

# An $\tilde{O}(m^2n)$ algorithm for Minimum Cycle Basis of graphs <sup>\*</sup>

Telikepalli Kavitha<sup>†</sup>   Kurt Mehlhorn<sup>‡</sup>   Dimitrios Michail<sup>‡</sup>  
Katarzyna E. Paluch<sup>‡</sup>

## Abstract

We consider the problem of computing a minimum cycle basis in a graph  $G$  with  $m$  edges and  $n$  vertices. The input to this problem is an undirected graph whose edges have non-negative weights. In this problem a  $\{0, 1\}$  incidence vector is associated with each cycle and the vector space over  $\mathbb{F}_2$  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 weights of the cycles is minimum is called a minimum cycle basis of  $G$ . Apart from its interest as a natural question, the problem of efficiently computing a minimum cycle basis is motivated by its use as a preprocessing step in several algorithms.

The previous best result for computing a minimum cycle basis in an undirected graph was an  $O(m^\omega n)$  algorithm, where  $\omega$  is the best exponent of matrix multiplication. It is presently known that  $\omega < 2.376$ . We obtain an  $O(m^2n + mn^2 \log n)$  algorithm for this problem. When the edge weights are integers, we have an  $O(m^2n)$  algorithm. For unweighted graphs which are reasonably dense, our algorithm runs in  $O(m^\omega)$  time. For any  $\epsilon > 0$ , we also design a  $1 + \epsilon$  approximation algorithm to compute a cycle basis which is at most  $1 + \epsilon$  times the weight of a minimum cycle basis. The running time of this algorithm is  $O(\frac{m^\omega}{\epsilon} \log(W/\epsilon))$  for reasonably dense graphs, where  $W$  is the largest edge weight.

**keywords:** cycle basis, matrix multiplication, approximate minimum cycle basis

## 1 Introduction

Let  $G = (V, E)$  be an undirected graph with  $m$  edges and  $n$  vertices. A *cycle* of  $G$  is any subgraph of  $G$  where each vertex has even degree. Associated with each cycle  $C$  is an *incidence vector*  $x$ , indexed on  $E$ , where for any  $e \in E$

$$x_e = \begin{cases} 1 & \text{if } e \text{ is an edge of } C \\ 0 & \text{otherwise} \end{cases}$$

The vector space over  $\mathbb{F}_2$  generated by the incidence vectors of cycles is called the *cycle space* of  $G$ . It is well-known that when  $G$  is connected, this vector space has dimension  $m - n + 1$ , 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*.

---

<sup>\*</sup>A preliminary version of this paper appeared in the Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP), 2004 [13].

<sup>†</sup>Indian Institute of Science, Bangalore, India. kavitha@csa.iisc.ernet.in

<sup>‡</sup>Max-Planck-Institut für Informatik, Saarbrücken, Germany. {mehlhorn,michail,paluch}@mpi-sb.mpg.de

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 consider the problem of computing a minimum cycle basis of  $G$ . (We sometimes use the abbreviation MCB to refer to a minimum cycle basis.)

The problem of computing a minimum cycle basis has been extensively studied, both in its general setting and in special classes of graphs. Its importance lies in understanding the cyclic structure of a graph and its use as a preprocessing step in several algorithms. That is, a cycle basis is used as an input for a later algorithm, and using a minimum cycle basis instead of any arbitrary cycle basis usually reduces the amount of work that has to be done by this later algorithm. Such algorithms include algorithms for diverse applications like electrical circuit theory [2] and structural engineering [1].

**History of the problem.** The oldest known references to the minimum cycle basis are Stepanec [18] and Zykov [22]. Though polynomial time algorithms for this problem were claimed, these algorithms were not correct [12, 14]. The first polynomial time algorithm for the minimum cycle basis problem was given by Horton [11], and had running time  $O(m^3n)$ .

Horton's approach was to create a set  $M$  of  $mn$  cycles which he proved was a superset of an MCB and then extract the MCB as the shortest  $m - n + 1$  linearly independent cycles from  $M$  using Gaussian elimination. Golynski and Horton [8] observed that the shortest  $m - n + 1$  linearly independent cycles could be obtained from  $M$  in  $O(m^\omega n)$  time using fast matrix multiplication algorithms, where  $\omega$  is the best exponent for matrix multiplication. It is presently known [4] that  $\omega < 2.376$ . The  $O(m^\omega n)$  algorithm was the best known algorithm for the MCB problem.

De Pina [6] gave an  $O(m^3 + mn^2 \log n)$  to compute an MCB in a graph. The approach in [6] is different from that of Horton; de Pina's algorithm is similar to the algorithm of Padberg and Rao [15] to solve the minimum weighted  $T$ -odd cut problem. Our new algorithm to compute an MCB is also based on the same approach, which is algebraic in nature.

For planar graphs, Hartvigsen and Mardon [9] showed that an MCB can be computed in  $O(n^2 \log n)$  time. In [10] Hartvigsen and Mardon study the structure of minimum cycle bases and characterise graphs whose short cycles<sup>1</sup> form an MCB.

Closely related to problem of computing an MCB is the problem of finding a minimum *fundamental cycle basis*. In this problem, given a connected graph  $G$ , we are required to find a spanning tree  $T$  of  $G$  such that the fundamental cycle basis (where each cycle is of the form: one edge from  $G \setminus T$  and a path in  $T$ ) is as small as possible. This problem has been shown to be NP-complete [5]. The minimum cycle basis problem is also NP-complete when negative edge weights are allowed.

In this paper we obtain the following new results: For graphs with arbitrary non-negative weights on edges, we give an  $O(m^2n + mn^2 \log n)$  algorithm to compute an MCB, improving upon the current  $O(m^\omega n)$  upper bound. In particular, whenever  $m \geq n \log n$ , we have an  $O(m^2n)$  algorithm. Also, when the edge weights are integers, we have an  $O(m^2n)$  algorithm. When the edge weights are small integers (which also includes unweighted graphs), we have an  $\tilde{O}(mn^\omega) + O(m^\omega)$  algorithm. If the graph is reasonably dense, that is, if  $m \geq n^{1+1/(\omega-1)} \text{poly}(\log n)$  then the  $O(m^\omega)$  term dominates and so this is an  $O(m^\omega)$  algorithm then.

---

<sup>1</sup>A cycle  $C$  is considered a short cycle if it is the shortest cycle through one of its edges

We use an all pairs shortest paths (APSP) algorithm as a subroutine in our algorithm. The running time of our algorithm is actually  $O(n \times \text{time for an all pairs shortest paths computation in } G)$ . Using Dijkstra's algorithm for this computation, we obtain the above time of  $O(m^2n + mn^2 \log n)$ . We obtain the better running times for integer edge weights and unweighted graphs by using faster all pairs shortest path algorithms for these cases [17, 7, 19, 20]. Similarly, when the graph is sparse, there are faster APSP algorithms, using which, our algorithm can be made faster<sup>2</sup>. Using the APSP algorithm in [16], the running time of our algorithm is  $O(mn\alpha(m, n))$ , where  $\alpha(m, n)$  is Tarjan's inverse Ackermann function.

