

Improved minimum cycle bases algorithms by restriction to isometric cycles *

E. Amaldi[†] C. Iuliano[†] T. Jurkiewicz[‡] K. Mehlhorn[‡] R. Rizzi[§]

Abstract

We present improved algorithms for finding minimum cycle bases in undirected and directed graphs. For general graphs, the new algorithms are Monte Carlo and have running time $O(m^\omega)$, where m is the number of edges (arcs) and ω is the exponent of fast matrix multiplication, assuming $\omega > 2$. For planar graphs, the algorithm is deterministic and has running time $O(n^2)$, where n is the number of nodes, while the previous best one was $O(n^2 \log n)$. We also observe that this algorithm for planar graphs solves the problem for more specialized classes of cycle bases, namely integral, totally unimodular and weakly fundamental cycle bases.

A key ingredient to our improved running times is the insight that the search for minimum cycle bases can be restricted to a subset of at most nm candidate cycles, the so-called isometric cycles, whose total number of edges is bounded above by nm .

1 Introduction

Let $G = (V, E)$ be a connected undirected graph with n nodes and m edges. Assume that G is simple, i.e., without loops and multiple edges. We denote the edges by e_{uv} , where u and v are the endpoints (omitted when not needed). By arbitrarily fixing an orientation for each edge of G , we obtain a directed graph. We use G to denote the resulting directed graph and also the underlying undirected graph.

Let κ be a field. A κ -cycle¹ C in G is a vector in κ^E such that for any node $v \in V$ we have

$$\sum_{e \in \delta^+(v)} C_e = \sum_{e \in \delta^-(v)} C_e,$$

where $\delta^+(v)$ and $\delta^-(v)$ denote respectively the set of edges leaving and entering the node v . The composition of two cycles is the componentwise addition of the two corresponding vectors, each one possibly multiplied by a non-null element of κ .

The set of all κ -cycles with the operation of composition forms a vector space over κ , called the κ -cycle space of G .²

*A preliminary version of this work appeared in ESA 2009 [1]

[†]Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano, Italy

[‡]Max-Planck-Institut für Informatik, Saarbrücken, Germany

[§]Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Udine, Italy

¹Note that κ -cycles are often referred to as κ -flows in the literature related to other combinatorial optimization problems on graphs.

²If we reverse the orientation of an edge e in G , all cycles derive from those of G by changing the value of the component associated to e with the opposite one. Thus the choice of edge orientations is not essential.

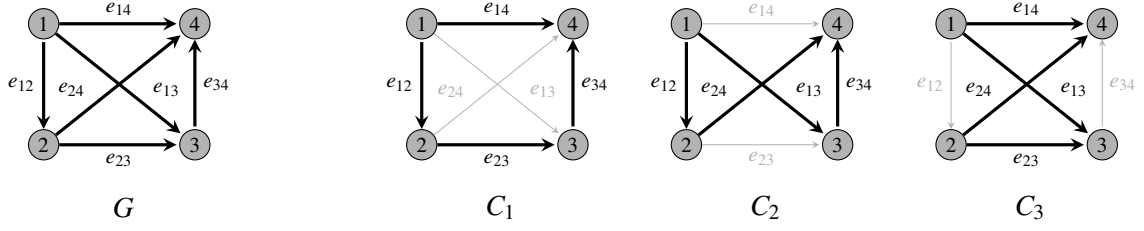


Figure 1: A graph G and three cycles C_1 , C_2 , and C_3 . The edges of G are oriented as shown. The components of the vectors are ordered as follows: e_{12} , e_{13} , e_{14} , e_{23} , e_{24} , e_{34} . Since by a counter-clockwise walk on C_1 we can traverse the edges e_{12} , e_{23} , and e_{34} in forward direction and the edge e_{14} in backward direction, we write $C_1 = (1, 0, -1, 1, 0, 1)$. Similarly $C_2 = (1, -1, 0, 0, 1, -1)$ and $C_3 = (0, 1, -1, -1, 1, 0)$. Cycles C_1 , C_2 , and C_3 form a directed cycle basis of G . But C_1 , C_2 , and C_3 do not form an undirected cycle basis. In fact, in $GF(2)$ each of the three cycles is equal to the sum of the other two.

A cycle C is *simple* if $C_e \in \{-1, 0, +1\}$ for each edge $e \in E$. A simple cycle C is *elementary* if the set of its edges $\{e \in C\} = \{e \in E : C_e \neq 0\}$ forms a connected subgraph of G in which every node has degree 2. Every cycle can be written as the composition of elementary cycles. There exists a walk through the edges of an elementary cycle C such that $C_e = 1$ or $C_e = -1$ if the edge e is traversed in the forward or in the backward direction, respectively.

A κ -*cycle basis* is a set of simple κ -cycles forming a basis of the cycle space. If the graph G is connected the cycle basis consists of $v := m - n + 1$ cycles, where v is the *cyclomatic number* of G . The cycle space and cycle basis are referred to as *directed cycle space* and *basis* when $\kappa = \mathbb{Q}$, the field of rationals, and as *undirected cycle space* and *basis* when $\kappa = GF(2)$, the field of two elements $\{0, 1\}$ where the addition is the *sum modulo 2*. Note that in the latter case the orientation plays no role (since $-1 = 1$), i.e., the graph is undirected, all cycles are simple and correspond to subsets of edges such that every node has even degree, and the cycle composition is the symmetric difference of the edges sets.

Every undirected cycle basis becomes a directed cycle basis by taking into account the orientation of the edges. The reverse does not hold, as illustrated in Figure 1 (see [14]; we are not guaranteed to obtain an undirected cycle basis from a directed one by neglecting the orientation).

Assume that a non-negative weight w_e is assigned to each edge $e \in E$. The weight of a simple cycle is the sum of the weights of its edges, i.e.,

$$w(C) := \sum_{e \in C} w_e,$$

and the weight of a cycle basis B is the sum of the weights of its cycles, i.e.,

$$w(B) = \sum_{C \in B} w(C).$$

The *length* of a simple cycle C is its number of edges, i.e., $|C|$. For unweighted graphs, the weight equals the length.

Given a graph G with a non-negative weight on each edge, we consider the problem of finding a *minimum κ -cycle basis* of G , that is, a κ -cycle basis with minimum weight. Since all edge weights are assumed to be non-negative, there always exists a minimum cycle basis consisting of elementary cycles. Therefore, we restrict our attention to minimum cycle bases consisting of elementary cycles.

Minimum cycle bases are of interest in a number of applications, including, the analysis of electrical networks [4], the analysis of chemical and biological pathways [7], periodic scheduling [5], surface reconstruction [21], and graph drawing [16]. The reader is referred to [13] for a survey.

