Approximating Connectivity Domination in Weighted Bounded-Genus Graphs

Vincent Cohen-Addad[*]
Département d'informatique,
École normale supérieure
France

Éric Colin de Verdière[†]
CNRS,
Département d'informatique,
École normale supérieure
France

Philip N. Klein[‡]
Brown University
United States

Claire Mathieu[§]
CNRS,
Département d'informatique,
École normale supérieure
France

David Meierfrankenfeld[¶]
Brown University
United States

ABSTRACT

We present a framework for addressing several problems on weighted planar graphs and graphs of bounded genus. With that framework, we derive polynomial-time approximation schemes for the following problems in planar graphs or graphs of bounded genus: *edge-weighted tree cover* and *tour cover*; *vertex-weighted connected dominating set*, *max-weight-leaf spanning tree*, and *connected vertex cover*. In addition, we obtain a polynomial-time approximation scheme for *feedback vertex set* in planar graphs. These are the first polynomial-time approximation schemes for all those problems in *weighted* embedded graphs. (For unweighted versions of some of these problems, polynomial-time approximation schemes were previously using bidimensionality.)

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical algorithms and problems—*Computations on discrete structures; Geometrical problems and computations*; G.2.2 [**Discrete Mathematics**]: Graph theory—*Graph algorithms; Network problems.*

General Terms

Algorithms, Theory

Keywords

Approximation algorithm, polynomial-time approximation scheme, graph, bounded genus, planar graph, connected dominating set, feedback vertex set

1. INTRODUCTION

An *approximation scheme* for an optimization problem is an algorithm that, for any given $\epsilon > 0$, outputs a solution whose value is within a $1+\epsilon$ factor of optimal. The asymptotic running time is stated assuming ϵ is a constant. It is called a *polynomial-time approximation scheme* (PTAS) if the running time is polynomial. It is called a *quasi-PTAS* if the running time is exponential in a polylogarithm.

For many fundamental NP-hard optimization problems in graphs, there are no polynomial-time approximation schemes unless P=NP. However, it has turned out that polynomial-time approximation schemes often do exist when the graph is required to be *planar* or, more generally, bounded-genus.

1.1 Previous Frameworks

There is a rather long history of research on approximation schemes for planar graphs, going back to 1977. Three approaches jointly yield most polynomial-time approximation schemes known for planar graphs: Baker's method [4], approximation via bidimensionality (Demaine and Hajiaghayi) [17], and a framework of Klein [36, 37].

Each of these methods has its limitations. Baker's method only addresses problems that are local in character, e.g., min-weight vertex cover and dominating set. Bidimensionality is only defined for problems without weights, and this approach only yields approximation schemes for such problems, though it does address very nonlocal problems such as feedback vertex set and connected dominating set. The

[*]Research funded by the French ANR Blanc project ANR-12-BS02-005 (RDAM)

[†]Research funded by the French ANR Blanc project ANR-12-BS02-005 (RDAM)

[‡]Research funded by NSF Grant CCF-10-12254 with additional support from the Radcliffe Institute of Advanced Study, Harvard University

[§]Research funded by the French ANR Blanc project ANR-12-BS02-005 (RDAM)

[¶]Research funded by NSF Grant CCF-10-12254

STOC'16, June 19–21, 2016, Cambridge, MA, USA
ACM. 978-1-4503-4132-5/16/06...$15.00
http://dx.doi.org/10.1145/2897518.2897635

framework of [36, 37] has been used for a variety of weighted, nonlocal problems; it has yielded, for example, a linear-time approximation scheme for traveling salesman, near-linear-time approximation schemes for Steiner tree and generalizations, and polynomial-time approximation schemes for cut problems such as multiway cut and graph bisection. However, for each problem addressed, it requires a kind of sparsification that approximately preserves optimality; for some problems, obtaining such a sparsification seems difficult.

1.2 Our Framework: Ubiquity

In this paper, we present a new approach that yields polynomial-time approximation schemes (PTASs) for some weighted, nonlocal problems for which no PTAS was previously known: feedback vertex set, connected dominating set, connected vertex cover, tree cover, tour cover, spanning tree maximizing the weight of leaves, and others. Our approach works for planar graphs, or more generally for graphs of *bounded genus* (drawable without crossings on a fixed surface such as a torus or more complicated topological surfaces).

For these problems, a solution consists either of a subgraph or of a set of edges and/or vertices. In the latter case, the solution can be equivalently expressed by the subgraph induced by that set. The value of the solution is always the weight of the subgraph.

For the unweighted versions of these problems, bidimensionality gives polynomial-time approximation schemes. Bidimensionality applies to a problem only if a solution is necessarily *dense* in the graph; in particular if for grid graphs a solution necessarily uses a constant fraction of the edges. Recall that bidimensionality applies only when every element (vertex/edge) has the same weight. Thus density is measured in terms of the *value* of the optimum.

Similarly, the first step of an algorithm employing the framework of [36, 37] is to thin the graph (using deletions or contractions) so that the total weight of the graph is a small factor times the value of the optimum. Thus again the key is ensuring that the optimum value is a large fraction of the weight of the graph. It remains unknown for many optimization problems whether such a thinning step can be carried out in polynomial time.

We therefore want to identify a property of problems for which no thinning step is needed, like bidimensionality, but we want our property to work for problems with weights, unlike bidimensionality. The key idea is to define the property in terms of graph structure rather than solely in terms of the optimum value. Intuitively, rather than require that a solution include a constant fraction of the edges of a graph, we require that the edges *not* belonging to a solution form a subgraph with a simple structure. As is often the case in recent research in graph algorithms, "simple structure" is formalized as small *treewidth*[1] or, equivalently, small branchwidth.[2] Frequently these algorithms use the fact that many NP-hard graph problems can be solved quickly on graphs of small treewidth. However, in our framework it wouldn't help to solve the problem in the subgraph of edges not in a solution. We make a different use of the small treewidth of the graph of edges not in the solution; it enables us to prove the existence of a certain kind of separator structure for the entire graph.

[1]This is a a well-known concept in graph theory. See, e.g., [19] for the definition.
[2]Defined in Section 4.

Definition 1.1. *Let t be an integer. We say a graph problem is* t-ubiquitous *(or simply* ubiquitous*) if, for every input graph G and every feasible solution S, S is connected and G/S has treewidth at most t.*

Planar graphs, bounded-genus graphs, and, more generally, members of a minor-closed graph family excluding some apex graph all have the diameter-treewidth property [22]: the treewidth of such a graph is upper-bounded by some function of its unweighted diameter. When referring to unweighted distance in a graph, we use the term *hops* to distinguish this from measuring distance according to edge- or vertex-weights.

Observation 1.2. *Suppose a graph problem restricts the input graphs to have bounded genus. Suppose also that for some integer t, for every input graph G and for every feasible solution S, S is connected and every vertex of G is within t hops of S. Then the graph problem is $O(t)$-ubiquitous.*

By the observation, for a problem on bounded-genus graphs to be considered ubiquitous, it is enough that every solution be "everywhere" in the sense that every vertex is close to the solution in terms of number of hops.[3] Let g be a positive integer, considered a constant for the purpose of stating running times in our main theorem, which is as follows:

Theorem 1.3. *Let P be a minimization problem on edge- or vertex-weighted graphs with genus at most g, such that contracting an edge of an input graph can only reduce the optimal value, and*

1. $O(1)$-approximation of ubiquitous problem: *For some constant t, P is t-ubiquitous, and there is a polynomial-time algorithm that, given an input G for P, outputs an α-approximation[4] for P, for some constant α.*
2. Dynamic program: *There is an $2^{O(b)}poly(n)$ algorithm to find an optimal or $1+\epsilon$-approximate solution to instances of P with branchwidth at most b.*
3. Lifting: *There is a constant β and a polynomial-time algorithm that, given a graph G and a subgraph K, and given a solution S to problem P for input G/K, outputs a solution for G of weight at most $w(S) + \beta \cdot w(K)$.*

Then there is a polynomial-time approximation scheme for P.

This theorem is a consequence of a slightly more general version in which Condition 1 is replaced with:
Condition 1′. There are constants $\alpha \geq 1$ and t and a polynomial-time algorithm that, given an input G for P, outputs a connected subgraph B such that B has weight at most α times the optimal value for G, and G/B has treewidth at most t.

The key ingredient to prove Theorem 1.3 is the following result.

Theorem 1.4 (Branchwidth Reduction Theorem)**.** *Let $\varepsilon > 0$ and b, g be two integers. There is a polynomial-time algorithm for the following:*

Input: *Graph G_0 of genus at most g with edge weights and/or vertex weights, connected subgraph H_0 of G_0 such that G_0/H_0 has branchwidth at most $b - 1$.*

[3]In fact, it is sufficient even if we measure number of hops in the *face-vertex incidence graph* (a.k.a. the *radial graph*).
[4]This is a variation to what Demaine and Hajiaghayi call a "backbone" in the bidimensional approach [17].

Output: *Subgraph K of H_0 such that the total weight of the edges and vertices of K is at most ε times the total weight of the edges and vertices of H_0, and G_0/K has branchwidth $O(\log n)$, where n is the number of vertices of G_0.*

The branchwidth depends linearly on b and ϵ^{-1}, and polynomially on g.

Proof of Theorem 1.3. Here is the algorithm.

Algorithm 1 Meta-Algorithm for Ubiquitous Problems

```
1: Input: A graph $G = (V, E)$ of genus $g$ with nonnegative vertex and edge weights.
2: $H \leftarrow$ an $\alpha$-approximation for the problem in $G$.
3: $K \leftarrow$ BranchwidthReduction$(G, H)$. By Theorem 1.4, $S_1$ has total weight at most $\varepsilon \cdot \alpha \cdot OPT$ and $G/K$ has branchwidth $O(t/\varepsilon) \cdot \log n$.
4: $Y \leftarrow$ an optimal solution for the problem in $G/K$.
5: Output: $S_3 \leftarrow$ a solution for $G$ based on $Y$ and $K$.
```

For the analysis, combining the three assumptions and Theorem 1.4, the running time of the algorithm is polynomial. The solution obtained has cost $w(S_2) + \beta \cdot w(S_1)$, combining the three assumptions and Theorem 1.4, the total cost is $(1 + \alpha \cdot \beta \cdot \varepsilon)$OPT. □

1.3 Our Concrete Results

We can apply our framework to several concrete problems, where we obtain the first polynomial-time approximation schemes for bounded-genus graphs; a summary of the results is given in Table 1. (Previously such approximation schemes were known only for the unweighted versions of the problems.) In Section 5, we formally define these problems, summarize previous work, and show how to obtain our concrete results.

