A Polynomial Time Algorithm for Minimum Cycle Basis in Directed Graphs

Telikepalli Kavitha and Kurt Mehlhorn*

Max-Planck-Institut für Informatik,
Saarbrücken, Germany
{kavitha, mehlhorn}@mpi-sb.mpg.de

Abstract. We consider the problem of computing a minimum cycle basis in a directed graph G with m arcs and n vertices. The arcs of G have non-negative weights assigned to them. We give an $\tilde{O}(m^4n)$ algorithm, which is the first polynomial time algorithm for this problem. We also present an $\tilde{O}(m^3n)$ randomized algorithm. The problem of computing a minimum cycle basis in an *undirected* graph has been well-studied. However, it is not known if an efficient algorithm for undirected graphs automatically translates to an efficient algorithm for directed graphs.

1 Introduction

1.1 The Problem

Let $G = (V, A)$ be a directed graph with vertex set V and arc set A (no self-loops). We will consider cycles in the underlying undirected graph and assign each such cycle C a vector in $\{-1, 0, 1\}^{|A|}$. This incidence vector, also called C, is defined as follows. For each arc $a \in A$

$$C(a) = \begin{cases} 1 & \text{if } C \text{ traverses } a \text{ in forward direction} \\ -1 & \text{if } C \text{ traverses } a \text{ in backward direction} \\ 0 & \text{if } a \notin C \end{cases}$$

The *cycle space* of G is the vector space over $\mathbb{Q}$ that is spanned by these incidence vectors. The cycle space of a connected digraph has dimension $d = m - n + 1$, where $|A| = m$ and $|V| = n$. A *cycle basis* of G is a set of cycles $C_1, ..., C_d$ whose incidence vectors permit a unique linear combination of the incidence vector of any cycle of G.

We assume that there is a weight function $w : A \to \mathbb{R}^+$, i.e., the arcs of G have non-negative weights assigned to them. The weight of a cycle is the sum of the weights of its arcs. The weight of a cycle basis is the sum of the weights of its cycles. A *minimum cycle basis* of G is a cycle basis with minimum weight, that is, a cycle basis $\mathcal{B}$ such that $\sum_{C \in \mathcal{B}} \sum_{a \in C} w(a)$ is minimum. We consider the problem of computing a minimum cycle basis in a given digraph.

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

V. Diekert and B. Durand (Eds.): STACS 2005, LNCS 3404, pp. 654–665, 2005.

1.2 Background

The importance of the problem of computing a minimum cycle basis lies in its use as a preprocessing step in several algorithms. That is, a cycle basis is generally not wanted for its own sake, but to be used as an input for a later algorithm. And the importance of a minimum cycle basis is to reduce the amount of work that has to be done by this later algorithm. In the problem of computing a minimum cycle basis of an undirected graph $U = (N, E)$, with each cycle we associate a $\{0, 1\}$ 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 these vectors is called the *cycle space* of U. A minimum cycle basis of U is a set of linearly independent (over GF(2)) cycles that span the cycle space of U and whose sum of weights is minimum. The problem of computing a minimum cycle basis in undirected graphs has been well-studied [2, 5, 7, 8, 9, 10] and the current fastest algorithm for computing a minimum cycle basis in an undirected graph with m edges and n vertices runs in $O(m^2n + mn^2 \log n)$ time [10].

In many cases the network graphs of interest are intrinsically directed. For a directed graph G, we obtain the underlying undirected graph of G by removing the directions from the arcs. A set of cycles $C_1, ..., C_d$ of G projects onto an undirected cycle basis, if by removing the orientations of the arcs in the cycles, we obtain a cycle basis for the underlying undirected graph. It was shown by Liebchen and Peeters in [11] that if $\mathcal{C} = \{C_1, ..., C_d\}$ is a set of cycles in a directed graph G that projects onto an undirected cycle basis, then $\mathcal{C}$ is a cycle basis of G. But the the converse is not true. Similarly, a minimum cycle basis of a digraph need not project onto a cycle basis of the underlying undirected graph. The books by Deo [6] and Bollobás [3] have an in-depth coverage of the subject of cycle bases.

Our Results. In this paper we give an an $\tilde{O}(m^3n)$ randomized algorithm and an $\tilde{O}(m^4n)$ deterministic algorithm to compute a minimum cycle basis in a digraph $G = (V, A)$ where $|A| = m$ and $|V| = n$. Very recently, Liebchen and Rizzi [12] have also given an $\tilde{O}(m^4n)$ deterministic algorithm to compute a minimum cycle basis in a directed graph. They adapt Horton's greedy approach [9] and using fast matrix multiplication, their algorithm can be implemented in $\tilde{O}(m^{\omega+1}n)$ time, where ω is the best exponent of matrix multiplication. Our approach is complementary to theirs. Our algorithms use simple linear algebra and elementary number theory and are in the domain of arithmetical algorithms. The techniques used here might be of independent interest.

2 The Algorithm

Our algorithm is broadly based on the approach used in [5, 2, 10] for computing a minimum cycle basis in an undirected graph. The basic idea is to have an iterative algorithm that computes a new cycle C_i of the minimum cycle basis in the i-th iteration. There is no loss of generality in assuming that the underlying undirected graph of G is connected. Then $d = m - n + 1$ is the dimension of the cycle space of G. We can assume that $m \geq 2$.

