Computation of Cycle Bases in Surface Embedded Graphs

Kyle Fox
University of Texas at Dallas, Richardson, TX, USA

Thomas Stanley
Unaffiliated, Dallas, TX, USA

Abstract

We present an $O(n^3g^2 \log g + m) + \tilde{O}(n^{\omega+1})$ time deterministic algorithm to find the minimum cycle basis of a directed graph embedded on an orientable surface of genus g. This result improves upon the previous fastest known running time of $O(m^3n + m^2n^2 \log n)$ applicable to general directed graphs.

While an $O(n^\omega + 2^{2g}n^2 + m)$ time deterministic algorithm was known for *undirected graphs*, the use of the underlying field $\mathbb{Q}$ in the directed case (as opposed to $\mathbb{Z}_2$ for the undirected case) presents extra challenges. It turns out that some of our new observations are useful for both variants of the problem, so we present an $O(n^\omega + n^2g^2 \log g + m)$ time deterministic algorithm for undirected graphs as well.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Computational geometry; Mathematics of computing → Graphs and surfaces

Keywords and phrases cycle basis, surface embedded graphs, homology

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2022.13

Funding *Kyle Fox*: Research partially supported by NSF grant CCF-1942597.

Acknowledgements Research by T. Stanley was partially performed while this author was a student at the University of Texas at Dallas.

1 Introduction

For a given connected undirected graph $G = (V, E)$, let $m = |E|$ and $n = |V|$ be the number of edges and vertices. We define a **cycle** to be a subset of the edges such that each vertex is incident to an even number of edges in the subset. It is known that cycles constitute a vector space with addition defined as symmetric difference of the edges and that this vector space is isomorphic to $\mathbb{Z}_2^{m-n+1}$.

We can form a vector space over cycles in directed graphs as well but we require some more complicated definitions. To this end, we represent a directed graph using its underlying *undirected* graph $G = (V, E)$ along with a mapping between each edge $e = uv$ and its two underlying **darts** $u \rightarrow v$ and $rev(u \rightarrow v) = v \rightarrow u$. One of these two darts is designated as the original/**canonical orientation** $\vec{e}$ of e. A **cycle** is a function $C : e \rightarrow \mathbb{Q}$ subject to certain restrictions. Informally, we could say the "amount" of cycle (read flow) entering each vertex is equal to the amount leaving. Formally, for each vertex v, we require $\sum_{e:\vec{e}=u\rightarrow v} C(e) = \sum_{e:\vec{e}=v\rightarrow w} C(e)$. Note that in general, cycles will have negative assignments to some edges. In other words, a cycle is allowed to travel "backwards" relative to the canonical orientation of an edge. Addition of cycles is defined to be an element-wise sum over the edges, and multiplication by a scalar $q \in \mathbb{Q}$ is element-wise multiplication by q over the edges. The cycles again form a vector space known to be isomorphic to $\mathbb{Q}^{m-n+1}$.

© Kyle Fox and Thomas Stanley;
licensed under Creative Commons License CC-BY 4.0

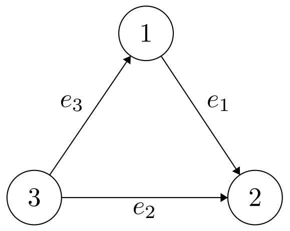

Figure 1 A small directed graph. Cycle sequence $\langle \vec{e_1}, rev(\vec{e_2}), \vec{e_3}\rangle$ corresponds to a cycle assigning 1 to edges e_1 and e_3 and -1 to edge e_2. The reversal of the sequence corresponds to a cycle assigning -1 to edges e_1 and e_3 and 1 to edge e_2.

We define a **cycle sequence** of G to be a sequence of darts $S = \langle v_0{\to}v_1, v_1{\to}v_2, \ldots, v_{k-1}{\to}v_k\rangle$ such that $v_0 = v_k$. Cycle sequence S corresponds to a cycle C_S where $C_S(e)$ is equal to the number of times $\vec{e}$ appears in S minus the number of times $rev(\vec{e})$ appears in S. We define the reversal of S as $rev(S) = \langle v_k{\to}v_{k-1}, v_{k-1}{\to}v_{k-2}, \ldots, v_1{\to}v_0\rangle$. Observe that $C_{rev(S)} = -C_S$. See Figure 1.

In both cases of an undirected or directed graph G, a **cycle basis** is a set of $d := m-n+1$ independent cycles. It is well known that a cycle basis of G can be obtained from the fundamental cycles of any spanning tree of G. Given an assignment of non-negative weights $w : E \to \mathbb{Q}_{\geq 0}$ to the edges, we define the **weight** of a cycle C as $\sum_{e \in C} w(e)$ if G is undirected or $\sum_{e \in E} |C(e)|w(e)$ if G is directed. Note that the canonical orientations of the edges is irrelevant when computing the weight of a cycle. The **weight** of a cycle basis is defined as the sum of its constituent cycles' weights.

In undirected graphs, a **minimum cycle basis** is a cycle basis of minimum weight. Because we can always reduce the weight of a directed graph cycle by dividing it by a sufficiently large scalar, we define the **minimum cycle basis** of a directed graph as a minimum weight cycle basis in which every member corresponds to a cycle sequence (equivalently, every member has only integral assignments to the edges). We emphasize that the set of integral cycle bases in a directed graph and its underlying undirected graph may not be same. A counter example is given in Figure 2 by Hariharan, Kavitha, and Mehlhorn [12]. In this figure we look at three cycle sequences $\langle e_1, e_2, e_3, e_4\rangle$, $\langle e_1, rev(e_5), rev(e_3), e_6\rangle$, and $\langle e_2, e_6, rev(e_4), e_5\rangle$. In the directed graph, the corresponding cycles are linearly independent. However, in the underlying undirected graph, the sum of two of these cycles equals the third, implying they are linearly dependent.