We also look at approximation algorithms for computing a minimum cycle basis in a graph. Given any  $c > 1$ , we have a  $c$ -approximation algorithm by relaxing the shortest paths subroutine to a  $c$ -stretch paths subroutine. (A  $c$ -stretch  $(s, t)$  path is a path which is at most  $c$  times the length of a shortest  $(s, t)$  path.) The running time of our algorithm which computes a cycle basis whose weight is at most twice the weight of an MCB is  $\tilde{O}(m^{3/2}n^{3/2}) + O(m^\omega)$  using the result in [3] to compute 2-stretch paths. For reasonably dense graphs (say,  $m \geq n^{1.5/(\omega-1.5)}$  poly( $\log n$ )), this is an  $O(m^\omega)$  algorithm. Using the all pairs  $(1 + \epsilon)$ -stretch paths algorithm [21], for any  $\epsilon > 0$ , we have an  $\tilde{O}(mn^\omega/\epsilon \log(W/\epsilon)) + O(m^\omega)$  algorithm to compute a cycle basis which is at most  $1 + \epsilon$  times the weight of an MCB, where  $W$  is the largest edge weight in the graph. If  $m \geq n^{1+1/(\omega-1)}$  poly( $\log n$ ) and all edge weights are polynomial in  $n$ , then this is an  $O(m^\omega/\epsilon \log(1/\epsilon))$  algorithm. We also give an  $O(m^\omega)$  algorithm to construct a *witness* of a minimum cycle basis.

The rest of this paper is organised as follows. In Sections 2 and 3 we present a simple algebraic framework (based on De Pina's algorithm) to compute a minimum cycle basis in a graph. In Section 4 we give our algorithm. In Section 5 we give a  $c$ -approximation algorithm to compute a cycle basis whose weight is  $\leq c \cdot \text{weight of an MCB}$ . In Section 6 we give an algorithm to obtain a certificate or witness of a minimum cycle basis.

## 2 A Simple MCB Algorithm

Let  $G = (V, E)$  be an undirected graph with  $m$  edges and  $n$  vertices, and with non-negative weights on its edges. It is well-known that the dimension of the cycle space of  $G$  is  $m - n +$  the number of connected components in  $G$ . We can assume that  $G$  is connected since a minimum cycle basis of a graph is the union of the minimum cycle bases of its connected components. Then  $N = m - n + 1$  is the dimension of the cycle space of  $G$ .

De Pina [6] gave the combinatorial algorithm in Figure 1 to compute a minimum cycle basis in  $G$ . Let  $T$  be any spanning tree in  $G$ . Let  $e_1, \dots, e_N$  be the edges of  $G \setminus T$  in some arbitrary but fixed order.

Note that each set  $S_{k,k}$  always contains the edge  $e_k$ . So the fundamental cycle formed by edges of  $T$  and  $e_k$  has intersection exactly 1 with  $S_{k,k}$ . Hence, the set of cycles with an odd number of edges in  $S_{k,k}$  is non-empty.

Before we show the correctness of de Pina's algorithm, we would like to interpret this algorithm algebraically. We feel that the intuition behind the above algorithm and the idea as to why it works is not clear from the combinatorial version of the algorithm. From the algebraic version of the algorithm, the scope for improvement is also clear.

---

<sup>2</sup>Our algorithm cannot be made to run faster than  $m^\omega$  though.

Initialize  $S_{1,i} = \{e_i\}$  ( $i = 1, \dots, N$ ).  
**for**  $k = 1, \dots, N$  **do**  
    Find a minimum weight cycle  $C_k$  with an odd number of edges in  $S_{k,k}$ .  
    Define for  $i = k + 1, \dots, N$ :

$$S_{k+1,i} = \begin{cases} S_{k,i} & \text{if } C_k \text{ has an even number of edges in } S_{k,i} \\ S_{k,i} \triangle S_{k,k} & \text{if } C_k \text{ has an odd number of edges in } S_{k,i} \end{cases}$$

**end for**

The algorithm returns  $\{C_1, \dots, C_N\}$ .

Figure 1: De Pina's combinatorial algorithm for computing a MCB

**An algebraic description.** A cycle in  $G$  can be viewed in terms of its incidence vector. So each cycle is a vector (with 0's and 1's in its coordinates) in the space spanned by all the edges. Here we will only look at these vectors restricted to the coordinates indexed by  $\{e_1, \dots, e_N\}$ . That is, each cycle can be represented as a vector in  $\{0, 1\}^N$ .

In SIMPLE-MCB (see Figure 2) we compute the cycles of a minimum cycle basis and their *witnesses*. A witness  $S$  of a cycle  $C$  is a subset of  $\{e_1, \dots, e_N\}$  which will prove that  $C$  belongs to our minimum cycle basis. We will view these witnesses or subsets in terms of their incidence vectors over  $\{e_1, \dots, e_N\}$ . Hence, both cycles and their witnesses are vectors in the space  $\{0, 1\}^N$ .

$\langle C, S \rangle$  stands for the standard inner product or dot product of the vectors  $C$  and  $S$ . We say that a vector  $S$  is orthogonal to  $C$  if  $\langle C, S \rangle = 0$ . Since we are in the field  $\mathbb{F}_2$ , observe that  $\langle C, S \rangle = 1$  if and only if  $C$  contains an odd number of edges of  $S$ . We present in Figure 2 a succinct description of the algorithm SIMPLE-MCB.

For  $i = 1$  to  $N$  do the following:

1. Let  $S_i$  be any arbitrary non-zero vector in the subspace orthogonal to  $\{C_1, C_2, \dots, C_{i-1}\}$ .

That is,  $S_i \neq \vec{0}$  satisfies:  $\langle C_k, S_i \rangle = 0$  for  $k \in \{1, \dots, i-1\}$ .

[Initially,  $S_1$  is any arbitrary non-zero vector in the space  $\{0, 1\}^N$ .]

2. Compute a minimum weight cycle  $C_i$  such that  $\langle C_i, S_i \rangle = 1$ .

Figure 2: SIMPLE-MCB: An algebraic framework for computing an MCB

Since each  $S_i$  is non-zero, it has to contain at least one edge  $e$  from  $G \setminus T$ . The cycle  $C_e$  formed by the edges of  $T$  and  $e$  has intersection of size exactly 1 with  $S_i$ . So, there is always at least one cycle that satisfies  $\langle C_e, S_i \rangle = 1$ .

It is easy to see that  $C_i$  is independent of  $C_1, \dots, C_{i-1}$ . This is because any vector  $v$  in the span of  $\{C_1, \dots, C_{i-1}\}$  satisfies  $\langle v, S_i \rangle = 0$  since  $\langle C_j, S_i \rangle = 0$  for each  $1 \leq j \leq i-1$ . But  $C_i$  satisfies  $\langle C_i, S_i \rangle = 1$ . Hence  $C_i$  does not lie in the subspace spanned by  $\{C_1, \dots, C_{i-1}\}$ . Thus it follows immediately that  $\{C_1, \dots, C_N\}$  is a basis. Let us now prove that  $\{C_1, \dots, C_N\}$  is a minimum cycle basis.

**Theorem 1** *The set  $\{C_1, C_2, \dots, C_N\}$  determined in SIMPLE-MCB is a minimum cycle basis.*

**Proof.** (from [6]) Suppose not. Then there exists some  $0 \leq i < N$  such that there is a minimum cycle basis  $\mathcal{B}$  that contains  $\{C_1, \dots, C_i\}$  but there is no minimum cycle basis that contains  $\{C_1, \dots, C_i, C_{i+1}\}$ . Since the cycles in  $\mathcal{B}$  form a spanning set, there exist cycles  $B_1, \dots, B_k$  in  $\mathcal{B}$  such that

$$C_{i+1} = B_1 + B_2 + \dots + B_k. \quad (1)$$

Since  $\langle C_{i+1}, S_{i+1} \rangle = 1$ , there exists some  $B_j$  in the above sum such that  $\langle B_j, S_{i+1} \rangle = 1$ . But  $C_{i+1}$  is a *minimum* weight cycle such that  $\langle C_{i+1}, S_{i+1} \rangle = 1$ . So the weight of  $C_{i+1} \leq$  the weight of  $B_j$ .

