58th Annual IEEE Symposium on Foundations of Computer Science

Polylogarithmic approximation for minimum planarization (almost)

Ken-ichi Kawarabayashi
National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku
Tokyo, Japan
Email: k_keniti@nii.ac.jp

Anastasios Sidiropoulos
Department of Computer Science
University of Illinois at Chicago
Chicago, USA
Email: sidiropo@uic.edu

***Abstract*—In the *minimum planarization* problem, given some n-vertex graph, the goal is to find a set of vertices of minimum cardinality whose removal leaves a planar graph. This is a fundamental problem in topological graph theory. We present a $\log^{O(1)} n$-approximation algorithm for this problem on general graphs with running time $n^{O(\log n/\log\log n)}$. We also obtain a $O(n^{\varepsilon})$-approximation with running time $n^{O(1/\varepsilon)}$ for any arbitrarily small constant $\varepsilon > 0$. Prior to our work, no non-trivial algorithm was known for this problem on general graphs, and the best known result even on graphs of bounded degree was a $n^{\Omega(1)}$-approximation [1].**

As an immediate corollary, we also obtain improved approximation algorithms for the crossing number problem on graphs of bounded degree. Specifically, we obtain $O(n^{1/2+\varepsilon})$-approximation and $n^{1/2}\log^{O(1)} n$-approximation algorithms in time $n^{O(1/\varepsilon)}$ and $n^{O(\log n/\log\log n)}$ respectively. The previously best-known result was a polynomial-time $n^{9/10}\log^{O(1)} n$-approximation algorithm [2].

Our algorithm introduces several new tools including an efficient grid-minor construction for apex graphs, and a new method for computing irrelevant vertices. Analogues of these tools were previously available only for exact algorithms. Our work gives efficient implementations of these ideas in the setting of approximation algorithms, which could be of independent interest.

***Keywords*-minimum planarization; approximation algorithm; polylogarithmic approximation; quasi-polynomial time**

I. Introduction

In the *minimum planarization* problem, given a graph G, the goal is to find a set of vertices of minimum cardinality whose removal leaves a planar graph. This is a fundamental problem in topological graph theory, which been extensively studied over the past 40 years. It generalizes planarity, and has connections to several other problems, such as crossing number and Euler genus. The problem is known to be fixed-parameter tractable [3]–[5], but very little is known about its approximability.

A. Our contribution

Prior to our work, no non-trivial approximation algorithm for minimum planarization was known for general graphs. The only prior result was a $n^{\Omega(1)}$-approximation for graphs of bounded degree [1]. We present the first non-trivial approximation algorithms for this problem on general graphs. Our main results can be summarized as follows:

Theorem 1.1: There exists a $O(\log^{32} n)$-approximation algorithm for the minimum vertex planarization problem with running time $n^{O(\log n/\log\log n)}$.

Theorem 1.2: For any arbitrarily small constant $\varepsilon > 0$, there exists a $O(n^{\varepsilon})$-approximation algorithm for the minimum vertex planarization problem with running time $n^{O(1/\varepsilon)}$.

Applications to crossing number: The *crossing number* of a graph G, denoted $\mathsf{cr}(G)$, is the minimum number of crossings in any drawing of G into the plane (see [2]). Prior to our work, the best-known approximation for the crossing number of bounded-degree graphs was due to Chuzhoy [2]. Given a bounded-degree graph, her algorithm computes a drawing with $(\mathsf{cr}(G))^{10}\log^{O(1)} n$ crossings, which implies a $n^{9/10}\log^{O(1)} n$-approximation. We now explain how our result on minimum planarization implies an improved approximation algorithm for crossing number on bounded-degree graphs. It is easy to show that for any graph G, $\mathsf{mvp}(G) \leq \mathsf{cr}(G)$, simply by removing one endpoint of one edge involved in each crossing in some optimal drawing. Thus, using our α-approximation algorithm for minimum planarization, we can compute a planarizing set of size at most $\alpha \cdot \mathsf{cr}(G)$. Thus, in graphs of maximum degree Δ, we can compute some $F \subset E(G)$, with $|F| \leq \alpha\Delta\mathsf{cr}(G)$, such that $G \setminus F$ is planar. Chimani and Hlinený [6] (see also [7]) have given a polynomial-time algorithm which given some graph G and some $F \subset E(G)$, such that $G \setminus F$ is planar, computes a drawing of G with at most $O(\Delta^3 \cdot |F| \cdot \mathsf{cr}(G) + \Delta^3 \cdot |F|^2)$ crossings. Combining this with our result we immediately obtain an algorithm with running time $n^{O(\log n/\log\log n)}$, which given a graph G of bounded degree, computes a drawing of G with at most $(\mathsf{cr}(G))^2\log^{O(1)} n$ crossings. Similarly, we obtain an algorithm with running time $n^{O(1/\varepsilon)}$, which given a graph G of bounded degree, computes a drawing of G with at most $(\mathsf{cr}(G))^2 n^{\varepsilon}\log^{O(1)} n$ crossings, for any fixed $\varepsilon > 0$. Combining this with existing approximation algorithms for crossing number of graphs of bounded degree that are based on balanced separators, we obtain the following (see [2] for

0272-5428/17 $31.00 © 2017 IEEE
DOI 10.1109/FOCS.2017.77

details).

Theorem 1.3: There exists a $n^{1/2}\log^{O(1)} n$-approximation algorithm for the crossing number of graphs of bounded degree, with running time $n^{O(\log n/\log\log n)}$. Furthermore there exists a $n^{1/2+\varepsilon}$-approximation algorithm for the crossing number of graphs of bounded degree, with running time $n^{O(1/\varepsilon)}$, for any fixed $\varepsilon > 0$.

B. Related work

In the *$\mathcal{F}$-deletion problem*, the goal is to compute a minimum vertex set S in an input graph G such that $G-S$ is $\mathcal{F}$-minor-free. Characterizing graph properties for which the corresponding vertex deletion problem can be approximated within a constant factor or a polylogarithmic factor is a long standing open problem in approximation algorithms [8], [9]. In spite of a long history of research, we are still far from resolving the status of this problem. Constant-factor approximation algorithms for the vertex Cover problem (i.e., $\mathcal{F} = C_3$) are known since 1970s [10], [11].

Yannakakis [12] showed that approximating the minimum vertex set that needs to be deleted in order to obtain a connected graph with some property P within factor $n^{1-\varepsilon}$ is NP-hard, for a very broad class of properties (see [12]). There was not much progress on approximability/non-approximability of vertex deletion problems until quite recently. Fomin et al. [13] showed that for every graph property P expressible by a finite set of forbidden minors $\mathcal{F}$ containing at least one planar graph, the vertex deletion problem for property P admits a constant factor approximation algorithm. They explicitly mentioned that the most interesting case is when $\mathcal{F}$ contains a non-planar graph (they said that perhaps the most interesting case is when $\mathcal{F} = \{K_{3,3}, K_5\}$), because there is no poly-logarithmic factor approximation algorithm so far. Indeed, the planar graph case and the non-planar case for the family $\mathcal{F}$ may be quite different, as the graph minor theory suggests. The main result of this paper almost settles the most interesting case. We believe that our techniques can lead to further results on approximation algorithms for minor-free properties.

II. High level description of the algorithm

We now give a brief overview of our approach and highlight some of the main challenges. Our approximation algorithm is inspired by fixed-parameter algorithms for the minimum planarization problem, where one assumes that the size of the optimum planarizing set is some fixed constant k (see [3]–[5]).