We draw the reader's attention to two problems in particular: *(undirected) feedback vertex set* and *connected dominating set.* As Demaine and Hajiagayi state [17], these are "important problems that have been studied extensively in the literature." Feedback vertex set was one of the 21 original problems shown NP-complete by Karp [35]. Connected dominating set arises, e.g., in virtual backbone-based routing in ad hoc wireless networks (e.g. [54]). Feedback vertex set does not fit into the framework of Theorem 1.3 but we show how to reduce the problem in planar graphs to connected dominating set.

For vertex-weighted connected dominating set, in order to satisfy Property 1, we use a constant-factor approximation for bounded-genus graphs. None was previously known (Demaine and Hajiaghayi give one for the unweighted version of the problem) so we supply one.

1.4 Algorithmic Ingredients

In Section 4, we prove the Branchwidth Reduction Theorem (Theorem 1.4) by recursively invoking a separator theorem. Planar separators were used in the first approximation scheme for planar graphs, due to Lipton and Tarjan [41], and again used by Grigni, Koutsoupias and Papadimitriou [30] in an approximation scheme for the unweighted traveling salesman problem in planar graphs, and one by Arora, Grigni, Karger, Klein, and Woloszyn [2] for the weighted problem. We provide a new separator result, which shows how to find a closed curve that travels along some edges but also jumps

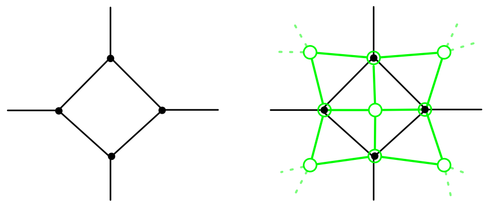

Figure 1: On the left is a fragment of an embedded graph G. On the right is the corresponding fragment of $\tilde{G}$, where we have added vertices and edges of the face-vertex incidence graph of G.

across faces. We bound both the weight of the edges traversed and the number of jumps. A similar separator result formed the basis of the weighted TSP PTAS [2] but that separator result was based on the weight of the entire graph, whereas we need one based on the weight of only the backbone. Our separator result is stated and proven in Section 2.

In order to handle bounded-genus graphs, we also need a low-weight planarization result; in Section 3, we show that there is a low-weight subgraph whose removal renders the graph planar; the subgraph consists of a small number of connected components.

1.5 Preliminaries and Notations

Throughout the article, we consider graphs $G = (V, E)$ that are undirected multigraphs, possibly with loops. We consider weights on edges and vertices.

For any graph G and subset of edges S, we use G/S to denote the graph resulting from the contraction of the edges of S in G.

Unless otherwise specified, all surfaces are implicitly assumed to be connected, orientable, and without boundary. An *embedding* of G on a surface S is a drawing of G on S with no crossing. Namely, the images of the vertices of G in S are pairwise distinct and the image of an edge u, v is a path on S which starts and ends at u and v and does not intersect any other path or vertex. We say that an embedding $\mathcal{E}$ of a graph G *extends* an embedding $\mathcal{E}'$ of a subgraph G' of G if the images of the vertices and edges of G' by $\mathcal{E}'$ are the same in $\mathcal{E}$.

An embedding is called *cellular* if every face is homeomorphic to an open disk. Every graph of genus g has a cellular embedding on a surface of genus g. In this paper, we consider only such cellular embeddings. A graph of genus 0 is a planar graph.

The following definitions are illustrated in Figure 1. Given a connected graph $G = (V, E)$ cellularly embedded on a surface, the *radial graph* (a.k.a. the *face-vertex incidence graph*) of G is the embedded graph whose vertex set includes the vertices of G and also a vertex v_f for each face f of G; it contains an edge between v and v_f if v is a vertex of G that is incident to the face f of G. We define $\tilde{G} = (\tilde{V}, \tilde{E})$ to be the union of G and its radial graph. That is, $\tilde{G}$ contains the vertices of the radial graph and the edges of G and of the radial graph.

In a walk in a graph G, a *spur* is the occurence of a single edge used twice consecutively in oppposite directions.

A *branch decomposition* [53] of a graph is a maximal noncrossing collection of subsets of edges of the graph, equivalently a rooted binary tree in which each node corresponds

Table 1: Summary of our results. All the problems are APX-hard in general graphs and the approximation ratios of the Weighted Dominating Set, the Vertex-Weighted Connected Vertex Cover and the Vertex-Weighted Connected Dominating Set problems are $\Omega(\log(n))$ for general graphs assuming P $\neq$ NP. All the problems are NP-hard in planar graphs. Previous to our work, polynomial-time approximation schemes were known [17] for the unweighted versions of these problems in planar graphs, and "almost-PTASs" were known for bounded-genus graphs. For each of the weighted versions, the best approximation known before our work was the approximation for general graphs. We obtain PTASs for bounded-genus weighted graphs, except for feedback vertex set, where the algorithm is restricted to weighted planar graphs.

Problem	General Weights	
	Prev. (for general graphs)	**New**
(Edge-weights) Tree Cover	2 [46]	$1+\varepsilon$
(Edge-weights) Tour Cover	3 [39]	$1+\varepsilon$
(Vertex-weights) Connected Dominating Set	$O(\log(n))$ [32]	$1+\varepsilon$
(Vertex-weights) Maximum Leaf Spanning Tree	$O(\log(n))$ [32]	$1+\varepsilon$
(Vertex-weights) Connected Vertex Cover	$O(\log(n))$ [27]	$1+\varepsilon$
(Vertex-weights) Feedback Vertex Set	2 [3]	$1+\varepsilon$

to a subset of edges, and the two children of an internal node correspond to disjoint subsets whose union corresponds to the parent. Each subset of edges in a branch decomposition induces a subgraph of the graph, which we call a *cluster* of the branch decomposition. The *boundary* of a cluster is the set of vertices that are incident both to edges belonging to the cluster and edges not belonging to the cluster. The *width* of a cluster is the number of boundary vertices, and the width of a branch decomposition is the maximum cluster width. The *branchwidth* of a graph is the minimum width of a branch decomposition. The branchwidth w and treewidth t of a graph are related by

$$w - 1 \leq t \leq \lfloor \frac{3}{2} w \rfloor - 1.$$

For fixed w, there is a linear-time algorithm [10] to determine if a graph has branchwidth at most w and, if so, construct a branch decomposition of width at most w. There is a polynomial-time algorithm [53] to find an optimal branch decomposition of a planar graph.

A *noose* of an embedded graph is a Jordan curve that intersects only vertices of the graph and not edges.

Consider a planar embedded graph G. A *sphere-cut decomposition* [19] of a planar graph G is a branch decomposition in which for each cluster there is a noose that encloses exactly the edges in the cluster. The vertices that the noose intersects are exactly the boundary of the cluster. The nooses can be assumed to be mutually noncrossing.

Building on work of Seymour and Thomas [53], Dorn et al. [19] show that every planar embedded graph has a sphere-cut decomposition whose width equals the graphs' branchwidth.

Lemma 1.5. *If G is a planar graph G of branchwidth at most w then $\tilde{G}$, the union of G with the radial graph, has branchwidth at most $2w$.*

Proof. Since G has branchwidth at most w, there exists a sphere-cut decomposition T of width at most w. Consider a cluster C of T and the noose N that encloses the edges of that cluster. The noose can be represented as a cycle in $\tilde{G}$ that uses only edges of the radial graph. The noose passes through at most w vertices, so the cycle passes through at most $2w$ vertices of $\tilde{G}$. Let these vertices be $v_1, v_2, \ldots, v_{2k}$ in the order in which they appear on the cycle, where $v_1, v_3, \ldots, u_{2k-1}$ are original vertices of G and $v_2, v_4, \ldots, v_{2k}$ are the vertices of $\tilde{G}$ representing faces of G. To form the branch decomposition of $\tilde{G}$, we replace the cluster C with $2k+1$ clusters $C_0, C_1, \ldots, C_{2k}$, where C_i is obtained from C by modifying it to include the edges $v_1v_2, v_2v_3, \ldots, v_{i-1}v_i$. The new cluster with the largest boundary is C_{2k}, which has a boundary of size $2k$. In addition, we add the singleton clusters $\{v_1v_2\}, \{v_2v_3\}, \ldots, \{v_{2k-1}v_{2k}\}$. $\square$

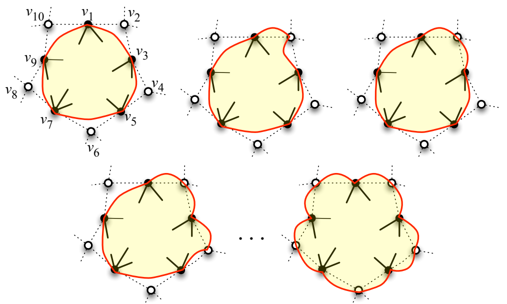

Figure 2: This figure illustrates the proof of Lemma 1.5. The first diagram shows a cluster in the original branch decomposition. Original vertices of G are represented by solid circles, and vertices of $\tilde{G}$ that represent faces of G are represented by open circles. The remaining diagrams show some new clusters added to form the branch decomposition of $\tilde{G}$.

2. A SEPARATOR THEOREM

In this section, we prove the following separator theorem for planar graphs. The balance is with respect to a given *mass* function that assigns a nonnegative number to each face, called the mass of that face.

Theorem 2.1 (Separator Theorem). *Let b and k be integers. Let $G = (V, E)$ be a planar graph, and H denote a connected subgraph of G such that G/H has branchwidth at most $b-1$. Let w_V and w_E be functions assigning nonnegative weights to, respectively, the vertices and the edges of G.*

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be the union of G and its face-vertex incidence graph. Suppose the vertices and faces of $\tilde{G}$ have been assigned nonnegative masses, summing to M, with no face or vertex having mass more than $M/2$.

Then $\tilde{G}$ has a cycle C, which may repeat vertices and edges but does not cross itself and has no spurs, such that:

- *C is a balanced separator: the mass of the vertices of G strictly inside (resp., strictly outside) C is at most $3M/4$;*
- *C is mostly a light piece of H: there exists a set V' of $O(bk)$ vertices of $\tilde{G}$, and a set E' of $O(bk)$ edges of $\tilde{G}$, such that $C-(E' \cup V')$ is a subgraph of H and $w_V(V(C)-V')+w_E(E(C)-E') \le W/k$, where W is the total weight of the vertices and edges in H.*

Moreover, C, V', and E' can be computed in time $O(k^2 n)$.

