# Minimum Cycle Bases: Faster and Simpler

Kurt Mehlhorn Dimitrios Michail

Max-Planck-Institut für Informatik
Saarbrücken, Germany

**Abstract**

We consider the problem of computing exact or approximate minimum cycle bases of an undirected (or directed) edge-weighted graph $G$ with $m$ edges and $n$ vertices. In this problem, a $\{0,1\}$ ($\{-1,0,1\}$) incidence vector is associated with each cycle and the vector space over $\mathbb{F}_2$ ($\mathbb{Q}$) generated by these vectors is the cycle space of $G$. A set of cycles is called a cycle basis of $G$ if it forms a basis for its cycle space. A cycle basis where the sum of the weights of the cycles is minimum is called a minimum cycle basis of $G$. Cycle bases of low weight are useful in a number of contexts, e.g. the analysis of electrical networks, structural engineering, chemistry, and surface reconstruction.

There exists a set of $\Theta(mn)$ cycles which is guarantied to contain a minimum cycle basis. A minimum basis can be extracted by Gaussian elimination. The resulting algorithm [Horton 1987] was the first polynomial time algorithm. Faster and more complicated algorithms have been found since then.

We present a very simple method for extracting a minimum cycle basis from the candidate set, which improves the running time for sparse graphs. Furthermore, in the undirected case by using bit-packing we improve the running time also in the case of dense graphs. Our results improve the running times of both exact and approximate algorithms. Finally, we derive a smaller candidate set with size in $\Omega(m) \cap O(mn)$.

## 1 Introduction

Let $G = (V, E)$ be an undirected graph with $m$ edges and $n$ vertices. A *cycle* of $G$ is any subgraph of $G$ in which each vertex has even degree. Associated with each cycle $C$ is an *incidence vector* $x$, indexed by $E$, where for any $e \in E$, $x_e$ is 1 if $e$ is an edge of $C$ and 0 otherwise. The vector space over $\mathbb{F}_2$ generated by the incidence vectors of cycles is called the *cycle space* of $G$. We assume w.l.o.g. that $G$ is connected. It is well known that this vector space has dimension $N = m - n + 1$, where $m$ is the number of edges of $G$ and $n$ is the number of vertices. A maximal set of linearly independent cycles is called a *cycle basis*. The edges of $G$ have non-negative weights assigned to them. A cycle basis where

the sum of the weights of the cycles is minimum is called a *minimum cycle basis* of $G$. We use the abbreviation MCB to refer to a minimum cycle basis.

It is well known that the set of cycles of a graph form a matroid. Thus, the MCB problem is solvable by the greedy algorithm. However, since a graph may have an exponential number of cycles, further ideas are required for a polynomial time algorithm. See the paper of Lee and Ryan [14] concerning algorithmic aspects of matroid theory. Moreover, the books by Deo [7] and Bollobás [4] contain an in-depth coverage of cycle bases.

One of the most important areas of application of the MCB problem is electric networks [18, 6, 3]. Many problems arising in the design and analysis of electric networks can be formulated in graph-theoretic terms. In fact, in the analysis of complex electric networks by graph theoretical methods, a basic problem is to determine the solvability of the "network equation", a system of algebraic differential equations that describes the relation of currents and voltages in a network as functions of time. In order to check structural solvability of that system quickly by a heuristic matching approach, fast algorithms are needed that compute sparse representations. The equations corresponding to the Kirchhoff voltage law are critical, since for the remaining equations, a sparse representation is readily available. Hence the central problem is that of computing a sparse cycle basis to describe the "voltage law" part of the system.

Other applications are in structural engineering [5], chemistry and biochemistry [8, 15], and surface reconstruction from point clouds [19]. In most applications, the computation of an MCB is a preprocessing step. The use of an MCB ensures sparseness and translates into faster running times of the main algorithm.

There has been a considerable amount of work concerning minimum cycle bases. An early paper is by Stepanec [17]. Horton [11] presented the first polynomial time algorithm. Faster and/or alternative algorithms were later presented by de Pina [6], Golynski and Horton [9], Berger et al. [3], and Kavitha et al. [13]. The current fastest algorithm [13] has running time $O(m^2n+mn^2 \log n)$. This assumes fast matrix multiplication. Without fast matrix multiplication the fastest running time is $O(m^3 + mn^2 \log n)$ [6].