A. Overview of previous fixed-parameter algorithms.

The known fixed-parameter algorithms for minimum planarization work as follows: If the treewidth of the input graph G is large enough (say, k^c, for some constant $c > 0$), then one can efficiently compute a large grid minor H in G (that is, a minor of G that is isomorphic to some grid). A subgraph J of G is called *flat* if it admits some planar drawing with outer face F, such that all edges between J and $G \setminus J$ have one endpoint in F. If some vertex $v \in H$ is surrounded by a flat subgrid of H of size $\Omega(k)$, then it is *irrelevant*; this means that by removing v, we do not change any optimal solution. Thus, if such an irrelevant vertex v exists, we recurse on $G \setminus \{v\}$, and return the optimum solution found. We define the *face cover* of some set of vertices U to be the minimum number of faces of H that are needed to cover U. If there exists some vertex u such that the neighborhood of u has face cover of size $\Omega(k)$, then u is *universal*; that is, removing u decreases the size of some optimum planarizing set by 1. Thus, if such a universal vertex u exists, we recurse on $G \setminus \{u\}$, and return the optimum solution found, together with u. If the grid H is large enough, then we can always find either an irrelevant or a universal vertex. Thus, by repeatedly removing such vertices, we arrive at a graph of bounded treewidth, where the problem can be solved using standard dynamic programming techniques.

B. Obtaining an approximation algorithm.

We now discuss the main challenges towards extending the above approach to the approximate setting. In order to simplify the exposition, we discuss the $\log^{O(1)}$-approximation algorithm. The n^{ε}-approximation is essentially identical, after changing some parameters.

1. The small treewidth case. In the above fixed-parameter algorithms, the problem is eventually reduced to the bounded-treewidth case. That is, one has to solve the problem on a graph of treewidth $f(k)$, for some function f. Since the optimum k is assumed to be constant, this can be done in polynomial time (in fact, linear time). However, in our setting, k can be as large as $\Omega(n)$, and thus this approach is not applicable. Instead, we try to find some small balanced vertex separator S. If the treewidth is at most $k\log^{O(1)} n$, then we can find some separator of size $k\log^{O(1)} n$. In this case, we recurse on all non-planar connected components of $G \setminus S$, and we add S to the final solution. It can be shown that $|S|$ can be charged to the optimum solution, so that the total increase in the cost of the solution is $k\log^{O(1)}$.

2. The large treewidth case. We say that a graph is k-*apex* if it can be made planar by the removal of at most k vertices. Since any planar graph of treewidth t contains a grid minor of size $\Omega(t) \times \Omega(t)$, it easily follows that any k-apex graph of treewidth $t > ck$, for some universal constant $c > 0$, also contains a grid minor of size $\Omega(t) \times \Omega(t)$. To see that, first delete some planarizing set of size k, and then find a grid minor in the resulting planar graph, which has treewidth at least $t - k$. However, even thought it is trivial to prove the existence of such a large grid minor, computing it in polynomial time when k is not fixed turns out to be a significant challenge. We remark that it is known how to compute a grid minor of size $\Omega(k) \times \Omega(k)$ when

$t = \Omega(k^2)$ [1], [14], and this is enough to obtain a $k^{O(1)}$-approximation algorithm. However, in order to obtain a $\log^{O(1)} n$-approximation, we need to find a grid minor when $t = k \log^{O(1)} n$.

3. Doubly-well-linked sets. The first main technical contribution of this work is an algorithm for computing a large grid minor in k-apex graphs, when k is not fixed. Suppose that the treewidth of G is $t > 2k$. As a first step, we compute some separation (U, U') of order $t \log^{O(1)} n$ (that is, some $U, U' \subset V(G)$ with $V(G) = U \cup U'$ and $|U \cap U'| = t \log^{O(1)} n$), and some $Y \subseteq U \cap U'$ such that Y is *well-linked* in both sides of the separation. Intuitively, for a set Y to be well-linked in some graph G' means that G' does not have any sparse cuts, w.r.t. Y; in other words, contracting G' into Y results in an "expander-like" graph (see Section III for a formal definition). We refer to such a set Y as *doubly-well-linked*. We remark that the notion of doubly-well-linked set considered here is similar to, and inspired by, the *well-linked bipartitions* introduced by Chuzhoy [2] in her work on the crossing number problem. It is well-known that in any graph, such a separation can be found so that Y is well-linked in at least one of the two sides. However, as we explain below, we need Y to be well-linked in both sides of the separation.

4. From a doubly-well-linked set to a grid minor. There are several algorithms for computing large grid minors in planar graphs [15], [16]. A key ingredient in these algorithms is the duality between cuts and cycles in embedded planar graphs. That is, any cut of a planar graph corresponds to a collection of cycles in its dual. The algorithms for the planar case exploit this duality by first computing a well-linked set Z, and then finding some disk $\mathcal{D}$ in the plane, that contains Z and has a large fraction of Z on its boundary. Then, one can find two sets of paths $\mathcal{P}$ and $\mathcal{Q}$, with endpoints on the boundary of $\mathcal{D}$, such that every path in $\mathcal{P}$ intersects every path in $\mathcal{Q}$. By planarity, this yields a grid minor. In our case we cannot apply this idea since we don't have a planar drawing of the graph (indeed, this is precisely what we want to compute). However, it turns out that, intuitively, any doubly-well-linked set Y behaves as a Jordan curve. That is, if we remove any optimal solution from G (that is, any planarizing set of minimum cardinality), then there exists a planar drawing of $G[U]$ such that most of the vertices in Y are close to the outer face. Since Y is well-linked in $G[U]$, we can route in $G[U]$, with low congestion, a multicommodity flow that routes a unit demand between every pair of vertices in Y. We then sample c paths from this flow, for some sufficiently large constant $c > 0$. The fact that the congestion is low, can be used to deduce that the resulting paths will avoid all vertices in some optimal solution, with some constant probability. Thus, the union of the sampled paths admits a planar drawing. Furthermore, since Y is doubly-well-linked, we can show that, with some constant probability, the union of the sampled paths can be drawn so that their endpoints are all in the outer face. We thus use the union of the sampled paths to construct a *skeleton* graph. Specifically, we find a suitable subgraph of G which does not intersect some optimal solution. We then sample $t/\log^{O(1)} n$ more paths from the flow, and partition them into two sets $\mathcal{P}$ and $\mathcal{Q}$, depending on the structure of their intersection with the skeleton graph. Conditioned on the event that the skeleton graph does not intersect some optimal solution, we can find sets of paths $\mathcal{P}$ and $\mathcal{Q}$ such that every path in $\mathcal{P}$ intersects every path in $\mathcal{Q}$.

5. Computing a partially triangulated grid minor. Having computed a large grid minor H, we wish to use H to find either universal or irrelevant vertices. To that end, we need to ensure that there are no edges between different faces of H. We first compute some $X \subset V(G)$, such that $G \setminus X$ can be contracted into some grid H' of size $t/\log^{O(1)} n \times t/\log^{O(1)} n$, where H' is obtained by "eliminating" some rows and columns on H.

6. Computing a semi-universal set. The next main technical challenge in our algorithm is the computation of universal vertices. In the fixed-parameter algorithms described above, in order to compute a universal vertex, one needs a grid of size at least $\Omega(k^2)$. However, in our case, we only have a grid of size $k/\log^{O(1)} n \times k/\log^{O(1)} n$. Thus, we cannot always find a universal vertex. We overcome this obstacle by introducing the notion of a *semi-universal* vertex: We say that a set A of vertices is semi-universal if deleting A from G decreases the cost of the optimum by at least $(2/3)|A|$. We can prove that if the size of the neighborhood of X in H' is at least $k \log^{\Omega(1)} n$, then we can find some $A \subseteq X$ that is semi-universal. Intuitively, the algorithm finds some $A \subseteq X$ that behaves as an expander: for every $A' \subseteq A$, the size of the face cover of A' is at least $|A|/\log^{O(1)} n$.

