2013 IEEE 54th Annual Symposium on Foundations of Computer Science

Approximation algorithms for Euler genus and related problems

Chandra Chekuri*
Dept. of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
chekuri@illinois.edu

Anastasios Sidiropoulos†
Dept. of Computer Science and Engineering
Ohio State University
Columbus, OH 43210
sidiropo@gmail.com

***Abstract*—The Euler genus of a graph is a fundamental and well-studied parameter in graph theory and topology. Computing it has been shown to be NP-hard by Thomassen [23], [24], and it is known to be fixed-parameter tractable. However, the approximability of the Euler genus is wide open. While the existence of an $O(1)$-approximation is not ruled out, only an $O(\sqrt{n})$-approximation [3] is known even in bounded degree graphs. In this paper we give a polynomial-time algorithm which on input a bounded-degree graph of Euler genus g, computes a drawing into a surface of Euler genus $g^{O(1)} \cdot \log^{O(1)} n$. Combined with the upper bound from [3], our result also implies a $O(n^{1/2-\alpha})$-approximation, for some constant $\alpha > 0$.**

Using our algorithm for approximating the Euler genus as a subroutine, we obtain, in a *unified* fashion, algorithms with approximation ratios of the form $\mathsf{OPT}^{O(1)} \cdot \log^{O(1)} n$ for several related problems on bounded degree graphs. These include the problems of orientable genus, crossing number, and planar edge and vertex deletion problems. Our algorithm and proof of correctness for the crossing number problem is simpler compared to the long and difficult proof in the recent breakthrough by Chuzhoy [5], while essentially obtaining a qualitatively similar result. For planar edge and vertex deletion problems our results are the first to obtain a bound of form $\mathrm{poly}(\mathsf{OPT}, \log n)$.

We also highlight some further applications of our results in the design of algorithms for graphs with small genus. Many such algorithms require that a drawing of the graph is given as part of the input. Our results imply that in several interesting cases, we can implement such algorithms even when the drawing is unknown.

I. Introduction

A *drawing* of a graph G into a surface $\mathcal{S}$ is a mapping that sends every vertex $v \in V(G)$ into a point $\varphi(v) \in \mathcal{S}$, 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. In this paper we deal with closed surfaces (compact and without boundary). A surface $\mathcal{S}$ can be orientable or non-orientable. The *Euler genus* $\mathsf{eg}(\mathcal{S})$ of a nonorientable surface $\mathcal{S}$ is defined to be the nonorientable genus of $\mathcal{S}$. The Euler genus $\mathsf{eg}(\mathcal{S})$ of an orientable surface $\mathcal{S}$ is equal to 2γ, where γ is the orientable genus of $\mathcal{S}$. For a graph G, the Euler genus of G, denoted by $\mathsf{eg}(G)$, is defined to be equal to the infimal Euler genus of a surface $\mathcal{S}$, such that G can be drawn into $\mathcal{S}$. The orientable genus of a graph G, denoted by $\mathsf{genus}(G)$, is the infimal genus of an orientable surface $\mathcal{S}$ into which G can be drawn.

Drawings of graphs into various surfaces are of central importance in graph theory [18], topology, and mathematics in general, and have been the subject of intensive study. In particular, surface embedded graphs are an important ingredient in the seminal work of Robertson and Seymour on graph minors and the proof of Wagner's conjecture. Surface embedded graphs are also important in computer science, and engineering, since they can be used to model a wide variety of natural objects.

We consider two simple-to-state and fundamental optimization problems in topological graph theory: given a graph G, compute $\mathsf{eg}(G)$ and $\mathsf{genus}(G)$. Thomassen [23] showed that computing these quantities is NP-hard. Deciding whether a graph has Euler genus 0, i.e. planarity testing, can be done in linear time by the seminal result of Hopcroft & Tarjan [10]. Deciding if $\mathsf{eg}(G) \leq g$ is fixed-parameter tractable. In fact, Mohar [16] gave a linear time algorithm for this problem, and subsequently a relatively simple linear-time algorithm was given by Kawarabayashi, Mohar & Reed [11]. The dependence of the running time in the above mentioned algorithms is at least exponential in g. We note that, for any fixed g, the set of all graphs with genus at most g, denoted by $\mathcal{G}_g$ is minor-closed. From the work of Robertson and Seymour [21], $\mathcal{G}_g$ is characterized as the class of graphs that exclude as a minor all graphs from a finite family of graphs $\mathcal{H}_g$. However, $\mathcal{H}_g$ is not known explicitly even for small values of g and $|\mathcal{H}_g|$ can be very large.

In this paper we consider the case when g is not a fixed constant and examine the *approximability* of $\mathsf{eg}(G)$ and $\mathsf{genus}(G)$. Perhaps surprisingly, this problem is very poorly understood. We briefly describe the known results and illustrate the technical difficulties. In general, $\mathsf{eg}(G)$ can be as large as $\Omega(n^2)$ where n is the number of nodes of G (e.g. for the complete graph K_n), and Euler's characteristic implies that any n-vertex graph of Euler genus g has at most $O(n+g)$ edges. Since any graph can be drawn into a surface that has one handle for every edge, this immediately implies

*Supported in part by NSF grant CCF-1016684.
†Work done while author was a post-doctoral scholar at Univ. of Illinois, Urbana-Champaign. Supported in part by David and Lucille Packard Fellowship, NSF AF award CCF-0915984, and NSF grant CCF-0915519.

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.26

an $O(n/g)$-approximation, which is a $\Theta(n)$-approximation in the worst case. In other words, even though we currently cannot exclude the existence of an $O(1)$-approximation, the state of the art only gives a trivial $O(n)$-approximation. Using the fact that graphs of small genus have small balanced vertex-separators, Chen, Kanchi & Kanevsky [3] obtained a simple $O(\sqrt{n})$-approximation for graphs of *bounded degree* which is currently the best known approximation ratio. In fact, if we do not assume bounded degree, nothing better than the trivial $O(n/g)$-approximation is known. Consider the case of apex graphs which are graphs that contain a single vertex whose removal makes them planar. Mohar [17] showed that the genus problem for even these graphs is NP-hard. He also gave an elegant characterization of the genus for apex graphs, which can in turn be used to obtain a $O(1)$-approximation for such graphs. It is worth mentioning that essentially nothing is known for graphs with a constant number of apices! We also remark that by Euler's formula, there is a trivial $O(1)$-approximation if the average degree is at least $6 + \varepsilon$, for some fixed $\varepsilon > 0$. Finally, we mention a recent result by Makarychev, Nayyeri & Sidiropoulos [14], who gave an algorithm that given a Hamiltonian graph G along with a Hamiltonian path P, draws the graph into a surface of Euler genus $g^{O(1)} \log^{O(1)} n$ where g is the orientable genus of G. We note that their algorithm does not assume bounded degree which is its strength but assumes Hamiltonicity which is a limitation. Moreover, the techniques in [14] rely heavily on using the given Hamiltonian path P while our techniques here are based on treewidth related ideas among several others.