In [11], Horton proposes the first polynomial-time algorithm for finding a minimum cycle basis in undirected graphs. It has running time $O(m^3n)$. In a sequence of papers [5, 8, 3, 15, 20, 2], the running time is reduced to $O(m^2n/\log n)$. In [14] the first polynomial-time algorithm for minimum directed cycle bases with running time $O(m^4n)$ is proposed. In a sequence of papers [17, 12, 9, 20] the running time is reduced to $O(m^3n)$ deterministic time and $O(m^2n)$ randomized time.

We present improved algorithms for finding minimum undirected and directed cycle bases. We design a randomized algorithm with running time $O(m^\omega)$, where ω is the exponent of fast matrix multiplication. The previous best algorithms already use fast matrix multiplication. Our improvement is due to new structural and algorithmic insights. A key ingredient to our improved running times is the observation that the search for minimum cycle bases can be restricted to a subset of the set of candidate cycles considered by Horton [11] that has a total length of at most nv . For planar graphs, the algorithm is deterministic and we reduce the running time from $O(n^2 \log n)$ [10] to $O(n^2)$. This implies a similar improvement for the all-pairs minimum cut problem. As shown in [22], $O(n^2)$ is optimal if an explicit representation of the basis is required and can be improved only if an implicit computation is allowed.

The paper is organized as follows. In the next section we improve upon Horton’s result by characterizing the so-called isometric cycles, deriving the upper bound of nv on their total number of edges and presenting an efficient $O(nm)$ procedure for detecting them. In Section 3, we exploit this structural insight to derive an $O(m^\omega)$ Monte Carlo algorithm for minimum undirected and directed cycle bases. In Section 4, we exploit it to derive an $O(n^2)$ deterministic algorithm for minimum cycle bases in planar graphs. We also show that for planar graphs this algorithm solves the problem for more specialized classes of cycle bases.

2 Structural results

For each pair of nodes u and v , let p_{uv} denote a minimum weight path in G from u to v . We assume that the collection of the minimum weight paths in the undirected graph G is consistent, i.e., if nodes y and z lie on p_{uv} then p_{yz} is a subpath of p_{uv} , so that, in particular, p_{vu} is the reverse of p_{uv} . From now on we will refer to such a set of minimum weight paths as the *shortest* paths.

This assumption can, for instance, be guaranteed by the following “lexicographic ordering”. Given an arbitrary numbering of the nodes from 1 to n , a path p is considered shorter than a path q of the same total weight if the length of p is strictly smaller than the length of q . In case of ties, the shortest path between p and q will be the one that contains the node with minimum index in the non-common part. We adopt a similar criterion also for comparing cycles. Notice that with the above criterion there is a unique shortest path between u and v for any two nodes u and v . For a modified all-pairs shortest path algorithm that ensures lex-shortest paths in time $O(nm + n^2 \log n)$ see [10]. We indicate with T_x the shortest path tree rooted at node x , that is the union of the paths p_{xv} for all nodes $v \in V$.

The results in this section hold for both undirected and directed cycle bases.

2.1 Isometric cycles

In [11], Horton proposes the first polynomial-time algorithm for finding a minimum undirected cycle basis in undirected graphs. The main observation is the following.

Property 2.1 ([11]) For any two nodes u and v of a cycle C in a minimum cycle basis, p_{uv} is contained in C .

Proof: Suppose not, and call q'_{uv} and q''_{vu} the two paths connecting u and v in C . Then by substituting C with $C' = q'_{uv} + p_{vu}$ or $C'' = q''_{vu} + p_{uv}$ we would obtain a basis of smaller weight. The fact that at least one of these two substitutions results into a new basis follows from the fact that $C = C' + C''$. ■

A node/edge pair (x, e_{uv}) is a *representation* of an elementary cycle $C = C(x, e_{uv}) := p_{xu}e_{uv}p_{vx}$ when p_{xu} , e_{uv} , and p_{vx} are edge-disjoint.

Proposition 2.2 ([11]) For any node x of a cycle C in a minimum cycle basis, there is an edge e_{uv} in C such that $C = C(x, e_{uv})$

Proof: Let $C = (v_0, v_1, \dots, v_k)$, where $x = v_0 = v_k$. Since the empty path is the shortest path from x to x and C is not the shortest path from x to x , there must be an i such that $p_{xv_i} = (v_0, v_1, \dots, v_i)$ but $p_{xv_{i+1}} \neq (v_0, v_1, \dots, v_i, v_{i+1})$. Then, by Property 2.1, $p_{xv_{i+1}} = (v_k, v_{k-1}, \dots, v_{i+1})$ and hence $e_{v_i v_{i+1}}$ is the desired edge. ■

Proposition 2.2 implies that the polynomial-size subset of candidate cycles consisting of all the cycles of the form $C(x, e_{uv})$, for any possible choice of a node x and an edge e_{uv} , is guaranteed to contain a minimum cycle basis. We denote this set by \mathcal{H} and we refer to it as the set of Horton candidate cycles. Degenerate cases, when e_{uv} is in the shortest path tree T_x or p_{xu} and p_{vx} have more than node x in common (see Figure 2(a)), must be discarded. Since for each node x there are v edges not in T_x , the number of Horton candidate cycles is at most nv , that is $O(nm)$.

This result directly leads to Horton's polynomial-time algorithm. Since the family of the independent sets of cycles of a graph defines a matroid, it suffices to order all Horton candidate cycles by non-decreasing weight and then use a linear independence test (e.g., binary Gaussian elimination) to select from \mathcal{H} the v shortest cycles according to the Greedy algorithm.

But, as already mentioned in [11], the set \mathcal{H} is larger than needed. It contains more cycles than those identified by Property 2.1.

Definition 2.3 An elementary cycle is *isometric* if it contains the shortest path between every pair of its nodes.

We restrict attention to the set of all isometric cycles, that we will denote by \mathcal{I} . Each isometric cycle is clearly a Horton candidate cycle, that is $\mathcal{I} \subseteq \mathcal{H}$, and, by Property 2.1, \mathcal{I} is also guaranteed to contain a minimum cycle basis.

\mathcal{H} is a multi-set because each cycle C in \mathcal{H} can have different representations (x, e_{uv}) where x ranges over the nodes of C . Note that there is no representation for a given node x if C contains more than one co-tree edge with respect to T_x . This is equivalent to the existence of a shortcut between x and another node in C , i.e., the shortest path joining them does not belong to the cycle itself. See Figure 2 (b) and (c) for examples of non-isometric cycles.

Interestingly, the isometric cycles can be characterized in terms of the number of representations, namely each isometric cycle C has exactly $|C|$ representations in \mathcal{H} .

Proposition 2.4 (Isometric cycles) An elementary cycle C is isometric if and only if for any node x of C there is an edge e_{uv} such that $C = C(x, e_{uv})$, i.e., there is a representation for each node in C .

