A faster algorithm for Minimum Cycle Basis of graphs

Telikepalli Kavitha[1⋆], Kurt Mehlhorn[1⋆], Dimitrios Michail[1⋆], Katarzyna Paluch[2⋆⋆]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany.
{kavitha, mehlhorn, michail}@mpi-sb.mpg.de
[2] Institute of Computer Science, University of Wroclaw, Poland.
abraka@ii.uni.wroc.pl.

Abstract. In this paper we consider the problem of computing a minimum cycle basis in a graph G with m edges and n vertices. The edges of G have non-negative weights on them. The previous best result for this problem was an $O(m^{\omega}n)$ algorithm, where ω 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. Our algorithm also uses fast matrix multiplication. 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.

1 Introduction

1.1 The problem

Let $G = (V, E)$ be a graph. A *cycle* of G is any subgraph in which each vertex has even degree. Associated with each cycle is an *incidence vector* x, indexed on E, where $x_e = 1$ if e is an edge of C, $x_e = 0$ otherwise. The vector space over $GF(2)$ 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 $N = m - n + 1$, 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. The weight of a cycle is the sum of the weights of its edges. The weight of a cycle basis is the sum of the weights of its cycles. We consider the problem of computing a cycle basis of minimum weight in a graph. (We use the abbreviation MCB to refer to a minimum cycle basis.)

⋆ Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

⋆⋆ Work done while the author was at MPII supported by Marie Curie Doctoral Fellowship.

1.2 Background

This problem 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. Such algorithms include algorithms for diverse applications like electrical circuit theory [2], structural engineering [1], and periodic event scheduling [5].

The oldest known references to the minimum cycle basis are Stepanec [13] and Zykov [17]. Though polynomial time algorithms for this problem were claimed, these algorithms were not correct [9, 10]. The first polynomial time algorithm for the minimum cycle basis problem was given by Horton [8], and had running time $O(m^3 n)$.

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 [7] 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 ω 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 [5] gave an $O(m^3 + mn^2 \log n)$ to compute an MCB in a graph. The approach in [5] is different from that of Horton; de Pina's algorithm is similar to the algorithm of Padberg and Rao [11] to solve the minimum weighted T-odd cut problem. Our new algorithm to compute an MCB is also based on the same approach.

1.3 New Results

In this paper we obtain the following new results.

For graphs with arbitrary non-negative weights on edges, we give an $O(m^2 n + mn^2 \log n)$ 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 $O(m^2 n)$ algorithm. We use an all pairs shortest paths (APSP) algorithm as a subroutine in our algorithm. We obtain better running times for integer edge weights and unweighted graphs by using faster all pairs shortest path algorithms for these cases [12, 6, 14, 15]

We also look at approximation algorithms for computing a minimum cycle basis in a graph. Given any $\alpha > 1$, we have an α-approximation algorithm by relaxing the shortest paths subroutine to an α stretch paths[3] subroutine. We also show that a witness of a minimum cycle basis can be constructed in $O(m^\omega)$ time.

[3] An α stretch (s, t) path is a path which is at most α times the length of a shortest (s, t) path.

2 A Simple MCB Algorithm

De Pina [5] gave a combinatorial algorithm to compute a minimum cycle basis in a graph with non-negative weights on its edges. We feel that the intuition behind the algorithm and the idea as to why it works is not clear from the combinatorial version of the algorithm. So, we interpret this algorithm algebraically. From the algebraic version of the algorithm, the scope for improvement is also clear.

2.1 An algebraic interpretation

Let $G = (V, E)$ be an undirected graph with m edges and n vertices and with non-negative weights on its edges. We 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. Let T be any spanning tree of G. Let $e_1, \ldots, e_N$ be the edges of $G \setminus T$ in some arbitrary but fixed order.

A cycle in G can be viewed in terms of its incidence vector and 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 these vectors restricted to the coordinates indexed by $\{e_1, ..., e_N\}$.

In SIMPLE-MCB (see Fig. 1) we compute the cycles of a minimum cycle basis and their *witnesses*. A witness S of a cycle C is a subset of $\{e_1, ..., e_N\}$ 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 $\{e_1, ..., e_N\}$.