Our algorithms for approximating genus also give us, in a unified fashion, algorithms for two related problems on drawing a graph on a planar surface, namely crossing number and planar edge/vertex deletion. The guarantees of our algorithms are of the form $\mathsf{OPT}^{O(1)} \log^{O(1)} n$. These problems have also been well-studied and have the common feature that the know hardness results are weak (either NP-Hardness or APX-Hardness) while known approximation bounds are polynomial in n even in bounded-degree graphs. In this context we make some remarks on why the bounded-degree assumption is interesting despite being a limitation in some ways. First, we can assume that the graph has bounded average degree since otherwise the lower bound on the instance is very high and it becomes easy to approximate (see previous comment on genus). It is not uncommon in applications such as VLSI design and graph layout to assume some form of an upper bound on the degree; heuristically algorithms that work for bounded degree graphs can be extended to handle the case of graphs that can be made bounded degree by the removal of a small number of edges. Second, from a theoretical point of view, understanding the approximability even when all degrees are bounded (in fact at most 3) is non-trivial and there has been very limited progress over two decades. It is only very recently that Chuzhoy, in a breaktrough and technically difficult work, obtained a bound of the form $\mathsf{OPT}^{O(1)} \log^{O(1)} n$ for crossing number problem in bounded degree graphs. We now describe our results formally.

Our results. Our main result is an approximation algorithm for the Euler genus of bounded degree graphs. More specifically, given a graph G of Euler genus g, our algorithm computes a drawing of G into a surface of Euler genus $\Delta^{O(1)} g^{O(1)} \log^{O(1)} n$ where Δ is the maximum degree. The algorithm's running time is polynomial in both g and n. Combined with the simple $O(n^{1/2})$-approximation from [3], our result gives a $O(n^{1/2-\alpha})$-approximation for some fixed constant $\alpha > 0$. The following theorem summarizes our main result.

Theorem I.1 (Main result)**.** *There is a polynomial-time algorithm which given a graph G of maximum degree Δ, and an integer $g \geq 0$, either outputs a drawing of G into a surface of Euler genus $O(\Delta^2 g^{12} \log^{19/2} n)$, or correctly decides that the Euler genus of G is greater than g.*

Remark I.1. *Kawarabayashi, Mohar and Reed [11] claim an exact algorithm to compute the Euler genus of a given graph in time $2^{O(\mathsf{OPT})} n$ time, which in particular implies a polynomial-time algorithm when $\mathsf{OPT} = O(\log n)$; this simplifies and improves a previous linear-time algorithm of Mohar [16] which had a doubly-exponential dependence on OPT. Theorem I.1, when combined with the algorithm in [11], implies a polynomial-time algorithm that given a graph G outputs a drawing on a sufrace with Euler genus $O(\Delta^3 \mathsf{OPT}^{O(1)})$.*

We build on our main result to obtain several other non-trivial results; we describe the outline of the unified methodology in Section I-B. First, we obtain an algorithm to approximate $\mathsf{genus}(G)$, the orientable genus of a given graph G, summarized in the theorem below. Note that $\mathsf{genus}(G)$ can be $\Omega(\sqrt{|V(G)|})$ even when $\mathsf{eg}(G) = O(1)$ [9].

Theorem I.2 (Approximating the orientable genus)**.** *There exists a polynomial-time algorithm which given a graph G of maximum degree Δ, and an integer $g > 0$, either correctly decides that $\mathsf{genus}(G) > g$, or outputs a drawing of G into a surface of orientable genus $O(\Delta^3 g^{14} \log^{19/2} n)$.*

Crossing number. In the crossing number problem the input is a graph G which may not be planar and the goal is to draw it into the Euclidean plane with as few edge crossings as possible. When we deal with this problem, we will allow the edges in a graph drawing to intersect in their interiors. The point where the interiors of two edges intersect, is called a *crossing* of the drawing. We do not allow the interiors of edges to intersect vertices, and we also assume that there are no three edges, with their interiors intersecting at the same point. The *crossing number* of a graph G, denoted by $\mathsf{cr}(G)$, is defined to be the smallest integer k, such that G admits

a drawing into the plane, with at most k crossings.

The crossing number problem has also been a difficult problem to approximate, and the focus has been primarily on bounded degree graphs. It is an NP-hard problem but for each fixed k there is a linear time algorithm to decide if $\mathsf{cr}(G) \leq k$ [12]. In a recent breakthrough paper, Chuzhoy [5] described an algorithm that given a graph G outputs a drawing into the plane with $O(\mathsf{cr}(G)^{10}\,\mathrm{poly}(\Delta \log n))$ crossings; as a corollary she obtained the first algorithm that had an approximation ratio that is sub-linear in $|V(G)|$. The algorithm and proof in [5] occupy almost 80 pages. It is a simple observation that if the crossing number of a graph G is k then $\mathsf{genus}(G) \leq k$ since one can add a handle for each edge that participates in a crossing. We can apply our approximation algorithm to find a drawing of G into an orientable surface, via Theorem I.2, of genus $O(\Delta^4 k^9 \log^{13/2} n)$. Interestingly, having a drawing on a relatively low genus surface, allows us to obtain a rather simple algorithm for crossing number. Our result is summarized below.

Theorem I.3 (Approximating the crossing number)**.** *There exists a polynomial-time algorithm which given a graph G of maximum degree Δ, and an integer $k \geq 0$, either correctly decides that $\mathsf{cr}(G) > k$, or outputs a drawing of G into the plane with at most $O(\Delta^9 k^{30} \log^{19} n)$ crossings.*

We note that the dependence on k in our theorem is worse than that in [5]. However, we believe that our approach, in addition to giving a simpler proof, is interesting because it appears to differ from that in [5] in going via a somewhat indirect route through a low genus drawing. We refer the interested reader to [4]–[6] for various pointers to the extensive work on crossing number and related problems.

Planar Edge and Vertex Deletion. We extend our approach via genus to obtain an approximation algorithm for the minimum planar edge/vertex deletion problems. In these problems we are given a graph G and the goal is to remove the smallest number of edges/vertices to make it planar. We denote by $\mathsf{edge\text{-}planarization}(G)$ the minimum size of such a planarizing set of edges and similarly by $\mathsf{vertex\text{-}planarization}(G)$ for vertices. The best known approximation for this problems is $O(\sqrt{n \log n})$ due to Tragoudas via the separator algorithms [13], and recently Chuzhoy [5] gives an algorithm that outputs a solution of size $O(\mathsf{cr}(G)^5\,\mathrm{poly}(\Delta \cdot \log n))$; we observe that the $\mathsf{cr}(G)$ could be $\Omega(\sqrt{n})$ even though there may be a single edge e such that $G - e$ is planar. We obtain the first non-trivial approximation algorithm for this problem. Our result is summarized in the following Theorem.

Theorem I.4 (Approximating the minimum planar edge/vertex deletion)**.** *There exists a polynomial-time algorithm which given a graph G of maximum degree Δ, and an integer $k > 0$, either correctly decides that $\mathsf{edge\text{-}planarization}(G) > k$, or outputs a set $Y \subseteq E(G)$, with $|Y| = O(\Delta^5 k^{15} \log^{19/2} n)$, such that $G \setminus Y$ is planar. Similarly, there is a polynomial-time algorithm that either correctly decides that $\mathsf{vertex\text{-}planarization}(G) > k$ or outputs a set $X \subset V$ with $|X| = O(\Delta^4 k^{15} \log^{19/2} n)$ such that $G \setminus X$ is planar.*

Remark I.2. *Our approach via genus gives algorithms with ratios $O(\Delta^{O(1)}\mathsf{OPT}^{O(1)})$ for crossing number and planar edge/vertex deletion. It is useful to note that, unlike for genus, crossing number and planar edge/vertex deletion do not yet have fixed-parameter-tractable algorithms that have a singly-exponential dependence on OPT.*

Further algorithmic applications. Our approximation algorithm for Euler genus has further consequences in the design of algorithms for problems on graphs of small genus. Most algorithms that take advantage of the fact that a graph can be drawn on a surface of small genus require a drawing of the input graph be given as part of the input. If the genus $g = O(\log n)$ then one can use existing exact algorithms that run in $2^{O(g)}\mathrm{poly}(n)$ time to obtain a drawing. Our result implies that we can obtain a drawing even when $g = \Omega(\log n)$, that while not being optimal, nevertheless yields interesting results. A concrete example of this application is the following. Recently, Erickson and Sidiropoulos [7] have obtained a $O(\log g / \log\log g)$-approximation for Asymmetric TSP on graph of Euler genus g, when a drawing of the graph is given as part of the input; this improves the bounds of Oveis-Gharan and Saberi [19] who gave an $O(\sqrt{g}\log g)$-approximation and also required the drawing as an input. Our result implies the following corollary: There exists a polynomial-time $O(\log g / \log\log g)$-approximation for ATSP on bounded-degree graphs of genus g, even when a drawing of the graph is not given as part of the input[1].

