An $\tilde{O}(m^2n)$ Randomized Algorithm to compute a Minimum Cycle Basis of a Directed Graph

T. Kavitha

Indian Institute of Science
Bangalore, India
kavitha@csa.iisc.ernet.in

Abstract. We consider the problem of computing a minimum cycle basis in a directed graph G. The input to this problem is a directed graph whose arcs have positive weights. In this problem a $\{-1,0,1\}$ incidence vector is associated with each cycle and the vector space over $\mathbb{Q}$ generated by these vectors is the cycle space of G. A set of cycles is called a cycle basis of G if it forms a basis for its cycle space. A cycle basis where the sum of weights of the cycles is minimum is called a minimum cycle basis of G. The current fastest algorithm for computing a minimum cycle basis in a directed graph with m arcs and n vertices runs in $\tilde{O}(m^{\omega+1}n)$ time (where $\omega < 2.376$ is the exponent of matrix multiplication). If one allows randomization, then an $\tilde{O}(m^3n)$ algorithm is known for this problem. In this paper we present a simple $\tilde{O}(m^2n)$ randomized algorithm for this problem.
The problem of computing a minimum cycle basis in an *undirected* graph has been well-studied. 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 the graph. It is not known if an efficient algorithm for undirected graphs automatically translates to an efficient algorithm for directed graphs. The fastest known algorithm for computing a minimum cycle basis in an undirected graph runs in $O(m^2n + mn^2 \log n)$ time and our randomized algorithm for directed graphs almost matches this running time.

1 Introduction

Let $G = (V,A)$ be a directed graph with m arcs and n vertices. A *cycle* C in G consists of *forward arcs* C^+ and *backward arcs* C^- such that $C = C^+ \dot{\cup} C^-$ and reorienting all arcs in C^- results in a closed path. Associated with each cycle is a $\{-1,0,1\}$ vector, indexed on the arc set A. This vector, also called C, is defined as follows. For each arc $a \in A$

$$C(a) = \begin{cases} 1 & \text{if } a \text{ is a forward arc of } C \\ -1 & \text{if } a \text{ is a backward arc of } C \\ 0 & \text{if } a \notin C \end{cases}$$

The *cycle space* of G is the vector space over $\mathbb{Q}$ that is generated by the incidence vectors of cycles in G. When G is connected, the cycle space has dimension $d = m - n + 1$. A *cycle basis* of G is a basis of the cycle space of G.

The arcs of G have positive 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. In this paper we consider the problem of computing a minimum cycle basis in a given digraph.

1.1 Background

The problem of computing a minimum cycle basis in a graph is well-studied. Apart from its interest as a natural question, it is motivated by 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 structural engineering [4], cycle analysis of electrical networks [5], and chemical ring perception [7]. And in many cases the network graphs of interest are directed graphs. The problem of computing a minimum cycle basis in a directed graph is related to the problem of computing a minimum cycle basis in an undirected graph. In an undirected graph $U = (N, E)$, with each cycle C we associate a $\{0,1\}$ 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 $\mathbb{F}_2$ generated by these vectors is called the *cycle space* of U. A minimum cycle basis of U is a set of linearly independent (over $\mathbb{F}_2$) cycles that span the cycle space of U and whose sum of weights is minimum.

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. 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 Appendix contains such examples. The books by Deo [6] and Bollobás [3] have an in-depth coverage of the subject of cycle bases.

Previous Results. Algorithms for computing a minimum cycle basis in undirected graphs have been well-studied [2, 5, 9, 10, 12] and the current fastest algorithm for this problem runs in $O(m^2n + mn^2 \log n)$ time [12], where m is the number of edges and n is the number of vertices. The first polynomial time algorithm for computing a minimum cycle basis in a directed graph had a running time of $\tilde{O}(m^4n)$ [11]. Liebchen and Rizzi [15] gave an $\tilde{O}(m^{\omega+1}n)$ algorithm for this problem, where $\omega < 2.376$ is the exponent of matrix multiplication. This is the current fastest deterministic algorithm known for this problem. But faster randomized algorithms are known. Kavitha and Mehlhorn [11] gave an $\tilde{O}(m^3n)$ Monte Carlo algorithm for this problem.

Algorithms for cycle bases of directed graphs have also been studied in [13, 14, 8]. The focus in these papers is on special classes of cycle bases, like integral cycle bases [13, 14] - these are cycle bases whose $d \times m$ cycle-arc incidence matrix has the property that its regular $d \times d$ submatrices have determinant ± 1 (such cycle bases of minimum length are important in cyclic timetabling). In [8] algorithms for computing cycle bases in strongly connected digraphs that consist of cycles which always follow the directions of the arcs (these cycles have no backward arcs) were studied. Such cycle bases are of particular interest in metabolic flux analysis.

1.2 What is new?

In this paper we present a simple $O(m^2n \log n)$ randomized algorithm to compute a minimum cycle basis in a directed graph G with m arcs and n vertices. This algorithm always returns a cycle basis and we show that with high probability this cycle basis is a minimum cycle basis. We obtain this algorithm through an effective use of randomization, which enables us to work entirely over the finite field $\mathbb{F}_p$ for a randomly chosen prime p instead of working over $\mathbb{Q}$.

We recall here that the $\tilde{O}(m^3n)$ randomized algorithm given in [11] works by sampling $\log^2 m$ random primes independently in each iteration of the algorithm. Then the algorithm either uses at least $\log m$ *suitable* primes from this sample to compute cycles in that iteration or it quits if there is no large enough subset of suitable primes in this sample.

In Section 2.1 we first present a deterministic algorithm, whose correctness is simple to show. But the numbers used in this algorithm grow very large and so this algorithm is not interesting from an implementation point of view. However, the analysis of this algorithm leads to our efficient randomized algorithm. Section 2.2 contains our randomized algorithm and its analysis.

A key step in our algorithm is a subroutine to compute a shortest cycle whose inner product with a given vector is nonzero modulo p. Such a subroutine was also used in [11] and we first review that method in Section 3.1. However that subroutine is not good enough for us since using it would make the running time of our algorithm $\tilde{O}(m^4n)$. Here we modify Dijkstra's algorithm for this particular problem and improve this subroutine. Section 3.2 contains this implementation. This leads to an $O(m^3 + m^2 n \log n)$ randomized algorithm for computing a minimum cycle basis in a directed graph.

