optimal solution.

To conclude our algorithm, we note that we need at most four groups. If there are indeed four, then there is a part between them for which we have a polynomial time algorithm. For the groups themselves, we know the paths between their attachment points has a crossing sequence of small size, and we can guess the optimal crossing sequences by exhaustive enumeration. For each group, we use a dynamic program that runs in polynomial time. By applying Frank and Schrijver's algorithm [31, Section 5], we can find shortest paths that follow a particular crossing sequence in polynomial time. See Section 7.3 for details.

The total running time of this algorithm is indeed $f(k)n^{O(k)}$. The latter term originates from the guessing of the terminals corresponding to the branching points of the skeleton, the guessing of the intervals, and the guessing of the nerves at the bookends of the groups. For each shrunken bone, we call this information a *broken bone* (we effectively guess its parts) and the solution that fixes it a *splint*.

2.3 Towards the Final Algorithm We now develop the ideas for the final algorithm. To avoid guessing the broken bones globally, we aim to apply the fact that the shrunken skeleton has bounded treewidth. Using a tree decomposition directly is cumbersome and not very intuitive. Instead, we use sphere-cut decomposition.

A sphere-cut decomposition is a branch decompositions, which can be thought of as a set of separators of the graph organized in a tree-like fashion. The tree structure enables dynamic programming in the usual manner. The crux is that the vertices of each separator induce a noose in the planar graph. A *noose* in this sense is a (closed) curve in the plane that intersects every face of the planar graph at most once and intersects the drawing only in the vertices of the separator. The tree-structure of the decomposition is organized in such a way that each separator has two child-separators and the symmetric difference of the corresponding two nooses is the noose of the parent-separator. A formal definition is in Section 3.1.

We now apply a dynamic program where we just maintain the broken bones for the shrunken bones of the faces intersected by a noose of the decomposition. In fact, this still is too much information, and instead we maintain only a relevant part of those broken bones, namely the first nerve that we encounter in either direction on the shrunken bones of each face that is intersected by the noose. We argue that this yields sufficient information to know all broken bones and obtain an optimum solution. See Section 8 for details.

It is known that there is a sphere-cut decomposition of a planar multigraph on k vertices where all nooses (and separators) have $O(\sqrt{k})$ vertices [24, 46, 50]. This will lead to the $n^{\mathcal{O}(\sqrt{k})}$ running time. This decomposition requires that the planar graph is connected, has no bridges, nor has self-loops. Unfortunately, our shrunken skeleton does not satisfy any of those demands out of the box. The issue of self-loops is quickly handled by not dissolving all vertices of degree 2 when obtaining a shrunken skeleton from a skeleton, but only those that do not lead to a self-loop.

Next, we ensure connectivity of the shrunken skeleton. To this end, we consider a connected component of the shrunken skeleton that is 'innermost' in the embedding of the shrunken skeleton. We can guess which of the terminal faces are enclosed by this connected component. However, for one of those terminal faces, some terminal might not be enclosed by the connected component. We cannot guess this terminal (as there are n choices), even though it is necessary to know the terminal to correctly guess the structure of the optimum solution. To circumvent this issue, we argue that we can pick a single terminal of this face as a representative of this 'exposed' terminal. Thus we completely avoid knowing which terminal is exposed. This enables a $2^{\mathcal{O}(k)}n^{O(1)}$ time subroutine to guess the components of the dual of an optimum solution and to reduce to the case where such duals are connected. We then argue that this implies that the shrunken skeleton is connected as well. See Section 4 for details.

Finally, we consider bridges of the shrunken skeleton. It seems hard to avoid them completely. Instead, we design another dynamic program (see Section 8.1). We use a bridge block tree of the skeleton to guide this dynamic program. Recall that a bridge block tree is a tree representation of the bridge blocks (bridgeless components) of a graph and the cut vertices between those bridge blocks, generalizing the more familiar block cut tree. To ensure this bridge block tree is suitable for the dynamic program, we need a modified version of a bridge block tree that organizes itself according to the embedding of the shrunken skeleton. To this end, we develop an embedding-aware bridge block tree (see Section 3.2), which may be of independent interest.

To conclude, our final algorithm is as follows. First, we perform a subroutine to ensure that the dual of any minimum multiway cut is connected. Then we guess the structure of the dual of such a solution, namely its shrunken skeleton, how the nerves of each bone are grouped, and what the crossing sequences are of each path between nerves of the same group and between the groups. We call this a *topology*[3] We guess the optimal topology by exhaustive enumeration. Consider its shrunken skeleton and define a dynamic program on its bridge blocks to combine partial solutions for each bridge block. For each bridge block, we show that it has a sphere-cut decomposition with nooses of bounded size, which enables to find a partial solution using a dynamic program. The guessing of the topology and the dynamic programs combined lead to an algorithm running in time $2^{\mathcal{O}(k^2 \log k)}n^{\mathcal{O}(\sqrt{k})}$.

[3]The word topology has many well known meanings, including a branch of mathematics. We use the term here in a cartographic sense, as an abstract map of the solution. This is in line with previous uses in the literature, *e.g.* [22].

3 Preliminaries

Graphs A *graph* is a pair $G = (V, E)$, such that $E \subseteq [V]^2$. The elements of V are called the vertices of the graph and the elements of E its edges. The number of vertices of a graph is its order denoted by $|G|$. A *subgraph* $G' = (V', E')$ of G, written as $G' \subseteq G$, is a graph such that $V' \subseteq V$ and $E' \subseteq E$. If $G' \subseteq G$ and $G' \neq G$, then G' is a *proper subgraph* of G. G' is an *induced subgraph* of G, if for all $x, y \in V'$, if $xy \in E$ then $xy \in E'$.

A *path* is a non-empty graph $P = (V, E)$ of the form $V = \{x_0, x_1, \ldots, x_k\}$ and $E = \{x_0x_1, \ldots, x_{k-1}x_k\}$, where all x_i are distinct. The number of edges in a path is its *length*. We use the notation $P[x_i, x_j]$ to denote the subpath of P between vertices x_i and x_j. For a tree N, we use $N[x, y]$ to denote the unique path in N between the vertices x and y.

A graph is *connected* if there is a path between any two vertices in a graph, and *disconnected* otherwise.

The *contraction* of an edge $e = (u, v)$ of G is the operation of identifying u and v, while removing any loops or parallel edges that arise. We use the notation G/e. This extends to sets $F \subseteq E(G)$ of edges, for which we can use the notation G/F.

Given a subset $T \subseteq V(G)$ (called terminals), a *Steiner tree* on T is a minimal connected subgraph H of G such that there is a path in H between any two terminals in T.

Connectivity A *cut vertex* of a connected graph is a vertex whose removal yields a disconnected graph. A *biconnected graph* is a graph without cut vertices. A *biconnected component* or *block* of a graph is a maximal subgraph that is biconnected.

A *bridge* of a connected graph is an edge whose removal yields a disconnected graph. A *bridgeless graph* is a graph that has no bridges. A *bridgeless component* or *bridge block* of a graph is a maximal subgraph that is bridgeless. A bridge block that consists of more than one edge is called a *nontrivial* bridge block; the other bridge blocks consist of just a single edge, which is a bridge in the graph. Bridge blocks are incident on each other at cut vertices of the graph.

Given two disjoint vertex subsets $X, Y \subseteq V(G)$, an (X, Y)*-cut* is a set of edges whose removal leaves no path between any vertex of X and any vertex of Y.

Given a subset $T \subseteq V(G)$, a *(edge) multiway cut* of T is a set of edges whose removal leaves no path between any pair of distinct vertices in T.

Topology and Planar Graphs A Jordan *arc* in the plane is an injective continuous map of $[0, 1]$ to $\mathbb{R}^2$. A Jordan *curve* in the plane is an injective continuous map of $\mathbb{S}^1$ to $\mathbb{R}^2$. If one of the points on this curve is special, we may also call this a closed arc on this point. Consider a set Z of Jordan (possibly closed) arcs in the plane. Let $P(Z)$ denote the union of the set of points of each arc of Z. Observe that $\mathbb{R}^2 \setminus P(Z)$ is an open set. A *region* of Z is a maximal subset X of $\mathbb{R}^2 \setminus P(Z)$ that is *arc-connected*; that is, there is a Jordan arc in X (meaning all its points belong to X) between any pair of points in X. If $\mathbb{R}^2 \setminus P(Z)$ has more than one region, then Z is *separating*. Let R be a region of a separating set Z. The *boundary* ∂X of a region X is the set of all points $p \in \mathbb{R}^2$ such that every open disk around p contains both a point of X and of $\mathbb{R}^2 \setminus X$. The *complement* of a region X is the union of $Y \cup \partial Y$ for each region of $\mathbb{R}^2 \setminus (X \cup \partial X)$. Note that the complement of a region is not necessarily a region itself, but possibly a union of regions (and their boundaries), particularly if X has holes.

We say that a region X *encloses* a set Y if $Y \subseteq X \cup \partial X$ and *strictly encloses* Y if $Y \subseteq X$.

For the definition of planar graphs, we follow Diestel [23]. A graph $G = (V, E)$ is *plane* if there V corresponds to a set of points (vertex points) in the plane and E corresponds to a set of arcs in the plane (edge arcs) between the points corresponding to its endpoints, such that the interior of each edge arc contains no vertex point and no point of any other edge arcs. We call the vertex points and edge arcs an *embedding* of G. If G admits an embedding, we call the graph a *planar graph*. A *face* of a plane graph is any region of the set of edge arcs. Exactly one face is *unbounded*, also called the *outer face*, whereas all other faces are *bounded*.

Two plane graphs are *equivalent* if they are isomorphic and the circular order of the edges around each vertex is the same in both embeddings. In particular, this means that boundaries of the faces of the embeddings have the same edge sets and there is a bijection between the faces of both embeddings.

Let $G(V, E, F)$ and $G^*(V^*, E^*, F^*)$ be two planar graphs. V, E and F (V^*, E^* and F^*) denote the set of vertices, edges, and faces of G (G^*). G^* is called a plane dual of G, if there exist bijections

$\begin{array}{ccc} f^* : V \to F^* & e^* : E \to E^* & v^* : F \to V^* \\ v \to f^*(v) & e \to e^*(e) & f \to v^*(f) \end{array}$ satisfying the following conditions:

(a.) $v^*(f) \in f, \forall f \in F$

(b.) e and e^* intersect in exactly one point, which lies in the interior of both e and e^*.

(c.) $v \in f^*(v), \forall v \in V$

We note the following basic properties, which follow from Diestel [23]. We will often use them without explicitly referring to this proposition.

Proposition 3.1. • *If G is connected, then the edges of G bounding each face form a closed walk (also known as a face walk).*

- *If G is bridgeless, then for any edge, the faces on both sides are distinct. In particular, the outer face is a simple cycle.*

Let O be a set of points in the plane, called *obstacles*. Consider two Jordan arcs a, b between the same pair of points, such that neither arc contains a point of O. Then these arcs are *homotopic* if there is a continuous deformation between a and b that does not cross a point of O. In the plane, this means that the region induced by the union of a and b does not contain any point of O.

3.1 Sphere-cut decompositions A main component of our algorithm is a dynamic program over a planar graph of bounded treewidth. However, using a normal tree decomposition is rather cumbersome in this case, and it turns out to be much easier to instead use a sphere-cut decomposition, a branch decomposition especially suited for planar graphs. We define all necessary notions and state the relevant theorem below.

A *branch decomposition* of a graph $G = (V, E)$ is a pair (R, η) of a ternary tree R and a bijection η between the leaves of R and the edges in E. For an edge e of R, we define the *middle set* $\mathbf{mid}(e)$ to be the set of vertices in V for which an incident edge is mapped by η to a leaf in the one component of $R-e$ and an incident edge is mapped by η to a leaf in the other component. The *width* of the branch decomposition is defined as the maximum size of the middle set of any edge of R. The *branchwidth* of G is then the minimum width of any branch decomposition of G.

Let G be a (planar) graph with a fixed embedding on the sphere. Then a *noose* $\vec{\gamma}$ (with respect to G) is a closed, directed curve in the sphere that meets the embedding of G only in its vertices and that traverses each face exactly once. The *length* of the noose is equal to the number of vertices of G traversed by it. If we enumerate the vertices of the noose, we implicitly assume that this enumeration follows the order of appearance on the noose, that is, following its direction. Note that a noose cuts the sphere into two regions, each homeomorphic to an open disk. The region bounded by and to the right when following the noose with its direction is denoted by $\mathbf{enc}(\vec{\gamma})$ and the other by $\mathbf{exc}(\vec{\gamma})$.

A sphere-cut decomposition of a graph G with a fixed embedding on the sphere is a triple (R, η, δ) consisting of a branch decomposition (R, η) and a mapping δ from an ordered pair of adjacent vertices x, y of R to nooses (with respect to G) on the sphere, such that

- $\delta(x, y)$ is the same noose as $\delta(y, x)$ but with the direction reversed. Note that then it holds $\mathbf{enc}(\delta(x, y)) = \mathbf{exc}(\delta(y, x))$;
- $\delta(x, y)$ meets the embedding of G exactly in the vertices of the middle set $\mathbf{mid}(x, y)$; moreover, $\mathbf{enc}(x, y)$ contains all the embeddings of all edges of the one component of $R - xy$ and $\mathbf{exc}(x, y)$ contains the embeddings of all other edges.

As noted by Dorn *et al.* [24] and Pilipczuk *et al.* [50], we may assume that a sphere-cut decomposition is *faithful*. That is, for every internal vertex x of R with adjacent vertices y_1, y_2, y_3, we may assume that $\mathbf{enc}(x, y_1)$ is equal to the disjoint union of $\mathbf{enc}(y_2, x)$, $\mathbf{enc}(y_3, x)$, and $(\delta(y_2, x) \cap \delta(y_3, x)) \setminus \delta(x, y_1)$. We also note that $\delta(y_2, x) \cap \delta(y_3, x) \cap \delta(x, y_1)$ consists of two points, each of which may (or may not) coincide with a vertex of G.