The proof of Theorem I.1 is somewhat technical and uses several ingredients. To aid the reader we first give an overview of the algorithmic ideas and highlight the ingredients that are needed. We assume that the reader is familiar with the notion of the treewidth of a graph. Section I-B highlights the high-level idea that allows us to leverage an algorithm for Euler genus for the other problems considered in the paper.

A. Overview of the algorithm for Euler genus

It is convenient to work with a promise version of the problem where we assume that $\mathsf{eg}(G)$ is at most a given number g. This allows us to assume certain properties that G needs to satisfy. Our algorithm may find that G does not

[1]More precisely, there exists a polynomial-time algorithm which given a bounded-degree graph G (the instance of ATSP), and an integer g, either correctly decides that $\mathsf{eg}(G) > g$, or it outputs a $O(\log g / \log\log g)$-approximate TSP tour in G.

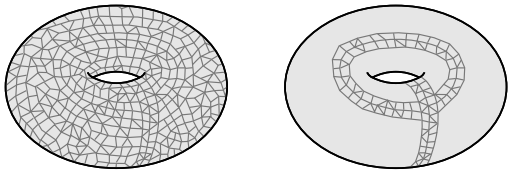


Figure 1. A high-treewidth graph drawn into the torus (left), and the low-treewidth skeleton obtained after removing irrelevant vertices (right).

satisfy such a property in which case it obtains a certificate that $\mathsf{eg}(G) > g$.

An idea from exact algorithms. Our algorithm is inspired by fixed-parameter algorithms that run in polynomial time for any fixed genus [11], [16], [21]. It is instructive to briefly describe how such algorithms work. Let G denote the input graph, and suppose we want to find a drawing into a surface of Euler genus g, if one exists. If G happens to have bounded treewidth, say $f(g)$ for some function f, then one can compute its Euler genus exactly via a dynamic program, in time roughly $2^{O(f(g))}n^{O(1)}$. If on the other hand G has treewidth larger than $f(g)$, by choosing f to be sufficiently large, a theorem of Robertson, and Seymour [20], [22], asserts that G contains a *flat* $((2g+1)\times(2g+1))$-grid minor H. Here, being flat means that the graph H admits a planar drawing, such that all edges leaving H are incident to the outer face. The central vertex v of such a grid can be shown to be *irrelevant*, i.e. such that G admits a drawing into a surface of Euler genus g, if and only if $G - v$ does. Therefore, we can simply remove v from G, and recurse on the remaining graph. We continue removing irrelevant vertices in this fashion, until the treewidth becomes at most $f(g)$. We call the resulting low-treewidth graph a *skeleton* of G (see Figure 1). After drawing the skeleton, we can extend the drawing to all the removed irrelevant vertices.

Challenges when g is not a fixed constant. Our algorithm is based on modifying the above approach, so that it works in the approximate setting when g is part of the input. We now briefly describe the main challenges towards this goal. Let us begin with considering the case of a bounded-degree graph G of small treewidth, say at most $g^{O(1)}$, where g is the Euler genus of G. By repeatedly cutting along balanced separators, we can compute in polynomial time a set of at most $\Delta^{O(1)}g^{O(1)}\log^{O(1)} n$ edges $E^* \subset E(G)$, such that $G \setminus E^*$ is planar. By introducing one handle for every edge in E^*, we get a drawing of G, into a surface of Euler genus (in fact orientable genus) $\Delta^{O(1)}g^{O(1)}\log^{O(1)} n$.

Let us now consider the general case when treewidth of the graph G is larger than g^c for some sufficiently large constant c. Let us assume for now that we can again find an irrelevant vertex in G. It might seem at first that we are done, by proceeding as in the exact case and recursing on the reduced instance. However, this is the critical point where things break down in the approximate setting. Suppose that we remove a set $U \subset V(G)$ of irrelevant vertices, such that the skeleton $G \setminus U$ has treewidth $g^{O(1)}$. We know that the skeleton can be embedded with genus g iff G can. However, we only have an approximate algorithm for handling a low-treewidth graph. Using such an algorithm, we can compute a drawing φ of $G \setminus U$ into a surface of Euler genus $\Delta^{O(1)}g^{O(1)}\log^{O(1)} n$. Unfortunately, now we are stuck! Since the drawing φ is not into a surface of Euler genus g, there might be no way of extending φ to U.

Ensuring extendability. We overcome the above issue by carefully computing irrelevant parts, that have some extra structure. This structure guarantees that the resulting approximate drawing of the skeleton can be extended to the whole G, by introducing only a small number of additional handles. To that end, we define a structure that we call a *patch*. A patch is simply a subgraph $X \subset G$, together with a cycle C, which we can think of as its "boundary". We also think of $X \setminus C$ as the "interior" of the patch. Our goal is to compute patches $X_1, \ldots, X_k$, satisfying the following two conditions:

(C1) After removing the interiors of all patches, the resulting skeleton has treewidth at most $g^{O(1)}$.

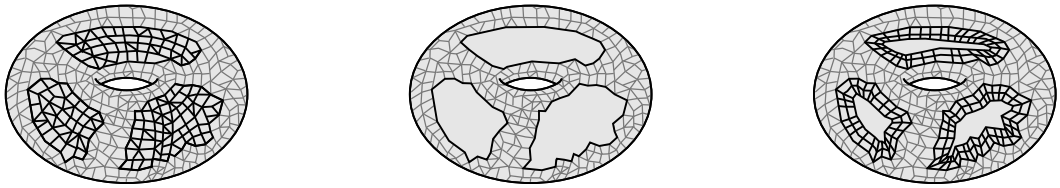
(C2) There exists a drawing φ_{OPT} of G into a surface $\mathcal{S}$ of genus $\mathsf{eg}(G)$, such that every patch X_i is drawn inside a disk $\mathcal{D}_i$, with its boundary being mapped to the boundary of $\mathcal{D}_i$. Moreover, the disks $\mathcal{D}_i$ have pairwise disjoint interiors, and there is no part of G drawn inside each $\mathcal{D}_i$, other than X_i (see Figure 2(a)). We remark that we do not explicitly know φ_{OPT}, but we can nevertheless guarantee its existence.

Let us suppose for now that we can compute such a skeleton, with a corresponding collection of patches $X_1, \ldots, X_k$. Let C_i be the boundary cycle of each X_i. Let G' be the skeleton $G \setminus \bigcup_{i=1}^{k}(X_i \setminus C_i)$ (see Figure 2(b)). Let us now revisit the algorithm for low-treewidth graphs: We repeatedly remove balanced separators of size $g^{O(1)}\log^{O(1)} n$, until all connected components become planar. After removing a set E' of at most $g^{O(1)}\log^{O(1)} n$ edges, we end up with a planar graph $H' = G' \setminus E'$. Fix a planar drawing φ' of H'. We would like to extend φ' to a low-genus drawing of the whole G. To that end, ideally, we would like every cycle C_i to bound a face in φ'. There are two things that can go wrong:

(P1) A cycle might be broken into several different paths.

