FINDING CYCLES WITH TOPOLOGICAL PROPERTIES IN EMBEDDED GRAPHS*

SERGIO CABELLO†, ÉRIC COLIN DE VERDIÈRE‡, AND FRANCIS LAZARUS§

Abstract. Let G be a graph cellularly embedded on a surface $\mathcal{S}$. We consider the problem of determining whether G contains a cycle (i.e., a closed walk without repeated vertices) of a certain topological type in $\mathcal{S}$. We show that the problem can be answered in linear time when the topological type is one of the following: contractible, non-contractible, or non-separating. In each case, we obtain the same time complexity if we require the cycle to contain a given vertex. On the other hand, we prove that the problem is NP-complete when considering separating or splitting cycles. We also show that deciding the existence of a separating or a splitting cycle of length at most k is fixed-parameter tractable with respect to k plus the genus of the surface.

Key words. topological graph theory, algorithm, graph, cellular embedding, cycle, contractible, separating

AMS subject classifications. 05C10, 05C85, 57M15, 68Q17, 68R10, 68W05

1. Introduction.

Topological graph theory studies combinatorial embeddings of graphs on surfaces. This includes the design of efficient algorithms for finding optimal cycles with prescribed topological properties. This last subject has received much attention since Thomassen's seminal work [21] (see also Mohar and Thomassen [20, Chapter 4]) to compute a shortest cycle that is *non-contractible* (cannot be deformed continuously on the surface to a point) or *non-separating* (whose removal leaves the surface connected). Since then, more efficient algorithms have been proposed, also in the context of weighted graphs [3,7,18]. As it turns out, the shortest non-contractible or non-separating closed walk is automatically a cycle; this follows from Thomassen's *3-path condition* [21], a structural property that is heavily used in the aforementioned algorithms.

However, for other topological properties, shortest closed walks are not necessarily cycles, and thus different tools are needed to compute them. For instance, Cabello [2] has given a polynomial-time algorithm to find a shortest contractible cycle in a surface-embedded graph, and has proven that it is NP-hard to compute a shortest contractible cycle through a given vertex, or a shortest separating cycle.

In many similar problems, one usually relaxes the condition that the output be a cycle—without repeated vertices—and instead only requires a shortest closed walk with the given topological property. For example, it is known how to compute in (small) polynomial time a shortest non-contractible or non-separating closed walk passing through a given vertex [13], possibly in directed graphs [4], a shortest essential closed walk [16], a closed walk in a non-trivial homotopy class that is as short as

*Research partially supported by the Slovenian Research Agency, program P1-0297 and project BI-FR/09-10-PROTEUS-014, and by the French Ministry of Foreign and European Affairs.

†Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia; sergio.cabello@fmf.uni-lj.si; http://www.fmf.uni-lj.si/~cabello/.

‡Laboratoire d'informatique, École normale supérieure, CNRS, Paris, France; Eric.Colin.de.Verdiere@ens.fr; http://www.di.ens.fr/~colin/.

§GIPSA-Lab, CNRS, Grenoble, France; Francis.Lazarus@gipsa-lab.grenoble-inp.fr; http://www.gipsa-lab.inpg.fr/~francis.lazarus/.

possible within that homotopy class [6], or a shortest closed walk homotopic to a given walk [9]. In contrast, finding a shortest *splitting* (separating but non-contractible) closed walk is NP-hard [8].[1]

In this article, we consider the problem of deciding whether there *exists* a *cycle* of a certain topological type in a given surface-embedded graph. In other words, we drop the optimization objective, but insist in having a cycle without repeated vertices. (Removing this latter constraint would make the problem trivial.) We emphasize that we consider *cellular* graph embeddings, where each face is an open disk. Nevertheless, a given edge may have the same face on both of its sides.

We exhibit a strong dichotomy in the complexity of these problems, depending on the topological property required. As it turns out, there are linear-time algorithms when the corresponding optimization problem (even for closed walks) has a polynomial-time algorithm. This is the case for contractible, non-contractible, or non-separating cycles. On the other hand, we again obtain NP-hardness results for splitting or separating cycles, as in the optimization version of these problems. For those cases, we also propose algorithms to decide the existence of a separating or splitting cycle of length at most k, whose complexities are near-linear or near-quadratic when k and the genus g of the surface are fixed (these algorithms are thus fixed-parameter tractable with respect to $k + g$). We emphasize that our arguments quite differ from the ones used in the above cited papers and are more inclined towards basic graph theory.

We now describe precisely our results. Throughout this article, $G = (V, E)$ denotes a graph cellularly embedded on a surface $\mathcal{S}$ with genus g (the surface may be orientable or not, but may not have boundary; we discuss extensions to surfaces with boundary in Section 7.1).

THEOREM 1.1. *We can determine in $O(|E|)$ time if G admits:*

1. *a contractible cycle on $\mathcal{S}$,*
2. *a contractible cycle on $\mathcal{S}$ passing through a given vertex,*
3. *a non-contractible cycle on $\mathcal{S}$ passing through a given vertex,*
4. *a non-separating cycle on $\mathcal{S}$ passing through a given vertex,*

and return one such cycle if it exists.

Note that the last two problems become rather trivial if we do not enforce the cycle to contain a given vertex. Indeed, if $\mathcal{S}$ is not a sphere, then any cycle in a cut graph (see the next section for a definition) is non-separating, hence non-contractible.

THEOREM 1.2. *Deciding the existence of a cycle in G of any of the following type is NP-complete:*

1. *separating on $\mathcal{S}$,*
2. *splitting on $\mathcal{S}$,*
3. *separating on $\mathcal{S}$ and passing through a given vertex of G,*
4. *splitting on $\mathcal{S}$ and passing through a given vertex of G.*

We mention that (1) answers negatively to an open problem raised by Mohar and Thomassen [20, Problem 4.3.3(b)]. As a side note, (1) reduces to (3) (and similarly (2) reduces to (4)) using *Cook reductions*: to solve (1), simply solve problem (3), taking each vertex of G in turn, and similarly for (2) with (4). However, NP-completeness is defined in terms of *Karp reductions*, which are more restrictive. It is not clear a priori that (1) reduces to (3) by Karp reductions, namely, whether an instance of (1)

[1] In several variants of these problems, it is required that the closed walk be "non-self-crossing", namely, that it can be slightly deformed on the surface to remove all its self-intersections. This technicality is taken care of using the notion of combinatorial or cross-metric surface [9, 10].

can be transformed to an instance of (3) such that the answer is the same on both instances. Therefore, (3) and (4) do not follow trivially from (1) and (2).

We finally propose algorithms for parameterized versions of those NP-complete problems relying on the color-coding approach of Alon et al. [1].