As described by Dorn *et al.* [24], we can root any sphere-cut decomposition (R, η, δ) by subdividing an arbitrary edge e of R. Let u be the newly created vertex and e', e'' be the newly created edges. Set $\mathbf{mid}(e') = \mathbf{mid}(e'') = \mathbf{mid}(e)$. Add a new vertex r, connect it to u, and set $\mathbf{mid}(ru) = \emptyset$. We then direct the tree R towards the root r. In the remainder, we assume our sphere-cut decomposition are rooted.

The following result was observed by Dorn *et al.* [24] and follows from [33, 52] (see also Marx and Pilipczuk [46] and Pilipczuk *et al.* [50]).

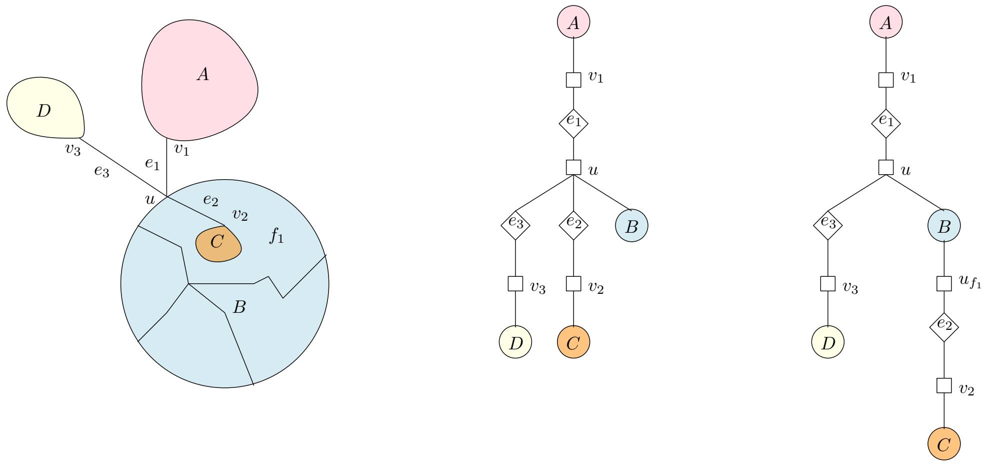

Figure 2: The leftmost image is that of a graph with bridge blocks A, B, C and D. The bridges are denoted by e_1, e_2 and e_3, whereas the cut vertices are u, v_1, v_2, and v_3. The figure in the middle shows a bridge-block tree corresponding to the graph. The rightmost figure shows the embedding aware bridge block tree.

Theorem 3.1. *Every n-vertex connected, bridgeless multigraph without self-loops but with a fixed embedding on the sphere has a faithful sphere-cut decomposition of width $\sqrt{4.5n}$. Moreover, such a sphere-cut decomposition can be found in $O(n^3)$ time.*

3.2 Embedding-Aware Bridge Block Trees An important aspect of our algorithm will be to deal with bridge blocks of a planar graph, as a sphere-cut decomposition can not. We define the following notion, which lends itself in a nice way to the dynamic programming algorithm we develop towards the end of the paper.

We can define a *bridge block tree* of a graph H as follows. Consider the graph F that has a node for every bridge block, called the BB-nodes of F, and for every cut vertex, called the C-nodes of F. There is an edge between a BB-node corresponding to a bridge block B and a C-node that corresponds to a cut vertex v if $v \in V(B)$. It is immediate that this graph is a tree if H is connected and a forest otherwise. For simplicity, we call two bridge blocks *neighboring* if they share a cut vertex. We can observe that any nontrivial bridge block neighbors only trivial bridge blocks.

Theorem 3.2. (Tarjan [53]) *A bridge block tree of a graph can be computed in linear time.*

If H is plane and connected, then we use an extension of this definition. An *embedding-aware bridge block tree* (or *eabb* tree) $L = L(H)$ of H is formed from the bridge block tree F as follows. Root F at a BB-node $\ell = \ell(F)$ that corresponds to a bridge block that has an edge bordering the outer face of H. We now perform two operations on the BB-nodes.

First, for any nontrivial bridge block B whose corresponding BB-node b has a C-node parent p in F corresponding to a cut vertex v, note that all other BB-node children c' of p correspond to trivial bridge blocks. For each bounded face f of B, create a new C-node child c_1^f of b corresponding to v and f, and for any child $c' \neq b$ of p corresponding to a bridge edge that is contained in f, make the subtree of F rooted at c' a child of c_1^f. We only add c_1^f if there are any such children c'.

Second, for any nontrivial bridge block B whose corresponding BB-node p has a C-node child c in F corresponding to a cut vertex v, note that all BB-node children c' of c correspond to trivial bridge blocks. For each bounded face f of B, create a new C-node child $c_2^{v,f}$ of p corresponding to v and f, and for any child c' of c corresponding to a bridge edge that is contained in f, make the subtree of F rooted at c' a child of $c_2^{v,f}$. We only add $c_2^{v,f}$ if there are any such children c'.

Perform these two operations on all nontrivial bridge blocks. Observe that the operations essentially apply to the children of C-nodes corresponding to cut vertices contained in a nontrivial bridge block. As nontrivial bridge blocks are not neighboring, the sets of cut vertices contained in nontrivial bridge blocks are pairwise disjoint. Hence, these operations do not interfere with each other and can be performed independently.

Then, finally, order the children of a C-node p in the tree according to the order in which their edges appear around the corresponding cut vertex. The resulting tree is $L(H)$.

LEMMA 3.1. (♣) *An embedding-aware bridge block tree of a plane, connected graph H can be computed in polynomial time.*

We now make several observations about $L(H)$. Let B and B' be distinct bridge blocks of a plane graph H. Since H is plane (and ignoring possible intersections of the embedding on the cut vertex), B is enclosed by a bounded face of B', or vice versa, or B and B' are both in each other's outer face. We now define a strict partial order $\prec_P$ on the bridge blocks of H, where $B \prec_P B'$ if B is embedded in a bounded face of B'.

LEMMA 3.2. (♣) *Let B and B' be two bridge blocks of a connected plane graph H. If $B \prec_P B'$, then the node b corresponding to B is a descendant of the node b' corresponding to B' in $L(H)$.*

LEMMA 3.3. (♣) *Let H be a connected plane graph. Let c and c' be two C-nodes in $L(H)$ corresponding to the same cut vertex v of H. Then on the path between c and c' in $L(H)$, all other C-nodes correspond to v.*

4 Basic Properties and Connectivity of the Planar Multiway Cut Dual

Let $G = (V, E)$ be a connected simple undirected planar graph on n vertices and m edges with a fixed embedding. Let $T \subseteq V$ be the set of terminals. The set of faces covering all the terminals in T is denoted by $\mathcal{F} = \{F_\alpha : 1 \leq \alpha \leq k\}$. We call these the *terminal faces* of G. The edges of E are weighted. By removing edges of weight 0 or less and then scaling the weights of the remaining edges, we can obtain an equivalent instance with weights specified by the function $\omega : E \to [1, \ldots, W]$ for some integer W. Note that during this transformation, possibly, the set of terminal faces changes, but there will still be at most k of them. Moreover, the graph might become disconnected, but we can solve the instance associated with each connected component independently. Hence, by abuse of notation, we may assume that our instance is still defined by G, T, k, $\mathcal{F}$, and ω as defined previously.

In the remainder, we use edge weight ∞ to indicate undeletable edges. Instead of ∞, one could use $mW + 1$, but using ∞ simplifies later notation. Note that the initial instance has no undeletable edges, and thus has a finite weight solution. In future transformations and reductions, we shall always maintain the property that a finite-weight solution exists. By abuse of notation, we still use $\omega : E \to [1, \ldots, W] \cup \{\infty\}$ to denote the weights.

Arbitrarily assign each terminal $t \in T$ to a face of $\mathcal{F}$ that has t on its boundary. For each face $F_\alpha \in \mathcal{F}$, let $T_\alpha \subseteq T$ be the set of terminals on the boundary of F_α that are assigned to F_α. Let $p_\alpha = |T_\alpha|$. We may assume that $p_\alpha > 0$ for each terminal face, or we could reduce the set of terminal faces. Observe that the sets T_α form a partition of T. Note that $\mathcal{F}$ can be partitioned into $\mathcal{F}_1 = \{F_\alpha \mid p_\alpha = 1\}$ (called the *singular faces*) and $\mathcal{F}_2 = \mathcal{F} \setminus \mathcal{F}_1$ (called the *plural faces*); possibly, one or both of these sets is empty.

For a terminal face F_α, we order the terminals in T_α as follows. Note that G is connected and thus the boundary of F_α forms a closed walk. Pick an arbitrary starting vertex on this closed walk. Now traverse the walk in clockwise direction and add a terminal to the ordering at the first moment it is encountered. Index the terminals in T_α as ${}^\alpha t_1, \ldots, {}^\alpha t_{p_\alpha}$ according to this ordering.

We prove the following transformation is possible, which is reminiscent of Chen and Wu [13, Lemma 8].

LEMMA 4.1. (♣) *In polynomial time, one can transform the instance into an equivalent instance where G is bridgeless, the faces of $\mathcal{F}$ are vertex disjoint, and all vertices of each face in $\mathcal{F}$ are terminals.*

We call an instance *transformed* if it has the properties as set forth in Lemma 4.1. We note that the transformation might make G no longer simple, but have parallel edges or self-loops. In the remainder, we assume that the instance is transformed.

Let G^* be the dual of G. By definition, G^* has an embedding in the plane such that each vertex of G^* is embedded in the corresponding face and each dual edge crosses the corresponding primal edge exactly once and no other edges. For practical purposes, any time we consider a set C^* of dual edges, we also denote by C^* the subgraph of the dual induced by the edges in C^*. Then C^* is again a planar graph with an embedding where each edge of C^* is embedded as it is in the embedding of G^*. We denote by C the set of edges in G corresponding to the dual edges in C^*.

The following was observed by Dahlhaus *et al.* [21], based on the original observation of Reif [51].

PROPOSITION 4.1. *Let C be a (minimum) multiway cut of (G,T,ω) and let C^* be the set of corresponding dual edges. Then each face of C^* encloses at most (exactly) one terminal.*

We will often use this fact without explicitly referring to it.

We now describe the algorithm. Let $\mathcal{A}$ be any algorithm for PLANAR MULTIWAY CUT that always outputs a feasible solution, but is only guaranteed to find an optimum solution if for all optimum solutions C to the instance it holds that C^* is connected. We show in later sections that we can find such an algorithm $\mathcal{A}$ with the required running time bounds. Using it as a black box for now, we can give a recursive algorithm for PLANAR MULTIWAY CUT.

Let (G,T,ω) be a transformed instance of PLANAR MULTIWAY CUT with $\mathcal{F}$ the set of terminal faces. Recall that $\mathcal{F}$ can be partitioned into the set $\mathcal{F}_1$ of singular faces and the set $\mathcal{F}_2$ of plural faces; possibly, one or both of these sets is empty.

In the first phase of the algorithm, consider two new types of sub-instances. For the first type, if $\mathcal{F}_2 \neq \emptyset$, consider each $F_\beta \in \mathcal{F}_2$ and each set $\tilde{T}$ that is the union of the sets $\{T_\alpha \mid F_\alpha \in \tilde{\mathcal{F}}\}$ for a subset $\tilde{\mathcal{F}}$ of $\mathcal{F} \setminus \{F_\beta\}$, and solve recursively on the transformed version of the sub-instance (G,T'',ω'') where $T'' = \tilde{T} \cup \{{}^\beta t_1\}$ and ω'' is equal to ω, except it is set to ∞ for all edges of F_β. Effectively, this uses ${}^\beta t_1$ as a representative of the whole set T_β. For the second type, consider each set T'' that is the union of the sets $\{T_\alpha \mid F_\alpha \in \tilde{\mathcal{F}}\}$ for a strict subset $\tilde{\mathcal{F}}$ of $\mathcal{F}$, and solve recursively on the transformed version of the sub-instance (G,T'',ω'') where $\omega'' = \omega$.

Let B be the solution that is recursively found for any such a sub-instance (G,T'',ω'') and let B^* denote the set of corresponding dual edges. In the second phase of the algorithm, let T' be the set of terminals in the outer face of B^*. If there is a terminal face F_α for which $\emptyset \subset T' \cap T_\alpha \subset T_\alpha$, then consider the sub-instance $(G,(T' \setminus T_\alpha) \cup \{{}^\alpha t_1\},\omega')$, where ω' is obtained from ω'' by setting the weight of each edge of F_α to ∞. Otherwise, consider the sub-instance (G,T',ω). Let Z_B be the solution that is recursively found for such a sub-instance. Among all such combinations $B \cup Z_B$ that are a feasible solution and $\mathcal{A}(G,T,\omega)$, return the feasible solution of minimum weight.

This finishes the description of the algorithm. By turning the above recursive algorithm into a dynamic programming algorithm, we prove the following result.

LEMMA 4.2. (♣) *Let $\mathcal{A}$ be any algorithm for* PLANAR MULTIWAY CUT *that always outputs a feasible solution, but is only guaranteed to find an optimum solution if for all optimum solutions C to the instance it holds that C^* is connected. Let $T(n,k)$ be a monotone function that describes the running time of $\mathcal{A}$ on instances with n vertices and k terminal faces. Then* PLANAR MULTIWAY CUT *can be solved in $O(2^{O(k)}T(n,k)\text{poly}(n))$ time.*

Our approach for this lemma is inspired by Klein and Marx [40, Lemma 3.1], who also considered biconnected components but individually. Then in their sub-instances, for a subset X of terminals, they not only need to pairwise separate the terminals of X, but also need to separate X from $T \setminus X$. We use the planarity of the solution to effectively argue that the latter condition is not necessary. Hence, our sub-instances are 'normal' instances of PLANAR MULTIWAY CUT, without further constraints.