(P2) A maximal segment P of a cycle C_i in H' might not be "one-sided". That is, there might be no face of φ' containing P as a subpath.

Problem (P1) can be easily addressed: If a cycle gets broken into t pieces, then we can "fix" this by adding at most t extra handles. Since we remove only a small number of edges, and every edge can break at most two cycles, it follows that we only need to add a small number of new handles because of (P1).



(a) A collection of patches (in bold). (b) The skeleton obtained after removing the interiors of all patches. (c) The graph G'' obtained after attaching a width-3 cylinder along every boundary cycle.

Figure 2. Constructing a skeleton.

Overcoming problem (P2) is somewhat more difficult: Intuitively, while computing the drawing of the skeleton G', we modify G' by attaching a cylinder of width 3 on each cycle C_i (see Figure 2(c)). This ensures that in the resulting planar drawing of H', each segment of every cycle is one-sided. In reality, things are more complicated, but this is the high-level idea. After computing a planar drawing φ' as above, where every segment of a cycle is one-sided, we can extend φ' to a low-genus drawing of G.

Computing the skeleton. The missing ingredient is an algorithm to compute the skeleton satisfying the conditions described above. The challenging part is to satisfy condition (C2). One difficulty is that we can only compute patches iteratively. Hence, if we compute the patches naively, it is possible that a patch can "interfere" with previous patches. We avoid this by showing that every new patch, either is interior-disjoint from all previous ones, or it contains some of them in its interior. In the latter case, we can simply merge all internal patches with the current one. This is the technical part of the paper. Our proof uses several tools from the theory of graph minors, and topological graph theory, such as the grid-minor/treewidth duality, Whitney flips, and results on the so-called *planarly-nested sequences* [15].

B. Orientable genus, Crossing number and Planar edge/vertex deletion

Our algorithms for $\mathsf{genus}(G)$, $\mathsf{cr}(G)$, $\mathsf{edge\text{-}planarization}(G)$ and $\mathsf{vertex\text{-}planarization}(G)$ rely on the algorithm for $\mathsf{eg}(G)$. Interestingly having a drawing (even if it is into a non-orientable surface) helps via the following conceptually simple methodology. First we consider the problem of computing $\mathsf{genus}(G)$. Suppose we have a drawing φ of G into a surface $\mathcal{S}$ whose Euler genus is $g^{O(1)}\log^{O(1)} n$ where g is $\mathsf{genus}(G)$. Note that $\mathsf{eg}(G) \leq \mathsf{genus}(G)$ and hence $\mathsf{eg}(G)$ provides a lower bound for $\mathsf{genus}(G)$. We can efficiently check if $\mathcal{S}$ is orientable or non-orientable. If $\mathcal{S}$ is orientable then we are done. Suppose not. Then we compute ρ, the *representativity* (equivalently facewidth) of the drawing φ which captures how "densely" G is embedded in the surface. If ρ is "small" relative to g we can cut a small number of edges along a non-contractible cycle and reduce the genus of the surface. We repeat this process until we obtain a drawing into a surface $\mathcal{S}'$, such that either $\mathcal{S}'$ is orientable, or $\mathcal{S}'$ is nonorientable, and the representativity is "large". If $\mathcal{S}'$ is orientable then we can add handles for all the edges cut along the way and obtain a drawing of the original graph into an orientable surface. The interesting case is when $\mathcal{S}'$ is non-orientable and has high representativity. In this case we can show via results in [2] that G has a large Möbius grid minor that certifies that $\mathsf{genus}(G) > g$.

A similar approach works for $\mathsf{cr}(G)$ and $\mathsf{edge\text{-}planarization}(G)$. It is an easy observation that for each of these problems we have $\mathsf{OPT} \geq \mathsf{genus}(G)$ where OPT is the optimum value for the corresponding problem. Thus we can use our algorithm for $\mathsf{genus}(G)$ to first obtain an embedding into an orientable surface of genus comparable to OPT. We once again use the idea of representativity. Either we can iteratively keep cutting along short non-contractible cycles to reduce the genus by at least one in each step and obtain a planar graph, or we get stuck with an embedding on a non-planar surface with large representativity. In the latter case we find a certificate that OPT is large. In the former case we need to handle the small number of edges removed to obtain the planar graph. There is nothing to do for planar edge deletion since they are part of the output. For crossing number we can add these edges to the planar graph without incurring too many crossings via the results in [4], [6].

Discussion: One could argue that the main reason for the difficulty in approximating graph drawing problems is to get a suitable lower bound on the optimum value. Previous algorithms were based on divide and conquer based approach [1], [3], [13]. However, this approach incurs an additive term that depends on the size of a graph and therefore one only obtains a polynomial-factor approximation. On the other hand the problems are fixed parameter tractable so when OPT is quite small, one can obtain an exact algorithm. Chuzhoy's algorithm for crossing number, and our results, address the intermediate regime when OPT is not too small but is not so large that an additive term that depends on n can be ignored. Chuzhoy's algorithm and analysis are technically very involved but in essense her algorithm finds large rigid substructures in the given

graph (via well-linked sets and grid minors) that have to be necessarily planar in any drawing with crossing number at most $\mathsf{cr}(G)$. Our algorithms for crossing number and planar edge/vertex deletion, are indirect in that they are based on algoritms for genus. Consequently, the bounds we obtain are quantitatively somewhat weaker than those of Chuzhoy for crossing number. However, our algorithm offers a different perspective and approach which may be useful.

Organization: Due to space constraints several proofs and algorithms are omitted. A full version is available on the ArXiv: http://arxiv.org/abs/1304.2416. Section II has some basics and a procedure to simplify the input graph. Section III contains the formal description of our algorithm for Euler genus; it assumes an algorithm for computing the skeleton which is given in Section IV.

II. Preliminaries

For an orientable surface $\mathcal{S}$, let $\mathsf{genus}(\mathcal{S})$ denote its orientable genus. Similarly, for a graph G, let $\mathsf{genus}(G)$ denote its orientable genus. For a graph G, and for $X, Y \subseteq V(G)$, we use $E(X,Y)$ to denote the set of edges with one end point in X and the other in Y. For $X \subseteq V(G)$ we use $N_G(X)$ to denote the neighbors of X, i.e. the set of vertices in $V(G) \setminus X$ that have an edge to some vertex in X.

A graph H is a minor of a graph G if it is obtained from G by a sequence of edge deletions, edge contractions, and deletions of isolated vertices.

Definition II.1 (Minor mapping)**.** *Let G be a graph, and let H be a minor of G. Then there exists a function $\sigma : V(H) \to 2^{V(G)}$, satisfying the following conditions:*

(1) For every $v \in V(H)$, $\sigma(v)$ induces a connected subgraph in G.
(2) For any $u \neq v \in V(H)$, we have $\sigma(u) \cap \sigma(v) = \emptyset$.
(3) For any $\{u, v\} \in E(H)$, there exist $u' \in \sigma(u)$, and $v' \in \sigma(v)$, such that $\{u', v'\} \in E(G)$.

