Time Space Optimal Algorithm for Computing Separators in Bounded Genus Graphs

Chetan Gupta
Aalto University, Finland

Rahul Jain
Fernuniversität in Hagen, Germany

Raghunath Tewari
Indian Institute of Technology Kanpur, India

Abstract

A graph separator is a subset of vertices of a graph whose removal divides the graph into small components. Computing small graph separators for various classes of graphs is an important computational task. In this paper, we present a polynomial-time algorithm that uses $O(g^{1/2}n^{1/2}\log n)$-space to find an $O(g^{1/2}n^{1/2})$-sized separator of a graph having n vertices and embedded on an orientable surface of genus g.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph algorithms

Keywords and phrases Graph algorithms, space-bounded algorithms, surface embedded graphs, reachability, Euler genus, algorithmic graph theory, computational complexity theory

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.23

Related Version *Extended Version*: https://arxiv.org/abs/2005.06419

Funding *Chetan Gupta*: Academy of Finland Grant 321901 and Visvesvaraya PhD Grant.
Raghunath Tewari: DST Inspire Faculty Grant and Visvesvaraya Young Faculty Fellowship.

1 Introduction

Graph separator is a valuable tool in designing divide and conquer based algorithms for various graph problems. In a graph, a separator is a small set of vertices of the graph whose removal divides the graph into pieces such that the size of each piece is at most a fraction of the original graph. Lipton and Tarjan's pioneering result showed that there exists a separator of size $O(n^{1/2})$ in planar graphs [12]. Subsequently, this separator was used to design many algorithms to solve various problems in planar graphs.

Recently, researchers have been interested in designing memory-constrained algorithms for various graph problems. They aim to optimize the space required by the algorithm while maintaining the polynomial time-bound. Graph separators have been used in designing memory-constrained algorithms for the *reachability* problem. Imai et al. and Ashida et al. presented polynomial-time algorithms that use $O(n^{1/2}\log n)$ space to find a separator of size $O(n^{1/2})$ in a planar graph [9, 3]. Imai et al. also gave a memory-constrained algorithm to solve the reachability problem using this separator [9]. A natural extension of planar graphs is the set of graphs that we can embed on a surface of constant *genus*. For such graphs, we know that a separator of size $O(n^{1/2})$ exists. Chakraborty et al. gave a polynomial-time algorithm which uses $O(n^{2/3}\log n)$ space to construct a separator of size $O(n^{2/3})$ in constant genus graphs [4].

Jain and Tewari formalized the connection between separators in a class of undirected graphs and the reachability problem in the class of directed versions of those graphs [10]. They mainly show that if there exists a polynomial-time algorithm that uses $O(w \log n)$-space to find a separator of size $O(w)$, then there exists a polynomial-time algorithm that uses $O(w \log n)$ space to solve reachability as well.

In this paper, we continue along the above line of work and present a polynomial-time algorithm that uses $O(g^{1/2}n^{1/2} \log n)$ space to construct a separator of size $O(g^{1/2}n^{1/2})$ in a g-genus graph. Therefore, combining this construction with [10], we get a polynomial-time algorithm that uses $O(g^{1/2}n^{1/2} \log n)$ space to solve the reachability problem in g-genus graphs. Thus, for constant genus graphs, our approach gives a polynomial-time algorithm that uses $O(n^{1/2} \log n)$ space.

Our separator construction follows the standard paradigm used in previous constructions of separators for planar graphs and surface-embedded graphs. Hence some familiarity with earlier results such as those shown by Gazit and Miller [7], Koutis and Miller [11], Imai et al. [9], Ashida et al. [3], and Chakraborty et al. [4] is beneficial in understanding our construction. In particular, since our result is a generalization of Ashida et al. [3], we heavily borrow their framework.

Our Result

In this paper, we prove the following theorem.

▶ **Theorem 1.** *There exists a polynomial-time algorithm that takes as an input a graph G on n vertices along with its combinatorial embedding of genus g and outputs its separator of size $O(g^{1/2}n^{1/2})$. This algorithm uses $O(g^{1/2}n^{1/2} \log n)$ space.*

The running time of our algorithm is a polynomial in *both* the number of vertices n and the value of the genus g. To achieve the desired space-time bounds, given a graph G, we first find a maximal set of vertices in G whose *k-neighbourhoods* do not intersect each other. We call the vertices in this set *Boss vertices*. We associate each vertex of the graph to one of the Boss vertices. We call the set of vertices associated with the same Boss vertex a *Voronoi region*. If a non-contractible cycle of length $O(k)$ in the graph spans at most two of these Voronoi regions, we find that cycle and remove it from the graph. Removal of such a non-contractible cycle from the graph reduces its genus by at least one. Otherwise, if there exists no such cycle, we proceed by dividing the original graph further into a total of at most $O(n/k + g)$ regions so that each of them is bounded by a simple cycle of length $O(\sqrt{k})$ and the number of vertices present inside each region is at most $n/3$. We then use these regions to construct a *Frame Graph*, which is the graph induced by the vertices on these cycles. We assign weights to each face of the frame graph such that the weight of a face is equal to the number of vertices of the original graph *inside* the corresponding cycle. While constructing the frame graph, we might encounter some properties of the original graph, which allows us to output either a separator or a non-contractible cycle of size $O(k)$. Thus, in the end, we either have a non-contractible cycle of size $O(k)$, a small separator or the frame graph. If the result is a separator, then we output that separator. If the result is a non-contractible cycle C, we store it and restart the algorithm with the graph $G \setminus C$ as the input. The final output will be the union of C with the separator of $G \setminus C$. If the result is the frame graph, we use the algorithm of Gilbert et al. [8] to find a separator. For an appropriate value of k, our algorithm achieves the desired time and space bound.

Comparison with the previous result

The construction of separator by Chakraborty et al. [4] proceeds (roughly) as follows: Given a graph G they first find a subgraph H of the graph G, such that removal of vertices of H from G makes the resulting graph $G \setminus H$ planar. Subsequently, they obtain a separator of $G \setminus H$ using the algorithm of Imai et al. [9] and add it to the vertices of H to get a separator of the graph G. The subgraph H obtained in Chakraborty et al. [4] might not be a connected subgraph of G. We find a smaller separator by first finding a 2-connected weighted subgraph of G such that a weight-separator of this weighted subgraph acts as a separator of the original graph. We find this 2-connected subgraph by using *Ridge edges* which was previously used by Gazit and Miller [7], and Ashida et al. [3] for the case of planar graphs. We show that by first efficiently removing non-contractible cycles from Voronoi regions, ridge edges can be used in graphs of a higher genus. We call this 2-connected subgraph a *Frame graph*. We then show that a weight-separator for this Frame graph can be found efficiently and hence get our result.