We can improve the running time even further. As mentioned earlier, the current fastest algorithm for computing a minimum cycle basis in an undirected graph has a running time of $O(m^2n + mn^2 \log n)$. This running time is achieved through the use of fast matrix multiplication to speed up certain operations on vectors. In Section 4 we use the same technique to get rid of the m^3 term in our running time and we thus get an $O(m^2 n \log n)$ algorithm.

2 An $\tilde{O}(m^3)$ randomized algorithm

Our algorithm is broadly based on the approach used in [5, 2, 12, 11] for computing a minimum cycle basis. We are given a digraph $G = (V, A)$, where $|V| = n$ and $|A| = m$. 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. The notation $\langle v_1, v_2 \rangle$ denotes the standard inner product or dot product of the vectors v_1 and v_2.

First we will assume that we have ordered the arcs in the arc set A so that the arcs $a_{d+1}, ..., a_m$ 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 outside the tree T and the last $n-1$ coordinates are the arcs of T.

Before we present our randomized algorithm, let us first consider the following deterministic algorithm. The deterministic algorithm is not an efficient algorithm and we are not interested in its computational complexity. But its analysis leads to our randomized algorithm, which is presented in Section 2.2.

2.1 Deterministic-MCB

1. Initialize the vectors $S_1, \ldots, S_d$ of $\mathbb{Q}^m$ to the first d vectors $e_1, \ldots, e_d$ of the standard basis of $\mathbb{Q}^m$. (The vector e_i has 1 in the i-th position and 0's elsewhere.)
2. For $i = 1$ to d do
   - compute C_i to be a shortest cycle such that $\langle C_i, S_i \rangle \neq 0$.
   - for $j = i + 1$ to d do

$$\text{update } S_j \text{ as: } S_j = S_j - S_i \frac{\langle C_i, S_j \rangle}{\langle C_i, S_i \rangle}$$

Lemma 1. *For $1 \leq i \leq d-1$ the above algorithm maintains the following invariant: at the end of the i-th iteration the vectors $S_{i+1}, \ldots, S_d$ are orthogonal to the cycles $C_1, \ldots, C_i$.*

Proof. We will prove this by induction on i. At the end of the first iteration $S_j = S_j - S_1 \frac{\langle C_1, S_j \rangle}{\langle C_1, S_1 \rangle}$ for each $j = 2, \ldots, d$. The inner product

$$\begin{aligned} \langle C_1, S_j \rangle &= \langle C_1, S_j \rangle - \langle C_1, S_1 \rangle \frac{\langle C_1, S_j \rangle}{\langle C_1, S_1 \rangle} \\ &= 0 \end{aligned}$$

This proves the base case. Suppose the above invariant is maintained till the end of iteration k. So at the beginning of the $(k+1)$-th iteration $S_{k+1}, ..., S_d$ are orthogonal to $C_1, ..., C_k$. After the update step

of the $(k+1)$-th iteration, it is easy to see that $S_{k+2},...,S_d$ are orthogonal to C_{k+1}. But each of the new $S_{k+2},...,S_d$ is a linear combination of the corresponding old $S_{k+2},...,S_d$ and S_{k+1} - but all these vectors are already orthogonal to $C_1,...,C_k$. Hence their linear combinations are also orthogonal to $C_1,...,C_k$. Thus we maintain our invariant at the end of the $(k+1)$-th iteration. □

It is easy to see that the set $\{C_1,\ldots,C_d\}$ is a minimum cycle basis.

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

Proof. (from [5, 11]) Suppose $\{C_1,\ldots,C_d\}$ does not form a minimum cycle basis. Then there exists a minimal $i \geq 1$ 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 $\mathcal{C}$. Then

$$C_i = \lambda_1 K_1 + \lambda_2 K_2 + \cdots + \lambda_l K_l \text{ for some } \{K_1,\ldots,K_l\} \subseteq \mathcal{C} \text{ and } \lambda_1,\ldots,\lambda_l \in \mathbb{Q} \text{ and each } \lambda_t \neq 0.$$

Since $\langle S_i, C_i\rangle \neq 0, \exists K_t \in \{K_1,...,K_l\}$ such that $\langle S_i, K_t\rangle \neq 0$. Then by the very definition of C_i, it follows that weight$(K_t) \geq$ weight(C_i). Hence $\mathcal{C}' = \mathcal{C} \cup \{C_i\} \setminus \{K_t\}$ is also a minimum cycle basis. The cycle K_t that has been omitted from $\mathcal{C}'$ 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 $\langle S_i, K_t\rangle \neq 0$. Hence, $\{C_1,...,C_i\} \subseteq \mathcal{C}'$, which is a minimum cycle basis - a contradiction. □

Let us now understand the structure of the vectors S_j in Deterministic-MCB. The vector S_j gets updated in each iteration till iteration j. Call the version of S_j at the beginning of iteration i as S_j^i. And S_j^j is finally used in iteration j to compute the cycle C_j. (Let us denote the final version S_j^j by S_j itself.) S_j^i has the form $(r_1, r_2, ..., r_{i-1}, 0, ..., 0, 1, 0, ..., 0)$, where $r_1, ..., r_{i-1}$ are some rational numbers and the 1 occurs in the j-th coordinate. Since S_j^i is orthogonal to $C_1, ..., C_{i-1}$, we have $C_k \cdot (r_1, \ldots, r_{i-1}, 0, \ldots, 1, 0, \ldots)^T = 0$ for $k = 1, \ldots, i-1$.

Let the incidence vector of C_k be $(c_{k1}, \ldots, c_{km})$ and let $\tilde{C}_k$ be the restriction of this vector to its first $i-1$ coordinates. Then $(r_1, ..., r_{i-1})$ is a solution to

$$\tilde{C}_k \cdot (x_1, ..., x_{i-1})^T = -c_{kj} \text{ for } k = 1, ..., i-1. \tag{1}$$

We will show that this set of equations has a unique solution. Suppose the linear combination

$$\sum_{j=1}^{i-1} \alpha_j \tilde{C}_j = 0 \tag{2}$$

and not all α_j are 0. Then consider the largest t such that $\alpha_t \neq 0$ and take the inner product of both sides of Equation (2) with $\tilde{S}_t$, where $\tilde{S}_t$ is the restriction of the vector S_t to its first $i-1$ coordinates.