We do note the the problem of minimum directed cycle basis does have an alternative definition where cycles in the basis are required to follow edges in the correct direction. Using this variation the graph does not necessarily contain a cycle basis. This variation is not addressed in this paper. The minimum cycle basis has applications in many fields for both the directed [6,10] and undirected [5,16,19] cases.

From the definition of independence in vector spaces, sets of independent cycles form a matroid. Therefore, one can use the standard greedy algorithm of sorting and eliminating to find the minimum cycle basis. However, the number of cycles in G is exponential in the undirected case and infinite in the directed case. Horton reduced the search space for the greedy algorithm to $O(mn)$ cycles by showing that every cycle in the minimum cycle basis must be a fundamental cycle of a shortest path tree, giving the first polynomial time algorithm [14]. Several other deterministic polynomial time algorithms have been given [1,7,13,17], the fastest being an $O(nm^2/\log n + n^2m)$ time algorithm for the undirected case by Mehlhorn and Michail [17] and an $O(m^3n + m^2n^2\log n)$ time algorithm for the

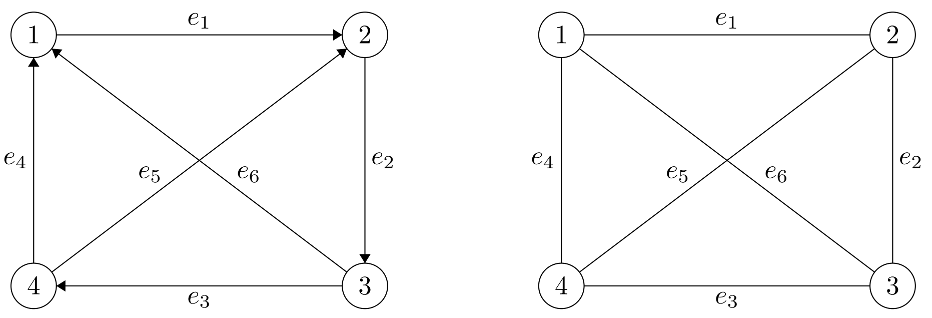

Figure 2 A directed graph with its underlying undirected graph [12].

directed case by Hariharan, Kavitha, and Mehlhorn [12]. There also exist faster $O(m^\omega)$ time randomized algorithms for both undirected and directed graphs where $O(m^\omega)$ is the time needed to multiply two $m \times m$ matrices [1,18].

A surface or 2-manifold with boundary Σ is defined as a compact Hausdorff space such that every point lies in an open neighborhood homeomorphic to the Euclidean plane or the closed half plane. The boundary of the surface is the set of all points whose open neighborhood is homeomorphic to the closed half plane, and every boundary component is homeomorphic to the circle. A cycle *in the surface* is a continuous mapping from the unit circle to the surface, and the cycle is called simple if the mapping is injective. A surface is said to be orientable if it does not contain a subset homeomorphic to the Möbius band, and non-orientable otherwise. The genus of a surface, denoted g, is the maximum number of disjoint cycles in the surface such that their removal leaves a surface that is still connected. Two surfaces are homeomorphic if their genus, number of boundary components, and whether or not they are orientable all agree.

We can improve upon the deterministic running times by restricting ourselves to graphs that can be embedded on a surface of a certain genus. It is possible to bypass the matrix multiplication bound in planar graphs, ultimately resulting in a *near-linear* time algorithm that builds a data structure for quickly retrieving individual cycles [1,3,13]. The minimum cycle bases are identical in planar undirected and directed graphs, so these results work in both settings. Borradaile, Chambers, Fox, and Nayyeri [2] presented an $O(n^\omega + 2^{2g}n^2 + m)$ time algorithm for computing the minimum cycle basis in undirected graphs embedded on orientable surfaces of genus g.

1.1 Our results

We give an $O(n^3 g^2 \log g + m) + \tilde{O}(n^{\omega+1})$ time deterministic algorithm to find the minimum cycle basis for a directed graph embedded in an orientable surface of genus g.[1] At a high level, we follow the same strategy used by others for computing minimum cycle bases [2,7,12]. We describe a way to represent each cycle as an integer vector of dimension d while also maintaining an ordered collection of d-dimensional *support vectors*. We maintain the property that after finding i members of the minimum cycle basis, the latter $d - i$ members of the collection are all orthogonal to the basis cycles' representations. The $i + 1$st cycle is simply the lightest cycle whose representation is *not* orthogonal to the $i + 1$st support vector.

[1] We use $\tilde{O}(\cdot)$ to hide terms polylogarithmic in n.

In general sparse directed graphs, a deterministic search for a single lightest non-orthogonal cycle takes $O(n^3 \log n)$ time [12]. Even in undirected graphs, such a search would take roughly quadratic time. However, Borradaile et al. [2] show how to perform the search in only $O(2^{2g}n)$ time given an undirected graph embedded on an orientable surface of g. Their idea is to partition a collection of candidate cycles into 2^{2g} subsets so that each subset of cycles nest. One can then use the nesting structure to search for the lightest non-orthogonal candidate cycle of any group in only $O(n)$ time.

We offer two main technical contributions on top of Borradaile et al.'s [2] algorithm to get our algorithm for directed graphs. First, we use a recent result of Greene [11] to argue that the number of subsets needed in the partition is actually $O(g^2 \log g)$ (we return to this point when we describe our second result). Second, we show how to extend the search within each group of candidates to work in the directed graphs. Doing so requires us to carefully acknowledge the orientation of edges and cycles as we do our searches while also dealing with the very large support vector elements that may arise over the course of our algorithm from no longer working over a finite field.