THEOREM 1.3. *Let $k \geq 1$ be an integer, and let s be a vertex of G. There is an algorithm that in $2^{O(g+k)}|E|\log|V|$ time decides if G has a separating, respectively splitting, cycle on $\mathcal{S}$ through s of length at most k and reports one, if one exists.*

For the same problem, there is a randomized algorithm with running time $2^{O(g+k)} \times |E|$: the only case where the algorithm errs, with probability at most $1/2$ (or any constant strictly smaller than 1), is when there exists a cycle of the desired type, but the answer of the algorithm is negative.

By running this algorithm once for every choice of s, we can of course drop the basepoint condition:

COROLLARY 1.4. *We can decide if G has a separating, respectively splitting, cycle on $\mathcal{S}$ of length at most k and report one, if one exists, in $2^{O(g+k)}|E||V|\log|V|$ time. As in Theorem 1.3, there is also a randomized algorithm, faster by a $\Theta(\log|V|)$ factor, with one-sided error allowed with constant positive probability.*

2. Background. We review some basic terminology and properties of graphs and their embedding on surfaces. We follow standard graph theory terminology, as in the book by West [22]. All the considered graphs may have loops and multiple edges. A *cycle* in a graph is a closed walk without repeated vertices. A *loop* is a closed walk with one distinguished vertex, its *basepoint*. All walks are oriented; given a walk w, we denote by w^{-1} the same walk with the opposite orientation.

2.1. Blocks. Let $G = (V, E)$ be a graph. The *blocks* of G are its subgraphs induced by the classes of the following equivalence relation on its set E of edges: $e \sim e'$ if $e = e'$ or if there is a cycle in G that contains both e and e'. The blocks of G can be determined in $O(|E|)$ time using depth-first search. (See West [22, p. 157].)

2.2. T-Loops, T-Cycles, and Cycle Group. Let T be a spanning tree in G and s be a vertex of G. To every edge e of G, we can associate the loop $\tau(T, s, e)$ composed of the path in T joining s to an endpoint of e, the edge e, and the path in T joining the other endpoint of e to s. We call $\tau(T, s, e)$ the *T-loop* associated to e; the vertex s is the *basepoint* of the T-loop.

We can also associate to e the closed walk $\tau(T, e)$ composed of e and the path in T joining the endpoints of e. If e is not an edge of T, $\tau(T, e)$ is a cycle, called the *T-cycle* associated to e.

An *even subgraph* is a subgraph of G, each vertex of which has even degree. The set of even subgraphs form an Abelian group, where the sum corresponds to the symmetric difference of the even subgraphs. This group is called the *cycle group* of G. When G is connected, it is again part of the folklore that the set of T-cycles associated to the set of *chords* $E(G) \setminus E(T)$ form a basis of the cycle group of G.

2.3. Surfaces. We only consider surfaces without boundary. A *surface* (or 2-manifold) $\mathcal{S}$ is a compact, connected, topological space where each point has a neighborhood homeomorphic to the plane. Every surface is homeomorphic to a sphere where:

- either the interiors of $2g \geq 0$ disjoint closed disks are removed and g cylinders are attached to connect the resulting circles by pairs, or
- the interiors of $g \geq 1$ disjoint closed disks are removed and a Möbius band is attached to each resulting circle.

The surface is called *orientable* in the former case and *non-orientable* in the latter case. In both cases, g is the *genus* of the surface.

2.4. Cellular Graph Embeddings. A graph G is *cellularly embedded* on a surface $\mathcal{S}$ if every open face of (the embedding of) G on $\mathcal{S}$ is a disk. As it is customary, we will assume that the input graphs are cellularly embedded. (At some intermediary steps we may have graphs that are not cellularly embedded.) Following Mohar and Thomassen [20], the embedding of G can be encoded by adjoining to the data of G a *rotation system* and a *signature*. The rotation system provides for every vertex in V a cyclic permutation of its incident edges and the signature assigns a sign to every edge to indicate whether the rotation systems of its endpoints are compatible or not. Storing a cellular embedding takes a space linear in the total number of its vertices, edges, and faces. We note that, by Euler's formula, this number is linear in the sole number of edges of G.

From a rotation system, one can deduce the *facial walks* of G, i.e., the closed walks obtained by following the boundary of an open face of G (see [20, p. 93] for a detailed description). Every face corresponds to two opposite facial walks. We will not differentiate these two opposite facial walks and will refer to *the* facial walk of a face as any one of its two facial walks.

An edge e of an embedded graph G may be incident to two distinct faces or to a single face. In the former case, e is called *regular* and *singular* in the latter. Note that a regular edge appears exactly once in each facial walk of its incident faces, while a singular edge appears twice, with or without the same orientation, in the facial walk of its incident face.

There are data structures to maintain and operate efficiently with embedded graphs, like for example the gem representation [12, 19]. With such data structures we can traverse the neighbors of a vertex in time proportional to its degree, obtain a facial walk in time proportional to its length, or cut the surface along a path or cycle in time proportional to its length.

2.5. Duality. Let G be a graph embedded on a surface $\mathcal{S}$. Its *dual graph*, denoted by G^*, has for vertices the set of faces of G and for edges the set of edges (dual to) $E(G)$: two faces are adjacent if they share an edge of G. The edge dual to e is denoted by e^*, and it connects the two faces adjacent to e in the embedding. An edge dual to a singular edge is a loop edge. For a set of edges $A \subseteq E(G)$, we use the notation $A^* = \{e^* \mid e \in A\}$.

2.6. Homotopy and Homology. Let G be a graph embedded in an ambient space X (for example, a surface). Let s be a vertex of G. Two loops in G with basepoint s are *homotopic* in X if one can be deformed continuously to the other within X, keeping the basepoint s fixed during the deformation. (The loops may have self-intersections.) The equivalence classes of homotopic loops are called *homotopy classes*, and we use $\langle\alpha\rangle$ to denote the homotopy class containing the loop α. The homotopy classes form a group, where the multiplication in the group corresponds to the concatenation of the loops. Its unit is the set of *contractible* loops, i.e., the set of loops that are homotopic to the constant loop. When the ambient space X is a surface $\mathcal{S}$ where the graph is cellularly embedded, we denote this group by $\pi_1(\mathcal{S}, s)$. Indeed, the fact that G is cellularly embedded implies that this group, called the *fundamental group* of $\mathcal{S}$, depends only on the surface $\mathcal{S}$. When we regard G as a 1-dimensional complex and take G itself as the ambient space, we obtain the fundamental group of G, denoted by $\pi_1(G, s)$. If G is connected and T is a spanning tree

of G, it is a well-known fact that the set of T-loops with basepoint s associated to the set of chords $E(G) \setminus E(T)$ form a basis of $\pi_1(G, s)$.