Then the left hand side is $\sum_{k=1}^{t} \alpha_k \langle \tilde{C}_k, \tilde{S}_t\rangle = \sum_{k=1}^{t} \alpha_k \langle C_k, S_t\rangle$ since $\tilde{S}_t$ has all the non-zero entries of S_t for each $1 \leq t \leq i-1$. This is equal to $\alpha_t \langle C_t, S_t\rangle$ since $\langle C_k, S_t\rangle = 0$ for $k < t$. Since $\langle C_t, S_t\rangle \neq 0$ and the right hand side is 0 we get $\alpha_t = 0$ - a contradiction. Hence each α_k has to be 0 for $1 \leq k \leq i-1$. So the $\tilde{C}_k$'s are linearly independent. So we can conclude the following lemma.

Lemma 2. *The $(i-1) \times (i-1)$ matrix $\mathcal{M}_i$ whose k-th row is the vector $\tilde{C}_k$ for $1 \leq k \leq i-1$ is nonsingular.*

Thus Equation (1) has a unique solution, which is $(r_1, \ldots, r_{i-1})$. By Cramer's rule, each r_l is of the form $r_l = y_l/k_i$, where k_i is the determinant of $\mathcal{M}_i$ and y_l is the determinant of the matrix obtained by replacing the l-th column of $\mathcal{M}_i$ by the vector on the right hand side of Equation (1). So multiplying S_j^i with k_i gives us an integral vector $N_j^i = (y_1, ..., y_{i-1}, 0, ..., k_i, 0, ...)$. Since k_i is the determinant of an $(i-1) \times (i-1)$

matrix whose entries are $-1, 0, 1$, it follows from Hadamard's inequality that $|k_i| \leq (i-1)^{\frac{i-1}{2}}$. Similarly, the absolute value of each y_l is bounded from above by $(i-1)^{\frac{i-1}{2}}$. So we have $\|N_i\|_1 \leq i(i-1)^{\frac{i-1}{2}} \leq d^{\frac{d+1}{2}}$ since $i \leq d$. Let us denote each N_j^j by N_j, respectively.

Definition 1. *Call a prime p* good *if for each $i = 1, ..., d$: $\langle C_i, N_i \rangle \neq 0 (\bmod p)$. Call a prime p* bad *if it is not good.*

Lemma 3. *Let P be a set of d^2 primes, each of which is at least d^2. Then at least $3/4$-th of the set P is good.*

Proof. For any i, $\langle C_i, S_i \rangle \neq 0$ is equivalent to $\langle C_i, N_i \rangle \neq 0$ since $N_i = k_i S_i$ and $k_i = \det(\mathcal{M}_i) \neq 0$ by Lemma 2. So for each $1 \leq i \leq d$, it holds that $\langle C_i, N_i \rangle \neq 0$. Since C_i is a $\{-1, 0, 1\}$ vector we also get that $|\langle C_i, N_i \rangle| \leq \|N_i\|_1$. So $|\langle C_i, N_i \rangle| \leq d^{\frac{d+1}{2}}$.

Since $N_1 = S_1 = (1, 0, \ldots, 0)$, the number $\langle C_1, N_1 \rangle$ is always ± 1. So no prime can divide it. For $i \geq 2$, we will use $0 \neq |\langle C_i, N_i \rangle| \leq d^{\frac{d+1}{2}}$. Since each prime in P is at least d^2, at most $(d+1)/4$ elements in P can be divisors of $\langle C_i, N_i \rangle$. So the number of primes in P that can divide at least one of $\langle C_2, N_2 \rangle, \langle C_3, N_3 \rangle, \ldots, \langle C_d, N_d \rangle$ is at most $(d-1)(d+1)/4$. Hence the fraction of bad primes in P is at most $(d-1)(d+1)/4d^2 < 1/4$. □

Now we present the algorithm Randomized-MCB. This is similar to Deterministic-MCB. But here we work over the field $\mathbb{F}_p$ for a randomly chosen prime from the set P, instead of working over $\mathbb{Q}$.

2.2 Randomized-MCB

1. Compute a set P of d^2 primes $p_0, p_1, \ldots$ where each $p_j \geq d^2$. Choose a prime p uniformly at random from this set.
2. Initialize the vectors $X_1, \ldots, X_d$ of $\mathbb{F}_p^m$ to the first d vectors of the standard basis $e_1, \ldots, e_d$.
3. For $i = 1$ to d do
   - compute B_i to be a shortest cycle such that $\langle B_i, X_i \rangle \neq 0 (\bmod p)$.
   - for $j = i+1$ to d do

$$\text{update } X_j \text{ (over the finite field } \mathbb{F}_p) \text{ as: } X_j = X_j - X_i \frac{\langle B_i, X_j \rangle}{\langle B_i, X_i \rangle}$$

(The proof of Lemma 1 shows that $X_{i+1}, \ldots, X_d$ are now orthogonal to $B_1, \ldots, B_i$ over $\mathbb{F}_p$.)

We will now show that $\{B_1, \ldots, B_d\}$ is always a cycle basis.

Lemma 4. *The cycles $\{B_1, ..., B_d\}$ are linearly independent.*

Proof. We know that $\langle B_j, X_i \rangle = 0 (\bmod p)$ for all $j < i$. It is now easy to see that B_i is linearly independent of $\{B_1, ..., B_{i-1}\}$ over $\mathbb{F}_p$. X_i is a witness of this linear independence since $\langle B_j, X_i \rangle = 0 (\bmod p)$ for each $j < i$, so the inner product of X_i with any linear combination of $B_1, \ldots, B_{i-1}$ has to be zero modulo p but $\langle B_i, X_i \rangle \neq 0 (\bmod p)$. Hence the whole set $\{B_1, \ldots, B_d\}$ is linearly independent over $\mathbb{F}_p$, which means that it is linearly independent over $\mathbb{Q}$. □

We will next show that the set $\{B_1, \ldots, B_d\}$ is a minimum cycle basis with probability at least $3/4$.

Theorem 2. *When p is good, Randomized-MCB computes a minimum cycle basis.*

We will assume that the algorithm Deterministic-MCB breaks ties for the shortest cycle in the same way as Randomized-MCB breaks them. That is, both the algorithms use the same rule to determine the *shorter* cycle between two cycles of equal weight. Then we can show the following.