It turns out that our observation concerning the size of the candidate cycle partition applies to undirected graphs as well. Consequently, we get an $O(n^\omega + n^2g^2 \log g + m)$ time deterministic algorithm to find the minimum cycle basis of a surface embedded undirected graph. The better running time requires no change to the algorithm of Borradaile et al. [2] beyond the observation which we will describe in the context of directed graphs, so we need not elaborate further beyond stating the relevant theorem.

▶ **Theorem 1.** *Let G be an undirected graph with n vertices and m non-negatively weighted edges, cellularly embedded in an orientable surface of genus g. We can deterministically compute a minimum-weight cycle basis of G in $O(n^\omega + n^2g^2 \log g + m)$ time.*

2 Preliminaries

For a given graph $G = (V, E)$, an embedding of G on Σ is a mapping taking vertices to distinct points on Σ and edges to internally disjoint paths on Σ with endpoints that lie on their incident vertices' points. A face of the embedding is defined to be a maximally connected subset of Σ such that the subset does not intersect the embedded graph. If every face on an embedding is homeomorphic to an open disk we say the embedding is cellular. Only orientable cellular embeddings of graphs will be considered from now on. Without loss of generality, we also assume the surface has exactly one boundary component. Let ℓ denote the number of faces in an embedding of G. Based on our assumptions, Euler's formula guarantees $n - m + \ell = 1 - 2g$.

Every embedded graph G has a dual graph G^* which is constructed by creating a vertex for every boundary component in Σ as well as every face of G. Edges are then created between two vertices in G^* if the corresponding faces and boundary components are separated by an edge. We define the canonical orientation of edge e's dual to cross $\vec{e}$ from left to right. Finally, faces in G^* now correspond to vertices in G. The original graph is then known as the primal graph, where primal vertices are dual to dual faces, and dual vertices are dual to primal faces. We make no notational distinction between corresponding primal and dual objects in this paper.

2.1 Cycle signatures and homology

Let G be directed. A spanning tree of G is a subset of edges of G that form a tree containing every vertex of G. A tree-cotree decomposition is a partition of the edges of G into three sets, T a spanning tree of G, D a spanning tree of G^*, and L the leftover edges. Let $\beta = |L|$. Euler's formula implies $\beta = 2g$ [8, 9].

Let (T, L, D) be an arbitrary tree-cotree decomposition of G. Define c_i for $i \in \{1, \dots, \beta\}$ to be either orientation of the unique simple cycle in G^* created by adding the ith edge in L to D. Let $f_{\beta+1}, \dots f_{m-n+1}$ denote the faces of G. Define c_i for $i \in \{\beta+1, \dots, m-n+1\}$ to be the simple path in D from f_i to the unique dual vertex for Σ's boundary. We define the **signature** $[e] \in \{-1, 0, 1\}^{m-n+1}$ of an edge e as a vector with the ith component defined as follows.

$$[e]_i = \begin{cases} 1 & \text{if } \vec{e} \text{ is in and oriented along } c_i \\ -1 & \text{if } rev(\vec{e}) \text{ is in and oriented along } c_i \\ 0 & \text{otherwise} \end{cases}$$

The **signature** of a cycle C is $[C] = \sum_{e \in E} C(e) \cdot [e]$. Borradaile, et al. [2] show that a similar cycle signature definition produces an isomorphism to the cycle space in an undirected graph. We use a nearly identical argument to prove the following lemma.

▶ **Lemma 2.** *Cycle signatures are an isomorphism between the cycle space of a directed graph and $\mathbb{Q}^{m-n+1}$. In particular, two cycles C and C' are equal if and only if $[C] = [C']$.*

Proof. The definition of cycle signatures immediately implies $[C + C'] = [C] + [C']$ and $[q \cdot C] = q \cdot [C]$ for any two cycles C and C'. Therefore, cycle signatures form a linear map.

Now, consider an arbitrary $w \in \mathbb{Q}^{m-n+1}$. From w, we will construct a cycle C such that $[C] = w$, implying cycle signatures are a surjection. Combined with them being a linear map between equal dimensional vector spaces, we conclude they must be an isomorphism. For each $i \in \{1, \dots, \beta\}$, let C^i correspond to the unique simple cycle sequence in G created from adding the ith edge of L as defined above to T, oriented so that $[C^i]_i = 1$. Next, for each $i \in \{\beta+1, \dots, m-n+1\}$, let C^i correspond to the boundary of face f_i as defined above, again oriented so that $[C^i]_i = 1$. Finally, let $C = \sum_{i=1}^{m-n+1} w_i \cdot C^i$.

Fix any $i, j \in \{1, \dots, m-n+1\}$ such that $i \neq j$. If $i \in \{1, \dots, \beta\}$, then C^i completely avoids the dual cycle or path c_j, implying $[C^i]_j = 0$. If $i \in \{\beta+1, \dots, m-n+1\}$, then either C^i avoids c_j or c_j has exactly one dart entering f_i and one dart leaving f_i, again implying $[C^i]_j = 0$. We conclude $[C]_i = w_i$ for all i. ◀

The homology of G is an algebraic description of the topology of the surface as well as G's embedding. We are only concerned with the one-dimensional cellular homology over the finite field $\mathbb{Z}_2$ and use the underlying undirected graph when referencing the homology of G. We say a cycle sequence or its corresponding cycle is null-homologous if it is the boundary of a subset of faces. Two cycle sequences are homologous if their symmetric difference is null-homologous. These definitions allows us to partition the cycle sequences of G into 2^{2g} homology classes. We define the operation $G \nmid S$ as cutting both G and Σ along some cycle sequence S, creating two copies of S. Cutting G using any cycle sequence from the null-homology class cuts Σ into two separate surfaces, and cutting G along any two non-crossing sequences in the same homology class cuts Σ into two separate surfaces. We also define the **homology signature** $[e]^h \in \{0, 1\}^\beta$ of an edge e as a vector with the ith component defined as follows.

$$[e]_i^h = \begin{cases} 1 & \text{if } \vec{e} \text{ is in } c_i \\ 0 & \text{otherwise} \end{cases}$$

The **homology signature** of a cycle C is $[C]^h$ the bitwise exclusive-or of the homology signatures of its edges. We only use the homology signature to seperate the cycles by homology class and therefore do not use the direction of the edges to define the signature. Therefore this definition matches the undirected case of Borradaile, et al. [2].