This theorem is used recursively, with H an $O(1)$-approximation of the ubiquitous problem we consider, to prove Theorem 1.4 (see Section 4). We note that in the problems described in this paper, b is a small constant.

Algorithm 2 Balanced Separator Algorithm for Planar Graphs

```
1: Input: A planar graph $G = (V, E)$ and a subgraph $H$ such that $G/H$ has branchwidth at most $b-1$. Nonnegative weights on vertices and edges of $H$. Masses on the faces of $\tilde{G}$ summing to $M$, each at most $M/2$.
2: $w_1 \leftarrow$ new edge weights on $H$ derived from Lemma 2.2 to represent both edge and vertex weights.
3: $w'_E \leftarrow$ edge weights on $\tilde{G}$ defined in Proposition 2.3
4: $\tilde{G}_1 \leftarrow$ ShortFacesSubgraph($\tilde{G}, H, w'_E$) as per Proposition 2.5.
5: if there exists a face $f$ of $\tilde{G}_1$ with mass at least $M/2$ then
6:     return $C \leftarrow$ fundamental cycle separator of $f$ in $\tilde{G}$ as per Proposition 2.6.
7: else
8:     return $C \leftarrow$ cycle separator in $\tilde{G}_1$ as per Proposition 2.7.
9: end if
10: Output: A cycle separator $C$, fulfilling the requirements of Theorem 2.1.
```

2.1 Reduction to a Simpler Problem with no Vertex Weights

Lemma 2.2. *Without loss of generality we may assume that all vertices have weight zero.*

Proof. Recall that H is connected; let H' be a spanning tree of H. A cycle satisfying the conclusion of the theorem with H' instead of H also satisfies the conclusion of the original theorem. This is because W can only decrease by this operation, and the branchwidth of G/H' equals the branchwidth of G/H (indeed, G/H' can be obtained from G/H by adding loop edges). Henceforth we assume that H is a tree.

The algorithm for Theorem 2.1 proceeds as follows. Select an arbitrary vertex r of H to be the root, direct the edges of H toward r, and for each edge uv of H (directed from u to v), define $\hat{w}_E(uv) = w_E(uv) + w_V(u)$. For every vertex u, define $\hat{w}_V(u) = 0$ for every vertex u. Assume that Theorem 2.1 holds when vertex weights are zero, and apply it with the weight functions $\hat{w}_E$ and $\hat{w}_V$. Let C be the resulting cycle, and let E' be the resulting edge subset, i.e. such that $\hat{w}_E(E(C)-E') \le W/k$ where W is the sum of weights. We prove below that C is also a solution for the weights w_V and w_E, for the same subset E' of edges and for a suitable subset V' of vertices.

Since $|E'| = O(bk)$, we have that $C \cap H$ is made of $O(bk)$ paths of H. Let P be such a path. The path's weight $\hat{w}_E(P)$ includes the weight $w_E(e)$ for every edge e in P. Moreover, for every vertex v of P, since the weight $w_V(v)$ of a vertex v is transferred to the parent edge, $w_V(v)$ is also included in $\hat{w}_E(P)$ except if (i) v is equal to the root r of H, or (ii) v has no outgoing edge in P. Since every vertex of H has at most one outgoing edge, and P has no spur, every vertex of P has at most one outgoing edge. So, when walking along P (oriented arbitrarily), we first encounter an arbitrary nonnegative number of forward edges, and then an arbitrary nonnegative number of backward edges. It follows that at most one vertex of P, let us call it v_P, has no outgoing edge in P. Let V' consist of the root r together with the vertex v_P for each path P comprising $C \cap H$. Then $|V'| = O(bk)$ and the weight of $C - (V' \cup E')$ with respect to w_V and w_E is at most the weight of $C - E'$ with respect to $\hat{w}_E$, which is at most W/k. ☐

Let c be a a constant $c \ge 4$ to be determined.

Proposition 2.3. *Finding a balanced separator satisfying the Separator Theorem (Theorem 2.1) can be reduced to finding a balanced separator cycle in $\tilde{G}$ that has weight at most W/k with respect to the edge-weight assignment*

$$w'_E(e) = \begin{cases} \min\{w_E(e), W/(cbk^2)\} & \text{if } e \in E(H) \\ W/(cbk^2) & \text{otherwise} \end{cases} \qquad (1)$$

Proof. By Lemma 2.2, we can assume that there are no vertex weights. Assume that we find a cycle C as above. Let E' be the set of edges e used by C such that $w'_E(e) = W/(cbk^2)$. Since C has weight at most W/k, we have $|E'| \le cbk$. For each edge $e \in C - E'$, we have $w'_E(e) = w_E(e)$. Since C has weight at most W/k, we have $w_E(C - E') \le W/k$. ☐

2.2 Adding Edges to Reduce the Weight of Face Boundaries

The algorithm described in Proposition 2.3 first selects edges to add to H so as to ensure that each face of the resulting graph has small weight.

Let $\ell = W/ck^2$. Every edge has weight at most ℓ/b.

Lemma 2.4. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. Let f be a face of $\tilde{H}$ with boundary weight at least $(12+\frac{3}{b})\ell$ with respect to w'_E. Then there are two vertices u and v of $\tilde{G}$ on the boundary of f, and a path p in $\tilde{G}$ lying in f (possibly touching its boundary), such that:*

- *each of the two paths between u and v on the boundary of f has weight at least 3ℓ;*
- *p has at most $2b$ edges.*

Moreover, p can be computed in time linear in the complexity of the subgraph of $\tilde{G}$ inside f.

Proof. Let ∂f be the closed walk that is the boundary of f. We write ∂f as the concatenation of four paths N, W, S, E in this order, such that each of these paths has weight at least 3ℓ. (To prove that this is possible, first take N, W, and S with weight between 3ℓ and $(3+\frac{1}{b})\ell$, which is always possible since each edge has weight at most ℓ/b; then the remaining part E has weight at least 3ℓ, since the boundary of ∂f has weight at least $(12 + \frac{3}{b})\ell$.)

Let $\tilde{G}'$ be the part of $\tilde{G}$ inside or on the boundary of f. Similarly, let $G' := G \cap \tilde{G}'$. Since G/H has branchwidth at most $b-1$, $G'/\partial f$ has branchwidth at most $b-1$. By Lemma 1.5, it follows that $\tilde{G}'/\partial f$ has branchwidth at most $2b-2$.

Observe that the distance from any vertex of N to any vertex of S along ∂f is at least 3ℓ. Assume that there is no path p as stated in the lemma. Then every path in $\tilde{G}'$ connecting N to S has at least $2b+2$ vertices. This implies that any vertex cut separating W and E has at least $2b$ vertices. (Indeed, any vertex cut of size j in $\tilde{G}'$ separating W and E corresponds to a closed curve separating W and E in the plane, intersecting j vertices of $\tilde{G}'$; since $\tilde{G}'$ is a triangulation, except for the outer face, the part of that curve inside f can be pushed to $\tilde{G}'$, leading to a path of hop-length j.) Menger's theorem now implies that there are at least $2b$ vertex-disjoint paths between W and E.

Similarly, there are at least $2b$ vertex-disjoint paths between N and S. This implies the existence of a grid minor of size $2b \times 2b$ in $\tilde{G}'$ (similar arguments were used elsewhere [9, p. 88], [13, Proof of Theorem 3.1], and seem to originate from Robertson, Seymour, and Thomas [50]), hence a grid minor of size $2b \times 2b$ in $\tilde{G}'/\partial f$, which contradicts the fact that $\tilde{G}'/\partial f$ has branchwidth at most $2b-2$. So there exists such a path p. Computing such a path takes linear time using two shortest path computations in the planar graph $\tilde{G}'$ [34]. $\Box$

Proposition 2.5. *There is an $O(k^2 n)$ algorithm that computes a subgraph $\tilde{H}$ of $\tilde{G}$ containing H such that:*
- *$\tilde{H}$ has weight at most $2W$, and*
- *every face boundary of $\tilde{H}$ has weight at most $(12+\frac{3}{b})W/ck^2$ (w.r.t. w'_E).*

Proof. Initially, let $\tilde{H} := H$. We iteratively apply Lemma 2.4 by adding edges of the path p in $\tilde{H}$, until every face boundary of $\tilde{H}$ has weight less than $(12 + \frac{3}{b})\ell$. The path p has weight at most 2ℓ. This operation splits the face f into at least two faces, among which some number m have boundary length at least $(12 + \frac{3}{b})\ell$.

We claim that the value of $\varphi := \sum_f (w'_E(\partial f) - 3\ell)$, where the sum is over all faces f of weight at least $(12 + \frac{3}{b})\ell$, decreases by at least ℓ when adding p. Indeed, this is clear if $m = 0$; if $m = 1$, the new face f' with length at least $(12+\frac{3}{b})\ell$ satisfies $w'_E(\partial f') \leq w'_E(\partial f)+2\ell-3\ell = w'_E(\partial f)-\ell$; and if $m \geq 2$, the contribution of the m new subfaces to φ is at most $w'_E(\partial f) + 2 \cdot 2\ell - 3m\ell$.

Thus the total number of iterations is at most $2W/\ell = 2k^2$. At each step, we add at most $2b$ edges, each of weight $W/(cbk^2)$, so the total weight of the added edges is at most $4W/c$. Since $c \geq 4$, the total weight of $\tilde{H}$ is at most $2W$. The time complexity follows from the fact that there are $O(k^2)$ iterations. $\Box$

The algorithm of Proposition 2.5 produces a subgraph $\tilde{H}$ of weight $\tilde{W} \leq 2W$.

2.3 A Balanced Cycle Separator for Weighted Planar Graphs with Light Faces

Recall that the faces and vertices of $\tilde{G}$ have been assigned nonnegative masses, that M is the sum of masses, and that no single mass exceeds $M/2$. Our goal is to give a separator algorithm for the subgraph $\tilde{H}$ whose existence is guaranteed by Proposition 2.5. Recall that the total weight $\tilde{W}$ of $\tilde{H}$ is at most $2W$

Note that each face of $\tilde{H}$ is (essentially) the union of a collection of faces and vertices and edges of $\tilde{G}$. We define the mass of a face of $\tilde{H}$ to be the sum of the masses of the corresponding faces and vertices of $\tilde{G}$.

There are two cases: when a face of $\tilde{H}$ has mass greater than $M/2$ and when no such face does. In the first case, we use a simple construction based on sphere-cut decomposition.