7. Computing an irrelevant vertex. The next technical difficulty is computing an irrelevant vertex, when the size of the neighborhood of X in H' is at most $k \log^{O(1)} n$. As in the computation of universal vertices, this is a fairly easy task when the treewidth is at least $t = \Omega(k^2)$. However, here we can only find a grid minor of size $k \log^{O(1)} n \times k \log^{O(1)} n$. We overcome this difficulty as follows: We first partition the grid minor into subgrids, in a fashion similar to a quad-tree decomposition: There are $O(\log n)$ partitions of H', each into subgrids of size $2^i \times 2^i$, for all $i \in \{0, \ldots, \log n\}$. Then, for each subgrid in this collection of partitions, we compute an upper estimate on its planarization number: If the number of vertices of G in this subgrid is at least $n/\log n$, then we set the estimate to be k, and otherwise we recursive approximate its minimum planarization number. Finally, we add to this estimate the number of neighbors of X in this subgrid. We say that some subgrid of H' of size $2^i \times 2^i$ is *active* if its upper estimate is at least $2^i/c$, for some constant $c > 0$. Since the size of the neighborhood of X is small, we can show that there exists

some $v \in V(H')$ that lies outsize all active subgrids; we can then show that v must be irrelevant.

8. Embedding into a higher genus surface. Given the above algorithms for computing grid minors, semi-universal vertices, and irrelevant vertices, the algorithm proceeds as follows. We iteratively compute one of the following (1) a small balanced separator, (2) a semi-universal set, or (3) a set of irrelevant vertices. We refer to such a sequence of reductions as a *pruning sequence*. The graph obtained at the end of some pruning sequence is planar. Let X' be the set of vertices removed in Cases (1) and (2), throughout the pruning sequence. We have $|X'| \leq k \log^{O(1)} n$. We say that the *cost* of the pruning sequence is $|X'|$. However, the number of irrelevant vertices removed can be as large as $\Omega(n)$. We need to add these vertices to the planar drawing of the resulting graph. It turns out that this is not always possible. The reason is that the irrelevant vertices removed are only guaranteed to be irrelevant w.r.t. any *optimal* planarizing set; in contrast, the set X' does not form an optimal planarizing set (indeed, we remove $k \log^{O(1)} n$ vertices). We overcome this obstacle using a technique that was first introduced in [1]: When deleting a set of irrelevant vertices, we add a grid of width 3, referred to as a *frame*, around the "hole" that is created. The point of adding this grid is that we can inductively add the irrelevant vertices back to the graph as follows: If X' does not intersect the frame, then we can simply add the irrelevant vertices back to the graph without violating planarity. Otherwise, if the frame intersets m vertices from X', then we can extend the current drawing to the irrelevant vertices corresponding to that frame by adding at most ℓ handles or antihandles. This leads to an embedding into a new non-planar surface. Repeating this process over all frames, we obtain an embedding of $G \setminus X'$ into some surface of Euler genus $k \log^{O(1)}$.

9. The final alorithm. The last remaining step is to compute a planarizing set for $G \setminus X'$. It turns out that this can be done by exploiting the embedding of $G \setminus X'$ into the surface $\mathcal{S}$ of Euler genus $g = k \log^{O(1)} n$, that was computed above. Using tools from the theory of graphs of surfaces, we show that we can decrease the Euler genus of $\mathcal{S}$ by one, while deleting at most $O((1+k/g) \log n)$ vertices. Repeating this process g times, we obtain a planar graph after deleting a set X'' of at most $O(g \log n + k \log^2 n)$ vertices. The final output of the algorithm is $X' \cup X''$, which is a planarizing set for G of size $k \log^{O(1)}$.

C. Organization

The rest of the paper is organized as follows. Section III introduces some basic definitions and results that are used throughout the paper. Section IV presents the main algorithm, by putting together the main ingredients of our approach. Section V presents our algorithm for computing a doulby-well-linked set. Section VI introduces the notion of a *pseudogrid*, which are used to construct grid minors. Section VII presents the algorithm for computing a grid minor. Section VIII gives the algorithm for contracting the graph into a partially triangulated grid, with a small number of apices. Section IX shows how to compute a semi-universal set, given a partially-triangulated grid contraction, such that the apex set has a large neighborhood. Section X shows how to compute irrelevant vertices, for the case where the apex set of the partially triangulated grid contraction has a small neighborhood. Section XI shows how, given an algorithm for computing irrelevant vertices, we can compute a patch. Section XII combines the above algorithms for computing grid minors, semi-universal vertex sets, and patches, to obtain an algorithm for computing a pruning sequence of low cost. Due to lack of space, the algorithm which given a pruning sequence of low cost, computes an embedding into a surface of low Euler genus, as well as the algorithm for planarizing a graph embedded into some non-planar surface, are deferred to the full version[1].

III. Definitions and preliminaries

This section provides some basic notations needed in this paper.

Graph notation: Let G be a graph and $X \subseteq V(G)$. We use d_G to denote the shortest path metric on G. For any $v \in V(G)$ we write $d_G(X, v) = d_G(v, X) = \min_{u \in X} d_G(v, u)$. For any $r \geq 0$ we define $N_G(X, r) = \{v \in V(G) : d_G(v, X) \leq r\}$. For a simple path P and $u, v \in V(P)$ we denote by $P[u, v]$ the subpath of P between u and v. We define the $(r \times \ell)$ grid to be the Cartesian product $P_r \times P_\ell$, where P_i denote the path with i vertices. Let H be the $(r \times \ell)$-grid. For each $v \in V(H)$ we denote by $\mathsf{row}(v) \in \{1, \ldots, r\}$ and by $\mathsf{col}(v) \in \{1, \ldots, \ell\}$ the indexes of the row and column of v respectively. We denote by ∂H the boundary cycle of H. Every face in H other than ∂H is a cycle of length 4. We say that some graph H' is the *partially triangulated* $(r \times \ell)$*-grid* if H' is obtained from H by adding for every face F of H, with $F \neq \partial H$, at most one diagonal edge.

For some graph G, and some $X \subseteq V(G)$ such that $G \setminus X$ is planar, we say that X is *planarizing* (for G). We denote by $\mathsf{mvp}(G)$ the minimum vertex planarization number of G, i.e.

$$\mathsf{mvp}(G) = \min\{|X| : X \subseteq V(G) \text{ is planarizing for } G\}.$$

We remark that deciding whether $\mathsf{mvp}(G) = 0$ is precisely the problem of deciding whether G is planar, which can be solved in linear time [17].

Minors and contractions: A graph H obtained via a sequence of zero or more edge contractions on a graph G is called a *contraction* of G. Unless stated otherwise, we will replace a set of parallel edges in a contraction of a graph by a single edge. The induced mapping $V(H) \to 2^{V(G)}$ is

[1]A full version of this paper is available at http://sidiropoulos.org

called the *contraction mapping* (w.r.t. H and G). Similarly, we say that H is a *minor* of G if it can be obtained via a sequence of zero or more edge contractions, edge deletions, and vertex deletions, performed on G. The induced mapping $V(H) \to 2^{V(G)}$ is called the *minor mapping* (w.r.t. H and G). For any $X \subseteq V(G)$ we also write $\mu(X) = \bigcup_{v \in X} \mu(v)$. For any $v \in V(G)$ we will abuse notation slightly and write $\mu^{-1}(v)$ to denote the unique vertex $v' \in H$ with $v \in \mu(v')$.