Let G be a graph cellularly embedded in a surface $\mathcal{S}$. The *boundary graph* of a face f of G is the even subgraph of G induced by the union of edges of the facial walk of f occurring *exactly once* in this facial walk (in other words, by the *regular* edges of G belonging to this facial walk). Two even subgraphs are said *homologous* if their sum in the cycle group of G is equal to the sum of the boundary graphs of some faces. The equivalence classes of homologous even subgraphs, called *homology classes*, form an Abelian group under the symmetric difference. Equivalently, this group, called the *homology* group, can be defined as the quotient of the cycle group of G by the subgroup of even subgraphs homologous to the empty graph. In particular, a generating family for the homology group can be obtained by taking the homology classes of a basis of the cycle group of G. It can actually be shown that the homology group depends only on the surface $\mathcal{S}$ and not on the embedded graph G; we therefore denote this homology group by $H_1(\mathcal{S})$. (This is known as $\mathbb{Z}_2$-homology in algebraic topology, but it is the only homology we will deal with.)

2.7. Topological Types of Embedded Cycles. Every loop in G without repeated vertices forms a cycle in G. It turns out that such a loop is contractible if and only if the corresponding cycle bounds a disk in $\mathcal{S}$. In this case, we say that the cycle is *contractible*. A cycle in G is *separating* if the surface is disconnected by cutting it along that cycle. It is a well-known fact that a cycle separates a surface if and only if its homology class is trivial. A cycle is *splitting* if it cuts the surface into two components, neither of which is a disk. In other words, a splitting cycle is a separating and non-contractible cycle.

2.8. Cut Graphs and Homology Bases. If H is a subgraph of G, we will denote by $\mathcal{S}\setminus\setminus H$ the topological space obtained after cutting $\mathcal{S}$ along H (this is a surface with boundary). The *dual graph* of $\mathcal{S}\setminus\setminus H$ has for vertices the set of faces of G and for edges the (dual) set of edges $E(G) \setminus E(H)$: two faces are adjacent if they share an edge that is not in H. If $\mathcal{S}\setminus\setminus H$ is a closed disk (equivalently, H is cellular and has a single face), then H is called a *cut graph*. A cut graph is *spanning* if it contains all the vertices of G. In this case, the dual graph of $\mathcal{S}\setminus\setminus H$ is a tree. A spanning cut graph can be computed in time linear in the number of edges of G [5, 12].

A homology basis of $H_1(\mathcal{S})$ can be computed as follows. Let H be a subgraph of G that is a cut graph, and let T be a spanning tree of H. The set of T-cycles associated to the set of chords $E(H) \setminus E(T)$ form a homology basis for $\mathcal{S}$. Said differently, a basis of $H_1(\mathcal{S})$ can be obtained from a homology basis of a cut graph. From Euler's formula, it is easily derived that a homology basis of $\mathcal{S}$ has $2g$ (respectively g) cycles if $\mathcal{S}$ is an orientable (respectively non-orientable) surface of genus g. A homology class can thus be represented by a vector of $O(g)$ bits, where each bit stands for the occurrence of a basis cycle in this sum [15, Section 4]. We will use $[\alpha]$ to denote the bit vector of the homology class of an even subgraph α, and use $\oplus$ to make the bitwise sum between classes. Thus, if an even subgraph β is the symmetric difference of two even subgraphs α and α', then $[\beta] = [\alpha] \oplus [\alpha']$.

Suppose H is a spanning cut graph. Let T be a spanning tree of H, hence of G. We can compute the bit vectors of the homology classes of the T-cycles associated to the edges of G as follows. The bit vector of the T-cycle associated to an edge of T is obviously the zero vector. The homology class of the T-cycle associated to an edge in $E(H) \setminus E(T)$ has one non-zero bit for this T-cycle. Now, cutting $\mathcal{S}$ along the cut graph H yields a closed disk D. Since H is spanning, every edge uv in $E(G)\setminus E(H)$ has

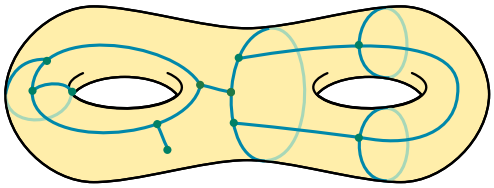

FIG. 3.1. *A cellular embedding of a graph without contractible cycle.*

its endpoints u and v on the boundary of D; therefore, the homology class of $\tau(T, uv)$ is the mod 2 sum of the bit vectors of the walk connecting u and v on the boundary of the disk D (both possible choices will give the same result). Assume one of the two pieces of D cut along $e = uv$ is a single face f of G; we may compute the bit vector of e as indicated above, by running along the boundary of f. Then we remove f and recurse on the disk $D \setminus f$. Therefore, the following lemma holds.

LEMMA 2.1 (See also [14, Lemma 3.1]). *We can compute the homology class of all the T-cycles associated to the edges of G in $O(g|E|)$ time.*

3. Contractible Cycles. In this section we prove points (1) and (2) of Theorem 1.1: we can determine in $O(|E|)$ time if G contains a contractible cycle[2]. The same is true if we impose the contractible cycle to contain a given vertex of G. Figure 3.1 shows a simple example of cellular graph embedding without contractible cycle. Recall that an edge is regular if it is incident to two distinct faces. The edges of G can be classified as regular or singular in a simple traversal of all the facial walks: edges appearing once (respectively twice) in a facial walk can be marked regular (respectively singular). This clearly takes linear time by assumption on the data structure for storing the embedded graph G.

LEMMA 3.1. *Let e be a regular edge of a face F of G. Then e belongs to a cycle of G whose edges appear in the facial walk of F. Moreover, such a cycle can be extracted in time proportional to the length of the facial walk of F.*

Proof. Consider the subgraph G_F of G induced by the edges of the facial walk of F. Since e is regular, the complementary walk of e in this facial walk does not use e. Hence, the graph $G_F - e$ is connected and we can extract from this graph a path between the endpoints of e to form a cycle with e. ∎

We denote by $c(F, e)$ the cycle extracted by the above procedure. The following lemma is a direct consequence of the Jordan curve theorem [20, p. 25].

LEMMA 3.2. *Let e be a regular edge incident to a face F. Assume that F is contained in a closed disk of $\mathcal{S}$ bounded by a cycle of G. Then, the cycle $c(F, e)$ bounds a closed disk in $\mathcal{S}$.*

Given a vertex s, we construct a set of cycles $\mathcal{C}(s)$ as follows. For every face F incident to at least one regular edge e having s as an endpoint, we add to $\mathcal{C}(s)$ the cycle $c(F, e)$. Clearly, $\mathcal{C}(s)$ can be constructed in time proportional to the complexity of G. Also, since every edge of $c(F, e)$ is incident to F, we remark that any edge of G may appear in at most two cycles in $\mathcal{C}(s)$.

The set $\mathcal{C}$ is defined similarly, dropping the condition on s: For every face F incident to at least one regular edge, we add to $\mathcal{C}$ the cycle $c(F, e)$, where e is such an edge. Again, $\mathcal{C}$ can be constructed in time proportional to the complexity of G.