Let  $\mathcal{B}' = \mathcal{B} \cup \{C_{i+1}\} \setminus \{B_j\}$ . Since  $B_j$  can be generated as the sum of  $C_{i+1}$  and  $\{B_1, \dots, B_k\} \setminus \{B_j\}$  (refer Equation (1)), it is easy to see that  $\mathcal{B}'$  is also a basis. And  $\mathcal{B}'$  has weight at most the weight of  $\mathcal{B}$  which is a minimum cycle basis. So  $\mathcal{B}'$  is also a minimum cycle basis.

We have  $\{C_1, C_2, \dots, C_{i+1}\} \subseteq \mathcal{B}'$  because by assumption  $\{C_1, \dots, C_i\} \subseteq \mathcal{B}$  and the cycle  $B_j$  that was omitted from  $\mathcal{B}$  cannot be equal to any one of  $C_1, \dots, C_i$  because  $\langle B_j, S_{i+1} \rangle = 1$  whereas  $\langle C_j, S_{i+1} \rangle = 0 \quad \forall j \leq i$ . The existence of the minimum cycle basis  $\mathcal{B}'$  that contains the cycles  $C_1, \dots, C_{i+1}$  contradicts our assumption that there is no minimum cycle basis containing  $\{C_1, \dots, C_i, C_{i+1}\}$ . Hence,  $\{C_1, C_2, \dots, C_N\}$  is indeed a minimum cycle basis.  $\blacksquare$

Thus we have shown the correctness of the algorithm SIMPLE-MCB (Figure 2), which is equivalent to the combinatorial algorithm in Figure 1. There are two subroutines in SIMPLE-MCB: computing a non-zero vector  $S_i$  in the subspace orthogonal to  $\{C_1, \dots, C_{i-1}\}$  and computing a minimum weight cycle  $C_i$  such that  $\langle C_i, S_i \rangle = 1$ . Let us now describe how to compute such a minimum weight cycle  $C_i$  and in Section 3 we shall see a simple method to compute a non-zero vector  $S_i$  that is orthogonal to  $C_1, \dots, C_{i-1}$ .

## 2.1 Computing the cycles

Given  $S_i$ , it is easy to compute a minimum weight cycle  $C_i$  such that  $\langle C_i, S_i \rangle = 1$  by reducing it to  $n$  shortest path computations in an appropriate graph  $G_i$ . The following construction is well-known. The graph  $G_i$  is defined from  $G = (V, E)$  and  $S_i \subseteq E$  in the following manner.

- $G_i$  has two copies of each vertex  $v \in V$ . Call them  $v^+$  and  $v^-$ .
- for** every edge  $e = (v, u) \in E$  **do**
- if**  $e \notin S_i$  **then**
- Add edges  $(v^+, u^+)$  and  $(v^-, u^-)$  to the edge set of  $G_i$ .
- {Assign their weights to be the same as  $e$ .}
- else**
- Add edges  $(v^+, u^-)$  and  $(v^-, u^+)$  to the edge set of  $G_i$ .
- {Assign their weights to be the same as  $e$ .}
- end if**
- end for**

$G_i$  can be visualised as 2 levels of  $G$  (the  $+$  level and the  $-$  level). Within each level, we have edges of  $E \setminus S_i$ . Between the levels we have the edges of  $S_i$ . In Figure 3 is an example

of  $G_i$  when the graph  $G$  has 4 vertices  $\{1, 2, 3, 4\}$  and 4 edges  $\{(1, 2), (1, 4), (2, 4), (3, 4)\}$  and only the edge  $(1, 2)$  is in  $S_i$ .

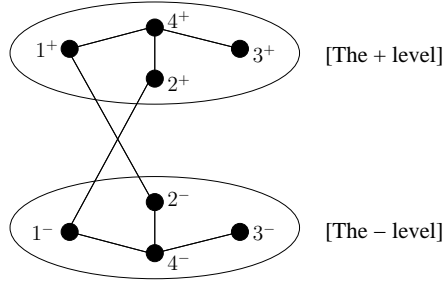


Figure 3: An example of the graph  $G_i$ . Since the edge  $(1, 2) \in S_i$ , we have the edges  $(1^-, 2^+), (1^+, 2^-)$  going across the + and - levels. The edges not in  $S_i$ , i.e.  $(1, 4), (2, 4), (3, 4)$  have copies inside the + level and inside the - level.

Given any  $v^+$  to  $v^-$  path  $p$  in  $G_i$ , we can correspond to it a cycle in  $G$  by identifying the vertices and edges in  $G_i$  with their corresponding vertices and edges in  $G$ . For instance, the path  $1^- - 2^+ - 4^+ - 1^+$  in  $G_i$  in Figure 3 corresponds to the cycle  $1-2-4-1$  in  $G$ . Because we identify both  $v^+$  and  $v^-$  with  $v$ , any  $v^+$  to  $v^-$  path  $p$  in  $G_i$  corresponds to a cycle  $C$  in  $G$ .

More formally, take the incidence vector of the path  $p$  (over the edges of  $G_i$ ) and obtain an incidence vector over the edges of  $G$  by identifying  $(v^*, u^\dagger)$  with  $(v, u)$  where  $*$  and  $\dagger$  are + or -. Suppose the path  $p$  contained two copies of the same edge. (It could have contained both  $(v^+, u^-)$  and  $(v^-, u^+)$  for some  $(v, u)$ .) Then add the number of occurrences of that edge modulo 2 to obtain an incidence vector over the edges of  $G$ .

Let  $p = \min_{v \in V}$  shortest  $(v^+, v^-)$  path in  $G_i$ . Then we can show the following lemma.

**Lemma 1** *The path  $p$  in  $G_i$  corresponds a minimum weight cycle  $C$  in  $G$  that has odd intersection with  $S_i$ .*

**Proof.** Since the endpoints of the path  $p$  are  $v^+$  and  $v^-$ ,  $p$  has to contain an odd number of edges of  $S_i$ . This is because only edges of  $S_i$  provide a change of sign and  $p$  goes from a + vertex to a - vertex. We might have deleted some edges of  $S_i$  while forming  $C$  since those edges occurred with a multiplicity of 2. But this means that we always delete an even number of edges from  $S_i$ . Hence  $C$  has an odd number of edges of  $S_i$  present in it. Also, the weight of  $C \leq$  the weight (or length) of  $p$  since edges have non-negative weights.

We should now prove that  $C$  is a minimum weight cycle among such cycles. Let  $C'$  be any other cycle in  $G$  with an odd number of edges of  $S_i$  in it. If  $C'$  is not a simple cycle, then  $C'$  is a union of simple cycles (with disjoint edges) and at least one of those simple cycles  $C_0$  should have an odd number of edges of  $S_i$  present in it. And the weight of  $C_0 \leq$  the weight of  $C'$ .

