Algorithms to Compute Minimum Cycle Basis in Directed Graphs*

Telikepalli Kavitha[†] Kurt Mehlhorn[‡]

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 nonnegative weights assigned to them. 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. This paper presents an $\tilde{O}(m^4 n)$ algorithm, which is the first polynomial time algorithm for computing a minimum cycle basis in G. We then improve it to an $\tilde{O}(m^4)$ algorithm.

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 $\mathbb{GF}(2)$ generated by these vectors is the cycle space of the graph. There are directed graphs in which the minimum cycle basis has lower weight than any cycle basis of the underlying undirected graph. Hence algorithms for computing a minimum cycle basis in an undirected graph cannot be used as black boxes to solve the problem in directed graphs.

---

*A preliminary version of this paper appeared in STACS 2005 [14].

†*Corresponding Author.* CSA Department, Indian Institute of Science, Bangalore 560012, India. Email: `kavitha@csa.iisc.ernet.in`, tel: +91-80-22932386, fax: +91-80-23602911. This work was done while the author was at the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

‡Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany. Email: `mehlhorn@mpi-inf.mpg.de`, tel: +49-681-9325100, fax: +49-681-9325199. Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

1 Introduction

Let $G = (V,A)$ be a directed graph with m arcs and n vertices. A *cycle* C in G will consist 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 directed path in G.

That is, we will consider cycles in the underlying undirected graph of G and we assign each such cycle C a $\{-1,0,1\}$ incidence 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 spanned by the incidence vectors of cycles in G. 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 basis of the cycle space of G, or in other words, it is a set of cycles $C_1, \ldots, C_d$ 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 $w : A \longrightarrow \mathbb{R}^+$, i.e., the arcs of G have nonnegative 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, it 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.

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. Such algorithms include algorithms for diverse applications like chemical "ring perception"[6], structural flexibility analysis [12], cycle analysis of electrical networks [4]. In many cases the network graphs of interest are intrinsically directed. Let us first try to see the relation between cycle bases of directed and undirected graphs.

In an undirected graph $U = (N,E)$, with each cycle C 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 $\mathbb{GF}(2)$ generated by these vectors is called the *cycle space* of U. A minimum cycle basis of U is a set of linearly independent (over $\mathbb{GF}(2)$)

cycles that span the cycle space of U and whose sum of lengths 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 $C_1,\ldots,C_d$ 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 [17] that if $\mathcal{C} = \{C_1,\ldots,C_d\}$ 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 converse is not true. Similarly, a minimum cycle basis of a digraph need not project onto a cycle basis of the underlying undirected graph. Further, there are directed graphs no minimum cycle basis of which projects onto a cycle basis of the underlying undirected graph. Section 2 contains such examples. The books by Deo [5] and Bollobás [3] have an in-depth coverage of the subject of cycle bases. The problem of computing a minimum cycle basis in undirected graphs has been well-studied [2, 4, 7, 10, 11, 15] 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 [15].

In this paper we first give an $\tilde{O}(m^4n)$ algorithm and then an $\tilde{O}(m^4)$ algorithm to compute a minimum cycle basis in a digraph $G = (V,A)$ where $|A| = m$ and $|V| = n$. The first algorithm was already a part of the conference version of this paper [14], the improved algorithm is new. In [14] we also gave an $\tilde{O}(m^3n)$ Monte-Carlo algorithm; this was since then improved to $\tilde{O}(m^2n)$ by the first author [13]. Because of this improvement, we do not include the Monte-Carlo algorithm in this paper.

After our paper [14] appeared in preprint form, Liebchen and Rizzi [18] gave an $\tilde{O}(m^{\omega+1}n)$ algorithm to compute a minimum cycle basis in a directed graph, where $\omega < 2.376$ is the exponent of matrix multiplication. Their algorithm is better than our original algorithm for all graphs and is better than our improved algorithm for dense graphs. For sparse graphs, that is when $m = O(n^{1/(3-\omega)}) \approx O(n^{1.6})$, our improved algorithm is better. Networks arising in practice are typically sparse. Also, our algorithm can be extended to obtain an $\tilde{O}(m^3n)$ algorithm to verify if a given set of cycles is a minimum cycle basis. It may be noted that an $O(m^3n + m^2n^2 \log n)$ algorithm is now known for the minimum cycle basis problem in directed graphs [9]. This improved algorithm is based on the approaches used in the algorithms here.