Hence, both cycles and witnesses are vectors in the space $\{0, 1\}^N$. $\langle C, S\rangle$ stands for the standard inner 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 $GF(2)$, observe that $\langle C, S\rangle = 1$ if and only if C contains an odd number of edges of S. We present in Fig. 1 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
     $\{C_1, C_2, ..., C_{i-1}\}$. That is, $S_i$ is a non-trivial solution to the set of linear
     equations:
                    $\langle C_k, x\rangle = 0$ for $k = 1$ to $i - 1$.
     (Initially, $S_1$ is any arbitrary non-zero vector in the space $\{0, 1\}^N$.)
  2. Compute a shortest cycle $C_i$ such that $\langle C_i, S_i\rangle = 1$.
```

Fig. 1. 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 formed by edges of T and e has intersection of size exactly 1 with S_i. So, there is always at least one cycle with an odd number of edges of S_i.

Note that C_i is independent of $C_1, .., C_{i-1}$ because any vector v in the span of $\{C_1, ..., C_{i-1}\}$ satisfies $\langle v, S_i \rangle = 0$ (since $\langle C_j, S_i \rangle = 0$ for each $1 \leq j \leq i-1$), whereas $\langle C_i, S_i \rangle = 1$. Hence, it follows immediately that $\{C_1, ..., C_N\}$ is a basis.

We still have to describe how to compute a shortest cycle C_i such that $\langle C_i, S_i \rangle = 1$ and how to compute a non-zero vector S_i in the subspace orthogonal to $\{C_1, ..., C_{i-1}\}$. We will do that in Sections 2.2 and 2.3 respectively. We will first prove that $\{C_1, ..., C_N\}$ computed in SIMPLE-MCB forms an MCB.

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

Proof. (from [5]) Suppose not. Then there exists an $0 \leq i < N$ such that there is a minimum cycle basis B that contains $\{C_1, ..., C_i\}$ but there is no minimum cycle basis that contains $\{C_1, ..., C_i, C_{i+1}\}$. Since the cycles in B form a spanning set, there exist cycles $D_1, ..., D_k$ in B such that

$$C_{i+1} = D_1 + D_2 + \cdots + D_k$$

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

Let $B' = B \cup \{C_{i+1}\} \setminus \{D_j\}$. It is easy to see that B' is also a basis. And the weight of B' is at most the weight of B which is a minimum cycle basis. So B' is also a minimum cycle basis. It is easy to show that $\{C_1, C_2, ..., C_{i+1}\} \subseteq B'$ because by assumption $\{C_1, ..., C_i\} \subseteq B$ and the cycle D_j that was omitted from B cannot be equal to any one of $C_1, ..., C_i$ because $\langle D_j, S_{i+1} \rangle = 1$ whereas $\langle C_j, S_{i+1} \rangle = 0 \ \forall j \leq i$.

The existence of the basis B' contradicts that there is no minimum cycle basis containing $\{C_1, ..., C_i, C_{i+1}\}$. Hence, $\{C_1, C_2, ..., C_N\}$ is indeed a minimum cycle basis. □

2.2 Computing the cycles

Given S_i, it is easy to compute a shortest 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.

G_i has two copies v^+ and v^- of each vertex $v \in V$. For each edge $e = (u, v) \in E$ do: if $e \notin S_i$, then add edges (u^+, v^+) and (u^-, v^-) to the edge set of G_i and assign their weights to be the same as e. If $e \notin S_i$, then add edges (u^+, v^-) and (u^-, v^+) to the edge set of G_i and assign their weights to be the same as e. 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.

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. Because we identify both v^+ and v^- with v, the path in G corresponding to p would be a cycle C.

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 more than one copy of some edge(s). (It could have contained both (v^+, u^-) and (v^-, u^+) for some (v, u).) Then add the number of occurrences of each such 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. The following lemma is simple to show.

Lemma 1. *The path p corresponds to a shortest cycle C in G that has odd intersection with S_i.*

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. 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 $C_1, C_2, ..., C_N$, we spend $O(mn(m + n \log n))$ time, since $N = m - n + 1$.

2.3 Computing the subsets

We will now consider the problem of computing the subsets S_i, for $i = 1$ to N. S_i is a non-zero vector in the subspace orthogonal to $\{C_1, ..., C_{i-1}\}$. 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 $\{S_i, S_{i+1}, ..., S_N\}$ which is a basis of the space $\mathcal{C}^\perp$ orthogonal to the space $\mathcal{C}$ spanned by $\{C_1, ..., C_{i-1}\}$. We use S_i to compute C_i and update $\{S_{i+1}, ..., S_N\}$ to a basis $\{S'_{i+1}, ..., S'_N\}$ of the subspace of $\mathcal{C}^\perp$ which is orthogonal to C_i. The update step of phase i is as follows:

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

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

The following lemma holds.

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

This completes the description of the algorithm SIMPLE-MCB.

Running Time of SIMPLE-MCB: During the update step of phase i, the cost of updating each $S_j, j > i$ is N and hence it is $N(N-i)$ for updating $S_{i+1}, ..., S_N$. Since we have N phases, 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, independent of which all-pairs-shortest-paths algorithm is used to compute the cycles, the cost of updating the witnesses is the bottleneck.

Note that in each phase we needed just one vector from the subspace orthogonal to $\{C_1, ..., C_i\}$. But the algorithm maintained $N-i$ such vectors: $S_{i+1}, ..., S_N$. This was the limiting factor in the running time of the algorithm.

3 Our improvement

The maintenance of the basis of $\mathcal{C}^{\perp}$ costed us m^2 in each iteration. In order to improve the running time of SIMPLE-MCB, we relax the invariant that $S_{i+1}, ..., S_N$ form a basis of the subspace orthogonal to $C_1, ..., C_i$. Since we need just one vector in this subspace, we can afford to relax this invariant and maintain the correctness of the algorithm. We will use a function *extend_cycle_basis* to compute the minimum cycle basis. This function works in a recursive manner.

The procedure *extend_cycle_basis*$(\{C_1, ..., C_i\}, \{S_{i+1}, \ldots, S_{i+k}\}, k)$ takes a partial basis $C_1, ..., C_i$ and k subsets $S_{i+1}, ..., S_{i+k}$ with the property that these subsets are all orthogonal to $C_1, ..., C_i$ and it recursively computes k new elements $C_{i+1}, ..., C_{i+k}$ of the minimum cycle basis. It first computes $C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}$ using $S_{i+1}, \ldots, S_{i+\lfloor k/2 \rfloor}$. Then it updates $S_{i+\lfloor k/2 \rfloor+1}, \ldots, S_{i+k}$ so that the updated sets are orthogonal to $C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}$ and they continue to be orthogonal to $C_1, ..., C_i$. Then it computes $C_{i+\lfloor k/2 \rfloor+1}, \ldots, C_{i+k}$. We present in Fig. 2 the overall algorithm FAST-MCB and the procedure *extend_cycle_basis*. Recall that the edges $e_1, ..., e_N$ are the edges of $G \setminus T$, where T is a spanning tree of G.

3.1 The function *update*:

The function *update* is the key subroutine in our procedure *extend_cycle_basis*. After computing the cycles $C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}$, we call the function *update* with $\{S'_{i+1}, \ldots, S'_{i+\lfloor k/2 \rfloor}\}$ (the final versions of the subsets $S_{i+1}, \ldots, S_{i+\lfloor k/2 \rfloor}$) and $\{S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}\}$) as inputs. We want to update the sets $S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}$ so that the updated sets lie in the subspace orthogonal to the space spanned by $\mathcal{C} \cup \{C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}\}$. We know that $S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}$ are all orthogonal to $\mathcal{C}$ and now we need to ensure that the updated $S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}$ (call them $T_{i+\lfloor k/2 \rfloor+1}, \ldots, T_{i+k}$) are all orthogonal to $\mathcal{C} \cup \{C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}\}$.

We now want to update the sets $S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}$, i.e., we want to determine $T_{i+\lfloor k/2 \rfloor+1}, \ldots, T_{i+k}$ such that for each j in the range for $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$: (i) T_j is orthogonal to $C_{i+1}, \ldots, C_{i+\lfloor k/2 \rfloor}$ and (ii) T_j continues to remain

- 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*$(\{\}, \{S_1, \ldots, S_N\}, N)$.

(A call to *extend_cycle_basis*$(\{C_1, ..., C_i\}, \{S_{i+1}, \ldots, S_{i+k}\}, k)$ extends the cycle basis by k cycles. $\mathcal{C}$ denotes the current partial cycle basis which is $\{C_1, ..., C_i\}$.)

The procedure *extend_cycle_basis*$(\mathcal{C}, \{S_{i+1}, \ldots, S_{i+k}\}, k)$:

```
– if $k = 1$, compute a shortest cycle $C_{i+1}$ such that $\langle C_{i+1}, S_{i+1} \rangle = 1$.
– if $k > 1$, use recursion.
    • call extend_cycle_basis$(\mathcal{C}, \{S_{i+1}, \ldots, S_{i+\lfloor k/2 \rfloor}\}, \lfloor k/2 \rfloor)$ to extend the current
      cycle basis by $\lfloor k/2 \rfloor$ elements. That is, the cycles $C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}$ are computed
      in a recursive manner. During the above recursive call, $S_{i+1}, ..., S_{i+\lfloor k/2 \rfloor}$ get updated.
      Call their final versions as $S'_{i+1}, ..., S'_{i+\lfloor k/2 \rfloor}$.
    • call update$(\{S'_{i+1}, \ldots, S'_{i+\lfloor k/2 \rfloor}\}, \{S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}\})$ to update
      $\{S_{i+\lfloor k/2 \rfloor+1}, ..., S_{i+k}\}$. Let $\{T_{i+\lfloor k/2 \rfloor+1}, ..., T_{i+k}\}$ be the output returned by update.
    • call extend_cycle_basis$(\mathcal{C} \cup \{C_{i+1}, ..., C_{i+\lfloor k/2 \rfloor}\}, \{T_{i+\lfloor k/2 \rfloor+1}, \ldots, T_{i+k}\}, \lceil k/2 \rceil)$
      to extend the current cycle basis by $\lceil k/2 \rceil$ cycles. That is, the cycles
      $C_{i+\lfloor k/2 \rfloor+1}, ..., C_{i+k}$ will be computed recursively.
```

Fig. 2. FAST-MCB: A faster minimum cycle basis algorithm

orthogonal to $C_1, ..., C_i$. So, we define T_j (for each $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$) as follows:

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

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

Let

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

We will determine the coefficients $a_{j1}, ..., a_{j\lfloor k/2 \rfloor}$ for all $i + \lfloor k/2 \rfloor + 1 \leq j \leq i + k$ simultaneously.

We want

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

where A is a $\lceil k/2 \rceil \times \lfloor k/2 \rfloor$ matrix whose ℓth row has the unknowns $a_{j1}, ..., a_{j\lfloor k/2 \rfloor}$, 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,

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

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

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

where

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

Then

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

If X is invertible, then $A = -YX^{-1} = YX^{-1}$ since we are in $GF(2)$. We can determine A in k^ω time using fast matrix multiplication and inverse algorithms.

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

is an upper diagonal 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 $A = YX^{-1}$.

Lemma 3 follows from the implementation of the function *update*.

Lemma 3. *When $k = 1$, i.e., whenever we call extend_cycle_basis($\{C_1, ..., C_i\}$, $S_{i+1}, 1$), S_{i+1} is orthogonal to $\{C_1, ..., C_i\}$. 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 $\{C_1, ..., C_i\}$. And C_{i+1} is a shortest cycle such that $\langle C_{i+1}, S_{i+1}\rangle = 1$. Hence, the correctness of FAST-MCB follows then from Theorem 1.

3.2 The running time of FAST-MCB

The recurrence of our FAST-MCB algorithm is as follows:

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

Cost of update: The computation of matrices X and Y takes time $mk^{\omega-1}$ using the fast matrix multiplication algorithm. We can also invert X in $O(k^\omega)$ time and then we use fast matrix multiplication to multiply Y and X^{-1} to get the matrix A. Then we use fast matrix multiplication again to multiply the matrix $(A\ I)$ with the matrix whose rows are $S'_{i+1}, ...S_{i+k}$ to get the updated subsets $T_{i+\lfloor k/2\rfloor+1}, ...T_{i+k}$. So the time required for all these computations is $O(mk^{\omega-1})$.

Using the algorithm described in Section 2.2 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 $T(m) = O(mn^2 \log n)$.

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 $O(mn)$ [14, 15], 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 $\tilde{O}(n^\omega)$ [12, 6]. When such graphs are reasonably dense, say $m \geq n^{1+(1+\delta)/(\omega-1)}$, then the m^ω term dominates the running time of our algorithm.

Theorem 3. *A minimum cycle basis in a graph with integer edge weights can be computed in time $O(m^2n)$. For unweighted graphs that satisfy $m \geq n^{1+(1+\delta)/(\omega-1)}$ for a constant $\delta > 0$, we have an $O(m^\omega)$ algorithm to compute a minimum cycle basis.*

4 An approximation algorithm for Minimum Cycle Basis

The bottleneck in the running time of our minimum cycle basis algorithm is the computation of the shortest 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 α 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 α 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 (Fig. 2). In the original algorithm, we computed a shortest 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 α times the weight of a shortest cycle that has odd intersection with S_{i+1}. The method of updating the subsets S_i would be identical to the way the updation is done in FAST-MCB.

A succinct description of our algorithm is given in Fig. 3.

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

  – Let $S_i$ be any arbitrary non-zero vector in the subspace orthogonal to
    $\{D_1, D_2, ..., D_{i-1}\}$ i.e., $S_i$ is a non-trivial solution to the set of equations:
                        $\langle D_k, x \rangle = 0$ 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 shortest cycle that has odd intersection with $S_i$.
```