2.1 The Basic Idea

Recall that each cycle in G is encoded as a $\{-1, 0, 1\}$ vector in $\mathbb{Q}^m$. Let $\langle S, C\rangle = \sum_{i=1}^{m} s_i c_i$ denote the standard inner product between $S = (s_1, ..., s_m)$ and $C = (c_1, ..., c_m)$, which are vectors in the space $\mathbb{Q}^m$. A high-level description of our algorithm is as follows.

For $i = 1, ..., d$ do:

1. let $S_i \in \mathbb{Q}^m$ be a non-zero vector such that $\langle S_i, C_j\rangle = 0$ for all j where $j < i$.
2. compute C_i to be a shortest cycle in G such that $\langle S_i, C_i\rangle \neq 0$.

That is, S_i is a non-zero vector orthogonal to the cycles computed in the first $i-1$ iterations. And the shortest cycle which is *not* orthogonal to S_i is C_i. Before we get into the details of how to implement these steps, let us first check if this approach gives us what we seek.

Theorem 1. *The set $\{C_1, \ldots, C_d\}$ is a minimum cycle basis of G.*

Proof. It is easy to see that C_i is linearly independent of $\{C_1, ..., C_{i-1}\}$. S_i is a witness of this linear independence since $\langle S_i, C_j\rangle = 0$ for all $j < i$, so the inner product of S_i with any linear combination of $C_1, \ldots, C_{i-1}$ has to be zero but $\langle S_i, C_i\rangle \neq 0$. Hence the whole set $\{C_1, \ldots, C_d\}$ is linearly independent.

Suppose $\{C_1, \ldots, C_d\}$ does not form a minimum cycle basis. Then there exists a minimal i such that $\{C_1, \ldots, C_i\} \not\subseteq$ any minimum cycle basis. So $\{C_1, \ldots, C_{i-1}\} \subseteq$ some minimum cycle basis $\mathcal{B}$. Then

$$C_i = \lambda_1 B_1 + \lambda_2 B_2 + \cdots + \lambda_l B_l \text{ where each } \lambda_t \in \mathbb{Q} \text{ and each } \lambda_t \neq 0,$$

for some $\{B_1, \ldots, B_l\} \subseteq \mathcal{B}$. Since $\langle S_i, C_i\rangle \neq 0, \exists B_k \in \{B_1, ..., B_l\}$ such that $\langle S_i, B_k\rangle \neq 0$. Then by the very definition of C_i, it follows that weight$(B_k) \geq$ weight(C_i). Hence $\mathcal{B}' = B \cup \{C_i\} \setminus \{B_k\}$ is also a minimum cycle basis. The cycle B_k that has been omitted from $\mathcal{B}'$ cannot be one of $C_1, ..., C_{i-1}$ since the inner product of each of $C_1, ..., C_{i-1}$ with S_i is zero whereas $\langle S_i, B_k\rangle \neq 0$. Hence, $\{C_1, ..., C_i\} \subseteq \mathcal{B}'$, which is a minimum cycle basis - a contradiction. □

So our basic idea works. Let us now consider how to implement the two steps in the basic idea.

2.2 Implementation

Computing a shortest cycle C_i such that $\langle S_i, C_i\rangle \neq 0$ for $S_i \in \mathbb{Q}^m$ can be reduced to computing a shortest cycle C_i such that $\langle S_i, C_i\rangle \neq 0$ for $S_i \in \mathbb{Z}^m$. So let us look at the following implementation. More specifically, in the i-th iteration:

Step 1. Compute $S_i \in \mathbb{Z}^m$ such that S_i is a nontrivial solution to the set of equations:

$$\langle \boldsymbol{x}, C_j\rangle = 0 \ \forall j < i.$$

We will show that we can find an S_i with $\|S_i\|_\infty \leq 2^{f(i)}$, where $f(i)$ is $O(i \log i)$.

Step 2. Compute $f(i)+1$ distinct primes $p_0, ..., p_{f(i)}$, where each $p_t \geq m$.
For $t = 0, ..., f(i)$ do:
- compute a shortest cycle B_t such that $\langle S_i, B_t \rangle \neq 0 \pmod{p_t}$.
- Now we have a list (probably, a multiset) of cycles $(B_0, \ldots, B_{f(i)})$.

$$C_i := \min(B_0, \ldots, B_{f(i)}).$$

That is, C_i is assigned to be that cycle which has the least weight in this list. If there is more than one cycle with the same least weight, then C_i can be any one of such cycles.

Lemma 1. *C_i is a shortest cycle in G such that $\langle S_i, C_i \rangle \neq 0$.*

Proof. Suppose there is a shorter cycle D_i. Since D_i is not in the list$(B_0, ..., B_{f(i)})$, it must be the case that $\langle S_i, D_i \rangle = 0 \pmod{p_0}, \pmod{p_1}, \cdots, \pmod{p_{f(i)}}$. This forces $\langle S_i, D_i \rangle$ to be a multiple of $\Pi_t p_t$ since all the p_t's are distinct primes. Since each $p_t \geq m$, $\Pi_t p_t > m^{f(i)+1}$.