In the case of directed graphs (base field $\mathbb{Q}$ instead of $\mathbb{F}_2$) the methods used are similar. However, the different base field introduces arithmetic complications since the numbers handled can grow large. Moreover, directed cycle bases do not necessarily project onto undirected cycle bases and thus extra ideas are required. The best running times are $O(m^3n + m^2n^2 \log n)$ deterministic and $O(m^2n + mn^2 \log n)$ randomized [10].

**Our results.** We present faster algorithms in both undirected and directed graphs. For undirected graphs we derive an $O(m^2n/\log n + n^2m)$ algorithm, which is by a logarithmic factor faster for all graph densities. For directed graphs we get an $O(m^3n)$ deterministic and an $O(m^2n)$ randomized algorithm. This is a logarithmic improvement for sparse graphs. Also, our algorithm is simpler and requires only $O(n)$ shortest path computations (instead of $O(nm)$

as in [6, 13]). This may lead to further improvements.

We also present faster approximation algorithms for the MCB problem. We give a 2-approximation with an $O(m^2\sqrt{n/\log n} + n^2m + m^\omega)$ running time. This improves by a factor of $\log^{3/2} n$ the best known for relatively dense graphs. Moreover, we give an improved algorithm which computes a $(2k-1)$-approximate MCB, for any integer $k > 1$, of an undirected graph with non-negative edge weights in time $O(n^{3+2/k}/\log n + n^{3+1/k})$. This is again a logarithmic improvement.

The structure of the paper is as follows. In Section 2 we give some preliminaries and discuss previous work and algorithms. There is a set of $\Theta(nm)$ candidate cycles which is guarantied to contain an MCB; it can be constructed by $n$ shortest path computations. In Section 3 we present a subset of this set which is still guarantied to contain an MCB. The new candidate set can still have size $\Omega(mn)$; however, it should lead to an improvement in practice. Extracting the correct cycles from these sets is the difficult part. In Section 4 we present a new simple and efficient way to extract these cycles. All these results are presented w.r.t undirected graphs. In Section 4.2 we extend them to directed graphs. Finally we show how to use these results in order to obtain approximate MCBs in Section 5 and conclude with some open problems in Section 6.

# 2 Preliminaries and Related Work

Let $T$ be any spanning tree in $G(V, E)$, let $e_1, \ldots, e_N$ be the edges of $E \setminus T$ in some arbitrary but fixed order, and let $e_{N+1}, \ldots, e_m$ be the edges in $T$ in some arbitrary but fixed order. We frequently view cycles in terms of restricted incidence vectors, that is, each cycle is a vector in $\{0,1\}^N$. It is easy to see that linear independence of the restricted incidence vectors implies linear dependence of the full incidence vectors and vice versa. Thus, we may restrict attention to the restricted incidence vectors when discussing questions of linear independence.

We use $S$ to denote subsets of $E \setminus T$. Each such subset gives rise to an incidence vector in $\{0,1\}^N$. We use $\langle C, S\rangle$ to denote the standard inner product of vectors $C$ and $S$. We say that a vector $S$ is *orthogonal* to $C$ if $\langle C, S\rangle = 0$. In the field $\mathbb{F}_2$, $\langle C, S\rangle = 1$ if and only if $C$ contains an odd number of edges of $S$.

For a cycle $C$, we use $w(C) = \sum_{e\in C} w(e)$ to denote its weight. We use $w_G(\text{MCB})$ to denote the weight of a minimum cycle basis of graph $G$. When it is clear by the context we omit $G$ and write $w(\text{MCB})$.

## 2.1 Previous algorithms

We briefly review previous work and algorithms.

The first approach is due to Horton [11], later improved by Golynski and Horton [9]. The running time of their algorithm is $O(m^\omega n)$. Horton proved that a set of $O(mn)$ cycles contains an MCB. An MCB can then be found by determining the least weight $N = m - n + 1$ linearly independent cycles from

this set, using Gaussian elimination. The set can be described as follows: For a vertex $v \in V$ and edge $e \in E$, let $C[v, e]$ be the cycle consisting of $e$ and the shortest paths from $v$ to the endpoints of $e$ in $G \setminus e$. Horton's collection, denoted by $\mathcal{H}$, contains cycles $C[v, e]$ for all $v \in V$ and $e \in E$.