Proposition 2.6. *Suppose $\tilde{H}$ has a face f whose mass is greater than $M/2$. Then there is a cycle C, which may repeat vertices and edges but does not cross itself and has no spur, of weight $4\tilde{W}/k^2$, such that the mass of the faces inside (resp., outside) C is at most $3M/4$. Moreover, C can be computed in linear time.*

Proof of Proposition 2.6. Let $\tilde{G}_f$ be the subgraph of $\tilde{G}$ consisting of the interior and boundary of f. Since f has mass greater than $M/2$, every face of $\tilde{G}_f$ that is not part of f has mass less than $M/2$. Let L be the graph obtained from $\tilde{G}_f$ by contracting all but one of the edges of the boundary of f. Since $\tilde{G}/H$ has branchwidth at most $2b-2$, so does L. Since f has mass greater than $M = 2$, the face of L corresponding to the part of $\tilde{G}$ not in f has mass at most $M/2$. Thus each face of L has mass at most $M/2$.

Consider a sphere-cut decomposition of L. It defines a rooted binary tree in which each node corresponds to a noose and a cluster consisting of the edges enclosed by the noose. Define the mass of a node of the binary tree to be the mass of the faces fully enclosed by, or intersecting, the corresponding noose. Let v be a deepest node in the binary tree such that v's mass is greater than $M/2$. Among v's two children let v_1 be the child with the greater weight. The sum of the masses of v's two children is greater than or equal to the mass of v, thus the mass of v_1 is at least $M/4$.

Let C_1 be the Jordan curve corresponding to v_1. The total mass of the faces stricly enclosed by C_1 is at most the mass of v_1, which is at most $M/2$. The total mass of the faces strictly outside C_1 equals M minus the mass of v_1, which is at most $M - M/4 = 3M/4$.

We construct a cycle C_2 in L from C_1 by pushing each part of the curve which passes through a face onto part of the face's boundary. We sequentially choose the direction in which to push faces: each face is added to the currently lighter side. As the mass of each face is at most $M/2$, the new cycle is 3/4 balanced. As L has maximum face degree 3, the curve C_1 passes through each face at most once, so the resulting cycle C_2 is non-self-crossing. If any spurs are formed in C_2, we (iteratively) remove them. Removing a spur does not affect the balance at all, and can only reduce the weight of the cycle.

Since L has branchwidth at most $2b-2$, the curve corresponding to v passes through at most $2b-2$ vertices, and thus at most $2b-2$ faces. Since each face has degree at most 3, each path through a face is pushed to at most 2 edges. Thus C_2 contains at most $4b$ edges. Since each edge has weight at most $W/(cbk^2)$, C_2 has weight at most $4W/ck^2$.

C_1 can by lifted to a cycle C in $\tilde{G}$ with the same balance by adding to it some (possibly empty) part of the boundary of f. Since the total weight of the boundary of each face in $\tilde{H}$ is at most $(12 + \frac{3}{b})\ell$, C has weight at most W/k for an appropriate choice of the constant c. Each step of the algorithm implied in the proof can be implemented in linear time. □

If no face is massive, we use a variation of Miller's simple cycle separator theorem [44]. The main differences are that we do not require 2-connectivity, and that the edges are weighted. The proof is adapted from the simplified proof of Miller's theorem in [38].

Proposition 2.7. *Suppose that no face of $\tilde{H}$ has mass larger than $M/2$. Then there is a cycle C, which may repeat vertices and edges but does not cross itself and has no spur, of weight $O(\tilde{W}/k)$, such that the mass of the faces inside (resp., outside) C is at most $3M/4$. Moreover, C can be computed in linear time.*

Before proving this proposition, we show why it, together with Proposition 2.6, implies Theorem 2.1:

Proof of the Separator Theorem (Theorem 2.1). Let $\tilde{H}$ be the graph obtained after applying Proposition 2.5. By Proposition 2.3, it is sufficient to show that there exists a balanced cycle separator C of total weight W/k in $\tilde{G}$ (with respect to w'_E). If one of the faces of $\tilde{H}$ has mass at greater than $M/2$, applying Proposition 2.6 yields such a cycle. Otherwise, since $\tilde{H}$ is a subgraph of $\tilde{G}$, it suffices to find such a C in $\tilde{H}$. The advantage is that the total weight $\tilde{W}$ of $\tilde{H}$ is linear in W, the weight of H, while each face boundary has weight $O(\hat{W}/k^2)$. So applying Proposition 2.7 gives a cycle of weight W/k, as needed. □

Proof of Proposition 2.7. Let T be a breadth-first search in the face-vertex incidence graph J of $\tilde{H}$, rooted at an arbitrary face f; for clarity of exposition below, we assume that f is the outer face of $\tilde{H}$ in the planar drawing that we consider (thus the notions of "inside" and "outside" are well defined). We define the *level* of f to be zero, and the levels of the other vertices and faces of $\tilde{H}$ by induction: The level of a vertex of $\tilde{H}$ is equal to one plus the level of the parent face in T, and the level of a face of $\tilde{H}$ is equal to the level of the parent vertex in T.

Define a mass function on vertices of J in which vertices corresponding to faces of $\tilde{H}$ have mass equal to the mass of the corresponding faces, and those corresponding to vertices of $\tilde{H}$ have mass zero. There exists a simple cycle $\tilde{C}$ in J consisting of a path in T and an edge e not in T such that the mass on either side of the cycle is at most $2M/3$ [40]. Such a cycle is called a *fundamental cycle*.

Let i be an integer. The set of faces of $\tilde{H}$ of level at least i can be partitioned into regions, by declaring that two such faces are in the same component if they share an edge, and extending this relation by transitivity. A *component of level i* is the topological closure of such a region.

We claim that the boundary ∂K of such a component K is a simple cycle in $\tilde{H}$. Since K is the closure of a union of faces, ∂K is a subgraph of $\tilde{H}$ with each vertex of even degree. If some vertex v has degree at least four in ∂K, then v has level i, and its incident faces all have level i and $i-1$. Because v has degree at least four, there are two faces f' and f'' of $\tilde{H}$ with level i that are separated by faces of level $i-1$ in the cyclic ordering around v. Since f' and f'' are in K, there is a simple topological cycle γ passing through f', v, and f'', in this order, entirely lying in the interior of K (except at v). But then all vertices and faces inside γ must have level at least i, which contradicts the assumption. So ∂K is the disjoint union of cycles. Two such cycles cannot be nested, for a similar reason, and they cannot be separated as well, because their interiors would not be connected to each other. So ∂K is a single simple cycle in $\tilde{H}$. This proves the claim.

Since $\tilde{C}$ is a fundamental cycle in J, the levels of its vertices and faces are increasing and then decreasing when walking along $\tilde{C}$ starting from the common ancestor in T of the endpoints of e. Therefore, $\tilde{C}$ enters the interior of at most one component at a given level i.

Let $i_{min}-1$ and i_{max} be the minimum and maximum levels faces in $\tilde{C}$; for each i, $i_{min} \le i \le i_{max}$, let K_i be the (unique) component at level i penetrated by $\tilde{C}$. Moreover, let $K_{i_{min}-1} = F(\tilde{H})$ and $K_{i_{max}+1} = \emptyset$. The K_i's are nested, and the boundaries ∂K_i of the K_i's, for $i_{min} \le i \le i_{max}$, form disjoint simple cycles. Indeed, by construction, a vertex on ∂K_i is incident with some faces of level $i-1$, some faces of level i, and no face of other levels.

Let i_{med} be such that $i_{min}-1 \le i_{med} \le i_{max}$, and the mass of the faces of $\tilde{H}$ inside $K_{i_{med}+1}$ or outside $K_{i_{med}}$ is at most $3M/4$. (For this purpose, one can let i_{med} be as large as possible such that the mass of the faces of $\tilde{H}$ outside $K_{i_{med}}$ is at most $3M/4$.) Let i_- be the largest level smaller or equal to i_{med} such that ∂K_{i_-} has weight at most $W_1/8k$. Similarly, let i_+ be the smallest level larger or equal to $i_{med}+1$ such that ∂K_{i_+} has weight at most $W_1/8k$. Then the total weight of $\partial K_{i_+} \cup \partial K_{i_-}$ is at most $2W_1/8k$, which is at most $W/2k$.

Since $K_{i_-+1}, K_{i_-+2}, \ldots, K_{i_+-1}$ each have weight larger than $W_1/8k$, there can be at most $8k$ such levels, so we have $i_+ - i_- \le 8k+1$.

We now consider the part of $\tilde{C}$ inside K_{i_-} but outside K_{i_+}, which consists of two paths in J. We push each of these two paths into $\tilde{H}$: Each time such a path traverses a face of $\tilde{H}$, we push the corresponding part onto one of the face boundaries. We sequentially choose the direction in which to push faces: each face is added to the currently lighter side. Since the mass of each face is at most $M/2$, the new cycle is 3/4 balanced. Further, as $\tilde{C}$ enters each face at most once, the resulting cycle is non-self-crossing. If any spurs are formed, we (iteratively) remove them. Removing a spur does not affect the balance at all, and can only reduce the weight of the paths. Let P_1 and P_2 be the resulting two paths. Each of them has weight at most $(8k+1)\cdot(12+\frac{3}{b})W/ck^2$ since the corresponding part of $\tilde{C}$ we pushed was traversing at most $8k+1$ faces of $\tilde{H}$, each of boundary weight at most $(12+\frac{3}{b})W/ck^2$. By choice of c, we can ensure that the weight of these two paths is at most $W/2k$.

Let $S := P_1 \cup P_2 \cup \partial K_{i_+} \cup \partial K_{i_-}$. By construction, S has weight at most W/k. S separates J into four pieces (some of which can be empty or disconnected):

- the part of J strictly outside K_{i_-};
- the part of J strictly inside K_{i_+};
- the part of J strictly inside K_{i_-}, strictly outside K_{i_+}, and strictly inside $\tilde{C}$;
- the part of J strictly inside K_{i_-}, strictly outside K_{i_+}, and strictly outside $\tilde{C}$.

By construction, each such piece encloses faces of mass at most $3M/4$. The three smallest pieces together have face mass at most $3M/4$, and the largest one at most $3M/4$.

Thus, we can take for separator the subset of S that bounds the larger of these pieces; this is indeed a balanced separator, and a non-self-crossing cycle without spur in $\tilde{H}$. Each step of the algorithm implied in the proof can be implemented in linear time. $\square$

3. PLANARIZATION THEOREM

In this section, we show the following theorem.