Since $\|S_i\|_\infty \leq 2^{f(i)}$ and D_i is a vector in $\{-1, 0, 1\}^m$, we have

$$|\langle S_i, D_i \rangle| \leq m 2^{f(i)} \leq m^{f(i)+1} < \Pi_t p_t.$$

So the only way $\langle S_i, D_i \rangle$ can be a multiple of $\Pi_t p_t$ is that $\langle S_i, D_i \rangle = 0$.

Hence, any cycle D_i with a lesser weight than C_i necessarily has to obey $\langle S_i, D_i \rangle = 0$. □

A question that needs to be answered is why should there always be some cycle C_i such that $\langle C_i, S_i \rangle \neq 0$. We will show that the S_i that we compute has the property that such a cycle always exists.

2.3 Computing S_i

Let us first order the arcs in the arc set A so that $a_{d+1}, ..., a_m$ form the edges of a spanning tree T of the underlying undirected graph. This means that in the incidence vector representation of cycles, the first d coordinates correspond to arcs $a_1, \ldots, a_d$ which are outside the tree T and the last $n-1$ coordinates are the arcs of T.

This will enable us to maintain the invariant that each S_i is of the form $(s_{i1}, \ldots, s_{ii}, 0, \ldots, 0)$ with $s_{ii} \neq 0$. So only the first i coordinates of S_i can be non-zero and s_{ii} has to be non-zero. The fundamental cycle F_i formed by the adding the arc a_i to the edges of the spanning tree T has the incidence vector $(0, \ldots, 0, 1, 0, \ldots, 0, *, \ldots, *)$. That is, in the first d coordinates only $F_i(a_i) \neq 0$ and the $*$'s, which take $\{-1, 0, 1\}$ values, are in the last $n-1$ coordinates. $\langle F_i, S_i \rangle = s_{ii} \neq 0$. Hence, there is always at least one cycle whose inner product with S_i is non-zero.

In the first iteration, S_1 is any non-zero vector. So we assign S_1 to be the vector $(1, 0, \ldots, 0)$. Thus S_1 satisfies our invariant. In the i-th iteration we need to find a nontrivial solution to the set of equations $\langle \boldsymbol{x}, C_j \rangle = 0 \ \forall j < i$. We do this as follows.

– compute a vector $(r_1, ..., r_{i-1}, 1, 0, ...0) \in \mathbb{Q}^m$ that is orthogonal to C_j for each $j < i$.

Let the j-th cycle C_j have the incidence vector $(c_{j1}, ..., c_{jm})$. Since the vector $(r_1, ..., r_{i-1}, 1, 0, ...0)$ is orthogonal to C_j,

$$\sum_{k=1}^{i-1} c_{jk} r_k = -c_{ji}, \text{ for } 1 \leq j \leq i-1.$$

Let $\tilde{C}_j = (c_{j1}, ...c_{j(i-1)})$ be the restriction of C_j to its first $i-1$ coordinates. So $(r_1, ..., r_{i-1})$ is a solution to the set of equations:

$$\tilde{C}_j \cdot (x_1, ..., x_{i-1}) = -c_{ji} \text{ for } j = 1, ..., i-1.$$

A solution always exists to the above set of equations because $\tilde{C}_1, \ldots, \tilde{C}_{i-1}$ are linearly independent. Suppose the linear combination

$$\sum_{j=1}^{i-1} \alpha_j \tilde{C}_j = \mathbf{0} \tag{1}$$

and not all α_j are 0. Then consider the largest k such that $\alpha_k \neq 0$ and take the inner product of both sides of Equation (1) with that $\tilde{S}_k$, where $\tilde{S}_k$ is the restriction of the vector S_k to its first $i-1$ coordinates. (Note that $\tilde{S}_k$ has all the non-zero entries of S_k for each $1 \leq k \leq i-1$. So $\langle \tilde{C}_j, \tilde{S}_k \rangle = \langle C_j, S_k \rangle$.) Then the left hand side is $\sum_{j=1}^{k} \alpha_j \langle C_j, S_k \rangle = \alpha_k \langle C_k, S_k \rangle$ since $\langle C_j, S_k \rangle = 0$ for all $j < k$. Since α_k and $\langle C_k, S_k \rangle$ are non-zero while the right hand side is zero, we get a contradiction. Hence each $\alpha_j = 0$ for $1 \leq j \leq i-1$.

Thus the $(i-1) \times (i-1)$ matrix of $\tilde{C}$'s which has $\tilde{C}_1, \ldots, \tilde{C}_{i-1}$ as its rows is invertible and so there exists a unique solution to the set of equations:

$$\begin{pmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_{i-1}^T \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} -c_{1i} \\ \vdots \\ -c_{(i-1)i} \end{pmatrix} \tag{2}$$

Let $(r_1, \ldots, r_{i-1})$ be the solution to the above set of equations. Then $S'_i = (r_1, \ldots, r_{i-1}, 1, 0, ..., 0)$ is a vector in $\mathbb{Q}^m$ that is orthogonal to $C_1, ..., C_{i-1}$.