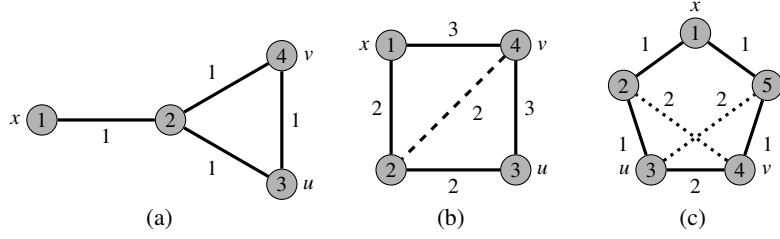


Figure 2: Examples of non-isometric cycles. (a) $C(1, e_{34})$ is not a cycle, because p_{13} and p_{14} have also node 2 in common. The contained cycle $C(2, e_{34})$ has three representations: $(2, e_{34})$, $(3, e_{24})$ and $(4, e_{23})$. (b) p_{13} consists of edges e_{12} and e_{23} . $C(1, e_{34})$ is a non-isometric cycle because of the shortcut, depicted in dashed. (c) $C(1, e_{34})$ is a non-isometric cycle with two shortcuts and no representations for any other node.

Proof: For the *if* direction, consider any two nodes x and z on C and let (x, e_{uv}) the representation for node x . Node z lies on one of the two paths p_{xu} and p_{xv} . Since the set of shortest paths is consistent, p_{xz} is contained in C .

For the *only if* direction, see the proof of Proposition 2.2. ■

By restricting attention to the set of isometric cycles \mathcal{I} instead of \mathcal{H} , we have the following simple but fundamental property.

Property 2.5 *The total length of the isometric cycles is at most nv .*

Proof: An isometric cycle C occurs $|C|$ times in the Horton multi-set and hence $\sum_{C \in \mathcal{I}} |C|$ can be no larger than the size of the Horton multi-set. ■

Note that the sum here is only over the isometric cycles, as we have no control over the number of appearances of non-isometric cycles.

The upper bound of Property 2.5 is tight for instance for the complete graph K_n with n nodes and equal weights on the edges. For any node x , the cycle obtained by adding to T_x any co-tree edge is a triangle. \mathcal{H} consists of nv triangles that are clearly isometric. Since there are three representations of each possible triangle, obtained by taking as x each one of its 3 nodes, \mathcal{I} consists of $nv/3$ triangles. Therefore, the total length of the isometric cycles is exactly nv .

The total length of the isometric cycles may however be much smaller than nv . Consider an $s \times s$ grid with equal weights on the edges. Since $n = s^2$ and $m = 2s(s-1)$, we have $v = (s-1)^2$, and m and v are $O(n)$. The isometric cycles are exactly the grid squares and hence their total length is $4(s-1)^2$, that is $O(n)$, whereas the upper bound of Property 2.5 is $nv = s^2(s-1)^2$, that is $O(n^2)$.

2.2 Efficient detection of isometric cycles

We now present an $O(nm)$ procedure for detecting all isometric cycles from the set of Horton candidate cycles \mathcal{H} . We assume that the shortest paths trees $T_x, x \in V$ are given.

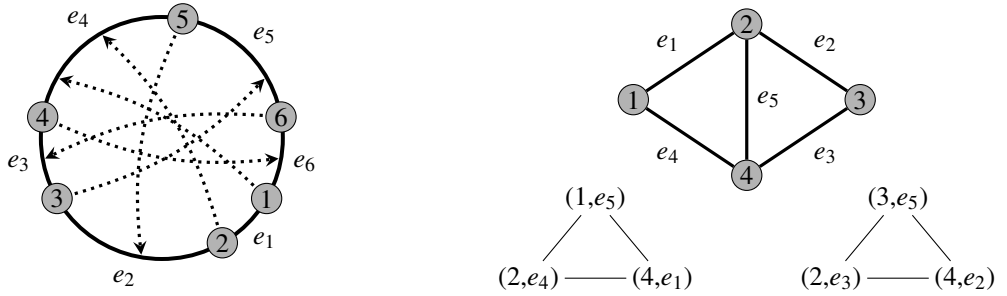


Figure 3: The figure on the left shows an isometric circuit C embedded on a circle. The edges correspond to the circular arcs between the vertices and the length of an arc is proportional to the weight of the corresponding edge. For any vertex x , we have $C = C_{x,e}$ where e contains the mirror image of x with respect to the center of the circle. If we move x around the circle we discover all representations of C . We have the following connections: C_{1,e_4} and C_{2,e_4} are connected by condition 2a, C_{2,e_4} and C_{5,e_2} are connected by condition 2b, and so on.

In the graph on the right all edges have weight one; we select e_1e_2 as the minimum weight path connecting 1 and 3. The circuits C_{1,e_3} and C_{3,e_4} are bad by condition 2c. For the former circuit let $x = 1$, $u = 3$, $v = 4$; then $s_1(3) = 2$ and $s_2(4) \neq 1$ and $s_4(2) \neq 3$. The other circuits are connected as shown below the graph.

First, we give some intuition. Let C be any isometric cycle. Imagine a metric embedding of C onto a circle, i.e., the embedding is such that the length of the arc representing an edge is proportional to the weight of the edge. Then for any vertex $x \in C$, we have $C = C(x,e)$, where e contains the mirror image of x with respect to the center of the circle, see Figure 3. Imagine now that we move x around the circle. Whenever x or its mirror image crosses the endpoint of an edge, we discover a new representation of C .

We are actually facing a somewhat different task. We are given the representations of the Horton cycles and need to filter out the isometric cycles. The intuition above is helpful. When moving around a cycle, we count the different representations of the cycle and link them. For an isometric cycle of length ℓ , we link the ℓ representations and keep the cycle. For non-isometric cycles, the linked representations are discarded together with the corresponding cycle. We next give the details.

We use the following notation. For every node $y \neq x$, let $s_x(y)$ be the child of x in T_x containing y in its subtree. In other words, $s_x(y)$ is the first node on the shortest path from x to y . The vectors s_x for all $x \in V$ can be computed in time $O(n^2)$.

We start with the set of Horton candidate cycles \mathcal{H} . Recall that a Horton candidate cycle is given by $C(x, e_{uv}) := p_{xu}e_{uv}p_{vx}$ for any choice of a node x and an edge e_{uv} such that $C(x, e_{uv})$ is a cycle, namely e_{uv} is a co-tree edge with respect to T_x and p_{xu} and p_{xv} have only node x in common, i.e., $s_x(u) \neq s_x(v)$ (see Figure 2(a)).