The second approach, due to de Pina [6], was further improved in [13] to reach a time bound of $O(m^2n + mn^2 \log n)$. This is faster than the collection approach. In these algorithms the cycles of an MCB are computed sequentially. Assume that $i - 1$ cycles $C_1, C_2, \ldots, C_{i-1}$ of an MCB are already known. In order to compute cycle $C_i$ we first compute a non-zero vector $S_i \in \{0, 1\}^N$, called a *support vector*, s.t. $\langle C_j, S_i \rangle = 0$ for all $1 \leq j < i$. Then cycle $C_i$ is the shortest cycle $C$ in the graph $G$ s.t. $\langle C, S_i \rangle = 1$. The fact that $C_i$ is not orthogonal to $S_i$ ensures linear independence, the shortest cycle computation ensures the optimality of the resulting cycle basis.

The algorithm operates in $N \leq m$ phases, one for each cycle of the MCB. Computing all necessary support vectors can be performed in $O(m^\omega)$ time. Computing each cycle is performed by a reduction to $n$ single source shortest path computations in an appropriate graph $G(S_i)$, which is different in each phase $1 \leq i \leq N$. Thus, we need $O(n(m+n \log n))$ for each cycle using Dijkstra's algorithm, for a total of $O(m^2n + mn^2 \log n)$. This time bound dominates the $O(m^\omega)$ bound.

# 3 A Reduced Collection

In this section we define a collection of cycles $\mathcal{R}$, which is a subset of Horton's collection, and still contains an MCB. Asymptotically $\mathcal{R}$ does not improve on the size of $\mathcal{H}$ but in practice should be smaller.

The graph has non-negative edge weights. We can deal with length zero edges in a pre- and post-processing step. Contract length zero edges and compute a minimum cycle basis in the contracted graph. For any connected component of length zero edges choose a spanning tree. Lift the MCB to the original graph by inserting appropriate paths of length zero edges, i.e., if $u$ and $v$ are contracted into the same node and a cycle uses edges incident to $u$ and $v$, then fill the gap with a path of length zero edges. Also for each length zero edge which does not belong to the spanning tree, add a length zero cycle. In this way, we obtain an MCB of the original graph.

So, we may assume that all edge weights are positive. Let now $Z$ be a *feedback vertex set*. A feedback vertex set is a set of vertices covering all cycles, i.e., every cycle in the graph passes through a node in $Z$. Such a set can be found for example by a greedy approach and $Z = V$ certainly works. Computing the minimum feedback vertex set is known to be APX-hard. For undirected graphs a 2-approximation can be computed efficiently [1]. For each $v \in V$ let $T_v$ be a shortest path tree rooted at $v$. We define the reduced Horton collection $\mathcal{R}$ to be all cycles $C[z, e]$ such that $z \in Z$ and the endpoints of $e$ lie in different subtrees of $T_z$ (in other words the least common ancestor of the endpoints of $e$ is the root $z$).

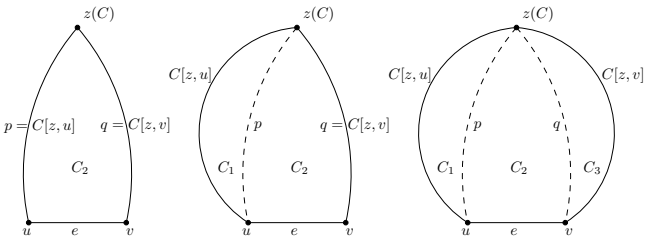


Figure 1: The three cases in the proof of Lemma 3.1 (not showing symmetrical cases).

Let $C$ be any cycle and let $z(C) \in Z \cap C$ be a vertex which minimizes the number of non-tree edges of $C$ w.r.t $T_z$. We call $z(C)$ the base node of $C$.

**Lemma 3.1.** *$\mathcal{R}$ contains a minimum cycle basis.*

*Proof.* Consider the greedy algorithm run on the set of all cycles. Cycles are ordered lexicographically according to

$$(\text{weight of } C, \text{number of edges outside } T_{z(C)}, \text{number of edges in } C) \ .$$

Observe that the cycles in $\mathcal{R}$ have second coordinate equal to one and hence come first among cycles of equal weight.

Let $C$ be the first cycle that is selected by the greedy algorithm and that does not belong to $\mathcal{R}$. Let $z = z(C)$ and let $e = (u, v)$ be a non-tree edge on $C$. Let $p$ and $q$ be the tree path in $T_z$ connecting $z$ to $u$ and $v$, respectively. The cost of $p$ is at most the cost of either cycle path from $z$ to $u$ and the cost of $q$ is at most the cost of either cycle path from $z$ to $v$.

Consider the cycles $C_1 = C[z, u] \circ p$, $C_2 = p \circ e \circ q$, $C_3 = q \circ C[z, v]$. The weight of $C_1$, $C_2$ and $C_3$ is at most the weight of $C$.