Following Lemma 4.2, we need to develop the required algorithm $\mathcal{A}$. Hence, in the remainder, we assume that the dual of any optimum solution to our instance of PLANAR MULTIWAY CUT is connected.

5 Augmented Planar Multiway Cut Dual: Skeleton and Nerves

An important challenge for our algorithm is to find the part of an optimum solution that cuts the many terminal pairs incident on a single terminal face (of course, in concert with cutting terminal pairs on other faces). In the case that $k = 1$, Chen and Wu [13] proposed the notion of an augmented dual in order to reduce to an instance of PLANAR STEINER TREE wherein all the terminals lie on a single face. This, in turn, can be solved in polynomial time [29, 6]. Such a simple reduction does not work in our case, but we do borrow and extend the notion of an augmented dual.

DEFINITION 5.1. *The augmented dual graph G^+ of G with respect to $\mathcal{F}$ is constructed as follows. Starting with the planar dual of G, for each face $F_\alpha \in \mathcal{F}$, replace the corresponding dual vertex v_α by a set of vertices $T^+_\alpha = \{{}^\alpha t^+_1, {}^\alpha t^+_2, \ldots, {}^\alpha t^+_{p_\alpha}\}$ (called the augmented terminals, each being an end point of an edge incident on v_α. For $1 \leq j \leq p_\alpha$, each ${}^\alpha t^+_j$ and its incident edge represents the edge $({}^\alpha t_{j-1}, {}^\alpha t_j)$ on the boundary of F_α. The weight function on the edges of G^+ is denoted by ω^+; all the edges in the augmented dual graph have the same weight as their corresponding edge in G under ω.*

Figure 3 illustrates the structure of C^+.

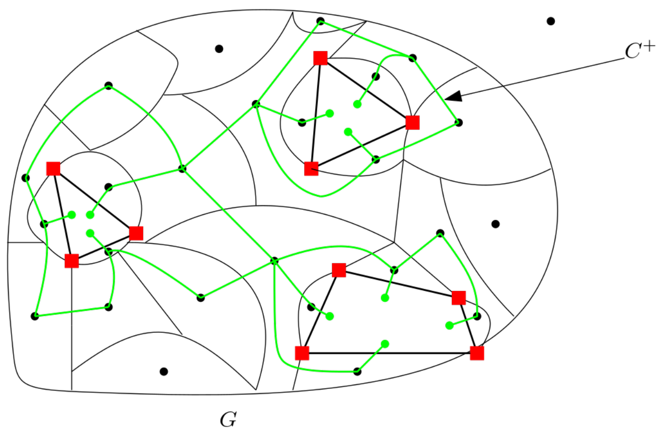

Figure 3: The back lines are edges of the graph G. The red blocks represent terminals. By green dots we show the augmented vertices, while the black dots represent the vertices for each non-terminal face in the augmented dual. The green curves correspond to the graph C^+.

Observe that there is still a one-to-one correspondence between edges of G and (augmented dual) edges of G^+, as there is between edges of G and (dual) edges of G^*. We also note that an alternative construction of G^+ would be to subdivide each dual edge of G^* incident on the dual vertex v_α corresponding to a terminal face $F_\alpha \in \mathcal{F}$ and subsequently removing v_α. For practical purposes, any time we consider a set C^+ of augmented dual edges, we also denote by C^+ the subgraph of the dual induced by the edges in C^+.

Lemma 5.1. (♣) *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then C^+ is a connected planar graph with k faces, one per terminal face in $\mathcal{F}$.*

5.1 Skeleton Considering Lemma 5.1, we note in particular that C^+ has k faces. However, each of these k faces can be highly complex. By 'zooming out', we can show that C^+ in fact has a very nice global structure with k faces that are much easier to describe. To this end, we prove the following.

We define the action of *dissolving* a vertex v of degree 2 with *distinct* neighbors u and w, as removing v from the graph and adding an edge from u to w. If we drop the constraint that u, w are distinct, we speak of *strongly dissolving*. We explicitly mention that while dissolving a vertex we retain any parallel edges that may arise. Moreover, no self-loops may arise while dissolving a vertex, while this is possible when stricly dissolving a vertex.

Definition 5.2. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the set of augmented dual edges that remain after exhaustively removing all the edges incident on a vertex of degree 1 of C^+ is called the skeleton S^+ of C^+. The set of augmented dual edges that remain after dissolving all vertices of degree 2 from S^+ is called the shrunken skeleton of C^+.*

Lemma 5.2. (♣) *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the shrunken skeleton of C^+ is a connected planar multi-graph without self-loops but with k faces, such that each of its faces strictly encloses exactly one terminal face of $\mathcal{F}$.*

It is crucial to note that while the skeleton is connected, it is not necessarily bridgeless. This has important algorithmic consequences discussed later.

Definition 5.3. *Let $F_\alpha \in \mathcal{F}$ and let f_α be the corresponding face of the skeleton of C^+. We call the set of augmented dual edges of f_α the spine of F_α with respect to C^+.*

Note that a spine is not necessarily isomorphic to a cycle, but only to a closed walk, because the skeleton is not necessarily bridgeless.

Definition 5.4. *Let $F_\alpha \in \mathcal{F}$. An enclosing cut is a cut that separates T_α from $T \setminus T_\alpha$ and also separates each terminal in T_α from every other terminal.*

Remark 5.1. *Chen and Wu use the term island cut to denote the $(T_\alpha, T \setminus T_\alpha)$-cut for each $F_\alpha \in \mathcal{F}$. Since they find a 2-approximate solution instead of an exact one, it suffices to separate the graphs into islands and then find a multiway cut for each island.*

Corollary 5.1. (♣) *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Consider a face f_α of the skeleton of C^+ and let $F_\alpha \in \mathcal{F}$ be the single terminal face strictly enclosed by it. Then the set of edges of G corresponding to the spine of F_α is a $(T_\alpha, T \setminus T_\alpha)$-cut and the set of edges of G corresponding to the augmented dual edges of C^+ enclosed by f_α is an enclosing cut.*

The following lemma is proved as in Dahlhaus *et al.* [21, Lemma 2.2] with some modifications.

Lemma 5.3. (♣) *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then the shrunken skeleton of C^+ has at most $4k$ vertices and at most $7k$ edges.*

5.2 Single Face Per Corollary 5.1, for each face $F_\alpha \in \mathcal{F}$ and corresponding face f_α of the skeleton, the spine of F_α is an island cut and the set of augmented dual edges of C^+ enclosed by f_α is an enclosing cut. We show that the enclosing cut has a very specific structure.

Definition 5.5. *Let $F_\alpha \in \mathcal{F}$. An interval of augmented terminals of T_α^+ is the set $\{{}^\alpha t_i^+ \mid i \in I\}$, where $I \subseteq \{1, \ldots, p_\alpha\}$ is any set such that either $|I| = p_\alpha$ or for each $i \in \{1, \ldots, p_\alpha\}$ except exactly one, it holds that there is a $j \in I$ where $j = i + 1 \mod p_\alpha$. We say that the interval is* between i and j *$(1 \leq i, j \leq p_\alpha)$ if $i \leq j$ and $I = \{i, \ldots, j\}$, or $i > j$ and $I = \{i, \ldots, p_\alpha, 1, \ldots, j\}$.*

It is important to note that we treat intervals as ordered sets of terminals. In particular, the intervals $\{{}^\alpha t_1^+, \ldots, {}^\alpha t_{p_\alpha}^+\}$ and $\{{}^\alpha t_2^+, \ldots, {}^\alpha t_{p_\alpha}^+, {}^\alpha t_1^+\}$ are distinct.

This definition invites several additional definitions that will prove useful in later parts of the paper. Two intervals I, I' of the same terminal face F_α are *consecutive* if they are disjoint and there is a third (possibly empty) interval I'', disjoint from I and I', such that $I \cup I' \cup I'' = T_\alpha^+$.

Given two consecutive intervals $I = \{{}^\alpha t_1^+, \ldots, {}^\alpha t_a^+\}$ and $I' = \{{}^\alpha t_{a+1}^+, \ldots, {}^\alpha t_b^+\}$ of the same terminal face F_α, a terminal $t \in T_\alpha$ is *inbetween* I and I' if $t = {}^\alpha t_a$. Note that there is exactly one terminal inbetween two consecutive intervals, unless $I \cup I' = T_\alpha^+$, in which case there are exactly two. A terminal $t \in T_\alpha$ is *between* $I = \{{}^\alpha t_1^+, \ldots, {}^\alpha t_a^+\}$ if $t \in \{t_1, \ldots, t_{a-1}\}$.

A *subinterval* of an interval $I = \{{}^\alpha t_1^+, \ldots, {}^\alpha t_a^+\}$ is any subset $\{{}^\alpha t_b^+, \ldots, {}^\alpha t_c^+\}$ of I such that $1 \leq b \leq c \leq a$ or is the empty interval. A *prefix* of an interval $I = \{{}^\alpha t_1^+, \ldots, {}^\alpha t_a^+\}$ is any subinterval of I containing ${}^\alpha t_1^+$ or is the empty interval. A *suffix* of an interval $I = \{{}^\alpha t_1^+, \ldots, {}^\alpha t_a^+\}$ is any subinterval of I containing ${}^\alpha t_a^+$ or is the empty interval.

(Note that we have started the intervals at ${}^\alpha t_1^+$ only for simplicity and to avoid modulo-calculus, but the definitions extend in the obvious manner.)

Enclosing Cut on your Nerves Using the notion of intervals, we can describe the enclosing cut. To this end, we rely on the following result.

Theorem 5.1. (Chen and Wu [13, Lemma 3]) *Let (G, T, ω) be an instance of* Edge Multiway Cut *where G is a biconnected plane graph such that all terminals of T lie on the outer face. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in the augmented dual on the augmented terminals.*

In this result, the biconnectivity of G ensures that the augmented dual G^+ is connected. For the conclusion of the above theorem to hold, it suffices that G^+ is connected.

Corollary 5.2. *Let (G, T, ω) be an instance of* Edge Multiway Cut *where G is a plane graph such that all terminals of T lie on the outer face and G^+ is connected. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in G^+ on the augmented terminals.*

We can also obtain the following corollary.

Corollary 5.3. *Let (G, T, ω) be an instance of* Edge Multiway Cut *where G is a plane graph such that the outer face is bounded by a cycle and all terminals of T lie on this cycle. Then a set of edges is a minimal multiway cut for this instance if and only if the corresponding augmented dual edges form a minimal Steiner tree in the augmented dual on the augmented terminals.*

It is important to note that the mentioned cycle might be two parallel arcs or a self-loop.

These results suggest that Steiner trees in the augmented dual are important for a multiway cut. This is indeed the case, although the situation is substantially more involved when $|\mathcal{F}| > 1$.

Lemma 5.4. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Consider a face f_α of the skeleton S^+ of C^+ and let $F_\alpha \in \mathcal{F}$ be the single terminal face strictly enclosed by it. Let Y^+ be the spine of F_α and let X^+ be the set of remaining augmented dual edges enclosed by f_α. Then X^+ is a forest of minimum Steiner trees with the terminals of each minimum Steiner tree lying in some interval of T^+_α, such that the union of all intervals is T^+_α, and each minimum Steiner tree having a single vertex on Y^+ that is incident on a single edge of that tree.*

Proof. By Corollary 5.1, $G \setminus Y$ does not contain any path connecting the terminals of T_α to terminals of $T \setminus T_\alpha$. Therefore, the region of G^+ bounded by Y^+ and the boundary of F_α does not contain any terminal of $T \setminus T_\alpha$. The edges in X thus correspond to a multiway cut of T_α in $G \setminus Y$. Moreover, this multiway cut is minimal in $G \setminus Y$: any edge that could be removed from X while it remains a multiway cut in $G \setminus Y$ can also be removed from C while it remains a multiway cut in G, which would contradict the minimality of C.

Now consider $G \setminus Y$. Note that the spine does not contain any augmented terminals, as those have degree 1 in C^+ and thus are not part of the skeleton by definition. Hence, Y does not contain any edges of F_α and F_α persists in $G \setminus Y$. Let Q be the component of $G \setminus Y$ that contains F_α. Since the instance is assumed to be transformed by Lemma 4.1, F_α is a cycle in Q. Moreover, X is a minimal multiway cut for T_α in Q, as argued above. Hence, by Corollary 5.3, X^+ is a minimal Steiner tree in the augmented dual of Q on the terminal set T^+_α. It is in fact of minimum weight; otherwise, we could replace X^+ by a minimum-weight Steiner tree on T^+_α and obtain a minimum multiway cut for G of smaller weight using Corollary 5.3 and the fact that Y forms an island cut.

Now we consider what X^+ looks like in the augmented dual of G. We note that all edges of the augmented dual of $G \setminus Y$ also appear in the augmented dual of G. Hence, we can think of X^+ as a set of edges in G. Let X be the corresponding set of edges of G. Deleting the set of edges in Y from G is equivalent to contracting the edges of Y^+ in the augmented dual. Let y^+ be the vertex formed by contracting all the edges in Y^+. Since C^+ is connected by Lemma 5.1, it follows that at least one edge of X^+ is incident on y^+.

Deleting y^+ from X^+ creates $\deg(y^+)$ many connected components of X^+. We treat each of these connected components as subsets of edges, which includes a single edge incident on y^+. We claim that each of these components is a minimum Steiner tree in G^+ containing some interval of T^+_α in its vertex set. Clearly, each of the components is a Steiner tree on y^+ and some subset of T^+_α. We first show that the augmented terminals contained in each component form an interval of T^+_α. Due to the planarity of G^+, no two tree edges cross each other. Therefore, if the terminals of any two connected components were to cross each other on T^+_α, then they must intersect in some vertex. However, since each connected component is maximally connected, we reach a contradiction. Hence, the terminals form an interval.