By Cramer's rule, each r_j is of the form $r_j = y_j/k$, where k is the determinant of the matrix of $\tilde{C}$'s (call this matrix $\mathcal{M}_i$) and y_j is the determinant of the matrix obtained by replacing the j-th column of $\mathcal{M}_i$ by the vector on the right hand side of Equation (2). In order to get an integral vector S_i from S'_i, we multiply S'_i with k. So $S_i = kS'_i = (y_1, ..., y_{i-1}, k, 0, ..., 0)$ is an integral vector that is orthogonal to all the cycles $C_1, ..., C_{i-1}$. And we have also maintained our invariant that S_i has non-zero entries in only its first i coordinates and its i-th coordinate is non-zero. Equivalently, $(y_1, ..., y_{i-1})$ is the (integral) solution to the set of equations:

$$\tilde{C}_j \cdot \boldsymbol{x} = -kc_{ji} \text{ for } j = 1, ..., i-1. \tag{3}$$

Let us now bound the L_∞ norm of S_i. Since k is the determinant of an $(i-1)\times(i-1)$ matrix whose entries are $-1, 0$ or 1, using Hadamard's inequality we get

$$|k| \leq \Pi_{j=1}^{i-1}\|\tilde{C}_j\| \leq \Pi_{j=1}^{i-1}\sqrt{i} \leq 2^{(i\log i)/2}.$$

Similarly, each $|y_j| \leq 2^{(i\log i)/2}$. Hence $\max\{y_1,\ldots,y_j,\ldots,k\} \leq 2^{(i\log i)/2}$. Thus we have shown that $\|S_i\|_\infty \leq 2^{f(i)}$, where $f(i) = (i\log i)/2$.

The vector $(y_1,\ldots,y_{i-1})$ can be obtained by Gaussian elimination, or by multiplying the matrix $\mathcal{M}_i^{-1}$ with the column vector $(-kc_{1i},\ldots,-kc_{(i-1)i})$. These computations can be implemented in $O(i^\omega)$ steps, where ω is the best exponent of matrix multiplication. We also need to account for the cost of performing arithmetic operations, since we do arithmetic on large numbers. Assuming that arithmetic on $O(\ell\log\ell)$ bits takes $O(\ell)$ time, we have the following lemma.

Lemma 2. *A nontrivial vector $S_i \in \mathbb{Z}^m$ such that $\langle S_i, C_j\rangle = 0\ \forall j < i$ and $\|S_i\|_\infty \leq 2^{f(i)}$ can be computed in $O(m^{\omega+1})$ time.*

3 Computing B_t

In order to compute a shortest cycle whose inner product with S_i is non-zero modulo p_t, we build an undirected graph $U_{i,t}$ using the given directed graph G, the vector S_i and the number p_t. The graph $U_{i,t}$ can be visualised as p_t levels of the digraph G. Call these levels as level 0,..., level (p_t-1). Each level has a copy of every vertex $v \in V$. Let v_j be the copy of vertex v in level j. The edge set of $U_{i,t}$ also consists of p_t copies of each arc $a \in A$. The edges corresponding to arc $a = (u,v)$ are (u_j, v_k) where $k = (j + S_i(a))$ modulo p_t for each $j = 0, 1, \ldots, p_t - 1$.

That is, the edge in $U_{i,t}$ that corresponds to copy j (for $j = 0,\ldots,p_t-1$) of the arc $a = (u,v)$ of G goes from the copy of vertex u in level j to the copy of vertex v in level $(j + S_i(a))$ mod p_t. Also, each edge (u_j, v_k) in $U_{i,t}$ inherits the weight of its corresponding arc (u,v) of G.

Thus there is a well-defined map from the vertex set of $U_{i,t}$ to the vertex set V of G and from the edge set of $U_{i,t}$ to the arc set A of G. We can extend this map to paths of $U_{i,t}$. So given any path q in $U_{i,t}$, we can map q to a *chain*[1] in G by mapping the vertices and edges of q to their images in G.

Lemma 3 captures the essence of the graph $U_{i,t}$ and Lemma 4 gives us an efficient way of computing the desired cycle. These lemmas are simple to show and their proofs will be included in the full version of the paper.

Lemma 3. *Any (v_r, v_s) path in $U_{i,t}$, whose edges map to distinct arcs of G, maps to a cycle C in G. The incidence vector of such a cycle C satisfies $\langle C, S_i\rangle = \pm(s-r) (\mathrm{mod} p_t)$.*

[1] A chain is an alternating sequence of vertices and arcs $(x_0, a_1, x_1, a_2, ..., a_r, x_r)$ such that either $a_k = (x_{k-1}, x_k)$ or $a_k = (x_k, x_{k-1})$.

Relabel the arcs of G so that $a_{d+1}, \ldots, a_m$ are the arcs of a spanning tree of the underlying undirected graph.
Compute distinct primes $p_0, \ldots, p_{f(m)}$, where each prime $\geq m$. {This can be done by a sieving algorithm.}
for $i = 1, \ldots, d$ **do**
  Compute $S_i = (s_{i1}, \ldots, s_{ii}, 0, \ldots, 0) \in \mathbb{Z}^m$ such that $s_{ii} \neq 0$ and $\langle S_i, C_j \rangle = 0$ for all $j < i$.
  for $t = 0, \ldots, f(i)$ **do**
    Compute the graph $U_{i,t}$ from G using S_i and p_t (as described in Section 3).
    Let $q = \min_v \min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path in $U_{i,t}$. Let B_t be the cycle in G that the path q corresponds to.
  end for
  $C_i = \min(B_0, \ldots, B_{f(i)})$.
end for
Return $\{C_1, \ldots, C_d\}$.