We refer to σ as a minor mapping *(for H). For a set $U \subset V(H)$, we will use the notation $\sigma(U) = \bigcup_{v \in U} \{\sigma(v)\}$.*

Definition II.2 (Grids and cylinders)**.** *Let $r \geq 1$, $k \geq 3$. We define the* $(r \times k)$-cylinder *to be the Cartesian product of the r-path P, with the k-cycle C. We fix an endpoint v of P, and let u be the other endpoint. We refer to the copy of the k-cycle $\{v\} \times C$, as the* top*, and to $\{u\} \times C$, as the* bottom *(of the cylinder).*

Similarly, for $s \geq 1$, $t \geq 1$, the $(s \times t)$-grid *is the Cartesian product of the s-path P, with the t-path Q. We fix an endpoint v of P, and let u be the other endpoint. We refer to the copy of the t-path $\{v\} \times Q$, as the* top*, and to $\{u\} \times Q$, as the* bottom *(of the grid).*

We will make use of the following result of Feige et al. for computing balanced vertex-separators.

Theorem II.3 (Feige et al. [8])**.** *There exists a polynomial time $O(\sqrt{\log n})$-pseudo approximation for balanced vertex separators. Moreover, given a graph G of treewidth t, we can compute in polynomial time a tree decomposition of G of width $O(t\sqrt{\log t})$.*

A. Graph normalization

Before we begin with the description of our algorithm, we give a procedure for simplifying the input graph. Throughout the proof of the main result, we will need to compute and maintain structures that satisfy certain properties in *any* optimal drawing. In order to achieve this, we need to enforce a certain type of "local rigidity" of drawings. To that end, it suffices to ensure that there are no planar components that can "flip" along a small vertex separator. The following is a formal definition of precisely this situation.

Definition II.4 (Freedom)**.** *Let G be a graph, and let $H \subseteq G$ be a vertex-induced subgraph of G. We say that H is* free *(in G) if it satisfies the following conditions:*

(1) There exist at most two vertices in $V(H)$, called portals*, with neighbors in $V(G) \setminus V(H)$.*
(2) If H has two portals t, t', then H is not a path between t and t'.
(3) There exists a planar drawing of H such that all portals lie in the boundary of the outer face.

If there exists only one portal, then we say that H is a petal*, and if there exist two portals, then we say that it is a* clump*.*

Definition II.5 (Normalized graph)**.** *We say that a graph G is* normalized *if there exists no free subgraph $H \subseteq G$.*

The following lemma allows us to restrict our attention to normalized graphs. A similar statement is proven in [15].

Lemma II.6. *Given a graph G of maximum degree Δ, we can compute in polynomial time a graph G' of maximum degree at most Δ, satisfying the following conditions:*

(1) The graph G' is normalized.
(2) Given a drawing of G' into a surface $\mathcal{S}$, we can compute in polynomial time a drawing of G into $\mathcal{S}$.

III. The algorithm

We begin by formally defining the notion of a patch, which we alluded to in Section I-A.

Definition III.1 (Patch)**.** *Let G be a graph. Let $X \subseteq G$ be a subgraph, and let $C \subsetneq X$ be a cycle. Then, we say that the ordered pair (X, C) is a* patch *(of G).*

Note that the above definition of a patch is completely combinatorial, i.e. it is completely independent from drawings of the graph G. We will often refer to a patch, w.r.t. a specific drawing. This is captured in the following definition.

Definition III.2 (φ-Patch)**.** *Let G be a graph, and let (X, C) be a patch of G. Let φ be a drawing of G into a surface*

$\mathcal{S}$. We say that (X, C) is a φ-patch *(of G), if there exists a disk $\mathcal{D} \subset \mathcal{S}$, satisfying the following conditions:*

(1) $\partial\mathcal{D} = \varphi(C)$.

(2) $\varphi(G) \cap \mathcal{D} = \varphi(X)$.

The following definition captures the notion of a pair of "interfering" patches.

Definition III.3 (Overlapping patches)**.** *Let G be a graph, and let (X_1, C_1), (X_2, C_2) be patches of G. We say that (X_1, C_1), and (X_2, C_2) are* overlapping *if either $(X_1 \setminus C_1) \cap X_2 \neq \emptyset$, or $(X_2 \setminus C_2) \cap X_1 \neq \emptyset$. In particular, if (X_1, C_1), and (X_2, C_2) are non-overlapping, then this definition implies $X_1 \cap X_2 = C_1 \cap C_2$.*

Our general goal will be to compute patches that do not interfere precisely in the above sense. We now have all the notation in place, to state the main result for computing a skeleton of the input graph.

Lemma III.4 (Computing a skeleton)**.** *There exists a polynomial-time algorithm which given a graph G of treewidth $t \geq 1$, and maximum degree Δ, and an integer $g > 0$, either correctly decides that $\mathsf{eg}(G) > g$, or outputs a collection of pairwise non-overlapping patches $(X_1, C_1), \ldots, (X_r, C_r)$ of G, so that the following conditions are satisfied:*

(1) If $\mathsf{eg}(G) \leq g$, then there exists a drawing φ of G into a surface of Euler genus g, such that for any $i \in \{1, \ldots, r\}$, (X_i, C_i) is a φ-patch. We emphasize the fact that φ is not explicitly computed by the algorithm.

(2) The graph $G \setminus (\bigcup_{i=1}^{r}(X_i \setminus C_i))$ has treewidth at most $O(\Delta g^{11} \log^8 n)$.

Lemma III.4 is the main technical part of the paper. In the interest of clarity, we postpone its proof to later sections, and we instead show now how it can be used to obtain our approximation algorithm for Euler genus.

Before we describe the actual algorithm, we need to define a local "framing" operation, which we use to modify the skeleton. Intuitively, this is needed to ensure that when computing a drawing for the skeleton, the boundaries of the patches are drawn in a "nearly one-sided" fashion. This "near one-sidedness" in turn will allow us to extend the drawing of the skeleton, to a drawing of the whole graph. Note that framing is a combinatorial operation and does not rely on a drawing.

Definition III.5 (Graph framing)**.** *Let G be a graph, and let $\mathcal{C} = C_1, \ldots, C_k \subseteq G$ be a collection of cycles. Let G' be the graph obtained from G by taking for every $i \in \{1, \ldots, k\}$, a copy K_i of the $(3 \times |V(C_i)|)$-cylinder, and identifying the top of K_i with C_i. We refer to G' as the* $\mathcal{C}$-framing of G *(see Figure 3(a)).*

More generally, we define the framing operation for subgraphs. Let $H \subseteq G$ be a subgraph of G. We define a graph H' as follows. Consider some $C_i \in \mathcal{C}$. If $C_i \subseteq H$, then we take a copy of the $(3 \times |V(C_i)|)$-cylinder, and we identify its top with C_i. If $C_i \not\subseteq H$, then let $P_1, \ldots, P_a$ be the set of maximal subpaths of C_i that are contained in H. For every such P_j, we take a copy of the $(3 \times |V(P_j)|)$-grid, and we identify its top with P_j. We repeat this process for all $C_i \in \mathcal{C}$, and we define H' to be the resulting graph. We refer to H' as the $\mathcal{C}$-framing of H *(see Figure 3(b)). The reader can check that the definition of the $\mathcal{C}$-framing of H agrees with the one given above, when $H = G$.*

We now state a basic property of framing whose proof follows directly from the definition of a $\mathcal{C}$-framing.

Lemma III.6. *Let G be a graph, and let $\mathcal{C}$ be a collection of cycles in G. Let $\mathcal{H}$ be a collection of pairwise vertex-disjoint subgraphs of G. Let G' be the $\mathcal{C}$-framing of G, and for any $H \in \mathcal{H}$, let H' be the $\mathcal{C}$-framing of H. Then, the graph $\bigcup_{H \in \mathcal{H}} H'$ is (isomorphic to) a subgraph of G'.*

We first argue that framing does not increase the genus of the skeleton.

