New Time-Space Upperbounds for Directed Reachability in High-genus and H-minor-free Graphs*

Diptarka Chakraborty[1], A. Pavan[2], Raghunath Tewari[3], N. V. Vinodchandran[4], and Lin Forrest Yang[5]

1 **Indian Institute of Technology, Kanpur.** `diptarka@cse.iitk.ac.in`
2 **Iowa State University.** `pavan@cs.iastate.edu`
3 **Indian Institute of Technology, Kanpur.** `rtewari@cse.iitk.ac.in`
4 **University of Nebraska–Lincoln.** `vinod@cse.unl.edu`
5 **Johns Hopkins University.** `lyang@pha.jhu.edu`

Abstract

We obtain the following new simultaneous time-space upper bounds for the directed reachability problem. (1) A polynomial-time, $\widetilde{O}(n^{2/3}g^{1/3})$-space algorithm for directed graphs embedded on orientable surfaces of genus g. (2) A polynomial-time, $\widetilde{O}(n^{2/3})$-space algorithm for all H-minor-free graphs given the tree decomposition, and (3) for $K_{3,3}$-free and K_5-free graphs, a polynomial-time, $O(n^{1/2+\epsilon})$-space algorithm, for every $\epsilon > 0$.

For the general directed reachability problem, the best known simultaneous time-space upper bound is the BBRS bound, due to Barnes, Buss, Ruzzo, and Schieber, which achieves a space bound of $O(n/2^{k\sqrt{\log n}})$ with polynomial running time, for any constant k. It is a significant open question to improve this bound for reachability over general directed graphs. Our algorithms beat the BBRS bound for graphs embedded on surfaces of genus $n/2^{\omega(\sqrt{\log n})}$, and for all H-minor-free graphs. This significantly broadens the class of directed graphs for which the BBRS bound can be improved.

1998 ACM Subject Classification F.1.3

Keywords and phrases Reachability, Space complexity, Time-Space Efficient Algorithms, Graphs on Surfaces, Minor Free Graphs, Savitch's Algorithm, BBRS Bound

1 Introduction

Given a graph G and two vertices s and t, is there a path from s to t in G? This problem, known as *the reachability problem*, is of fundamental importance in the study of space bounded complexity classes as various versions of it characterize important complexity classes (such as NL, RL, L and NC^1 [4, 20, 21]). Progress in understanding the space complexity of graph reachability problems directly relates to the progress in space complexity investigations. We refer the readers to a survey by Wigderson [28] to further understand the significance of reachability problems in complexity theory. Because of its central role, designing space and time efficient deterministic algorithms for reachability problems is a major concern of complexity theory. In this paper we focus on algorithms for reachability over directed graphs that run in polynomial-time and use sub-linear space.

Two basic algorithms for directed reachability are the Breadth First Search algorithm (BFS) and Savitch's algorithm [23]. BFS uses linear space and runs in polynomial time, whereas Savitch's algorithm uses only $O(\log^2 n)$ space, but takes super-polynomial ($\theta(n^{\log n})$) time. Thus BFS is

* Research supported in part by NSF grants 0916797, 0916525, 1421163, 1422668, and Research-I Foundation

time-efficient and Savitch's algorithm is space-efficient. Hence a natural and significant question that researchers have considered is whether we can design an algorithm for reachability whose time-bound is better than that of Savitch's algorithm and the space-bound is better than that of BFS. A concrete open question is: *Can we design a polynomial-time algorithm for the directed graph reachability problem that uses only $O(n^{1-\epsilon})$ space for some small constant ϵ?* [28].

The best known result in this direction is the bound due to Barnes, Buss, Ruzzo, and Schieber [3]. By cleverly combining BFS and Savitch's algorithm, they designed a polynomial-time algorithm for reachability that uses $O(n/2^{k\sqrt{\log n}})$ space, for any constant k. Henceforth we refer to this bound as the BBRS bound. Improving the BBRS bound remains a significant open question regarding the complexity of the graph reachability problem.

Recently there has been some progress on improving the BBRS bound for certain restricted classes of directed graphs. Asano and Doerr showed that, for any $\epsilon > 0$, there is a polynomial-time algorithm that takes $O(n^{1/2+\epsilon})$ space for reachability over directed *grid graphs* [2]. In [14], it was shown that, for any $\epsilon > 0$, the directed *planar* reachability problem can also be solved in polynomial-time and $O(n^{1/2+\epsilon})$ space. In [24], it was shown that the reachability problem for directed *acyclic* graphs with $O(n^{1-\epsilon})$ *sources nodes* and embedded on surfaces of $O(n^{1-\epsilon})$ genus can be solved in polynomial time and $O(n^{1-\epsilon})$ space. See a recent survey article [26] for more details on known results.

In this paper we design reachability algorithms that beat the BBRS bound for a substantially larger class of graphs than known before. Our main approach is to use a *space-efficient kernelization* where we compress the given graph to a smaller kernel graph preserving reachability. Once such a kernel graph is computed, we can use known algorithms (such as BFS) on the kernel graph to solve the reachability problem.

There are indications that it may be difficult to improve the BBRS bound for general directed graphs using earlier known techniques. This is because there are matching lower bounds known for general reachability on certain restricted model of computation known as NNJAG [6, 10, 19]. All the known algorithms for the general reachability problem can be implemented in NNJAG without significant blow up in time and space. However, we believe that our kernel-based approach has a potential to overcome the NNJAG bottleneck.

Our main motivation to design space-efficient algorithms for reachability problems comes from their importance in computational complexity theory. However, designing polynomial-time, sub-linear space algorithms is of clear significance from a general algorithmic perspective, especially in the context of computations over large data sets. Thus our algorithms may be of interest to a more general audience.

Our Contributions

Our first result is a new algorithm for the directed reachability problem for surface-embedded graphs.

▶ **Theorem 1.** *There is an algorithm that, given a directed graph G embedded on an orientable surface of genus g with the combinatorial embedding and two vertices s and t, decides whether there is a directed path from s to t in G. This algorithm runs in polynomial-time and uses $\tilde{O}(n^{2/3}g^{1/3})$ space, where n is the number of vertices of the graph.*

For the case when $g = n^{1-\epsilon}$, our algorithm uses $\tilde{O}(n^{1-\epsilon/3})$ space and runs in polynomial time (by $\widetilde{O}(s(n))$ we mean $O(s(n)(\log n)^{O(1)})$). In general, for graphs that are embedded on surfaces of genus $g = n/2^{\omega(\sqrt{\log n})}$, our algorithm beats the BBRS bound.

For proving the above theorem, we first give an algorithm for constructing a *planarizing set* (a set S of nodes of a graph G, so that $G \setminus S$ is a planar graph) of size $O(n^{2/3}g^{1/3})$ of the underlying undirected graph in polynomial-time and space $\tilde{O}(n^{2/3}g^{1/3})$. This space-efficient algorithm for computing a planarizing set may be of independent interest.

There are known algorithms that compute a planarizing set of a high-genus graph [9, 12, 13]. However, we cannot rely on these existing algorithms since the starting point of all these algorithms is a BFS tree computation of the input graph. In general computing a BFS tree (even for an undirected graph) is as difficult as the directed reachability problem. Avoiding a BFS tree computation of the entire graph is the the main technical challenge that we overcome in our space efficient algorithm for constructing a planarizing set.