The next lemma shows how to identify different representations of the same isometric cycle and how to discover non-isometric cycles. Given a cycle $C(x, e_{uv})$, the idea is to check for two specific nodes x' and x'' of C whether the shortest path $p_{x'x''}$ between them belongs to C . The nodes x' and x'' are chosen so that a negative answer obviously implies that the cycle is non-isometric whereas a positive answer gives a different representation of C for one of x' and x'' . This is achieved by taking $x' = s_x(u)$ and $x'' = v$. Observe, that either x' or x'' would be the next endpoint hit in the circular movement around C . In fact, if $p_{x'v}$ belongs to C there are only two possibilities: $p_{x'v} = e_{x'x}p_{xv}$ and

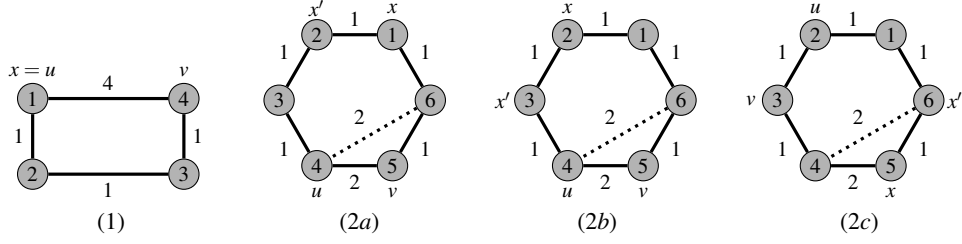


Figure 4: Examples for the different cases of Lemma 2.6. (1) $C(1, e_{14})$ where $x = u = 1$. (2a), (2b) and (2c) are three different representations of the same cycle. (2a) $C(1, e_{45})$ where $s_2(5) = 1$ and we obtain $C(2, e_{45})$. (2b) $C(2, e_{45})$ where $s_3(5) \neq 2$ but $s_5(3) = 4$ and we obtain $C(5, e_{23})$. (2c) $C(5, e_{23})$ where $s_6(3) \neq 5$ and $s_3(6) \neq 2$, because p_{36} consists of edges e_{34} and e_{46} . This implies that the cycle is not isometric. The shortcut is in dashed line.

another representation for C is (x', e_{uv}) ; $p_{x'v} = e_{vu}p_{ux'}$ and another representation for C is $(v, e_{xx'})$. When node x' does not exist because node x coincides with node u , another representation is (v, e_{uv}) .

Lemma 2.6 explains how to check (in constant time) the conditions that allow to identify the different cases, which are illustrated in Figures 3 and 4.

Lemma 2.6 Let $C = C(x, e_{uv})$, with u and v the two endpoints of e_{uv} .

1. If $s_x(u) \neq s_x(v)$ and $x = u$ then $x \neq v$ and $C = e_{uv}p_{vu} = C(v, e_{uv})$.
2. If $s_x(u) \neq s_x(v)$, $x \neq u$, and $x' = s_x(u)$ is the first node on the shortest path from x to u then:
 - (a) if $x = s_{x'}(v)$, then $C = C(x', e_{uv})$,
 - (b) if $x \neq s_{x'}(v)$ and $u = s_v(x')$ then $C = C(v, e_{xx'})$, and
 - (c) if $x \neq s_{x'}(v)$ and $u \neq s_v(x')$ then C is not isometric.

Proof:

If $x = u$, $C = e_{uv}p_{vu} = p_{vu}e_{uv} = C(v, e_{uv})$, which proves the first statement.

If $x \neq u$ and x' is the first node on the shortest path from x to u , we have $p_{xu} = e_{xx'}p_{x'u}$.

If x is the first node on the shortest path from x' to v , then $p_{ux'}p_{x'v} = p_{ux}p_{xv}$. Thus $C = C(x', e_{uv})$, which establishes 2a.

If x is not the first node on the shortest path from x' to v and u is the first node on the shortest path from v to x' then $C = p_{vx}e_{xx'}p_{x'v} = C(v, e_{xx'})$. This establishes 2b.

If x is not the first node on the shortest path from x' to v and u is not the first node on the shortest path from v to x' , C does not contain the shortest path between nodes x' and v . Thus, C is not isometric, which establishes 2c. ■

The procedure implied by Lemma 2.6 allows to identify the different representations of the same isometric cycle and to exclude the non-isometric cycles. We test the condition of the Lemma 2.6 for any element in \mathcal{H} . We set up a directed graph G_R whose nodes are all the elements in \mathcal{H} . We link two representations with an arc according to Lemma 2.6. An arc goes from the tested representation to another representation if conditions 1, 2a and 2b are satisfied. If condition 2c is verified, the representation is labeled as *bad*.

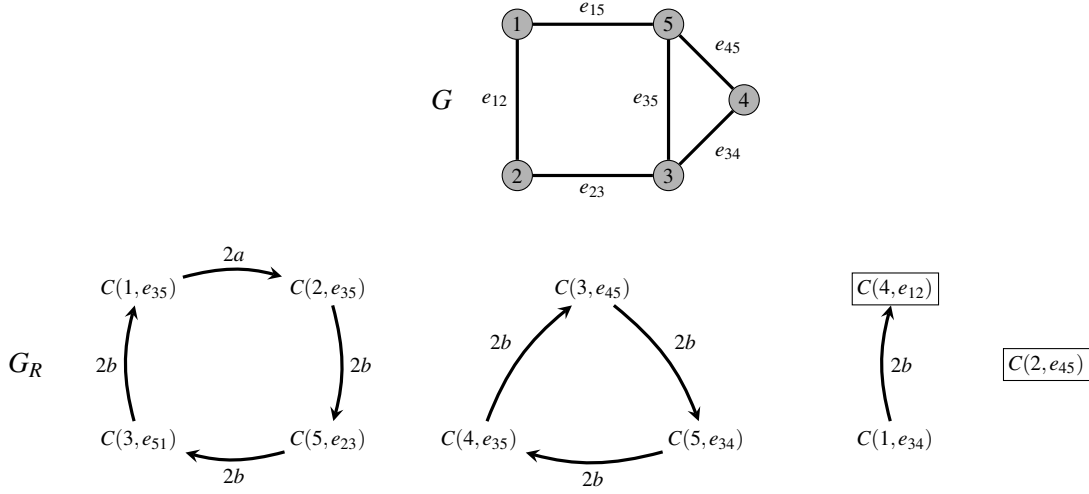


Figure 5: A graph G with $n = 5$, $m = 6$, and $v = 2$ where p_{13} consists of edges e_{12} and e_{23} , p_{25} consists of edges e_{21} and e_{15} . G_R is the graph of the representations. The 10 representations in \mathcal{H} are linked according to the procedure implied by Lemma 2.6. The labels on the arcs indicate the condition of the lemma that allows to proceed from one representation to another one. The boxed representations are *bad* by condition 2c. The first two connected components on the left correspond to the only two isometric cycles of G , the square and the triangle, respectively. Each of them has a number of representations equal to the number of its nodes. The other two non-connected components on the right correspond to the same non-isometric cycle, the outer cycle with 5 nodes. In fact, it has 3 representations in \mathcal{H} . The corresponding shortcut is e_{35} .