The approach used in [18] for the $\tilde{O}(m^{\omega+1}n)$ algorithm is based on adapting Horton's algorithm for computing a minimum cycle basis in an undirected graph [11]. This works by computing a family of at most mn cycles based on shortest paths and determining the least weight $m-n+1$ linearly independent cycles in this family. Our approach is complementary to theirs. Our algorithms are iterative and in each iteration we compute a new cycle of the minimum cycle basis. Our algorithms use simple linear algebra and elementary number theory and are in the domain of arithmetical algorithms.

Cycle bases of directed graphs have been studied earlier in [16, 17, 8]. The focus in these papers is on algorithms for special classes of cycle bases, like integral cycle bases [16, 17] - 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.

In Section 2 we present some examples that illustrate the differences between cycle bases of directed graphs vs. their underlying undirected graphs. In Section 3 we present the $\tilde{O}(m^4 n)$ algorithm. An important subroutine in our algorithm, which computes a shortest cycle whose inner product with a given vector is nonzero, is presented in Section 4. Section 5 has the $\tilde{O}(m^4)$ algorithm. In Section 6 we present the $\tilde{O}(m^3 n)$ verification algorithm.

2 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 (Figure 1) shows this. The

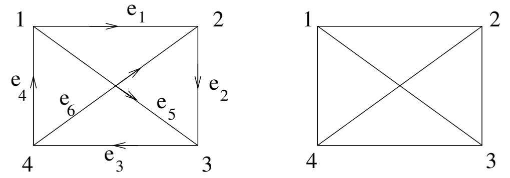

Figure 1: Directed K_4 and the underlying undirected graph.

three 4-cycles $C_1 = (e_1, e_2, e_3, e_4)$, $C_2 = (e_1, e_6, e_3, e_5)$ and $C_3 = (e_2, e_5, e_4, e_6)$ correspond to the vectors $(1,1,1,1,0,0)$, $(1,0,-1,0,-1,-1)$, and $(0,1,0,-1,-1,1)$ and 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. We present below an example, given in [18], of a directed graph whose unique minimum cycle basis does not project onto a cycle basis of the underlying undirected graph. Thus this is a directed graph in which the minimum cycle basis has lower weight than any undirected cycle basis.

Consider the generalized Petersen graph $P_{7,2}$ in Figure 2. Call an edge (u,v) an *inner edge* if $\{u,v\} \subset \{0,1,\ldots,6\}$. Similarly call an edge (u,v) an *outer edge* if $\{u,v\} \subset \{a,\ldots,g\}$. The seven edges that remain are called *spokes*. Assign a weight function w by assigning weight two to the seven inner edges and weight three to the outer edges and spokes. It can be shown that $(P_{7,2}, w)$ has girth 14 and there are precisely eight cycles having weight 14.

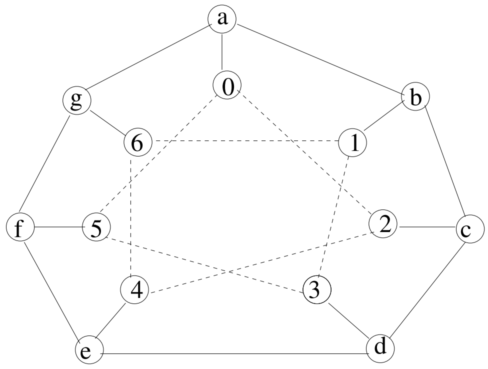

Figure 2: The generalized Petersen graph $P_{7,2}$.