Once a planarizing set is computed, we construct a new directed graph $\tilde{G}$, called the kernel graph on G whose vertex set is the planarizing set, so that reachability in G reduces to reachability in $\tilde{G}$. This reduction uses the $O(n^{1/2+\epsilon})$ space algorithm for directed planar reachability from [14] as a subroutine. Finally we solve reachability on $\tilde{G}$ using BFS. Since the size of $\tilde{G}$ is $O(n^{2/3}g^{1/3})$, we get the desired space bound.

Our second contribution is a new reachability algorithm for H-minor-free graphs, that improves upon the BBRS bound, where H is an arbitrary but fixed graph. To design this algorithm we assume that we are provided with the *tree decomposition* of the H-minor-free graph.

▶ **Theorem 2.** *Given a graph H, there is an algorithm that, given any H-minor-free graph G together with*

(i) a tree decomposition (T, X) of G, and
(ii) for every $X_i \in X$, the combinatorial embedding of the subgraph G_0 of $G[X_i]$,

and two vertices s and t in G, decides whether there is a directed path from s to t in G. The algorithm runs in polynomial-time and uses $\tilde{O}(n^{2/3})$ space, where n is the number of vertices of the graph.

The reader may refer to Section 4.1 to understand the notation that we use in Theorem 2. This theorem is proved by first designing a $\tilde{O}(n^{2/3})$-space and polynomial-time algorithm for constructing a $2/3$-separator of size $O(n^{2/3})$ for the given graph. Once such a separator is obtained, we use ideas from [14] to design the reachability algorithm. To construct such a separator for H-minor-free graphs, we use the tree decomposition of the given graph by [22] and find a "separating node" in that tree. Then we construct a bounded-genus graph from the graph induced by the separating node. Finally by using the planarizing set construction used to prove Theorem 1, we design an algorithm to construct a planarizing set of size $O(n^{2/3})$ of the underlying undirected graph in polynomial-time and $\tilde{O}(n^{2/3})$ space.

For $K_{3,3}$-free and K_5-free graphs we give a better upper bound than the one given in Theorem 2. Kuratowski's theorem states that planar graphs are exactly those graphs that do not contain $K_{3,3}$ and K_5 as minors. Hence it is a natural question whether results on planar graphs can be extended to graphs that do not contain either a $K_{3,3}$ minor (known as $K_{3,3}$-free graphs) or a K_5 minor (known as K_5-free graphs). Certain complexity upper bounds that hold for planar graphs have been shown to hold for $K_{3,3}$-free and K_5-free graphs [5, 7, 8, 25]. On the other hand, there are problems for which upper bounds that hold for planar graphs are not known to extend to such minor-free graphs (such as computing a perfect matching in bipartite graphs [17]). We show that the time-space bound known for planar graphs can also be obtained for both these classes of graphs. Here it is important to note that even though directed reachability in $K_{3,3}$-free and K_5-free graphs reduces to directed planar reachability [25], the reduction blows up the size of the graph by a polynomial factor and hence we can not use this approach for our purposes.

▶ **Theorem 3.** *For any constant $0 < \epsilon < 1/2$, there is a polynomial time and $O(n^{1/2+\epsilon})$ space algorithm that given a directed $K_{3,3}$-free or K_5-free graph G on n vertices, decides whether there is a directed path from s to t in G.*

Although for Theorem 2 we require additional inputs (such as the tree decomposition and the embeddings of the bounded genus parts), in Theorem 3 we do not have any such requirements. The proof idea of Theorem 3 is similar to that of Theorem 2. However we use the known algorithm to compute a planar separator instead of a bounded genus separator. This gives better space bound compared to the case of H-minor-free graphs.

The rest of the paper is organized as follows. In Section 2 we give some basic definitions and notations that we use. In Section 3, we give a construction of planarizing set for high-genus graphs and also provide a proof of Theorem 1. In Section 4, we present the algorithm for reachability in H-minor-free graphs and as a corollary we show Theorem 3. Due to space constraints, most of the proofs appear in the Appendix.

2 Preliminaries

We first define some notations which will be used later in this paper. Given a graph G and a set of vertices X, $G[X]$ denotes the subgraph of G induced by X and $V(G)$ denotes the set of vertices present in the graph G. Now we define necessary notions on graphs embedded on surfaces. We refer the reader to the excellent book by Mohar and Thomassen [18] for a comprehensive treatment of this topic. In this paper we only consider *closed orientable* surfaces. These surfaces are obtained by adding "handles" to a sphere.

Let $G = (V, E)$ be a graph and for each $v \in V$, let π_v be a cyclic permutation of edges incident on v. Let $\Pi = \{\pi_v \mid v \in V\}$. We say that Π is a *combinatorial embedding* of G. Given a combinatorial embedding we can define Π-*facial walk*. Let $e = \langle v_1 v_2 \rangle$ be an edge. Consider the closed[1] walk $f = v_1 e_1 v_2 e_2 v_3 \cdots v_k e_k v_1$ where $\pi_{v_{i+1}}(e_i) = e_{i+1}$, and $\pi_{v_1}(e_k) = e_1$. We call f a *face* of the graph G.

Given a Π-embedding of a graph G, the Π-*genus* of G is the g such that $n - e + f = 2 - 2g$, where n, e and f denote the number of vertices, edges and faces of the graph G. This is popularly known as the *Euler-Poincaré formula*.

It is known that given any graph with Π-genus g, it can be embedded on a closed orientable surface of genus g such that every face is homeomorphic to an open disc. Let Π be a combinatorial embedding of a graph G and H be a subgraph of G. The embedding Π naturally induces an embedding Π' on $G \setminus H$. By abuse of notation, we still refer to the induced embedding as Π-embedding.

Given a cycle C of a graph, we can define *left* and *right* sides of the cycle C. Two vertices are on the same side of C if they are path connected such that the path does not cross the cycle C. We use $G_l(C)$ and $G_r(C)$ to denote the left and right sides of G. Given a cycle C, we say that it is *contractible* if one of $G_l(C) \cup C$ or $G_r(C) \cup C$ has Π-genus zero (i.e. planar). We say that a cycle is *surface separating* if $G_l(C)$ and $G_r(C)$ have no edges in common. Note that every contractible cycle is surface separating. A cycle that is not surface separating is called a *non-separating cycle*. We now mention some fundamental facts about these cycles that are used throughout this paper.

▶ **Proposition 1.** *Let C be a cycle of a graph with Π-genus g. If C is non-separating, then Π-genus of $G \setminus C$ is $\leq g - 1$. If C is surface separating, then sum of Π-genera of $G_l(C) \cup C$ and $G_l \cup C$ equals g.*

An edge that appears on a facial walk f may appear once or twice on f. Any edge that appears twice on a facial walk is called *singular edge*.

▶ **Proposition 2.** *Let G be a graph with Π-genus g, and e be a singular edge such that $G \setminus e$ is connected. The Π-genus of $G \setminus e$ is $g - 1$.*

[1] A priori it is not obvious that that this leads to a closed walk. However, it can shown that this walk comes back to v_1. See [18] Chapter 3.2 for a proof.