Organization of the Paper

The rest of the paper is organized as follows. In section 2, we give some preliminary notations and definitions. We first divide the graph into Voronoi regions. We explain this procedure in section 3. We further divide Voronoi regions by using pre-frame-loops in section 3.1. In section 4, we show how to process the pre-frame-loops and construct a *frame graph* and then make *floor* modifications and *ceiling* modifications in this frame graph. We thus get the required subgraph. Finally, in section 5, we put it all together to construct the separator.

2 Preliminaries

A graph is an ordered triple $G = (V(G), E(G), \partial)$ where $V(G)$ is the set of *vertices*, $E(G)$ is the set of *edges* and ∂ is a function that assigns to each edge a pair of vertices. Let p be a path. We use $\mathsf{first}(p)$ to denote the first edge of p and $\mathsf{last}(p)$ to denote the last edge of p. In an undirected graph G, it is helpful to regard each edge in E as a pair of directed edges, or *darts*. Each dart goes from one vertex, called its *tail*, to another vertex, called its *head*. For a dart e, we use $\mathsf{tail}(e)$ to denote the tail of the dart, and similarly, we use $\mathsf{head}(e)$ to denote the head of the dart. The two darts that results from a single undirected edge are said to be *reverse* of each other. If two darts e_1 and e_2 are reverse of each other, we denote e_2 by $\mathsf{rev}(e_1)$ and e_1 by $\mathsf{rev}(e_2)$.

The genus of a surface Σ is the maximum number of non-intersecting simple closed curves in Σ such that the surface remains connected after cutting along these curves. The genus of a graph G is the smallest g such that G can be embedded on a surface of genus g. A surface is called orientable if it has two distinct sides; else, it is called non-orientable. In this paper, we only consider graphs that can be embedded on an *orientable* surface. Let G be a graph embedded on a surface S of genus g. The *faces* of the embedding of G are the connected components of $S \setminus G$. If a face is homeomorphic to an open disk, it is called a 2-cell. If every face is homeomorphic to an open disk, the embedding is called a 2-cell embedding. A combinatorial embedding of G is defined as $\pi = \{\pi_v \mid v \in V(G)\}$ where for each vertex v, π_v is a cyclic permutation of darts whose tail is v. This permutation of darts goes clockwise as per the embedding on the surface. For a dart e, we use $\mathsf{left}(e)$ to denote the face which is on the left of e and $\mathsf{right}(e)$ to denote the face on the right of e. A *triangulated graph* is a graph that is embedded on a surface such that every face is a 2-cell and has three boundary edges.

We define a dual graph $\tilde{G}$ of G for an embedding in the following way: $\tilde{G}$ contains a vertex $\tilde{v}$ corresponding to every face of G and two vertices of $\tilde{G}$ have an edge between them if their corresponding faces share an edge in G. We say that an edge $\tilde{e}$ of $\tilde{G}$ crosses an edge e of G if the faces at the endpoints of $\tilde{e}$ shares the edge e of G.

Let G be a graph embedded on a surface of genus g. Let U be a subset of vertices of G, F be a subset of edges of G, and R be a subset of faces of G. Then $G[U]$ denotes the subgraph of G, induced by the vertices in the set U. Similarly, $G[F]$ denotes the subgraph of G, containing all the edges of F together with their endpoints. By $G[R]$, we denote the graph containing all the vertices and edges in the boundary of a face in R.

Let G be a graph of genus g. A set R of its faces is a *region* if $\tilde{G}[R]$ is connected. The set of edges of G whose only one side has a face in R is called the *boundary* of R.

If G is a graph embedded on a surface and c is a cycle in G, then we define the *left graph* and *right graph* of c as follows: If e is a dart of c followed by $e' = \pi^k_{\mathsf{head}(e)}(\mathsf{rev}(e))$, then all edges $\pi_{\mathsf{head}(e)}(\mathsf{rev}(e)), \pi^2_{\mathsf{head}(e)}(\mathsf{rev}(e)), \ldots, \pi^{k-1}_{\mathsf{head}(e)}(\mathsf{rev}(e))$ are said to be on the *left* side of c. An edge e'' which is not incident with c and which is connected by a path in $G \setminus c$ to an end of an edge of the left side of c is also said to be on the left side. Now the left graph of c is defined as the edges on the left side of c together with all their ends. The right graph G is defined analogously. We will often use the term *inside of* c to denote the left graph of c and *outside of* c to denote the right graph of c. We do not include the cycle c itself in either of these sides.

▶ **Definition 2.** *A cycle c of a surface embedded graph G is called a* contractible cycle *if and only if one of the sides of c is planar. A cycle that is not contractible is called a* non-contractible cycle.

We say that a set C of cycles satisfies the 3-path-condition if the following property holds: If u and v are vertices of G and P_1, P_2 and P_3 are internally vertex disjoint paths from u to v. If two of the cycles $C_{i,j} = P_i \cup P_j (1 \leq i < j \leq 3)$ are not in C then the third one is also not in C. It is a well known fact that the set of non-contractible cycles satisfies the 3-path-condition [15].

We define $\mathsf{dist}(u, v)$ to be the length of the shortest path between two vertices u and v. We introduce a total order (denoted by $<_v$) in the vertex set V of the graph based on the distance from v. For any vertices u and w, we say that u is nearer to v than w (written as $u <_v w$) if we have either

- $\mathsf{dist}(u, v) < \mathsf{dist}(w, v)$ or
- $\mathsf{dist}(u, v) = \mathsf{dist}(w, v)$ and u has a smaller index than w

For any set W of vertices of G, $\mathsf{nrst}_v(W)$ denotes a vertex u in W such that $u <_v w$ of all $w \in W \setminus \{u\}$. For any sets W and W' of vertices, we write $W <_v W'$ if $\mathsf{nrst}_v(W) <_v \mathsf{nrst}_v(W')$.

Let G be a graph of genus g. A closed loop c is a sequence of *distinct* darts $e_1, e_2, \ldots, e_m$ of G such that $\mathsf{head}(e_i) = \mathsf{tail}(e_{(i+1) \bmod m})$.

▶ **Definition 3.** *Let G be a weighted-graph with positive integral weights on each vertex that sums to n and $\alpha \in (0, 1)$. An α-separator of G is a set S of vertices of G such that the removal of S creates disconnected subgraphs, each of which has at most αn weight, where the weight of a subgraph is the sum of the weights of the vertices in it.*

We use a multitape Turing machine model to discuss the space-bounded polynomial-time algorithms. A multi-tape Turing machine consists of a read-only input tape, a write-only output tape, and a constant number of work tapes. We measure the space complexity of a multitape Turing machine by the total number of bits used in the work tapes.

We note that Allender and Mahajan [2] showed that the problem of testing whether a graph is planar or not is in SL. They also gave the SL algorithm to construct the planar embedding. Subsequently, Reingold [14] showed that SL = L hence there exists a logspace algorithm to test if a graph is planar and also produce its embedding. We summarize this fact in the following Lemma.