Lemma 5. *When p is good, $B_i = C_i$, for each $1 \le i \le d$.*

We will show this by induction. The vector $X_1 = S_1 = (1,0,\ldots,0)$. The inner product of any cycle with $(1,0,...,0)$ is ± 1 or 0. The inner product will be ± 1 if and only if the cycle contains the arc a_1. Also, looking at the inner product modulo p does not change a 1 or a -1 to 0. So B_1 is a shortest cycle that contains the arc a_1. C_1 is also a shortest cycle that contains the first arc a_1 and so by our assumption that both these algorithms break ties identically, we have that $B_1 = C_1$.

Let us now assume that $B_j = C_j$ for $j \le i-1$. N_i is a vector in $\mathbb{Z}^m$ of the form $(y_1,...,y_{i-1},k_i,0,...,0)$, where $(y_1,...,y_{i-1})$ is the unique solution to

$$\begin{pmatrix} \tilde{C}_1 \\ \vdots \\ \tilde{C}_{i-1} \end{pmatrix} x = \begin{pmatrix} -k_i c_{1i} \\ \vdots \\ -k_i c_{(i-1)i} \end{pmatrix} \tag{3}$$

where $\tilde{C}_j$ is the incidence vector of cycle C_j restricted to its first $i-1$ coordinates and $k_i = \det(\mathcal{M}_i)$ (recall that $\mathcal{M}_i$ is the $(i-1)\times(i-1)$ matrix above whose rows are $\tilde{C}$'s). Note that $(y_1,...,y_{i-1}) \bmod p$ is a solution to this set of equations in $\mathbb{F}_p$.

X_i is a vector in $\mathbb{F}_p^m$ of the form $(x_1,\ldots,x_{i-1},1,0,\ldots)$ and X_i is orthogonal to $B_1,...,B_{i-1}$ in $\mathbb{F}_p$. Since $B_j = C_j$ for $j \le i-1$, this means that X_i is orthogonal to $C_1,...,C_{i-1}$ in $\mathbb{F}_p$. So in $\mathbb{F}_p$, $(x_1,\ldots,x_{i-1})$ is a solution to

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

So $k_i(x_1,\ldots,x_{i-1}) \bmod p$ is a solution to Equation (3) in $\mathbb{F}_p$. We would like to prove that Equation (3) has a unique solution in $\mathbb{F}_p$.

Lemma 6. *If $\langle C_j, N_j\rangle \neq 0 (\bmod p)$ for $1 \le j \le i-1$, then $k_i \neq 0 (\bmod p)$.*

Let us assume the above Lemma and complete the argument. Then we will prove Lemma 6. Since $\det(\mathcal{M}_i) = k_i \neq 0 (\bmod p)$, Equation (3) should have a unique solution in $\mathbb{F}_p$. So $k_i(x_1,\ldots,x_{i-1}) \bmod p = (y_1,\ldots,y_{i-1}) \bmod p$. In other words, $k_i(x_1,\ldots,x_{i-1},1,0,\ldots,0) \bmod p = (y_1,\ldots,y_{i-1},k_i,0,\ldots,0) \bmod p$. That is,

$$k_i \cdot X_i \ (\bmod p) = N_i \ (\bmod p)$$

So N_i is just a scalar multiple of X_i when these vectors are viewed as elements of $\mathbb{F}_p^m$. Hence for any cycle D, $\langle D, N_i\rangle \neq 0 (\bmod p)$ if and only if $\langle D, X_i\rangle \neq 0 (\bmod p)$. Since p is a good prime, $\langle C_i, N_i\rangle \neq 0 (\bmod p)$. So $\langle C_i, X_i\rangle \neq 0 (\bmod p)$.

Let K be any cycle such that $\langle K, X_i\rangle \neq 0 (\bmod p)$. Then $\langle K, N_i\rangle \neq 0 (\bmod p)$, hence $\langle K, N_i\rangle \neq 0$. This is equivalent to $\langle K, S_i\rangle \neq 0$. So every cycle that satisfies $\langle K, X_i\rangle \neq 0 (\bmod p)$ is also a candidate cycle of Deterministic-MCB in its i-th iteration. Since C_i was the shortest among all these cycles, we get that C_i also has to be the shortest cycle for Randomized-MCB in its i-th iteration. That is, $B_i = C_i$. This proves the induction step.

Proof of Lemma 6. We know that $\langle C_j, S_l\rangle = 0$ for $j < l$, so when we multiply the $(i-1)\times m$ matrix whose rows are C's with the $m \times (i-1)$ matrix whose columns are N's we get:

$$\begin{pmatrix} C_1 \\ \vdots \\ C_{i-1} \end{pmatrix} \cdot \begin{pmatrix} N_1^T \ldots N_{i-1}^T \end{pmatrix} = \begin{pmatrix} \langle C_1, N_1\rangle & 0 & 0 & \ldots & 0 \\ * & \langle C_2, N_2\rangle & 0 & \ldots & 0 \\ * & * & \langle C_3, N_3\rangle & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & \ldots & \langle C_{i-1}, N_{i-1}\rangle \end{pmatrix}$$

Since each N_j has only 0's after its j-th coordinate, we can restrict the matrix of N's to its first $i-1$ rows and the matrix of C's to its first $i-1$ columns and we still have:

$$\begin{pmatrix} \tilde{C}_1 \\ \vdots \\ \tilde{C}_{i-1} \end{pmatrix} \cdot \left(\tilde{N}_1^T \dots \tilde{N}_{i-1}^T\right) = \begin{pmatrix} \langle C_1, N_1\rangle & 0 & 0 & \dots & 0 \\ * & \langle C_2, N_2\rangle & 0 & \dots & 0 \\ * & * & \langle C_3, N_3\rangle & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & \dots & \langle C_{i-1}, N_{i-1}\rangle \end{pmatrix}$$

where $\tilde{N}_j$ is the restriction of N_j to its first $i-1$ coordinates. Now all the matrices are square matrices. The determinant of the matrix of $\tilde{C}$'s is k_i and the determinant of the matrix of $\tilde{N}_j$'s is an integer. So k_i divides the determinant on the right hand side, which is $\langle C_1, N_1\rangle \cdots \langle C_{i-1}, N_{i-1}\rangle$. Since no $\langle C_j, N_j\rangle = 0 (\mathrm{mod}\, p)$ for $1 \le j \le i-1$, the prime p does not divide this product. So p cannot divide k_i. Hence $k_i \neq 0 (\mathrm{mod}\, p)$. □

This completes the proof of Lemma 5. This lemma immediately implies Theorem 2 since $\{C_1, \dots, C_d\}$ is a minimum cycle basis. So the cycle basis computed by Randomized-MCB is a minimum cycle basis with probability at least $3/4$ (from Lemma 3).