We now distinguish cases (see Figure 3). Assume first that $e$ is the only non-tree edge on $C$. Then $C_1$ and $C_3$ are trivial cycles since $p = C[z, u]$ and $q = C[z, v]$. If $u$ and $v$ lie in distinct subtrees of $T_z$, then $C \in \mathcal{R}$. This is a contradiction to the choice of $C$. So assume that $u$ and $v$ belong to the same subtree. Let $x$ be their least common ancestor and let $C'$ be the cycle consisting of $e$ and the tree paths (in $T_z$) from $x$ to $v$ and $u$, respectively. Observe that these are not necessarily tree path in $T_x$ nor do we necessarily have $x \in Z$. Since all edges have positive weight $w(C') < w(C)$ and hence $C'$ is considered before $C$. Also $C = C'$ as cycles; it is only the representation which is different. So $C$ is not the first cycle selected by the greedy algorithm that does not belong to $\mathcal{R}$.

Assume next that $C$ contains more than one non-tree edge. In this case at least one of the cycles $C_1$ and $C_3$ is non-trivial, and thus either $C = C_1 + C_2 + C_3$, or $C = C_1 + C_2$, or $C = C_2 + C_3$. In either case with respect to $z$ all cycles have at least one fewer non-tree edge than $C$ and hence this is also true with respect to their respective base vertices.

Thus all these cycles are before $C$ in the ordering. Also, at least one of them is independent of the current basis. So it was independent at the time is was considered and hence should have been added. This either contradicts our definition of $C$ (first cycle outside $\mathcal{R}$ added to the basis) or the operation of the greedy algorithm (a cycle not added although it is independent). □

Since $\mathcal{R} \subseteq \mathcal{H}$, Horton's collection also contains an MCB. For an alternative proof see [11]. We use $\mathcal{A}$ to denote the candidate set. If $\mathcal{A} = \mathcal{R}$, $Z$ denotes a node set covering all cycles, if $\mathcal{A} = \mathcal{H}$, $Z = V$.

# 4 The New Algorithm

The algorithm proceeds as follows. Assume that we have cycles $C_1, \ldots, C_{i-1}$ of an MCB and a non-trivial support vector $S_i \in \{0,1\}^N$ s.t. $\langle C_j, S_i \rangle = 0$ for all $1 \leq j \leq i-1$. In order to obtain $C_i$ we look for the shortest cycle $C \in \mathcal{A}$ s.t. $\langle C, S_i \rangle = 1$. We do not describe here how to compute $S_i$ in each phase, the interested reader is referred to [13]. We next show that the algorithm computes an MCB. The proof is almost identical with the proof of de Pina's original algorithm, the only difference being that we search for cycles only in set $\mathcal{A}$.

**Theorem 4.1.** *The above algorithm returns an MCB.*

*Proof.* Suppose not and let $C_1, \ldots, C_{N'}$ be the set of cycles returned by the algorithm. Then, there exists an $i$, $0 \leq i \leq N'$ such that there exists a minimum cycle basis $\mathbb{B} \subseteq \mathcal{A}$ that contains $\{C_1, \ldots, C_i\}$ and either $i = N' < N$ or there is no MCB in $\mathcal{A}$ containing $\{C_1, \ldots, C_{i+1}\}$.

Let $\mathbb{B} = \{B_1, \ldots, B_N\}$. In the former case there is clearly a $B_j$ with $\langle B_j, S_{i+1} \rangle = 1$. Otherwise, we get that $S_{i+1} = \emptyset$ which is a contradiction. Thus, the algorithm finds a $C_{i+1}$ and the case cannot arise. In the latter case, there are cycles in $\mathbb{B}$ s.t. $C_{i+1} = B_1 + B_2 + \cdots + B_k$. We know that $\langle C_{i+1}, S_{i+1} \rangle = 1$ which implies that there exists a $1 \leq j \leq k$ s.t. $\langle B_j, S_{i+1} \rangle = 1$. Note that both $C_{i+1}$ and $B_j$ belong to $\mathcal{A}$ and by construction $C_{i+1}$ is the shortest cycle in $\mathcal{A}$ non-orthogonal to $S_{i+1}$. Thus, $w(C_{i+1}) \leq w(B_j)$.