Treewidth, pathwidth, and grid minors: A *tree decomposition* of a graph G is a pair (T, R), where T is a tree and R is a family $\{R_t \mid t \in V(T)\}$ of vertex sets $R_t \subseteq V(G)$, such that the following two properties hold:

(W1) $\bigcup_{t \in V(T)} R_t = V(G)$, and every edge of G has both ends in some R_t.

(W2) If $t, t', t'' \in V(T)$ and t' lies on the path in T between t and t'', then $R_t \cap R_{t''} \subseteq R_{t'}$.

The *width* of a tree decomposition (T, R) is $\max\{|R_t| \mid t \in V(T)\} - 1$, and the *treewidth* of G is defined as the minimum width taken over all tree decompositions of G. If T is a path, then we can define the *pathwidth* of G as the minimum width taken over all path decompositions of G. We use $\mathsf{tw}(G)$ and $\mathsf{pw}(G)$ to denote the treewidth and pathwidth of G respectively.

Well-linked sets: Our proof needs to handle a graph of large tree-width. It is well-known that such a graph must have a "highly-connected" subset, which is often referred to as "well-linked".

Definition 3.1 (Cut-linked set): Let G be a graph, let $X \subseteq V(G)$ and $\alpha > 0$. We say that X is *α-cut-linked* (in G), iff for any partition of $V(G)$ into $\{A, B\}$, we have

$$|E(A, B)| \geq \alpha \cdot \min\{|A \cap X|, |B \cap X|\}.$$

Graph on surfaces: A *drawing* of a graph G into a surface $\mathcal{S}$ is a mapping ϕ that sends every vertex $v \in V(G)$ into a point $\phi(v) \in \mathcal{S}$ and every edge into a simple curve connecting its endpoints, so that the images of different edges are allowed to intersect only at their endpoints. The *Euler genus* of a surface $\mathcal{S}$, denoted by $\mathsf{eg}(\mathcal{S})$, is defined to be $2 - \chi(\mathcal{S})$, where $\chi(\mathcal{S})$ is the Euler characteristic of $\mathcal{S}$. This parameter coincides with the usual notion of genus, except that it is twice as large if the surface is orientable. For a graph G, the Euler genus of G, denoted by $\mathsf{eg}(G)$, is defined to be the minimum Euler genus of a surface $\mathcal{S}$, such that G can be embedded into $\mathcal{S}$.

Sparsest-cut and the multi-commodity flow-cut gap:

Definition 3.2 (Sparsest Cut): Consider a graph $G = (V, E)$. The *sparsity* of a cut $(S, \overline{S})$ equals

$$\phi(S) = \frac{E(S, \overline{S})}{\min\{|S|, |\overline{S}|\}}$$

where $\overline{S} = V - S$ and $E(S, \overline{S})$ is the number of cut edges, that is, the number of edges from S to $\overline{S}$. The *Sparsest Cut problem* asks to find a cut $(S, \overline{S})$ with smallest possible sparsity $\phi(S)$.

Let $\alpha \in (0, 1)$. For a graph G, we say that some $X \subseteq V(G)$ is a *α-balanced vertex separator* if every connected component of $G \setminus X$ contains at most $\alpha|V(G)|$ vertices. We also say that some $F \subseteq E(G)$ is a *α-balanced edge separator* if every connected component of $G \setminus F$ contains at most $\alpha|V(G)|$ vertices. The following is the well-known for approximating the sparsest cut. The *uniform sparsest-cut* means the sparsest-cut problem with uniform weight on every edge.

Randomization: Some of the algorithms presented in this paper are randomized. In order to simplify notation, we say that an algorithm succeeds *with high probability* when the failure probability is at most n^{-c}. When the target running time is polynomial, we need c to be some sufficiently large constant. Similarly, when the target running time is $n^{O(\log n / \log\log n)}$, we need $c = \Omega(\log n / \log\log n)$. Both guarantees can be achieved by repeating some algorithm that succeeds with constant probability, either at most $O(\log n)$ or $O(\log^2 n / \log\log n)$ times respectively.

IV. The main algorithm

In this Section we present the main algorithm for approximating minimum planarization. We first introduce some definitions. Intuitively, a patch is a small irrelevant subgraph, that is contained inside some disk in any optimal solution. The framing of a patch is a new graph, that does not contain the interior of the patch, and instead contains grid of constant width attached to the boundary of the patch. Computing a framing in this manner will allow us to extend an approximate solution to the whole graph via an embedding into a higher genus surface.

Definition 4.1 (Patch): Let G be a graph, let $\Gamma \subseteq G$, and let $C \subseteq \Gamma$ be a cycle. Suppose that there exists a planar drawing of Γ having C as the outer face. Then we say that the ordered pair (Γ, C) is a *patch* (of G).

Definition 4.2 (Framing): Let G be a graph and let (Γ, C) be a patch of G. Suppose that $V(C) = \{v_0^0, \ldots, v_{t-1}^0\}$, and $E(C) = \{\{v_0^0, v_1^0\}, \ldots, \{v_{t-2}^0, v_{t-1}^0\}, \{v_{t-1}^0, v_0^0\}\}$. Let G^{framed} be the graph with $V(G^{\mathsf{framed}}) = (V(G) \setminus (V(\Gamma) \setminus V(G))) \cup \bigcup_{i=1}^{t-1} \bigcup_{j=1}^{3} \{v_i^j\}$, where $v_0, \ldots, v'_{t-1}$ are new vertices, and $E(G^{\mathsf{framed}}) = (E(G) \setminus (E(\Gamma) \setminus E(C))) \cup \left(\bigcup_{i=0}^{t-1} \bigcup_{j=0}^{2} \{\{v_i^j, v_i^{j+1}\}, \{v_i^j, v_{i+1 \bmod t}^j\}\}\right)$. We refer to the graph G^{framed} as the *(Γ, C)-framing of G* (see Figure 1 for an example).

Using the above definitions, we can now define the concept of a pruning sequence. Intuitively, this consists of a sequence of operations that inductively simplify the graph until it becomes planar.

Definition 4.3 (Pruning sequence): Let G be a graph. Let $\mathcal{G} = (G_0, A_0), \ldots, (G_\ell, A_\ell)$ be a sequence satisfying the following properties:

(1) For all $i \in \{0, \ldots, \ell\}$, G_i is a graph. Moreover $G_0 = G$ and G_ℓ is planar.

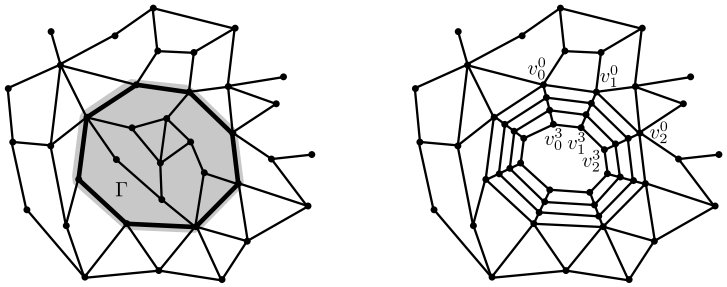

Figure 1. A patch (Γ, C) in a graph G, where the shaded area contains the graph Γ, and the cycle C is drawn in bold (left), and the (Γ, C)-framing of G (right).

(2) For all $i \in \{1, \ldots, \ell\}$, exactly one the following holds:

(2.1) $A_{i-1} \subset V(G_{i-1})$ and $G_i = G_{i-1} \setminus A_{i-1}$. We say that $i-1$ is a *deletion step* (of $\mathcal{G}$).