Lemma III.7 (The genus of a framed skeleton)**.** *Let G be a graph, and let φ be a drawing of G into some surface $\mathcal{S}$. Let $\mathcal{P}$ be a collection of pairwise non-overlapping φ-patches of G. Let $G' = G \setminus \left(\bigcup_{(X,C) \in \mathcal{P}} X \setminus C\right)$. Let G'' be the $\{C\}_{(X,C) \in \mathcal{P}}$-framing of G'. Then, $\mathsf{eg}(G'') \leq \mathsf{eg}(G)$.*

Lemma III.8 (Planarizing the skeleton)**.** *There exists a polynomial-time algorithm which given a graph G of maximum degree Δ, an integer $g > 0$, and a collection of pairwise non-overlapping patches $(X_1, C_1), \ldots, (X_r, C_r)$ of G satisfying the assertion of Lemma III.4, it either correctly decides that $\mathsf{eg}(G) > g$, or it outputs a set $S \subseteq V(G)$, satisfying the following conditions:*

(1) $|S| = O(\Delta g^{12} \log^{19/2} n)$.

(2) Let $\mathcal{C} = \{C_1, \ldots, C_r\}$, and let $G' = G \setminus (\bigcup_{i=1}^{r}(X_i \setminus C_i))$. For every connected component H of $G' \setminus S$, we have that the $\mathcal{C}$-framing of H is planar.

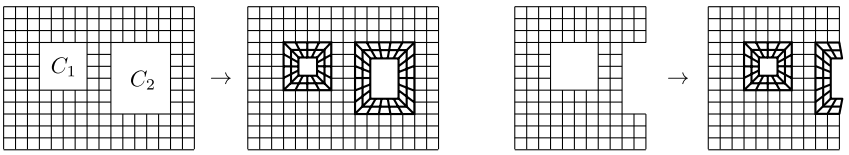
We can now prove the main result of this paper.

Proof of Theorem I.1: By Lemma III.4, in polynomial time, we can either correctly decide that $\mathsf{eg}(G) > g$, or we can compute a (possibly empty) collection $\mathcal{P}$ of pairwise non-overlapping patches of G, satisfying the following conditions:

- If $\mathsf{eg}(G) \leq g$, then there exists a drawing φ of G into a surface $\mathcal{S}$ of Euler genus g, such that any $(X, C) \in \mathcal{P}$ is a φ-patch.
- The graph $G' = G \setminus \left(\bigcup_{(X,C) \in \mathcal{P}} X \setminus C\right)$ has treewidth $t' = O(\Delta g^{11} \log^8 n)$.

Let $\mathcal{C} = \{C : (X, C) \in \mathcal{P}\}$. For any subgraph $H \subseteq G'$, let H^{framed} denote the $\mathcal{C}$-framing of H.

Let $S \subseteq V(G)$ be the set computed by Lemma III.8. Let $G'' = G' \setminus S$, and let $\mathcal{H}$ be the set of connected components of G''. Since for every $H \in \mathcal{H}$ the graph H^{framed} is planar, it



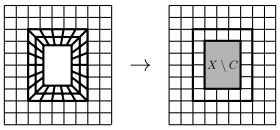
(a) $\{C_1, C_2\}$-Framing of a graph. (b) $\{C_1, C_2\}$-Framing of a subgraph.

Figure 3. Examples of graph framing.

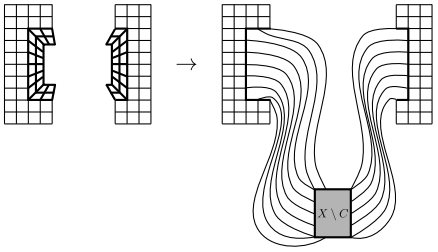
follows that $(G'')^{\mathsf{framed}} = \bigcup_{H \in \mathcal{H}} H^{\mathsf{framed}}$ is also planar. Pick a planar drawing ψ of $(G'')^{\mathsf{framed}}$ (which can be computed, e.g. by the algorithm of Hopcroft, and Tarjan [10]).

We now proceed to obtain a drawing of G, by modifying the drawing ψ of $(G'')^{\mathsf{framed}}$. We iterate over all patches $(X, C) \in \mathcal{P}$. Consider some $(X, C) \in \mathcal{P}$. Since every $(X, C) \in \mathcal{P}$ is a φ-patch, it follows that the graph X admits a planar drawing $\varphi_{(X,C)}$ into a disk $\mathcal{D}_{(X,C)}$, with $\varphi_{(X,C)}(C) = \partial\mathcal{D}_{(X,C)}$. Let $n_C = |E(C) \setminus E(G'')|$. We consider two cases:

(i) If $n_C = 0$, it follows that $C \subseteq G''$. Therefore, $(G'')^{\mathsf{framed}}$ contains a $(3 \times |V(C)|)$-cylinder K. The top of K is identified with C in $(G'')^{\mathsf{framed}}$. Since K is 3-vertex-connected, it admits a unique planar drawing. Let C' be the bottom of K. It follows K bounds a face F in ψ. We can therefore extend the current drawing to $X \setminus C$, by placing the open disk $\mathcal{D}_{(X,C)} \setminus \partial\mathcal{D}_{(X,C)}$ inside the face F, deleting all vertices of K that do not belong to its top, and connecting the vertices in X, with their neighbors in the copy of C in G''. Notice that in this case, we do not increase the genus of the current surface.



(ii) If $n_C > 0$, we proceed as follows. First, we place the disk $\mathcal{D}_{(X,C)}$ in the unbounded face of the current drawing. Let $P_1, \ldots, P_{n_C}$ be the set of maximal segments of C contained in G''. Consider a maximal segment P_j. There exists a $(3 \times |V(P_j)|)$-grid L in $(G'')^{\mathsf{framed}}$, such that the top of L is identified with P_j. We argue that the bottom of L is a segment of a face F in ψ: If $|V(P_j)| \leq 2$, this is immediate, and if $|V(P_j)| \geq 3$, it follows by Whitney's theorem, since L is 3-connected, and therefore has a unique planar drawing. If the orientation of P_j along a clockwise traversal of $\partial\mathcal{D}_{(X,C)}$ agrees with the orientation of P_j along a clockwise traversal of F, then we attach a handle between P_j in X, and the bottom of L. Otherwise, we attach a Möbius band. Next, we delete all the vertices in L that do not belong to its bottom, and we also delete the copy of P_j in X (i.e. the copy that lies on the boundary of $\mathcal{D}_{(X,C)}$. Finally, we draw the edges between $X \setminus C$, and P_j, by routing them along the new handle, or Möbius band. We continue in the same fashion, with all the remaining maximal segments. For every maximal segment, we increase the Euler genus of the underlying surface by at most 3. Therefore, the total increase in the Euler genus is at most $3n_C$.



After considering all patches in $\mathcal{P}$, we arrive at a drawing into some surface. We remove any remaining vertices from $(G'')^{\mathsf{framed}} \setminus G''$. We arrive at a drawing ψ' of the graph $\Gamma = G\left[V(G'') \cup \left(\bigcup_{(X,C)\in\mathcal{P}} X \setminus C\right)\right] = G \setminus S$.

Since the cycles $\{\varphi(C)\}_{C\in\mathcal{C}}$ bound disks with disjoint interiors in the drawing φ, it follows that every edge of G' is contained in at most two cycles in $\mathcal{C}$. Therefore, $\sum_{C\in\mathcal{C}} n_C \leq 2 \cdot |E(G') \setminus E(G'')| \leq \Delta \cdot |V(G') \setminus V(G'')| = \Delta \cdot |S| = O(\Delta^2 g^{12} \log^{19/2} n)$. It follows that the resulting drawing ψ' of Γ is into a surface of Euler genus at most $\sum_{C\in\mathcal{C}} 3n_C = O(\Delta^2 g^{12} \log^{19/2} n)$.