2.2 Simplifying assumptions

We assume $g = O(n^{1-\varepsilon})$ for some constant $\varepsilon > 0$; otherwise our algorithms make no improvements upon what is known for general graphs. If G has no faces of degree 1 or 2, then Euler's formula implies the number of edges and faces to be $O(n)$. We guarantee this property of G as follows. If an edge e bounds a face of degree 1 on one side, then $S = \langle \vec{e} \rangle$ corresponds to the lightest cycle for which $C(e) = 1$. Any other cycle with $C(e) \neq 0$ can be made cheaper by setting $C(e)$ to 0, so we can safely assume C_S is in a minimum basis and remove e from G.

Suppose a face f has degree 2. If f is bounded twice by the same edge, then G must be embedded in the sphere and consist of only that single edge. There are no non-zero cycles, so we terminate the algorithm. Otherwise, f is bound by two distinct edges e_1 and e_2. Assume without loss of generality that $w(e_1) \leq w(e_2)$, and let $u{\to}v = \vec{e}_2$. Let σ denote the shortest path in G from u to v, and let $S = \vec{e}_2 \circ rev(\sigma)$, the concatenation of $\vec{e}_2$ with $rev(\sigma)$. Cycle C_S is the lightest cycle for which $C(e_2) = 1$, so we may assume it belongs to a minimum cycle basis. Observe $w(\sigma) \leq w(e_1) \leq w(e_2)$. Any other cycle C with $C(e_2) \neq 0$ can be made strictly lighter by adding or subtracting an appropriate multiple of C_S, so we may then remove e_2 from G.

We can guarantee all faces have degree at least 3 by computing all-pairs shortest paths in the subgraph of G that includes only the lightest member of each set of parallel edges in $O(n^2 \log n + m)$ time. Removing edges as described above takes $O(m)$ additional time total. From here on, we assume there are $O(n)$ edges and faces.

3 Algorithm

Our algorithm computes cycles of the minimum basis one by one. We do this by maintaining a set of support vectors that form the basis for the subspace orthogonal to the set of cycles already computed for the basis. To compute a new cycle in the basis, we choose a support vector that we have not used so far and find the cycle of minimum weight that is not orthogonal the chosen support vector. The unchosen support vectors are then updated so they remain orthogonal to the current incomplete basis we have computed. This is the method that many algorithm have used to compute minimum cycle bases [2, 7, 12].

Specifically, our algorithm uses the prime field modifications for directed graphs made by Hariharan, Kavitha, and Mehlhorn for dealing with the potentially large numbers generated by the coefficients from $\mathbb{Q}$ [12]. For choosing the non-orthogonal cycle our algorithm follows the basic idea of Borradaile, et al. [2] of constructing so-called *region trees* based on homology classes to improve the cycle selection time. However, modifications must be made to the cycle selection procedure to account for the coefficients from $\mathbb{Q}$.

3.1 Computing Support Vectors

The method of computing support vectors that are used to calculate the minimum cycle basis follows from the following theorem given and proved by Hariharan, Kavitha, and Mehlhorn [12]. We have adapted their theorem to use our cycle signatures. Its proof requires no changes thanks to the isomorphisms between various representations of the cycles.

▶ **Theorem 3.** *Integral cycles $C_1, \dots C_d$ form a minimum cycle basis if there are vectors $N_1 \dots N_d$ in $\mathbb{Q}^m$ such that for all i, $1 \leq j \leq i$:*

1. *Prefix orthogonality: $\langle N_i, [C_j] \rangle = 0$ for all j, $1 \leq j < i$.*
2. *Nonorthogonality: $\langle N_i, [C_i] \rangle \neq 0$.*
3. *Lightness: C_i is a lightest integral cycle with $\langle N_i, [C_i] \rangle \neq 0$*

Algorithm 1 is a simple deterministic algorithm that was given by Kavitha and Mehlhorn to compute the N_i's and C_i' [15].

Algorithm 1 An algorithm to compute N_i's and C_i's.

```
$N_1, \dots, N_d \leftarrow \hat{u}_1, \dots \hat{u}_d$        ▷ ($\hat{u}_d$ has a 1 in the $i$th position and 0's everywhere else)
for $i \leftarrow 1$ to $d$ do
    $C_i \leftarrow$ lightest cycle with non-zero dot product with $N_i$
    for $j \leftarrow i+1$ to $d$ do
        $N_j \leftarrow N_j - N_i \frac{\langle [C_i], N_j \rangle}{\langle [C_i], N_i \rangle}$
        $N_j \leftarrow N_j \frac{\langle [C_i], N_i \rangle}{\langle [C_{i-1}], N_{i-1} \rangle}$
    end for
end for
```

The correctness of this algorithm is based on a lemma given and proved by Kavitha and Mehlhorn [15].