Let  $u$  be a vertex in  $C_0$ . We will identify  $C_0$  with a path in  $G_i$  by traversing  $C_0$  starting at the vertex  $u$  and identifying it with  $u^+$ . If we traverse an edge  $e$  of  $S_i$ , then we identify the vertices incident on  $e$  with opposite signs. And if we traverse an edge outside  $S_i$ , then we identify the vertices incident on  $e$  with the same sign. Since  $C_0$  is a cycle, we come back to the vertex  $u$ . Also,  $C_0$  has an odd number of edges of  $S_i$  present in it. So the sign of the final vertex is of the opposite sign to the sign of the starting vertex. Hence,  $C_0$  translates to a  $u^+$  to  $u^-$  path  $p'$  in  $G_i$ . And the weight of  $p' =$  the weight of  $C_0$ .

But  $p$  was the minimum weight path among all shortest  $(v^+, v^-)$  paths in  $G_i$  for all  $v \in V$ . Hence, the weight of  $p \leq$  the weight of  $p'$ . So we finally we get that the weight of  $C \leq$  the weight of  $p \leq$  the weight of  $p' \leq$  the weight of  $C'$ . This proves that  $C$  is a minimum weight cycle that has odd intersection with  $S_i$ .  $\blacksquare$

The computation of the path  $p$  can be done by computing  $n$  shortest  $(v^+, v^-)$  paths (each by Dijkstra's algorithm) in  $G_i$  and taking their minimum or by one invocation of an all-pairs-shortest paths algorithm in  $G_i$ . This computation takes  $O(n(m + n \log n))$  time. Note that depending on the relation between  $m$  and  $n$ , the algorithm can choose which shortest path algorithm to use. For example, in the case when the edge weights are integers or the unweighted case it is better to use faster all-pairs-shortest paths algorithms than run Dijkstra's algorithm  $n$  times.

Since we have to compute totally  $N$  such cycles  $C_1, C_2, \dots, C_N$ , we spend  $O(mn(m + n \log n))$  time, since  $N = m - n + 1$ .

### 3 Computing the subsets

We will now consider the problem of computing the subsets  $S_i$ , for  $i = 1$  to  $N$ . We want  $S_i$  to be a non-zero vector in the subspace orthogonal to  $\{C_1, \dots, C_{i-1}\}$ . One way to find a non-zero vector in a subspace is to maintain the whole basis of the subspace. Any vector in that basis will then be a non-zero vector in the subspace.

Initially,  $S_j = \{e_j\}$  for all  $j$ ,  $1 \leq j \leq N$ . This corresponds to the standard basis of the space  $\{0, 1\}^N$ . At the beginning of phase  $i$ , we have  $\{S_i, S_{i+1}, \dots, S_N\}$  which is a basis of the space  $\mathcal{C}^\perp$  orthogonal to the space  $\mathcal{C}$  spanned by  $\{C_1, \dots, C_{i-1}\}$ . We use  $S_i$  to compute  $C_i$  and update  $\{S_{i+1}, \dots, S_N\}$  to a basis  $\{S'_{i+1}, \dots, S'_N\}$  of the subspace of  $\mathcal{C}^\perp$  that is orthogonal to  $C_i$ . The update step of phase  $i$  is as follows:

For  $i + 1 \leq j \leq N$ , let

$$S'_j = \begin{cases} S_j & \text{if } \langle C_i, S_j \rangle = 0 \\ S_j + S_i & \text{if } \langle C_i, S_j \rangle = 1 \end{cases}$$

**Lemma 2**  $S'_{i+1}, \dots, S'_N$  form a basis of the subspace orthogonal to  $C_1, \dots, C_i$ .

**Proof.** We will first show that  $S'_{i+1}, \dots, S'_N$  belong to the subspace orthogonal to  $C_1, \dots, C_i$ . We know that  $S_i, S_{i+1}, \dots, S_N$  form a basis of the subspace orthogonal to  $C_1, \dots, C_{i-1}$ . Since each  $S'_j$ ,  $i + 1 \leq j \leq N$  is a linear combination of  $S_j$  and  $S_i$ , it follows that  $S'_j$  is orthogonal to  $C_1, \dots, C_{i-1}$ . If an  $S_j$  is already orthogonal to  $C_i$ , then we leave it as it is, i.e.,  $S'_j = S_j$ . Otherwise,  $\langle S_j, C_i \rangle = 1$ , and we update  $S_j$  as  $S'_j = S_j + S_i$ . Since both  $\langle C_i, S_j \rangle$  and  $\langle C_i, S_i \rangle$  are equal to 1, it follows that each  $S'_j$  is now orthogonal to  $C_i$  also. Hence,  $S'_{i+1}, \dots, S'_N$  belong to the subspace orthogonal to  $C_1, \dots, C_i$ .

Now we will show that  $S'_{i+1}, \dots, S'_N$  are linearly independent. Suppose there is a linear dependence among them. Substitute  $S'_j$ 's in terms of  $S_j$ 's and  $S_i$  in the linear dependence relation.  $S_i$  is the only vector that might occur more than once in that relation. So either  $S_i$  occurs an even number of times and gets cancelled and we get a linear dependence among  $S_{i+1}, \dots, S_N$  or  $S_i$  occurs an odd number of times, in which case we get a linear dependence

among  $S_i, S_{i+1}, \dots, S_N$ . Either case contradicts the linear independence of  $S_i, S_{i+1}, \dots, S_N$ . Hence,  $S'_{i+1}, \dots, S'_N$  are linearly independent.  $\blacksquare$

This completes the description of the algorithm SIMPLE-MCB. Let us now bound the running time of this algorithm. During the update step of the  $i$ -th iteration, the cost of updating each  $S_j, j > i$  is  $N$  and hence it is  $N(N - i)$  for updating  $S_{i+1}, \dots, S_N$ . Since we have  $N$  iterations, the total cost of maintaining this basis is  $N^3$ , which is  $O(m^3)$ .

The total running time of the algorithm SIMPLE-MCB, by summing the costs of computing the cycles and witnesses, is  $O(m^3 + mn^2 \log n)$ . So, using Dijkstra's algorithm or a faster algorithm for computing all-pairs-shortest-paths is not really crucial; the time taken to compute  $S_i$  is the real bottleneck.

## 4 A faster implementation

Recall our approach to compute the vectors  $S_i$ . We maintained a basis of  $\mathcal{C}^\perp$  in each iteration and that costed us  $m^2$  in each iteration. Note that we needed just one vector from the subspace orthogonal to  $\{C_1, \dots, C_i\}$ . But the algorithm maintained  $N - i$  such vectors:  $S_{i+1}, \dots, S_N$ . This was the limiting factor in the running time of the algorithm. In order to improve the running time of SIMPLE-MCB, we relax the invariant that  $S_{i+1}, \dots, S_N$  form a basis of the subspace orthogonal to  $C_1, \dots, C_i$ . Since we need just one vector in this subspace, we can afford to relax this invariant and maintain the correctness of the algorithm.

In SIMPLE-MCB in the  $i$ -th iteration we update  $S_{i+1}, \dots, S_N$ . Our idea now is to update only those  $S_j$ 's where  $j$  is close to  $i$  and postpone the update of the later  $S_j$ 's. During the postponed update, many  $S_j$ 's can be updated simultaneously. This simultaneous update is implemented as a matrix multiplication step. And using a fast algorithm for matrix multiplication causes the speedup.

Our main procedure is called *extend\_cycle\_basis*. The procedure *extend\_cycle\_basis* works in a recursive manner. We present in Figure 4 the overall algorithm FAST-MCB and the procedure *extend\_cycle\_basis*.