Due to the minimality of X^+ in G^+/Y^+, with the terminal set $T^+_\alpha \cup \{y^+\}$, we already know that each of the components is a minimum Steiner tree on G^+/Y^+ containing its respective interval of T^+_α. Suppose that one of these trees is not a minimum Steiner tree in G^+. Then, there must exist in G^+ a Steiner tree W^+ of lower weight on the same terminal set, such that it either contains some edges of Y^+ or crosses Y^+. In the latter case, there is a subpath of this Steiner tree that intersects Y^+ in at least two vertices, leading to the creation of a cycle enclosing no terminal of T_α. That would contradict the minimality of X^+. In the former scenario, suppose that we replace the concerned component of X^+ with W^+. We claim that all the previously enclosed terminals by the component of X^+, remain enclosed after the replacement. If for some terminal $t \in T_\alpha$ in the interval, W^+ does not contain a subpath enclosing it, then the path between the augmented vertices flanking it contains edges of Y^+. This path in W^+ crosses all the paths from t to $T_\alpha \setminus \{t\}$, and the segment of Y^+ between the points

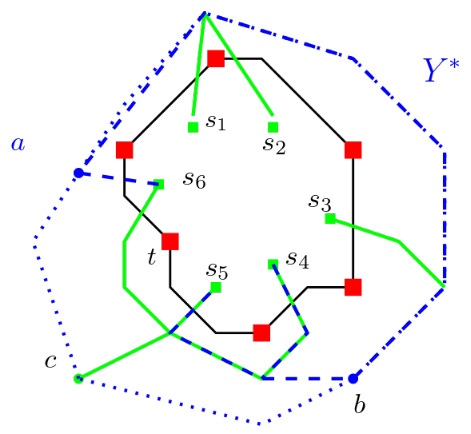

Figure 4: The black cycle is the boundary of the face F_α. The terminals of T_α lying on the boundary are drawn as red boxes. The spine Y^+ of the minimum enclosing cut dual is drawn in blue dotted lines. The green trees are the minimum Steiner trees of $X^+ \setminus \{y^+\}$ in G^+/Y^+. The blue dashed lines represent the minimum Steiner tree in G^+ on the terminal set $\{s_4, s_5, s_6\}$, in which the path between s_5 and s_6 contains the subpath of Y^+ between the vertices a and b.

of intersection crosses all the paths from t to terminals in $T \setminus T_\alpha$. Therefore, the region bounded by W^+ and Y^+ that contains t cannot contain any other terminal of T_α, and t remains enclosed. The replacement would, thus, yield an enclosing cut of T_α, strictly smaller than $X^+ \cup Y^+$. This, again, contradicts the minimality of X^+. Figure 4 illustrates the argument.

Finally, suppose that some minimum Steiner tree intersects the spine in more than one vertex. Then $X^+ \cup Y^+$ would contain a cycle that encloses no terminals, which contradicts Proposition 4.1.

Therefore, X^+ is a forest of minimum Steiner trees on intervals of T_α^+ with the terminals of each minimum Steiner tree lying in some interval of T_α^+, such that the union of all intervals is T_α^+, and each minimum Steiner tree having a single vertex on Y^+ that is incident on a single edge of that tree. $\square$

Remark 5.2. *For the above lemma and proof, it is important to observe that we can replace X^+ by any other forest of minimum Steiner trees on the same intervals and same vertices on Y^+. Indeed, such a forest would form a minimum Steiner tree in G^+/Y^+. Then we can replace X^+ in the argument of the lemma by the new forest.*

This remark inspires the following definition.

Definition 5.6. *Each minimum Steiner tree in the forest X^+ is called a nerve. Each nerve intersects the spine in exactly one vertex, which is incident on a single edge of the nerve. The unique vertex of the spine at which a nerve is rooted is called its attachment point.*

A nerve for a plural face $F_\alpha \in \mathcal{F}$ can be specified by a triple (v, i, j) with $1 \le i, j \le p$ such that the nerve is a minimum Steiner tree in G^+ with terminal set being the attachment point v and the interval of augmented terminals between i and j, and a single edge incident on v.

For a nerve (v, i, j), we may speak of the interval between i and j as simply the interval of the nerve. In this way, we can extend the definitions of consecutive, between, and in between, originally defined for intervals, to nerves in the straightforward manner.

Such a Steiner tree might not necessarily exist, but we have argued in Lemma 5.4 that our solution is built only from such Steiner trees.

We now argue that nerves can be computed in polynomial time.

Lemma 5.5. (♣) *Given vertices v and an interval I of a terminal face F_α between i and j, we can compute in polynomial time a nerve (v, i, j), if it exists.*

Now note that nerves are not necessarily unique for a triple (v, i, j). From now on, we associate a unique minimum Steiner tree with each triple (if it exists), namely the one computed by Lemma 5.5. We call this the *unique nerve* (v, i, j). In the remainder, whenever we talk about a nerve, we mean the unique nerve on the same interval and attachment point. Following Remark 5.2, we may assume that minimum-weight multiway cuts are built from a skeleton and (unique) nerves.

Definition 5.7. *Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding augmented dual edges. Then C^+ (and by extension, C) is called nerved if the edges of C^+ that are not part of the skeleton each belong to a nerve.*

Observe that following Remark 5.2 and the discussion above, we can assume that any minimum-weight multiway cut is nerved.

6 Bones and Homotopy

We now wish to describe the structure of the paths between the attachment points of nerves and to branching points of the skeleton. To this end, we start with the following definition.

Definition 6.1. *Each edge of the shrunken skeleton is called a shrunken bone and corresponds to a path of the skeleton, called a bone. Any vertex of the shrunken skeleton is called a branching point.*

By Lemma 5.2, each (shrunken) bone is incident on one or two faces of the (shrunken) skeleton, and thus separates at most two faces of $\mathcal{F}$, called the *separated terminal faces* of the (shrunken) bone. Note that both sides of a (shrunken) bone might be incident on the same face, as the shrunken skeleton can contain bridges.

6.1 Orienting Nerves To further discussions in the remainder of the paper, it helps to define an orientation of the nerves, which specifies where a nerve goes with respect to a bone. Let C be any minimum multiway cut of (G, T, ω) and C^+ the set of corresponding edges in the augmented dual. For any augmented dual vertex x, there is a small ball centered on x that only contains x and points of its incident augmented dual edges. Let e and e' be two augmented dual edges incident on x. Then this small ball is split into two parts by the union of e and e'. An augmented dual edge is *west* of x with respect to e and e' if it is contained in the part of the ball clockwise from e' to e and *east* otherwise.

Now consider two augmented dual edges e and e' of the skeleton of C^+ that appear in counterclockwise order on the boundary of a face f_α of the skeleton of C^+ (so e appears after e'). Let x be their shared augmented dual vertex. If the boundary of f_α is a walk, then we only consider the first appearance of an augmented dual edge to define the ordering. We then say that e is *north* of e'. Let $F_\alpha \in \mathcal{F}$ be the terminal face corresponding to f_α. We then say that a nerve (x, h, i) for F_α with $1 \leq h, i \leq p_\alpha$ *extends west* of x if it holds that the augmented dual edge incident on x that belongs to the nerve is west of x. If the augmented dual edge incident on x that belongs to the nerve is east of x, then the nerve *extends east*.

By definition, a nerve has only a single incident edge, and thus it either extends east or it extends west.

6.2 Homotopy and the Optimum Solution We now want to describe the structure of bones. While it might seem that these are just shortest paths between certain branching and attachment points, this does not account for their role in separating pairs of terminals. In order to specify this role, we need assistance from homotopy.

To facilitate the discussion of homotopy, we first need some definitions.

First, we need the notion of crossings between two paths P and Q in a planar graph. Then we say that P *crosses* Q if, after contracting any common edges of P and Q, there are two edges of P and two edges of Q, all distinct and incident on the same vertex, such that a clockwise rotation around the vertex will see the two edges of P and Q alternatingly. We call this particular vertex the associated *crossing vertex*.

Now we construct a *cut graph* K for G^*, which is a Steiner tree in G^* with the set $V^*_{\mathcal{F}}$ of dual vertices of the terminal faces in $\mathcal{F}$ as terminal set. Here we follow Frank and Schrijver [31, Proposition 1]. First, compute a shortest path between each pair of dual vertices in $V^*_{\mathcal{F}}$. Let $\mathcal{P}$ denote the resulting set of paths. By a slight modification of the weights for the sake of this computation, we can assume all shortest paths in G^* are unique. Then we can assume that any pair P, Q of shortest paths in $\mathcal{P}$ either crosses at most once, or has a common endpoint and do not cross. Now build an auxiliary complete graph on $V^*_{\mathcal{F}}$ with the lengths of the paths in $\mathcal{P}$ as weights and find a minimum spanning tree in this auxiliary graph. The paths in $\mathcal{P}' \subseteq \mathcal{P}$ corresponding to this spanning tree form the cut graph K.

As explained by Frank and Schrijver [31, Proposition 1], we may assume the paths in $\mathcal{P}'$ do not cross. We call the paths in $\mathcal{P}'$ the *spokes* of the cut graph. Note that K has k spokes, each of which can be associated with a unique identifier. We also orient each spoke in an arbitrary direction. Then any oriented path P that crosses a spoke Q can be said to cross Q in a particular direction, depending on whether P goes towards the west or east side of Q.

Definition 6.2. *The crossing sequence or homotopy string of a path P in G^+ is an ordered sequence of the spokes of K crossed by it, along with an indication of the orientation of each crossing. Let P and P' be two paths in G^+. We say that P is* homotopic *to P' if they have the same endpoints as well as the same crossing sequence with respect to K.*

Note that we perform a slight abuse here by considering crossings of a path in G^+ with paths in G^*. However, we will never consider crossings of paths in G^+ that have an endpoint in an augmented dual terminal, alleviating any possible cause for confusion.

Using this precise definition of homotopy, we now fix a particular solution to the instance. Among all possible nerved minimum multiway cuts C of (G, T, ω), we assume henceforth that C is one of which the skeleton of C^+ has the minimum number of crossings with K. Note that the skeleton contains no augmented dual vertices as those have degree 1 in G^+.

From now on, we fix this solution as 'the' optimal solution or 'the' optimum. Let C^+ be the set of corresponding augmented dual edges. It is reasonable to assume that such a solution is inclusion wise minimal, that is, removing any of its edges must make it an infeasible multiway cut.

Our main goal in the next subsections will be to show that (a sufficient part of) the bones of the skeleton of the optimum solution have short homotopy strings and that replacing such parts by parts with the same homotopy string leads to another optimum solution.

6.3 Single Face We now consider a subpath of a bone that contains only attachment points of nerves that extend towards the same direction. In particular, the nerves extend towards the same terminal face. We call such a subpath a *nerve path*. We aim to prove the following crucial property of nerve paths, namely that any nerve path has a homotopy string of bounded length.

Lemma 6.1. *Any nerve path crosses any path of K $\mathcal{O}(k)$ times.*

Proof. We modify the proof of [22, Lemma 5.2] to prove the claim above. Let K be the graph defined above. We treat the dual vertices of the terminal faces as boundary components. Since P_b is a path in the augmented dual G^+, P_b does not pass through any terminal of K, in particular, through the end points of γ.

For sake of simplicity of notation, we assume that the nerve path is the full path P_b. Let the nerves attaching to the ends of P_b be denoted N_1 and N_2.

We push all the crossings of P_b with γ together to a single point p/p. Then the subpaths of P_b between consecutive crossings of P_b with γ form cycles through p/p, which we call loops. The corresponding system of loops is $P_b^* = P_b/p$. Then, P_b^* is a set of pairwise disjoint loops L intersecting at p/p. The contracted point is referred to as the base point of the loops. A face of L is called a *monogon* if it is homemorphic to an open disc (contains no terminal face in its interior) and has only one copy of the base point on its boundary. Likewise, a *bigon* is an open disc with two copies of p/p on its boundary. In the bone P_b, a bigon occurs as a strip bounded by two pieces of the path γ and two pieces of P_b. To show that the number of crossings of P_b with γ is $\mathcal{O}(k)$, we need to show that the degree of p/p is $\mathcal{O}(k)$ in P_b^*.

We claim that no loop in L can be incident on two bigons. That is, in P_b, two strips, each of which is bounded by two pieces of the path γ and two pieces of the path P_b, cannot be glued to each other. We assume the contrary. First, we observe that if a nerve attached to an end point of P_b that originated outside a bigon were to enter it, we could replace it by an arc of γ. Consequently, we would get a multiway cut dual of length smaller than that of C^+, leading us to a contradiction. This, in turn, would exclude configurations where any end point of P_b is enclosed by an arc of it on one side and γ on the other.