The notions of planarizing set and separator defined below are crucial in this paper. A set S of vertices of a graph G is called a *planarizing set* if $G \setminus S$ is a planar graph. An (α, β)-separator of a graph $G = (V, E)$ having n vertices, is a subset S of V such that $|S| \leq O(\alpha)$ and every connected component in $V \setminus S$ has at most βn vertices.

Next we state two theorems about planar graphs that are used in this paper. In [14] the authors construct a $(n^{1/2}, 8/9)$-separator. By running their algorithm repeatedly (a constant number of times), we can obtain a $(n^{1/2}, 1/3)$ separator.

▶ **Theorem 4.** *[14] Given a planar graph G there is an algorithm that computes a $(n^{1/2}, 1/3)$-separator of G in polynomial time and $\tilde{O}(n^{1/2})$ space.*

We refer to the algorithm of this theorem as `PlanarSeparator` algorithm. In [14], this algorithm is used to obtain a time-space efficient algorithm for reachability on directed planar graphs

▶ **Theorem 5.** *[14] For any constant $0 < \epsilon < 1/2$, there is an algorithm that, given a directed planar graph G and two vertices s and t, decides whether there is a path from s to t. This algorithm runs in time $n^{O(1/\epsilon)}$ and uses $O(n^{1/2+\epsilon})$ space, where n is the number of vertices of G.*

3 A Reachability Algorithm for High Genus Graphs

In this section we prove Theorem 1. We will use a space-efficient construction of a planarizing set to establish this result. We first assume that the following theorem holds and then prove Theorem 1. Proof of Theorem 6 will appear in Section 3.1.

▶ **Theorem 6.** *There is an algorithm that given a combinatorial embedding of an undirected graph G embedded on an orientable surface of genus g, outputs a planarizing set of G of size $O(n^{2/3}g^{1/3})$. This algorithm runs in polynomial time and uses space $\tilde{O}(n^{2/3}g^{1/3})$. Here n denotes the number of vertices of G.*

Proof of Theorem 1. Let $\langle G, s, t\rangle$ be an instance of reachability where G whose Π-genus is g. Consider the underlying undirected graph G_{un}. By using the algorithm from Theorem 6 we first compute a planarizing set S of G_{un}. Let $\mathcal{S} = S \cup \{s, t\}$. Let G_p be the planar graph obtained by removing all vertices (and the edges incident on them) of $\mathcal{S}$ from G.

Consider the following reduction that outputs an instance $\langle \mathcal{G}, s, t\rangle$, where $\mathcal{G} = (\mathcal{S}, \mathcal{E})$. Given two nodes a and b in $\mathcal{S}$, we place a directed edge from a to b in $\mathcal{E}$, if there is a directed edge from a to b in the original directed graph G. Additionally, we place an edge from a to b in $\mathcal{E}$, if there exist vertices u and v in the vertex set of G_p such that all of the following conditions hold: 1) there is a directed edge from a to u in G, 2) there is a directed edge from v to b in G, and 3) there is a directed path from u to v in the directed planar graph G_p. Determining whether there is path from u to v in G_p can be done in polynomial-time and $O(n^{2/3})$ space, by setting ϵ to $1/6$ by Theorem 5. By Theorem 6, S can be computed in polynomial time and $\tilde{O}(n^{2/3}g^{1/3})$ space. Thus this reduction runs in polynomial time and uses $\tilde{O}(n^{2/3}g^{1/3})$ space.

We now claim that there is a path from s to t in G if and only if there is a path from s to t in $\mathcal{G}$. Consider any s-t path in $\mathcal{G}$, let $e_1, e_2, \cdots e_k$ be the edges of this path. Consider an edge $e_i = (a, b)$. Note that the reduction places this edge in $\mathcal{G}$ when, either there is a directed path or an edge from a to b in G. This implies that there is path from s to t in G. Now we prove the converse direction. Let P be a path from s to t in G. We can decompose P into $p_1e_1q_1h_1p_2e_2q_2h_2 \cdots p_k$. Here e_i is an edge from a vertex in $\mathcal{S}$ to a vertex in G_p and h_i is an edge from a vertex in G_p to a vertex in $\mathcal{S}$, q_i is the part of the path P from head of e_i to the tail of h_i so that it completely lies within G_p, and p_i is the part of the path P that completely lies in the graph induced by the planarizing set $\mathcal{S}$. By the construction of $\mathcal{G}$, there is an edge o_i from the tail of e_i to the head of h_i in $\mathcal{G}$. Thus $p_1 o_1 p_2 o_2 \cdots p_k$ is a path from s to t in $\mathcal{G}$.

Reachability in the directed graph $\mathcal{G}$ can be solved using BFS. Since the number of vertices in $\mathcal{G}$ is $O(n^{2/3} g^{1/3})$, the BFS algorithm runs in polynomial-time and uses in $\tilde{O}(n^{2/3} g^{1/3})$ space. By combining the above reduction with the reachability algorithm on $\mathcal{G}$, we obtain an algorithm that solves reachability in G that runs in polynomial time and uses $\tilde{O}(n^{2/3} g^{1/3})$ space. This completes the proof of Theorem 1. ◀

3.1 Proof of Theorem 6

The structure of the proof is as follows. Given an embedded graph, we decompose the graph into several regions. We first look for a small non-contractible cycle C inside some region. If we find one, then we add the vertices of C into the planarizing set. If C is non-separating, by Proposition 1, removal of the vertices of C will result in a graph whose genus $\leq g - 1$. If C is surface separating, since C is non-contractible, by Proposition 1, we get two components each with genera $0 < g_1, g_2 < g$ so that $g_1 + g_2 = g$. In both cases, since the genus of each component is $< g$, we can iterate this process. If this iteration stops, then all the regions of all the resulting components are homeomorphic to an open disc. In this case, for each component we identify a small subgraph based on the regions, and argue that this subgraph is a planarizing set of that component. Our final planarizing set is the collection of planarizing sets of each component together with the non-contractible cycles. Notice that at any stage the components obtained can be implicitly represented by the original graph and the cycles that are removed. Thus we do not have to explicitly store the components. We only store the non-contractible cycles that are removed. We now proceed to give a formal proof. The algorithm given in the following lemma is the core of the planarizing set algorithm.

▶ **Lemma 7.** *There is an algorithm that given a connected undirected graph G, its Π-embedding, and an integer k as input, outputs one of the following:*

1. *A non-separating cycle of size $O(k)$ or a singular edge e so that $G \setminus e$ is connected. The output of this step (either a cycle or a singular edge) is called a genus reduction set.*
2. *a non-contractible and surface-separating cycle of size $O(k)$*
3. *a planarizing set of size $O((n/k + g)\sqrt{k})$*

The algorithm runs in polynomial-time and uses $\tilde{O}(n/k + k)$ space.

The proof of the above lemma is given in the Appendix. Now using this lemma, we prove Theorem 6.