Let $\mathbb{B}' = \mathbb{B} \cup \{C_{i+1}\} \setminus \{B_j\}$. Then, $w(\mathbb{B}') \leq w(\mathbb{B})$. We also claim that $\mathbb{B}'$ is a basis. Observe that $\langle C_{i+1}, S_{i+1} \rangle = 1$ while $\langle C_q, S_{i+1} \rangle = 0$ for all $1 \leq q \leq i$ which means that $B_j$ cannot be any of the cycles $C_1, \ldots, C_i$. Moreover, if $\mathbb{B}'$ is not a basis then $\mathbb{B}$ is also not a basis.

Thus, $\mathbb{B}'$ is an MCB in $\mathcal{A}$ s.t. $\{C_1, \ldots, C_i, C_{i+1}\} \subseteq \mathbb{B}'$, a contradiction. □

## 4.1 Cycles computation

Now that we established correctness we go on showing how to implement the search for such cycles. We begin by describing the previous approach and then go on to describe the new approach.

During phase $i$ we have a non-trivial vector $S_i$ and need to compute the shortest cycle in $G$ s.t. $\langle C, S_i \rangle = 1$. Such a cycle $C$ can be computed as follows.

**input** : Graph $G(V, E)$.
**output** : An MCB of $G$.

Construct shortest path trees $T_v$ for all $z \in Z$.
Construct candidate set $\mathcal{A}$ (sorted) in non-decreasing order of weight.

**for** $i = 1$ to $N$ **do**
Compute non-trivial vector $S_i \in \{0,1\}^N$ s.t $\langle C_j, S_i \rangle = 0$ for $1 \leq j < i$ (as described in [13]).
For all trees $T_v$, $v \in Z$ update vertex labels based on $S_i$.
For each cycle $C$ in the (sorted) candidate set $\mathcal{A}$, compute $\langle C, S_i \rangle$ using the labels.
Return the least-weight cycle non-orthogonal to $S_i$.

Figure 2: The new algorithm

We set up an auxiliary graph $G^\dagger$ with two copies, say $v'$ and $v''$, for each vertex $v$, and two copies $e'$ and $e''$ for each edge $e = (u, v) \in E$. If $e \in S_i$, the copies are $(u', v'')$ and $(u'', v')$ and if $e \notin S_i$, the copies are $(u', v')$ and $(u'', v'')$. Then a shortest cycle $C$ with $\langle C, S_i \rangle = 1$ corresponds to a shortest path connecting the two copies of some vertex $v$ minimized over all $v$. Such a path can be found by $n$ shortest path computations in the auxiliary graph. The time spend for this reduction is $O(n(m + n \log n))$ for each cycle.

The new computation is done by extracting the correct cycle from the candidate collection. We thus need to describe how we represent $\mathcal{A}$. For each vertex $v \in Z$ we run a single-source shortest path computation and store the resulting shortest path tree, call it $T_v$; and for each $w$ its distance $d(v, w)$ from $v$. Tree $T_v$ encodes at most $m$ cycles, namely cycles $C[v, e]$ for all $e \in E \setminus T_v$ s.t. the least common ancestor of the endpoints of $e$ is the root of $T_v$. Computation of the $|Z|$ trees requires $O(n(m + n \log n))$ time.

For practical purposes we can also do the following. After computing all trees we also compute the length of all cycles in time $O(nm)$. Finally we sort these $O(nm)$ cycles in non-decreasing weight in time $O(nm \log n)$. This sorting is not really necessary for our algorithm's correctness.

Consider now phase $i$. Given the support vector $S_i \in \{0,1\}^N$ and a tree $T_v$ we traverse the tree from the root to the leaves. For each node $w \in T_v$ we compute the label (with respect to $S_i$) $\ell_v(w)$. This label $\ell_v(w)$ is equal to $\langle p_v(w), S_i \rangle \in \{0,1\}$ where $p_v(w)$ denotes the path from $v$ to $w$ in $T_v$. This is done in $O(n)$ time for each tree and thus $O(n|Z|) \in O(n^2)$ for all trees.

Then we go over our sorted list of cycles. For a cycle $C[v, e]$ we find $T_v$ and the endpoints of $e = (u, w)$ in the tree. We assume that we can do this in constant time by storing reverse pointers to the trees. Using $\ell_v(u)$, $\ell_v(w)$ and whether $e$ belongs to $S_i$ we can compute in constant time the value of

$\langle C[v,e], S_i\rangle$. We traverse our sorted list until we find the first cycle $C$ with an odd intersection with $S_i$, i.e., $\langle C, S_i\rangle = 1$. This cycle is $C_i$. Thus, searching for the cycle requires $O(|Z|m) \in O(nm)$ time. Since the algorithm needs to compute $N \leq m$ cycles $C_1, \ldots, C_N$ we perform the above procedure at most $m$ times. This gives us an $O(m^2 n)$ algorithm. Observe that the new algorithm is not only faster, but also simpler.