Fig. 3. APPROX-MCB: An α-approximate MCB

The linear independence of the D_i's follows from the existence of S_i's (by using S_i to show that D_i is linearly independent of $\{D_1, ..., D_{i-1}\}$). Similarly, note that the subsets $\{S_1, ..., S_N\}$ are linearly independent since each S_i is independent of $\{S_{i+1}, ..., S_N\}$ because $\langle S_i, D_i \rangle = 1$ whereas $\langle S_j, D_i \rangle = 0$ for each $j > i$.

4.1 Correctness of APPROX-MCB

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 $\{A_1, ..., A_N\}$ 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 [5] 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. $\sum_{i=1}^{N} |A_i| \leq$ *weight of MCB.*

Proof. We will look at the A_i's in sorted order i.e., let π be a permutation on $[N]$ such that $|A_{\pi(1)}| \leq |A_{\pi(2)}| \leq ... \leq |A_{\pi(N)}|$. Let $\{C_1, ..., C_N\}$ be the cycles of an MCB and let $|C_1| \leq |C_2| \leq ... \leq |C_N|$. 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 ℓ with $1 \leq k \leq i \leq \ell \leq N$. Otherwise, the $N - i + 1$ linearly independent vectors $S_{\pi(i)}, S_{\pi(i+1)}, ..., S_{\pi(N)}$ belong to the subspace orthogonal to $C_1, ..., C_i$; 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 weight of the basis $\{D_1, ..., D_N\}$ computed by APPROX-MCB is at most α times the weight of a minimum cycle basis.*

4.2 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:

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

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 [16]. Then we have an $\tilde{O}(\frac{mn^{\omega}}{\epsilon} \log(W/\epsilon)) + O(m^{\omega})$ 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 $m \geq n^{1+(1+\delta)/(\omega-1)}$ for a constant $\delta > 0$ and all edge weights are polynomial in n, then APPROX-MCB is an $O(\frac{m^{\omega}}{\epsilon} \log(1/\epsilon))$ algorithm.

5 Computing a Certificate of Optimality

Given a set of cycles $\mathcal{C} = \{C_1, ..., C_N\}$ we would like to construct a certificate to verify the claim that $\mathcal{C}$ 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 $\mathcal{C} = \{C_1, ..., C_N\}$ 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 shortest cycle such that $\langle C_i, S_i \rangle = 1$.

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

This theorem follows from an algorithm that inverts a matrix whose rows are the incidence vectors of $C_1, ..., C_N$ over the edges of $G \setminus T$ (T is a spanning tree of G).

Acknowledgment. We wish to thank Jaikumar Radhakrishnan for his helpful comments.

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. J.C. de Pina. *Applications of Shortest Path Methods.* PhD thesis, University of Amsterdam, Netherlands, 1995.
6. 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.
7. 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.
8. J. D. Horton. A polynomial-time algorithm to find a shortest cycle basis of a graph. *SIAM Journal of Computing*, 16:359–366, 1987.
9. 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.
10. E. Kolasinska. On a minimum cycle basis of a graph. *Zastos. Mat.*, 16:631–639, 1980.
11. Padberg and Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7:67–80, 1982.
12. R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computing Systems and Sciences*, 51:400–403, 1995.
13. G. F. Stepanec. Basis systems of vector cycles with extremal properties in graphs. *Uspekhi Mat. Nauk*, 19:171–175, 1964.
14. M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46:362–394, 1999.
15. M. Thorup. Floats, integers, and single source shortest paths. *Journal of Algorithms*, 35:189–201, 2000.
16. 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.
17. A. A. Zykov. *Theory of Finite Graphs.* Nauka, Novosibirsk, 1969.