Proof of Theorem 6. The planarizing set construction algorithm applies the algorithm from Lemma 7 iteratively. We will describe the algorithm by describing an iteration. After the i^{th} iteration, we will have a collection of components $G_1, G_2, \ldots, G_m$. We will describe the $(i+1)^{st}$ iteration: The algorithm considers the first component $\hat{G}$ whose Π-genus $\hat{g}$ is non-zero and apply the algorithm from Lemma 7 on $\hat{G}$. This results in either (1) a genus-reduction set of $\hat{G}$ (2), non-contractible surface separating cycle of $\hat{G}$ (3) or planarizing set of $\hat{G}$. In cases (1) and (2) the algorithm stores the corresponding cycles. In case (3) it adds the planarizing set obtained to the final planarizing set. This process stops when all the components are planar.

We claim that after any iteration, the total number of vertices in all of the components together is at most n, and the total genera of all of the components together is at most g. Assume that this claim holds after i^{th} iteration. Let $\hat{G}$ be the component considered at the $(i+1)^{st}$ iteration. In case (1), by Propositions 1 and 2, $\hat{G}$ is reduced to a component whose genus is at most $\hat{g} - 1$. In case (2), since we have a non-contractible surface separating cycle, by Proposition 1, we get two components whose sum of the genera is at most $\hat{g}$. In case (3), $\hat{G}$ is reduced to a planar graph. Thus sums of the genera of all components is $\leq g$ and, since no vertex is repeated in more than one component, vertices in all of the components together is at most n.

Clearly this algorithm produces a planarizing set and runs in polynomial-time. We will now bound the size of the planarizing set and the space used by the algorithm.

Notice that the algorithm stores only the cycles and singular edges and will not store the components: At any stage, given the original graph, the cycles or singular edges computed so far, and an index of the component, the edge relations of that component can be computed without additional space. After at most g iterations, we are left with at most g components each of whose genus is at most 1. Since each iteration may produce a cycle of length $O(k)$, the algorithm will store at most $2g$ cycles each of length $O(k)$. Consider a component G_i in which case (3) of the lemma happens. The size of the corresponding planarizing set produced is $O(n_i/k + g_i)\sqrt{k}$. Since $\sum_i n_i \leq n$ and $\sum_i g_i \leq g$, the total size of the planarizing set is $O((n/k + g)\sqrt{k} + kg)$. Total space used is $\tilde{O}(n/k + k + kg + (n/k + g)\sqrt{k})$ (including the space to store the planarizing set).

By choosing $k = \max\{(n/g)^{2/3}, 1\}$, we get that the total space-bound of the algorithm to compute the planarizing set is $\tilde{O}(n^{2/3}g^{1/3})$, and the size of the planarizing set produced is $O(n^{2/3}g^{1/3})$. ◀

4 A Reachability Algorithm for H-minor-free Graphs

In this section, we prove Theorem 2 by first giving an algorithm to construct a separator of the input graph. Towards this we define the notion of a tree decomposition of a graph which is crucial to the construction.

4.1 Graph Minor Decomposition Theorem

A graph H is said to be a *minor* of a graph G if H can be obtained from a subgraph of G by contracting some edges. A graph G is said to be *H-minor-free* if G does not contain H as a minor, for some graph H.

▶ **Definition 8.** A *tree decomposition* of a graph $G = (V, E)$ is the tuple (T, X) where $T = (V_T, E_T)$ is a tree and $X = \{X_i \mid i \in V_T\}$, such that, (a) $\cup_i X_i = V$, (b) for every edge (u, v) in G, there exists i, such that u and v belong to X_i, and (c) for every $v \in V$, the set of nodes $\{i \in V_T \mid v \in X_i\}$ forms a connected subtree of T.

We will refer to the X_i's as *bags* of vertices. Note that each bag corresponds to a node (we call vertices of T as *nodes*) in the tree T. The *width* of a tree decomposition (T, X), is the maximum over the size of X_i's minus 1. The *treewidth* of a graph is the minimum width over all possible tree decompositions of G. A tree decomposition is said to be a path decomposition if $T = (V_T, E_T)$ is a path and *pathwidth* of a graph is the minimum width over all possible path decompositions of G. For a fixed graph H, Robertson and Seymour, gave a tree decomposition for every H-minor-free graph [22]. Before we see the Theorem we need to state some definitions. A graph G is called *almost h-embeddable* if there exists a set of vertices Y (called the *apices*) of size at most h such that, (i) $G \setminus Y$ can be written as $G_0 \cup G_1 \cup \ldots \cup G_h$, (ii) G_0 has an embedding on a surface of genus at most h (say $\mathcal{S}$), (iii) for $i = 1, \cdots, h$, G_i's are pairwise disjoint (we shall refer to them as *vortices*), (iv) there exists faces $F_1, \cdots, F_h$ of G_0 and pairwise disjoint discs $D_1, \cdots, D_h$ on $\mathcal{S}$ such that for all $i \in \{1, \ldots, h\}$, $D_i \subseteq F_i$ and $U_i := V(G_0) \cap V(G_i) = V(G_0) \cap D_i$, and (v) for each graph G_i, there is a path decomposition $(\mathcal{P}_u)_{u \in U_i}$ of width at most h such that $u \in \mathcal{P}_u$, for all $u \in U_i$. The sets of vertices in $\mathcal{P}_u$ are ordered according to the ordering of the corresponding u's as vertices along the boundary of face F_i in G_0.

Let G and H be two graphs each containing cliques of equal sizes. The *clique-sum* of G and H is formed by identifying pairs of vertices in these two cliques to form a single shared clique, and then possibly deleting some of the clique edges (may be none). A *k-clique-sum* is a clique-sum in which both cliques have at most k vertices. The k-clique-sum of G and H is denoted as $G \oplus_k H$. The set of shared vertices in this operation is called the *join set*.

We are now ready to state the decomposition theorem for H-minor-free graphs.

▶ **Theorem 9** ([22]). *For every graph H, depending only on $|V(H)|$, there exists an integer $h \geq 0$ such that every H-minor-free graph can be represented as at most h-clique-sum of "almost h-embeddable" graphs in some surface on which H cannot be embedded.*

Henceforth, we will assume that the tree decomposition of the original graph and the combinatorial embedding of all subgraphs (the G_0's in each almost h-embeddable graph) that are embedded on the surface are provided as part of the input. We will refer to this as *tree decomposition with combinatorial embedding* of H-minor-free graphs.

4.2 Constructing Separator for H-minor-free Graphs

Now we will show that given a decomposition of a H-minor-free graph stated in the last subsection, how to construct a separator. We start with the following lemma whose proof is given in the Appendix.

▶ **Lemma 10.** *There exists a log-space algorithm, that given a tree decomposition (T, X) of a graph G on n vertices, outputs a node $i \in T$ such that every connected component in $G[V \setminus X_i]$ has at most $n/2$ vertices.*

We now give a separator construction for all H-minor-free graphs which is the main contribution of this whole section.