Note that 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 $\delta > 0$.

3 Running Time of Randomized-MCB

The value of $\pi(r)$, the number of primes less than r, is given by $r/6\log r \le \pi(r) \le 8r/\log r$ [1]. So the elements in P can be bounded by $100d^2 \log d$. Using sieving, we can compute the set of primes in the first $100d^2 \log d$ numbers in $O(d^2 \log^2 d)$ time. So the set P can be determined in $O(d^2 \log^2 d)$ time.

3.1 Computing B_i

Now we consider the problem of computing a shortest cycle in G whose inner product with X_i is nonzero modulo p. Let us first review the technique in [11] and then we describe our improved algorithm for this problem. Using the digraph $G = (V, A)$ and the vector X_i, an undirected graph $U_{i,p}$ can be constructed. The graph $U_{i,p}$ can be visualised as a graph with p levels. Call these levels as level 0,..., level $(p-1)$. 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 $U_{i,p}$ also consists of p copies of each arc $a \in A$. The edges corresponding to arc $a = (u, v)$ are (u_j, v_k) where $k = (j + X_i(a))$ modulo p for each $j = 0, 1, \dots, p-1$. Each edge (u_j, v_k) in $U_{i,p}$ inherits the weight of its corresponding arc (u, v) of G. For example, let $a = (u, v), X_i(a) = 3$ and $p = 5$. Then $U_{i,p}$ has 5 copies of a which are $(u_0, v_3), (u_1, v_4), (u_2, v_0), (u_3, v_1), (u_4, v_2)$.

So we have a well-defined map from the vertex set of $U_{i,p}$ to the vertex set of G and from the edge set of $U_{i,p}$ to the arc set of G. We can extend this map to paths of $U_{i,p}$. Any path in $U_{i,p}$ maps to a *chain*[1] in G by mapping the vertices and edges in $U_{i,p}$ to their images in G. We say that path $(e_0, \dots, e_r)$ in the graph $U_{i,p}$ has *repeated* edges if e_i and e_j for some $i \neq j$, map to the same arc of G.

The following properties of $U_{i,p}$ capture the essence of this graph.

- any (v_0, v_ℓ) path in $U_{i,p}$ maps to a closed chain in G.
- a (v_0, v_ℓ) path in $U_{i,p}$ with no repeated edges maps to a cycle in G.
- the inner product of such a cycle with X_i is ℓ (in $\mathbb{F}_p$).

[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 $a_k = (x_k, x_{k-1})$.

The following lemma from [11] is what we need. Its proof follows easily from the above properties.

Lemma 7. *Let $q = \min_v \min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path in the graph $U_{i,p}$. Then q corresponds to a shortest cycle in G whose inner product with X_i is nonzero modulo p.*

So B_i can be computed by running Dijkstra's algorithm from v_0 for each $v \in V$ and taking the minimum over v, of these shortest $(v_0, v_\ell), \ell \neq 0$ paths. Since $U_{i,p}$ has pn nodes and pm edges, Dijkstra's algorithm takes $O(pm + pn\log n)$ time for each v_0. Hence the total time taken to compute the cycle B_i is $O(n \cdot (pm + pn\log n))$.

Now we will show that we can modify Dijkstra's algorithm for this application so that we take $O(m\log n)$ time for each v_0 instead of $O(pm + pn\log n)$ time.

3.2 Improved implementation of computing a shortest $(v_0, v_\ell), \ell \neq 0$ path

We will not build the graph $U_{i,p}$ explicitly. Whenever we are at a vertex u_j, we know its neighborhood as follows. If there is an arc a between u and a vertex w in G, then in $U_{i,p}$, w_k is a neighbor of u_j where $k = (j + X_i(a)) \bmod p$ if $a = (u, w)$ (directed from u to w), and $k = (j - X_i(a)) \bmod p$ if $a = (w, u)$ (directed from w to u).

The key observation here is that to compute $\min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path, it is enough to look at those intermediate vertices which are the closest or second closest of their "type" to v_0. That is, if u_j is a vertex in $\min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path, then u_j is closest or second closest to v_0 among all of $\{u_0, u_1, ..., u_{p-1}\}$. So while running Dijkstra's algorithm to determine $\min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path, we only explore neighborhoods of such vertices in the priority queue. Then the maximum number of vertices that we process in the priority queue is $2\sum \text{degree}(u) = O(m)$. So Dijkstra's algorithm for this application can be modified to run for at most $O(m)$ iterations and not $O(pn)$ iterations and the number of edges that we look at is also $O(m)$ and not $O(pm)$.

More formally, we will have $\mathsf{dist}[u] = \infty$ for each vertex u in $U_{i,p}$ and the priority queue Q contains all the nodes of $U_{i,p}$, keyed by their dist values. Then we start computing single-source shortest paths for each vertex in level 0. Call one such vertex as v_0. This procedure runs as follows:

- set $\mathsf{dist}[v_0] = 0$.
- Maintain an array marked for the n vertices of G and initially $\mathsf{marked}[u] = 0$ for each $u \in V$.
- Repeat
  - Extract the vertex x with the smallest dist value from Q.
    (in case of ties, x is the vertex which was assigned this dist value earliest)

    If x is v_ℓ for some $\ell \neq 0$, then store $\mathsf{dist}[v_\ell]$ and the path computed to v_ℓ and quit the Repeat loop.
    Else let $x = u_k$.
    – if $\mathsf{marked}[u] < 2$, then increment $\mathsf{marked}[u]$ and for each neighbor w of x do

    $$\mathsf{dist}[w] = \min(\mathsf{dist}[w], \mathsf{dist}[x] + \text{weight}(x, w))$$

    and set predecessor of w to x if its dist value has changed.
    – else do nothing.
- For each vertex whose distance was made finite in our loop, set its dist back to ∞ and insert back to Q the deleted vertices. (so that we can now run this procedure for another vertex w_0 of level 0)

Remark. There is always a (v_0, v_ℓ) path for some $\ell \neq 0$ in the graph $U_{i,p}$ for each $v \in G$. This is because X_i on its last $n-1$ coordinates (which are the arcs of the spanning tree T) is 0 and $X_i \neq 0$. So each level of the $U_{i,p}$ graph is connected and there is at least one edge from level 0 to some nonzero level.