[2]Note that the problem becomes trivial for a graph embedded with face-width at least two since, in this case, all the facial walks are cycles. See [20, Proposition 5.5.11].

LEMMA 3.3. *G contains a contractible cycle through s if and only if some cycle in $\mathcal{C}(s)$ is contractible. Similarly, G contains a contractible cycle if and only if some cycle in $\mathcal{C}$ is contractible.*

Proof. Since every cycle in $\mathcal{C}(s)$ contains s, the "if" condition of the first equivalence is trivial. On the other hand, suppose G has a contractible cycle c through s. Let e be an edge of c incident to s. Since c bounds some closed disk D in $\mathcal{S}$, the edge e must be regular and must have an incident face F in D. By construction, $\mathcal{C}(s)$ contains a cycle $c(F, e')$ for some regular edge e' incident to s. This cycle contains s, and, by Lemma 3.2, it is contractible. The proof for the second part of the lemma is entirely similar, dropping the condition on s and replacing $\mathcal{C}(s)$ by $\mathcal{C}$. □

LEMMA 3.4. *$\mathcal{C}(s)$ contains a contractible cycle if and only if there is a closed disk in $\mathcal{S}$ whose boundary is a cycle of $\mathcal{C}(s)$ and whose interior is disjoint from the cycles in $\mathcal{C}(s)$. The same is true if we replace everywhere $\mathcal{C}(s)$ by $\mathcal{C}$.*

Proof. The "if" part is obvious. For the reverse implication, consider a contractible cycle $c(F, e) \in \mathcal{C}(s)$. It bounds a closed disk D on $\mathcal{S}$. We choose this cycle so as to minimize the number of faces of G in D. Consider another cycle $c(F', e') \in \mathcal{C}(s)$. We claim that $c(F', e')$ does not cross the interior of D. Indeed, suppose for the sake of contradiction that an edge a of $c(F', e')$ is interior to D. Then the faces incident to a, one of which is F', must be contained in D. So $c(F', e')$ would also be contained in D. By Lemma 3.2, this would be in contradiction with the minimality of D. A formal substitution of $\mathcal{C}$ for $\mathcal{C}(s)$ proves the second part of the lemma. □

Proof of points (1) and (2) of Theorem 1.1. We prove (2). Again, a proof of (1) can be obtained by a formal substitution of $\mathcal{C}$ for $\mathcal{C}(s)$.

By Lemma 3.3, it suffices to test if one of the cycles in $\mathcal{C}(s)$ is contractible. By Lemma 3.4, this happens if and only if one component of the surface $\mathcal{S}$ cut through $\cup\mathcal{C}(s)$—the set of edges in at least one cycle in $\mathcal{C}(s)$—is a closed disk whose boundary is a cycle of $\mathcal{C}(s)$. This can be checked in linear time as follows. First label each edge of G with the cycles of $\mathcal{C}(s)$ that contain this edge. As remarked above, an edge can get at most two labels. Cutting $\mathcal{S}$ through the edges of the cycles in $\mathcal{C}(s)$ takes linear time and we can extract the components that are disks by looking at their Euler characteristic. For each disk component, we can easily check in constant time per edge if all the boundary edges share a same label, i.e., if this component is bounded by a cycle in $\mathcal{C}(s)$. □

4. Non-Contractible and Non-Separating Cycles. In this section we prove points (3) and (4) of Theorem 1.1: we can determine in linear time if G contains a non-contractible cycle or a non-separating cycle through a given vertex s.

Let T be a spanning tree of G. Denote by C^* the subgraph of the dual graph G^* with the same vertex set as G^* and edge set $E(G^*) \setminus E(T)^*$. The following lemma appears in [5, Corollary 2].

LEMMA 4.1. *Let $e \in E(G) \setminus E(T)$. The T-cycle $\tau(T, e)$ is separating on $\mathcal{S}$ if and only if $C^* - e^*$ is not connected. The T-cycle $\tau(T, e)$ is contractible if and only if $C^* - e^*$ has a connected component that is a tree (possibly reduced to a single vertex).*

Proof of point (3) in Theorem 1.1. Remark that, by definition of a block, any cycle in G through the given vertex s is contained in a single block of G. We can thus restrict the search of a non-contractible cycle to the union of the blocks of G incident to s. Call H this union. Next we will see that the following two statements are equivalent:

- there exists a non-contractible cycle through s in H;
- there exists a non-contractible cycle in H.

Indeed, suppose γ is a non-contractible cycle in H that does not contain s. We exhibit a non-contractible cycle through s in H. As remarked above, γ is contained in a single block $B \subseteq H$. Still by definition of a block, there exists a cycle $c \in B$ through s and some edge of γ. Let p be the subpath of c between s and the first encountered vertex x of c in γ. Similarly, let q be the subpath of c^{-1} between s and the first encountered vertex y of c^{-1} in γ. The vertices x and y cut γ into two paths α and β. The two cycles $p \cdot \alpha \cdot q^{-1}$ and $p \cdot \beta \cdot q^{-1}$ contain s and one of them must be non-contractible, since otherwise $\gamma = \beta \cdot \alpha^{-1}$ would also be contractible.

In order to test if H has a non-contractible cycle, we compute a spanning tree T of G *that extends* a spanning tree of H. Since the fundamental group $\pi_1(H, s)$ is generated by the loops $\tau(T, s, e)$, for $e \in H \setminus T$, the graph H contains a cycle that is non-contractible in $\mathcal{S}$ if and only if one of these T-loops is non-contractible in $\mathcal{S}$. Equivalently, one of the corresponding T-cycles should be non-contractible. From Lemma 4.1, $\tau(T, e)$ is contractible if and only if $C^* - e^*$ has a connected component that is a tree. The dual edges e^* satisfying this condition are exactly those that are removed when "pruning" the graph C^*, by iteratively removing degree-one vertices with their incident edge. Therefore, we can test in linear time whether there is an edge $e \in H \setminus T$ satisfying this condition. □

Proof of point (4) in Theorem 1.1. Our proof starts literally as the proof of point (3) in Theorem 1.1, replacing non-contractible with non-separating. In particular, there exists a non-separating cycle through s in G if and only if there exists a non-separating cycle in H, the union of blocks incident to s. In order to test this last condition, we first compute a spanning tree T of G *that extends* a spanning tree of H. As recalled in Section 2, the T-cycles associated to the set of chords of T in H form a basis of the cycle space of H. Hence, H has a non-separating cycle if an only if one of these chords has an associated T-cycle that is non-zero-homologous, i.e., non-separating. From Lemma 4.1, this holds if and only if the corresponding dual edge does not separate C^*, i.e., is not a bridge in C^*. This can be tested for all the chordal edges in linear time by first marking the bridges of C^*. Recall that the bridges of a graph are its one-edge blocks and can thus be determined in linear time. □

5. Separating and Splitting Cycles. In this section we show Theorem 1.2: It is NP-hard to decide if G contains separating and splitting cycles. Our NP-hardness proof is inspired by a former article [8], but is more complicated. It proceeds by reduction from the following NP-complete problem: determine whether a given planar bipartite graph H with maximum degree 3 has a Hamiltonian cycle [17, Lemma 2.1]. (Actually, we will not use the fact that H is bipartite.) See Figure 5.1 for an overview of the reduction.