▶ **Theorem 11.** *Given a H-minor-free graph G and its tree decomposition with combinatorial embedding, there exists an $\tilde{O}(n^{2/3})$ space, polynomial time algorithm that computes a $(n^{2/3}, 2/3)$-separator of G.*

Proof. Given an input graph G and its tree decomposition, compute the vertex i using Lemma 10. The separator for G that we would construct would be a subset of X_i. Let i have m neighbors in T, say $i_1, \ldots, i_m$. Now for every $j \in [m]$, $G[X_i]$ is joined with $G[X_{i_j}]$ using the clique-sum operation of at most h (constant depending only on H) vertices. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ where C_j is a set of at most h vertices in X_i, such that $G[X_i]$ is joined with $G[X_{i_j}]$ via C_j. Let T_j be the connected subtree of $T \setminus i$ containing the node j. We define the subgraph G_j to be the induced subgraph of G corresponding to the vertices in the subtree T_j. In other words, $G_j = G[\cup_{l \in T_j} X_l]$. Let $k_j = |G_j|$.

Now if $|X_i| \leq O(n^{2/3})$, then it follows from Lemma 10 that X_i is a $(n^{2/3}, 1/2)$-separator of G. Otherwise, consider the node i and its corresponding almost h-embeddable graph $K = G[X_i]$. Now consider the representation of K using apices and vortices. Let Y be the set of apices and $K \setminus Y$ can be written as $K_0 \cup K_1 \cup \cdots \cup K_h$ where each of K_i has a path decomposition $(\mathcal{P}_u)_{u \in U_i}$ of width less than h. Now build a new graph K' from K_0 using the following steps: for $i = 1, \cdots, h$, add a cycle of length $|\mathcal{P}_u|$ attached to the vertex $u \in U_i$ inside the face F_i and then connect those cycles such that they form a path like structure similar to the corresponding path decomposition. The new graph K' is a graph embedded on a constant genus and so from Theorem 6, we can get a $(n^{2/3}, 2/3)$-separator S (which is union of planarizing set of K', say Z and output of `PlanarSeparator` on the graph $K' \setminus Z$) using $\tilde{O}(n^{2/3})$ space and polynomial time. If S contains some vertices from a newly added cycle, then we add all the vertices present in the corresponding "bag" of vertices of the respective path decomposition. We also add all the apices of K_0 and we get a new set S'. As the size of S is

$O(n^{2/3})$, so the size of S' will be at most $O(hn^{2/3}) = O(n^{2/3})$.

▶ **Claim 1.** *S' is a $(n^{2/3}, 2/3)$-separator of K.*

Proof. Observe that by construction, K' is a graph embedded on a bounded genus surface. Moreover there is a canonical injective map (say σ) from vertices in K to vertices in K'. To see this, note that $K' = K_0 \cup$ {newly added cycles} and by construction, for every vertex in the bag X_i there is a vertex in the newly added cycle in K'.
Since S is a $(n^{2/3}, 2/3)$-separator of K', S' is also a $(n^{2/3}, 2/3)$-separator of K. Let C be a connected component in $K \setminus S'$. Then the vertices corresponding to C in K' (via the map σ) also form a connected component. Since every connected component in $K' \setminus S$ has size at most $2|K'|/3$, so S' is a $(n^{2/3}, 2/3)$-separator of K. ◀

By running the above construction repeatedly (a constant number of times), we can get a $(n^{2/3}, 1/6)$-separator $\overline{S}$. As according to Lemma 10, $G[V \setminus X_i]$ contains at most $n/2$ vertices, so the set $\overline{S}$ also acts as a $(n^{2/3}, 2/3)$-separator for the whole graph G. It is clear from the construction of $\overline{S}$ that this algorithm will take $\tilde{O}(n^{2/3})$ space and polynomial time. ◀

We also consider the special case when H is either the $K_{3,3}$ or the K_5.

▶ **Theorem 12** ([25,27]). *Let (T, X) be a tree decomposition of a $K_{3,3}$-free or K_5-free graph G. Then*

(i) for every $X_i \in X$, $G[X_i]$ is either a planar graph or the K_5 (if G is $K_{3,3}$-free) or V_8 (if G is K_5-free) (see Figure 2 in Appendix), and
(ii) G is the 3-clique-sum of $G[X_i]$ and $G[X_j]$ for every adjacent vertices i, j in T.

Moreover given a $K_{3,3}$-free or K_5-free graph G, such a tree decomposition can be computed in logspace.

Thierauf and Wagner have shown how to compute the tree decomposition of a $K_{3,3}$-free or K_5-free graph given in Theorem 12 in log-space [25] and thus we get the following corollary for these special class of H-minor-free graphs.

▶ **Corollary 13.** *Given a $K_{3,3}$-free or K_5-free graph G, there exists an $\tilde{O}(n^{1/2})$ space, polynomial time algorithm that computes a $(n^{1/2}, 2/3)$-separator of G.*

The detailed proof of the above stated corollary is given in the Appendix.

Proof of Theorem 2. Observe that the planar reachability algorithm of Theorem 5 essentially uses the properties that (I) a subgraph of a planar graph is also planar, and (II) their exists an algorithm that computes a $(n^{1/2}, 2/3)$-separator of a planar graph in polynomial time and $\tilde{O}(n^{1/2})$ space. Note that by the definition itself, all the subgraphs of a H-minor-free graph is also H-minor-free and given a tree decomposition, from Theorem 11 we get an algorithm that computes a $(n^{2/3}, 2/3)$-separator of a H-minor-free graph in polynomial time and $\tilde{O}(n^{2/3})$ space. Now using the algorithm stated in Theorem 5, we get our desired result. ◀

Just mimicking the above proof, we can achieve a better simultaneous time-space bound for the directed reachability problem over $K_{3,3}$-free or K_5-free graphs as stated in Theorem 3 using the separator obtained from the Corollary 13.

References

1 Eric Allender and Meena Mahajan. The complexity of planarity testing. *Information and Computation*, 189:117–134, 2004.

2 Tetsuo Asano and Benjamin Doerr. Memory-constrained algorithms for shortest path problem. In *CCCG*, 2011.

3 Greg Barnes, Jonathan F. Buss, Walter L. Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed s-t connectivity. In *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, pages 27–33, 1992.

4 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

5 Chris Bourke, Raghunath Tewari, and N. V. Vinodchandran. Directed planar reachability is in unambiguous log-space. *ACM Transactions on Computation Theory*, 1(1):1–17, 2009.

6 Stephen A. Cook and Charles Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM J. Comput.*, 9(3):636–652, 1980.

7 Samir Datta, Nutan Limaye, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Planar graph isomorphism is in log-space. In *Annual IEEE Conference on Computational Complexity*, pages 203–214, 2009.

8 Samir Datta, Prajakta Nimbhorkar, Thomas Thierauf, and Fabian Wagner. Graph isomorphism for $K_{\{}3,3\}$-free and K_5-free graphs is in log-space. In *FSTTCS*, pages 145–156, 2009.

9 Hristo Djidjev and Shankar M. Venkatesan. Planarization of graphs embedded on surfaces. In *Workshop on Graph Theoretic Concepts in Computer Science*, pages 62–72, 1995.

10 Jeff Edmonds, Chung Keung Poon, and Dimitris Achlioptas. Tight lower bounds for st-connectivity on the NNJAG model. *SIAM J. Comput.*, 28(6), 1999.

11 Hillel Gazit and Gary L. Miller. A parallel algorithm for finding a separator in planar graphs. In *FOCS*, pages 238–248, 1987.