Running time of the above algorithm. We look at neighborhoods of only those vertices which are of the type u_j such that u_j is the first or second among all the vertices in $\{u_0, ..., u_{p-1}\}$ to be extracted from the priority queue. For such vertices we make the dist value of their neighbors to be finite. The total number of vertices whose distance is ever made finite in our loop is bounded by $\sum_{u \in G} \deg(u_j)$ for all u_j which are closest or second closest to v_0 among the "u" vertices. Since $\deg(u_j) = \deg(u)$, we get the bound of $2\sum_u \deg(u) = O(m)$.

Let us implement the priority queue as a binary heap so that each of the operations needed above can be implemented in $O(\log(pn)) = O(\log n)$ amount of time. In the Repeat loop we charge the cost of extracting a vertex x to x's predecessor in shortest-path(v_0, u). So for each vertex u_j which is closest or second closest among "u" vertices to v_0, we do $O(\deg(u) \cdot \log n)$ amount of work. For the other vertices we do no work. We take $O(pn)$ time to build the binary heap. But we do this just once in the entire algorithm, at the beginning of our first iteration. Thereafter, we simply reset to infinity the dist value of only those vertices which were made finite while running our procedure for the previous vertex. Also, we insert the deleted vertices back into the heap. This takes $O(m \log n)$ work.

In iteration i, once we compute $\min_{\ell \neq 0}$shortest(v_0, v_ℓ) path for all $v \in V$, we have determined B_i. This takes time $O(n \cdot m \log n)$ given the priority queue Q containing all the vertices with their dist values. So the total amount of time to compute all the cycles $B_1, \ldots, B_d$ given the vectors $X_1, \ldots, X_d$ is $O(pn + d(n \cdot m \log n))$ which is $O(m^2 n \log n)$. All we need to show now is the following lemma.

Lemma 8. *In order to compute* $\min_{\ell \neq 0}$ *shortest* (v_0, v_ℓ) *path in* $U_{i,p}$*, it is enough to look at vertices which are of the form: closest or second closest of their "type" to* v_0*.*

Proof. Let v_r be the closest vertex to v_0 among $\{v_1, \ldots, v_{p-1}\}$. Let q be the shortest path from v_0 to v_r. Suppose u_k is a vertex on q which is *not* of the description: closest or second closest among "u" vertices to v_0. By the construction of the graph $U_{i,p}$, we know that if there is a (u_k, v_r) path, then there is also a $(u_t, v_{(r+k-t) \bmod p})$ path which is just a translation of the (u_k, v_r) path, for every $t = 0, 1, \ldots, p-1$. (Note that all these paths correspond to the same chain in G with endpoints u and v, so they have the same length.) Let u_f be the closest "u" vertex to v_0. Add the $(u_f, v_{(r+k-f) \bmod p})$ path described above to the shortest-path(v_0, u_f). This gives us a path from v_0 to $v_{(r+k-f) \bmod p}$, which is shorter than q. So $v_{(r+k-f) \bmod p}$ is closer than v_r to v_0. But there is one catch: it could be the case that $(r+k-f) \bmod p = 0$.

Then consider the path which is obtained by adding the analogous $(u_s, v_{(r+k-s) \bmod p})$ path to the shortest-path(v_0, u_s), where u_s is the second closest "u" vertex to v_0. It cannot be the case that both $(r+k-f) \bmod p$ and $(r+k-s) \bmod p$ are 0 because that would mean that $f - s = 0 (\bmod p)$ but f and s are distinct numbers in $\{0, ..., p-1\}$. So we get the result that in order to compute $\min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path, it is enough to consider only the special vertices as intermediate vertices. □

The overall running time of Randomized-MCB. Under the assumption that arithmetic on $O(\log m)$ bits takes unit time, it follows that addition, subtraction and multiplication in $\mathbb{F}_p$ can be implemented in unit time since p is $O(d^2 \log d)$. However we also need to implement division efficiently since the update step of X_j involves division. Once p is chosen, we will compute the multiplicative inverses of all elements in $\mathbb{Z}_p^*$ by the extended Euclid's gcd algorithm by solving $ax = 1 (\bmod p)$ for each $a \in \mathbb{Z}_p^*$. This takes time $O(\log p)$ for each element and hence $O(p \log p)$ for all the elements. Thereafter, division in $\mathbb{F}_p$ gets implemented as multiplication with the inverse of the divisor.

We need to account for the time taken to update the vectors $X_{i+1}, ..., X_d$ in iteration i. Adding a scalar multiple of X_i to a vector X_j takes $\Theta(i)$ time. So the time taken by the update step in iteration i to update $d - i$ vectors is $\Theta(i(d-i))$. So the entire time taken by the update steps of all the iterations is $\Theta(d^3)$. So the total time taken by the algorithm Randomized-MCB is $O(m^3 + m^2 n \log n)$.

4 Faster implementation

Instead of spending $\Theta(m^3)$ time for the update step, using the technique in [12] we can implement the update step in $O(m^\omega)$ time, where $\omega < 2.376$ is the exponent of matrix multplication. This then gives us an $O(m^2 n \log n)$ randomized algorithm for computing a minimum cycle basis. The algorithm FAST-Randomized-MCB is described below.

- Compute a set P of d^2 primes $p_0, p_1, \ldots$ where each $p_j \geq d^2$. Choose a prime p uniformly at random from this set.
- Call the procedure *extend_cycle_basis*$(\{\}, \{e_1, \ldots, e_d\}, d)$, where $e_1, \ldots, e_d$ are the first d vectors of the standard basis.

The procedure *extend_cycle_basis* takes as input a partial cycle basis, say, $\{D_1, ..., D_i\}$ (denoted by $\mathcal{D}$), a parameter k, and k vectors $v_{i+1}, \ldots, v_{i+k}$ of $\mathbb{F}_p^m$ which are orthogonal to $\{D_1, ..., D_i\}$ over $\mathbb{F}_p$ and computes k new cycles $D_{i+1}, ..., D_{i+k}$ to extend the cycle basis. The role of $v_{i+1}, \ldots, v_{i+k}$ is identical to the role played by the vectors $X_{i+1}, X_{i+2}, ...$ at the beginning of iteration i in the algorithm Randomized-MCB (Section 2.2). Just as the vectors X_j got updated in Randomized-MCB, the vectors v_j get updated during the course of *extend_cycle_basis*. But the difference is that we will update *many* v_j's with respect to *many* cycles in one bulk update step and this bulk update can be implemented efficiently using fast matrix multiplication. For clarity we will sometimes use the notation v_j^ℓ to denote the version of v_j that is orthogonal to the cycles $D_1, \ldots, D_{\ell-1}$.