A node in G_R has no outgoing arcs if and only if it is labeled as *bad*. Note also that the consistency of the shortest paths in G implies that no two arcs in G_R have the same head or tail. A connected component in G_R corresponds to a distinct isometric cycle of G if and only if it does not contain a *bad* node or equivalently if it is a circuit, i.e., a cycle where all edges have the same orientation, in G_R . Indeed, the conditions of Lemma 2.6 allow to switch among all the representations of a given isometric cycle, linking them in the same connected component of G_R . Conversely, if a cycle is not isometric, the representation for at least one of its nodes does not exist (Proposition 2.4) and this implies that a node in G_R is labeled as *bad* (condition 2c) and is not linked to any other node. See an example in Figure 5.

For the correctness of the procedure, we have to prove the following lemma.

Lemma 2.7 *All representations of an isometric cycle are linked in G_R , they correspond to the nodes of the same connected component.*

Proof: Let $C = (v_0, v_1, \dots, v_k)$ be any isometric cycle, with $v_0 = v_k$. We show that starting from an arbitrary representation of C we can return, according to the conditions of Lemma 2.6, to the same representation after finding the representations for all other nodes of C .

Condition 1 occurs if and only if all representations of C share the same edge e_{uv} . Indeed, C contains the shortest path between the endpoints of e_{uv} , and the representations are linked sequentially from one endpoint to the other one by condition 2a, with condition 1 that allows to switch the role of the two endpoints.

- 1: Initialize S_j to the j -th unit vector for $1 \leq j \leq v$
- 2: **for** $i \leftarrow 1, \dots, v$ **do**
- 3: Compute a minimum weight cycle C_i with $\langle C_i, S_i \rangle \neq 0$
- 4: **for** $j \leftarrow i+1, \dots, v$ **do**
- 5: $S_j = S_j - \frac{\langle C_i, S_j \rangle}{\langle C_i, S_i \rangle} S_i$
- 6: **end for**
- 7: **end for**
- 8: Output $\{C_1, \dots, C_v\}$.

Figure 6: De Pina's algorithm for finding a minimum cycle basis

If condition 1 does not occur, all representations do not share the same edge and w.l.o.g. we suppose that the first and the last node v_0 and v_{k-1} are such that the two corresponding representations do not have the same edge. For a generic representation for node v_i , there is $j(i)$ such that $C(v_i, e_{v_{j(i)}v_{j(i)+1}})$. Thus, we start from representation $C(v_0, e_{v_{j(0)}v_{j(0)+1}})$. Condition 2a allows to link $C(v_i, e_{v_{j(i)}v_{j(i)+1}})$ to $C(v_{i+1}, e_{v_{j(i)}v_{j(i)+1}})$, whereas condition 2b allows to link it to $C(v_{j(i)+1}, e_{v_i v_{i+1}})$. Thus, if we find a representation for a node v_i , we have surely found the representations for the nodes with index from 0, if $i \leq j(0)$, or from $j(0)+1$, if $i \geq j(0)+1$, to i . Suppose we arrive at the representation for node v_{k-1} , i.e., $C(v_{k-1}, e_{v_{j(k-1)}v_{j(k-1)+1}})$. Since this representation does not share the same edge with that for $v_k = v_0$, i.e., $v_{j(k-1)} \neq v_{j(0)}$, and the representation for a node is clearly unique, only condition 2b can be satisfied. Thus, we obtain $C(v_{j(k-1)+1}, e_{v_{k-1}v_k})$. For the same reason, only condition 2a can be satisfied for this representation until $j(k-1) = j(0)$. When this occurs, we can return to the initial configuration $C(v_0, e_{v_{j(0)}v_{j(0)+1}})$ from $C(v_{j(0)+1}, e_{v_{k-1}v_k})$ by condition 2b, after finding all the representations of the isometric cycle C . \blacksquare

Since the conditions of Lemma 2.6 need to be checked only once for each element in \mathcal{H} , we have:

Theorem 2.8 *We can detect a representation for each isometric cycle in $O(nm)$.*

3 Improved randomized algorithms for general graphs

We refine de Pina's approach [5, 13] for finding minimum cycle basis $\{C_1, \dots, C_v\}$, see Figure 6. It operates in phases. In each phase, one cycle is added to the basis. The algorithm also maintains a basis of the orthogonal space; more precisely, at the beginning of the i -th iteration it has a set $\{S_1, \dots, S_v\}$ of linearly independent vectors $S_j \in \kappa^E$ with $\langle C_h, S_j \rangle = 0$, for $1 \leq h < i \leq j \leq v$, where $\langle \cdot, \cdot \rangle$ is the inner product of vectors over κ . Throughout this section, $\kappa = GF(p)$ for a prime p with $p = O(m \log m)$. In particular, arithmetic in $GF(p)$ takes constant time. At the beginning of the computation S_j is initialized to the j -th unit vector for $1 \leq j \leq v$, where the numbering of the edges is such that edges e_{v+1} to e_m form a spanning tree of G .

Steps (4) and (5) of the algorithm make the S_j orthogonal to C_i while maintaining orthogonality of S_j to C_1, \dots, C_{i-1} , for $j > i$. Updating the vectors S_j as shown takes time $O(m^2)$ per phase and hence total time $O(m^3)$. In [15], this was improved to time $O(m^\omega)$.

The idea of our algorithm is that of finding in step (3) the minimum weight cycle C_i with $\langle C_i, S_i \rangle \neq 0$ by means of a Monte Carlo binary search on the set of isometric cycles \mathcal{S} . The method improves the total time for step (3) to $o(m^\omega)$, leading to a $O(m^\omega)$ Monte Carlo algorithm.

After computing the all-pairs shortest paths in $O(nm + n^2 \log n)$, we detect isometric cycles and the corresponding set \mathcal{S} with the $O(nm)$ procedure presented in Section 2.2. Then we sort the $|\mathcal{S}| \leq nm$ isometric cycles by nondecreasing weight in time $O(nm \log n)$. Note that all this is dominated by steps (3) to (5) of de Pina's algorithm, that are the bottlenecks.

We now describe how to perform the Monte Carlo binary search. We put a binary tree (of depth at most $\log(nm)$, that is $O(\log n)$) on top of the sorted list of the isometric cycles, so that each isometric cycle is under a leaf.

If we randomly pick a coefficient $\lambda_C \in GF(p)$ for each isometric cycle C , for every non-leaf node of the binary tree we can compute a random linear combination $D = \sum_C \lambda_C C$ of the isometric cycles C under the subtree rooted in that node. For a given set of coefficients $\lambda_C \in GF(p)$, a random linear combination in a non-leaf node is equal to the sum of the random linear combinations of its two children, if they are both non-leaf nodes. For leaf children the corresponding cycle is multiplied by its coefficient. Starting from the leaves to the root, the cost of computing the random linear combinations for all non-leaf nodes at each level is proportional to the length of the vectors in the level below. By Property 2.5, for the lowest level, if it contains all the leaves, the total length is $O(nm)$. Since for the upper levels this length does not increase, the cost for obtaining one random linear combination for each node of the tree is $O(nm \log n)$. Given k independent sets of coefficients $\lambda_C \in GF(p)$ for every isometric cycle C , the corresponding k random linear combinations for each non-leaf node can be computed in $O(knm \log n)$.