Every edge of $P_{7,2}$ belongs to precisely two of the eight cycles with weight 14. Therefore in the undirected case, these $8 = m - n + 1$ cycles are not independent over $\mathbb{GF}(2)$. Thus in the undirected case, every cycle basis of $(P_{7,2}, w)$ has weight at least 113. In the directed case, these 8 cycles under any orientation of edges can be shown to be linearly independent. So there is a directed cycle basis of weight 112.

3 The Algorithm

Our algorithm for computing a minimum cycle basis in a directed graph is broadly based on the approach used by De Pina [4] for computing a minimum cycle basis in an undirected graph. This approach was also used as the starting point in [2] and [15] for the minimum cycle basis problem in undirected graphs. 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 also assume that there are no multiple edges in G. It is easy to see that whenever there are two arcs from u to v, the heavier arc (call it a) can be deleted from A and the least weight cycle $C(a)$ containing a can be added to the minimum cycle basis computed on $(V, A \setminus \{a\})$. The cycle $C(a)$ consists of the arc a and the shortest path between u and v in the underlying undirected graph. All such cycles can be computed by an all-pairs-shortest-paths computation, which takes $\tilde{O}(mn)$ time, in the underlying undirected graph of G. Hence we can assume that $m \leq n^2$.

3.1 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 usual inner product of $S = (s_1, \ldots, s_m)$ and $C = (c_1, \ldots, c_m)$, which are vectors in the space $\mathbb{Q}^m$. A high-level description of our algorithm is as follows.

For $i = 1, \ldots, d$ 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 $j < i$.

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 any 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: [4] It is easy to see that C_i is linearly independent of $\{C_1, \ldots, C_{i-1}\}$. S_i is a witness of this linear independence since $\langle S_i, C_j \rangle = 0$ for each $j < i$, 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 $\{C_1, \ldots, C_i\} \nsubseteq \mathcal{C}$ for any minimum cycle basis $\mathcal{C}$. So

$\{C_1,\ldots,C_{i-1}\} \subseteq \mathcal{B}$ for some minimum cycle basis $\mathcal{B}$. Then $C_i = \lambda_1 B_1 + \lambda_2 B_2 + \cdots + \lambda_l B_l$, where $\lambda_1, \lambda_2, \ldots$ are are non-zero rationals and $\{B_1,\ldots,B_l\} \subseteq \mathcal{B}$. Since $\langle S_i, C_i\rangle \neq 0$, there exists a $B_k \in \{B_1,\ldots,B_l\}$ 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 $\mathcal{B}' = \mathcal{B} \cup \{C_i\} \setminus \{B_k\}$ is a set of cycles whose weight is at most the weight of $\mathcal{B}$. We will show that $\mathcal{B}'$ forms a basis, which implies that $\mathcal{B}'$ is a minimum cycle basis. Recall that $C_i = \lambda_1 B_1 + \cdots + \lambda_k B_k + \cdots + \lambda_l B_l$, where $\lambda_k \neq 0$. So B_k can be expressed as a rational linear combination of the cycles in $\mathcal{B} \cup \{C_i\} \setminus \{B_k\}$. Since $\mathcal{B}$ forms a basis, it follows that $\mathcal{B} \cup \{C_i\} \setminus \{B_k\}$ forms a basis.

Now we wish to show that $\{C_1,\ldots,C_i\} \subseteq \mathcal{B}'$. The cycle B_k that has been omitted from $\mathcal{B}'$ cannot be one of $C_1,\ldots,C_{i-1}$ since the inner product of each of $C_1,\ldots,C_{i-1}$ with S_i is zero whereas $\langle S_i, B_k\rangle \neq 0$. Hence, $\{C_1,\ldots,C_i\} \subseteq \mathcal{B}'$, 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.

3.2 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:

$$\langle \vec{x}, C_j\rangle = 0, \text{ for all } j < i.$$

We will show that we can find an S_i with $\|S_i\|_\infty \leq i^{i/2}$.

Step 2. Compute $f(i)+1$ distinct primes $p_0,\ldots,p_{f(i)}$, where $f(i) = \lceil i/2 \rceil$ and each $p_t \geq m$.