Let s be an arbitrary vertex of H of degree 3. In H, we replace s with a triangle, as shown in Figure 5.2(a-b), obtaining a graph H_1. Let one of the three new edges be called e. We mark all vertices of H_1 except the three new vertices as *required*. The following lemma is easy.

LEMMA 5.1. *H has a Hamiltonian cycle if and only if H_1 has a cycle using e and all required vertices.*

It is convenient, at this point, to fix an embedding of H_1 on the sphere. Note that e has two different incident faces in H_1. We color one of them in black and the other one in white. We surround every required vertex of H_1 with a ring, as shown in Figure 5.3. This creates two or three new faces per required vertex of H_1; we color exactly one of them (chosen arbitrarily) in black and another one in white; the last one, if present, is not colored. Label each of the k uncolored faces with distinct

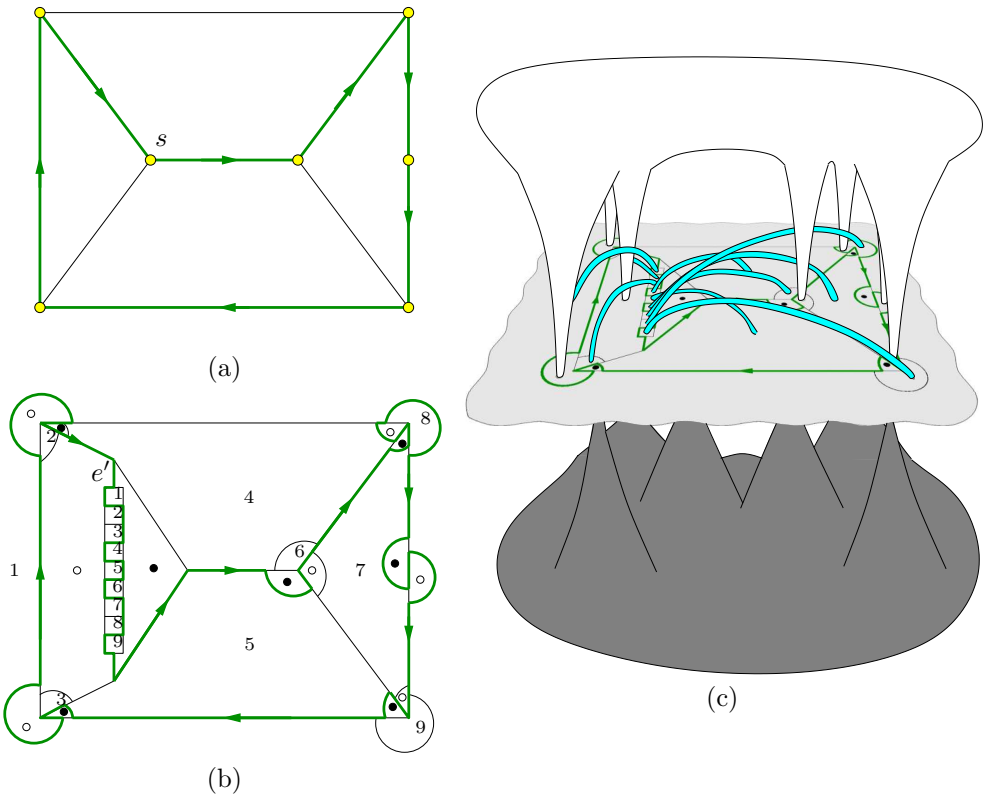

FIG. 5.1. *Overview of the reduction from Hamiltonian cycle in planar graphs with maximum degree 3. (a) An original instance H with a solution. (b) The corresponding graph H_2. The disks inside the faces indicate their color. (c) A part of the corresponding surface (only a part of the middle gray area is shown; it was initially a sphere).*

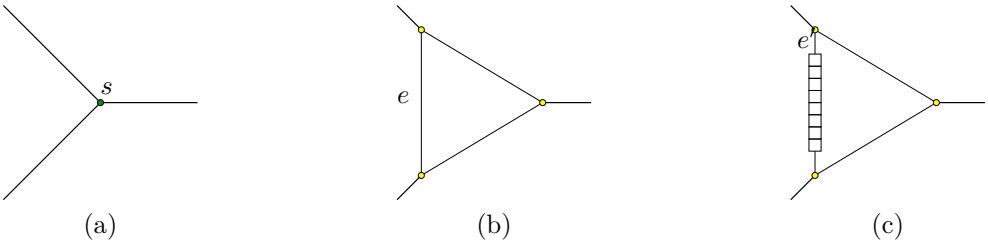

FIG. 5.2. *(a) A degree-3 vertex s of H. (b) Replacement of s by a triangle to obtain H_1. (c) Insertion of the grid on edge e to obtain H_2.*

integers between 1 and k. Split e into three subedges; call one of the extremal subedges e'; replace the middle subedge with a $((k+1) \times 2)$-grid, as shown in Figure 5.2(c), creating k *grid faces*; these grid faces are also labeled with distinct integers between 1 and k. We have obtained a new graph H_2 with a planar embedding, where every face got either a color (black or white) or a label between 1 and k. Moreover, every label is represented by exactly one grid face and exactly one non-grid face.

Now we build the surface $\mathcal{S}$; see Figure 5.1. First, we remove a disk from every

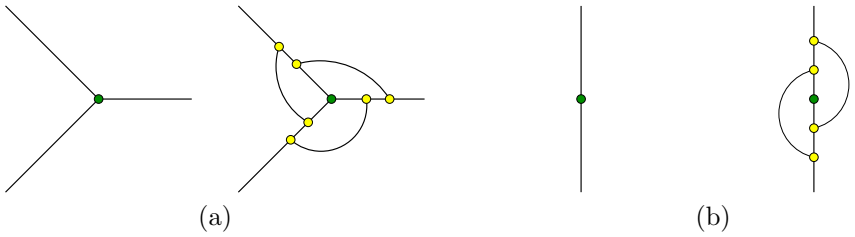

FIG. 5.3. *Creation of the rings in H_1: (a) for a degree-3 vertex; (b) for a degree-2 vertex. (We may clearly assume that H has no vertex of degree one.)*

labeled face, and attach k cylinders to these $2k$ punctures to connect the pairs of faces with corresponding labels. Second, we remove disks from every white face, and we attach a single sphere with boundaries to them. We similarly attach another sphere with boundaries to the black faces.