Theorem 3.1. *Let $b > 1$ be a constant. There exists a polynomial-time algorithm for the following: The algorithm is given a positive integer parameter k, an edge-weighted graph G that is cellularly embedded on a surface of genus g, and a connected subgraph H of G such that G/H has branchwidth at most $b-1$.*

The algorithm outputs a subgraph S of $\tilde{G}$ such that $G - S$ is planar, S contains at most $O(g^2 + k)$ edges not in H, S has at most $O(g^2 + k)$ connected components, and the total weight of S is at most $g^{O(1)}W/k$ where W is the total weight of H.

Using the argument of Section 2.1 (which does not use planarity), we can assume that all vertex weights are zero.

As in Proposition 2.3, the algorithm for Theorem 3.1 assigns edge-weights to $\tilde{G}$ according to Equation 1. Edges not in H have weight W/cbk^2. The algorithm then finds a subgraph S whose weight is $O(W/k)$ with respect to this edge-weight assignment. As a consequence, the number of edges in S that are not in H is $O(g^2 + k)$.

The next lemma follows from [49, Theorem 4.1].

Lemma 3.2. *Consider a graph G that is cellularly embedded on a surface of genus g and a subgraph H of G such that G/H has treewidth at most t. Let f be a face of H of genus at least one and G_f be the subgraph of G induced by f and its interior. There exists a non-separating cycle in $\tilde{G}_f$ that intersects at most $O(t)$ vertices of G_f.*

Proposition 3.3. *Consider a graph G_0 with r connected components embedded on a surface of genus g. Then the number of faces of G_0 that are not disks is $O(r+g)$. Moreover, the total number of boundary components of all non-disk faces is at most $O(r+g)$.*

Proof. For any graph G, let $\varphi(G)$ denote the sum, over all non-disk faces f of G, of the number of boundary components of f. We will prove that $\varphi(G_0) = O(g+r)$.

First, we define a graph G_1 obtained from G_0 by adding $r-1$ edges, so that G_1 is connected. Observe that $\varphi(G_0) \leq \varphi(G_1) + O(r)$: indeed, consider the addition of an edge e in some face f, during the transformation of G_0 into G_1. Edge e connects two distinct boundary components of f, so it does not separate f. Moreover, the number of boundary components of f decreases by at most one.

Second, we define a graph G_2 obtained from G_1 by contracting the edges of a spanning tree of G_1; the graph G_2 has a single vertex, and we have $\varphi(G_1) = \varphi(G_2)$.

Third, we iteratively apply the following operation to G_2: While there is a disk of G_2 bounded by a single loop, we remove that loop, and similarly while there is a disk of G_2 bounded by exactly two loops, we remove one of the loops. The non-disk faces of this new graph, G_3, have the same topology as those in G_2, so $\varphi(G_2) = \varphi(G_3)$.

Under these conditions, it is known [12, Lemma 2.1] that the number of loops in G_3 is $O(g)$; in particular, $\varphi(G_3) = O(g)$, which by the above equalities implies $\varphi(G_0) = O(r+g)$.

That immediately implies that the number of faces of G_0 that are not disks is $O(r+g)$, hence the proposition holds. $\square$

We can now prove the following lemma.

Lemma 3.4. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. Let f be a face of $\tilde{H}$. Assume f has a boundary component f_0 with weight at least $(12 + \frac{3}{b})\ell$. Then there exist two vertices u and v of $\tilde{G}$ on the boundary of f, and a path p in $\tilde{G}$ with at most $2b$ edges and lying in f, such that:*

- *if u and v are both in f_0, then each of the two paths between u and v in f_0 has weight at least 3ℓ;*
- *otherwise, u and v belong to different boundary components of f, and the path p intersects the boundary of f only at u and v.*

Moreover, p can be computed in time linear in the complexity of the subgraph of $\tilde{G}$ inside f.

Proof. The proof is similar to the proof of Lemma 2.4. We explain how to adapt its proof. Observe that by Proposition 3.3, the number of boundary components is at most $O(g^2)$. Thus, the graph G_f which consists of the interior of f where each boundary component is contracted to a vertex has branchwidth $O(g^2)+b$. Indeed, the graph corresponding to the interior of f where all the boundary components are contracted into a single vertex has branchwidth at most b. Form a width-b branch decomposition of this graph. When each boundary component is represented by a single vertex, the width increases by at most the number of such vertices.

Now, we apply the argument of Lemma 2.4. If the short path that is found does not intersect any vertex resulting from the contractions, we just return the path and it satisfies the conditions of the lemma. Otherwise, consider a short path from u to v intersecting at least one vertex resulting from the contractions, and return a shortest subpath connecting a vertex u' in f_0 with a contracted vertex v'. This path connects two different boundary components of f, without intersecting the boundary of f except at its endpoints, thus satisfying the conditions of the lemma. $\square$

We can derive the following proposition whose proof resembles that of Proposition 3.5.

Proposition 3.5. *Let $\tilde{H}$ be a subgraph of $\tilde{G}$ containing H. There exists an algorithm to compute a subgraph $\tilde{H}_1$ of $\tilde{G}$ containing $\tilde{H}$ such that:*

- *$\tilde{H}_1$ has $O(b(k^2+g))$ edges not in $\tilde{H}$;*
- *every boundary component of every face f of $\tilde{G}_1$ has weight at most $(12 + \frac{3}{b})W/ck^2$.*

The running time of the algorithm is $O((k^2+g)n)$.

Theorem 3.6. *Let k be an integer. Consider a graph G with positive edge weights that is cellularly embedded on a surface S of Euler genus g and such that every face is a disk. Let W denote its total weight, and assume that every face has boundary weight at most W/k^2. There exists a subgraph G' of G, such that cutting S along G' gives a surface with genus zero (possibly with several boundary components), with the following properties: G' has weight $O(\sqrt{g}W/k)$, and has at most g connected components. Furthermore, G' can be computed in linear time.*

G' is called a *planarizing* subgraph of G.

Proof. The proof is a refinement on a result by Eppstein [23]; see also [24]. Let J be the face-vertex incidence graph of G. Let r be an arbitrary vertex of G, and let T be a breadth-first search tree in J rooted at r. We define the *level* $\ell(u)$ of a face or vertex u of G to be the number of edges of the path in T from r to u.

Let E be the set of edges of J. For each edge $uv \in E$, let $L(uv)$ be the loop rooted at r that is the concatenation of the path from r to u in T, edge uv, and the path from v to r in T. Loop $L(uv)$ has $\ell(uv) = \ell(u) + \ell(v) + 1$ edges. Let C be the primal edges of a *maximum* spanning tree of $(E - T)^*$, where the weight of an edge $(uv)^* \in E^*$ equals $\ell(uv)$. Finally, let $X := E - (T \cup C)$.

Euler's formula implies that $|X| = g$. It is known from [25, Section 3.4]) (and not hard to see) that $\bigcup_{uv \in X} L(uv)$ cuts S into a topological disk, and that an alternative greedy way to compute X is to iteratively add to X the edge $u'v'$ with smallest value of $\ell(u'v')$ such that $\bigcup_{uv \in X} L(uv) \cup L(u'v')$ does not disconnect the surface.

Let $M := 2\left\lceil k/(2\sqrt{g})\right\rceil$. Recall that W denotes the total weight. We choose an even $i \in \{0, \ldots, M-1\}$ such that the subgraph G_1 of G induced by the vertices of level equal to i modulo M has weight $O(\sqrt{g}W/k)$.

For each edge $uv \in X$, we consider the smallest level $i_{uv} \geq \max(\ell(u), \ell(v)) - M$ that is equal to i modulo M. Define $L'(uv)$ to be the part of $L(uv)$ of level at least i_{uv}. Since $L'(uv)$ traverses $O(k/\sqrt{g})$ faces of G, each of boundary weight $O(W/k^2)$, we can "push" $L'(uv)$ to a walk $L'_p(uv)$ of G, of weight $O(W/(k\sqrt{g}))$.

Let G_2 be the union of the subgraph G_1 and of the walks $L'_p(uv)$, for $uv \in X$. By construction, and since $|X| = g$, the weight of G_2 is $O(\sqrt{g}W/k)$. We will now (i) prove that cutting S along G_2 results in a genus zero surface (possibly with several boundary components), and then (ii) extract from G_2 a subgraph still having that property, but having $O(g)$ connected components.

For (i), let i' be equal to i modulo M. It suffices to prove that the part of the surface S that is the closure of the union of the faces of levels between $i'+1$ and $i'+M-1$, minus G_2, has genus zero; or, equivalently, minus G_1 union the $L'(uv)$ for $uv \in X$. Actually, that latter surface is contained in the closure of the faces of G at level at most $i'+M-1$ minus the union of the loops $L(uv)$ with $i_{uv} \leq i' + M$, so it suffices to prove that this latter surface, S' has genus zero. To simplify the discussion, we attach a disk to each boundary of S'. The restriction of T to S' is also breadth-first search tree of the restriction of J to S'. If S' has positive genus, then it has a non-separating loop based at r that has the form $L(u'v')$ for some edge $u'v'$ [11, Lemma 5]; that loop is also non-separating in S minus the loops $L(uv)$ with $i_{uv} \leq i' + M$. But this contradicts the greedy algorithm mentioned above (which should have inserted $u'v'$ in X, since $\ell(u'v') \leq i' + M$). This contradiction proves (i).

For (ii), we consider an inclusionwise maximal subgraph G_3 of G_2 such that cutting S along G_3 results in a connected surface (which therefore has genus zero as well); computing G_3 can be done in linear time, by computing a spanning tree of the "dual" graph of G_2 and keeping the primal edges of the complement. Finally, let G' be obtained from G_3 by removing any connected component of G_3 that is a tree. Cutting S along G' still results in a genus zero surface, so G' is planar. Moreover, G' has at most g cycles, because otherwise the complement of G' would be disconnected (by definition of the genus). Since each connected component of G' contains a cycle, G' has at most g connected components.

Finally, G' can be computed in linear time. □

We can now prove the theorem.

Proof of Theorem 3.1. We consider the following algorithm to construct S.

1. $S \leftarrow \emptyset$
2. While there is a face with positive genus: apply Lemma 3.2 to H in order to obtain a graph H' where each face has genus 0.
3. While there exists a face whose boundary has large weight: apply Lemma 3.4. Obtain a subgraph H'' with $O(g)$ connected components, that contains H, and of maximum face weight at most $O(g^2W/k^2)$.
4. For each face f of H'', if f is not a disk, then add the entire boundary of f to S and remove f. Obtain H'''.
5. Apply Theorem 3.6 to each connected component of H''' to obtain a planarizing subgraph S', and add it to S.
6. Return S

We prove that the subgraph S satisfies the conditions of Theorem 3.1.