Fig. 1. Algorithm-MCB: Algorithm to compute a minimum cycle basis in a digraph

Lemma 4. *Let $q = \min_v \min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path[2] in the graph $U_{i,t}$. Then q corresponds to a shortest cycle B_t in G such that $\langle B_t, S_i \rangle \neq 0 \pmod{p_t}$.*

Remark. Whenever $S_i \bmod p_t$ is not the zero vector, then there is always a path in $U_{i,t}$ between v_0 and v_ℓ for some $v \in V$ and $\ell \neq 0$. If $S_i \bmod p_t$ is the zero vector, then q does not exist and so there would be no cycle B_t in the list $(B_0, \ldots, B_{f(i)})$. Indeed, there can be no cycle in G whose inner product with S_i is non-zero modulo p_t, given that $S_i \bmod p_t$ is the zero vector.

Cost of Computing B_t. Computation of the path q can be accomplished by a shortest paths computation in the graph $U_{i,t}$ from each vertex v_0 in level 0 and taking the shortest $(v_0, v_\ell), \ell \neq 0$ path over all $v \in V$. This can be done in $O(n(p_t m + p_t n \log p_t n))$ time since one single-source shortest paths computation in $U_{i,t}$ would take $O(p_t m + p_t n \log p_t n)$ time by Dijkstra's algorithm.

The value of $\pi(r)$, the number of primes less than r, is given by $r/6 \log r \leq \pi(r) \leq 8r/\log r$ [1]. So each of the primes p_t can be bounded from above by $O(f(m) \log m)$. Hence we have shown the following lemma.

Lemma 5. *We can compute a shortest cycle B_t such that $\langle B_t, S_i \rangle \neq 0 \pmod{p_t}$ in $\tilde{O}(nmf(m))$ time.*

3.1 The Entire Algorithm

A summary of our algorithm to compute a minimum cycle basis in $G = (V, A)$ is given in Fig. 1. The correctness of the algorithm follows from Lemmas 1, 3, 4 and Theorem 1. Lemmas 2 and 5 ensure polynomial running time of the algorithm.

[2] In case of many paths tied for the minimum, choose q to be any of these paths that has the least number of edges.

Running Time. Recall that the cost of computing S_i is $O(m^{\omega+1})$ (by Lemma 2). This is $o(m^3 n)$ since $\omega < 2.376$ [4]. The limiting factor in the running time of the i-th iteration is the computation of the cycles $B_0, \ldots, B_{f(i)}$. Since each of them can be computed in $\tilde{O}(nmf(m))$ time (by Lemma 5), the time required to compute C_i is $\tilde{O}(nmf(m)f(i))$. Since $f(i)$ is $O(i \log i)$, the i-th iteration takes $\tilde{O}(m^3 n)$ time. Thus Theorem 2 immediately follows.

Theorem 2. *Algorithm-MCB computes a minimum cycle basis of G in $\tilde{O}(m^4 n)$ time.*

4 An $\widetilde{O}(m^3 n)$ Randomized Algorithm

In this section we present a Monte Carlo algorithm to compute a minimum cycle basis in G. The underlying ideas are the same as in Algorithm-MCB, except that we will use the primes $p_0, \ldots, p_{f(m)}$ more sparingly now.

Let us call a prime $p \in \{p_0, \ldots, p_{f(i)}\}$ a *witness* of C_i if $\langle S_i, C_i \rangle \neq 0 \pmod{p}$. Lemma 1 shows that since $\|S_i\|_\infty \leq 2^{f(i)}$, there is at least one witness of C_i in $\{p_0, \ldots, p_{f(i)}\}$. We can easily extend this idea to get the following lemma.

Lemma 6. *If $p_1, \ldots, p_{2f(i)}$ are distinct primes where each prime $\geq m$, then C_i has at least $f(i)$ witnesses in $\{p_1, \ldots, p_{2f(i)}\}$.*

So a prime p chosen uniformly at random from $\{p_1, \ldots, p_{2f(i)}\}$ has probability $\geq 1/2$ of being a witness of C_i. So instead of computing $f(i)+1$ cycles $B_0, \ldots, B_{f(i)}$ and taking their minimum as C_i, we could first sample a few primes with uniform distribution from $\{p_0, \ldots, p_{2f(i)}\}$ and compute the cycles corresponding only to these few sampled primes. We will call the minimum of these cycles as C_i. If the number of sampled primes is $\text{poly}(\log m)$, then we spend only $\tilde{O}(m^2 n)$ time to compute C_i now. But of course, we have introduced some error. So this C_i need not always be the cycle that we seek, however we can bound the error probability.