LEMMA 5.2. *H_1 has a cycle using e and all required vertices if and only if H_2 contains a separating (or splitting) cycle in $\mathcal{S}$.*

Proof. Note that a cycle γ in H_2 separates $\mathcal{S}$ if and only if, when we consider γ in the planar embedding H_2:

- the black faces are on the same side of γ,
- the white faces are on the same side of γ, and
- for each label, the two faces with this label are on the same side of γ.

If H_1 has a cycle using e and all required vertices, assume without loss of generality that it uses e by leaving the black face incident with e to its left. We transform it to a cycle in H_2 as follows: within each ring, modify the cycle so as it still passes through each central vertex at most once, and leaves the black face of the ring to the left and the white face of the ring to the right (this is always possible). Within the grid, modify the cycle so that it leaves a grid face with label i to its left if and only if it leaves the non-grid face labeled i to its left. This yields a separating (and even splitting) cycle.

Conversely, consider a separating cycle γ on H_2. It must use edge e': otherwise it uses (1) only grid edges, in which case only grid faces (at least one) are on one of its sides, or (2) only non-grid edges, in which case the black and white faces incident with e' and all grid faces are on the same side of γ, though all faces cannot be on this side. In both cases it contradicts the fact that γ is separating. Since γ separates the black faces from the white faces, γ must use a part of all rings of required vertices of H_1. Since every ring is separated from the rest of H_2 by three edges, it is used at most once. Finally, this yields a cycle in H_1 using e and all required vertices. □

H_2 is not cellular on $\mathcal{S}$. We now augment it to a graph H_3 that is cellular on $\mathcal{S}$ as follows. Every face f in H_2 that is not cellular is a punctured sphere. Put a new vertex inside f, and connect it with one vertex per boundary component of f (Figure 5.4).

LEMMA 5.3. *Any separating (or, in particular, splitting) cycle in H_3 belongs to H_2.*

Proof. Let a be an edge added to H_2 to form H_3; that edge a has the same face of H_3 to its left and to its right, and therefore there is a simple closed curve γ_a on the surface that crosses a exactly once and crosses no other edge of H_3. A separating cycle crosses any closed curve on the surface an even number of times; hence γ_a cannot be crossed by a separating cycle; consequently, a cannot be used by a separating cycle.

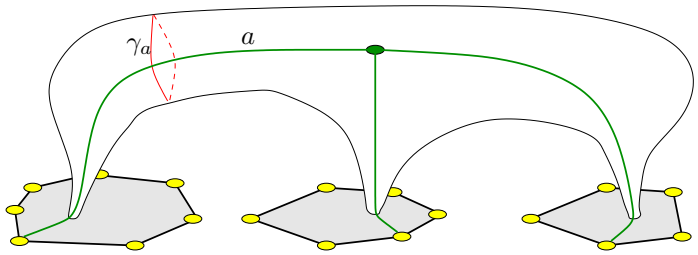

FIG. 5.4. *Extension of H_2 to a cellular graph H_3.*

□

Proof of Theorem 1.2. These problems are clearly in NP. Statements (1) and (2) follow directly from Lemmas 5.1, 5.2, and 5.3. Furthermore, every separating cycle in H_3 uses edge e' and its incident vertices; this proves (3) and (4). □

6. Computing Separating and Splitting Cycles Anyway. We now discuss a parameterized version of the last NP-hard problems. Given an integer k, decide whether G contains a separating, respectively splitting, cycle of length at most k—and report such a cycle in case of positive answer. These problems again admit two variants depending on whether or not we force the cycle to pass through a given vertex. Using the color-coding approach of Alon et al. [1], we propose randomized algorithms for these problems. Henceforth, T will designate a spanning tree of the graph G. In order to test if a cycle is separating, we shall use the equivalence with zero-homologous cycles. To this end, we precompute the $O(g)$-bit vectors of the T-cycles associated to the edges of G. By Lemma 2.1, this takes $O(g|E|)$ time.

6.1. Separating Cycles. Choose a random k-coloring $\kappa : V \to \{1, \ldots, k\}$ of the vertices of G. Hence, each vertex gets a color independently drawn from a bag of k colors, where each color has probability $1/k$ of occurrence. Suppose G has a separating cycle of length at most k through a given vertex s. With probability at least $k!/k^k = 2^{-\Theta(k)}$, the vertices of that cycle get different colors. More generally, a path or cycle in G is said *colorful* if all its vertices get a different color.

Following Alon et al. [1] we use a dynamic programming approach to search for a colorful separating cycle. For this, we consider the following directed graph $\mathcal{H}$ with arcs labelled by edges of G. We refer to the nodes and arcs of $\mathcal{H}$ in order to avoid confusion with the vertices and edges of G. The graph $\mathcal{H}$ has nodes of the form (v, c, h) where $v \in V$ is a vertex of G, $c \subseteq \{1, \ldots, k\}$ is a subset of colors, and $h \in H_1(\mathcal{S})$ is a homology class. Two nodes (v, c, h) and (v', c', h') are linked by an arc labelled with edge e if and only if the following conditions are satisfied:

C1. the endpoints of e are v and v',
C2. $\kappa(v') \notin c$ and $c' = c \cup \{\kappa(v')\}$, and
C3. $h' = h \oplus [\tau(T, e)]$. (Recall that T is an arbitrary spanning tree of G, chosen at the beginning of the algorithm, and that $[\tau(T, e)]$ is the homology class in $H_1(\mathcal{S})$ of the T-cycle $\tau(T, e)$.)

The graph $\mathcal{H}$ has $2^k 2^{O(g)}|V|$ nodes since a homology class is represented by an $O(g)$-bit vector. The number of arcs of $\mathcal{H}$ is at most $2^k 2^{O(g)}|E|$ since, for c and h fixed, the total number of arcs outgoing from the set of nodes $\{(v, c, h)\}_{v \in V}$ is at most $\sum_{v \in V}(\text{degree of } v \text{ in } G)$.

LEMMA 6.1. *Let ℓ be an integer, $1 \leq \ell \leq k$. There is a separating colorful cycle*

in G through s that has length ℓ if and only if there is a directed path of length ℓ in $\mathcal{H}$ from $(s, \emptyset, 0)$ to a node $(s, c, 0)$ for some $c \subseteq \{1, \dots, k\}$.