In order to find the minimum weight cycle that has a nonzero inner product with S_i , corresponding to the leftmost leaf of the binary tree, we perform binary search on the sorted list using the binary tree. We start from the root and end at the leaf of the selected cycle. For each non-leaf node of the tree that we meet, we have to decide whether to go to the left child L or to the right child R . If L is a leaf and C is its corresponding cycle, we simply test if $\langle C, S_i \rangle \neq 0$. With a positive answer we end at L , and at R otherwise. If L is not a leaf, we call D_1, \dots, D_k its random linear combinations that have been previously computed. We proceed to L if $\exists i, 1 \leq i \leq k$ such that $\langle D_i, S_i \rangle \neq 0$, and to R otherwise. Note that the random linear combinations must be computed only in the non-leaf nodes that are left children of another node, but for sake of simplicity in the previous analysis we have considered all nodes. An example is shown in Figure 7.

A path in the binary tree consists of $O(\log n)$ decisions. Each decision consists of k inner products of vectors of length m , hence, has cost $O(km)$. Thus, selecting one C_i takes time $O(km \log n)$ and selecting all v cycles takes times $O(km^2 \log n)$. This dominates the time for preparing the random linear combinations and hence is the total complexity of step (3).

For the analysis of the error probability, we need a technical result.

Lemma 3.1 *Let \mathcal{C} be a collection of cycles. For each cycle $C \in \mathcal{C}$, let $\lambda_C \in GF(p)$ be chosen randomly and let $D = \sum_{C \in \mathcal{C}} \lambda_C C$. Let S be a nonzero vector in $GF(p)^E$. If all cycles in \mathcal{C} are orthogonal to S , also D is orthogonal to S . If \mathcal{C} contains a cycle that is non-orthogonal to S , D is orthogonal to S with probability at most $1/p$.*

Proof: Clearly, if every cycle in \mathcal{C} is orthogonal to S , then also D is.

So assume that $C' \in \mathcal{C}$ is non-orthogonal to S and consider an arbitrary but fixed choice of coefficients λ_C for the cycles $C \in \mathcal{C}$, $C \neq C'$. Also assume that there are two distinct choices α and β for $\lambda_{C'}$ such that $\sum_{C \in \mathcal{C}} \lambda_C C$ are orthogonal to S . Then $\alpha C' + \sum_{C \in \mathcal{C}, C \neq C'} \lambda_C C$ and $\beta C' + \sum_{C \in \mathcal{C}, C \neq C'} \lambda_C C$ are orthogonal to S . Thus $(\beta - \alpha)C'$ is orthogonal to S , which is a contradiction. Thus, the probability that $\langle D, S \rangle = 0$ is at most $1/p$. ■

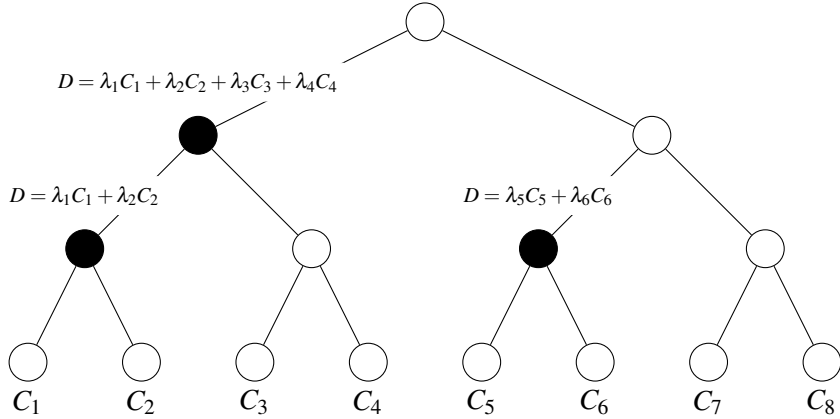


Figure 7: A binary tree for the Monte Carlo algorithm on a set of $|\mathcal{S}| = 8$ isometric cycles. The bold nodes are the non-leaf nodes for which we really need to compute the k random linear combinations of the cycles below them. Given a random choice of coefficients $\lambda_i \in GF(p)$, $1 \leq i \leq 8$, the corresponding expression is reported for each one of these nodes.

Let C_i be the minimum weight cycle with $\langle C_i, S_i \rangle \neq 0$. If our Monte Carlo binary search fails to find C_i , there must be a node where a wrong decision is made, i.e., the path to C_i goes to the left child while the right child is chosen. Thus by the preceding lemma, the probability that the Monte Carlo search takes a wrong decision in a node is at most p^{-k} . Since there are only $\log(nm)$ choices for q , the total failure probability is bounded by $\log(nm)p^{-k}$. Thus the Monte Carlo search correctly identifies C_i with probability³ at least $1 - \log(nm)p^{-k}$.

Since the total error of the v searches is bounded by the sum of the errors in a single search, i.e., $v \log(nm)p^{-k}$, we have the following result.

Theorem 3.2 *For any integer $k \geq 1$, the above Monte Carlo algorithm finds a minimum $GF(p)$ -basis in $O(m^\omega + km^2 \log n)$ time and errs with probability at most $v \log(nm)p^{-k}$. For $\omega > 2$ and $k = m^{(\omega-2)/2}$, the error probability is exponentially small, and the running time is $O(m^\omega)$.*

Specializing to $p = 2$ yields:

Corollary 3.3 *For any integer $k \geq 1$, the above Monte Carlo algorithm finds a minimum undirected cycle basis in $O(m^\omega + km^2 \log n)$ time and errs with probability at most $v \log(nm)p^{-k}$. For $\omega > 2$ and $k = m^{(\omega-2)/2}$, the error probability is exponentially small, and the running time is $O(m^\omega)$.*

For directed cycle bases we use an observation in [13, Section 3.5], namely that a minimum $GF(p)$ -basis for a random p with $p = \Theta(m \log m)$ is a minimum directed basis with probability at least $15/16$.

Theorem 3.4 *For any integer $L \geq 1$, there is a Monte Carlo algorithm that finds a minimum directed cycle basis in $O(L(m^\omega + m^2 \log^2 n))$ time and errs with probability at most $(3/4)^L$.*

Proof: Choose $2L + 1$ random primes p_i , $1 \leq i \leq 2L + 1$ with $p_i = \Theta(m \log n)$ and fix k such that $v \log(nm)p_i^{-k} \leq 1/16$; k can be chosen in $O(\log n)$. For every i , compute a minimum $GF(p_i)$ basis

³A more detailed analysis is possible but it is omitted for the sake of simplicity.

according to Theorem 3.2. If at least $L + 1$ runs yield bases of the same cost, return a basis of this cost. Otherwise, return any basis.