We next describe an improvement of the above technique to time $O(nm2^b + m(n^2 + nm/b) + T)$ where $b$ is a parameter and $T = m^\omega$ is the time to compute the support vectors. Setting $b = \frac{1}{2}\log m$ results in an $O(m^2 n/\log n + mn^2)$ algorithm.

The idea behind this improvement is the following. Consider an edge $e = (u, v)$ and the cycles $C[v, e]$ for all $v \in Z$. We want to compute the inner product of these cycles with $S_i$ and find the shortest non-orthogonal cycle in time $O(n/\log n)$. We do this by forming two vectors, one for $u$ and one for $v$ which contain values $\ell_w(v)$ and $\ell_w(u)$ for all $w \in Z$. Moreover, we form a third vector which is either all ones or all zeroes depending on whether $e \in S_i$. All three vectors can be packed into words of logarithmic length and thus we can do a bitwise x-or in $O(n/\log n)$ time. Each entry of the result corresponds to a vertex $w$ of the graph and the value of the entry is the inner product of cycle $C[w, e]$ with $S_i$. However, among the cycles which are non-orthogonal with $S_i$ we need to find the one with minimum length. In order to do this in $o(n)$ we perform some preprocessing. We next describe the above idea in more detail.

Assuming a fixed numbering of the vertices, we divide the vertices into blocks of $b$ vertices each. We perform the following precomputation. Consider a fixed block $B$ of vertices. There are $2^b$ subsets of $B$. For each edge $e$ and each block $B$ we compute a table of size $2^b$ with one entry for each subset $A$ of $B$. Call this entry $i_{e,B}(A)$. We have

$$i_{e,B}(A) = v \quad \text{where } v \in A \text{ and } C[v,e] \text{ is the cheapest cycle among all cycles } \{C[z,e] \mid z \in A\}\ .$$

For each $e$ and $B$ we can compute the table in time $O(2^b)$, so the total time of precomputation is $O(mn2^b)$.

We now come to each phase $i$. In the first part, we compute a $|Z| \times n$ matrix. For each node $w$ and each tree $T_v$ we compute the label $\ell_v(w)$ of $w$ in $T_v$. Index the rows by $v$ and the columns by $w$. We next compute a compressed version of this matrix. We pack $b$ entries of each column in one word of size $b$. All of this takes time $O(n^2)$.

We next scan the edges one by one. Consider edge $e = (u, w)$. We scan the columns corresponding to $u$ and $w$ in parallel. For each block $B$, we compute the x-or of the corresponding block in column $u$, column $w$ and vector of all ones (if the label of $e$ is one, i.e., $\langle\{e\}, S_i\rangle = 1$) or the vector of all zeroes (if the label of $e$ is zero, i.e., $\langle\{e\}, S_i\rangle = 0$). This gives us a subset $A$ of $B$. It is the subset of vertices $v \in B$ such that $\langle C[v,e], S_i\rangle \neq 0$. We index the appropriate table with $A$ and obtain the best cycle. All of this takes constant time per block and hence time $O(nm/b)$ per phase.

**Theorem 4.2.** *A minimum cycle basis of an undirected graph with non-negative edge weights can be computed in time* $O(m^2n/\log n + n^2m)$.

The above theorem assumes a RAM model of computation which allows bitwise operations in constant time. In the context of our paper this assumption is no restriction because the linear algebra related part[1] of our algorithm requires constant time multiplication of numbers of logarithmic length.

## 4.2 Directed graphs

In directed graphs the situation is similar. A cycle in a directed graph is a cycle in the underlying undirected graph with edges traversable in both directions. A $\{-1, 0, 1\}$ edge incidence vector is associated with each cycle: edges traversed by the cycle in the right direction get 1 and edges traversed in the opposite direction get $-1$. The cycle space is the space generated by these cycle vectors over $\mathbb{Q}$. Note that the weight of a cycle is simply the sum of the weight of its edges, independent of the orientation of these edges.