We describe below the recursive procedure *extend_cycle_basis*. Note that all the arithmetic that we do here is over the field $\mathbb{F}_p$.

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

- if $k = 1$, compute a shortest cycle D_{i+1} such that $\langle D_{i+1}, v_{i+1} \rangle \neq 0 (\bmod p)$.
- if $k > 1$, use recursion. Let $t = \lfloor k/2 \rfloor$.
  1. call *extend_cycle_basis*$(\mathcal{D}, \{v_{i+1} \ldots, v_{i+t}\}, t)$ to extend the current cycle basis by t elements. That is, the cycles $D_{i+1}, ..., D_{i+t}$ are computed in a recursive manner.
  2. call *update*$(\{v_{i+1}, \ldots, v_{i+t}\}, \{v_{i+t+1}, ..., v_{i+k}\})$. This updates $\{v_{i+t+1}^{i+1}, ..., v_{i+k}^{i+1}\}$ *en masse* into the desired versions $\{v_{i+t+1}^{i+t+1}, ..., v_{i+k}^{i+t+1}\}$ that are orthogonal to $D_1, \ldots, D_{i+t}$.
  3. call *extend_cycle_basis*$(\mathcal{D} \cup \{D_{i+1}, ..., D_{i+t}\}, \{v_{i+t+1}, ..., v_{i+k}\}, k-t)$ to extend the current cycle basis by $k-t$ cycles. That is, the cycles $D_{i+t+1}, ..., D_{i+k}$ will be computed recursively.

The key subroutine in *extend_cycle_basis* is *update*. We want to update the vectors $v_{i+t+1}, ..., v_{i+k}$ in *update*. Each of these vectors is now in the version v_j^{i+1} and we want to update it to its version v_j^{i+t+1}. The version v_j^{i+t+1} will be orthogonal to $D_1, \ldots, D_{i+t}$. Let us call v_j^{i+t+1} as w_j. We define w_j as follows:

$$w_j = v_j + a_{j1} v_{i+1} + a_{j2} v_{i+2} + \cdots + a_{jt} v_{i+t} \quad \text{for some } a_{j1}, ..., a_{jt} \in \mathbb{F}_p \tag{4}$$

Since w_j is a linear combination of vectors which are orthogonal to $D_1, \ldots, D_i$, w_j will always be orthogonal to $D_1, \ldots, D_i$. We will determine these coefficients $a_{j1}, ..., a_{jt}$ so that w_j is orthogonal to the cycles $D_{i+1}, \ldots, D_{i+t}$.

Writing Equation (4) for all $i+1 \leq j \leq i+k$ in matrix form, we have

$$\begin{pmatrix} w_{i+t+1} \\ \vdots \\ w_{i+k} \end{pmatrix} = (A\ I) \cdot \begin{pmatrix} v_{i+1} \\ \vdots \\ v_{i+k} \end{pmatrix}$$

where A is a $(k-t)\times t$ matrix whose ℓth row has the unknowns $a_{j1},...,a_{jt}$, where $j = i+t+\ell$. I is the $(k-t)\times(k-t)$ identity matrix. And w_j (similarly, v_j) represents a row with the coeffi cients of w_j (resp., v_j) as its row elements.

Let us multiply both sides of this equation with an $m\times t$ matrix whose columns are the cycles $D_{i+1},\ldots,D_{i+t}$. That is,

$$\begin{pmatrix} w_{i+t+1} \\ \vdots \\ w_{i+k} \end{pmatrix} \cdot \left(D_{i+1}^T \ldots D_{i+t}^T\right) = (A\ I) \cdot \begin{pmatrix} v_{i+1} \\ \vdots \\ v_{i+k} \end{pmatrix} \cdot \left(D_{i+1}^T \ldots D_{i+t}^T\right) \tag{5}$$

Then the left hand side is the 0 matrix since each of the vectors $w_{i+t+1},...,w_{i+k}$ has to be orthogonal to each of $D_{i+1},...,D_{i+t}$. Let us split the product of the matrix of v's and the matrix of D^T's on the right hand side as

$$B = \begin{pmatrix} v_{i+1} \\ \ldots \\ v_{i+t} \end{pmatrix} \cdot \left(D_{i+1}^T \ldots D_{i+t}^T\right);\ C = \begin{pmatrix} v_{i+t+1} \\ \ldots \\ v_{i+k} \end{pmatrix} \cdot \left(D_{i+1}^T \ldots D_{i+t}^T\right)$$

Then Equation (5) becomes

$$0 = (A\ I) \cdot \begin{pmatrix} B \\ C \end{pmatrix}$$

We get $AB+C=0$. We can determine $A=-CB^{-1}$ if B is invertible.

$$B = \begin{pmatrix} \langle v_{i+1}, D_{i+1}\rangle & \ldots & \langle v_{i+1}, D_{i+t}\rangle \\ \langle v_{i+2}, D_{i+1}\rangle & \ldots & \langle v_{i+2}, D_{i+t}\rangle \\ \vdots & \vdots & \vdots \\ \langle v_{i+t}, D_{i+1}\rangle & \ldots & \langle v_{i+t}, D_{i+t}\rangle \end{pmatrix} = \begin{pmatrix} * & * & * & \ldots & * \\ 0 & * & * & \ldots & * \\ 0 & 0 & * & \ldots & * \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & * \end{pmatrix}$$

B is an upper triangular matrix over $\mathbb{F}_p$ with nonzero elements on the diagonal, since for indices j in the range $i+1\le j\le i+t$, the vector v_j is in its fi nal version v_j^j using which D_j was computed. This means $\langle v_j^j, D_j\rangle \neq 0 (\bmod p)$ and $\langle v_j^j, D_\ell\rangle = 0 (\bmod p)$ for all $\ell < j$. Hence, B is invertible over $\mathbb{F}_p$. Thus $A=-CB^{-1}$.