Claim 6.1. *Any nerve attached to an end point of P_b that originates outside one of the two bigons that an arc of P_b is incident on, never enters that bigon.*

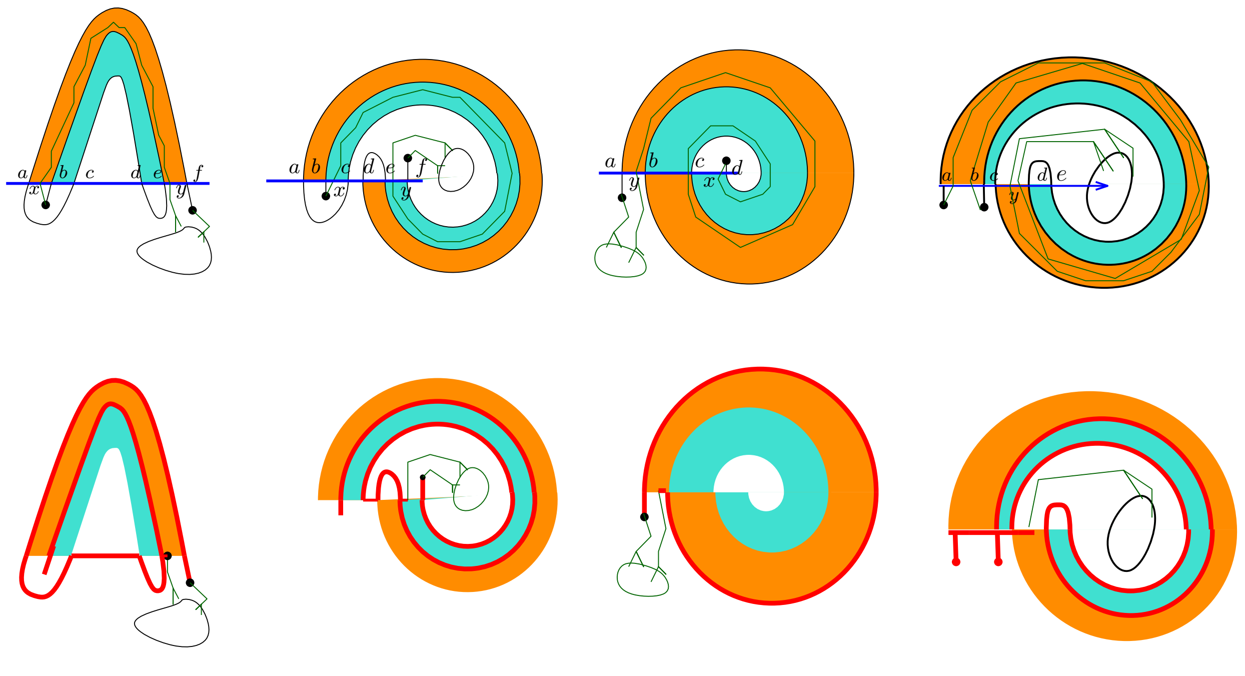

Figure 5: In the top row, there are configurations of P_b wherein a nerve attached to its end point enters one of the two bigons it is incident on. In the bottom row, we show the shortcut path. The two bigons are shaded in different colors. The path P_b is depicted as a black curve, while γ is shown as blue line. The thick red curve depicts the shortcut path.

Proof. Consider the leftmost configuration in Figure 5. Let $p = \{a, b, c, d, e, f\}$ be the set of points at which P_b crosses the path γ. By definition, no bigon must contain in its interior any terminal face vertex of G^*. In order to reach the nerve face F_β, a nerve that enters a bigon must also leave it. Without loss of generality, let this nerve be N_1. Due to planarity of the multiway cut dual, N_1 must not cross any arc of P_b. Hence, it may enter and exit any bigon only by crossing the two arcs of γ on its boundary. Let the first crossing of the nerve with γ be x lying between a and b, and the last one be denoted by y, which lies between e and f. As γ is a shortest path in G^* between any two vertices, $\gamma[x, y]$ is at most the length of $N_1[x, y]$. Also note that due to planarity, all the nerves attached to the arc $P_b[b, e]$, must leave the bigon by crossing $\gamma[e, y]$. Since, there are no terminals in the bigon, $N_1[x, y]$, $P_b[b, e]$, $\gamma[b, x]$ and $\gamma[e, y]$ bound a disc free from obstacles. We can, therefore, replace $N_1[x, y]$ by $\gamma[e, y]$ in C^+ to get a multiway cut of cost no greater than that of C^+. This is a contradiction to the minimality of C^+. We can argue analogously for the other configurations giving rise to this phenomenon. Some of them are shown in Figure 5. The other cases are either symmetric, or trivial to shortcut and for the purpose of brevity have not been explicitly shown here. $\square$

We refer the reader to [22, Figure 2] for an exhaustive list of configurations of any path crossing an edge of the cut graph K, which lead to the occurrence of a loop incident on two bigons in L. As a result of the preceding claim, several of those do not occur in our setting. Hence, we exclude those cases where it is clear that either of the two nerves attached to the end points of P_b must traverse a bigon, in order to reach the nerve face.

Corollary 6.1. *Figure 6 shows all the possible configurations of P_b that form a loop incident on two bigons in L.*

Using arguments similar to those in the proof of Claim 6.1, we can show that whenever a loop of L is incident on two bigons, it is possible to replace P_b by a shorter path. This requires extensive case analysis and we refer the reader to the full version of the paper for the same. We claim that the shortcut path encloses all the terminals that were enclosed by P_b but not $C^+ \setminus P_b$. Therefore, the resultant solution is a valid multiway cut.

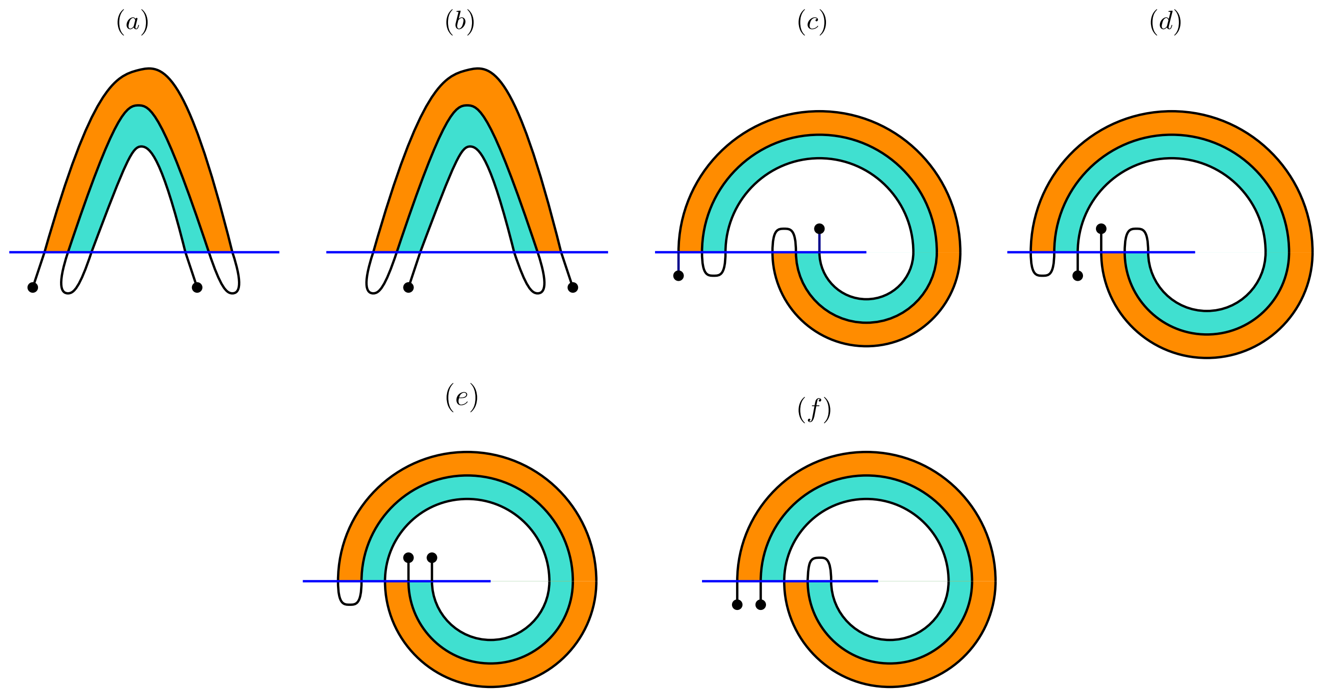

Figure 6: The figure shows all the possible configurations of P_b, when a loop of L is incident on two bigons. One of the bigons is shaded orange and the other turquoise. The black curve is P_b and the blue line is the path γ of K.

Claim 6.2. (♣) *Replacing the arcs of the path P_b along with the nerves attached to them, by the corresponding arcs of γ, yields a valid multiway cut.*

Next, we deal with the monogons. Some monogons may have nerves attached to them that extend towards the nerve face without ever crossing γ. These are the monogons that we cannot replace by the arc of γ bounding them on one side. However, we argue that only $\mathcal{O}(k)$ of these monogons actually exist in C^+.

Without loss of generality, assume that F_β lies to the west of γ. Consider the system of loops. Let *good loops* be those that are present to the east (or west) of γ, possibly have nerves attached to them extending to the west (or east) of γ, and border a monogon or enclose a bigon that it borders. We can remove the good loops from C^+ without contradicting its feasibility by shortcutting (see Figure 7). Loops that contain any terminal face (the end point of a path of K) in their interior are called *obstacle loops*. Now consider loops that form monogons. *Bad monogons* are those that are present on the west (or east) of γ and contain nerves attached to them that extend towards the west (or east) of γ. Figure 7 shows two bad monogons flanking a good loop. There does not seem an easy way to shortcut them, so we must retain them in the solution. We claim that between two obstacle loops, there can be at most four bad monogons.

Now, we deal with the remaining bigons. Whenever we encounter a face of L that is a bigon, we remove one of the two loops incident on it, and iterate until no bigons remain. Therefore, in the proof of [11, Lemma 2.1], which essentially counts the number of obstacle loops, we can multiply the total number of crossings by 10 to get the total number of crossings of P_b with K. The number of obstacle loops remaining in L is $|L'| \leq 6b + 2g - 3$, where g is the genus of the surface on which C^+ is embedded and b is the number of boundary components of the surface. We treat the obstacles we place on the dual vertices of the terminal faces as boundary components. Since there are k terminal faces, and therefore k "obstacles", there are k boundary components in all. Therefore $b = k$ and $g = 0$. This proves that the number of loops, and thus the degree of p/p is $\mathcal{O}(k)$. ∎

6.4 Two faces Let u, v be two endpoints of a shrunken bone b of the shrunken skeleton of C^+. Then b separates two terminal faces $F_\alpha, F_\beta \in \mathcal{F}$ (possibly $F_\alpha = F_\beta$ if b is a bridge of the shrunken skeleton). We denote by P_b

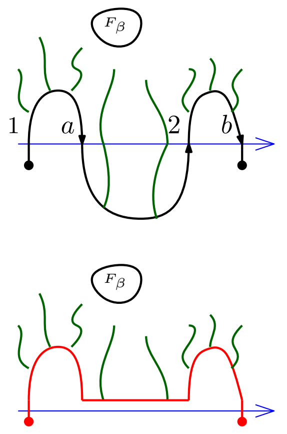

Figure 7: The blue line is the path γ and the arrow indicates its orientation. The black curve shows P_b. The dark green curves are the nerves attached to P_b. The first and third monogons are bad, and the middle loop is good. The figure at the bottom shows the shortcut of the good loop as a thick red curve.

the path corresponding to b in the skeleton of C^+. While the previous section builds sufficient understanding to build the path between consecutive nerves that extend west towards the same terminal face, this does not help if two consecutive attachment points have nerves towards different terminal faces, or towards the same terminal face but one extends east and one extends west. To this end, we need more elaborate methods.

Definition 6.3. *Let x and y be consecutive attachment points on P_b (possibly $x = y$) such that x is the attachment point for a nerve that extends west towards F_α and y is the attachment point for a nerve that extends east towards F_β. Then we call these nerves an alternating pair of nerves.*

Let N_u denote the pair of alternating nerves closest to u along with the path between their attachment points. Let N_v be the corresponding pair closer to v. Contracting the edges in N_u and N_v to form the vertices u^* and v^* respectively, is equivalent to deleting the corresponding edges in G. This forms an isolated, connected region in the plane with pieces of the boundary of F_α and F_β bounding it on two sides. We denote this region by B.

Lemma 6.2. (♣) *The augmented minimum multiway cut dual C^+ restricted to B comprises a minimum Steiner tree containing the vertices u^*, v^*, and the augmented terminals corresponding to terminals on the pieces of the boundary of F_α and F_β bounding B.*

We now consider the structure of the bone for the parts that are not in the region B, or what they look like when there are no two alternating pairs of nerves. In the previous section, we already discussed what happens to nerve paths. Hence, we only need to consider paths between attachment points of alternating nerves and between an attachment point of a nerve and a branching point. The following result is immediate from [22, Lemma 5.2] with the substitution $g = 0$ and $t = k$.

Lemma 6.3. *Each path on a bone of C^+ between the attachment points of a pair of alternating nerves on a bone, as well as the one from a branch point to its closest attachment point, crosses any path of K $\mathcal{O}(k)$ times.*

Finally, we argue that replacing certain subpaths of a bone by a homotopically equivalent path still yields an optimal solution. The proof of this lemma is reminiscent of the proof of [22, Lemma 7.2].

Lemma 6.4. (♣) *Let b be any shrunken bone of the shrunken skeleton of C^+ and let P_b be the path that forms the corresponding bone. Let x and y be two augmented dual vertices on P_b such that $P_b[x, y]$ contains no attachment points nor branching points, except possibly x or y. Then there exists a minimum multiway cut dual which contains a shortest path in the plane homotopic to $P[x, y]$.*

7 Towards a $n^{O(k)}$-time Algorithm

With the structure of the optimum solution in place, we are now ready to develop the algorithm. Our guiding light will be the topology of the optimum solution, which consists of its skeleton and a compact description of its bones and is defined more formally below. As we do not know any optimum solution, we enumerate all possible topologies and argue that we find a feasible, minimum-weight solution for the topology that corresponds to an optimum solution. The algorithm to do this consists of two parts. First, we show that each bridgeless component of the skeleton in the topology can be treated separately. Second, for each bridgeless component of the skeleton, we build a sphere-cut decomposition and perform dynamic programming on the decomposition that finds a solution according to the topology.

7.1 Topology We first define a topology and then show that all topologies can be efficiently enumerated.

DEFINITION 7.1. *A topology consists of a triple (S, s, h):*

- *a connected plane multigraph S (possibly containing parallel edges but no self-loops) with k faces. This is the (shrunken)* skeleton *of the topology. We direct the edges of the skeleton in an arbitrary fashion to obtain a* directed skeleton*;*
- *for each face of S, a unique corresponding face in $\mathcal{F}$;*
- *for each shrunken bone b, separating faces f_α, f_β of the skeleton (possibly $\alpha = \beta$), a structural description, describing:*
  - *a (possibly empty) ordered multisubset $s(b)$ of $\{\alpha, \alpha, \beta, \beta\}$ such that $s(b)[i]$ and $s(b)[i+1]$ are not equal for any $1 \leq i < |s(b)|$, except possibly for $i = 2$ when $|s(b)| = 4$;*
  - *for any $1 \leq i \leq 2|s(b)| + 1$, $h(b)[i]$ is a homotopy string of length $O(k)$. When $|s(b)| = 4$, then $h(b)[5]$ is unspecified.*

If $\alpha = \beta$, then we use α or β to denote east and west respectively in $s(b)$.