The procedure *extend\_cycle\_basis*( $\{C_1, \dots, C_i\}, \{S_{i+1}, \dots, S_{i+k}\}, k$ ) computes  $k$  new cycles  $C_{i+1}, \dots, C_{i+k}$  of the minimum cycle basis using the subsets  $S_{i+1}, \dots, S_{i+k}$ . We maintain the invariant that these subsets are all orthogonal to  $C_1, \dots, C_i$ . It first computes  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$  using  $S_{i+1}, \dots, S_{i+\lfloor k/2 \rfloor}$ . At this point, the remaining subsets  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$  need not be orthogonal to the new cycles  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$ . Our algorithm then updates  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$  so that they are orthogonal to  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$  and they continue to be orthogonal to  $C_1, \dots, C_i$ . Then it computes  $C_{i+\lfloor k/2 \rfloor+1}, \dots, C_{i+k}$ .

Let us see a small example as to how this works. Suppose  $N = 4$ . We initialize the subsets  $S_i, i = 1, \dots, 4$  and call *extend\_cycle\_basis*, which then calls itself with only  $S_1$  and  $S_2$  and then only with  $S_1$  and so computes  $C_1$ . Then it updates  $S_2$  so that  $\langle C_1, S_2 \rangle = 0$  and computes  $C_2$ . Then it simultaneously updates  $S_3$  and  $S_4$  which were still at their initial values so that the updated  $S_3$  and  $S_4$  (which we call  $T_3$  and  $T_4$ ) are both orthogonal to  $C_1$  and  $C_2$ . Then it computes  $C_3$  using  $T_3$  and updates  $T_4$  and then computes  $C_4$ .

Observe that whenever we compute  $C_{i+1}$  using  $S_{i+1}$ , we have the property that  $S_{i+1}$  is orthogonal to  $C_1, \dots, C_i$ . The difference is the function *update* which allows us to update many  $S_j$ 's simultaneously to be orthogonal to many  $C_i$ 's. As mentioned earlier, this simultaneous update enables us to use fast matrix multiplication algorithms which are crucial to the

**The algorithm FAST-MCB:**

- Initialize the cycle basis with the empty set and initialize  $S_j = \{e_j\}$  for  $1 \leq j \leq N$ .
- Call the procedure  $extend\_cycle\_basis(\{\}, \{S_1, \dots, S_N\}, N)$ .

A call to  $extend\_cycle\_basis(\{C_1, \dots, C_i\}, \{S_{i+1}, \dots, S_{i+k}\}, k)$  extends the cycle basis by  $k$  cycles. Let  $\mathcal{C}$  denote the current partial cycle basis which is  $\{C_1, \dots, C_i\}$ .

**The procedure  $extend\_cycle\_basis(\mathcal{C}, \{S_{i+1}, \dots, S_{i+k}\}, k)$ :**

- if  $k = 1$ , compute a minimum weight cycle  $C_{i+1}$  such that  $\langle C_{i+1}, S_{i+1} \rangle = 1$ .
- if  $k > 1$ , use recursion.
  1. call  $extend\_cycle\_basis(\mathcal{C}, \{S_{i+1}, \dots, S_{i+\lfloor k/2 \rfloor}\}, \lfloor k/2 \rfloor)$  to extend the current cycle basis by  $\lfloor k/2 \rfloor$  elements. That is, the cycles  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$  are computed in a recursive manner.

During the above recursive call,  $S_{i+1}, \dots, S_{i+\lfloor k/2 \rfloor}$  get updated. Call their final versions (at the end of this step) as  $S'_{i+1}, \dots, S'_{i+\lfloor k/2 \rfloor}$ .
  2. call  $update(\{S'_{i+1}, \dots, S'_{i+\lfloor k/2 \rfloor}\}, \{S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}\})$  to update  $\{S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}\}$ . Let  $\{T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}\}$  be the output returned by  $update$ .
  3. call  $extend\_cycle\_basis(\mathcal{C} \cup \{C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}\}, \{T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}\}, \lfloor k/2 \rfloor)$  to extend the current cycle basis by  $\lfloor k/2 \rfloor$  cycles. That is, the cycles  $C_{i+\lfloor k/2 \rfloor+1}, \dots, C_{i+k}$  will be computed recursively.

Figure 4: FAST-MCB: A faster minimum cycle basis algorithm

speedup. We describe these steps in detail now.

**The function  $update$ :**

When we call the function  $update(\{S'_{i+1}, \dots, S'_{i+\lfloor k/2 \rfloor}\}, \{S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}\})$ , the sets  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$  need not all be orthogonal to the space spanned by  $\mathcal{C} \cup \{C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}\}$ . We know that  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$  are all orthogonal to  $\mathcal{C}$  and now we need to ensure that the updated  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$  (call them  $T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}$ ) are all orthogonal to  $\mathcal{C} \cup \{C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}\}$ . We now want to update the sets  $S_{i+\lfloor k/2 \rfloor+1}, \dots, S_{i+k}$ , i.e., we want to determine  $T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}$  such that for each  $j$  in the range for  $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$  we have

1.  $T_j$  is orthogonal to  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$  and
2.  $T_j$  continues to remain orthogonal to  $C_1, \dots, C_i$ .

So, we define  $T_j$  (for each  $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$ ) as follows:

$$T_j = S_j + \text{a linear combination of } S'_{i+1}, \dots, S'_{i+\lfloor k/2 \rfloor}.$$

This makes sure that  $T_j$  is orthogonal to the cycles  $C_1, \dots, C_i$  because  $S_j$  and all of  $S'_{i+1}, \dots, S'_{i+\lfloor k/2 \rfloor}$  are orthogonal to  $C_1, \dots, C_i$ . Hence,  $T_j$  which is a linear combination of them will also be orthogonal to  $C_1, \dots, C_i$ . The coefficients of the linear combination will be chosen such that  $T_j$  will be orthogonal to  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$ .

Let

$$T_j = S_j + a_{j1}S'_{i+1} + a_{j2}S'_{i+2} + \dots + a_{j\lfloor k/2 \rfloor}S'_{i+\lfloor k/2 \rfloor}.$$

We will determine the coefficients  $a_{j1}, \dots, a_{j\lfloor k/2 \rfloor}$  for all  $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$  simultaneously. Writing all these equations in matrix form, we have

$$\begin{pmatrix} T_{i+\lfloor k/2 \rfloor+1} \\ \vdots \\ T_{i+k} \end{pmatrix} = (A \ I) \cdot \begin{pmatrix} S'_{i+1} \\ \dots \\ S'_{i+\lfloor k/2 \rfloor} \\ S_{i+\lfloor k/2 \rfloor+1} \\ \dots \\ S_{i+k} \end{pmatrix}$$

where  $A$  is a  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  matrix whose  $\ell$ th row has the unknowns  $a_{j1}, \dots, a_{j\lfloor k/2 \rfloor}$ , where  $j = i + \lfloor k/2 \rfloor + \ell$ . And  $T_j$  represents a row with the coefficients of  $T_j$  as its row elements.