4.1 Correctness

It follows from the initialization and the implementation of *update* that the vector v_i always has a 1 in its i-th coordinate and 0's in coordinates $i+1,...,m$. And whenever *extend_cycle_basis*$(\{D_1,...,D_{i-1}\}, v_i, 1)$ is called, v_i is a vector in $\mathbb{F}_p^m$ that is orthogonal to $\{D_1,...,D_{i-1}\}$ over $\mathbb{F}_p$. So if $D_j = B_j$ for $1\le j\le i-1$, where B_j is the cycle computed by our original algorithm Randomized-MCB in its j-th iteration, then the vector $v_i = X_i$. Then $D_i = B_i$, since both Randomized-MCB and FAST-Randomized-MCB will use the same subroutine to compute a shortest cycle whose inner product with X_i is nonzero modulo p. Hence, it follows from Section 2.2 that whenever p is good, FAST-Randomized-MCB computes a minimum cycle basis.

4.2 The running time of FAST-Randomized-MCB

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

$$T(k) = \begin{cases} \text{cost of computing a shortest cycle } D_i \text{ such that } \langle D_i, v_i\rangle \neq 0 (\bmod p) & \text{if } k=1 \\ 2T(k/2) + \text{cost of update} & \text{if } k>1 \end{cases}$$

The computation of matrices B and C takes time $O(mk^{\omega-1})$ using the fast matrix multiplication algorithm, where $\omega < 2.376$. To compute B (similarly, C) we are multiplying $\lfloor k/2 \rfloor \times m$ by $m \times \lfloor k/2 \rfloor$ matrices. We split the matrices into $2m/k$ square blocks and use fast matrix multiplication to multiply the blocks. Thus multiplication takes time $(2m/k)(k/2)^{\omega} = O(mk^{\omega-1})$. We can also invert B in $O(k^{\omega})$ time and we also multiply C and B^{-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 $v_{i+1}, ...v_{i+k}$ to get the updated vectors $w_{i+t+1}, ..., w_{i+k}$.

With a preprocessing cost of $O(m^2 n \log n)$, we can compute the set of primes P and initialise the binary heap for computing shortest paths. Thereafter, the computation of a cycle takes $O(mn \log n)$ time (from Section 3.2). So the recurrence turns into

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

This solves to $T(m) = O(m(mn \log n) + m^{\omega})$. We can assume that G is a simple graph,[2] so $m \leq n^2$. Then $m^{\omega} < m^2 n$ and the running time reduces to $T(m) = O(m^2 n \log n)$. Hence we can conclude the following theorem.

Theorem 3. *A minimum cycle basis of a directed graph can be computed with high probability in time* $O(m^2 n \log n)$.

Acknowledgments. I am grateful to Kurt Mehlhorn for useful discussions and his help in improving the presentation of the paper. I also wish to 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. A. C. Cassell and J. C. Henderson and K. Ramachandran Cycle bases of minimal measure for the structural analysis of skeletal structures by the flexibility method *Proc. Royal Society of London Series A*, 350: 61-70, 1976.
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. P. M. Gleiss *Short cycles: minimum cycle bases of graphs from chemistry and biochemistry*. PhD thesis, Universität Wien, 2001.
8. P. M. Gleiss, J. Leydold, and P. F. Stadler. Circuit bases of strongly connected digraphs. *Discussiones Math. Graph Th.*, 23: 241-260, 2003.
9. 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.
10. J. D. Horton. A polynomial-time algorithm to find a shortest cycle basis of a graph. *SIAM Journal of Computing*, 16:359–366, 1987.
11. T. Kavitha and K. Mehlhorn. A Polynomial Time Algorithm for Minimum Cycle Basis in Directed Graphs *In Proc. of STACS*, LNCS 3404: 654-665, 2005.
12. 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.
13. Christian Liebchen. Finding Short Integral Cycle Bases for Cyclic Timetabling. *In Proc. of ESA*, LNCS 2832: 715-726, 2003.
14. C. Liebchen and L. Peeters. On Cyclic Timetabling and Cycles in Graphs. Technical Report 761/2002, TU Berlin.
15. C. Liebchen and R. Rizzi. A Greedy Approach to compute a Minimum Cycle Basis of a Directed Graph. Technical Report 2004/31, TU Berlin.

[2] we can easily reduce the problem of computing a minimum cycle basis in a multigraph to that of computing one in a simple graph.

Appendix

Some Examples

As mentioned in Section 1, every cycle basis of a directed graph need not project onto an undirected cycle basis. The following example (Fig. 1) shows this.

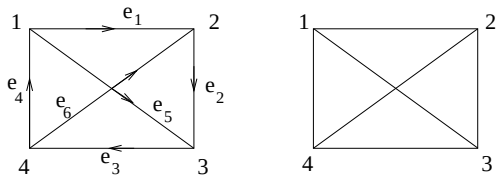

Fig. 1. Directed K_4 and the underlying undirected graph

The three 4-cycles in the above directed graph in Fig. 1: $C_1 = (e_1, e_2, e_3, e_4)$, $C_2 = (e_1, e_6, e_3, e_5)$ and $C_3 = (e_2, e_5, e_4, e_6)$ given by the vectors $(1,1,1,1,0,0), (1,0,-1,0,-1,-1)$, and $(0,1,0,-1,-1,1)$ are linearly independent over $\mathbb{Q}$. Hence they form a cycle basis for the directed K_4. But in the underlying undirected graph, each of these cycles is equal to the sum of the other two modulo 2, so C_1, C_2, C_3 do not form a cycle basis for undirected K_4.

Similarly, a *minimum* cycle basis of a digraph need not project onto a cycle basis of the underlying undirected graph. The following example was given in [14]: the smallest set of cycles that forms a minimum cycle basis for a directed graph but does not project onto a cycle basis for the underlying undirected simple graph is

$$(1,2,3)\ (1,2,4)\ (1,3,5)\ (1,4,6)\ (1,5,6)$$
$$(2,3,6)\ (2,4,5)\ (2,5,6)\ (3,4,5)\ (3,4,6)$$

on the directed K_6 (with edges oriented arbitrarily) and unit arc weights. On one hand, the set only contains triangles, hence its weight is minimum. And as there are 10×10 submatrices of the cycle matrix that have non-zero determinant, it is indeed a cycle basis of the graph K_6. On the other hand, as every edge is hit exactly twice, it cannot be a cycle basis for the undirected graph.