The intuition behind the definition of a topology is as follows. The directed skeleton guides the overall structure of the solution that will be found by our algorithm. Ideally, it is equivalent to the shrunken skeleton of an optimum solution. By Lemma 5.2, it has k faces and is connected. Moreover, each face of the shrunken skeleton corresponds to a unique face in $\mathcal{F}$. The direction of the skeleton will prove useful in later definitions and algorithms.

The multiset $s(b)$ describes the direction of maximal sets of consecutive nerves along the bone that have the same direction. By this intuition, it makes sense to force the directions to be alternating, as in the definition. We also note that we do not need to specify more than four such sets, because when we have four sets, we obtain two alternating pairs of nerves and can apply Lemma 6.2 to the region between them.

For each set of consecutive nerves as described by $s(b)$, we use $h(b)$ to denote the homotopy of the (up to four) subpaths of the bone to which the nerves attach. We also use $h(b)$ to describe the homotopy of the (up to five) subpaths of the bone between these sets of nerves. Following Lemma 6.1, each of those homotopy strings has length $O(k)$. When $|s(b)| = 4$, we do not (need to) specify the homotopy of the middle subpaths of the bone, as we handle this part as in Lemma 6.2.

Recall that C is the optimal solution and C^+ its set of corresponding augmented dual edges. We call a topology (S, s, h) *optimal* if the shrunken skeleton S^+ of C^+ is equivalent to (the underlying graph of) S, each face of $\mathcal{F}$ is assigned to the same face in S^+ and S, and for each bone b of the shrunken skeleton, directed in an arbitrary fashion:

- there are $|s(b)|$ maximal sets of nerves, where nerves in the i-th set are towards $F_{s(b)[i]}$, and their attachment points are consecutive and uninterrupted on b; the nerves preceding a nerve from the i-th set belong to the i-th set or the $i-1$-th set, or if $|s(b)| = 4$, can be arbitrary nerves to F_α or F_β, until a nerve of the $i-1$-th set is hit;
- the part of the bone between the nerves at the ends of the i-th set of nerves has homotopy string $h(b)[i]$;

- the part of the bone between the i-th and $i+1$-th sets of nerves, for $0 \leq i \leq |s(b)|$ where we pretend the 0-th nerve is the one end of the bone and the $|s(b)|+1$-th nerve is the other end of the bone, has homotopy string $h(b)[2i+1]$. When $|s(b)| = 4$, we do not have to satisfy this for $i = 2$.

Note that since we fixed C, there is a unique optimal topology, which effectively describes C. We may thus speak of 'the' optimal topology.

Our goal is to enumerate all topologies. Then for each topology, we aim to find a solution that corresponds to this topology. By considering all topologies, we ensure that we consider the optimal topology at some point. Then we argue that we find a multiway cut of minimum weight. Before proceeding with that algorithm, we bound the number of topologies and show how to enumerate them. We require the following auxiliary lemma.

Lemma 7.1. (♣) *There are $2^{O(k \log k)}$ connected plane multigraphs with k faces and without self-loops. Moreover, they can be enumerated in the same time.*

Lemma 7.2. (♣) *There are $2^{O(k^2 \log k)}$ different topologies. Moreover, they can be enumerated in the same time.*

7.2 Broken Bones and Splints We now become more formal about our interpretation of topologies. Let (S, s, h) be a topology. We define the following notion, which states how we interpret s and h for a shrunken bone b.

Definition 7.2. *Let (S, s, h) be a topology. Let b be a shrunken bone (edge) of the shrunken skeleton S, separating the faces f_α and f_β, and directed from u to v. Let $s(b) = \{\gamma_1, \ldots, \gamma_{|s(b)|}\}$, where each $\gamma_j \in \{\alpha, \beta\}$. Let F_α and F_β be the terminal faces enclosed by the faces f_α and f_β, respectively. A broken bone is a tuple that consists of intervals I_α and I_β of augmented terminals lying on F_α and F_β respectively, augmented dual vertices $x_1^+, \ldots, x_{|s(b)|+1}^+$ and $y_0^+, \ldots, y_{|s(b)|}^+$ that are not augmented dual terminals, and nerves $N_1, \ldots, N_{2|s(b)|}$ towards F_α and F_β, where:*

- *for each $1 \leq j \leq |s(b)|$, nerves N_{2j-1} and N_{2j} extend towards γ_i and have root x_j^+ and y_j^+ respectively. These nerves might be the same, but are non-empty.*
- *the interval of a nerve among $N_1, \ldots, N_{2|s(b)|}$ towards F_α (resp. F_β) is a subinterval of I_α (resp. I_β). Moreover, these subintervals appear in the order indicated by their indices on I_α and I_β (but do not necessarily cover I_α and I_β completely);*
- *if I_α contains at least two augmented terminals, then a prefix of I_α is the interval of a nerve among $N_1, \ldots, N_{2|s(b)|}$. The same holds for a suffix of I_α (possibly, this is the same nerve). The same holds with respect to I_β if it contains at least two augmented terminals.*

A splint fixing a given broken bone is a subset D^+ of the edges of the augmented dual graph G^+ with the following properties:

- *for each terminal t between I_α (or between I_β), there is a unique bounded face of D^* that encloses t, where D^* is the set of dual edges corresponding to the edges of D^+.*
- *there is a path P_b in D^+ between y_0^+ and $x_{|s(b)|+1}^+$. These two augmented dual vertices are called the* ends *of the splint.*
- *$x_1^+, \ldots, x_{|s(b)|+1}^+$ and $y_0^+, \ldots, y_{|s(b)|}^+$ are in D^+ and appear in the order y_0^+, x_1^+, y_1^+, x_2^+, y_2^+, $\ldots$, $x_{|s(b)|+1}^+$ on P_b.*
- *for each $0 \leq j \leq |s(b)|$ (except $j = 2$ when $|s(b)| = 4$), the (possibly empty) subpath of P_b from y_j^+ to x_{j+1}^+ consists only of vertices that have degree 2 in D^+ and has homotopy string equal to $h(b)[2j+1]$.*
- *for each $1 \leq j \leq |s(b)|$, the only vertices on the subpath of P_b between x_j^+ and y_j^+ that have degree more than 2 in D^+ are the attachment points of nerves towards F_{γ_j}; these nerves include N_{2j-1} and N_{2j}. The subpath has homotopy string equal to $h(b)[2j]$.*
- *if $|s(b)| = 4$, then for the vertices on the subpath of P_b between y_2^+ and x_3^+, the only vertices that have degree more than 2 in D^+ are the attachment points of nerves towards F_α or F_β.*

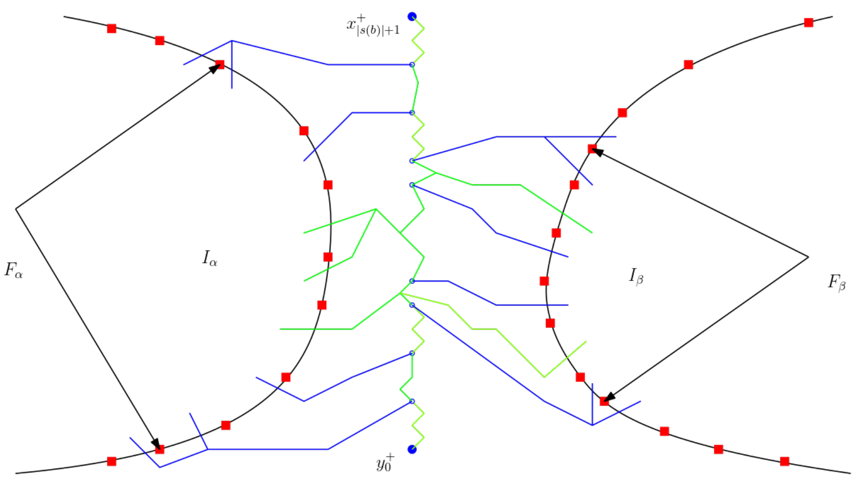

Figure 8: In this example, $s(b) = \{\alpha, \beta, \beta, \alpha\}$. The blue lines depict the "guessed" nerves attached to the broken bone. In green, we depict the splint that is used to fix the broken bone. The arrows point to the first and last terminals of the intervals I_α and I_β.

- *the intervals of all the aforementioned nerves jointly partition I_α and I_β with the exception that for each $1 \le j \le |s(b)|$, the nerves N_{2j-1} and N_{2j} are not necessarily distinct (but are distinct from all other nerves).*

We refer to Figure 8 for an illustration of a splint and its broken bone.

Note that in the definition of a broken bone, it is not strictly necessary to specify $x^+_1, \dots, x^+_{|s(b)|}$ and $y^+_1, \dots, y^+_{|s(b)|}$, as their definition is implied by $N_1, \dots, N_{2|s(b)|}$. Similarly, we do not actually need all the nerves we have specified when $|s(b)| = 4$. We still write this to streamline and simplify the definition (which admittedly is already quite complex).

We now define the notion of an optimal broken bone. We refer back to the definition of an optimal topology to recall the $|s(b)|$ maximal sets of nerves defined there. For the optimal solution C, the optimal topology, and a shrunken bone b, we call a broken bone $x^+_1, \dots, x^+_{|s(b)|+1}$, $y^+_0, \dots, y^+_{|s(b)|}$, $N_1, \dots, N_{2|s(b)|}$, I_α, I_β *optimal* if for the bone of C^+ corresponding to b, its ends are y^+_0 and $x^+_{|s(b)|+1}$ in the direction of the directed skeleton, the i-th set of nerves has the nerves N_{2i-1} and N_{2i} at the ends of the set (where N_{2i-1} and N_{2i} are possibly equal), and the intervals I_α and I_β correspond to the union of the intervals of all nerves that have their attachment point on the bone.

The latter may lead to a nerve that attaches to a branching point and its corresponding interval to be assigned to two broken bones (for the two bones that meet at the branching point and whose shrunken bones bound the same face of the skeleton). In that case, we break ties arbitrarily, but in such a way that all nerves that attach to this branching point are assigned to the same bone and no terminal face is enclosed by the union of any two nerves assigned to the same bone in this way. This ensures that each augmented terminal is in some interval of some optimal broken bone and moreover, no terminal face and its broken bones 'interrupt' the sequence of nerves of the bone.

Finally, we call a splint *optimal* if it fixes an optimal broken bone and all its nerves are the nerves of C^+ that were defined to belong to the bone. Again, an optimal broken bone and splint are unique with respect to C, so we may speak of 'the' optimal broken bone and splint of a bone.

For a given topology and a particular shrunken bone of the topology, our goal is to enumerate all broken bones for this shrunken bone and find a minimum-weight splint for it. By enumerating all broken bones, we ensure that we consider the optimal broken bone. Then the minimum-weight splint that we find will have the same weight and structure of the optimal splint. We discuss that algorithm in a moment, but first argue that we can enumerate all possible broken bones efficiently.

PROPOSITION 7.1. (♣) *There are $O(n^{24})$ distinct broken bones fixed by a splint.*

7.3 Splinting Algorithm We now describe an algorithm that, given a topology (S, s, h) and a broken bone $x_1^+, \ldots, x_{|s(b)|+1}^+, y_0^+, \ldots, y_{|s(b)|}^+, N_1, \ldots, N_{2|s(b)|}, I_\alpha, I_\beta$ for a bone b of S, finds a splint of minimum weight.

For any $0 \leq i \leq |s(b)|$ (except $i = 2$ when $|s(b)| = 4$), to find the path from y_i^+ to x_{i+1}^+, we invoke the algorithm by Frank and Schrijver [31, Section 5] to find the shortest path with the homotopy string $h(b)[2i + 1]$. These paths do not contain any attachments points of nerves.

The following algorithm computes the weight of the part of a minimum weight splint between the vertices x_i^+ and y_i^+ of the augmented dual, for all $1 \leq i \leq |s(b)|$. Note that this is a nerve path. For simplicity, we consider only $i = 1$, as the other cases are similar. Without loss of generality, $s(b)[1] = \alpha$. We find the part of the splint starting at x_1^+, with its corresponding nerve N_1, and ending at y_1^+, with its corresponding nerve N_2. Also, we assume that N_1 spans the interval $\{{}^\alpha t_\ell^+, \ldots, {}^\alpha t_{\ell'}^+\}$ and N_2 spans the interval $\{{}^\alpha t_j^+, \ldots, {}^\alpha t_{j'}^+\}$. Then the interval covered by all nerves attaching to this nerve path is $I_1 = \{{}^\alpha t_\ell, \ldots, {}^\alpha t_{j'}\}$.

We compute this part of the splint as follows. By $c[x^+, a, a']$, we denote the weight of the unique nerve on $x^+ \in V(G^+)$ and the interval $\{{}^\alpha t_a^+, \ldots, {}^\alpha t_{a'}^+\}$, which can be computed by Lemma 5.5. By $c'[x^+, a, a', e]$ we denote the weight of a partial splint passing through the vertex $x^+ \in V(G^+)$, that encloses every terminal between the augmented terminals $\{{}^\alpha t_\ell^+, \ldots, {}^\alpha t_{a'}^+\}$, contains the unique nerve (x^+, a, a'), and the partial nerve path of the homotopy given by the prefix of $h(b)[2]$ of length e.

The dynamic programming algorithm is given in Algorithm 1 below. We use $d_{h(b)[2](e',e]}(x^+, x'^+)$ to denote the length of a shortest path between x^+ and x'^+ with homotopy string equal to the substring of $h(b)[2]$ between indices e' and e (not including the symbol on index e'). In other words, this substring is equal to the prefix of length e minus the prefix of length e'. This shortest path length can again be computed by the algorithm by Frank and Schrijver [31, Section 5].