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

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

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

So the only way $\langle S_i, D_i \rangle$ can be a multiple of $\prod_t p_t$ 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. In Section 3.3 we give the details of how to compute such an S_i, which should also satisfy the L_∞ norm bound given in Step 1. We describe how to compute a shortest cycle B_t such that $\langle B_t, S_i \rangle \neq 0 \pmod{p_t}$ in Section 4.

3.3 Computing S_i

Now we want to compute a non-zero vector $S_i \in \mathbb{Z}^m$ that satisfies the set of equations $\langle \vec{x}, C_j \rangle = 0$ for all $j < i$ and $\|S_i\|_\infty \leq i^{i/2}$.

Let us first assume that the arcs in the arc set A have been ordered so that $a_{d+1}, \ldots, a_m$ 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. 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 arc a_i and the spanning tree path connecting its endpoints 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 $\langle \vec{x}, C_j\rangle = 0$, for all $j < i$. We do this as follows.

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

Let the j-th cycle C_j have the incidence vector $(c_{j1},\ldots,c_{jm})$. Since the vector $(r_1,\ldots,r_{i-1},1,0,\ldots 0)$ 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 $\tilde{C}_j = (c_{j1},\ldots,c_{j(i-1)})$ be the restriction of C_j to its first $i-1$ coordinates. So $(r_1,\ldots,r_{i-1})$ is a solution to the set of equations:

$$\tilde{C_j}^T \cdot (x_1,\ldots,x_{i-1}) = -c_{ji} \text{ for } j = 1,\ldots,i-1.$$

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

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

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 since $k \leq i-1$ and only the first k entries of S_k may be non-zero. 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 $j < k$. 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:

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

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

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 $S_i = kS'_i = (y_1,\ldots,y_{i-1},k,0,\ldots,0)$ is an integral vector that is orthogonal to all the cycles $C_1,\ldots,C_{i-1}$. 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, $(y_1,\ldots,y_{i-1})$ is the (integral) solution to the set of equations:

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

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

$$|k| \le \prod_{t=1}^{i-1} \|\tilde{C}_j\| \le \prod_{t=1}^{i-1} \sqrt{i} \le i^{i/2}.$$

Similarly, each $|y_j| \le i^{i/2}$. Hence $\max\{y_1,\ldots,y_j,\ldots,k\} \le i^{i/2}$. Thus we have shown that $\|S_i\|_\infty \le i^{i/2}$.

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 $\omega < 2.376$ is the 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 $\tilde{O}(\ell)$ bits takes $\tilde{O}(\ell)$ time, we have the following lemma.

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

4 Computing B_t

The cycle B_t is a shortest cycle such that $\langle B_t, S_i\rangle \ne 0$ modulo p_t. In order to compute such a cycle, we build an undirected graph $U_{i,t}$ using the given directed graph $G = (V,A)$, the vector S_i and the number p_t.

The graph $U_{i,t}$ can be visualized as a layered graph with p_t layers.

- Each layer has a copy of every vertex $v \in V$. Let $v_0, v_1, \ldots, v_{p_t-1}$ be the p_t copies of vertex v.
- Each arc $a = (u,v) \in A$ also has p_t copies in $U_{i,t}$. Copy j of arc $a = (u,v)$ of G goes from u_j to v_k where $k = (j + S_i(a)) \bmod p_t$. As j varies from $0, 1, \ldots, p_t - 1$, we have p_t such edges in $U_{i,t}$ corresponding to the arc a in G.

  For example, let $a = (u,v), S_i(a) = 8$ and $p_t = 5$. Then $U_{i,t}$ has 5 copies of a which are $(u_0,v_3),(u_1,v_4),(u_2,v_0),(u_3,v_1),(u_4,v_2)$. 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 it to a *chain*[1] in G by mapping the vertices and edges in $U_{i,t}$ to their images in G.