▶ **Lemma 4.** *For any i, at the end of iteration $i-1$, the vectors $N_i, \dots, N_d$ are orthogonal to $[C_1], \dots, [C_{i-1}]$ and moreover for any j with $i \leq j \leq d$,*

$$N_j = \langle [C_{i-1}], N_{i-1} \rangle (x_{j,1}, \dots, x_{j,i-1}, 0, \dots, 0, 1, 0, \dots, 0)$$

where 1 occurs in the jth coordinate and the vector $\boldsymbol{x} = (x_{j,1}, \dots, x_{j,i-1})$ is the unique solution to the set of equations:

$$\begin{pmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_{i-1}^T \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} -c_{1,j} \\ \vdots \\ -c_{i-1,j} \end{pmatrix}$$

Where $\tilde{C}_k$, $1 \leq k < i$, is the restriction of $[C_k]$ to its first $i-1$ coordinates and $c_{k,j}$ is the jth coordinate of $[C_k]$.

Furthermore, the running time of this algorithm was shown by Kavitha and Mehlhorn to be $\tilde{O}(m^4) + mO(\text{cycle})$, where $O(\text{cycle})$ is the time taken to find the lightest non-orthogonal integral cycle to N_i [15].

This simple algorithm was then improved upon by Hariharan, Kavitha, and Mehlhorn. By using a divide-and-conquer approach, the calculations spent updating the N_i vectors can be done in bulk [12]. Algorithm 2 gives the recursive step from index l to index h is as follows.

Algorithm 2 A faster algorithm to compute N_i's and C_i's.

```
$mid \leftarrow \lceil (l+h)/2 \rceil$
Find cycles $C_l, \ldots, C_{mid}$ using $N_l, \ldots, N_{mid}$ recursively
Update the vectors $N_{mid+1}, \ldots, N_h$
Find cycles $C_{mid+1}, \ldots, C_h$ using $N_{mid+1}, \ldots, N_h$ recursively
```

To compute the minimum cycle basis, we call this algorithm with $N_1, \ldots, N_d$ initialized to the first d unit vectors, $l = 1$, and $h = d$. Our goal when updating the vectors is to make the vectors $N_{mid+1}, \ldots, N_h$ orthogonal to the newly computed cycles $C_l, \ldots, C_{mid}$. To update the vectors, we make the following definitions.

- $A \in \mathbb{Q}^{k \times m}$, A's ith row is $[C_{l+i-1}]$
- $D \in \mathbb{Q}^{(h-k)\times(h-k)}$, D has the value $\langle N_{mid}, [C_{mid}] \rangle \,/\, \langle N_{l-1}, [C_{l-1}] \rangle$ in every diagonal
- $X \in \mathbb{Q}^{k \times (h-k)}$
- $N_d \in \mathbb{Q}^{m \times (h-k)}$, N_d's jth column is N_{mid+j}
- $N_u \in \mathbb{Q}^{m \times k}$, N_u's jth column is N_{l+j-1}

As shown by Hariharan, Kavitha, and Mehlhorn, we can update the vectors by solving the following system for X [12]:

$$AN_dD = -AN_uX$$

By construction (NN_u) is lower triangular with non-zero diagonal entries, and therefore is invertible. Hence we can write

$$X = -(AN_u)^{-1}AN_dD.$$

Finally our updated vectors $N_{mid+1}, \ldots, N_h$ can be found by computing $N_uX + N_dD$. This process of updating the vectors takes $O(nk^{\omega-1})$ arithmetic operations in total, where n^ω is the time it takes to multiply two $n \times n$ matrices using fast matrix multiplication. However, because we are in a directed graph, the elements of $(AN_u)^{-1}$ can be as large as $d^{\Theta(d^2)}$. Even assuming a model of computation that allows for constant time operations on words of up to $O(\log n)$ bits, we see each arithmetic operation can take up to $\tilde{\Theta}(d^2)$ time. Therefore, we get a runtime of $\tilde{\Theta}(n^{\omega+2})$ for the outermost step which is slower than the simpler algorithm for directed graphs [12].

In order to solve the problem of large intermediate elements we run the above algorithm over a ring $\mathbb{Z}_R$ where R is a specially chosen prime. Working over this ring allows us to do arithmetic operations in $O(1)$ time each. In order to be able to recover our N_j vectors, we must choose a R such that R is relativity prime to $\langle N_l, [C_l] \rangle, \langle N_{l+1}, [C_{l+1}] \rangle, \ldots, \langle N_{mid}, [C_{mid}] \rangle$ (to ensure AN_u is invertible in $\mathbb{Z}_R$), and relativity prime to $\langle N_{l-1}, [C_{l-1}] \rangle$ (to ensure that $\langle N_{mid}, [C_{mid}] \rangle \,/\, \langle N_{l-1}, [C_{l-1}] \rangle$ is well defined in $\mathbb{Z}_R$) [12]. We can select R for each iteration of the recursive step using Algorithm 3.

Pre-computing d^2 primes takes $\tilde{O}(d^2)$ time, and pre-computing the products $P_1, \ldots, P_d$ takes $\tilde{O}(d^3)$ time. The algorithm to select R runs in $\tilde{O}(d^2)$ time [12]. All together, the total time complexity for a single update step with modulo arithmetic is $\tilde{O}(n^2k^{\omega-1} + d^2k)$ or $\tilde{O}(n^2k^{\omega-1})$.