(2.2) $A_{i-1} = (\Gamma_{i-1}, C_{i-1})$, where (Γ_{i-1}, C_{i-1}) is a patch in G_{i-1}, and G_i is the (Γ_{i-1}, C_{i-1})-framing of G_{i-1}. We say that $i-1$ is a *framing step* (of $\mathcal{G}$).

We also let $A_\ell = \emptyset$.

We say that $\mathcal{G}$ is a *pruning sequence* (for G). We also define the *cost* of $\mathcal{G}$ to be

$$\mathsf{cost}(\mathcal{G}) = \sum_{i \in \{0,\ldots,\ell-1\} : i \text{ is a deletion step}} |A_i|$$

The next Lemma shows how to compute a pruning sequence of low cost. We remark that the algorithm for computing a pruning sequence calls recursively the whole approximation algorithm on graphs of smaller size. By controlling the size of these subgraphs, we obtain a trade-off between the running time and the approximation guarantee. We remark that this is the main technical contribution of this paper. The proof of the next Lemma uses several other results, and spans the majority of the rest of the paper.

Lemma 4.4 (Computing a pruning sequence): Let G be an n-vertex graph, and let $\rho > 0$. Suppose that there exists an algorithm Approx which for all $n' \in \mathbb{N}$, with $n' < n$, given an n'-vertex graph G', outputs some $S' \subset V(G')$, such that $G' \setminus S'$ is planar, with $|S'| \leq \alpha_{\mathsf{approx}} \cdot \mathsf{mvp}(G')$, for some $\alpha_{\mathsf{approx}} \geq 2\rho$, in time $T_{\mathsf{approx}}(n')$, where $T_{\mathsf{approx}} : \mathbb{N} \to \mathbb{N}$ is increasing and convex. Then the algorithm $\mathsf{Pruning}(G)$ returns some pruning sequence for G of cost at most $O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{15} n)$. Moreover $T_{\mathsf{pruning}}(n) \leq n^{O(1)} + T_{\mathsf{pruning}}(n,k)\log n$, and $T_{\mathsf{pruning}}(n,k) \leq n^{O(1)} + \max\{T_{\mathsf{pruning}}(n/4) + T_{\mathsf{pruning}}(3n/4), T_{\mathsf{approx}}(n/\rho)2\rho\log n + T_{\mathsf{pruning}}(n-1,k)\}$, where $T_{\mathsf{pruning}}(n)$ and $T_{\mathsf{pruning}}(n,k)$ denote the worst-case running time of $\mathsf{Pruning}(G)$ and $\mathsf{Pruning}(G,k)$ respectively on a graph of n vertices.

The next Lemma shows that given a pruning sequence of low cost, we can efficiently compute an embedding into a surface of low Euler genus, after deleting a small number of vertices.

Lemma 4.5 (Embedding into a higher genus surface): Let G be a graph and let $\mathcal{G}$ be a pruning sequence for G. Then there exists a polynomial-time algorithm which given G and $\mathcal{G}$ outputs some $X \subseteq V(G)$, with $|X| \leq \mathsf{cost}(\mathcal{G})$, and an embedding of $G \setminus X$ into some surface of Euler genus $O(\mathsf{cost}(\mathcal{G}))$.

The next Lemma gives an efficient algorithm for planarizing a graph embedded into some surface of low Euler genus.

Lemma 4.6 (Planarizing a surface-embedded graph): Let G be a graph, and let ϕ be an embedding of G into some surface of Euler genus $g > 0$. Then there exists a polynomial-time algorithm which given G and ϕ, outputs some planarizing set X for G, with $|X| = O(g\log n + \mathsf{mvp}(G)\log^2 n)$.

We are now ready to present our main results.

Theorem 4.7: Let $\rho > 0$, and $\alpha_{\mathsf{approx}} = \max\{2\rho, O(\log^{32} n)\}$. Then there exists a α_{approx}-approximation algorithm for the minimum vertex planarization problem with running time $T_{\mathsf{approx}}(n) \leq n^{O(1)} + \max\{T_{\mathsf{approx}}(n/4) + T_{\mathsf{approx}}(3n/4), T_{\mathsf{approx}}(n/\rho)2\rho\log n + T_{\mathsf{approx}}(n-1)\}$.

Proof: Let $\rho > 0$ be the parameter from Lemma 4.4, with $\alpha_{\mathsf{approx}} \geq 2\rho$. Using the algorithm from Lemma 4.4 we compute a pruning sequence $\mathcal{F}$ of G with $\mathsf{cost}(\mathcal{G}) = O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{15} n)$. Using the algorithm from Lemma 4.5, in polynomial time, we compute some $X_1 \subset V(G)$, with $|X_1| \leq \mathsf{cost}(\mathcal{G}) \leq O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{15} n)$, and an embedding ϕ of $G \setminus X_1$ into some surface of genus $g = O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{15} n)$. Using the algorithm from Lemma 4.6 we compute some $X_2 \subset V(G) \setminus X_1$, such that $G \setminus (X_1 \cup X_2)$ is planar, with $|X_2| = O(g\log n + \mathsf{mvp}(G \setminus X_1)\log^2 n) = O(g\log n + \mathsf{mvp}(G)\log^2 n) = O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{16} n)$. It follows that $X = X_1 \cup X_2$ is a planarizing set of G, with $|X| = |X_1| + |X_2| = O(\mathsf{mvp}(G)\sqrt{\alpha_{\mathsf{approx}}}\log^{16} n)$. Thus the resulting approximation factor is at most $\alpha_{\mathsf{approx}} = \max\{2\rho, O(\log^{32} n)\}$.

The total running time is dominated by the running time of the algorithm from Lemma 4.4. Thus we

have $T_{\mathsf{approx}}(n) \leq n^{O(1)} + \max\{T_{\mathsf{pruning}}(n/4) + T_{\mathsf{pruning}}(3n/4), T_{\mathsf{approx}}(n/\rho)2\rho\log n + T_{\mathsf{pruning}}(n-1, n-1)\} \leq n^{O(1)} + \max\{T_{\mathsf{pruning}}(n/4) + T_{\mathsf{pruning}}(3n/4), T_{\mathsf{approx}}(n/\rho)2\rho\log n + T_{\mathsf{pruning}}(n-1)\} \leq n^{O(1)} + \max\{T_{\mathsf{approx}}(n/4) + T_{\mathsf{approx}}(3n/4), T_{\mathsf{approx}}(n/\rho)2\rho\log n + T_{\mathsf{approx}}(n-1)\}$, which concludes the proof. ■

By Theorem 4.7, we easily obtain the following results.

Proof of Theorem 1.1: It follows from Theorem 4.7 by setting $\rho = \log n$. ■

Proof of Theorem 1.2: It follows from Theorem 4.7 by setting $\rho = n^{\varepsilon}$. ■

V. Computing a doubly cut-linked set

In this Section we show that when the treewidth is sufficiently large, we can efficiently compute a separation (U, U'), and some large $Y \subseteq U \cap U'$, such that Y is well-linked on both sides of the separation. This result will form the basis for our algorithm for computing a grid minor in the subsequent Sections. Due to lack of space, its proof is deferred to the full version.

Lemma 5.1 (Computing a doubly-cut-linked set): Let G be a n-vertex k-apex graph of treewidth $t > 2k$. Then there exists a polynomial-time algorithm which computes some separation (U, U') of G of order at most $O(t\log^{3/2} n)$ and some $Y \subseteq U \cap U'$ with $|Y| = \Omega(t/\log^{9/2} n)$ such that Y is $\Omega\left(\frac{1}{\log^{13/2} n}\right)$-cut-linked in both $G[U]$ and $G[U']$. Moreover let G_U be the graph obtained from $G[U]$ by removing all edges in $G[U \cap U']$, that is $G_U = G[U] \setminus E(G[U \cap U'])$. Then Y is also $\Omega\left(\frac{1}{\log^{13/2} n}\right)$-cut-linked in G_U.