Proof. If there is a directed path in $\mathcal{H}$ from $(s, \emptyset, 0)$ to a node (s, c, h), then, projecting onto the first coordinate of the nodes, we obtain a loop w in G with basepoint s, of the same length. Furthermore, the homology class $[w]$ is precisely h. By the way how we defined arcs in $\mathcal{H}$, all the vertices of w have different colors, so w is actually a cycle. This proves the "if" part. The converse is shown analogously: every colorful path cycle in G "lifts" to a path of the same length in $\mathcal{H}$. ∎

LEMMA 6.2. *Given a k-coloring of G, we can decide if G contains a colorful separating cycle through s of length at most k and report one, if one exists, in $2^{O(g+k)}|E|$ time.*

Proof. We use the above color-coding schema. We thus have to traverse $\mathcal{H}$ from the node $(s, \emptyset, 0)$ and test the conditions of Lemma 6.1. Exploring $\mathcal{H}$ from a node (v, c, h) takes $O(k + g)$ time per incident outgoing arc. Indeed, for an edge e with endpoints v and v' we have to check that $\kappa(v') \notin c$ and to compute the homology class $h \oplus [\tau(T, e)]$. (Recall that the bit vectors $[\tau(T, e)]$ have been precomputed.) Traversing $\mathcal{H}$ from $(s, \emptyset, 0)$ thus takes overall $O(\sum_{v,c,h} (k + g)d(v)) = 2^{O(g)}2^{O(k)}|E|$ time.

Note that, for any traversed node (v, c, h), the concatenation of the arc labels on its search path is a c-colored path p in G such that, if $T(v, s)$ denotes the path in T from v to s, we have $[p \cdot T(v, s)] = h$. This allows to backtrack a separating cycle of length $|c|$ in case of success of the previous test. ∎

The *homology cover* used very recently by Erickson and Nayyeri in [14] leads to an alternative to the above construction of $\mathcal{H}$. Indeed, G has a separating cycle through s if and only if the homology cover $\mathcal{S}_H$ of $\mathcal{S}$ has a cycle through a lift of s that projects to a cycle in G. The lift G_H of G in the cover has $2^{O(g)}|E|$ edges. Therefore, a simple application of the color-coding approach of Alon et al. to G_H would also lead to an algorithm of complexity $2^{O(k)}2^{O(g)}|E|$ (see [14, Section 4]).

6.2. Splitting Cycles. Our method to search for a splitting cycle through a given vertex s uses basically the same coloring schema as for a separating cycle. This time, however, we also need to check that the separating cycle is non-contractible. For this, we consider the graph $\mathcal{H}'$ with nodes of the type (v, c, h, α), where $v \in V$, $c \subseteq \{1, \dots, k\}$, $h \in H_1(\mathcal{S})$ as before, and α is a homotopy class in $\pi_1(\mathcal{S}, s)$. Two nodes (v, c, h, α) and (v', c', h', α') are linked by an arc labelled with edge e if the four conditions below hold.

C'1. the endpoints of e are v and v',
C'2. $\kappa(v') \notin c$ and $c' = c \cup \{\kappa(v')\}$,
C'3. $h' = h \oplus [\tau(T, e)]$, and
C'4. $\alpha' = \alpha \cdot \langle \tau(T, s, e) \rangle$, where $\langle \tau(T, s, e) \rangle$ is the homotopy class of the T-loop $\tau(T, s, e)$ oriented so as to traverse e from v to v'.

We then have the following analog of Lemma 6.1.

LEMMA 6.3. *Let ℓ be an integer, $1 \le \ell \le k$. There is a splitting colorful cycle in G through s that has length ℓ if and only if there is a directed path of length ℓ in $\mathcal{H}'$ from $(s, \emptyset, 0, 1)$ to a node $(s, c, 0, \alpha)$ for some $c \subseteq \{1, \dots, k\}$ and some non-trivial homotopy class α. (Here, 1 denotes the homotopy class of the constant loop.)*

As opposed to homology classes, there are usually an infinite number of homotopy classes. As a consequence, we cannot just traverse $\mathcal{H}'$ as we did with $\mathcal{H}$ for separating cycles. We circumvent this difficulty with the following simple observation. Suppose that there are two colorful paths from s to v that use the same subset of colors, are homologous, but are not homotopic. If there is a colorful separating cycle that

extends one of these paths, then there is also a splitting cycle that extends one of them. Indeed, replacing in any cycle one path by the other does not change the homology class, but does change the homotopy class. This leads to the following algorithm.

We *partially* traverse $\mathcal{H}'$ from $(s, \emptyset, 0, 1)$. To explore $\mathcal{H}'$ from a node (v, c, h, α) we inspect every edge e incident to v and create a new node (v', c', h', α') if and only if the following conditions hold: this node was not created before, the above conditions C'1-C'4 are verified, and there is at most one λ such that the node (v', c', h', λ) was already created. This last condition can be checked using a counting table with one entry per triple of the form (v', c', h'). We use an implicit trivial encoding of the homotopy class: the homotopy class α' in the node (v', c', h', α') is represented by the sequence of arc labels on the traversal path from $(s, \emptyset, 0, 1)$ to (v', c', h', α'). This indeed gives a path p in G such that $\alpha' = \langle p \cdot T(v, s) \rangle$, where as before $T(v, s)$ denotes the path in T from v to s. The path p can be backtracked when needed in $O(k)$ time. In order to perform the homotopy test between two classes α' and λ represented by the two paths p and q respectively, we can test if the loop $p \cdot q^{-1}$ is contractible using the contractibility test of Dey and Guha [11] in $O(k)$ time (after $O(|E|)$ time preprocessing). It follows that the cost for traversing an arc of $\mathcal{H}'$ and visiting a new node or performing the test of Lemma 6.3 is $O(g + k)$.

LEMMA 6.4. *Given a k-coloring of G, the above algorithm decides if G has a colorful splitting cycle through s of length at most k and report one, if one exists, in $2^{O(g+k)}|E|$ time.*

Proof. The partial traversal of $\mathcal{H}'$ in the algorithm visits a subgraph $\mathcal{H}''$ that is at most twice as big as $\mathcal{H}$. The fact that we can replace $\mathcal{H}'$ by $\mathcal{H}''$ in Lemma 6.3 follows from the above observation. The rest of the analysis is identical to the separating case as in Lemma 6.2. □

6.3. Proof of Theorem 1.3. Suppose G has a separating or a splitting cycle γ of length at most k. The cycle γ is colorful with probability at least $2^{-\Theta(k)}$. Thus, if we draw $2^{O(k)}$ independent random k-colorings of the vertex set of G, then with probability at least $1/2$ (or any arbitrary constant strictly smaller than 1), at least one of these k-colorings will make γ colorful. Lemmas 6.2 and 6.4 thus lead to algorithms with $2^{O(g+k)}|E|$ expected running time for finding γ. This provides a Monte Carlo linear time algorithm, with fixed parameter $k + g$, to decide if G contains such a cycle (and to find one in the affirmative).