12 John R. Gilbert, Joan P. Hutchinson, and Robert Endre Tarjan. A separator theorem for graphs of bounded genus. *J. Algorithms*, 5(3):391–407, 1984.

13 Joan P. Hutchinson and Gary L. Miller. Deleting vertices to make graphs of positive genus planar. *Discrete Algorithms and Complexity Theory*, 1986.

14 T. Imai, K. Nakagawa, A. Pavan, N.V. Vinodchandran, and O. Watanabe. An $O(n^{1/2+\epsilon})$-Space and Polynomial-Time Algorithm for Directed Planar Reachability. In *Computational Complexity (CCC), 2013 IEEE Conference on*, pages 277–286, 2013.

15 Philip Klein. On Gazit and Miller's parallel algorithm for planar separators: achieving greater efficiency through random sampling. In *Proceedings of the fifth annual ACM symposium on Parallel algorithms and architectures*, SPAA '93, pages 43–49, New York, NY, USA, 1993. ACM.

16 Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.

17 Gary L. Miller and Joseph Naor. Flow in planar graphs with multiple sources and sinks. *SIAM Journal on Computing*, 24:1002–1017, 1995.

18 Bojan Mohar and Carsten Thomassen. Graphs on surfaces. 2001. *Johns Hopkins Stud. Math. Sci*, 2001.

19 Chung Keung Poon. Space bounds for graph connectivity problems on node-named jags and node-ordered jags. In *FOCS*, pages 218–227, 1993.

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

21 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *STOC '06: Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*, pages 457–466, New York, NY, USA, 2006. ACM.

22 Neil Robertson and P. D. Seymour. Graph minors. xvi. excluding a non-planar graph. *J. Comb. Theory Ser. B*, 89(1):43–76, September 2003.

23 Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4:177–192, 1970.

24 Derrick Stolee and N. V. Vinodchandran. Space-efficient algorithms for reachability in surface-embedded graphs. In *IEEE Conference on Computational Complexity*, pages 326–333, 2012.

25 Thomas Thierauf and Fabian Wagner. Reachability in $K_{3,3}$-free Graphs and K_5-free Graphs is in Unambiguous Log-Space. In *17th International Conference on Foundations of Computation Theory (FCT)*, Lecture Notes in Computer Science 5699, pages 323–334. Springer-Verlag, 2009.

26 N. V. Vinodchandran. Space complexity of the directed reachability problem over surface-embedded graphs. Technical Report TR14-008, ECCC, 2014.

27 K. Wagner. Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114(1):570–590, 1937.

28 Avi Wigderson. The complexity of graph connectivity. *Mathematical Foundations of Computer Science 1992*, pages 112–132, 1992.

A Proof of Lemma 7

As mentioned in the beginning of Section 3.1, we first decompose the input graph into several regions. For this decomposition we use the notion of "Voronoi regions" introduced by Gazit and Miller [11] in the context of planar graphs. We find this notion applicable to surface-embedded graphs also.

Let G be a Π-embedded graph and let f be a *face*. Two faces of G are *edge-connected* if they share an edge. A set of faces R is *edge-connected* if for every pair of faces f and g in R, there exist faces $f_1, \cdots, f_i$ in R such that f and f_1 are edge-connected, f_i and g are edge-connected, and f_j and f_{j+1} are edge-connected for all j, $1 \leq j \leq i-1$. A *region* of a surface-embedded graph is a set of edge-connected faces. The boundary of a region is the set of edges such that each edge lies on exactly one face of the region. Given a region R, we denote the boundary with $\mathcal{B}(R)$. We next define the notions of face-vertex graph and neighborhood introduced by Gazit and Miller [11].

▶ **Definition 14.** The *face-vertex* graph of G is a graph denoted as $G' = (V', E')$ where V' is the set of faces of G and E' is the set of pairs (f_1, f_2) of faces of G such that f_1 and f_2 share a vertex in G.

The distance between two vertices v_1 and v_2 (denoted as $dist(v_1, v_2)$) is the length (i.e., the number of edges) of the shortest path between them. Similarly, for any faces f_1 and f_2 in G, the distance between f_1 and f_2 (denoted by $dist(f_1, f_2)$) is the length of the shortest path between f_1 and f_2 in G'. Note that if every face of G is a triangle, then for any v_1 and v_2 of G that lie on triangle faces f_1 and f_2 respectively, we have $dist(v_1, v_2) \leq dist(f_1, f_2) + 1$.

For a face f and an integer r, let $\ell(f, r)$ denote the set of faces that are at distance exactly r from f. For any $d \geq 1$, the *d-radius ball* around f is the set $B_d(f) = \{g \in V' \mid dist(f, g) \leq d\}$. Given a face f and a region R, the distance between f and R is defined by $dist(f, R) = \min_{g \in R}\{dist(f, g)\}$.

▶ **Definition 15.** For any face f of G, the *k-neighborhood of f* (denoted as $N_k(f)$) is the set of k faces closest to f with respect to the distance function $dist$. More formally, $N_k(f) = B_r(f) \cup F$, where r is the maximum integer such that $|B_r(f)| \leq k$, and F is an edge-connected subset of $\ell(f, r+1)$ so that $|N_k(f)|$ becomes exactly k.

▶ **Definition 16.** A set I of faces is a *k-maximal independent set* if (a) for every $f, g \in I$, $N_k(f) \cap N_k(g) = \emptyset$, and, (b) for every $f' \notin I$, there exists a face $f \in I$ such that $N_k(f') \cap N_k(f) \neq \emptyset$.

Note that the size of a k-maximal independent set is $O(n/k)$. We can compute a k-maximal independent set with time and space stated below by a straight forward greedy algorithm that considers faces in the lexicographic order. Thus, in our discussion, we may assume that some k-maximal independent set I is given (as a part of the input).

▶ **Lemma 17.** *There is an algorithm that takes a Π-embedded graph G and an integer k as an input, outputs a k-maximal independent set in polynomial time and $\widetilde{O}(n/k + k)$ space.*

Next, we define the notion of a "Voronoi region" [11]. We fix some k-maximal independent set I. For every face g of G, we associate a unique member of I as follows: If $g \in N_k(f)$ for some $f \in I$, then g is associated to f. For $g \notin N_k(f)$ for any $f \in I$, g is associated with the lexicographically first $f \in I$ such that $dist(N_k(f), g)$ is the smallest among all faces in I. The *Voronoi region* of $f \in I$, denoted as $V(f)$, is the set of faces that are associated with f.

We now state a lemma that shows that the BFS of any Voronoi region can be computed space-efficiently. The proof is identical to the proof given in [14].

▶ **Lemma 18** ([14]). *There is a polynomial-time and $\widetilde{O}(n/k + k)$-space algorithm that, given a triangulated Π-embedded graph G, a k-maximal independent set I, and $f \in I$, constructs a BFS tree of $V(f)$ rooted at f. The diameter of this tree is $O(k)$.*

Note that the above lemma assumes that the input graph G is triangulated. In general a surface-embedded graph G can be triangulated (given its π-embedding) easily, by adding a vertex in the interior of each face and connecting this vertex to all other vertices of that face. However, this process increases the number of vertices of the graph and will be a problem for us. We need a slightly different triangulation that can be done if the graph does not have *singular edges*.