The above Lemma is used in the algorithm for computing a grid minor, which is deferred to the full version.

VI. Pseudogrids

In this section we introduce the concept of a *pseudogrid*, which will be used in the construction of a grid minor.

Definition 6.1 (Pseudogrid): Let $\mathcal{P}$, $\mathcal{Q}$ be families of paths in some graph. Suppose that all paths in $\mathcal{P}$ are vertex disjoint and all paths in $\mathcal{Q}$ are vertex disjoint. Suppose further that for all $P \in \mathcal{P}$ and $Q \in \mathcal{Q}$ we have $V(P) \cap V(Q) \neq \emptyset$. Then we say that $(\mathcal{P}, \mathcal{Q})$ is a *pseudogrid*.

Definition 6.2 (Compatibility): Let $\mathcal{Q}$ be a collection of vertex-disjoint paths and let P be a directed path in some graph. Suppose that P intersects all paths in $\mathcal{Q}$. We define $\sigma_P^-(\mathcal{Q})$ (resp. $\sigma_P^+(\mathcal{Q})$) to be the total ordering of $\mathcal{Q}$ induced by P as follows: we traverse P and we order the paths in $\mathcal{Q}$ according to the first (resp. last) time they are visited. We say that P is *compatible* with $\mathcal{Q}$ if $\sigma_P^-(\mathcal{Q}) = \sigma_P^+(\mathcal{Q})$. If P is compatible with $\mathcal{Q}$ then we also write $\sigma_P(\mathcal{Q}) = \sigma_P^-(\mathcal{Q}) = \sigma_P^+(\mathcal{Q})$.

Definition 6.3 (Combed pseudogrids): Let $(\mathcal{P}, \mathcal{Q})$ be a pseudogrid. Suppose that all $P \in \mathcal{P}$ are compatible with $\mathcal{Q}$ and all $Q \in \mathcal{Q}$ are compatible with $\mathcal{P}$. Suppose further that for all $P, P' \in \mathcal{P}$ we have $\sigma_P(\mathcal{Q}) = \sigma_{P'}(\mathcal{Q})$ and for all $Q, Q' \in \mathcal{Q}$ we have $\sigma_Q(\mathcal{P}) = \sigma_{Q'}(\mathcal{P})$. Then we say that $(\mathcal{P}, \mathcal{Q})$ is *combed*.

VII. Computing a grid minor

In this section, we present one of the key procedures used by our algorithm. Namely, we give an algorithm for computing a grid minor in graphs of large treewidth. Our result is summarized in the following Corollary. Due to lack of space, its proof is deferred in the full version.

Corollary 7.1: There exists a randomized polynomial-time algorithm which given a graph G and $k, t \in \mathbb{N}$, with $t > \alpha' k \log^6 n$, for some universal constant $\alpha' > 0$, terminates with one of the following outcomes, with high probability:

(1) Correctly decides that $\mathsf{mvp}(G) > k$.

(2) Outputs some $3/4$-separator S of G with $|S| = O(t\log^{13/2} n)$.

(3) Outputs some combed pseudogrid $(\mathcal{P}, \mathcal{Q})$ in G, with $|\mathcal{P}| = |\mathcal{Q}| = r$, for some $r = \Omega(t/\log^4 n)$.

VIII. Computing a partially triangulated grid contraction with a few apices

In this section, we shall construct a partially triangulated grid contraction, after deleting a small set of vertices. The main results is summarized in the following.

Lemma 8.1: There exists a randomized polynomial-time algorithm, which given some graph G, some $k \in \mathbb{N}$, and some simple combed pseudogrid $(\mathcal{P}, \mathcal{Q})$ in G, with $|\mathcal{P}| = |\mathcal{Q}| = r$, where $r \geq ck\log n$, for some universal constant $c > 0$, terminates with one of the following outcomes, with high probability:

(1) Correctly decides that $\mathsf{mvp}(G) > k$.

(2) Outputs some $A \subset V(G)$, with $|A| = O(k\log n)$, some $(r'' \times r'')$-partially triangulated grid W'', for some $r'' \geq r/8$, and some contraction mapping $\xi' : V(W'') \to 2^{V(G)\setminus A}$.

Due to lack of space, the proof is deferred to the full version.

IX. Computing semi-universal vertex sets

In this Section we present our algorithm for computing semi-universal vertex sets.

Definition 9.1 (Semi-universality): Let G be a graph and let $Y \subset V(G)$. We say that Y is *semi-universal* (w.r.t. G) if for all $S \subseteq V(G)$, with $|S| = \mathsf{mvp}(G)$, such that $G \setminus S$ is planar, we have

$$|Y \cap S| \geq (2/3) \cdot |Y|.$$

The following summarizes the main result of this section.

Lemma 9.2 (Computing a semi-universal set): Let G be a graph, and let $k \in \mathbb{N}$. Let $X \subset V(G)$, and let $\mu : V(H) \to 2^{V(G)\setminus X}$ be a contraction of $G \setminus X$, where H is the $(r \times r)$-partially triangulated grid, for some

$r > 0$. Let $\mu' : V(H') \to 2^{V(G)}$ be the contraction of G induced by μ, where $V(H') = V(H) \cup X$, and μ' is the identity on X. Let $L = |N_{H'}(X)| = |\{v \in V(H) : d_H(v, \partial H) \geq 3 \text{ and } N_G(X) \cap \mu'(v) \neq \emptyset\}|$. Suppose that $L > (144|X| + 1296k)c_{FHL} \log^{3/2} n$. Then there exists a polynomial-time algorithm which given G, X, H, and μ, computes some non-empty $Y \subseteq X$, satisfying the following property: If $\mathsf{mvp}(G) \leq k$, then Y is semi-universal (w.r.t. G).

Due to lack of space, the proof is deferred to the full version.

X. Computing irrelevant vertex sets

In this Section we present our result for computing irrelevant vertices. For some $v \in V(G)$, we say that v is *irrelevant* if for any $X \subset V(G)$, with $|X| = \mathsf{mvp}(G)$, such that $G \setminus X$ is planar, we have $v \notin X$. Similarly, we say that some $U \subset V(G)$ is *irrelevant* if for all $v \in V(G)$, v is irrelevant. The main result of this Section is summarized in the following.

Lemma 10.1 (Computing an irrelevant subgrid): Let G be an n-vertex graph, and let $k \in \mathbb{N}$, and let $\rho > 0$. Let $X \subset V(G)$, with $|X| \leq ck \log n$, for some universal constant $c > 0$. Let H be the $(r \times r)$-partially triangulated grid, for some $r \geq c' \cdot (\log^{7/2} n) \cdot \sqrt{\alpha_{\mathsf{approx}}} \cdot k$, for some universal constant $c' > 0$. Suppose that H is a contraction of $G \setminus X$, with contraction mapping $\mu : V(H) \to 2^{V(G \setminus X)}$. Let $\mu' : V(H') \to 2^{V(G)}$ be the contraction of G induced by μ, where $V(H') = V(H) \cup X$, and μ' is the identity on X. Let $L = |N_{H'}(X)| = |\{v \in V(H) : d_H(v, \partial H) \geq 3 \text{ and } N_G(X) \cap \mu'(v) \neq \emptyset\}|$. Suppose that $L \leq (144|X|+1296k)c_{FHL} \log^{3/2} n$, Suppose that there exists an algorithm Approx which for all $n' \in \mathbb{N}$, with $n' < n$, given an n'-vertex graph G', outputs some $S' \subset V(G')$, such that $G' \setminus S'$ is planar, with $|S'| \leq \alpha_{\mathsf{approx}} \cdot \mathsf{mvp}(G')$, for some $\alpha_{\mathsf{approx}} \geq 2\rho$, in time $T_{\mathsf{approx}}(n')$, where $T_{\mathsf{approx}} : \mathbb{N} \to \mathbb{N}$ is increasing and convex. Then, there exists an algorithm $\mathsf{Irrelevant}$ which given G, k, X, μ, H, and L, terminates with one of the following outcomes:

(1) Correctly decides that $\mathsf{mvp}(G) > k$.