▶ **Lemma 4.** *There exists a logspace algorithm that tests whether the input graph is planar and if so, it outputs an embedding of the input graph.*

Gilbert, Hutchinson and Tarjan proved the existence of an $O(n^{1/2}g^{1/2})$ size separator for the graphs of genus g [8]. They also presented an $O(n+g)$ time algorithm to find the separator. Therefore we can conclude that their algorithm runs in $O((n+g)\log n)$ space. We can thus use the following Lemma for our result.

▶ **Lemma 5.** *There exists a polynomial-time algorithm that takes, as an input, an n-vertex graph of genus g along with its combinatorial embedding and finds its separator of size $O(n^{1/2}g^{1/2})$ using $O((n+g)\log n)$ space.*

We will need the notion of fundamental cycles in our separator construction; therefore, we define it formally.

▶ **Definition 6.** *Let G be a graph and T be a spanning tree of G. Let e be an edge that does not belong to T. A simple cycle c, which consists of e and the path in T joining the endpoints of e, is called a fundamental cycle.*

3 Voronoi Region

As we discussed in section 1, we start by dividing the input graph into something that we call *Voronoi regions*. In this section, we define the notion of Voronoi Regions and explain how they could be constructed in a space-efficient manner. This notion has been previously used in designing a separator for planar graphs by Imai et al., Ashida et al., Gazit and Miller, and Koutis and Miller [9, 3, 7, 11].

We first define the k-neighbourhood of a vertex. This is a key tool that will help us define and construct a Voronoi region.

▶ **Definition 7.** *Let G be a graph and v be a vertex of G. Let $L(v,i)$ be the set of vertices at distance i from v. The k-neighbourhood $N_k(v)$ of a vertex v is defined as:*

$$N_k(v) = \bigcup_{1 \le i \le d} L(v,i)$$

where d is the smallest integer such that $|\bigcup_{1\le i\le d} L(v,i)| \ge k$.

Note that we have defined k-neighbourhood in a slightly different way when compared to the definition of Imai et al. [9] and Chakraborty et al. [5]. In their work, $N_k(v)$ is chosen to contain at most k vertices, while here, it contains at least k vertices. We believe this definition makes our proof simpler to follow.

▶ **Definition 8.** *Let G be a graph. A set I of vertices of G is called a k-maximal independent set if the following holds:*

- *For every $b_1, b_2 \in I$, $N_k(b_1) \cap N_k(b_2) = \emptyset$.*
- *For every v that is not in I, we have a vertex $b \in I$ such that $N_k(v) \cap N_k(b) \neq \emptyset$.*

▶ **Lemma 9.** *There exists an $O((k+n/k)\log n)$-space and polynomial-time algorithm that takes a graph G as input and outputs a k-maximal independent set I.*

The proof of the above Lemma is quite straightforward. We refer readers to [9, 4].
For a graph G, we will use the notation $\mathsf{ind}(G)$ to denote the set returned by the algorithm of Lemma 9.

▶ **Definition 10.** *Let G be a graph. For any vertex v, the* boss-vertex *of v is a vertex b of $\mathsf{ind}(G)$ such that $N_k(b) <_v N_k(b')$, for all $b' \in \mathsf{ind}(G) \setminus \{b\}$. We define $\mathsf{vor}(b)$ to be the set of all vertices whose boss-vertex is b. We use $\mathsf{boss}(v)$ to denote the boss-vertex of v.*

Note that the graph induced by the vertices in the set $\mathsf{vor}(b)$ form a connected component in G. Therefore, the faces corresponding to these vertices form a region in $\tilde{G}$. We will henceforth call $\mathsf{vor}(b)$ the *Voronoi region* of b.

We note that while the Voronoi region of a vertex b can be large, its diameter is $O(k)$. We will now show that the BFS-tree of this Voronoi region can still be constructed in $O((n/k+k)\log n)$ space and polynomial time. To construct a rooted tree using small space, it suffices to show an algorithm to determine the parent of a given vertex $v \in \mathsf{vor}(b)$ in the BFS tree. The algorithm is the following: To determine the parent of a vertex v in $\mathsf{vor}(b)$, we first construct the BFS-tree T of k-neighbourhood of b. If $v \in N_k(b)$ then the parent of v is the same as its parent in T. Otherwise, construct the BFS-tree of $N_k(v)$. Let v' be the vertex in $N_k(v) \cap N_k(b)$ such that $\mathsf{dist}(b, v')$ is minimum. Break ties by picking the one with a smaller index. Consider the path from v to v' in the BFS tree of $N_k(v)$. The parent of v is the vertex adjacent to v in this path. We summarize in the following Lemma.

▶ **Lemma 11.** *Let G be a graph and b be a vertex of $\mathsf{ind}(G)$. There exists a polynomial time algorithm that constructs the BFS-tree of $\mathsf{vor}(b)$ in $O((k+n/k)\log n)$ space.*

A similar lemma was observed for planar graphs by Imai et al. [9]

The input graph might contain small non-contractible cycles. We require that the union of any two Voronoi regions do not have a non-contractible cycle, similarly as Chakraborty et al. [4]. Thus, we remove such non-contractible cycles from the graph using the following Lemma in our main algorithm.

▶ **Lemma 12** ([4])**.** *There is an $O((k+n/k)\log n)$-space and polynomial time algorithm that takes a graph G, and two boss-vertices b_1 and b_2 as input and checks for a non-contractible cycle of size $O(k)$ in $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$. The algorithm outputs one such cycle if it exists.*

Proof. First, consider the case when $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$ forms a connected subgraph of G. We know that $\mathsf{vor}(b_1)$ and $\mathsf{vor}(b_2)$ can be computed in $O((k+n/k)\log n)$ space and polynomial time by Lemma 11. We combine the BFS-trees of $\mathsf{vor}(b_1)$ and $\mathsf{vor}(b_2)$ using an arbitrary edge to get a spanning tree of $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$ with diameter $O(k)$. We denote this spanning tree as T. Note that T can be computed in polynomial-time and $O((n/k+k)\log n)$ space. We know that the set of all non-contractible cycles of any graph G satisfy 3-path condition [15]. Since the diameter of T is $O(k)$, any fundamental cycle of this tree of size $O(k)$. The 3-path condition implies that if a non-contractible cycle exists, then one of the fundamental cycles is non-contractible (see Allender et al. 2005, Lemma 5.1 [1]). We can check whether a cycle is contractible by checking the planarity of the left and the right sides of the cycle. The left and the right side of a given cycle can be computed in logspace by using reachability queries [14]. Therefore, by Lemma 4, we can check if a cycle is contractable in $O(\log n)$ space. Thus, the lemma follows. In the other case where $\mathsf{vor}(b_1)$ and $\mathsf{vor}(b_1)$ are not connected, we can apply the same procedure on spanning trees of $\mathsf{vor}(b_1)$ and $\mathsf{vor}(b_2)$ separately. ◀

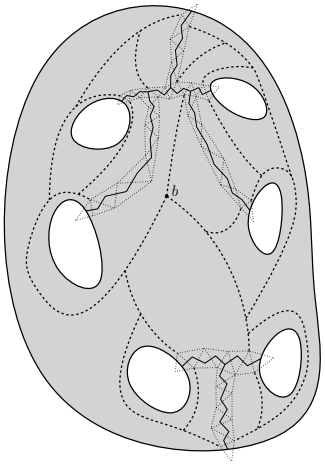

Figure 1 A diagram showing vor(b) for a boss vertex b. The part of the surface where the vertices of vor(b) are present is shown in grey colour. The boundary of the Voronoi region is shown using thick solid lines. The ridge edges are shown using normal solid lines. Dashed lines show some of the edges of the spanning tree of vor(b). Dotted lines show faces in G that corresponds to a vertex v, which is an endpoint of a ridge edge in $\tilde{G}$.