3.2 Finding the Minimum Cycle

What remains is to find the integral cycle of minimum weight that is non-orthogonal to a given support vector. To do this quickly, we modify an algorithm given by Borradaile, et al. [2] for undirected graphs. Their algorithm first computes a set of $O(2^{2g}n)$ candidate cycles that

Algorithm 3 An algorithm select a suitable R.

```
Require: $p_1, \ldots, p_{d^2}$, primes each of which is at least $d$, the products $P_1 = p_1 \ldots p_d, P_2 = p_1 \ldots p_{2d}, \ldots, P_d = p_1 \ldots p_{d^2}$ precomputed before running algorithm.
  $L \leftarrow \langle N_{l-1}, [C_{l-1}] \rangle \langle N_l, [C_l] \rangle \ldots \langle N_{mid}, [C_{mid}] \rangle$
  Binary search $P_1 \ldots P_d$ to find the smallest $s \geq 0$ such that $P_{s+1} \nmid L$
  Determine a $p \in \{p_{sd+1}, \ldots, p_{sd+d}\}$ such that $p \nmid L$
  return $p^d$
```

contain every member of some minimum cycle basis. The cycles are then partitioned into 2^{2g} sets and a tree representation of each set is built in $O(n^2)$ time. Each tree can then be searched in $O(n)$ time to find the minimum weight non-orthogonal cycle for that tree's subset of cycles [2].

We first show that it suffices to consider essentially the same small set of candidate cycles. A **Horton cycle** is defined to be a simple cycle sequence given by a shortest x, u-path, a shortest x, v-path, and an edge uv for some vertices x, u, and v. The set of all Horton cycles on a graph is given by the set of $m - n + 1$ fundamental cycles of the n shortest path trees [14]. A cycle sequence S is said to be **isometric** if for all vertices x and y appearing along S, the set of S's edges contain a shortest x, y-path.

▶ **Theorem 5.** *There exists a minimum cycle basis of directed graph G where every member corresponds to an isometric Horton cycle.*

Proof. Let $\mathcal{C}$ be a minimum cycle basis of G, let $C \in \mathcal{C}$, and let $\mathcal{C}' = \mathcal{C} \setminus \{C\}$. For any three paths α, β, and γ in G, we observe that if $C_{\alpha \circ rev(\beta)}$ and $C_{\beta \circ rev(\gamma)}$ are both dependent on cycles in $\mathcal{C}'$, then $C_{\alpha \circ rev(\gamma)}$ is dependent on cycles in $\mathcal{C}'$ by simple algebra on the cycle signatures. Therefore, the cycle sequences corresponding to dependent cycles follow the *three path condition* as defined by Cabello, Colin de Verdière, and Lazarus [4]. They show there exists a shortest cycle sequence *not* in the family corresponding to dependent cycles that is a fundamental cycle of a shortest paths tree, and that the root of this shortest paths tree can be any vertex of the cycle sequence. Such a cycle sequence is an isometric Horton cycle. ◀

From here on, we must assume all shortest paths are unique, and this assumption can be enforced without increasing our already-quadratic running time [13]. We compute all Horton cycles of the graph and extract the isometric cycles as shown by Amaldi et al [1]. As they are both dependent upon one-another, we keep only one of each pair of an isometric Horton cycle and its reversal.

Borradaile, et al. [2] show the isometric cycles can be partitioned into subsets of size $O(n)$, each corresponding to one of the 2^{2g} different homology classes. However, we observe that the number of non-empty classes is much smaller. Uniqueness of shortest paths implies that if two isometric cycles intersect, they do so along a single shortest path [3]. A recent result of Greene [11] implies that a set of cycle sequences that pairwise intersect (or cross) at most once must live in at most $O(g^2 \log g)$ *homotopy* classes. Homotopy is a finer relation between cycle sequences than homology, implying the number of non-empty homology classes for a collection of isometric cycles to be $O(g^2 \log g)$ as well. In particular, our minimum cycle basis algorithm need only consider $O(ng^2 \log g)$ candidate cycles total.

In order to follow the approach of Borradaile, et al. [2], we must first convert their algorithm to use cycle signatures in $\mathbb{Q}^d$ instead of $\mathbb{Z}^d$. As shown by Hariharan, Kavitha, and Mehlhorn using Hadamard's inequality the support vector can have elements of up to size $d^{d/2}$ [12] so naively modifying the algorithm will lead to problems with the speed of arithmetic. We use a similar approach to that of Hariharan, Kavitha, and Mehlhorn [12] and first find the minimum weight cycles that are non-orthogonal to our chosen support vector modulo some prime p.

We first describe how to construct the region trees that will aid us in our search. We note that this step is done once before running the main algorithm and does not depend on the prime p. We compute the homology signatures for the candidate cycles and split them into their $O(g^2 \log g)$ homology classes. Then for each homology class, we compute its own region tree. Each vertex v of a region tree corresponds to a set of faces F^v, and each edge e corresponds to a candidate cycle C^e. A region tree $T_\mathcal{S}$ also has a designated cycle denoted $T^0_\mathcal{S}$. The **region trees** are defined as the result of Algorithm 4.

Algorithm 4 An algorithm to create a region tree for a homology class.

```
Require: Non-empty set of isometric Horton cycles $\mathcal{S}$ all belonging to the same homology
  class and the graph $G$
  $T_\mathcal{S}$ starts out with one vertex $v$ with $F^v$ equal to all faces of $G$
  if members of $\mathcal{S}$ have non-trivial homology then
      Choose an arbitrary $S_0 \in \mathcal{S}$
      $T_\mathcal{S}$ has a single edge $e$ looping on $v$ with $C^e = C_{S_0}$
      $G' \leftarrow G \nmid S_0$
      for $S \in \{\mathcal{C} \setminus S_0\}$ do
          $G' \leftarrow G' \nmid S$
          Cutting $G'$ splits some component of $G'$ into two new components
          Split vertex $v$ corresponding to cut component into two new vertices with corre-
  sponding faces from the newly created components
          Assign $C_S$ to the new edge
      end for
      $T^0_\mathcal{S} \leftarrow C_{S_0}$
      Remove edge corresponding to $S_0$
      Root $T_\mathcal{S}$ at the vertex whose region contains the boundary
  else
      $G' \leftarrow G$
      for $S \in \mathcal{S}$ do
          Cutting $G'$ splits some component of $G'$ into two new components
          Split vertex corresponding to cut component into two new vertices with correspond-
  ing faces from the newly created components
          Assign $C_S$ to the new edge
      end for
      $T^0_\mathcal{S}$ is assigned the 0-cycle
      Root $T_\mathcal{S}$ at the vertex whose region contains the boundary
  end if
  Negate all cycles as needed so the root region lies to the right of their corresponding
  sequences
```

In Algorithm 4, the operation $G \nmid \gamma$ takes $O(n)$ time, so the entire algorithm takes $O(n^2)$ time for each region tree. Therefore, we can preprocess G in $O(n^2 g^2 \log g)$ time. Examples of constructed region trees can be seen in Figures 3 and 4.