(2) Outputs some $(6 \log n \times 6 \log n)$-partially triangulated subgrid J of H, such that if $\mathsf{mvp}(G) \leq k$, then $\mu(V(J))$ is irrelevant.

Moreover, the running time of $\mathsf{Irrelevant}$ is at most $T_{\mathsf{irrelevant}}(n) \leq n^{O(1)} + T_{\mathsf{approx}}(n/\rho)2\rho \log n$.

Due to lack of space, the proof is deferred to the full version.

XI. Patches and frames

In this Section we use the algorithm for computing irrelevant vertices to compute a patch that can be used in the computation of a pruning sequence. A key desired property is that the framing of the patch must be smaller than the graph before the framing.

Lemma 11.1 (Computing a patch): Let G, n, ρ, k, X, μ, H, and L be as in Lemma 10.1. Suppose that there exists an algorithm Approx which for all $n' \in \mathbb{N}$, with $n' < n$, given an n'-vertex graph G', outputs some $S' \subset V(G')$, such that $G' \setminus S'$ is planar, with $|S'| \leq \alpha_{\mathsf{approx}} \cdot \mathsf{mvp}(G')$, for some $\alpha_{\mathsf{approx}} \geq 2\rho$, in time $T_{\mathsf{approx}}(n')$, where $T_{\mathsf{approx}} : \mathbb{N} \to \mathbb{N}$ is increasing and convex. Then, there exists an algorithm Patch, which given G, k, X, μ, H, and L, terminates with one of the following outcomes:

(1) Correctly decides that $\mathsf{mvp}(G) > k$.

(2) Computes some patch (Γ, C) of G, satisfying the following conditions: Let G^{framed} be the (Γ, C)-framing of G. Then $|V(G^{\mathsf{framed}})| < |V(G)|$. Moreover, if $\mathsf{mvp}(G) \leq k$, then $\mathsf{mvp}(G^{\mathsf{framed}}) \leq \mathsf{mvp}(G)$.

Moreover, the running time of Patch is at most $T_{\mathsf{patch}}(n) \leq n^{O(1)} + T_{\mathsf{approx}}(n/\rho)2\rho \log n$.

Due to lack of space, the proof is deferred to the full version.

XII. Computing a pruning sequence

In the previous section, we find irrelevant vertices. This allows us to define the following "pruning sequence" in our algorithm.

The main result of this section is to compute a pruning sequence, which will be presented in the next subsection.

A. The algorithm for computing a pruning sequence

We now describe an algorithm for computing a pruning sequence for a given graph G. We give two algorithms: $\mathsf{Pruning}$ and $\mathsf{Pruning\text{-}Decision}$. The algorithm $\mathsf{Pruning}$ gets as input some graph G and outputs a pruning sequence for G. The algorithm $\mathsf{Pruning\text{-}Decision}$ gets as input some graph G and some $k \in \mathbb{N}$, and either returns nil, which indicates the fact that $\mathsf{mvp}(G) > k$, or outputs some pruning sequence for G. The algorithm $\mathsf{Pruning}$ calls recursively algorithm $\mathsf{Pruning\text{-}Decision}$, and $\mathsf{Pruning\text{-}Decision}$ calls recursively either itself or $\mathsf{Pruning}$.

The algorithms use a parameter $\rho > 0$, which is as in Lemmas 10.1 and 11.1. The parameter ρ which allows us to obtain a trade-off between the approximation ratio and the running time. In particular, the approximation ratio of the final algorithm increases and the running time decreases when ρ increases. We also use a parameter $\alpha_{\mathsf{approx}} > 0$ which denotes the target approximation factor.

The formal description of algorithm $\mathsf{Pruning}$ is as follows.

Algorithm $\mathsf{Pruning}(G)$:

Step 1: The main loop. We consider all values $k = 1, 2, 4, \ldots, 2^i, \ldots, n$, which, intuitively, are used as "approximate guesses" for $\mathsf{mvp}(G)$. For each such value of k, we execute $\mathsf{Pruning\text{-}Decision}(G, k)$. If the execution returns nil then repeat Step 1 for the next value of k. Otherwise, we output the resulting pruning sequence found.

This completes the description of the algorithm Pruning. We next give the formal description of the algorithm Pruning-Decision.

Algorithm Pruning-Decision(G, k):

Step 1: Let $t = c'' \cdot \log^{15/2} n \cdot \sqrt{\alpha_{\mathsf{approx}}} \cdot k$, for some sufficiently large constant $c'' > 0$ to be determined. We have $t > \alpha' k \log^6 n$, where $\alpha' > 0$ is the universal constant in Corrolary 7.1. Using Corrolary 7.1, in polynomial time, we obtain one of the following outcomes:

Case 1: We correctly decide that $\mathsf{mvp}(G) > k$. In this case, we return nil.

Case 2: Computing a small balanced separator. We compute a $3/4$-balanced separator S of G, with $|S| = O(t \log^{13/2} n)$. In this case we partition $G \setminus S$ into two vertex-disjoint subgraphs G_1 and G_2 such that there are no edges between $V(G_1)$ and $V(G_2)$, and with $|V(G_1)| \leq 3n/4$, and $|V(G_2)| \leq 3n/4$. We recursively call Pruning(G_1) and Pruning(G_2), and we obtain pruning sequences $\mathcal{G}_1 = (G_{1,0}, A_{1,0}), \ldots (G_{1,\ell}, A_{1,\ell})$ for G_1, and $\mathcal{G}_2 = (G_{2,0}, A_{2,0}), \ldots (G_{2,\ell'}, A_{2,\ell'})$ for G_2. We output $\mathcal{G} = (G_{1,0} \cup G_{2,0}, A_{1,0}), \ldots, (G_{1,\ell} \cup G_{2,0}, A_{1,\ell}), (G_{1,\ell} \cup G_{2,1}, A_{2,0}), \ldots, (G_{1,\ell} \cup G_{2,\ell'}, A_{2,\ell'})$. Since $G_{1,\ell}$ and $G_{2,\ell'}$ are planar graphs, it follows that $G_{1,\ell} \cup G_{2,\ell'}$ is planar, and thus $\mathcal{G}$ is a pruning sequence for G. If $\mathsf{cost}(\mathcal{G}) > c''' k \sqrt{\alpha_{\mathsf{approx}}} \log^{15} n$, for some sufficiently large constant $c''' > 0$ to be specified, then we return nil, and otherwise we return $\mathcal{G}$.

Case 3: Computing a large grid minor. We compute some combed pseudogrid $(\mathcal{P}, \mathcal{Q})$ in G, with $|\mathcal{P}| = |\mathcal{Q}| = r$, for some $r = \Omega(t/\log^4 n)$. Setting c'' to be some sufficiently large constant, we get that $r \geq ck \log n$, where c is the universal constant from Lemma 8.1, and thus the conditions of Lemma 8.1 are satisfied. Using Lemma 8.1, in polynomial time, we obtain one of the following outcomes:

Case 3.1: We can correctly decide that $\mathsf{mvp}(G) > k$. In this case, we return nil.