If this algorithm gives an incorrect answer, at least $L + 1$ runs did not return a minimum directed cycle basis. For any p_i , the probability that either the minimum $GF(p_i)$ basis is not a minimum directed basis or the algorithm according to Theorem 3.2 does not compute a minimum $GF(p_i)$ basis is bounded by $1/16 + 1/16 = 1/8$. The probability that the algorithm returns the wrong answer is therefore bounded by

$$\begin{aligned} \sum_{i \geq L+1} \binom{2L+1}{i} \left(\frac{1}{8}\right)^i \left(\frac{7}{8}\right)^{2L+1-i} &\leq \left(\frac{1}{8}\right)^{L+1} \sum_{i \geq L+1} \left(\frac{e(2L+1)7}{(2L+1-i)8}\right)^{2L+1-i} \\ &\leq \left(\frac{1}{8}\right)^{L+1} \sum_{i \geq L+1} (2e)^{2L+1-i} \\ &\leq \left(\frac{1}{8}\right)^{L+1} (2e)^{L+1} \leq \left(\frac{3}{4}\right)^{L+1}. \end{aligned}$$

■

The paper [19, Theorem 13] discusses the question of whether the above algorithm can be made certifying. It is shown that the algorithm can be modified so as to operate in the time bounds and error bounds stated above and so as to compute a witness in addition to the minimum cycle basis. There is a randomized algorithm, called the checker, with the following properties. It operates in time $O(m^2)$. If the basis is not a minimum cycle basis, the checker rejects with high probability for any witness. If the basis is a minimum cycle basis then the basis together with the witness is accepted with high probability.

4 Improved deterministic algorithm for planar graphs

In [10] Hartvigsen and Mardon have shown that in planar graphs minimum undirected cycle bases can be found in $O(n^2 \log n)$. In this section, we improve the running time to $O(n^2)$ by restriction to isometric cycles and we show that the algorithm is also valid for more specialized classes of cycle bases.

Let G be a plane graph, a planar graph embedded into the plane. Any elementary cycle C of G divides the plane into two maximal open connected sets of points, one bounded and one unbounded. We use $interior(C)$ to denote the bounded set. If $interior(C)$ agrees with one of the faces of G , we call C a *face cycle*. Note that the number of edges and the number of face cycles are both $O(n)$. A collection of cycles is called *nested* if for any two cycles in the collection, the interiors are either disjoint or the interior of one is contained in the interior of the other.

For a collection B of cycles, let F_B be the face cycles that do not belong to B . We define the directed inclusion graph D_B with node set $B \cup F_B$ as follows. Let C and C' be cycles in $B \cup F_B$. We have an edge from C to C' if $interior(C) \supset interior(C')$ and there is no cycle $C'' \in B \cup F_B$ such that $interior(C) \supset interior(C'') \supset interior(C')$. The inclusion graph is acyclic; the nodes of D_B with no outgoing edges are precisely the face cycles of G . The inclusion graph is a forest if and only if B is nested.

In [10] Hartvigsen and Mardon show that the number of isometric cycles is at most twice the number of face cycles of any planar graph G and that there exists a minimum cycle basis that is

nested. Moreover, a nested collection of cycles B is a minimum cycle basis if and only if B is a minimum weight collection of cycles satisfying three properties:

- (1) every non-leaf in D_B has exactly one child in F_B ,
- (2) the cycles in F_B have parents in D_B ,
- (3) the inclusion graph D_B is a forest.

Although in [10] these results and the following algorithm are formulated for undirected cycle bases, it is easy to see that they also hold for directed cycle bases.

Our algorithm for finding a minimum cycle basis differs from that of [10] in two regards. First, we use the all-pairs shortest paths method for planar graph in $O(n^2)$ proposed in [6]. Then, the main improvement is to exploit the procedure implied by Theorem 2.8 to obtain the set of isometric cycles in $O(n^2)$. Thus, the bottleneck of $O(n^2 \log n)$ decreases to $O(n^2)$. The rest of the algorithm proceeds as in [10] but restricted to isometric cycles and we summarize it below for completeness. Recall that there are $O(n)$ isometric cycles and that sorting them by nondecreasing weight takes $O(n \log n)$.

We construct the incidence matrix A between isometric cycles and the faces of G . The entry corresponding to a cycle C and a face R is 1 if $R \subseteq \text{interior}(C)$. This matrix can clearly be computed in time $O(n^2)$.

We initialize the basis B to the empty set and set up the corresponding inclusion graph D_B . The nodes of D_B are the face cycles and there are no edges. As long as B does not have the right number of cycles and hence D_B does not satisfy Properties (1) and (2), we do the following.

If there is a non-leaf node C that has more than one child in F_B (case 1), let R_1 and R_2 be two faces of G limited by two face cycles in F_B having C as their common parent. If there is no such non-leaf node, there must be a face cycle in F_B without a parent (case 2). Let R_1 be the face limited by this face cycle and let R_2 be the unbounded face. In either case, we find the minimum weight cycle D containing exactly one of R_1 or R_2 in its interior. We can find D in time $O(n)$ by scanning the columns of A .

We add D to B and update D_B . If D is a face cycle, we only have to remove D from F_B . The inclusion graph remains unchanged. So assume that D is not a face cycle. Starting from the face cycles in $\text{interior}(D)$ (we can find them by inspecting matrix A), we determine the maximal subtrees of D_B that are contained in $\text{interior}(D)$. They become children of D . D either becomes a root (in case 2) or a child of C (in case 1). Updating D_B takes time $O(n)$.

Since it takes $O(n)$ for each cycle of the basis, we have the following result.

Theorem 4.1 *A minimum (directed or undirected) cycle basis of a planar graph can be found in time $O(n^2)$.*

As observed in [10] the minimum cycle basis problem is dual to the all-pairs minimum cut problem. Hence in planar graphs the all-pairs minimum cut problem can also be solved in time $O(n^2)$.

We will now show that in planar graphs a (directed or undirected) cycle basis that is both minimum and nested also belongs to three interesting subclasses of the set of all cycle bases. For a cycle basis B in a graph G we have (see e.g. [18, 13, Section 3]) that:

- B is *integral*, if each cycle $C \in G$ can be written as an integer linear combination of cycles C_1, \dots, C_ν of B , i.e.,

$$\exists \lambda_i \in \mathbb{Z} \text{ such that } C = \lambda_1 C_1 + \dots + \lambda_\nu C_\nu;$$

- B is *totally unimodular*, if each cycle $C \in G$ can be written as a linear combination with coefficients in $\{-1, 0, 1\}$ of cycles C_1, \dots, C_ν of B , i.e.,

$$\exists \lambda_i \in \{-1, 0, 1\} \text{ such that } C = \lambda_1 C_1 + \dots + \lambda_\nu C_\nu;$$

- B is *weakly fundamental*, if there exists an ordering C_1, \dots, C_ν of the cycles in B such that

$$C_i \setminus (C_1 \cup \dots \cup C_{i-1}) \neq \emptyset, \quad i = 2, \dots, \nu.$$

Theorem 4.2 *For any planar graph G , a minimum nested cycle basis is also integral, totally unimodular, and weakly fundamental.*