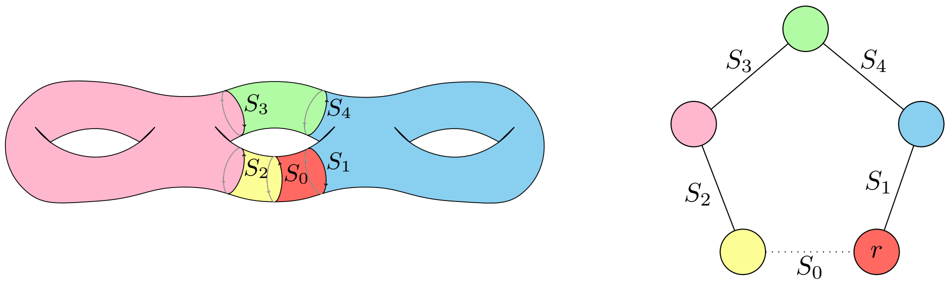

Figure 3 A region tree for a set of edges with non-trivial homology [2]. The root is depicted with r.

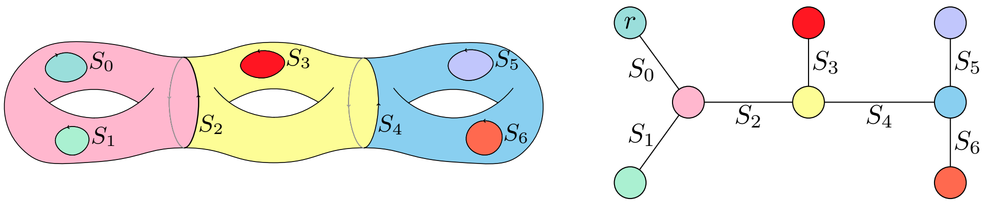

Figure 4 A region tree for a set of edges with null-homologous homology signature [2]. The root is depicted with r.

We can use the region trees to find a shortest cycle that is non-orthogonal modulo p with a given support vector N as shown in Algorithm 5. This algorithm is based on that of Borradaile, et al. [2] with modifications to account for the prime p and the importance of cycle orientations. Given a region tree $T_{\mathcal{S}}$, we start by computing $\langle N, [T^0_{\mathcal{S}}]\rangle$. We then travel the region tree in postorder, computing the inner product for each edge's cycle by adding the contributions from only the components of the cycle's signature that differ from those of the children edge's cycles. Recall the definition of the dual paths c_i used to define the cycle signatures in Section 2. In order to do add inner product contributions consistently, we need to know whether each new cycle C we consider crosses a dual path c_i a different number of times than $T^0_{\mathcal{S}}$. Fortunately, this number changes precisely when we consider the parent edge of the region containing face f_i. And because Algorithm 4 orients the cycles' sequences so the root region of $T_{\mathcal{S}}$ lies to each sequence's right, the net number of crossings and thus the corresponding component of the cycle signature changes by exactly 1. Algorithm 5 considers each face of the graph at most once, so the total runtime to walk up the tree is $O(n)$. We must run this algorithm once for every region tree, so the total runtime to find a non-orthogonal cycle modulo p is $O(ng^2 \log g)$.

In order to obtain a lightest non-orthogonal cycle from a collection of lightest non-orthogonal cycles modulo p, we first pre-compute primes $p_1, \ldots, p_{d/2}$ each of which is at least d. The ring $\mathbb{Z}_{\prod p_i}$ is isomorphic to $\mathbb{Z}_{p_1} \times \cdots \times \mathbb{Z}_{p_{d/2}}$, which implies that any non-zero element whose magnitude is less than $\prod_{i=1}^{d/2} p_i$ is mapped to a tuple of values that is not the zero vector. Therefore, if we run our algorithm for cycle searching $d/2$ times, once for each prime we pre-computed, each cycle that is non-orthogonal will also be non-orthogonal for some $p \in \{p_1, \ldots, p_{d/2}\}$. We get a total runtime of $O(n^2 g^2 \log g)$ to find a lightest non-orthogonal cycle for some given support vector.