The more difficult problem is to efficiently compute $S_i = (s_{i1}, \ldots, s_{ii}, 0, \ldots, 0)$ in $\mathbb{Z}^m$ where $s_{ii} \neq 0$ and $\langle S_i, C_j \rangle = 0$ for all $j < i$. We can no longer afford to spend $\Theta(m^{\omega+1})$ time to compute S_i now.

4.1 Computing S_i More Efficiently

The important observation is that we do not really need S_i, what we need is $S_i \bmod p_t$, that is, the vector $(s_{i1} \bmod p_t, \ldots, s_{ii} \bmod p_t, 0, \ldots, 0)$ in order to compute B_t. If $q_0, \ldots, q_r$ are the few sampled primes of iteration i, then $S_i \bmod q_0, \ldots, S_i \bmod q_r$ are the vectors that we need. We had computed S_i as $(y_1, \ldots, y_{i-1}, k, 0, \ldots, 0)$ where $(y_1, ..., y_{i-1})$ is the (integral) solution to the set of equations:

$$\tilde{C}_j \cdot \boldsymbol{x} = -k c_{ji} \text{ for } j = 1, ..., i-1$$

(this is Equation (3) from Section 2.3). The integer $k = \det(\mathcal{M}_i)$, where $\mathcal{M}_i$ denotes the matrix of $\tilde{C}$'s on the left of the above equation. Now we want

- Compute $3f(m)$ distinct primes $p_1, \ldots, p_{3f(m)}$, where each prime $\geq m$.
- For $i = 1, \ldots d$ do:
  1. initialize $Q = \emptyset$.
  2. For $j = 1, \ldots, \lfloor \log^2 m \rfloor$ do:
     - let r_j be a random element of $\{p_1, \ldots, p_{3f(m)}\} \setminus \{r_1, \ldots, r_{j-1}\}$ picked with uniform distribution.
     - if $\det(\mathcal{M}_i) \neq 0 \pmod{r_j}$, then $Q = Q \cup r_j$.
     ($\mathcal{M}_i$ is the $(i-1) \times (i-1)$ matrix discussed above. $\mathcal{M}_1$ is the empty matrix; let its determinant be ∞.)
  3. if $|Q| \leq \lceil \log m \rceil$, then declare *failure* and exit the program.
  4. For each $q_t \in Q$
     - compute in the field $\mathbb{Z}_{q_t}$: $\ell_i = \det(\mathcal{M}_i)$ and $(\ell_1, \ldots, \ell_{i-1}) = \mathcal{M}_i^{-1} b$.
     - set $S_i \bmod q_t = (\ell_1, \ldots, \ell_i, 0, \ldots, 0)$. (And when $i = 1$, we set $S_1 \bmod q_t = (1, 0, \ldots, 0)$.)
     - compute $B_t =$ shortest cycle such that $\langle B_t, S_i \bmod q_t \rangle \neq 0 \pmod{q_t}$.
  5. $C_i = \min(B_1, \ldots, B_{|Q|})$.
- Return $\{C_1, \ldots, C_d\}$.

Fig. 2. Randomized-MCB: Randomized algorithm to compute a minimum cycle basis in a digraph

to determine $S_i \bmod p$ directly, where p is a sampled prime. Let $S_i \bmod p = (\ell_1, \ldots, \ell_i, 0, \ldots, 0)$. It follows from Equation (3) that $(\ell_1, \ldots, \ell_{i-1})$ satisfies the set of equations:

$$\tilde{C}_j \cdot \boldsymbol{x} = -\ell_i c_{ji} \quad \text{for } j = 1, ..., i-1 \quad \text{in the field} \quad \mathbb{Z}_p \tag{4}$$

where $\ell_i = \det(\mathcal{M}_i) (\mathrm{mod} p)$. Whenever $\ell_i \neq 0$, $\mathcal{M}_i^{-1} b \pmod{p}$ is the unique solution of $\mathcal{M}_i \boldsymbol{x} = b$ in $\mathbb{Z}_p$, where b denotes the column vector of $-\ell_i c_{ji}$'s of Equation (4). Then we can determine $S_i \bmod p$ directly by computing $\det(\mathcal{M}_i)$ and $\mathcal{M}_i^{-1} b$ in the field $\mathbb{Z}_p$.

We know that $\det(\mathcal{M}_i) \neq 0$ and that $|\det(\mathcal{M}_i)| \leq 2^{f(i)}$ from Hadamard's inequality (see Section 2.3). So, by exactly the same argument as in the proof of Lemma 1, we can show the following lemma.

Lemma 7. *If $p_1, \ldots, p_{f(i)}$ are distinct primes, then for at least one prime p in $\{p_1, \ldots, p_{f(i)}\}$, $\det(\mathcal{M}_i) \neq 0 (\mathrm{mod} p)$.*

Call such a prime p a *witness* of $\mathcal{M}_i$. Again we can extend this argument as in Lemma 6 to show that if $p_1, \ldots, p_{3f(m)}$ are distinct primes, then $\mathcal{M}_i$ has at most $f(i) - 1$ *non-witnesses* in $\{p_1, \ldots, p_{3f(m)}\}$. So if we take a sample P of $\lfloor \log^2 m \rfloor$ elements from $\{p_1, \ldots, p_{3f(m)}\}$, each choice made uniformly at random (without replacement), then we can show that with high probability, there are at least $\lceil \log m \rceil$ witnesses for $\mathcal{M}_i$ in P. For each of these witnesses p, we can compute $S_i \bmod p$ in $O(m^\omega)$ time because arithmetic in $\mathbb{Z}_p$ takes $O(1)$ time. So the total time spent to compute $S_i \bmod p$ for all the elements in P is $O(m^\omega |P|)$ or $\tilde{O}(m^\omega)$. This is $o(m^2 n)$.