Proof: Let B be any minimum nested cycle basis of G . To show that B is totally unimodular, we need to verify that any cycle is a linear combination of the cycles in B with coefficients in $\{-1, 0, +1\}$. Any cycle C can be obtained as the sum of the face cycles that limit the faces in $\text{interior}(C)$. A face cycle either belongs to B or is equal to the difference of its parent $p(F)$ in D_B and the sum of the other children of $p(F)$ in D_B . Let us denote set of siblings of F as $s(F) = \{D \in B : D \text{ is a child of } p(F) \text{ in } D_B \text{ and } D \neq F\}$. Therefore,

$$C = \sum_{F \in B} F + \sum_{F \in F_B} \left(p(F) - \sum_{D \in s(F)} D \right).$$

If a cycle D occurs twice in the representation of C , it occurs once as a parent and once as a child. As a parent, its coefficient is $+1$, and as a child, its coefficient is -1 and hence the two occurrences cancel. Thus every cycle is a linear combination of the cycles in B with coefficients in $\{-1, 0, +1\}$.

We next show that B is weakly fundamental. We need to exhibit an ordering C_1, \dots, C_ν of the cycles in B such that $C_i \setminus (C_1 \cup \dots \cup C_{i-1}) \neq \emptyset$ for all i . Let D_B be the inclusion graph corresponding to B . If F_B is empty, every face cycle belongs to B . We determine a reverse ordering of the cycles C_ν, \dots, C_1 as described in [18]. Starting from the cycle C that limits the unbounded face, we add the face cycles with an edge in common with C . After removing the edges of C from G , we proceed in the same way. We now extend the previous result to the general case where F_B is not empty. Since every face cycle in F_B has a parent, we have a non-leaf node D in D_B whose children are all face cycles. One of these face cycles, say F , belongs to F_B and all the others belong to B . The same idea for constructing a reverse ordering is then applied to the cycles in B corresponding to the children of D starting from F . The face cycles among these with an edge in common with F are added and the edges of F that are not in D are removed. Then we proceed in the same way considering the cycle that limits the new face. Once all the children of D are added, they are deleted from D_B . This step is repeated until all nodes in D_B are isolated. The remaining graph only contains face cycles and we can apply the procedure in [18] to obtain a complete reverse ordering of the cycles.

Finally, unimodularity clearly implies integrality. ■

Since our algorithm yields a minimum nested cycle basis, also minimum integral, totally unimodular and weakly fundamental cycle bases can be found in $O(n^2)$.

5 Conclusion

We have investigated isometric cycles, which are special cases of Horton candidate cycles. In particular, we have proved that the total number of edges of all isometric cycles is at most $n\nu$ and we have

designed an efficient $O(nm)$ procedure for detecting them. By restricting attention to isometric cycles and exploiting these properties, we have devised improved algorithms for finding minimum cycle bases in directed and undirected graphs. For general graphs we have proposed $O(m^{\omega})$ randomized algorithms and for planar graphs a deterministic $O(n^2)$ algorithm that is also valid for more specialized classes of cycle bases. An open question is whether we can do away with the maintenance of a basis of the orthogonal subspace.

References

- [1] E. Amaldi, C. Iuliano, T. Jurkiewicz, K. Mehlhorn, and R. Rizzi. Breaking the $O(m^2n)$ barrier for minimum cycle bases. In A. Fiat and P. Sanders, editors, *European Symposium on Algorithms (ESA)*, volume 5757 of *Lecture Notes in Computer Science*, pages 301–312. Springer, 2009.
- [2] E. Amaldi, C. Iuliano, and R. Rizzi. Efficient deterministic algorithms for finding a minimum cycle basis in undirected graphs. In F. Eisenbrand and F. B. Shepherd, editors, *IPCO*, volume 6080 of *Lecture Notes in Computer Science*, pages 397–410. Springer, 2010.
- [3] F. Berger, P. Gritzmann, and S. de Vries. Minimum cycle bases for network graphs. *Algorithmica*, 40(1):51–62, 2004.
- [4] B. Bollobas. *Modern Graph Theory*. Volume 184 of Graduate Texts in Mathematics. Springer. 2nd printing, 2002.
- [5] J. C. De Pina. *Applications of shortest path methods*. PhD thesis, University of Amsterdam, 1995.
- [6] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Computing*, 16(6):1004–1022, 1987.
- [7] P. M. Gleiss. *Short cycles: minimum cycle bases of graphs from chemistry and biochemistry*. PhD thesis, Universität Wien, Austria, 2001.
- [8] A. Golynski and J. D. Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *SWAT 2002: Proceedings of the 8th Scandinavian Workshop on Algorithm Theory*, volume 2368 of *LNCS*, pages 200–209, 2002.
- [9] R. Hariharan, T. Kavitha, and K. Mehlhorn. Faster deterministic and randomized algorithms for minimum cycle basis in directed graphs. *SIAM J. Computing*, 38(4):1430–1447, 2008.
- [10] D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *SIAM J. Discrete Math.*, 7(3):403–418, 1994.
- [11] J. D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Computing*, 16(2):358–366, 1987.
- [12] T. Kavitha. An $\tilde{O}(m^2n)$ randomized algorithm to compute a minimum cycle basis of a directed graph. In *ICALP*, volume 3580 of *LNCS*, pages 273–284, 2005.
- [13] T. Kavitha, C. Liebchen, K. Mehlhorn, D. Michail, R. Rizzi, T. Ueckerdt, and K. A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3(4):199 – 243, 2009.

- [14] T. Kavitha and K. Mehlhorn. Algorithms to compute minimum cycle basis in directed graphs. *Theory of Computing Systems*, 40(4):485–505, 2007.
- [15] T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch. An $\tilde{O}(m^2n)$ algorithm for minimum cycle basis of graphs. *Algorithmica*, 52(3):333–349, 2008.
- [16] K. A. Lehmann and S. Kottler. Visualizing large and complex networks. In *Proceedings of the 14th International Symposium on Graph Drawing (GD'06)*, 2007.
- [17] C. Liebchen and R. Rizzi. A greedy approach to compute a minimum cycle basis of a directed graph. *Information Processing Letters*, 94(3):107–112, 2005.
- [18] C. Liebchen and R. Rizzi. Classes of cycle bases. *Discrete Applied Mathematics*, 155(3):337–355, 2007.
- [19] R. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, 2011.
- [20] K. Mehlhorn and D. Michail. Minimum Cycle Bases: Faster and Simpler. *ACM Transactions on Algorithms*, 6(1), 2009.
- [21] G. Tewari, C. Gotsman, and S. J. Gortler. Meshing genus-1 point clouds using discrete one-forms. *Comput. Graph.*, 30:917–926, December 2006.
- [22] C. Wulff-Nilsen. Minimum cycle basis and all-pairs min cut of a planar graph in subquadratic time. *CoRR*, abs/0912.1208, December 2009. Informal publication.