Algorithm 5 An algorithm find the lightest cycle non-orthogonal to a given support vector N modulo p for a given region tree $T_{\mathcal{S}}$.

```
Require: Region tree $T_{\mathcal{S}}$
  $m \leftarrow \infty$                                    ▷ The current minimum weight
  $C \leftarrow$ NULL                  ▷ The current cycle referring to the minimum weight
  for edge $e \in T_{\mathcal{S}}$ in postorder do
      if $e$ goes to a leaf then
          $z_e \leftarrow \langle N, [T^0_{\mathcal{S}}] \rangle$
      else
          $z_e \leftarrow 0$
      end if
      for child edge $e'$ of $e$ do
          $z_e \leftarrow z_e +_p z_{e'}$
      end for
      for $f_i \in F(\text{bottom}(e))$ do
          $z_e \leftarrow z_e +_p N^i$
      end for
      if $z_e \neq 0$ and $w(C^e) < m$ then $m \leftarrow w(C^e)$ and $C \leftarrow C^e$
  end for
  return $C$
```

3.3 Final analysis

Let $T(k)$ denote the time to run Algorithm 2 in our setting when $h - l + 1 = k$.

$$T(k) = \begin{cases} 2T(k/2) + \tilde{O}(n^2 k^{\omega-1}) & \text{if } k > 1 \\ n^2 g^2 \log g & \text{if } k = 1 \end{cases}$$

This recurrence solves to $O(n^3 g^2 \log g) + \tilde{O}(n^{\omega+1})$. Including the $O(m)$ time needed to guarantee all faces have degree 3 or greater, we get a total running time of of $O(n^3 g^2 \log g + m) + \tilde{O}(n^{\omega+1})$. As in Borradaile et al. [2], any method to improve the speed of selecting support vectors would improve the time required to find a minimum cycle basis.

We conclude with a theorem summarizing our main result.

▶ **Theorem 6.** *Let G be a directed graph with n vertices and m non-negatively weighted edges, cellularly embedded in an orientable surface of genus g. We can deterministically compute a minimum-weight cycle basis of G in $O(n^3 g^2 \log g + n^{\omega+1} + m)$ time.*

References

1 Edoardo Amaldi, Claudio Iuliano, Tomasz Jurkiewicz, Kurt Mehlhorn, and Romeo Rizzi. Breaking the o(m2n) barrier for minimum cycle bases. In Amos Fiat and Peter Sanders, editors, *Algorithms - ESA 2009*, pages 301–312, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

2 Glencora Borradaile, Erin Wolf Chambers, Kyle Fox, and Amir Nayyeri. Minimum cycle and homology bases of surface-embedded graphs. *J. Comput. Geom.*, 8(2):58–79, 2017. doi:10.20382/jocg.v8i2a4.

3 Glencora Borradaile, Piotr Sankowski, and Christian Wulff-Nilsen. Min *st*-cut oracle for planar graphs with near-linear preprocessing time. *ACM Trans. Algorithms*, 11(3):16:1–16:29, 2015. doi:10.1145/2684068.

4 Sergio Cabello, Éric Colin de Verdière, and Francis Lazarus. Finding shortest non-trivial cycles in directed graphs on surfaces. In *Proceedings of the Twenty-Sixth Annual Symposium on Computational Geometry*, SoCG '10, pages 156–165, New York, NY, USA, 2010. Association for Computing Machinery. `doi:10.1145/1810959.1810988`.

5 A. C. Cassell, J. C. De C. Henderson, K. Ramachandran, and Alec Westley Skempton. Cycle bases of minimal measure for the structural analysis of skeletal structures by the flexibility method. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 350(1660):61–70, 1976. `doi:10.1098/rspa.1976.0095`.

6 L. Chua and Li-Kuan Chen. On optimally sparse cycle and coboundary basis for a linear graph. *IEEE Transactions on Circuit Theory*, 20(5):495–503, 1973. `doi:10.1109/TCT.1973.1083752`.

7 J. Coelho de Pina. *Applications of shortest path methods*. PhD thesis, University of Amsterdam, 1995.

8 David Eppstein. Dynamic generators of topologically embedded graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 599–608, USA, 2003. Society for Industrial and Applied Mathematics.

9 Jeff Erickson and Amir Nayyeri. Minimum cuts and shortest non-separating cycles via homology covers. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pages 1166–1176, USA, 2011. Society for Industrial and Applied Mathematics.

10 Petra Gleiss, Josef Leydold, and Peter Stadler. Circuit bases of strongly connected digraphs. *Santa Fe Institute, Working Papers*, 23, January 2001. `doi:10.7151/dmgt.1200`.

11 Joshua Evan Greene. On loops intersecting at most once. *Geometric and Functional Analysis*, 29(6):1828–1843, December 2019. `doi:10.1007/s00039-019-00517-0`.

12 Ramesh Hariharan, Telikepalli Kavitha, and Kurt Mehlhorn. Faster algorithms for minimum cycle basis in directed graphs. *SIAM Journal on Computing*, 38(4):1430–1447, 2008. `doi:10.1137/060670730`.

13 David Hartvigsen and Russell Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM J. Discret. Math.*, 7(3):403–418, August 1994. `doi:10.1137/S0895480190177042`.

14 Joseph Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16:358–366, April 1987. `doi:10.1137/0216026`.

15 Telikepalli Kavitha and Kurt Mehlhorn. Algorithms to compute minimum cycle basis in directed graphs. *Theory of Computing Systems*, 40:485–505, June 2007. `doi:10.1007/s00224-006-1319-6`.

16 Donald E. Knuth. *The Art of Computer Programming, Volume 1 (3rd Ed.): Fundamental Algorithms*. Addison Wesley Longman Publishing Co., Inc., USA, 1997.

17 Kurt Mehlhorn and Dimitrios Michail. Minimum cycle bases: Faster and simpler. *ACM Trans. Algorithms*, 6(1), December 2010. `doi:10.1145/1644015.1644023`.

18 Abhishek Rathod. Fast algorithms for minimum cycle basis and minimum homology basis. In Sergio Cabello and Danny Z. Chen, editors, *36th International Symposium on Computational Geometry, SoCG 2020, June 23-26, 2020, Zürich, Switzerland*, volume 164 of *LIPIcs*, pages 64:1–64:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.SoCG.2020.64`.

19 Geetika Tewari, Craig Gotsman, and Steven Gortler. Meshing genus-1 point clouds using discrete one-forms. *Computers & Graphics*, 30:917–926, December 2006. `doi:10.1016/j.cag.2006.08.019`.