As mentioned in the introduction, we will use the Voronoi regions to construct our Frame graph. For this construction, we first divide the Voronoi Regions.

3.1 Dividing Voronoi Regions using Pre-Frame-Loops

In this subsection, we find a set of loops in the input graph G. Each of these loops contains vertices of at most two Voronoi regions *inside* them. We then further process these loops so that the number of vertices inside them is small.

Let G be a triangulated graph of genus g. Note that any connected component of G forms a region in $\tilde{G}$. Also, note that since the size of each face of G is three, all the vertices of the graph $\tilde{G}$ will have degree three. Thus, a region of faces in $\tilde{G}$ will have a boundary that is a set of vertex-disjoint simple cycles.

We require two kinds of edges in the dual graph to construct the desired loops. One is the set of the boundary edges of all the Voronoi regions, and the other is the set of *Ridge edges*. Ridge edges have been used previously by Gazit and Miller [7] and Ashida et al. [3]. We define them as follows.

▶ **Definition 13.** *Let G be a graph and b be a boss-vertex. Let T be the BFS tree of* vor(b). *For an edge e of G[*vor(b)*] that does not belong to T, let c_e be the fundamental cycle induced by e on T. If each of the two sides of the cycle c_e contains at least one boundary cycle of* vor(b), *then the edge $\tilde{e}$ of $\tilde{G}$ crossing e is called a ridge edge.*

Figure 1 shows ridge edges in the Voronoi region of a boss vertex b.

▶ **Definition 14.** *Let G be a graph of genus g. Let $\tilde{B}$ be the set of boundary edges of* vor(b) *for all boss vertices b. Similarly, let $\tilde{R}$ be the set of ridge-edges. A* branch vertex *is a degree three vertex in the graph $\tilde{G}[\tilde{B} \cup \tilde{R}]$. For each branch vertex $\tilde{v}$, the boundary of the face consisting of three vertices incident to $\tilde{v}$ is a* branch-triangle. *Two branch vertices are called* adjacent *to each other if a path connects them consists of darts corresponding to the edges in the set $\tilde{B} \cup \tilde{R}$ such that no other branch vertex exists on this path. The path connecting adjacent branch vertices is called a* connector. *We denote the set of connectors in $\tilde{G}$ by* con(G).

Let $\tilde{p}$ be a connector. Note that the end points of $\tilde{p}$ are an adjacent pair of branch vertices. Also note that, $\mathsf{boss}(\mathsf{left}(\mathsf{first}(p)))$ is same as $\mathsf{boss}(\mathsf{left}(\mathsf{last}(p)))$ and $\mathsf{boss}(\mathsf{right}(\mathsf{first}(p)))$ is same as $\mathsf{boss}(\mathsf{right}(\mathsf{last}(p)))$. We define a pre-frame-loop with respect to $\tilde{p}$ as follow.

▶ **Definition 15.** *Let G be a graph embedded on a surface of genus g. For any connector $\tilde{p}$, a pre-frame-loop (denoted by $\mathsf{pfloop}(\tilde{p})$) is a closed loop that consists of*

1. *A path from $\mathsf{right}(\mathsf{first}(p))$ to $\mathsf{boss}(\mathsf{right}(\mathsf{first}(p)))$ in the BFS-tree of $\mathsf{vor}(\mathsf{boss}(\mathsf{right}(\mathsf{first}(p))))$*
2. *A path from $\mathsf{boss}(\mathsf{right}(\mathsf{first}(p)))$ to $\mathsf{right}(\mathsf{last}(p))$ in the BFS-tree of $\mathsf{vor}(\mathsf{boss}(\mathsf{right}(\mathsf{first}(p))))$*
3. *A branch-triangle dart e_{last} from $\mathsf{right}(\mathsf{last}(p))$ to $\mathsf{left}(\mathsf{last}(p))$*
4. *A path from $\mathsf{left}(\mathsf{last}(p))$ to $\mathsf{boss}(\mathsf{left}(\mathsf{last}(p)))$ in the BFS-tree of $\mathsf{vor}(\mathsf{boss}(\mathsf{left}(\mathsf{last}(p))))$.*
5. *A path from $\mathsf{boss}(\mathsf{left}(\mathsf{last}(p)))$ to $\mathsf{left}(\mathsf{first}(p))$ in the BFS-tree of $\mathsf{vor}(\mathsf{boss}(\mathsf{left}(\mathsf{last}(p))))$.*
6. *A branch-triangle dart e_{fst} from $\mathsf{left}(\mathsf{first}(p))$ to $\mathsf{right}(\mathsf{first}(p))$.*

We denote the set of all the pre-frame-loop in G as $\mathsf{pfloop}(G)$

The proof of the following lemma is straightforward.

▶ **Lemma 16.** *There exists an $O((k + n/k) \log n)$-space and polynomial-time algorithm that takes G as an input and outputs the list $\mathsf{pfloop}(G)$ of all pre-frame-loops in G.*

In the next section, we will use these pre-frame loops to create faces of our subgraph. Following Ashida et al. [3], we call this new graph Frame Graph.

4 Frame Graph

We wish to use pre-frame-loops to create faces of the frame graph. In order to do this, we first preprocess these loops so that the *inside* of each loop is small, i.e., has at most $n/3$ vertices in it. This preprocessing would ensure that the weight on any face of the frame graph is bounded. In the second step, we remove those edges of the loop for which both of its darts are traversed and thus break the loop into simple cycles. These cycles will act as boundaries of the faces in the frame graph.

Consider a connector $\tilde{p}$ of the input graph G and the pre-frame-loop c induced by $\tilde{p}$. Note that c is in the union of two Voronoi regions. Since we have eliminated all non-contractible cycles from the union of any two Voronoi regions, c cannot contain a non-contractible cycle. Thus, c divides the surface. A pre-frame loop is of type A if it consists of two boss vertices, and it is of type B if it consists of only one boss vertex (see Figure 2).