Let us multiply both sides of this equation with an  $N \times \lfloor k/2 \rfloor$  matrix whose columns are the cycles  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$ . That is,

$$\begin{pmatrix} T_{i+\lfloor k/2 \rfloor+1} \\ \vdots \\ T_{i+k} \end{pmatrix} \cdot (C_{i+1}^T \dots C_{i+\lfloor k/2 \rfloor}^T) = (A \ I) \cdot \begin{pmatrix} S'_{i+1} \\ \dots \\ S'_{i+\lfloor k/2 \rfloor} \\ S_{i+\lfloor k/2 \rfloor+1} \\ \dots \\ S_{i+k} \end{pmatrix} \cdot (C_{i+1}^T \dots C_{i+\lfloor k/2 \rfloor}^T)$$

Then the left hand side is the 0 matrix since each of the vectors  $T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}$  has to be orthogonal to each of  $C_{i+1}, \dots, C_{i+\lfloor k/2 \rfloor}$ . Let

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} S'_{i+1} \\ \dots \\ S'_{i+\lfloor k/2 \rfloor} \\ S_{i+\lfloor k/2 \rfloor+1} \\ \dots \\ S_{i+k} \end{pmatrix} \cdot (C_{i+1}^T \dots C_{i+\lfloor k/2 \rfloor}^T)$$

where

$$X = \begin{pmatrix} S'_{i+1} \\ \dots \\ S'_{i+\lfloor k/2 \rfloor} \end{pmatrix} \cdot (C_{i+1}^T \dots C_{i+\lfloor k/2 \rfloor}^T); \quad Y = \begin{pmatrix} S_{i+\lfloor k/2 \rfloor+1} \\ \dots \\ S_{i+k} \end{pmatrix} \cdot (C_{i+1}^T \dots C_{i+\lfloor k/2 \rfloor}^T)$$

Then

$$0 = (A \ I) \cdot \begin{pmatrix} X \\ Y \end{pmatrix}$$

We now look at this problem as a problem in linear algebra.

**A Problem in Linear Algebra.** Consider the following problem. We are given an invertible  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  matrix  $X$  and a  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  matrix  $Y$  and we want to find a  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  matrix  $A$  such that:

$$(A \ I) \cdot \begin{pmatrix} X \\ Y \end{pmatrix} = 0$$

Here 0 stands for the  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  zero-matrix and  $I$  stands for the  $\lfloor k/2 \rfloor \times \lfloor k/2 \rfloor$  identity matrix. We need  $AX + Y = 0$  or  $A = -YX^{-1} = YX^{-1}$  since we are in the field  $\mathbb{F}_2$ . We can determine  $A$  in time  $k^\omega$  using fast matrix multiplication and inverse algorithms since the matrix  $X$  is invertible.

Let us now go back to the implementation of *update*. We have the problem of the preceding paragraph if we show that  $X$  is invertible. The matrix

$$X = \begin{pmatrix} \langle S'_{i+1}, C_{i+1} \rangle & \cdots & \langle S'_{i+1}, C_{i+\lfloor k/2 \rfloor} \rangle \\ \langle S'_{i+2}, C_{i+1} \rangle & \cdots & \langle S'_{i+2}, C_{i+\lfloor k/2 \rfloor} \rangle \\ \vdots & \vdots & \vdots \\ \langle S'_{i+\lfloor k/2 \rfloor}, C_{i+1} \rangle & \cdots & \langle S'_{i+\lfloor k/2 \rfloor}, C_{i+\lfloor k/2 \rfloor} \rangle \end{pmatrix} = \begin{pmatrix} 1 & * & * & \cdots & * \\ 0 & 1 & * & \cdots & * \\ 0 & 0 & 1 & \cdots & * \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

is an upper triangular matrix with 1's on the diagonal, since each  $S'_j$  is the final version of the subset  $S_j$  using which  $C_j$  is computed, which means that  $\langle S'_j, C_j \rangle = 1$  and  $\langle S'_j, C_\ell \rangle = 0$  for all  $\ell < j$ . Hence,  $X$  is invertible. Thus  $A = YX^{-1}$ . Hence we can compute all the coefficients  $a_{j1}, \dots, a_{j\lfloor k/2 \rfloor}$  for all  $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$  simultaneously using matrix multiplication and matrix inversion algorithms.

By the implementation of the function *update*, Lemma 3 follows.

**Lemma 3** *When  $k = 1$ , i.e., whenever we call `extend_cycle_basis` ( $\{C_1, \dots, C_i\}, S_{i+1}, 1$ ), the vector  $S_{i+1}$  is orthogonal to  $\{C_1, \dots, C_i\}$ . And  $S_{i+1}$  always contains the edge  $e_{i+1}$ .*

Hence, just before we compute  $C_{i+1}$ , we always have a non-zero vector  $S_{i+1}$  orthogonal to  $\{C_1, \dots, C_i\}$ . And  $C_{i+1}$  is a minimum weight cycle such that  $\langle C_{i+1}, S_{i+1} \rangle = 1$ . Hence, the correctness of FAST-MCB follows from Theorem 1.

#### 4.1 The running time of FAST-MCB

Let us analyse the running time of the algorithm FAST-MCB. The recurrence of the algorithm is as follows:

$$T(k) = \begin{cases} \text{cost of computing a minimum weight odd cycle } C_i \text{ in } S_i & \text{if } k = 1 \\ 2T(k/2) + \text{cost of update} & \text{if } k > 1 \end{cases}$$

**Cost of update.** The computation of matrices  $X$  and  $Y$  takes time  $mk^{\omega-1}$  using the fast matrix multiplication algorithm. To compute  $X$  (and similarly,  $Y$ ) we are multiplying  $\lfloor k/2 \rfloor \times N$  by  $N \times \lfloor k/2 \rfloor$  matrices. We split the matrices into  $2N/k$  square blocks and use fast matrix multiplication to multiply the blocks. Thus multiplication takes time  $(2N/k)(k/2)^\omega = O(mk^{\omega-1})$ . We can also invert  $X$  in  $O(k^\omega)$  time and we also multiply  $Y$  and  $X^{-1}$  using fast matrix multiplication in order to get the matrix  $A$ . And we use the fast matrix multiplication

algorithm again, to multiply the matrix  $(A \ I)$  with the matrix whose rows are  $S'_{i+1}, \dots, S_{i+k}$  to get the updated subsets  $T_{i+\lfloor k/2 \rfloor+1}, \dots, T_{i+k}$ .

Using the algorithm described in Section 2.1 to compute a shortest cycle  $C_i$  that has odd intersection with  $S_i$ , the recurrence turns into

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

This solves to  $T(k) = O(k(mn + n^2 \log n) + k^{\omega-1}m)$ . Thus  $T(m) = O(m^\omega + m^2n + mn^2 \log n)$ .

Since  $m^\omega < m^2n$ , this reduces to  $T(m) = O(m^2n + mn^2 \log n)$ .

For  $m > n \log n$ , this is  $T(m) = O(m^2n)$ . For  $m \leq n \log n$ , this is  $T(m) = O(mn^2 \log n)$ . Thus we have shown the following theorem.

**Theorem 2** *A minimum cycle basis of an undirected weighted graph can be computed in time  $O(m^2n + mn^2 \log n)$ .*

Our algorithm has a running time of  $O(m^\omega + m \cdot n(m + n \log n))$ , where the  $n(m + n \log n)$  term is the cost to compute all pairs shortest paths. This term can be replaced with a better term when the graph is unweighted or the edge weights are integers or when the graph is sparse.

When the edges of  $G$  have integer weights, we can compute all pairs shortest paths in time  $O(mn)$  [19, 20], that is, we can bound  $T(1)$  by  $O(mn)$ . When the graph is unweighted or the edge weights are small integers, we can compute all pairs shortest paths in time  $\tilde{O}(n^\omega)$  [17, 7]. When such graphs are reasonably dense, say  $m \geq n^{1+1/(\omega-1)} \text{poly}(\log n)$ , then the  $m^\omega$  term dominates the running time of our algorithm. We conclude with the following theorem.