We also use $\min^*$ to denote that the minimum is only allowed over certain combinations. In particular, in Line 7, the nerve (x^+, a, a') being considered in combination with nerve $(x'^+, z, a - 1)$ and the path between x'^+ and x^+ of homotopy string $h(b)[2](e', e]$ must define a region that only encloses ${}^\alpha t_{a-1}$ (the terminal inbetween the two nerves). Moreover, the nerve (x^+, a, a') itself must create a region for every terminal in $\{{}^\alpha t_a, \ldots, {}^\alpha t_{a'-1}\}$. A similar constraint holds in Line 8.

Algorithm 1 Nerve Path Algorithm

```
$c'[x_1^+, \ell, \ell', 0] = c[x_1^+, \ell, \ell']$
$c'[x^+, a, a', e] = \infty$ for all $x^+ \neq x_1^+$, $e \neq 0$, $a \neq \ell$, or $a' \neq \ell'$
for $\ell' < a' < j'$ do
    for $\ell' < a \leq a'$ do
        for $x^+ \in V(G^+)$ do
            for $0 \leq e \leq |h(b)[2]|$ do
                $c'[x^+, a, a', e] = \min^*_{\ell \leq z < a,\ x'^+ \in V(G^+),\ 0 \leq e' \leq e} \left\{ c'[x'^+, z, a-1, e'] + c[x^+, a, a'] + d_{h(b)[2](e',e]}(x'^+, x^+) \right\}$
return
    $\min^*_{\ell \leq z < j,\ x^+ \in V(G^+),\ 0 \leq e \leq |h(b)[2]|} \; c'[x^+, z, j-1, e] + d_{h(b)[2](e,|h(b)[2]|]}(x^+, y_1^+) + c[y_1^+, j, j']$
```

The same algorithm is used to compute all the other parts of the splints, too. Note that even though the algorithm only computes an optimal value, it can be easily modified to return the optimal solution (nerves and nerve path).

Finally, if $|s(b)| = 4$, then we know from Lemma 6.2 that embedded in the region bounded by the nerves attached to y_1^+ and x_2^+, and y_3^+ and x_4^+, along with the paths between them, as well as the segments of F_α and F_β bounding the region on either side is a minimum Steiner tree with its terminals on the boundary of the region. Using the algorithm of Erickson *et al.* [29] (see also Bern [6]), we find the minimum Steiner tree. This finishes the description of the splinting algorithm.

Lemma 7.3. (♣) *Given a topology (S, s, h), a bone b of S, and the optimal broken bone for b, we can fix the broken bone by a minimum-weight splint found through the splinting algorithm. Moreover, the splinting algorithm runs in time $n^{\mathcal{O}(1)}$.*

Using the algorithm, we can compute a splint for each broken bone of each topology. Since the homotopy of each nerve path and each path between branching points and the starts/ends of nerve paths are maintained, it follows that we can replace each bone and attached nerves of the optimum solution by the minimum-weight splint computed by the algorithm for the corresponding optimal broken bone. In particular, the homotopy strings ensure that the number of crossings of the bones with the cut graph K stays minimum.

Definition 7.3. *We call an optimum solution splinted if it has the property that each of its bones and attached nerves are splints.*

Remark 7.1. *From now on, we assume that the optimum solution is splinted.*

For the sake of intuition, we note that we have now gathered sufficient ideas to prove a $2^{O(k^2 \log k)} n^{O(k)}$ time algorithm for Planar Multiway Cut. In particular, we can enumerate all topologies in $2^{O(k^2 \log k)}$ time and then enumerate all broken bones for any shrunken bone in $n^{O(k)}$ time total. For the combination of the optimal topology and the optimal broken bones for each shrunken bone, the splinting algorithm then delivers an optimal solution for each shrunken bone, which can be combined to form an optimal solution to the whole (modulo some details). In the next section, we argue how to reduce the $n^{O(k)}$ factor down to $n^{O(\sqrt{k})}$.

8 Algorithm using Sphere-Cut Decomposition

We are now ready to discuss how we go from a topology to a multiway cut. If the topology is optimal, we argue that we find a minimum-weight multiway cut that has the same structure as the optimal multiway cut C. Our aim is to employ Theorem 3.1 on the shrunken skeleton to get a small branch decomposition and then apply a dynamic program. However, as already noted a few times, the shrunken skeleton might have bridges and Theorem 3.1 cannot be applied directly, but only on the bridge blocks of the shrunken skeleton. Therefore, we first develop a dynamic program that combines solutions of the bridge blocks of the shrunken skeleton, effectively reducing the problem.

8.1 Reduction to Bridge Blocks When we want to combine solutions of different bridge blocks, it is natural to use the bridge block tree in a dynamic program. However, if we do this naively, we immediately run into the issue that in order to compute a solution for a non-trivial bridge block, we need to know the solutions for all bridge blocks contained in its bounded faces. This can be resolved by enforcing an ordering on the computation of the bridge blocks that depends on the embedding. To that end, we proposed the embedding-aware bridge block (eabb) tree in Section 3.2. We now show how to use it.

Let (S, s, h) be a topology. Let $L = L(S)$ be the eabb tree for S and let $\mathcal{B}$ denote the set of bridge blocks of S. For a BB-node l of L, let $\mathcal{B}(l)$ denote the bridge block corresponding to l. Extending this notation, for a subtree L' of L, we use $\mathcal{B}(L')$ to denote the set of all bridge blocks corresponding to BB-nodes in L'. For a node l of L, we use L_l to denote the subtree of L rooted at l. In particular $L_{\ell(L)} = L$, where we recall that $\ell(L)$ is the root of L.

We need the following notion, which formalizes the idea of an 'innermost' component of a plane graph.

Definition 8.1. *Let H be any plane graph. Let J be the union of a subset of the biconnected components of H such that there is a single face f of $H - J$ that encloses J and there is a single face f' of J that encloses $H - J$. We call J an internal set, f its enclosing region, and f' its exclosing region. The intersection of the enclosing region and exclosing region is called the middle region.*

Lemma 8.1. (♣) *For any node l of L, $\mathcal{B}(L_l)$ is an internal set of S.*

We now perform a bottom-up dynamic programming with respect to L. The crux here is to understand the relation that a child l has with its parent. This is formed by two parts. The first, more easy part is the cut vertex shared by neighboring bridge blocks. The second, more complicated part is the terminal face F_l in the middle region induced by the internal set $\mathcal{B}(L_l)$. The terminals in T_l are covered jointly by $\mathcal{B}(L_l)$, the blocks induced by siblings of l in L, and by the block induced by l's parent. Using a similar argument as in the proof of Lemma 5.4, we can see that each of these is responsible for a single interval of T_l. To help in the computations, we additionally consider the first nerves that cover the prefix and suffix of this interval. We now expand on this intuition of the dynamic program and define the table more formally.

Let l be a node of L. Define $w = w(l)$ as follows: if l is a C-node, then let w be the corresponding cut vertex; if l is a BB-node and l has a parent in L, then this parent is a C-node and we let w be its corresponding cut vertex; otherwise, let w be any vertex of $\mathcal{B}(l)$. Let $F_l \in \mathcal{F}$ denote the unique terminal face in the middle region of $\mathcal{B}(L_l)$.

Definition 8.2. *Given a vertex w^+ of the augmented dual, an interval I_l of F_l, and two (possibly empty) nerves N_l^1, N_l^2 towards F_l, a stretcher is a set D^+ of augmented dual edges such that:*

(i) *the interval of N_l^1 is a prefix of I_l and the interval of N_l^2 is a suffix of I_l. N_l^1 and N_l^2 are either both empty or both non-empty and cannot be empty if I_l has more than one terminal.*

(ii) *there is a (possibly empty) set of nerves towards F_l whose augmented terminal sets are intervals that jointly partition I_l. N_l^1 and N_l^2 are among those nerves;*

(iii) *for any terminal $t \in T_l$ between I_l, there is a bounded face of D^* that encloses only t and no other terminals. Here D^* is the set of dual edges corresponding to the augmented dual edges in D^+;*

(iv) *for any terminal $t \in T_\alpha$ of a bounded face f_α in the subgraph of S induced by the bridge blocks of L_l, there is a bounded face of D^* that encloses only t and no other terminals;*

(v) *D^+ contains a splint for each bone in $\mathcal{B}(L_l)$ and N_l^1 and N_l^2 are included in these splints;*

(vi) *w^+ is the vertex of D^+ that is an end of all splints for the bones in $\mathcal{B}(L_l)$ that are incident on w.*

We call w^+, I_l, N_l^1, and N_l^2 a binder of the stretcher and a binder *for l. We say a binder is* valid *if it adheres to* (i).

Proposition 8.1. (♣) *For each node l of L, there are $n^{O(1)}$ binders. These can be enumerated in the same time.*

Consider any vertex l of the embedding-aware block-cut tree L of S. Let I_l be the union of intervals of the nerves in C^+ towards F_l that attach to a block in $\mathcal{B}(L_l)$ and let N_l^1 and N_l^2 be the nerves whose intervals are a prefix and suffix of I_l respectively. Following Remark 7.1, C^+ is a union of splints for each of the shrunken bones of S. Let w^+ be the vertex of C^+ that is an end of all splints for the bones in $\mathcal{B}(L_l)$ that are incident on $w = w(l)$. Then we call w^+, I_l, N_l^1, and N_l^2 an *optimal binder*. Note that for each optimal binder, a stretcher does exist, which we call an *optimal stretcher*. We may speak of 'the' optimal binder and stretcher, because the minimum-weight solution and topology are uniquely defined.

Then for each binder I_l, N_l^1, N_l^2, w^+, define $A_l[I_l, N_l^1, N_l^2, w^+]$ as a minimum-weight stretcher for this binder. If no such stretcher exists, then we define $A_l[I_l, N_l^1, N_l^2, w^+]$ to be the set of all augmented dual edges. We argue that the table A can be computed in a dynamic programming fashion.

We first consider how to compute a table entry for I_l and w^+ if l is a BB-node.

Lemma 8.2. *For any BB-node l of L and the optimal binder B for l, we can compute a minimum-weight stretcher when given minimum-weight stretchers for all optimal binders of all children of l. Moreover, it can be computed in $n^{O(\sqrt{|\mathcal{B}(l)|})}$ time.*

Proof. We only provide an intuitive sketch of the proof and defer the technical details to the full version.

Note that the BB-node is either a single edge or a connected, bridgeless graph without self-loops. Hence, it has a sphere-cut decomposition by Theorem 3.1. Assuming that the table A has been computed for all children of l in L (which are C-nodes), we compute the table entry for l. By Lemma 8.1, $\mathcal{B}(L_l)$ is an internal set. Let F_l be the terminal face in the middle region of $\mathcal{B}(L_l)$. Consider a binder I_l, N_l^1, N_l^2, w^+ for l. We assume that the binder is valid. We now wish to compute a stretcher for this binder by a dynamic program over the sphere-cut decomposition.

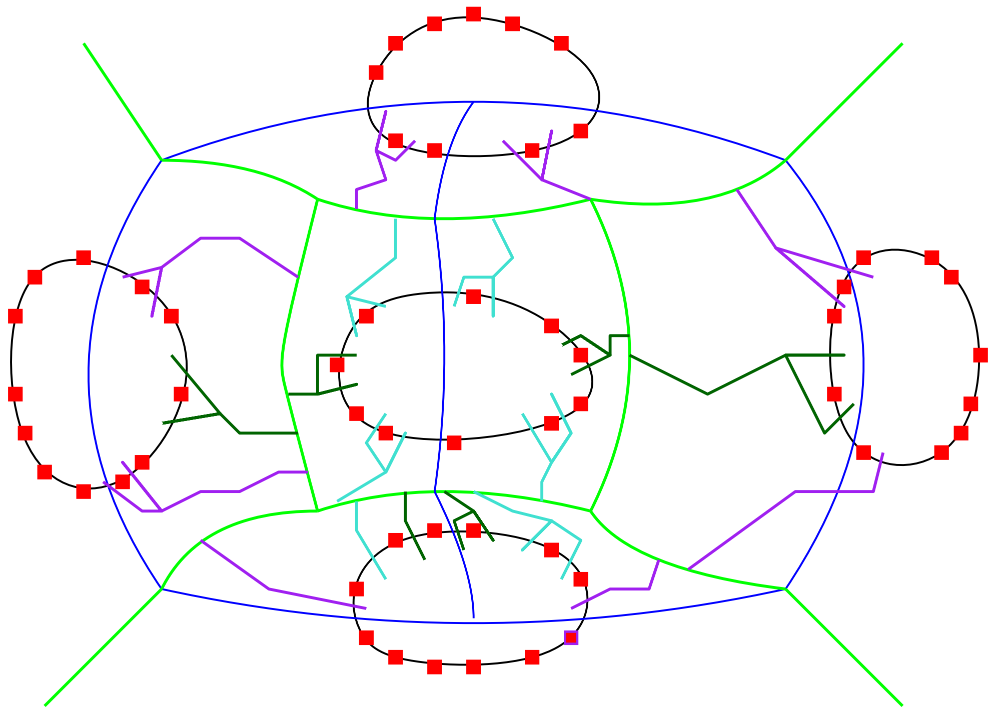

Figure 9: The noose $\vec{\gamma}$ is drawn in blue. It intersects the skeleton only in its vertices. The skeleton here is drawn in green. The outermost nerves N_1 and N_2 of each face of the skeleton are shown in purple. These form a part of the partial binder. The outermost nerves that are unique to the partial binders for the child edges of the current edge (x, y) in the sphere-cut decomposition tree R are drawn in turquoise.