Let the part of a connector $\tilde{p}$ excluding its first and last vertex be called the *body* of $\tilde{p}$. Let P_0 denote the surface of $G \setminus c$ that has the body of the connector $\tilde{p}$. Call this *inside* of c. Let n_0 be the number of vertices in P_0 not including the vertices of c. We say that the inside of c is *large* if n_0 is greater than $2n/3$. Note that the inside of c is included in the union of atmost two Voronoi regions. Let b_1 and b_2 be the boss vertices of these two regions. We use the BFS-Trees of $\mathsf{vor}(b_1)$ and $\mathsf{vor}(b_2)$ to find a spanning tree of $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$. Since $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$ does not have a non-contractible cycle, it has a planar embedding. Consider a spanning tree T of $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$. There exists a fundamental cycle of this tree in the triangulated version of the graph $G[\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)]$ which acts as its separator [12]. Since by removing the boundary of the pre-frame loop from the graph, we can get components, the largest of which is formed by the vertices $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$, we can combine the separator of $G[\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)]$ with the boundary of the pre-frame-loop to get a separator of the whole graph G. Since the length of the boundary of pre-frame-loop is $O(k)$ and the diameter of Voronoi region of any boss vertex is $O(k)$, we get the following lemma:

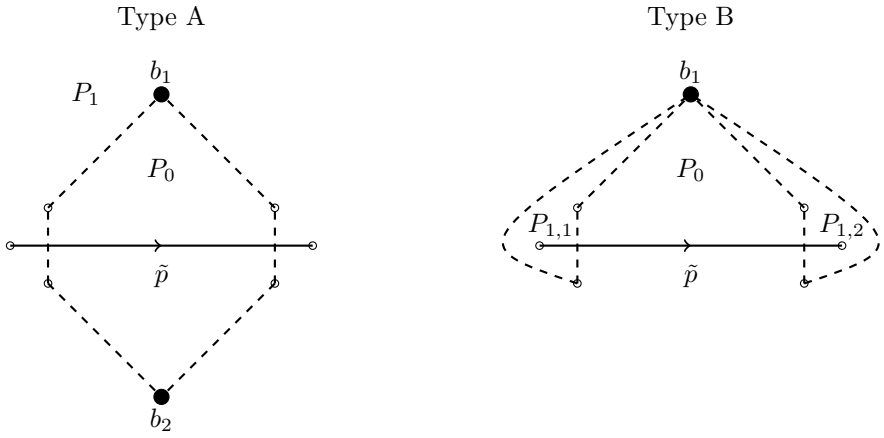

Figure 2 On the left, a pre-frame-loop of type A. The two boss vertices corresponding to the loop are b_1 and b_2. On the right, a pre-frame-loop of type B. The only boss vertex corresponding to this loop is b_1.

▶ **Lemma 17.** *Let G be a graph of genus g which contains a pre-frame loop whose inside is large. There exists a polynomial-time algorithm that takes as an input G and outputs a separator of G of size $O(k)$ in $O((k+n/k)\log n)$ space.*

Thus, if any of the pre-frame-loop acts as a separator or has a large inside, we can get a separator of the graph G. Otherwise, we construct a set C in the following way: We first add all the pre-frame-loop of type A into C. Note that if a pre-frame-loop c is of type B, it divides the surface into three parts. Call the two parts of the surface, which does not contain the body of the connector, $P_{1,1}$ and $P_{1,2}$ respectively. Let the number of vertices in P_0, $P_{1,1}$ and $P_{1,2}$ be n_0, $n_{1,1}$ and $n_{1,2}$ respectively. We see that either $n_{1,1} > 2n/3$ or $n_{1,2} > 2n/3$, for otherwise, our pre-frame-loop acts as a separator. Let us assume, without loss of generality, that $n_{1,2} > 2n/3$. We merge P_0 and $P_{1,1}$ into a single surface, and add the loop c_0 bounding this surface to the set C. The *inside* of c_0 is the side containing the surfaces P_0 and $P_{1,1}$.

Now, consider a loop c of C, that is not contained in the inside of any other loop c of C. Let E_c be the set of darts whose reverse does not appear in c. Let E be the union of E_c over all such c. We observe that E is a set of simple cycles, which we call frame-cycles and denote by $\mathsf{fcycle}(G)$.

4.1 Definition and construction of Frame Graph

▶ **Definition 18.** *Let G be a graph of genus g. Let E_1 be the set of all frame-cycles edges, and let E_2 be the set of all branch-triangle edges. A* frame-graph *of G is a subgraph $H = G[E_1 \cup E_2]$. For each face of a frame-graph H, its weight is the number of vertices of G located inside that face. We denote the frame graph of G by $\mathsf{frame}(G)$.*

▶ **Definition 19.** *Let G be a triangulated graph. Let $L(v,i)$ be the set of vertices at distance i from v. Let $d_{nb}(v)$ be the largest d such that $|\cup_{0\le i\le d} L(v,i)| < k$. For any boss-vertex $b \in \mathsf{ind}(G)$, let $d_{core}(b)$ denote the largest $d \le d_{nb}(b)$ such that $|L(b,d)| \le k^{1/2}$. The* core *of b (denoted by $\mathsf{core}(b)$) is defined by*

$$\mathsf{core}(b) = \bigcup_{0\le i\le d_{core}(b)} L(b,i)$$

Note that core(b) forms a region in $\tilde{G}$. The boundary of this region might not be a single cycle. In the next definition, we pick one of these cycles to be the core boundary-cycle and use it to construct the core cycle in the graph G.

▶ **Definition 20.** *Let G be a triangulated graph of genus g and b be a boss-vertex. The* core-boundary-cycle *of b is the boundary cycle of the region $\mathsf{core}(b)$ in $\tilde{G}$ that has the largest number of dual-vertices on its outside.*

The core-cycle *of $\mathsf{core}(b)$ is a directed cycle induced by the set of vertices in $\mathsf{core}(b)$ sharing an edge with the core boundary cycle. The inside of the core-cycle is the side with the boss-vertex b.*

For any $l \geq 1$, let $L_{nb}(l)$ denote a set of vertices v of G whose distance from its nearest k-neighborhood in $\{N_k(b)\}_{b \in \mathsf{ind}(G)}$ is l. More formally,

$$L_{nb}(l) = \{v \mid \mathsf{dist}(v, v_{\mathsf{nrst}}) = l, \text{where } v_{\mathsf{nrst}} = \mathsf{nrst}_v(N_k(\mathsf{boss}(v)))\}.$$

Let $\tilde{L}_{nb}(l)$ denotes the set of faces in $\tilde{G}$ corresponding to the vertices in the set $L_{nb}(l)$. Let C be a region of $\tilde{L}_{nb}(l)$. Each boundary edge of C is an edge between a pair of vertices of level either $l-1$ and l or l and $l+1$. Let us call the former one an *interior* edge and the latter one an *exterior* edge. We call a boundary cycle an *interior boundary cycle* if it consists of interior edges. Similarly, we call a boundary cycle an *exterior boundary cycle* if it consists of exterior edges.