Without loss of generality we may assume that the input graph is 2-edge connected, if not we can identity an edge that disconnects the (by using Reingold's algorithm) and another edge between its end points. Thus the first step of our algorithm looks for a singular edge e in the graph, and if such an edge exist it returns e. Since the graph is 2-edge connected, $G - e$ is connected.

Therefore, from now we assume that the graph does not have any singular edges. If a graph does not have any singular edges, then we can triangulate the graph as follows. In this case, each face is homeomorphic to a *closed disc*. For each face, identify a lexicographically smallest vertex on it, and add an edge from that vertex to all other vertices of that face. This will triangulate the graph. Note that this triangulation is done implicitly, we need not store these newly added edges. Given any pair of vertices, we can easily test whether there is an edge between them by either looking at the input graph or by testing whether the above triangulation process places an edge between them.

▶ **Lemma 19.** *Let $V(f)$ and $V(g)$ be two Voronoi regions that share their boundary. If $V(f) \cup V(g)$ has a non-contractible cycle, then it has a non-contractible cycle of size $O(k)$. Moreover, we can find such a cycle in polynomial-time and space $\tilde{O}(n/k + k)$.*

Proof. We first claim that there is a spanning tree of $V(f) \cup V(g)$ with diameter $O(k)$. Consider BFS trees T_f and T_g of $V(f)$ and $V(g)$ respectively. Consider all vertices of G that are common to both these trees; such vertices must be leaf nodes of these BFS trees. Consider the lexicographically smallest such vertex, disconnect all other common vertices from T_f. We now are left with a spanning tree of $V(f) \cup V(g)$. We denote this spanning tree as T_{fg}. Since the depth of each BFS tree is $O(k)$, the diameter of this spanning tree is $O(k)$. Note that T_{fg} can be computed in polynomial-time and space $\tilde{O}(n/k + k)$.

It is known that all non-contractible cycles of any graph G satisfy *3-path condition* (See Chapter 4 of [18] for a proof). The same proof shows that all non-contractible cycles that lie within $V(f) \cup V(g)$ also satisfy 3-path condition. Consider the spanning tree T_{fg}. Since its diameter is $O(k)$, any fundamental cycle of this tree of size $O(k)$. The 3-path condition implies that one of the fundamental cycles is non-contractible [18]. The tree T_{fg} can be computed using space $\tilde{O}(n/k + k)$, and checking whether a cycle is contractible is equivalent to checking planarity which can be done in $O(\log n)$ space [1]. Thus if $V(f) \cup V(g)$ has a non-contractible cycle, then we can find one such cycle of size $O(k)$ using $\tilde{O}(n/k + k)$ space and polynomial-time. ◀

The algorithm now proceeds as follows: For every two Voronoi regions $V(f)$ and $V(g)$ that share a boundary, it attempts to find a non-contractible cycle C in $V(f) \cup V(g)$ using Lemma 19. If such a cycle is found, then it outputs C. Note that C is either a non-separating cycle or a non-contractible, surface-separating cycle.

If the algorithm does not find a non-contractible cycle, then we would like to output a small subgraph which is a planarizing set. A natural candidate for this subgraph is the set of all edges that are common to the boundaries of Voronoi regions. Since none of the Voronoi regions have non-contractible cycles, each region is planar. Thus removal of all the boundary edges will planarize the graph. However, the number of boundary edges could be very large and thus we do not obtain a small planarizing set. To get around this problem, we need additional notions such as core and Voronoi nodes. These notions have been found to be useful in planar separator constructions [11, 14–16].

▶ **Definition 20** (Core)**.** Let f be a face and $N_k(f)$ be its k-neighborhood. Let r_0 be the largest number such that $\ell(f, r_0) \subseteq N_k(f)$ and $|\ell(f, r_0)| \leq \sqrt{k}$. The core of f is defined as the union of $\ell(f, i), 1 \leq i \leq r_0$.

The following lemma is critical. For a proof see [15].

► **Lemma 21.** *For every face f, (a) the size of the boundary of the core of f is at most $\sqrt{k}$, and (b) for every face $f' \in N_k(f)$ and not in core of f, there is a face g in the core of f such that $dist(f', g) \leq \sqrt{k}$.*

We can extend the notion of core a face to core of a vertex. For any vertex v of G, let f be a triangle face (for consistency we take the lexicographically smallest face) on which v lies. The *core of v* is simply the core of f.

For any Voronoi region, consider its boundary and vertices (of G) on the boundary. All such vertices belong to at least two Voronoi regions. Some of these vertices may belong to three or more different Voronoi region. We call such vertices *Voronoi vertices*.

Now we are ready to describe the algorithm claimed in Lemma 7 (given below).

```
  Input : Graph $G$ and a number $k$
1 Output the vertices of a singular edge if there exists one and halt;
2 for every pair $(V(f), V(g))$ whose boundaries intersect do
3     if there is a non-contractible cycle $C$ then Output $C$ and halt;
4 end
5 for every Voronoi vertex $v$ do
6     if there is a non-contractible cycle $C$ in the core$(v)$ then Output $C$ and halt;
7 end
8 Output the subgraph $\tilde{G}$ whose description is given below;
```

Algorithm 1: Algorithm in Lemma 7

We will define the subgraph $\tilde{G}$ in Step 6 of the above algorithm. Identical constructions have been used in earlier work in the context of planar separator algorithms [11, 14].

(1) For every $f \in I$, output the boundary of the core of f.
(2) For every $v \in V$, determine if it is a Voronoi vertex. If v is a Voronoi vertex, then output the boundary of the core of v.
(3) For every pair of faces f and g in I such that $\mathcal{B}(V(f))$ and $\mathcal{B}(V(g))$ intersect, do the following: Compute all Voronoi vertices that are common to $\mathcal{B}(V(f))$ and $\mathcal{B}(V(g))$. For every such vertex v, compute the BFS path $\widehat{P}_{f,v}$ from f to v in $V(f)$. Then select the part of $\widehat{P}_{f,v}$ that lies outside of the cores of f and v, and output it as $P_{f,v}$. Similarly, output $P_{g,v}$.

See Figure 1 for a pictorial description of the graph $\tilde{G}$.

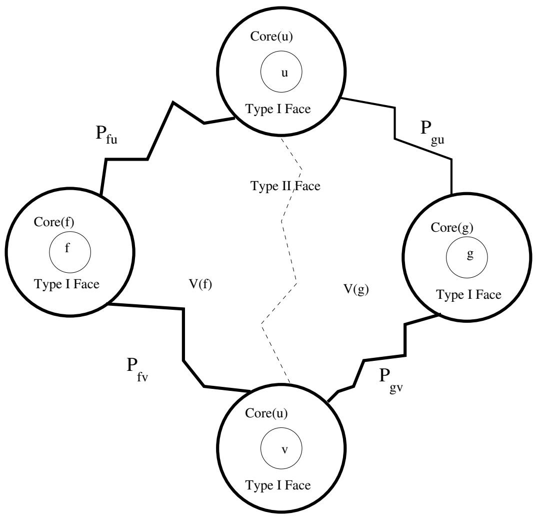

Figure 1 The graph $\tilde{G}$. The bold lines are the edges of $\tilde{G}$ and the dotted line is the boundary of Voronoi regions $V(f)$ and $V(g)$.