We first argue that iteratively applying Lemma 3.2 yields a graph H' of total weight at most $O(g^2W/k)$. Since for each face we add a non-separating cycle, the total genus of all the faces decreases by one at each iteration. By Lemma 3.2, the path added is short and so, the total weight of H' is bounded by $O(g^2W/k)$ and each face of H' has genus 0.

By applying Lemma 3.4 we either decrease the number of connected components of the boundary of a face or we reduce the weight of the face. By Proposition 3.3, the total number of connected components of all the faces is at most $O(g^2)$, thus the total weight of H'' is at most $O(g^4W/k)$.

We now turn to the analysis of the cost incurred by Step 4. By Proposition 3.3, there are at most $O(g)$ such faces. Again since the face weight of H'' is at most $O(W/k^2)$, the total weight added to S at step 4 is at most $O(gW/k^2)$.

Finally, observe that in the remaining graph, by Step 4 each face of H''' is a disk and contains a subgraph of G of genus 0. Moreover by Step 3, the maximum face weight is at most $O(W/k^2)$. It is thus possible to apply Theorem 3.6 in order to obtain a subgraph S' of H''' of total weight at most $O(\sqrt{g}W/k)$ and such that $H''' - S'$ is planar. The total number of connected components of S' is at most $O(k)$.

Since the number of connected components added at Step 4 is $O(g^2)$, the total number of connected components of S is thus $O(g^2 + k)$. □

4. BRANCHWIDTH REDUCTION

In this section we prove Theorem 1.4: we show that, for constants g, b, ϵ, there is a polynomial-time algorithm that, given a genus-g edge/vertex-weighted graph G_0 and a connected subgraph H_0 such that G_0/H_0 has branchwidth at most $b-1$, outputs a subgraph K of H_0 of weight at most ϵ times the weight of H_0 such that G_0/K has branchwidth $O(\log n)$, where n is the number of vertices of G_0. We give a procedure that returns a branch decomposition of G_0/K.

First we assume the graph is planar. At the end of this section, we discuss the case of positive genus.

An overview: The algorithm of Theorem 1.4 uses the algorithm of Theorem 2.1 to recursively find separators and uses them to decompose the graph into clusters of a branch decomposition. The boundaries of these clusters are subsets of the vertex sets of the separators. The boundaries might be large but, after contraction of the edges of H in the separator, the size of the boundary becomes small. Because the separators are balanced, the recursion depth is logarithmic.

In order to ensure the boundaries in the contracted graph remain small, the algorithm uses a variant of a strategy of [26]. The variant is described, e.g., [38] (and used elsewhere previously); occasionally, instead of ensuring the separator is balanced with respect to the size of the subgraphs, the algorithm ensures that the separator is balanced with respect to the number of vertices appearing on boundaries.

More details: We describe a recursive procedure to select edges to contract and find a branch decomposition for the contracted graph. The procedure is given a planar embedded graph G and a subgraph H such that G/H has branchwidth at most b. The procedure is also given a subset S of vertices of G, which we call *boundary vertices.*

Algorithm 3 BranchDecomp(G, H, S)

```
1: Input: a planar graph $G$, a subset $H$ of edges, and a set $S$ of vertices
2: if $G$ has at most $c_1$ vertices then return a branch decomposition of $G$ of width $\leq c_1$
3: else
4:     if $|S| > c_2\epsilon^{-1}\log n$ then
5:         assign mass 1 to vertices of $S$ and zero to other vertices
6:     else
7:         assign mass 1 to all vertices of $G$
8:     end if
9:     find a cycle separator $C$ as per Theorem 2.1 using $k = c_3\epsilon^{-1}\log n$
10:    $G' \leftarrow G/$edges of $C \cap H$
11:    $C' \leftarrow$ noose corresponding to $C$ in $G'$
12:    $S' \leftarrow$ vertices of $G'$ on $C'$
13:    $(E_1, E_2) \leftarrow C'$-induced bipartition of edges of $G'$
14:    $(S_1, S_2) \leftarrow C'$-induced bipartition of vertices of $G'$ not in $C'$
15:    $B_i \leftarrow$BranchDecomp$(G'[E_i], H \cap E_i, S_i \cup S')$ for $i = 1, 2$
16:    return $B_1 \cup B_2 \cup (\bigcup B_1 \cup B_2)$
17: end if
```

The initial invocation is BranchDecomp$(G_0, H_0, \emptyset)$ be the initial invocation. In any nonterminal invocation BranchDecomp(G, H, S), the two recursive calls in Line 15 operate on disjoint subsets of H. Therefore, for every level of recursion the invocations operate on disjoint subsets of H_0, so the total weight of these subgraphs is at most the weight of H_0. We will see that the recursion depth is $O(\log n)$. Therefore the total weight of all subgraphs H passed to all invocations is $O(\log n)$ times the weight of H_0.

In Line 16, the procedure takes branch decompositions returned by the recursive calls and adds one additional cluster, the cluster consisting of all the edges in the two branch decompositions. Therefore the procedure returns a branch decomposition of the graph induced on all those edges. Thus the inital invocation returns a branch decomposition of the graph obtained from G_0 by contracting all edges ever contracted during recursive invocations of the procedure.

In any nonterminal invocation BranchDecomp(G, H, S), in Line 9 the procedure finds a cycle separator C using the parameter value $k = c_3\epsilon^{-1}\log n$. The weight of edges of H in C is at most the weight of H divided by k. It follows that, for an appropriate choice of the constant c_3, the total weight of edges in all cycle separators found is at most ϵ times the weight of H_0. This bounds the weight of all edges contracted in Line 10 throughout all invocations.

When the edges of $C \cap H$ are contracted, C becomes a noose C' in the contracted graph G'. The noose C' partitions G' into two edge-induced subgraphs, and also partitions the boundary vertices S. In each of the two recursive calls, the vertices on C' are included as boundary vertices. This implies the invariant that, for any invocation BranchDecomp(G, H, S), any vertex of G_0 that is incident to an edge in G and an edge not in G is a member of S.

By Theorem 2.1, the number of vertices on the noose C' is $O(b\epsilon^{-1}k)$, which is $O(b\epsilon^{-1}\log n)$. Because of Line 5, one can choose the constant c_2 in Line 4 so that there is a constant c_4 such that the number of boundary vertices passed to the procedure never exceeds $c_4b\epsilon^{-1}\log n$, and that no two consecutive recursive invocations execute Line 5. As a consequence of the first statement, every branch decomposition returned has width at most $c_4b\epsilon^{-1}\log n$. As a consequence of the second statement, the recursion depth is $O(\log n)$ as promised.

Finally, consider the case in which the input graph has genus $g > 0$. In this case, the algorithm first applies Theorem 3.1's algorithm to the input graph G and obtain a subgraph S. For each piece L of $G - S$ that is planar, the algorithm recursively applies the planar separator theorem and obtains a set of edges S'_L such that L/S' has bounded branchwidth.

We now argue that the branchwidth of $G/(S \cup \bigcup_L S_L)$ is bounded. For each planar piece L of $G - S$ we take a branch decomposition of L/S_L of small width.

Since by Theorem 3.1, S contains at most $O(g^2 + k)$ connected components, these branch decompositions can be merged to form a branch decomposition of $G/(S \cup \cup_L S_L)$, increasing the width by $O(g^2 + k)$.

5. ALGORITHMIC IMPLICATIONS

A fairly large class of problems to which our metatheorem applies mix structure requirements and domination requirements, and can have several kinds of weights. More specifically, we consider problems that take as input a bounded-genus graph with weights on vertices, edges or faces; a solution must usually be connected, have a connected induced subgraph, or be a tour; and it must dominate all vertices, edges or faces of the graph. To derive PTASs for those problems, we rely on Algorithm 1 and appeal to Theorem 1.3. To have a subgraph H of total weight $O(\text{OPT})$ such that G/H has bounded treewidth, it is sufficient to take an $O(1)$ approximate solution, either available from previous work or obtained by designing $O(1)$-approximation when needed (for example for weighted connected dominating set).

In [51], the authors introduce a new tree-decomposition for graphs embedded on surface, called surface cut decomposition. All the problems listed in Table 1 are *packing-encodable* according to the definition in [51]. Even though this theorem is designed for unweighted versions of the problems, it is straightforward to extend it to weighted versions.

Theorem 5.1. *[51, Theorem 3.2] Every connected packing-encodable problem whose input graph G is embedded in a surface of genus g, and has branchwidth at most b, can be solved in time $g^{O(b+g)}b^{O(g)}n^{O(1)}$.*

5.1 Weighted Connected Dominating Set, Max Weighted Leaf Spanning Tree

Consider the Vertex-Weighted Connected Dominating Set problem defined as follows.

Definition 5.1. Weighted Connected Dominating Set. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a *connected dominating set* is a set of vertices S such that $G[S]$ is connected and such that every vertex of V is in S or adjacent to S. The objective is to find a connected dominating set of minimum weight.

Garey and Johnson's book [28] showed that the problem is NP-Hard, even for bounded degree planar graphs. Guha and Keller [31] obtained a $\log \Delta$ approximation where Δ is the maximum degree of the graph and that no polynomial time algorithm can do better in general graph unless NP $\subseteq$ DTIME$[n^{O(\log\log n)}]$. The vertex-weighted version of the problem received a lot of attention as it has applications in network testing problems (see [47]) and wireless communication problems (see [15]). For the unweighted version of the problem in graphs of bounded genus, a PTAS was obtained through the framework arising from the bidimensionality theory in [17]. A linear kernel was found for planar graphs in [42]. The FPT version of the problem was addressed in [16].

The vertex-weighted version of the problem was also considered by Guha and Keller in a later paper [32] who obtained a $(1.35 + \varepsilon) \log n$ approximation for general graphs and which remained the best approximation ratio until this work. Using Theorem 1.3, we obtain the following result for the connected dominating set problem.

Theorem 5.2. *Let $0 < \varepsilon \le 1/2$ and let g be a fixed integer. There exists an algorithm, based on Algorithm 1, that computes a $1 + \varepsilon$-approximation to the weighted connected dominating set problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

Clearly, any solution in the original graph remains a solution in the contracted graph, and its cost can only be reduced. We show that each of the three conditions of Theorem 1.3 hold, implying Theorem 5.2.