▶ **Definition 21.** *Let $\tilde{c}$ be any interior boundary cycle corresponding to $L_{nb}(l)$. Let c be the loop in C formed by the set of vertices sharing a boundary edge with $\tilde{c}$, and let D_c be the set of cycles obtained from c by removing all the darts in the loop whose reverse also appears in the loop. An interior-cycle is a cycle in D_c.*

We define an *exterior-cycle* in a similar way. A cycle is said to be a *small* cycle if it consists of at most $k^{1/2}$ vertices. We denote the set of small interior cycles by $\mathsf{smint}(G)$ and the set of small exterior cycles by $\mathsf{smext}(G)$. A contractible cycle is said to be *light* if it has less than $n/3$ vertices in its inside.

▶ **Definition 22.** *A floor cycle is a light and small interior cycle if it is not inside any other light and small interior cycle. For any boss-vertex b which is not contained in any floor-cycle, we regard the core-cycle of $\mathsf{core}(b)$ also as a floor-cycle. A ceiling-cycle is a light and small exterior cycle that is not inside any other light and small exterior-cycle, and that has at least one dual-vertex of some branch-triangle on its inside.*

▶ **Definition 23.** *Let G be a graph of genus g. Let F and C be respectively a set of floor-cycles and ceiling-cycles having at least one vertex of $\mathsf{frame}(G)$ in their insides. Let E'_1 be the set of edges of G that appear in some cycle in $F \cup C$ and E'_2 be the set of edges of $\mathsf{frame}(G)$ that are not in the inside of any cycle of $F \cup C$. A graph with vertices U' and edges D' is a modified frame-graph denoted as $\mathsf{mframe}(G)$, where $D' = E'_1 \cup E'_2$ and U' is the set of all vertices that are endpoints of edges of D'. For each face of a modified frame-graph $\mathsf{mframe}(G)$, its weight is the number of vertices of G located in the face.*

The following Lemma is a generalization of a result that was presented by Ashida et al. [3]. They presented a similar lemma for planar graphs. The proof of the following Lemma has been moved to Appendix.

▶ **Lemma 24.** *Let G be a graph of genus g such that voronoi region $\mathsf{vor}(b_1) \cup \mathsf{vor}(b_2)$ does not contain a non-contractible cycle for any two vertices $b_1, b_2 \in \mathsf{ind}(G)$, $\mathsf{pfloop}(\tilde{p})$ is not a separator of G for any connector $\tilde{p}$, the inside of any loop in $\mathsf{pfloop}(\tilde{p})$ is not large, $\mathsf{core}(b)$ is not a separator of G for any boss-vertex b, and no cycle in $\mathsf{smext}(G)$ or $\mathsf{smint}(G)$ is a non-contractible cycle. Following statements hold:*

1. *The weight of each face of* $\mathsf{mframe}(G)$ *is less than* $n/3$.
2. $\mathsf{mframe}(G)$ *is* 2*-connected.*
3. *Size of each face of* $\mathsf{mframe}(G)$ *is* $O(k^{1/2})$.
4. *The number of faces in* $\mathsf{mframe}(G)$ *is* $O(n/k+g)$

5 Construction of separator

Using the tools developed so far, we can obtain a space-efficient algorithm which, given a graph G as input, outputs either a separator, a non-contractible cycle or the modified frame graph $\mathsf{mframe}(G)$. We summarize this in the following Lemma.

▶ **Lemma 25.** *Let* G *be a* g*-genus triangulated graph of* n *vertices. For any positive integer* k*, there is a polynomial time,* $O((n/k+k)\log n)$*-space algorithm that takes* G *along with its combinatorial embedding as input and outputs one of the following:*

1. *A non-contractible cycle of size* $O(k)$ *of* G.
2. *A separator of size* $O(k)$ *of* G.
3. *A a weighted subgraph* H' *of* G *that satisfies the following conditions:*
   a. *The weight of each face* f *of* H' *is proportional to the number* n_f *of vertices of* G *located inside the face, and is less than* $n/3$
   b. H' *is 2-connected.*
   c. H' *contains* $O(n/k+g)$ *faces.*
   d. *The size of each face of* H' *is* $O(k^{1/2})$.

Proof. We first find a k-maximal independent set $\mathsf{ind}(G)$ of G in $O((n/k+k)\log n)$-space and polynomial time using lemma 9. We then check for a non-contractible cycle in $\mathsf{vor}(b_i)\cup\mathsf{vor}(b_j)$ for all pairs of boss-vertices b_i and b_j using lemma 12. If we manage to find such a cycle, we output it. Otherwise, we pick each pre-frame loops using lemma 16 see if it acts as a separator of the graph. If so, we output it.

If the algorithm has not produced an output so far, we see if the inside of any pre-frame-loop is large. If so, we use lemma 17 to find a separator of the graph. For every boss vertex b, we check if $\mathsf{core}(b)$ is a separator. If so, we output it. Next, we check if any cycle in $\mathsf{smext}(G)$ or $\mathsf{smint}(G)$ is a non-contractible cycle. If so, we output it. Otherwise, we output the modified frame graph $\mathsf{mframe}(G)$. ◀

With these ingredients, we are now ready to prove our main theorem.

Proof of Theorem 1. Elberfeld and Kawarabayashi presented an algorithm to construct a combinatorial embedding of a graph of the constant genus in logspace [6]. Hence, we do not require a combinatorial embedding as part of the input when dealing with a constant-genus graph. Otherwise, we require the combinatorial embedding of the graph as an input. We assume that the genus of the input graph g is at most $O(n)$. Let π be the combinatorial embedding of G. We first triangulate the input graph in logspace. To do this, for each face f of the input graph, we connect each vertex of f with the lowest index vertex in it. This triangulation is done implicitly, whenever required, as storing the triangulated graph will require a large amount of space. We call the resultant triangulated graph G. Note that triangulating the graph only introduces more edges; therefore, a separator for G will also be a separator for the input graph. Our objective now is to construct a separator of G. We do this by iteratively applying Lemma 25. We will describe the algorithm by describing an iteration of it. Before the ith iteration, we will have a set S of vertices which is empty before the first iteration. Let $G_1, G_2, \ldots, G_m$ be the set of connected components in $G \setminus S$. We will