Case 3.2: Computing a partially triangulated grid contraction with a few apices. We compute some $X \subset V(G)$, with $|X| = O(k \log n)$, some $(r' \times r')$-partially triangulated grid H, for some $r' \geq r/8$, and some contraction mapping $\mu : V(H) \to 2^{V(G \setminus X)}$. Let $\mu' : V(H') \to 2^{V(G)}$ be the contraction mapping of G induced by μ, where $V(H') = V(H) \cup X$, and μ' is the identity on X. Let $L = |N_{H'}(X)| = |\{v \in V(H) : d_H(v, \partial H) \geq 3 \text{ and } N_G(X) \cap \mu'(v)\}|$. We consider the following two cases:

Case 3.2.1: Computing a semi-universal set. Suppose that $L > (144|X| + 1296k) c_{FHL} \log^{3/2} n$. Then using Lemma 9.2 we can compute, in polynomial time, some non-empty $Y \subseteq X$, such that if $\mathsf{mvp}(G) \leq k$, then Y is semi-universal. We recursively call Pruning-Decision$(G \setminus Y, k)$. If the recursive call returns nil then we return nil. Otherwise, we obtain a pruning sequence $(G_0, A_0), \ldots, (G_\ell, A_\ell)$ for $G \setminus Y$. We define the pruning sequence $\mathcal{G} = (G, Y), (G_0, A_0), \ldots, (G_\ell, A_\ell)$ for G. If $\mathsf{cost}(\mathcal{G}) > c''' k \sqrt{\alpha_{\mathsf{approx}}} \log^{15} n$, for some sufficiently large constant $c''' > 0$ to be specified, then we return nil, and otherwise we return $\mathcal{G}$.

Case 3.2.2: Computing an irrelevant patch. Suppose that $L \leq (144|X| + 1296k) c_{FHL} \log^{3/2} n$. By setting c'' to be a sufficiently large constant, we get that $r' \geq c' (\log^{7/2} n) \sqrt{\alpha_{\mathsf{approx}}} k$, where c' is the universal constant from Lemma 11.1, and thus the conditions of Lemma 11.1 are satisfied. Using the algorithm from Lemma 11.1, we obtain one of the following two outcomes: (i) We either decide that $\mathsf{mvp}(G) > k$; in this case, we go to Step 1, and consider the next value for k. (ii) We compute some patch (Γ, C) of G, satisfying the following conditions: Let G^{framed} be the (Γ, C)-framing of G. Then $|V(G^{\mathsf{framed}})| < |V(G)|$. Moreover, if $\mathsf{mvp}(G) \leq k$, then $\mathsf{mvp}(G^{\mathsf{framed}}) \leq \mathsf{mvp}(G)$. We recursively call Pruning-Decision(G^{framed}, k). If the recursive call returns nil, then we return nil. Otherwise, we obtain a pruning sequence $(G_0, A_0), \ldots, (G_\ell, A_\ell)$ for G^{framed}. We return the pruning sequence $\mathcal{G} = (G, (\Gamma, C)), (G_0, A_0), \ldots, (G_\ell, A_\ell)$ for G.

This completes the description of the algorithm Pruning-Decision. Due to lack of space, the analysis, given as the proof of Lemma 4.4, is deferred to the full version.

ACKNOWLEDGMENT

Ken-ichi Kawarabayashi is supported by the JST ERATO Kawarabayashi Large Graph Project. Anastasios Sidiropoulos is supported by the NSF under grant CCF 1423230 and award CAREER 1453472.

REFERENCES

[1] C. Chekuri and A. Sidiropoulos, "Approximation algorithms for euler genus and related problems," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*. IEEE, 2013, pp. 167–176.

[2] J. Chuzhoy, "An algorithm for the graph crossing number problem," in *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, L. Fortnow and S. P. Vadhan, Eds. ACM, 2011, pp. 303–312. [Online]. Available: http://doi.acm.org/10.1145/1993636.1993678

[3] K. Kawarabayashi, "Planarity allowing few error vertices in linear time," in *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*. IEEE Computer Society, 2009, pp. 639–648. [Online]. Available: http://dx.doi.org/10.1109/FOCS.2009.45

[4] D. Marx and I. Schlotter, "Obtaining a planar graph by vertex deletion," *Algorithmica*, vol. 62, no. 3-4, pp. 807–822, 2012. [Online]. Available: http://dx.doi.org/10.1007/s00453-010-9484-z

[5] B. M. P. Jansen, D. Lokshtanov, and S. Saurabh, "A near-optimal planarization algorithm," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, C. Chekuri, Ed. SIAM, 2014, pp. 1802–1811. [Online]. Available: http://dx.doi.org/10.1137/1.9781611973402.130

[6] M. Chimani and P. Hlinený, "A tighter insertion-based approximation of the crossing number," *J. Comb. Optim.*, vol. 33, no. 4, pp. 1183–1225, 2017. [Online]. Available: http://dx.doi.org/10.1007/s10878-016-0030-z

[7] J. Chuzhoy, Y. Makarychev, and A. Sidiropoulos, "On graph crossing number and edge planarization," in *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, D. Randall, Ed. SIAM, 2011, pp. 1050–1069. [Online]. Available: http://dx.doi.org/10.1137/1.9781611973082.80

[8] M. Yannakakis, "Some open problems in approximation," in *Proceedings of the 2nd Italian Conference on Algorithms and Complexity (CIAC 1994)*, vol. 778, 1994, pp. 33–39.

[9] C. Lund and M. Yannakakis, "The approximation of maximum subgraph problems," in *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*, ser. Lecture Notes in Computer Science, A. Lingas, R. G. Karlsson, and S. Carlsson, Eds., vol. 700. Springer, 1993, pp. 40–51. [Online]. Available: http://dx.doi.org/10.1007/3-540-56939-1_60

[10] R. Bar-Yehuda and S. Even, "A linear-time approximation algorithm for the weighted vertex cover problem," *J. Algorithms*, vol. 2, pp. 198–203, 1981.

[11] G. L. Nemhauser and L. E. Trotter, Jr., "Properties of vertex packing and independence system polyhedra," *Math. Programming*, vol. 6, pp. 48–61, 1974.

[12] M. Yannakakis, "The effect of a connectivity requirement on the complexity of maximum subgraph problems," *J. ACM*, vol. 26, pp. 618–630, 1979.

[13] F. Fomin, D. Lokshtanov, N. Misra, and S. Saurabh, "Planar f-deletion: Approximation, kernelization and optimal fpt algorithms," in *Foundations of Computer Science(FOCS'12)*, 2012, pp. 470–479.

[14] K. Kawarabayashi and A. Sidiropoulos, "Beyond the euler characteristic: Approximating the genus of general graphs," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, R. A. Servedio and R. Rubinfeld, Eds. ACM, 2015, pp. 675–682. [Online]. Available: http://doi.acm.org/10.1145/2746539.2746583

[15] C. Chekuri, S. Khanna, and F. B. Shepherd, "Edge-disjoint paths in planar graphs," in *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*. IEEE, 2004, pp. 71–80.

[16] N. Robertson, P. D. Seymour, and R. Thomas, "Quickly excluding a planar graph," *J. Comb. Theory, Ser. B*, vol. 62, no. 2, pp. 323–348, 1994.

[17] J. E. Hopcroft and R. E. Tarjan, "Efficient planarity testing," *J. ACM*, vol. 21, no. 4, pp. 549–568, 1974. [Online]. Available: http://doi.acm.org/10.1145/321850.321852