Condition 2. The second condition is ensured by Theorem 5.1.

Condition 3. To prove that the last condition holds, we show that given a graph G, a subgraph $G_1 = (V_1, E_1)$ and a solution S for G/G_1, there exists an Algorithm `Lift` which computes a solution for G of total cost at most $w(S)+w(G_1)$: Since each vertex resulting from the contraction of G_1 has to be dominated, at least one of its neighbor belongs to S. Therefore, we can add all the vertices of G_1 to S and the solution remains connected. Furthermore, since each vertex is dominated in G/G_1 by S and since we add the all the vertices of G_1 to S, all the vertices of G are dominated by $S \cup V_1$. The total cost of the new solution $S \cup V_1$ is $w(S) + w(G_1)$.

Condition 1. To show the first condition, we provide the first known constant factor approximation. Indeed, since for each feasible solution S each vertex of the graph is dominated by $G[S]$, each vertex of $G/G[S]$ is at distance at most 1 from the vertex resulting from the contraction of $G[S]$. Therefore $G/G[S]$ has diameter at most 2. It follows that the branchwidth of $G/G[S]$ is $O(1)$. Therefore, any $O(1)$-approximation for the problem is a connected subgraph H of G such that G/H has branchwidth $O(1)$. We show the following lemma which is immediately subsumed by the Approximation Scheme result (Theorem 5.2).

Lemma 5.3. *There exists an $O(1)$-approximation algorithm for the Vertex-Weighted Connected Dominating Set problem for graphs of genus at most g.*

For any graph G of genus at most g, we first define a *ball* of radius i around a vertex v to be the set of points that are at edge distance at most i from v. We prove the correctness of Algorithm 4.

Algorithm 4 Constant factor approximation algorithm for weighted connected dominating set in bounded genus graphs.

```
 1: Input: A graph $G = (V, E)$ of genus at most $g$, a weight
    function $w : V \to \mathbb{R}_+$.
 2: $B \leftarrow$ set of disjoint balls of radius 1 that is maximal
    under inclusion.
 3: $V_0 \leftarrow \emptyset$
 4: for all ball $b \in B$ do
 5:     $V_0 \leftarrow V_0 \cup \{$ an element of $b$ of minimum weight $\}$
 6: end for
 7: $V_1 \leftarrow$ constant-approximation solution to the Vertex-
    Weighted Steiner Tree problem on $G$ with terminals $V_0$.
 8: $G_1 \leftarrow G/G[V_1]$, $G_1$ has bounded branchwidth by Lemma
    5.6.
 9: $V_2 \leftarrow$ an optimal solution to the problem on $G_1$ using
    Algorithm from Theorem 5.1 for bounded branchwidth
    graphs.
10: Output: $V_2 \cup V_1$
```

Lemma 5.4. *Consider the set of vertices V_0 after step 6 of Algorithm 4. There exists a solution S of value at most $2OPT$ such that $V_0 \subseteq S$.*

Proof. Consider an optimal feasible solution S_{OPT} and a ball $b \in B$. Since S_{OPT} is feasible, $\text{argmin}_{v \in b} w(v)$ is in S_{OPT} or at least one of its neighbors is in S_{OPT}. Hence $S = S_{\text{OPT}} \cup \{v \mid \exists b \in B \text{ s.t } v = \text{argmin}_{v\in b} w(v)\}$ is connected. We now argue that the cost of S is at most twice the cost of S_{OPT}. Again, since S_{OPT} is feasible, either the center of b belongs to S_{OPT} or at least one of its neighbors belongs to S_{OPT} It follows that the sum of the weights of the vertices in $S_{\text{OPT}} \cap b$ is at least $\min_{v\in b} w(v)$ and thus, the sum of the weights of the vertices in $S \cap b$ is at least $2 \min_{v \in b} w(v)$. Therefore, since the balls are disjoint, the total cost of S is at most twice the total cost of S_{OPT}. ☐

Line 7 of the Algorithm is achieved thanks to the following theorem. See also [6].

Theorem 5.5 ([18, Theorem 1])**.** *There exists a polynomial-time constant-factor approximation algorithm for the vertex-weighted Steiner tree problem.*

Lemma 5.6. *Consider the set V_1 at step 8 of the algorithm. $G/G[V_1]$ has bounded branchwidth.*

Proof. Note that by the maximality condition of Step 1 of Algorithm 4, we have that each vertex of the graph is at distance at most 1 from some ball b and so, at distance at most 3 from all the vertices of some ball b.

Because $G[V_1]$ is connected, the contraction of $G[V_1]$ in $G/G[V_1]$ results in a single vertex which is at distance at most 3 from all the other vertices of $G/G[V_1]$. Hence, the diameter of $G/G[V_1]$ is constant and so, the branchwidth of $G/G[V_1]$ is constant. □

Proof of Lemma 5.3. Lemma 5.4 and Theorem 5.5 ensure that $\text{cost}(S_1) \leq 12 \cdot \text{OPT}$. Moreover, Lemma 5.6 and Theorem 5.1 ensure that $\text{cost}(S_2) \leq \text{OPT}$. The running time of the algorithm follows directly from Theorems 5.5 and 5.1. □

The weighted connected dominating set problem is also related to the maximum weighted leaf spanning tree defined as follows.

Definition 5.2. Maximum Weighted Leaf Spanning Tree. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a *weighted leaf spanning tree* is a spanning tree of G whose cost is defined as the sum of the weights of its leaves. The objective is to find a leaf spanning tree of maximum weight.

The unweighted version of the problem has been studied in a series of results (see for example [8, 43]). The FPT version of the problem has also been extensively studied, for example in [21, 7].

Using an observation from [20] for the unweighted case, we derive the analogous observation for the weighted case, Lemma 5.8. Using Lemma 5.8, it is easy to derive Theorem 5.7 from the proof of Theorem 5.2.

Theorem 5.7. *Let $0 < \varepsilon \leq 1/2$ and g be a fixed integer. There exists an algorithm, based on Algorithm 1, that computes a $1+\varepsilon$-approximation to the maximum weighted leaf spanning tree problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

This lemma is standard and was proven in previous results on maximum weight leaf spanning tree.

Lemma 5.8. *Let $G = (V, E)$ be a graph with vertex weights $w : V \to \mathbb{R}^+$. Let W denote the sum of the weights of the vertices of G. The sum of the value of the optimal maximum weighted leaf spanning tree and the value of the optimal connected dominating set is equal to W.*

5.2 Tour Cover and Tree Cover

Definition 5.3. Tree cover. Given a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}^+$, a *tree cover* is a set of edges S such that $G[S]$ is connected and such that for each $(u, v) \in E$, $\exists e \in S$ such that e shares an endpoint with (u, v). The tree cover problem asks for a tree cover of minimum weight.

In the *tour cover* problem, the solution is required to form a tour instead of a tree. The Tour and Tree cover were introduced by Arkin et al. in [1] who obtained the first constant factor approximation and a proof of MAX-SNP hardness in general graphs. An approximation ratio of 3 was later obtained in [39] for both problems and to 2 for tree cover in [46]. The parameterized version of the problem was addressed in [33, 45].

Theorem 5.9. *Let $0 < \varepsilon \leq 1/2$ and let g be a fixed non-negative integer. There exist algorithms, based on Algorithm 1, that compute a $1+\varepsilon$-approximation to the tour cover problem and to the tree cover problem in graphs of genus bounded by g. Their running times are $n^{O(f(\varepsilon,g))}$.*

Proof of Theorem 5.9. We show that the three conditions of Theorem 1.3 are met.

Condition 1 This condition is fulfilled by an $O(1)$ approximation algorithm, see for example [46]. Consider a graph G and any feasible solution S. Since S is connected, any vertex of G/S is at distance at most 1 of the vertex resulting from the contraction of S and so, G/S has constant diameter and therefore constant branchwidth.

Condition 2 This condition is obtained by Theorem 5.1.

Condition 3 We show how to derive a Lift procedure. Note that each vertex resulting from the contraction of an edge (or more generally a path) has a loop in the contracted graph. Since the solution for the contracted graph has to cover all the edges of the graph, this vertex has to belong to the optimal solution. Therefore it is possible to add the contracted edges to the solution while preserving connectivity. For Tree Cover, the solution in G is simply $S \cup K$, while for tour cover, some edges must be taken twice to form a tour. In either case, the weight of the solution is bounded by $w(S) + 2w(K)$.

□

5.3 Weighted Connected Vertex Cover

Definition 5.4. Weighted Connected Vertex Cover. Given a graph $G = (V, E)$ with vertex weights $w : V \to \mathbb{R}^+$, a *connected vertex cover* is a set of vertices S such that $G[S]$ is connected and such that for each $(u, v) \in E$, $u \in S$ or $v \in S$. The weighted connected vertex cover problem asks for a connected vertex cover of minimum weight.

Savage [52] gave a 2 approximation algorithm which remains the best approximation algorithm for general graphs. There are PTASs for the unweighted case in restricted classes of graphs (see [55, 17]). The weighted connected vertex cover problem is very related to the tree cover problem. The difference is that the weights are on the vertices and not the edges. Fujito shows in [27] that, whereas the tree cover problem can be approximated within a constant factor in general graphs, the weighted vertex cover problem cannot be approximated within a factor better than $\log n$ in general graphs unless $\text{NP} \subseteq \text{DTIME}[n^{O(\log\log n)}]$ and provides an $O(\log n)$ approximation algorithm for the problem. See [33, 45] for results in the parameterized case.

Theorem 5.10. *Let $0 < \varepsilon \leq 1/2$ and let g be a fixed non-negative integer. There exists an algorithm, based on Algorithm 1, that computes a $1+\varepsilon$-approximation to the vertex-weighted connected vertex cover problem in graphs of genus bounded by g. Its running time is $n^{O(f(\varepsilon,g))}$.*

Proof of Theorem 5.10. We show that the conditions of Theorem 1.3 hold.

Condition 1'. Here we use the more general version of the first condition, where the backbone is not required to be a solution: Observe that the value of any optimal solution to the weighted connected dominating set problem is a lower bound on the value of the optimal solution for the weighted connected vertex cover problem. Therefore, it is possible to compute a subgraph H of the input graph G of total weight at most $O(\mathrm{OPT})$ such that G/H has treewidth at most $O(1)$ using Lemma 5.3.

Condition 2 is ensured by Theorem 5.1.