describe the ith iteration as follows. The algorithm takes the component G_j whose size n_j is greater than $2n/3$. If no such component exists, then the set S would be a separator of G, and the algorithm outputs S and halts. Otherwise, consider the embedding induced by π on G_j as its embedding. If the genus g_j of this component is zero, the algorithm uses Imai et al. planar separator algorithm to get its separator S_1 and outputs $S \cup S_1$. If its genus is non-zero we apply the algorithm from Lemma 25 on G_j with k set as $n_j^{1/2}/g_j^{1/2}$. If the result of the application of the algorithm from Lemma 25 on G_j is a non-contractible cycle, say S_2, then we add the vertices of S_2 to the set S and continue with the next iteration. If the result is a separator, say S_3, we output the set $S \cup S_3$ as the separator for the entire graph. Otherwise, if the result is a subgraph H' of G_j, we take its dual $\tilde{H}'$ and find its separator $\tilde{S}'$ using Lemma 5. $\tilde{S}'$ is a set of faces of H'. Consider the set S_4 of vertices on the boundary of these faces. We return the set $S \cup S_4$. To see that the size of separator returned by the above algorithm is $O(g^{1/2}n^{1/2})$, note that Lemma 25 returns a non-contractible cycle, the genus of the graph is reduced by at least one. It is sufficient for us to use the induced embeddings (see Mohar and Thomassen [13], Proposition 4.2.1 and Lemma 4.2.4). Hence, our algorithm can return at most g such cycles; each has length $O(k)$. For our value of k, the total number of vertices in all such cycles can be at most $O(g^{1/2}n^{1/2})$. If it does not returns a non-contractible cycle, then it returns either a separator of size $O(k) \leq O(g^{1/2}n^{1/2})$ or it returns the subgraph H'. The number of faces in H' is $O(n/k + g)$. Hence the size of the separator returned by using the algorithm of Gilbert et al. [8] on the dual of H' will be $O(g^{1/2}(n/k + g)^{1/2})$. Size of each face of H' is at most $k^{1/2}$, hence, size of the set S_4 is $O(k^{1/2}g^{1/2}(n/k + g)^{1/2})$. For our value of k, this is at most $O(g^{1/2}n^{1/2})$. ◀

We can use the following Lemma, which was formalized by Jain and Tewari [10] to get a space-efficient polynomial-time algorithm for reachability in constant-genus graphs. Reachability is determining if there is a directed path from one vertex to another in a directed graph.

▶ **Lemma 26.** *Let $\mathcal{G}$ be a class of graphs and $w : \mathcal{N} \mapsto \mathcal{N}$ be a function. If there exist a polynomial-time algorithm that uses $O(w(n) \log n)$ space to find a separator of size $w(n)$ then there exists a polynomial time algorithm to decide reachability in G that uses $O(w(n) \log n)$ space.*

▶ **Corollary 27.** *There exists a polynomial-time algorithm that uses $O(n^{1/2} \log n)$ space to solve reachability in a constant-genus graph.*

Previously, a polynomial-time algorithm that uses $O(n^{1/2} \log n)$ space for reachability was known for planar graphs [9]. While for constant-genus graphs, a polynomial-time algorithm that uses $O(n^{2/3} \log n)$ space was known [4]. Corollary 27 improves the space-bound to $O(n^{1/2} \log n)$. Our result can thus be seen as both a generalization of Imai et al. [9] and as an improvement to a previous result by Chakraborty et al. [5].

References

1 Eric Allender, Samir Datta, and Sambuddha Roy. The directed planar reachability problem. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, pages 238–249, 2005. doi:10.1007/11590156_19.

2 Eric Allender and Meena Mahajan. The complexity of planarity testing. *Information and Computation*, 189(1):117–134, 2004. doi:10.1016/j.ic.2003.09.002.

3 Ryo Ashida, Tomoaki Imai, Kotaro Nakagawa, A. Pavan, N. V. Vinodchandran, and Osamu Watanabe. A sublinear-space and polynomial-time separator algorithm for planar graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:91, 2019.

4 Diptarka Chakraborty, Aduri Pavan, Raghunath Tewari, N. V. Vinodchandran, and Lin F. Yang. New time-space upperbounds for directed reachability in high-genus and h-minor-free graphs. In *Proceedings of the 34th Annual Conference on Foundation of Software Technology and Theoretical Computer Science (FSTTCS 2014)*, pages 585–595, 2014.

5 Diptarka Chakraborty and Raghunath Tewari. An $O(n^{\epsilon})$ space and polynomial time algorithm for reachability in directed layered planar graphs. *ACM Transactions on Computation Theory (TOCT)*, 9(4):19:1–19:11, 2017.

6 Michael Elberfeld and Ken-ichi Kawarabayashi. Embedding and canonizing graphs of bounded genus in logspace. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 383–392. ACM, 2014. doi:10.1145/2591796.2591865.

7 H. Gazit and G. L. Miller. A parallel algorithm for finding a separator in planar graphs. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science (FOCS 1987)*, pages 238–248, October 1987. doi:10.1109/SFCS.1987.3.

8 John R Gilbert, Joan P Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, 1984. doi:10.1016/0196-6774(84)90019-1.

9 Tatsuya Imai, Kotaro Nakagawa, Aduri Pavan, N. V. Vinodchandran, and Osamu Watanabe. An $O(n^{\frac{1}{2}+\epsilon})$-space and polynomial-time algorithm for directed planar reachability. In *Proceedings of the 28th Conference on Computational Complexity (CCC 2013)*, pages 277–286, 2013.

10 Rahul Jain and Raghunath Tewari. Reachability in High Treewidth Graphs. In *Proceedings of the 30th International Symposium on Algorithms and Computation (ISAAC 2019)*, 2019.

11 Ioannis Koutis and Gary L. Miller. A linear work, $O(n^{1/6})$ time, parallel algorithm for solving planar laplacians. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, pages 1002–1011, 2007.

12 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

13 B. Mohar and C. Thomassen. *Graphs on Surfaces*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2001. URL: https://books.google.com.sg/books?id=_VFKscYKSicC.

14 Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM (JACM)*, 55(4):17, 2008.

15 Carsten Thomassen. Embeddings of graphs with no short noncontractible cycles. *Journal of Combinatorial Theory, Series B*, 48(2):155–177, 1990. doi:10.1016/0095-8956(90)90115-G.

A Appendix

A.1 Proof of Lemma 24

Proof. We will prove each of the four statements of the proof in order.

1. Consider a face of the frame-graph $\mathsf{frame}(G)$. The boundary of this face is either a frame-cycle or a branch triangle. The weight of a branch-triangle is zero, while the number of vertices inside a frame-cycle is less than $n/3$ by construction. Hence the weight of any face of $\mathsf{frame}(G)$ is less than $n/3$. The number of vertices inside a floor or a ceiling cycle is less than $n/3$ by definition. Hence the weight of any face of $\mathsf{mframe}(G)$ is also less than $n/3$.
2. We first prove that $\mathsf{frame}(G)$ is 2-connected. Let u and v be two distinct vertices of $\mathsf{frame}(G)$. We have the following cases:

Case 1 (Both u and v are on same frame cycle in $\mathsf{pfloop}(G)$) : Since u and v are on a cycle, there exist two vertex-disjoint paths from u to v.