**Theorem 3** *A minimum cycle basis in a graph with integer edge weights can be computed in time  $O(m^2n)$ . For unweighted graphs which satisfy  $m \geq n^{1+1/(\omega-1)} \text{poly}(\log n)$ , for some fixed polynomial, we have an  $O(m^\omega)$  algorithm to compute a minimum cycle basis.*

## 5 An approximation algorithm for Minimum Cycle Basis

The bottleneck in the running time of our minimum cycle basis algorithm is the computation of a minimum weight cycle  $C_i$  such that  $\langle C_i, S_i \rangle = 1$ . Suppose we relax our constraint that our cycle basis should have minimum weight and ask for a cycle basis whose weight is at most  $\alpha$  times the weight of an MCB. Then can we give a faster algorithm?

We show a positive answer to the above question. For any parameter  $\alpha > 1$ , we present below an approximation algorithm which computes a cycle basis whose weight is at most  $\alpha$  times the weight of a minimum cycle basis. To the best of our knowledge, this is the first time that an approximation algorithm for the MCB problem is being given.

This algorithm is obtained by relaxing the base step ( $k = 1$ ) in procedure *extend\_cycle\_basis* of our FAST-MCB algorithm (Figure 4). In the original algorithm, we computed a minimum weight cycle  $C_{i+1}$  such that  $\langle C_{i+1}, S_{i+1} \rangle = 1$ . Here, we relax it to compute a cycle  $D_{i+1}$  such that  $\langle D_{i+1}, S_{i+1} \rangle = 1$  and the weight of  $D_{i+1}$  is at most  $\alpha$  times the weight of a minimum weight cycle that has odd intersection with  $S_{i+1}$ . The method of updating the subsets  $S_i$  would be identical to the way the update is done in FAST-MCB.

We compute a set of cycles  $\{D_1, \dots, D_N\}$  in our approximation algorithm using the following idea (Figure 5).

For  $i = 1$  to  $N$  do the following:

- Let  $S_i$  be any arbitrary non-zero vector in the subspace orthogonal to  $\{D_1, D_2, \dots, D_{i-1}\}$   
i.e.,  $S_i$  satisfies:  $\langle D_k, S_i \rangle = 0$  for  $k = 1$  to  $i - 1$ .
- Compute a cycle  $D_i$  such that  $\langle D_i, S_i \rangle = 1$  and the weight of  $D_i \leq \alpha \cdot$  the weight of a minimum weight cycle that has odd intersection with  $S_i$ .

Figure 5: APPROX-MCB: An  $\alpha$ -approximate MCB

The linear independence of the  $D_i$ 's follows from the existence of  $S_i$ 's. That is,  $\langle D_i, S_i \rangle = 1$  while  $\langle D_k, S_i \rangle = 0$  for all  $k = 1, \dots, i-1$  shows that  $D_i$  is linearly independent of  $D_1, \dots, D_{i-1}$ . Similarly, note that the subsets  $\{S_1, \dots, S_N\}$  are linearly independent since each  $S_i$  is independent of  $\{S_{i+1}, \dots, S_N\}$  because  $\langle S_i, D_i \rangle = 1$  whereas  $\langle S_j, D_i \rangle = 0$  for each  $j > i$ .

Now we would like to prove the correctness of the algorithm in Figure 5. Let  $|C|$  denote the weight of cycle  $C$ . We need to show that  $\sum_{i=1}^N |D_i| \leq \alpha \cdot$  weight of MCB. Let  $A_i$  be a shortest cycle that has odd intersection with  $S_i$ . The set  $\{A_1, \dots, A_N\}$  need not be linearly independent since the subsets  $S_i$ 's were not updated according to the  $A_i$ 's. The following lemma was originally shown in [6] in order to give an equivalent characterisation of the MCB problem as a maximisation problem. We present a simple proof of the lemma here.

**Lemma 4**  $\sum_{i=1}^N |A_i| \leq \text{weight of MCB}$ .

**Proof.** We will look at the  $A_i$ 's in sorted order i.e., let  $\pi$  be a permutation on  $[N]$  such that  $|A_{\pi(1)}| \leq |A_{\pi(2)}| \leq \dots \leq |A_{\pi(N)}|$ . Let  $\{C_1, \dots, C_N\}$  be the cycles of an MCB and let  $|C_1| \leq |C_2| \leq \dots \leq |C_N|$ . We will show that for each  $i, |A_{\pi(i)}| \leq |C_i|$ . That will prove the lemma.

We will first show that  $\langle C_k, S_{\pi(\ell)} \rangle = 1$  for some  $k$  and  $\ell$  with  $1 \leq k \leq i \leq \ell \leq N$ . Otherwise, the  $N - i + 1$  linearly independent vectors  $S_{\pi(i)}, S_{\pi(i+1)}, \dots, S_{\pi(N)}$  belong to the subspace orthogonal to  $C_1, \dots, C_i$ ; however, this subspace has dimension only  $N - i$ .

This means that  $|A_{\pi(\ell)}| \leq |C_k|$  since  $A_{\pi(\ell)}$  is a shortest cycle such that  $\langle A_{\pi(\ell)}, S_{\pi(\ell)} \rangle = 1$ . But by the sorted order,  $|A_{\pi(i)}| \leq |A_{\pi(\ell)}|$  and  $|C_k| \leq |C_i|$ . This implies that  $|A_{\pi(i)}| \leq |C_i|$ . ■

Since  $|D_i| \leq \alpha \cdot |A_i|$  for each  $i$ , it follows from the above lemma that  $\sum_{i=1}^N |D_i| \leq \alpha \cdot$  weight of MCB. Thus Theorem 4 follows.

**Theorem 4** *The linearly independent cycles  $\{D_1, \dots, D_N\}$  computed in APPROX-MCB have weight at most  $\alpha$  times the weight of a minimum cycle basis.*

## 5.1 The running time of APPROX-MCB

Since all the steps of APPROX-MCB, except the base step corresponding to computing a cycle, are identical to FAST-MCB, we have the following recurrence for APPROX-MCB:

$$T(k) = \begin{cases} \text{cost of computing an } \alpha\text{-stretch cycle } D_i \text{ that is odd in } S_i & \text{if } k = 1 \\ 2T(k/2) + O(k^{\omega-1}m) & \text{if } k > 1 \end{cases}$$

So the running time of APPROX-MCB depends on which parameter  $\alpha$  is used in the algorithm. We will compute an  $\alpha$ -stretch cycle  $D_i$  that is odd in  $S_i$  by using the same method as in Section 2.1. But instead of a shortest  $(v^+, v^-)$  path in  $G_i$ , here we would compute an  $\alpha$ -stretch  $(v^+, v^-)$  path. It is easy to see that the minimum of such paths would correspond to an  $\alpha$ -stretch cycle in  $G$  that has odd intersection with  $S_i$ .

When  $\alpha = 2$ , we use the result in [3] to compute 2-stretch paths which would result in 2-stretch cycles. Then APPROX-MCB runs in time  $\tilde{O}(m^{3/2}n^{3/2}) + O(m^\omega)$ . For reasonably dense graphs (say,  $m \geq n^{(1.5+\delta)/(\omega-1.5)}$  for a constant  $\delta > 0$ ), this is an  $O(m^\omega)$  algorithm.