In their color-coding article [1], Alon et al. also show that they can compute a family of size $2^{O(k)} \log |V|$ of k-colorings with the property that every subset of k vertices is colorful for at least one of these colorings. In conjunction with the lemmas, this directly gives deterministic algorithms adding an extra $\log |V|$ factor to the complexity. □

7. Concluding Remarks.

7.1. Surfaces with Boundary. We briefly indicate how to extend our linear-time algorithms to surfaces with boundary. So let $\mathcal{S}$ be a surface with boundary, and let G be a graph cellularly embedded on it. We extend $\mathcal{S}$ and G to a surface $\bar{\mathcal{S}}$ and a graph $\bar{G}$ such that $\bar{G}$ has a contractible cycle on $\bar{\mathcal{S}}$ if and only if G has a contractible cycle on $\mathcal{S}$, and similarly for the other topological types.

- For the separating and non-separating cases, we can just attach a disk to each boundary, since this does not change whether a closed walk is separating or not.

- For the contractible case, we attach a once-punctured torus to every boundary component, and add two loop edges per such torus to make the graph cellular. Every cycle using a loop edge is non-contractible, and every other cycle in G is contractible in $\mathcal{S}$ if and only if it is contractible in $\bar{\mathcal{S}}$.
- For the non-contractible case, the only interesting case is when we require the cycle to pass through a given vertex s. We again attach a once-punctured torus to every boundary component; within each such torus, we put a new vertex v, connect it to a vertex of G, and add two loop edges based at v to transform the face of that once-punctured torus into an open disk. Since $v \neq s$, no cycle through s uses the new edges, so there is a cycle through s in G that is non-contractible on $\mathcal{S}$ if and only if there is a cycle through s in $\bar{G}$ that is non-contractible on $\bar{\mathcal{S}}$.
- For the splitting case, we consider a cycle to be splitting if it separates $\mathcal{S}$ into two non-zero genus subsurfaces, possibly with boundary. We can proceed as in the separating case by attaching a disk to each boundary. This does not change the property of being separating and preserves the genus of subsurfaces. Note that a splitting cycle in $\bar{\mathcal{S}}$ must cut $\bar{\mathcal{S}}$ into non-zero genus subsurfaces.

7.2. Shortest Closed Walks. When G contains a separating, respectively splitting, cycle, we can compute a shortest cycle of the corresponding type in $2^{O(g+\ell)}|E| \times |V| \log |V|$ time, where ℓ is the length of this shortest cycle. Indeed, we can apply Corollary 1.4 with $k = 1, 2, 3, \ldots$ until the algorithm reports the existence of a cycle, which obviously happens for $k = \ell$. The total cost is

$$\sum_{k=1}^{\ell} 2^{O(g+k)}|E||V| \log |V| = 2^{O(g+\ell)}|E||V| \log |V|.$$

Chambers et al. [8] have presented an algorithm with complexity $g^{O(g)}|E| \log |V|$ for computing a shortest splitting *closed walk* on G (the closed walk may have repeated vertices, but it should be non-self-crossing). In particular, if G has *no* splitting cycle, then the output of their algorithm will be a closed walk that is not a cycle. The problem tackled by Chambers et al. is thus different from the problem treated here. This difference suggests the following more general question: Given a closed walk in G, decide if there is a cycle in G of the same topological type, say in the same homotopy or homology class, and report one if it exists. Chambers et al. were also able to compute a shortest splitting closed walk that cuts the surface into two subsurfaces with prescribed topology [8, Section 6], i.e., fixing their genera and number of boundary components. It is not clear whether our present color coding approach can be extended to handle this case.

REFERENCES

[1] Noga Alon, Raphael Yuster, and Uri Zwick, *Color coding*, Journal of the ACM, 42 (1995), pp. 844–856.

[2] Sergio Cabello, *Finding shortest contractible and shortest separating cycles in embedded graphs*, ACM Transactions on Algorithms, 6 (2010), pp. 1–18.

[3] Sergio Cabello and Erin W. Chambers, *Multiple source shortest paths in a genus g graph*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2007, pp. 89–97.

[4] Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus, *Finding shortest non-trivial cycles in directed graphs on surfaces*, in Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG), 2010, pp. 156–165.

[5] ———, *Output-sensitive algorithm for the edge-width of an embedded graph*, in Proceedings of the 26th Annual ACM Symposium on Computational Geometry (SOCG), 2010, pp. 147–155.

[6] Sergio Cabello, Matt DeVos, Jeff Erickson, and Bojan Mohar, *Finding one tight cycle*, ACM Transactions on Algorithms, 6 (2010), p. Article 61.

[7] Sergio Cabello and Bojan Mohar, *Finding shortest non-separating and non-contractible cycles for topologically embedded graphs*, Discrete & Computational Geometry, 37 (2007), pp. 213–235.

[8] Erin W. Chambers, Éric Colin de Verdière, Jeff Erickson, Francis Lazarus, and Kim Whittlesey, *Splitting (complicated) surfaces is hard*, Computational Geometry: Theory and Applications, 41 (2008), pp. 94–110.

[9] Éric Colin de Verdière and Jeff Erickson, *Tightening nonsimple paths and cycles on surfaces*, SIAM Journal on Computing, 39 (2010), pp. 3784–3813.

[10] Éric Colin de Verdière and Francis Lazarus, *Optimal system of loops on an orientable surface*, Discrete & Computational Geometry, 33 (2005), pp. 507–534.

[11] Tamal K. Dey and Sumanta Guha, *Transforming curves on surfaces*, Journal of Computer and System Sciences, 58 (1999), pp. 297–325.

[12] David Eppstein, *Dynamic generators of topologically embedded graphs*, in Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2003, pp. 599–608.

[13] Jeff Erickson and Sariel Har-Peled, *Optimally cutting a surface into a disk*, Discrete & Computational Geometry, 31 (2004), pp. 37–59.

[14] Jeff Erickson and Amir Nayyeri, *Minimum cuts and shortest non-separating cycles via homology covers*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2011, pp. 1166–1176.

[15] Jeff Erickson and Kim Whittlesey, *Greedy optimal homotopy and homology generators*, in Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2005, pp. 1038–1046.

[16] Jeff Erickson and Pratik Worah, *Computing the shortest essential cycle*, Discrete & Computational Geometry, 44 (2010), pp. 912–930.

[17] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter, *Hamilton paths in grid graphs*, SIAM Journal on Computing, 11 (1982), pp. 676–686.

[18] Martin Kutz, *Computing shortest non-trivial cycles on orientable surfaces of bounded genus in almost linear time*, in Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SOCG), 2006, pp. 430–438.

[19] Sóstenes Lins, *Graph-encoded maps*, Journal of Combinatorial Theory, Series B, 32 (1982), pp. 171–181.

[20] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces*, John Hopkins University Press, 2001.

[21] Carsten Thomassen, *Embeddings of graphs with no short noncontractible cycles*, Journal of Combinatorial Theory, Series B, 48 (1990), pp. 155–177.

[22] Douglas B. West, *Introduction to Graph Theory*, Prentice Hall, second ed., 2001.