To complete the proof of the Lemma 7, we need to show the following claim.

▶ **Claim 2.** *The graph $\tilde{G}$ that the above algorithm outputs is a planarizing set of G. Moreover, $\tilde{G}$ has size $O((\frac{n}{k} + g)\sqrt{k})$ and can be computed in polynomial-time and space $O(n/k + k)$.*

Proof. We will analyze the graph $\tilde{G}$ by considering its faces which we categorize into Type I and Type II.

Type I: A face consisting of edges from the boundary of some core that are produced by steps (1) or (2).

Type II: A face consisting of the edges of four paths $P_{f,u}, P_{f,v}, P_{g,u}, P_{g,v}$ and some edges in boundaries of the cores of f, g, u, and v, where f and g are faces of G such that $\mathcal{B}(V(f))$ and $\mathcal{B}(V(g))$ intersect, and u and v are Voronoi vertices that appear in both $V(f)$ and $V(g)$. Note that the edges belonging to these paths are produced by step (3)

Now we claim that the vertices of $\tilde{G}$ is a planarizing set. Each Type I face of $\tilde{G}$ corresponds to a core of the original graph. Each Type II face is a sub-region of $V(f) \cup V(g)$ for some f and g whose boundaries intersect. Since Step 6 of the algorithm is reached only when none of the cores or $V(f) \cup V(g)$ has non-contractible cycle, all faces of $\tilde{G}$ are planar. Hence $\tilde{G}$ is a planarizing set of G. Now we bound the size of $\tilde{G}$. Consider $\tilde{G}$, contract every degree-two vertex to form a graph $\tilde{G}'$. Since contraction does increase the genus, $\tilde{G}'$ has genus at most g. Now every vertex in $\tilde{G}'$ has degree at least 3. By using Euler-Poincaré formula, we can bound the number of vertices in $\tilde{G}'$ to be $O(n/k + g)$, and thus the number of faces to be $O(n/k + g)$. This implies that the number of edges in G' is $O(n/k + g)$. Note that every edge in $\tilde{G}'$ corresponds to a path in $\tilde{G}$ consisting only of degree two vertices. This path is either part of the boundary of a core or a path of the form $P_{f,v}$. Thus by Lemma 21, the length of this path is $O(\sqrt{k})$. From this it follows that the number of edges of $\tilde{G}$ is $O(n/k + g)\sqrt{k}$. This completes the proof of the above claim and also the proof of Lemma 7. ◀

B Proofs of Section 4

Proof of Lemma 10. Pick a node $i \in T$. We shall refer to i as the current node. If every connected component in $G[V \setminus X_i]$ has at most $n/2$ vertices then output i and stop. Otherwise, let C be the connected component in $G[V \setminus X_i]$ such that $|C| > n/2$. Let j be the unique neighbor of i in T such that $X_j \cap C \neq \emptyset$. The reason why j is unique is because, if there are more one neighbors of i (say j_1 and j_2) in T, such that $X_{j_1} \cap C \neq \emptyset$ and $X_{j_2} \cap C \neq \emptyset$, then j_1 and j_2 are connected by a path in $T \setminus \{i\}$, since C is a subgraph of $G[V \setminus X_i]$. Now this path together with the edges (i, j_1) and (i, j_2) forms a cycle in T, which is a contradiction.

Now the sum of the number of vertices of all other connected components of $G[V \setminus X_i]$ other than C, together with X_i is less than $n/2$. Therefore, for the node $j \in T$, the largest connected component in $G[V \setminus X_j]$ has strictly lesser number of vertices than C. Now we set j as the current node and repeat the above process. The process terminates since $|C|$ that we obtain in each step, strictly decreases.

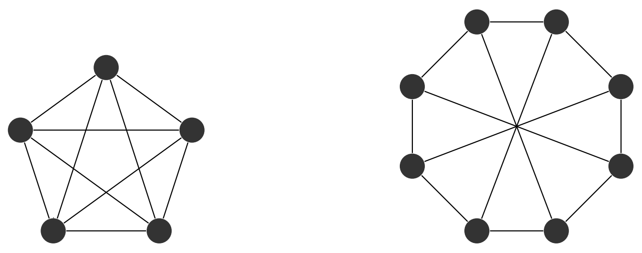
Figure 2 The K_5 and the V_8 (also known as Wagner's graph)

Proof of Corollary 13. Given an input graph G, first compute a decomposition tree and compute the vertex i by running the algorithm from Lemma 10. The separator for G that we would construct would be a subset of X_i. Let i have m neighbors in T, say $i_1, \ldots, i_m$. Now for every j, $G[X_i]$ is joined with $G[X_{i_j}]$ using the clique-sum operation of at most 3 vertices. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ where C_j is a set of at most 3 vertices in X_i, such that $G[X_i]$ is joined with $G[X_{i_j}]$ via C_j. Let T_j be the connected subtree of $T \setminus i$ containing the node j. We define the subgraph G_j to be the induced subgraph of G corresponding to the vertices in the subtree T_j. In other words, $G_j = G[\cup_{l \in T_j} X_l]$. Let $k_j = |G_j|$.

Now if $|X_i| \leq n^{1/2}$, then it follows from Lemma 10 that X_i is a $(n^{1/2}, 1/2)$-separator of G. Otherwise, we know from Proposition 12 that $G[X_i]$ is a planar graph. Construct a graph H by taking a copy of $G[X_i]$ and replacing C_j with a planar graph of size k_j, for every $j \in [m]$, such that connectivity is preserved (for example, the planar graph can be a cycle of length k_j). Let P_j denote the planar graph that replaces C_j. Note that $|H| \leq 2n$ since every vertex in G has at most two copies in H (that is if the vertex belongs to some C_j).

Observe that by construction, H is a planar graph. Moreover there is a canonical injective map (say σ) from vertices in G to vertices in H. To see this, note that every vertex in the bag X_i is present in H as H contains a copy of $G[X_i]$ and though we replaced the vertices in C_j with the planar graph P_j, the graph P_j contains vertices corresponding to the vertices in C_j, since both graphs $G[X_i]$ and $G[X_{i_j}]$ contain a copy of the vertices in C_j.

Now let S be a $(n^{1/2}, 1/3)$-separator of H as obtained by `PlanarSeparator`. Define

$$\mathcal{P} = \{j \in [m] \mid P_j \cap S \neq \emptyset\}.$$

Clearly, $|\mathcal{P}| \leq |S|$.

Let

$$S' = S \cup (\cup_{j \in \mathcal{P}} C_j)\,,$$

that is, S' is the separator S together with those sets C_j's such that the planar graph in H corresponding to the set C_j shares at least one vertex with S. Since each C_j has size at most 3, $|S'| \leq 4 \cdot |S|$.

Once we have a tree decomposition, it is easy to see that the set S' can be computed by a log-space algorithm with one oracle query to `PlanarSeparator`.