Based on these ideas, we have the Monte Carlo algorithm presented in Fig. 2. In some runs Randomized-MCB declares "failure" and does not return any set of cycles. In other runs it returns a set of cycles $\{C_1, \ldots, C_d\}$. It is easy to see that these cycles are always linearly independent, but they may not always be a minimum cycle basis. Call iteration i a "success" if in iteration i, the cycle C_i that Randomized-MCB computes is indeed a shortest cycle whose inner product with S_i is non-zero. Let A_i denote the event that iteration i is a success. When the event $A_1 \cap \cdots \cap A_d$ occurs, then Randomized-MCB remains faithful to the basic idea (Section 2.1) and so the cycle basis $\{C_1, \ldots, C_d\}$ computed by the algorithm is indeed a minimum cycle basis. So the probability that Randomized-MCB outputs a minimum cycle basis is $\Pr(A_1 \cap \cdots \cap A_d)$.

Lemma 8. *For each* $1 \leq i \leq d$, $\Pr[A_i | (A_1 \cap \cdots \cap A_{i-1})] \geq 1 - 1/m$.

Proof. For iteration i to begin, it must be the case that in the first $i-1$ iterations, the algorithm did not declare "failure" and exit the program. The event $A_1 \cap \cdots \cap A_{i-1}$ ensures that this is indeed the case. So iteration i begins and iteration i is a success whenever the random sample of $\lfloor \log^2 m \rfloor$ primes, that we pick in iteration i, has at least $\lceil \log m \rceil + 1$ witnesses of $\mathcal{M}_i$ and among these witnesses of $\mathcal{M}_i$, there is at least one witness of C_i. We can bound from below the probability that this event occurs by upper bounding the complement event. The complement is the union of two events: (i) the event that $|Q| \leq \lceil \log m \rceil$ and (ii) the event that Q has no witnesses of C_i given that $|Q| \geq \lceil \log m \rceil + 1$.

Let us look at the first of these two events. We know that there are at most $f(i) - 1$ non-witnesses of $\mathcal{M}_i$ in $\{p_1, \ldots, p_{3f(m)}\}$. In iteration i, when we pick the first random element r_1, the probability that r_1 is not a witness of $\mathcal{M}_i$ is at most $(f(i) - 1)/(3f(m))$. The j-th random element r_j is chosen uniformly at random from $\{p_1, \ldots, p_{3f(m)}\} \setminus \{r_1, \ldots, r_{j-1}\}$. The probability that r_j is a witness of $\mathcal{M}_i$ depends on how many of $\{r_1, \ldots, r_{j-1}\}$ are witnesses of $\mathcal{M}_i$. But we do not need this exact value. We can easily see that for each $j = 1, \ldots, \lfloor \log^2 m \rfloor$

$$\Pr[r_j \text{ is not a witness for } \mathcal{M}_i] \leq \frac{f(i) - 1}{3f(m) - j + 1} \leq \frac{m}{3m - \log m} \leq \frac{1}{2}$$

So the probability that there are exactly $\lceil \log m \rceil$ witnesses of $\mathcal{M}_i$ in Q is at most $\binom{\log^2 m}{\lceil \log m \rceil} \cdot (1/2)^{\lfloor \log^2 m \rfloor - \lceil \log m \rceil}$. Hence the probability that there are at most $\lceil \log m \rceil$ witnesses of $\mathcal{M}_i$ in Q is upper bounded by $\lceil \log m \rceil + 1$ times this number. This value is at most

$$(\lceil \log m \rceil + 1)(\log^2 m)^{\lceil \log m \rceil} \frac{4}{m^{\log m - 1}} < \frac{1}{2m} \quad \forall m \geq \text{ some constant } m_0.$$

(Also, we will modify Randomized-MCB so that for small m, i.e. when $m < m_0$, we do no random sampling - so Randomized-MCB is identical to our deterministic algorithm for small m.)

Let us now look at the second event that contributes to the failure of iteration i. Given that $|Q| > \lceil \log m \rceil$, we would like to upper bound the probability that Q has no witnesses of C_i. There are at least $3f(m) - f(i) \geq 2f(m)$ witnesses of

C_i in $\{p_1, \ldots, p_{3f(m)}\}$. $\mathcal{M}_i$ also has at least $2f(m)$ witnesses in $\{p_1, \ldots, p_{3f(m)}\}$. So at least half the witnesses of $\mathcal{M}_i$ are also witnesses of C_i. So the probability that a random subset Q of witnesses of $\mathcal{M}_i$ contains no witness of C_i is at most $(1/2)^{|Q|} \leq 1/2m$ since $|Q| \geq \log m + 1$.