Case 2 (u and v are on two different cycles in $\mathsf{pfloop}(G)$) : Let c_u and c_v be the cycles of $\mathsf{pfloop}(G)$ which contain vertices u and v respectively. Let $\tilde{p}_u$ and $\tilde{p}_v$ be the connectors whose bodies are contained in c_u and c_v respectively. We first note that there is a sequence of connecters $\tilde{p}_u = \tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_k = \tilde{p}_v$ such that $\tilde{p}_i$ and $\tilde{p}_{i+1}$ has a common end point for each $i \in [1, k-1]$. Also note that the body of $\tilde{p}_i$ is contained in a cycle c_i of $\mathsf{pfloop}(G)$. Orient the darts of $\tilde{p}_i$ to form a path from $\mathsf{first}(p_1)$ to $\mathsf{last}(p)_k$. Let c_i^{left} be the path from $\mathsf{left}(\mathsf{first}(\tilde{p}_i))$ to $\mathsf{left}(\mathsf{last}(\tilde{p}_i))$. Similarly, let c_i^{right} be the path from $\mathsf{right}(\mathsf{first}(\tilde{p}_i))$ to $\mathsf{right}(\mathsf{last}(\tilde{p}_i))$. We see that c_i^{left} and c_i^{right} do not share any vertex. Thus, we can see that there exist two vertex-disjoint paths q_{left} and q_{right} from u to v such that q_{left} contains vertices of c_i^{left} and q_{right} contains vertices of c_i^{right} for all $i \in [2, k-1]$.

The analysis of other cases is similar.

We will show that there are two vertex-disjoint paths between any two vertices u and v of the graph $\mathsf{mframe}(G)$.

Case 1 (u and v are both in $\mathsf{frame}(G)$): Since we have proved that $\mathsf{frame}(G)$ is two connected, we know that there exist two vertex disjoint paths q_{left} and q_{right} between u and v in $\mathsf{frame}(G)$. Note that several floor-cycles and ceiling-cycles were added to $\mathsf{frame}(G)$ and the vertices inside them were removed in order to construct $\mathsf{mframe}(G)$. Let c be one such cycle.

- If c intersects both q_{left} and q_{right}. Let u_{left} and v_{left} denote the first and the last vertices of q_{left} which intersects c. Similarly, let u_{right} and v_{right} denote the first and the last vertices of q_{right} which intersects c. These four vertices divide c into four paths c_1, c_2, c_3 and c_4. Let the set of these four paths be C. Then one of the following statements is true:
  - There exist paths from u_{left} to v_{left} and from u_{right} to v_{right} in C.
  - There exist paths from u_{left} to v_{right} and from u_{right} to v_{left} in C.

  For both the above cases, we see that there exist two disjoint paths from u to v.
- If c intersects only one of the path q_{left} and q_{right} then we can modify that path to contain part of the cycle.

Case 2 (u and v are on different floor-cycles or ceiling-cycles c_u and c_v): Let w_u and w_v be vertices of $\mathsf{frame}(G)$ inside c_u and c_v respectively. We know such vertices exits because of the way these cycles are defined. Since the graph $\mathsf{frame}(G)$ is 2-connected, there exist two disjoint paths q_{left} and q_{right} between w_u and w_v in it. Let u_{left} be the last intersection of q_{left} and c_u. Similarly let u_{right} be the last intersection of q_{right} and c_u. Note that, since the paths q_{left} and q_{right} are disjoint, $u_{left} \neq u_{right}$. We similarly define v_{left} and v_{right}.

Since the three vertices u, u_{left} and u_{right} lie on the cycle c_u, there exists two disjoint paths: first from u to u_{left} and second from u to u_{right}. Similarly, there exists two disjoint paths from v_{right} to v and from v_{left} to v. We can thus get two vertex-disjoint paths from u to v, using these. Note that there may be other floor and ceiling cycles intersecting these disjoint paths. In that case, we can use an argument similar to above to show the existence of two disjoint paths from u to v.

3. We now prove that the size of each face of $\mathsf{mframe}(G)$ is $O(k^{1/2})$. Note that the boundary of a face of the graph $\mathsf{mframe}(G)$ is one of the following:
   a. A floor-cycle of G.

b. A ceiling-cycle of G.
c. A branch-triangle of G.
d. A frame-cycle of $\mathsf{frame}(G)$ modified by floor-cycles and ceiling-cycles.

In the first three cases, the size bound of the face follows by definition. We thus consider the fourth case.

Consider any face defined by a modified frame-cycle, and let c denote the pre-frame-loop from which we have defined it. Consider any path p of c connecting a boss-vertex of c and a vertex of a branch-triangle used in c such that the path does not contain any other vertex of a branch-triangle. By our modification, we can use a part p' of p that is in the outside of the corresponding floor-cycle and ceiling-cycle (if it exists) as a component of the modified frame-cycle, and its length is bounded by $4k^{1/2}$. Note that the floor-cycle may not be used in $\mathsf{mframe}(G)$ if it only intersects with the darts that we have removed for defining the face. In this case, however, only a part of p' is used for the modified frame-cycle, which is even shorter. Thus, the modified frame-cycle consists of at most four such reduced paths, a part of two floor-cycles, a part of four ceiling-cycles, and two edges from two branch-triangles, and their total length is $O(k^{1/2})$.

4. We first prove that the number of connectors is $O(n/k + g)$, and the number of branch vertices is $O(n/k + g)$. Since there is a one-to-one correspondence respectively between branch-triangles and branch vertices, and between frame-cycles and connectors, it will follow that the number of faces in $\mathsf{frame}(G)$ is $O(n/k + g)$.

We first define a new graph G'. The vertex set of G' is the set of branch vertices in G. We add an edge between two vertices of G' if they are adjacent pair of branch vertices. Since every edge of G' corresponds to a connector of G, the graph G' can also be embedded on the surface of genus g where the embedding corresponds to the embedding of G. Let n', e', f' be the number of vertices, edges and faces in G' respectively. Thus, we have $n' - e' + f' = 2 - 2g$ by Euler's formula. Note that every branch vertex have a degree 3, therefore we have $2e' = 3n'$. This implies $e' = 6g + 3f' - 6 = O(f' + g)$. Since, there is one-to-one correspondence between voronoi regions and the faces of G', we have $f' = O(n/k)$. Hence, we can conclude that $e' = O(n/k + g)$ and $n' = O(n/k + g)$.

Now, to prove that the number of faces in $\mathsf{mframe}(G)$ is $O(n/k + g)$, we see that the new faces introduced by our modification are those defined by floor cycles or ceiling cycles. By definition, the number of these cycles is at most the number of boss-vertices or that of branch-triangles, which is bounded by $O(n/k + g)$. Note that we can divide a face defined by a frame-cycle of $\mathsf{frame}(G)$ by ceiling-cycles, but it is easy to see that each face is divided into at most some constant number of faces because the number of floor-cycles and ceiling-cycles overlapping each frame-cycle is constant, say, at most six. From these observations, we can bound the number of faces of $\mathsf{mframe}(G)$ by $O(n/k + g)$. ◀