Condition 3 is attained by letting the solution on G be $S \cup K$. Since every contracted vertex is either in S or adjacent to S, $S \cup K$ is connected. As S is a vertex cover in G/H, every edge in G has an endpoint in either S or K. $\square$

5.4 Weighted Feedback Vertex Set

There has been much research on *feedback vertex set*. Here we only mention a few representative results. The first constant-factor approximation algorithm for the *unweighted* case was achieved in [5]. It was later improved to a factor 2 for both weighted and unweighted in [3]. Primal-dual approximation algorithms for these and more general problems were given in [29] The parameterized problem was addressed in [14] and [48]. An approximation scheme for the unweighted version was given in [17].

Theorem 5.11. *There is a PTAS for* weighted feedback vertex set *in undirected planar graphs.*

We provide a reduction from weighted feedback vertex set to vertex-weighted connected dominating set.

Given a planar graph $G = (V, E)$ and a vertex-weight function $w(\cdot)$, we construct an instance for connected dominating set (CDS): the graph $\tilde{G} = (\tilde{V}, \tilde{E})$; weights of vertices of G in $\tilde{G}$ are preserved; others receive a weight of 0. It suffices to show that every FVS in G corresponds to a CDS in $\tilde{G}$ of the same weight, and vice versa.

Let S be a FVS in G. Let $V_f := \tilde{V} - V$ be the vertices in $\tilde{G}$ inside faces of G, and let $\tilde{S} := S \cup V_f$. Thus $w(S) = w(\tilde{S})$. As $\tilde{S}$ contains every vertex in V_f, $\tilde{S}$ is a FVS of $\tilde{G}$. Note that $\tilde{G}$ is triangulated, and in a triangulated planar graph, every minimal vertex cut is a simple cycle. Therefore $\tilde{S}$ hits every vertex cut in $\tilde{G}$, i.e., induces a connected graph. $\tilde{S}$ contains V_f, which dominates $\tilde{G}$. Therefore $\tilde{S}$ is a connected dominating set in $\tilde{G}$.

Now let $\tilde{S}$ be a CDS in $\tilde{G}$. Then $\tilde{S}' := S' \cup V_f$ is a CDS in $\tilde{G}$ with the same weight. Thus $\tilde{S}'$ hits every cycle in $\tilde{G}$ that strictly separates any two vertices in $\tilde{S}'$. Every cycle in G separates some two faces of G, and therefore the corresponding vertices in V_f. Thus $\tilde{S}'$ hits every cycle in G. Thus $S := \tilde{S}' \cap V = \tilde{S} \cap V$ hits every cycle in G, i.e., is a feedback vertex set, and $w(S) = w(\tilde{S})$.

6. ACKNOWLEDGEMENTS

We wish to thank Sergio Cabello, Eli Fox-Epstein, Zhentao Li, Arnaud de Mesmay, Patrice Ossona de Mendez, and Grigory Yaroslavtsev for joint discussions at various stages of this project.

7. REFERENCES

[1] E. M. Arkin, M. M. Halldórsson, and R. Hassin. Approximating the tree and tour covers of a graph. *Inf. Process. Lett.*, 47(6):275–282, 1993.

[2] S. Arora, M. Grigni, D. R. Karger, P. N. Klein, and A. Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In *ACM-SIAM Symp. on Discrete Algorithms*, pages 33–41, 1998.

[3] V. Bafna, P. Berman, and T. Fujito. Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In *Algorithms and Computations*, pages 142–151. Springer, 1995.

[4] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. of the ACM*, 41(1):153–180, 1994.

[5] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference. *SIAM J. on Computing*, 27(4):942–959, 1998.

[6] P. Berman and G. Yaroslavtsev. Primal-dual approximation algorithms for node-weighted network design in planar graphs. In *Approximation, Randomization, and Combinatorial Optimization. Alg. and Tech.*, pages 50–60. Springer, 2012.

[7] D. Binkele-Raible and H. Fernau. A faster exact algorithm for the directed maximum leaf spanning tree problem. In *Computer Science–Theory and Applications*, pages 328–339. Springer, 2010.

[8] H. L. Bodlaender. On linear time minor tests and depth first search. In *W. on Algorithms and Data Structures, WADS*, pages 577–590, 1989.

[9] H. L. Bodlaender, A. Grigoriev, and A. M. C. A. Koster. Treewidth lower bounds with brambles. *Algorithmica*, 51:81–98, 2008.

[10] H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In *Automata, Languages and Programming*, pages 627–637. Springer, 1997.

[11] S. Cabello, É. Colin de Verdière, and F. Lazarus. Algorithms for the edge-width of an embedded graph. *Computational Geometry: Theory and Applications*, 45:215–224, 2012.

[12] E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting (complicated) surfaces is hard. *Comput. Geom.*, pages 94–110, 2008.

[13] C. Chekuri, S. Khanna, and B. Shepherd. Edge-disjoint paths in planar graphs. In *Symp. on Foundations of Computer Science*, pages 71–80, 2004.

[14] J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for feedback vertex set problems. *J. of Comput. and Syst. Sci.*, 74(7):1188–1198, 2008.

[15] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *Networks*, 42(4):202–208, 2003.

[16] M. Cygan, J. Nederlof, M. Pilipczuk, M. Pilipczuk, J. M. van Rooij, and J. O. Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *Symp. on Foundations of Computer Science*, pages 150–159, 2011.

[17] E. D. Demaine and M. Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *ACM-SIAM Symp. on Discrete Algorithms*, pages 590–601, 2005.

[18] E. D. Demaine, M. T. Hajiaghayi, and P. N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. *ACM Trans. on Algorithms*, 10(3):1–20, 2014.

[19] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[20] R. J. Douglas. NP-completeness and degree restricted spanning trees. *Discrete Mathematics*, pages 41 – 47, 1992.

[21] R. G. Downey and M. R. Fellows. *Parameterized computational feasibility.* Springer, 1995.

[22] D. Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

[23] D. Eppstein. Dynamic generators of topologically embedded graphs. In *ACM-SIAM Symp. on Discrete Algorithms*, pages 599–608, 2003.

[24] J. Erickson. Graph separators, 2009. Available at `jeffe.cs.illinois.edu/teaching/comptop/2009/notes/separators.pdf`.

[25] J. Erickson and K. Whittlesey. Greedy optimal homotopy and homology generators. In *ACM-SIAM Symp. on Discrete Algorithms*, pages 1038–1046, 2005.

[26] G. Frederickson. Fast algorithms for shortest paths in planar graphs with applications. *SIAM J. on Computing*, 16:1004–1022, 1987.

[27] T. Fujito. On approximability of the independent/connected edge dominating set problems. *Inf. Process. Lett.*, 79(6):261–266, 2001.

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., 1979.

[29] M. X. Goemans and D. P. Williamson. Primal-dual approximation algorithms for feedback problems in planar graphs. *Combinatorica*, 18(1):37–59, 1998.

[30] M. Grigni, E. Koutsoupias, and C. Papadimitriou. An approximation scheme for planar graph TSP. In *Symp. on Foundations of Computer Science*, 1995.

[31] S. Guha and S. Khuller. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387, 1998.

[32] S. Guha and S. Khuller. Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Inf. Comput.*, 150(1):57–74, 1999.

[33] J. Guo, R. Niedermeier, and S. Wernicke. Parameterized complexity of vertex cover variants. *Theory of Computing Systems*, 41(3):501–520, 2007.

[34] M. R. Henzinger, Ph. N. Klein, S. Rao, and S. Subramanian. Faster shortest-path algorithms for planar graphs. *J. of Comput. and Syst. Sci.*, 55(1, part 1):3–23, 1997.

[35] R. Karp. *Complexity of Computer Comput.*, chapter Reducibility Among Combinatorial Problems. Plenum Press, 1972.

[36] P. N. Klein. A linear-time approximation scheme for planar weighted TSP. In *Symp. on Foundations of Computer Science*, pages 647–656, 2005.

[37] P. N. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM J. on Computing*, 37(6):1926–1952, 2008.

[38] P. N. Klein, S. Mozes, and C. Sommer. Structured recursive separator decompositions for planar graphs in linear time. In *ACM Symp. on Theory of Computing*, pages 505–514, 2013.

[39] J. Könemann, G. Konjevod, O. Parekh, and A. Sinha. Improved approximations for tour and tree covers. In *Approximation Algorithms for Combinatorial Optimization*, pages 184–193. Springer, 2000.

[40] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. on Applied Mathematics*, 36(2):177–189, 1979.

[41] R. J. Lipton and R. E. Tarjan. Applications of a planar separator theorem. *SIAM J. on Computing*, 9(3):615–627, 1980.

[42] D. Lokshtanov, M. Mnich, and S. Saurabh. Linear kernel for planar connected dominating set. In *Theory and Applications of Models of Comput.*, pages 281–290. Springer, 2009.

[43] H.-I. Lu and R. Ravi. Approximating maximum leaf spanning trees in almost linear time. *J. of Algorithms*, 29(1):132–141, 1998.

[44] G. L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986.

[45] D. Mölle, S. Richter, and P. Rossmanith. Enumerate and expand: Improved algorithms for connected vertex cover and tree cover. *Theory of Computing Systems*, 43(2):234–253, 2008.

[46] V. H. Nguyen. Approximation algorithms for metric tree cover and generalized tour and tree covers. *RAIRO - Operations Research*, 41(3):305–315, 2007.

[47] S. Paul and R. E. Miller. Locating faults in a systematic manner in a large heterogeneous network. In *Int. Conf. on Computer Com.*, pages 522–529, 1995.

[48] V. Raman, S. Saurabh, and C. Subramanian. Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Trans. on Algorithms*, 2(3):403–415, 2006.

[49] N. Robertson and P. Seymour. Graph minors. xi. circuits on a surface. *J. of Combinatorial Theory, Series B*, 60(1):72 – 106, 1994.

[50] N. Robertson, P. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. of Combinatorial Theory, Series B*, 62:323–348, 1994.

[51] J. Rué, I. Sau, and D. M. Thilikos. Dynamic programming for graphs on surfaces. *ACM Trans. on Algorithms*, 10(2):8, 2014.

[52] C. D. Savage. Depth-first search and the vertex cover problem. *Inf. Process. Lett.*, 14(5):233–237, 1982.

[53] P. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.

[54] J. Wu and H. Li. On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *Int. W. on Discrete Alg. and Methods for Mob. Comput. and Com.*, pages 7–14, 1999.

[55] Z. Zhang, X. Gao, and W. Wu. PTAS for connected vertex cover in unit disk graphs. *Theoretical Computer Science*, 410(52):5398–5402, 2009.