Now let (R, η, δ) be a sphere-cut decomposition of $\mathcal{B}(l)$; refer back to Section 3.1 for the definitions. For every pair x, y of adjacent vertices of R such that y is the parent of x, consider the noose $\vec{\gamma} = \delta(x, y)$. Let $\mathcal{F}_{\vec{\gamma}} \subseteq \mathcal{F}$ be the set of terminal faces for which the corresponding face of S is intersected by $\vec{\gamma}$ and let $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})} \subseteq \mathcal{F}$ be the set of terminal faces for which the corresponding face of S is enclosed by $\mathbf{enc}(\vec{\gamma})$. We include in $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ the bridge blocks of any associated children of any bone enclosed by $\mathbf{enc}(\vec{\gamma})$. Note that $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ is possibly empty, but $\mathcal{F}_{\vec{\gamma}}$ never is.

To develop the dynamic programming algorithm over the sphere-cut decomposition, we first need a notion of what the partial solution is that we compute during the algorithm. To this end, we need the notion of a partial binder and a partial stretcher. The intuition is that a partial binder is a dynamic programming state for the intersection of a noose $\vec{\gamma}$ with the hypothetical solution prescribed by the topology. For each vertex of the topology intersected by $\vec{\gamma}$, the state stores a corresponding vertex of the augmented dual. For each terminal face F_α in $\mathcal{F}_{\vec{\gamma}}$, we only see part of the hypothetical solution, which is described by an interval of T_α and the (possibly empty) first and last nerves that (possibly together with other nerves) cover the interval. Then a corresponding partial stretcher is the entry stored in the dynamic programming table for the partial binder, which essentially stores a solution for all terminal faces in $\mathcal{F}_{\mathbf{enc}(\vec{\gamma})}$ and a partial solution for all terminals between the mentioned interval.

The dynamic program itself follows along familiar lines, even though it is very technical. See Figure 9 for the idea of how to join partial binders and partial stretchers. We highlight two important special situations that occur.

First, consider the situation when F_l, the face in the middle region of the internal set $\mathcal{B}(L_l)$, is in $\mathcal{F}_{\vec{\gamma}}$. Recall that the binder specifies an interval of T_l (and certain nerves) that must be covered by the stretcher that we are computing. Hence, in this case, the partial solution needs to satisfy these demands as well (or at least, partially).

Second, we need to incorporate the solutions for the children of l in L. Since we effectively consider $\mathcal{B}(l)$ as a collection of (shrunken) bones, we need to associate a bone to each child to ensure this. We make this more formal. For each C-node of L that is child l' of l, corresponding to a cut vertex c, consider the middle region $f_{l'}$ of $\mathcal{B}(L_{l'})$ and let b, b' be the bones on the boundary of this middle region that are incident on c. Since l' is a child of l, it follows from the definition of an eabb tree that b and b' are well defined and distinct. Moreover, they both belong to $\mathcal{B}(l)$. Pick one of b, b' in a consistent manner (say b) and associate l' with this bone. We call l' an *associated child* of the bone b, the cut vertex c, and the middle region $f_{l'}$. In the base case of the algorithm, when we consider a single shrunken bone b, we will then use the table A for these associated children to determine their nerves towards $f_{l'}$. ◻

We now describe how to compute a table entry for I_l and w^+ if l is a C-node. Let $l_1, \ldots, l_q$ denote the children of l in L. Note that the children of l are all BB-nodes. We assume that the bridge blocks appear in this order around w in S. Then we compute $A_l[I_l, N_l^1, N_l^2, w^+]$ as the minimum-weight union of $A_{l_j}[I_{l_j}, N_{l_j}^1, N_{l_j}^2, w^+]$ over all families $I_{l_1}, \ldots, I_{l_q}$ of (possibly empty) intervals of F_l that form a partition of I_l and appear in this order on I_l and over all nerves $N_{l_j}^1, N_{l_j}^2$ whose interval is a prefix respectively suffix of I_{l_j}. During this computation, we discard any union that does not form a stretcher. If all unions are discarded in this way, we set $A_l[I_l, N_l^1, N_l^2, w^+]$ equal to the set of all augmented dual edges.

Lemma 8.3. (♣) *For any C-node l of L and the optimal binder B for l, we can compute a minimum-weight stretcher when given minimum-weight stretchers for all optimal binders of all children of l. Moreover, the table A will store a stretcher of minimum weight for the optimal binder. Finally, it can be computed in $O(qn^{O(1)})$ time*

9 Proof of Theorem 1.1

The following theorem implies the algorithmic part of Theorem 1.1. As already mentioned, the lower bound immediately follows from Marx's result [45].

Theorem 9.1. *Consider an instance (G, T, ω) of* Planar Multiway Cut. *We can compute a minimum-weight multiway cut of (G, T) in $2^{\mathcal{O}(k^2 \log k)} n^{\mathcal{O}(\sqrt{k})}$ time, where k is the number of faces needed to cover all terminals.*

Proof. First, in time $2^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ through the algorithm of Bienstock and Monma [7], we compute a set of faces of size k that covers all the terminals. This is the set $\mathcal{F}$. We then transform the instance through the transformations of Section 4, and particularly Lemma 4.1. We then apply the algorithm of Lemma 4.2 to reduce to the case when the dual of any optimum solution is connected.

Now, by Lemma 7.2, we can enumerate all topologies in $2^{\mathcal{O}(k^2 \log k)}$ time. For each topology (S, s, h), build an embedding-aware bridge block tree $L = L(S)$ in linear time by Lemma 3.1. Now we perform the dynamic program of Lemma 8.3 and 8.2 on $L(S)$. Since the nodes corresponding to the same cut vertex of H effectively form a subtree of L through Lemma 3.3, the binders contain the same cut vertex w and w^+ throughout. Then it follows that by induction on the depth of l in L that a minimum-weight stretcher is computed for each optimal binder. Finally, for the root node of L, we consider the minimum-weight stretcher that is found among all binders. One of those will be the optimal binder, and thus we find a stretcher of weight at most the optimal stretcher for the optimal binder. Recalling that a topology has $O(k)$ vertices, it follows from Lemma 8.3 and 8.2 that the running time is $n^{\mathcal{O}(\sqrt{k})}$. ◻

References

[1] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[2] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for planar multiway cut. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 639–655. SIAM, 2012.

[3] Mohammadhossein Bateni, Mohammadtaghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5), October 2011.

[4] Cédric Bentz. A polynomial-time algorithm for planar multicuts with few source-sink pairs. In Dimitrios M. Thilikos and Gerhard J. Woeginger, editors, *Parameterized and Exact Computation - 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, volume 7535 of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2012.

[5] Cédric Bentz. An FPT algorithm for planar multicuts with sources and sinks on the outer face. *Algorithmica*, 81(1):224–237, 2019.

[6] Marshall Bern. Faster exact algorithms for Steiner trees in planar networks. *Networks*, 20:109–120, 2006.

[7] Daniel Bienstock and Clyde L. Monma. On the complexity of covering vertices by faces in a planar graph. *SIAM J. Comput.*, 17(1):53–76, 1988.

[8] Glencora Borradaile, Philip N. Klein, and Claire Mathieu. Steiner tree in planar graphs: An $O(n \log n)$ approximation scheme with singly-exponential dependence on epsilon. In Frank Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *Algorithms and Data Structures*, pages 275–286, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[9] Gruia Călinescu, Howard Karloff, and Yuval Rabani. An improved approximation algorithm for multiway cut. In *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing*, pages 48–52, 1998.

[10] Yixin Cao, Jianer Chen, and Jia-Hao Fan. An $O^*(1.84^k)$ parameterized algorithm for the multiterminal cut problem. *Inf. Process. Lett.*, 114(4):167–173, 2014.

[11] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey. Splitting (complicated) surfaces is hard. In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, SCG '06, page 421–429, New York, NY, USA, 2006. ACM.

[12] Erin W. Chambers, Jeff Erickson, and Amir Nayyeri. Minimum cuts and shortest homologous cycles. In John Hershberger and Efi Fogel, editors, *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pages 377–385. ACM, 2009.

[13] Danny Chen and Xiadong Wu. Efficient algorithms for k-terminal cuts on planar graphs. *Algorithmica*, 38:299–316, 02 2004.

[14] Danny Z. Chen and Jinhui Xu. Shortest path queries in planar graphs. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, STOC '00, page 469–478, New York, NY, USA, 2000. ACM.

[15] Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. In Frank Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *Algorithms and Data Structures*, pages 495–506, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[16] Kevin K. H. Cheung and Kyle Harvey. Revisiting a simple algorithm for the planar multiterminal cut problem. *Oper. Res. Lett.*, 38(4):334–336, 2010.

[17] Rajesh Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Fixed-parameter tractability of directed multiway cut parameterized by the size of the cutset. *SIAM Journal on Computing*, 42(4):1674–1696, 2013.

[18] Vincent Cohen-Addad, Éric Colin de Verdière, and Arnaud de Mesmay. A near-linear approximation scheme for multicuts of embedded graphs with a fixed number of terminals. *SIAM J. Comput.*, 50(1):1–31, 2021.

[19] Vincent Cohen-Addad, Éric Colin de Verdière, Dániel Marx, and Arnaud de Mesmay. Almost tight lower bounds for hard cutting problems in embedded graphs. *J. ACM*, 68(4):30:1–30:26, 2021.

[20] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Trans. Comput. Theory*, 5(1), May 2013.

[21] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[22] Éric Colin de Verdière. Multicuts in planar and bounded-genus graphs with bounded number of terminals. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015*, pages 373–385, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[23] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[24] Frederic Dorn, Eelko Penninkx, Hans L. Bodlaender, and Fedor V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[25] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[26] Jeff Erickson, Kyle Fox, and Amir Nayyeri. Global minimum cuts in surface embedded graphs. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1309–1318. SIAM, 2012.

[27] Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1166–1176. SIAM, 2011.

[28] Jeff Erickson and Amir Nayyeri. Shortest non-crossing walks in the plane. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 297–208. SIAM, 2011.

[29] Ranel E. Erickson, Clyde L. Monma, and Arthur F. Veinott. Send-and-split method for minimum-concave-cost

network flows. *Mathematics of Operations Research*, 12(4):634–664, 1987.

[30] Lester R. Ford and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[31] András Frank and Alexander Schrijver. Vertex-disjoint simple paths of given homotopy in a planar graph. In *Polyhedral Combinatorics*, 1990.

[32] Greg N. Frederickson. Planar graph decomposition and all pairs shortest paths. *J. ACM*, 38(1):162–204, January 1991.

[33] Qian-Ping Gu and Hisao Tamaki. Improved bounds on the planar branchwidth with respect to the largest grid minor size. *Algorithmica*, 64(3):416–453, 2012.

[34] Sylvain Guillemot. Fpt algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization*, 8(1):61 – 71, 2011. Parameterized Complexity of Discrete Optimization.

[35] David Hartvigsen. The planar multiterminal cut problem. *Discret. Appl. Math.*, 85(3):203–222, 1998.

[36] Te C Hu. Integer programming and network flows. Technical report, Wisconsin Univ. Madison, Dept. of Computer Sciences, 1969.

[37] Bart M. P. Jansen, Marcin Pilipczuk, and Erik Jan van Leeuwen. A deterministic polynomial kernel for odd cycle transversal and vertex multiway cut in planar graphs. In Rolf Niedermeier and Christophe Paul, editors, *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*, volume 126 of *LIPIcs*, pages 39:1–39:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[38] David R. Karger, Philip Klein, Cliff Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, STOC '99, page 668–678, New York, NY, USA, 1999. ACM.

[39] Sándor Kisfaludi-Bak, Jesper Nederlof, and Erik Jan van Leeuwen. Nearly ETH-tight algorithms for planar Steiner tree with terminals on few faces. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1015–1034. SIAM, 2019.

[40] Philip N. Klein and Dániel Marx. Solving planar k-terminal cut in $O(n^{c\sqrt{k}})$ time. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 569–580, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[41] Robert Krauthgamer, James R. Lee, and Havana Rika. Flow-cut gaps and face covers in planar graphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 525–534. SIAM, 2019.

[42] Robert Krauthgamer and Havana Inbal Rika. Refined vertex sparsifiers of planar graphs. *SIAM Journal on Discrete Mathematics*, 34(1):101–129, 2020.

[43] Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1977.

[44] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[45] Dániel Marx. A tight lower bound for planar multiway cut with fixed number of terminals. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 677–688, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[46] Dániel Marx and Michał Pilipczuk. Optimal parameterized algorithms for planar facility location problems using voronoi diagrams. In Nikhil Bansal and Irene Finocchi, editors, *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, volume 9294 of *Lecture Notes in Computer Science*, pages 865–877. Springer, 2015.

[47] K. Matsumoto, Takao Nishizeki, and N. Saito. An efficient algorithm for finding multicommodity flows in planar networks. *SIAM J. Comput.*, 14:289–302, 1985.

[48] James K. Park and Cynthia A. Phillips. Finding minimum-quotient cuts in planar graphs. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 766–775, New York, NY, USA, 1993. ACM.

[49] Marcin Pilipczuk, Michał Pilipczuk, Piotr Sankowski, and Erik Jan van Leeuwen. Network sparsification for Steiner problems on planar and bounded-genus graphs. *ACM Trans. Algorithms*, 14(4):53:1–53:73, 2018.

[50] Michał Pilipczuk, Erik Jan van Leeuwen, and Andreas Wiese. Quasi-polynomial time approximation schemes for packing and covering problems in planar graphs. *Algorithmica*, 82(6):1703–1739, 2020.

[51] John H. Reif. Minimum s-t cut of a planar undirected network in $O(nlog^2(n))$ time. *SIAM J. Comput.*, 12(1):71–81, 1983.

[52] Paul D. Seymour and Robin Thomas. Call routing and the ratcatcher. *Comb.*, 14(2):217–241, 1994.

[53] Robert Endre Tarjan. A note on finding the bridges of a graph. *Inf. Process. Lett.*, 2(6):160–161, 1974.

[54] Mingyu Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theory of Computing Systems*, 46:723–736, 2009.

[55] Wei-Chang Yeh. A simple algorithm for the planar multiway cut problem. *J. Algorithms*, 39(1):68–77, 2001.