For  $1 + \epsilon$  approximation, we use the all pairs  $(1 + \epsilon)$ -stretch paths algorithm [21]. Then we have an  $\tilde{O}(mn^\omega/\epsilon \log(W/\epsilon)) + O(m^\omega)$  algorithm to compute a cycle basis which is at most  $1 + \epsilon$  times the weight of an MCB, where  $W$  is the largest edge weight in the graph. If  $m \geq n^{1+1/(\omega-1)} \text{poly}(\log n)$  for a constant  $\delta > 0$  and all edge weights are polynomial in  $n$ , then APPROX-MCB is an  $O(m^\omega/\epsilon \log(1/\epsilon))$  algorithm.

## 6 Computing a Certificate of Optimality

We conclude with the problem of constructing a certificate to verify a claim that a given set of cycles  $\mathcal{C} = \{C_1, \dots, C_N\}$  forms an MCB. A certificate is an “easy to verify” witness of the optimality of our answer.

For example, the sets  $S_i$ ,  $1 \leq i \leq N$  in our algorithm from which we calculate the cycles  $\mathcal{C} = \{C_1, \dots, C_N\}$  of the minimum cycle basis, are a certificate of the optimality of  $\mathcal{C}$ . The verification algorithm would consist of verifying that the cycles in  $\mathcal{C}$  are linearly independent and that each  $C_i$  is a minimum weight cycle such that  $\langle C_i, S_i \rangle = 1$ .

Though asymptotically, this verification algorithm and FAST-MCB have the same running time, the constants would be much smaller in the verification algorithm and also this algorithm would be conceptually much simpler. This motivates the following question: given a set of cycles  $\{C_1, \dots, C_N\}$ , compute its certificate.

The following algorithm computes witnesses  $S_1, \dots, S_N$  given  $C_1, \dots, C_N$ .

1. Compute a spanning tree  $T$ . Let  $\{e_1, \dots, e_N\}$  be the edges of  $G \setminus T$ .
2. Form the 0-1  $N \times N$  matrix  $\mathcal{C} = (C_1^T \dots C_N^T)$ , where the  $i$ -th column of  $\mathcal{C}$  is the incidence vector of  $C_i$  over  $\{e_1, \dots, e_N\}$ .
3. Compute  $\mathcal{C}^{-1}$ . The rows of  $\mathcal{C}^{-1}$  are our witnesses or certificate.

If the matrix inversion algorithm returns an error, it means that  $\mathcal{C}$  is singular. That is,  $\{C_1, \dots, C_N\}$  are linearly dependent. Hence, they cannot form a cycle basis.

The rows of  $\mathcal{C}^{-1}$  form our witnesses  $S_1, S_2, \dots, S_N$ . The property that we want from  $S_1, \dots, S_N$  is that for each  $i$ ,  $\langle C_i, S_i \rangle = 1$ . Since  $\mathcal{C}^{-1}\mathcal{C}$  is the identity matrix, this property is obeyed by the rows of  $\mathcal{C}^{-1}$ .

Suppose each  $C_i$  is a minimum weight cycle such that  $\langle C_i, S_i \rangle = 1$ . Then by Lemma 4, this means that  $\sum_{i=1}^N |C_i| \leq \text{weight of an MCB}$ . Since  $\{C_1, \dots, C_N\}$  are linearly independent (by the existence of  $\mathcal{C}^{-1}$ ), it means that  $\{C_1, \dots, C_N\}$  forms a minimum cycle basis.

On the other hand, if for some  $i$ ,  $C_i$  is not a minimum weight cycle such that  $\langle C_i, S_i \rangle = 1$ , then by replacing  $C_i$  with a minimum weight cycle that has odd intersection with  $S_i$  (as in the proof of Theorem 1), we get a cycle basis with smaller weight.

Hence, the cycles  $\{C_1, \dots, C_N\}$  form an MCB if and only if each  $C_i$  is a minimum weight cycle such that  $\langle C_i, S_i \rangle = 1$ . Since the inverse of an  $N \times N$  matrix can be computed in  $O(N^\omega)$  time, we have the following theorem.

**Theorem 5** *Given a set of cycles  $\mathcal{C} = \{C_1, \dots, C_N\}$  we can construct a certificate  $\{S_1, \dots, S_N\}$  in  $O(m^\omega)$  time.*

## 7 Conclusions

In this paper we considered the problem of computing a minimum cycle basis in an undirected graph. We gave an  $O(m^2n + mn^2 \log n)$  algorithm for this problem where  $m$  is the number of edges and  $n$  is the number of vertices. Improved running time estimates were given in special cases like integer edge weights or when the graph is unweighted. We also considered the approximate minimum cycle basis problem. Faster algorithms were presented for this problem using approximate shortest paths algorithms. It would be very interesting to design a faster algorithm for the general problem.

## References

- [1] 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.
- [2] L. O. Chua and L. Chen. On optimally sparse cycle and coboundary basis for a linear graph. In *IEEE Trans. Circuit Theory*, volume CT-20, pages 495–503, 1973.
- [3] E. Cohen and U. Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38:335–353, 2001.
- [4] D. Coppersmith and S. Winograd. Matrix multiplications via arithmetic progressions. *Journal of Symb. Comput.*, 9:251–280, 1990.
- [5] N. Deo, G. M. Prabhu, and M. S. Krishnamoorthy. Algorithms for generating fundamental cycles in a graph. *ACM Trans. Math. Software*, 8:26–42, 1982.
- [6] J.C. de Pina. *Applications of Shortest Path Methods*. PhD thesis, University of Amsterdam, Netherlands, 1995.
- [7] Z. Galil and O. Margalit. All pairs shortest paths for graphs with small integer length edges. *Journal of Computing Systems and Sciences*, 54:243–254, 1997.
- [8] Alexander Golynski and Joseph D. Horton. A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In *8th Scandinavian Workshop on Algorithm Theory*, 2002.
- [9] David Hartvigsen and Russell Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *Journal of Discrete Mathematics*, 7(3):403–418, 1994.

- [10] David Hartvigsen and Russell Mardon. When do short cycles generate the cycle space? *Journal of Combinatorial Theory, Series B*, 57:88–99, 1993.
- [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] E. Hubicka and M. M. Syslo. Minimal bases of cycles of a graph. In M. Fiedler, editor, *Recent Advances in Graph Theory*, pages 283–293, 1975.
- [13] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. A faster algorithm for Minimum Cycle Basis of graphs. In *Proc. of 31st International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 3142: 846–857, 2004.
- [14] E. Kolasinska. On a minimum cycle basis of a graph. *Zastos. Mat.*, 16:631–639, 1980.
- [15] Padberg and Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7:67–80, 1982.
- [16] S. Pettie and V. Ramachandran. Computing shortest paths with comparisons and additions. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 267–276, 2002.
- [17] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computing Systems and Sciences*, 51:400–403, 1995.
- [18] G. F. Stepanec. Basis systems of vector cycles with extremal properties in graphs. *Uspekhi Mat. Nauk*, 19:171–175, 1964.
- [19] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46:362–394, 1999.
- [20] M. Thorup. Floats, integers, and single source shortest paths. *Journal of Algorithms*, 35:189–201, 2000.
- [21] U. Zwick. All pairs shortest paths in weighted directed graphs - exact and approximate algorithms. In *Proc. of the 39th Annual IEEE FOCS*, pages 310–319, 1998.
- [22] A. A. Zykov. *Theory of Finite Graphs*. Nauka, Novosibirsk, 1969.