So the total error probability is at most $1/2m + 1/2m = 1/m$. Hence $\Pr[A_i|(A_1 \cap \cdots \cap A_{i-1})] \geq 1 - 1/m$, for each $1 \leq i \leq d$. □

Since $\Pr(A_1 \cap \cdots \cap A_d) = \Pi_{i=0}^{d} \Pr[A_i|(A_1 \cap \cdots \cap A_{i-1})]$, Lemma 8 shows that the success probability of Randomized-MCB is at least $(1 - 1/m)^d > (1 - 1/m)^m \approx 1/e$. Hence, by running Randomized-MCB a constant number of times and taking the cycle basis whose weight is the least, we can make the error probability less than δ for any given constant $\delta > 0$. The running time of the algorithm follows from the discussion at the beginning of Section 4. Hence Theorem 3 follows.

Theorem 3. *A minimum cycle basis can be computed with high probability in* $\tilde{O}(m^3 n)$ *time.*

5 Further Analysis

In this section we would like to prove that *any* minimum cycle basis from the set of all minimum cycle bases of G has a chance to be returned as $\{C_1, \ldots, C_d\}$ by a variant of Algorithm-MCB (Fig. 1). First, fix any spanning tree T of the underlying undirected graph of G and let $\{a_1, \ldots, a_d\}$ be the arcs of $G \setminus T$. Let $\{D_1, \ldots, D_d\}$ be some minimum cycle basis of G. Let us assume that these cycles are sorted by their weights. So we have $\text{weight}(D_1) \leq \cdots \leq \text{weight}(D_d)$. Let us form the $d \times d$ matrix $\mathcal{D}$ whose i-th column is the incidence vector of D_i restricted to the arcs $\{a_1, \ldots, a_d\}$. It is simple to show that $\mathcal{D}$ is a nonsingular matrix. The next observation is that the rows of $\mathcal{D}$ can be permuted so that for each i, the $i \times i$ submatrix consisting of the first i rows and first i columns is nonsingular. Alternately, the LUP decomposition of $\mathcal{D}$ gives us a permutation matrix P such that $P\mathcal{D}$ has this property. Permuting the rows of $\mathcal{D}$ is just renumbering the arcs $a_1, \ldots, a_d$.

Let us now describe the slight variation in Algorithm-MCB so that we can claim that $\{D_1, \ldots, D_d\}$ can be returned by our algorithm. The variation is that we do an extra step, right at the beginning, where we permute the order of the arcs in $G \setminus T$. That is, we generate a random permutation σ on $\{1, \ldots, d\}$ and the order of coordinates in the incidence vectors of cycles will be $a_{\sigma(1)}, \ldots, a_{\sigma(d)}, a_{d+1}, \ldots, a_m$. After this step, the rest of the algorithm is the same as Algorithm-MCB (Fig. 1). In the case of a tie while computing a shortest path or shortest cycle, let us assume that the algorithm breaks ties randomly so that each of the candidate cycles tied as the shortest cycles has a chance to be picked.

Our algorithm has a tree of possible executions and we want to show that there is at least one execution where $\{D_1, \ldots, D_d\}$ is computed as a minimum cycle basis. In the first step let us assume that the permutation π was generated,

where π is the permutation corresponding to the permutation matrix P in the LUP decomposition of $\mathcal{D}$. Using the special property of the $d \times d$ matrix $P\mathcal{D}$, that is, for each i, its leading $i \times i$ submatrix is nonsingular, we can show that for $1 \leq i \leq d$, there is a vector $L_i = (\ell_{i1}, \ldots, \ell_{ii}, 0, \ldots, 0)$ in $\mathbb{Q}^m$ such that (i) $\langle L_i, D_j \rangle = 0$ for all j where $j < i$ and (ii) D_i is a shortest cycle such that $\langle L_i, D_i \rangle \neq 0$.

The above statement is the crux of the argument and Theorem 4 follows directly from this. The details will be given in the full version of the paper.

Theorem 4. *The minimum cycle basis* $\{D_1, \ldots, D_d\}$ *can be returned by the modified Algorithm-MCB.*

Acknowledgment. We thank Jaikumar Radhakrishnan for his helpful comments.

References

1. T. M. Apostol. *Introduction to Analytic Number Theory.* Springer-Verlag, 1997.
2. F. Berger, P. Gritzmann, and S. de Vries. Minimum Cycle Bases for Network Graphs. *Algorithmica*, 40(1): 51-62, 2004.
3. B. Bollobás. *Modern Graph Theory* volume 184 of *Graduate Texts in Mathematics*, Springer, Berlin, 1998.
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. N. Deo. *Graph Theory with Applications to Engineering and Computer Science* Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, 1982.
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. 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.
9. J. D. Horton. A polynomial-time algorithm to find a shortest cycle basis of a graph. *SIAM Journal of Computing*, 16:359–366, 1987.
10. T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. A faster algorithm for Minimum Cycle Basis of graphs. *In Proc. of ICALP*, LNCS 3142: 846-857, 2004.
11. C. Liebchen and L. Peeters. On Cyclic Timetabling and Cycles in Graphs. Technical Report 761/2002, TU Berlin.
12. C. Liebchen and R. Rizzi. A Greedy Approach to compute a Minimum Cycle Basis of a Directed Graph. Technical Report 2004/31, TU Berlin.