It remains to extend the drawing to S. This can be done by adding at most $|S| \cdot \Delta$ additional handles (one for every edge incident to a vertex in S). The resulting drawing of G has Euler genus $O(\Delta^2 g^{12} \log^{19/2} n) + O(|S| \cdot \Delta) = O(\Delta^2 g^{12} \log^{19/2} n)$, as required. This concludes the proof. ■

IV. Computing a skeleton

The rest of the paper is devoted to the algorithm for computing a skeleton, i.e. the proof of Lemma III.4. At the high level, the algorithm proceeds iteratively as follows: (i) We compute a patch. (ii) We remove its interior. (iii) We repeat until the treewidth of the remaining graph becomes small enough. One issue with implementing the above approach is that we do not have access to an explicit optimal drawing of the input graph. Therefore, we cannot argue that a computed patch is a φ-patch for a *specific* optimal drawing. To that end, we will need a stronger notion of a patch. More precisely,

we need to compute patches that are φ-patches, in *any* optimal drawing φ. Moreover, because the above procedure computes patches in an ever decreasing graph, we need to make sure that the genus never decreases. This property will ensure that at the end of the procedure, all computed patches are φ-patches for *the same* optimal drawing φ. The following definition states precisely the properties that we need.

Definition IV.1 (Universal patch)**.** *Let G be a graph of Euler genus g. Let $X \subseteq G$ be a subgraph of G, and let $C \subsetneq X$ be a cycle in X. We say that (X, C) is a* universal patch *(of G) if it satisfies the following conditions:*

(1) For any *drawing ψ of G into a surface of Euler genus g, we have that (X, C) is a ψ-patch.*
(2) Let $G' = G \setminus (X \setminus C)$. Then, $\mathsf{eg}(G') = \mathsf{eg}(G) = g$.

The following lemma shows that we can compute a universal patch in polynomial time, in a normalized graph of sufficiently large treewidth. A crucial property of the algorithm is that after removing the interior of the computed patch, the resulting graph remains normalized. This fact allows us to inductively maintain a normalized graph, while computing the skeleton. Intuitively, dealing with a normalized graph is essential for avoiding "locally-pathological" planar drawings. Roughly speaking, a non-normalized graph can have an optimal drawing that locally looks rather complicated. This makes it very difficult to control how different overlapping patches interact with each other.

Lemma IV.2 (Computing a universal patch)**.** *There exists a universal constant $\alpha > 0$, such that the following holds. Let G be a normalized graph of Euler genus $g \geq 1$, treewidth $t \geq 1$, and maximum degree Δ. Suppose that $t \geq \alpha \Delta g^{11} \log^{15/2} n$. Then, we can compute in polynomial time a universal patch (X, C) in G, such that $G \setminus (X \setminus C)$ is normalized.*

The proof of Lemma IV.2 is rather long, and requires several other results, including our algorithm for computing a flat grid minor, and properties of planarly nested sequences due to Mohar [15]. For that reason, we defer it to the full version, and show here how to use it to construct a skeleton (i.e. to prove Lemma III.4).

Before we proceed with the proof of Lemma III.4, we first derive a property that will be used in showing the correctness of the algorithm. While computing a sequence of patches, it is possible that a patch overlaps a previously computed one. In this case, we can show that we can essentially "merge" the two patches. The following Lemma gives the necessary properties for doing exactly that.

Lemma IV.3 (Merging overlapping patches)**.** *Let G be a graph, let φ_1 be a drawing of G into a surface $\mathcal{S}$, and let (X_1, C_1) be a φ_1-patch of G. Let $G' = G \setminus (X_1 \setminus C_1)$. Let φ_2 be the drawing of G' into $\mathcal{S}$ obtained by restricting φ_1 to G'. Let (X_2, C_2) be a φ_2-patch of G'. Suppose further that $(V(X_2) \setminus V(C_2)) \cap V(C_1) \neq \emptyset$. Then, $(X_1 \cup X_2, C_2)$ is a φ_1-patch of G.*

Proof: Since (X_2, C_2) is a φ_2-patch, it follows that there exists $\mathcal{D}_2 \subset \mathcal{S}$, such that $\partial \mathcal{D}_2 = \varphi_2(C_2)$. It remains to show that $\varphi_1(G) \cap \mathcal{D}_2 = \varphi_1(X_1 \cup X_2)$. Since (X_1, C_1) is a φ_1-patch, it follows that there exists a disk $\mathcal{D}_1 \subset \mathcal{S}$, with $\partial \mathcal{D}_1 = \varphi_1(C_1)$. We claim that $\mathcal{D}_1 \subseteq \mathcal{D}_2$. Suppose to the contrary that there exists a point $p \in \mathcal{D}_1 \setminus \mathcal{D}_2$. Pick an arbitrary vertex $v \in (V(X_2) \setminus V(C_2)) \cap V(C_1)$, and let $q = \varphi_1(v) = \varphi_2(v)$. We have $q \in \mathcal{D}_2 \cap \mathcal{D}_1$. Moreover, since $v \in V(X_2) \setminus V(C_2)$, it follows that $q \notin \partial \mathcal{D}_2$. Therefore, there exists a segment of $\partial \mathcal{D}_2$ that lies inside $\mathcal{D}_1$. This implies that $\varphi_2(X_2)$ intersects the interior of $\mathcal{D}_1$. Thus, there exists $e \notin E(X_1)$, with $\varphi_2(e) = \varphi_1(e) \subset \mathcal{D}_1$, which contradicts the fact that (X_1, C_1) is a φ_1-patch. ■

We are now ready to prove Lemma III.4, which is the main result of this section. Before proceeding, we remark that the assertion of Lemma III.4 can in fact be slightly strengthened. More precisely, one can show that in the computed collection, all patches are universal. We omit the details since they are not relevant to our algorithmic application.

Proof of Lemma III.4: Fix a drawing φ of G into a surface of Euler genus g. We remark that we use g in the following argument, even though we do not know how to explicitly compute it in polynomial time.

We inductively compute a sequence $\mathcal{P}^{(0)}, \ldots, \mathcal{P}^{(s)}$, with $\mathcal{P}^{(0)} = \emptyset$, where for each $i \in \{1, \ldots, s\}$, $\mathcal{P}^{(i)}$ is a collection of pairwise non-overlapping φ-patches of G. The desired collection will be $\mathcal{P}^{(s)}$.

For any $\ell \in \{1, \ldots, s\}$, we define the graph $G^{(\ell)} = G \setminus \bigcup_{(X,C) \in \mathcal{P}^{(\ell)}} (X \setminus C)$. We maintain the inductive invariant that for any $\ell \in \{1, \ldots, s\}$,

$$G^{(\ell)} \text{ is normalized, and } \mathsf{eg}(G^{(\ell)}) = \mathsf{eg}(G) = g. \quad (1)$$

Given $\mathcal{P}^{(\ell)}$, for some $\ell \geq 0$, we proceed as follows. Let $\alpha > 0$ be the constant in the statement of Lemma IV.2. By Theorem II.3, there exists a polynomial time algorithm that given a graph of treewidth k, outputs a tree decomposition of width at most $\alpha' k \sqrt{\log k}$, for some universal constant α'. We run the algorithm of Theorem II.3 on $G^{(\ell)}$. If the algorithm returns a tree decomposition of width at most $\alpha \cdot \alpha' \cdot \Delta g^{11} \log^8 n$, then we have a certificate that the treewidth of $G^{(\ell)}$ is at most $O(\Delta g^{11} \log^8 n)$, and we set $r = \ell$. Otherwise, we know that the treewidth of $G^{(\ell)}$ is at least $\alpha \Delta g^{11} \log^{15/2} n$, and we proceed to compute $\mathcal{P}^{(\ell+1)}$.