Liebchen and Rizzi [16] showed that the directed version of $\mathcal{H}$ contains a directed MCB. Similarly, the approach with the support vectors computes an MCB. We use [10] as background material. In phase $i$ given a non-trivial support vector $S_i \in \mathbb{Z}^N$ s.t. $\langle C_j, S_i\rangle = 0$ for $1 \leq j \leq i-1$, the algorithm computes the shortest cycle $C$ in $G$ s.t. $\langle C, S_i\rangle \neq 0$. However, since the base field is no longer $\mathbb{F}_2$ the running times of deterministic algorithms suffer from extra cost in arithmetic. $S_i$ is now a vector with integer coordinates. The coordinates are as large as $N^N$ and also the scalar product can be as large as $N^N$. Computing the support vectors deterministically can be performed in $\tilde{O}(m^{\omega+1})$. Using randomization the required time is $O(m^\omega)$ for a 3/4 success probability. In the randomized algorithm one computes modulo a single prime of value about $N^2$ and in the deterministic algorithm one computes modulo $N$ such primes.

Based on the above, two algorithms have been designed, a deterministic with running time $O(m^3n + m^2n^2 \log n)$ and a randomized with $O(m^2n + mn^2 \log n)$. We show how to get rid off the logarithmic factors in sparse graphs.

The approach used in Section 4 can be used to compute the shortest cycle $C[v, e]$ with $\langle C[v, e], S_i\rangle \neq 0 \ (mod\ p)$ where $p$ is prime as stated above. The algorithm computes again the shortest path trees, using undirected single source shortest path computations. We then make the trees directed by introducing the directions of the edges of the graph $G$. During phase $i$ given $S_i$ we again process each tree. We traverse $T_v$ top-down and compute for each node $w \in T_v$ the label $\ell_v(w) = \langle p_v(w), S_i\rangle (mod\ p) \in \mathbb{Z}$ where $p_v(w)$ denotes the path from the root of $T_v$ to $w$. Given the labels we can find the shortest cycle needed among the directed $\mathcal{A}$ in time $O(nm)$. For the deterministic algorithm we need to compute modulo $N$ such primes, thus getting an $O(nm^2)$ algorithm for each cycle.

[1] Computing the support vector $S_i$ in each phase $i$ is performed using fast matrix multiplication [13].

This approach is faster and significantly simpler than previous methods which used complicated modifications of Dijkstra's algorithm with multiple frontiers.

**Theorem 4.3.** *A minimum cycle basis in a directed graph with non-negative edge weights can be computed in time $O(m^2n)$ with success probability $3/4$ and in time $O(m^3n)$ deterministically.*

# 5 Approximation Algorithms

It was shown in [13] that the support vectors approach produces an $\epsilon$-approximate MCB if in each phase $1 \leq i \leq N$ we compute a cycle $C$ non-orthogonal with $S_i$ and $C$ is an $\epsilon$-approximation of the shortest cycle in $G$ non-orthogonal with $S_i$. The computation of such an approximate cycle is reduced to $n$ approximate shortest path computations in an appropriate graph $G(S_i)$. Note that in each phase $i$ we do shortest paths in a different graph. Recently, Baswana and Kavitha [2] developed a new algorithm to compute 2-approximate paths. The algorithm has an $O(m\sqrt{n}\log n + n^2)$ expected running time. Using this algorithm the approach in [13] yields an $O(m^2\sqrt{n}\log n + n^2m + m^\omega)$ expected running time algorithm for a 2-approximate MCB.

In this section we develop faster algorithms which compute a 2-approximate MCB for undirected graphs. Recently, Kavitha et al. [12] constructed a set $\mathcal{H}_2 \subseteq \mathcal{H}$ of $O(m\sqrt{n\log n})$ cycles and proved that it contains a 2-approximate MCB. Constructing the $n$ shortest path trees and identifying which of the cycles in $\mathcal{H}$ are part of $\mathcal{H}_2$ can be done in expected time $O(n(m + n\log n))$. Then it is straightforward to use our new approach with set $\mathcal{H}_2$ without the extra bit packing. This would give us an $O(m^2\sqrt{n\log n} + n^2m + m^\omega)$ time algorithm. However, by changing the definition of $i_{e,B}(A)$ in order to return only vertices such that $C[v, e] \in \mathcal{H}_2$, we get an extra logarithmic speedup.

**Theorem 5.1.** *A 2-approximate MCB of an undirected graph with non-negative edge weights can be computed in expected time $O(m^2\sqrt{n/\log n} + n^2m + m^\omega)$.*

In the same paper, for any integer $k > 1$, a $(2k-1)$-approximation algorithm is presented. This is a general technique (for both undirected and directed graphs) where the approximate MCB computation is reduced to the computation of an MCB of a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges plus an additional $O(mn^{1+1/k})$ term. Combining this reduction with our new algorithm we get the following.