By Lemma IV.2, and since $G^{(l)}$ is normalized, we can compute in polynomial time a universal patch in (X^*, C^*) of $G^{(\ell)}$. Let $\mathcal{Q}^{(\ell)} = \{(X, C) \in \mathcal{P}^{(\ell)} : (X, C)$ and (X^*, C^*) are overlapping$\}$. Fix an ordering of the patches in $\mathcal{Q}^{(\ell)} = \{(Y_i, F_i)\}_{i=1}^{k_\ell}$, where $k_\ell = |\mathcal{Q}^{(\ell)}|$. Let $Y = \bigcup_{i=1}^{k_\ell} Y_i$. We argue that $(X^* \cup Y, C)$ is a φ-patch of G. Let $\Gamma^{(0)} = G^{(\ell)}$, and for any $j \in \{1, \ldots, k_\ell\}$,

let $\Gamma^{(j)} = \Gamma^{(j-1)} \cup Y_j$. Let also $\varphi^{(j)}$ be the drawing of $\Gamma^{(j)}$ induced by restricting φ to $\Gamma^{(j)}$. Since (X^*, C^*) is a universal patch of $\Gamma^{(0)} = G^{(\ell)}$, and $\varphi^{(0)}$ is a drawing into a surface of Euler genus g, it follows that (X^*, C^*) is also a $\varphi^{(0)}$-patch. Note that for any $j \in \{1, \ldots, k_\ell\}$, since (Y_j, F_j) is a φ-patch, and $\varphi^{(j)}$ is a restriction of φ, it follows that (Y_j, F_j) is also a $\varphi^{(j)}$-patch. By Lemma IV.3 we obtain that $(X^* \cup Y_1, C^*)$ is a $\varphi^{(1)}$-patch of $\Gamma^{(1)}$. By inductively applying Lemma IV.3 on the pair of patches (Y_j, F_j), and $\left(X^* \cup \left(\bigcup_{i=1}^{j-1} Y_i\right), C^*\right)$, we conclude that $(X^* \cup Y, C)$ is a $\varphi^{(k_\ell)}$-patch of $\Gamma^{(k_\ell)}$. Since the patch $(X^* \cup Y, C)$ is non-overlapping with any of the patches in $\mathcal{P}^{(\ell)} \setminus \mathcal{Q}^{(\ell)}$, it follows that $(X^* \cup Y, C)$ is a φ-patch. We set $\mathcal{P}^{(\ell+1)} = \left(\mathcal{P}^{(\ell)} \setminus \mathcal{Q}^{(\ell)}\right) \cup \{(X^* \cup Y, C^*)\}$. It is immediate that $\mathcal{P}^{(\ell+1)}$ is a collection of non-overlapping φ-patches.

We next show that the inductive invariant (1) is maintained. Observe that $G^{(\ell+1)} = G^{(\ell)} \setminus (X^* \setminus C^*)$. Since (X^*, C^*) is a universal patch of $G^{(l)}$, it follows that $\mathsf{eg}(G^{(l+1)}) = \mathsf{eg}(G^{(l)}) = g$. Moreover, by Lemma IV.2 we have that $G^{(\ell+1)}$ is normalized. This shows that the inductive invariant (1) is maintained.

It remains to argue that the above process terminates after polynomially many steps. Note that since (X^*, C^*) is a patch, we have $X^* \subsetneq C^*$. Therefore, $G^{(\ell+1)} \subsetneq G^{(\ell)}$. It follows that the algorithm terminates in polynomial time. This concludes the proof. ∎

Acknowledgments: We thank Jeff Erickson and Kenichi Kawarabayashi for useful discussions. We thank two anonymous reviewers of our FOCS 2013 submission for pointing out technical issues in a few proofs.

REFERENCES

[1] S. N. Bhatt and F. T. Leighton. A framework for solving vlsi graph layout problems. *J. Comput. Syst. Sci.*, 28(2):300–343, 1984.

[2] R. Brunet, B. Mohar, and R. B. Richter. Separating and non-separating disjoint homotopic cycles in graph embeddings. *J. Comb. Theory, Ser. B*, 66(2):201–231, 1996.

[3] J. Chen, S. P. Kanchi, and A. Kanevsky. A note on approximating graph genus. *Inf. Process. Lett.*, 61(6):317–322, 1997.

[4] M. Chimani and P. Hlinený. A tighter insertion-based approximation of the crossing number. In *ICALP (1)*, pages 122–134, 2011.

[5] J. Chuzhoy. An algorithm for the graph crossing number problem. In *STOC*, pages 303–312, 2011.

[6] J. Chuzhoy, Y. Makarychev, and A. Sidiropoulos. On graph crossing number and edge planarization. In *SODA*, pages 1050–1069, 2011.

[7] J. Erickson and A. Sidiropoulos. A near-optimal approximation algorithm for asymmetric TSP on embedded graphs. Manuscript, 2013.

[8] U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM J. Comput.*, 38(2):629–657, 2008.

[9] J. R. Fiedler, J. P. Huneke, R. B. Richter, and N. Robertson. Computing the orientable genus of projective graphs. *Journal of Graph Theory*, 20(3):297–308, 1995.

[10] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, Oct. 1974.

[11] K. Kawarabayashi, B. Mohar, and B. A. Reed. A simpler linear time algorithm for embedding graphs into an arbitrary surface and the genus of graphs of bounded tree-width. In *FOCS*, pages 771–780, 2008.

[12] K. Kawarabayashi and B. A. Reed. Computing crossing number in linear time. In *STOC*, pages 382–390, 2007.

[13] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[14] Y. Makarychev, A. Nayyeri, and A. Sidiropoulos. A pseudo-approximation algorithm for the genus of hamiltonian graphs. Manuscript, 2012.

[15] B. Mohar. Combinatorial local planarity and the width of graph embeddings. *Canad. J. Math.*, 44(6):1272–1288, 1992.

[16] B. Mohar. A linear time algorithm for embedding graphs in an arbitrary surface. *SIAM J. Discrete Math.*, 12(1):6–26, 1999.

[17] B. Mohar. Face covers and the genus problem for apex graphs. *J. Comb. Theory, Series B*, 82(1):102–117, 2001.

[18] B. Mohar and C. Thomassen. *Graphs on surfaces*. Johns Hopkins University Press, 2001.

[19] S. Oveis Gharan and A. Saberi. The asymmetric traveling salesman problem on graphs with bounded genus. In *SODA*, pages 967–975, 2011.

[20] N. Robertson and P. D. Seymour. Graph minors. V. Excluding a planar graph. *J. Comb. Theory, Ser. B*, 41(1):92–114, 1986.

[21] N. Robertson and P. D. Seymour. Graph minors. VIII. A Kuratowski theorem for general surfaces. *J. Comb. Theory, Ser. B*, 48(2):255–288, 1990.

[22] N. Robertson and P. D. Seymour. Graph minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B*, 89(1):43–76, 2003.

[23] C. Thomassen. The graph genus problem is np-complete. *J. Algorithms*, 10(4):568–576, 1989.

[24] C. Thomassen. Triangulating a surface with a prescribed graph. *J. Comb. Theory, Ser. B*, 57(2):196–206, 1993.