**Theorem 5.2.** *A $(2k-1)$-approximate MCB, for any integer $k > 1$, of an undirected graph with non-negative edge weights can be computed in time $O(n^{3+2/k}/\log n + n^{3+1/k})$.*

An important corollary of the above theorem is that an $O(\log n)$ approximation in undirected graphs is computed in $O(n^3)$ time.

# 6 Conclusions and Open Problems

We presented exact and approximate algorithms for computing minimum cycle bases in undirected and directed graphs. Our new technique improves the running time by a logarithmic factor, in some cases only for sparse graphs, in other for dense as well. We remark that the important case is sparse graphs, commonly appearing in practice. We also believe that this new approach is much simpler and more likely to be further improved. It brings together all previous research on MCB algorithms.

This brings Horton's collection approach into the foreground and, thus, raises an important open question. We would like to asymptotically reduce the size of $\mathcal{A}$, identifying a smaller set guarantied to contain an MCB. This would immediately improve all algorithms.

Furthermore, our algorithms exhibit an $O(n^2m)$ factor resulting from the label computation when traversing the $n$ shortest path trees in each of the $m$ phases. Is it possible to do bit compression while doing the traversal, thus reducing this factor to $O(n^2m/\log n)$. This would imply $o(n^3)$ algorithms for sparse graphs and would be a first important step into designing an $O(m^\omega)$ time algorithm for all graph densities.

# References


[1] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

[2] Surender Baswana and Telikepalli Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *Proceedings of the 47th Annual IEEE FOCS*, 2006.

[3] F. Berger, P. Gritzmann, and S. de Vries. Minimum cycle basis for network graphs. *Algorithmica*, 40(1):51–62, 2004.

[4] B. Bollobas. *Modern Graph Theory*. Springer-Verlag, 1998.

[5] A. C. Cassell, J. C. Henderson, and K. Ramachandran. Cycle bases of minimal measure for the structural analysis of skeletal structures by the flexibility method. In *Proc. Royal Society of London Series A*, volume 350, pages 61–70, 1976.

[6] J.C. de Pina. *Applications of Shortest Path Methods*. PhD thesis, University of Amsterdam, Netherlands, 1995.

[7] Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice Hall, 1974.

[8] Petra Manuela Gleiss. *Short Cycles, Minimum Cycle Bases of Graphs from Chemistry and Biochemistry*. PhD thesis, Fakultät Für Naturwissenschaften und Mathematik der Universität Wien, 2001.

[9] A. Golynski and J. D. Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *8th Scandinavian Workshop on Algorithm Theory*, 2002.

[10] Ramesh Hariharan, Telikepalli Kavitha, and Kurt Mehlhorn. Faster randomized and deterministic algorithms for minimum cycle bases in directed graphs. Preliminary versions of the results in this paper appeared in ICALP'05 and ICALP'06. submitted for publication, 2006.

[11] J. D. Horton. A polynomial-time algorithm to find a shortest cycle basis of a graph. *SIAM Journal of Computing*, 16:359–366, 1987.

[12] Telikepalli Kavitha, Kurt Mehlhorn, and Dimitrios Michail. New approximation algorithms for minimum cycle bases of graphs. In *24th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 4393 of *Lecture Notes in Computer Science*, pages 512–523, 2007.

[13] Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E. Paluch. A faster algorithm for minimum cycle basis of graphs. In *31st International Colloquium on Automata, Languages and Programming, Finland*, pages 846–857, 2004.

[14] Jon Lee and Jennifer Ryan. Matroid applications and algorithms. *ORSA Journal on Computing*, 4(1):70–96, 1992.

[15] Josef Leydold and Peter F. Stadler. Minimal cycle bases of outerplanar graphs. *Electron. J. of Combinatorics*, 5:1–14, 1998.

[16] Christian Liebchen and Romeo Rizzi. A greedy approach to compute a minimum cycle basis of a directed graph. *Inf. Process. Lett.*, 94(3):107–112, 2005.

[17] G. F. Stepanec. Basis systems of vector cycles with extremal properties in graphs. *Uspekhi Mat. Nauk*, 19:171–175, 1964.

[18] M. N. S. Swamy and K. Thulasiraman. *Graphs, Networks, and Algorithms.* John Wiley & Sons, New York, 1981.

[19] Geetika Tewari, Craig Gotsman, and Steven J. Gortler. Meshing genus-1 point clouds using discrete one-forms. *Computers and Graphics*, 2006. to appear.