Let $S_i \bmod p_t$ denote the vector where each coordinate in S_i is replaced by its remainder modulo p_t. That is, it is the vector $(S_i(a_1) \bmod p_t, \ldots, S_i(a_m) \bmod p_t)$. Then within each layer of $U_{i,t}$, we have edges corresponding to arcs of G whose coordinates in $S_i \bmod p_t$ take the value 0. Observe that each layer of $U_{i,t}$ is connected because the spanning tree T (recall that the arcs $a_{d+1}, \ldots, a_m$ form the edges of T) is a subgraph of each layer since $S_i(a_j) = 0$, for $d+1 \leq j \leq m$. Between the layers we have edges corresponding to arcs that have non-zero coordinates in $S_i \bmod p_t$. Only these edges allow us to go from one layer to another.

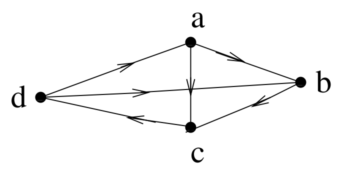

Figure 3: An example.

Consider the digraph G in Figure 3 with arcs (a,b), (a,c), (b,c), (d,a), (d,b), and (c,d). Let $S_i = (-2,3,2,0,0,0)$, corresponding to the arcs in the same order as given above. Let $p_t = 3$. Then $S_i \bmod 3$ is $(1,0,2,0,0,0)$ and the graph $U_{i,t}$ looks as shown in Figure 4. The path d_0-b_0-a_2-d_2 in Figure 4 maps to the chain

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

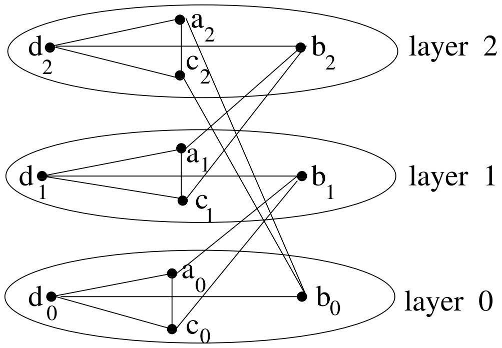

Figure 4: Since the arcs $(a,c),(d,a),(d,b),(c,d)$ have coordinate value 0 in $S_i \bmod 3$, in $U_{i,t}$ we have the edges (a_k,c_k), $(d_k,a_k),(d_k,b_k),(c_k,d_k)$ in each layer k for $k = 0,1,2$. Since $S_i((a,b)) \bmod 3$ is 1, we have the edges $(a_0,b_1),(a_1,b_2),(a_2,b_0)$. Similarly we have the edges $\{(b_0,c_2),(b_1,c_0),(b_2,c_1)\}$ since $S_i((b,c)) \bmod 3$ is 2.

$(d,(d,b),b,(a,b),a,(d,a),d)$ in the directed graph of Figure 3. We say that a path $p = (e_0,\ldots,e_r)$ in the graph $U_{i,t}$ has *repeated* edges if e_i and e_j for some $i \neq j$, map to the same arc of G. For instance, (a_0,b_1) and (a_1,b_2) map to the same arc (a,b) of G. So the path a_0-b_1-d_1-a_1-b_2 (Figure 4) has repeated edges. A path in $U_{i,t}$ with no repeated edges maps to a simple chain in G, that is, a chain that contains distinct arcs.

The following lemma captures the essence of the graph $U_{i,t}$.

Lemma 3 *Any (u_r,u_s) path in $U_{i,t}$ with no repeated edges maps to a cycle C in G. The incidence vector of such a cycle C satisfies*

$$\langle C,S_i\rangle = \pm(s-r) \mod p_t.$$

Proof: Any (u_r,u_s) path in $U_{i,t}$ with no repeated edges maps to a *simple* chain in G, that is, a chain that contains all distinct arcs. Since u_r and u_s map to the same vertex u in G, this chain is simple and closed. So this (u_r,u_s) path (call it ρ) corresponds to a cycle C in G.

In $U_{i,t}$, traversing an edge (v_ℓ,w_k) takes us from layer ℓ to layer k. So the change in layer numbers is $k-\ell$ which is $\pm S_i(a) \bmod p_t$, where a is the original arc in G corresponding to (v_ℓ,w_k) in $U_{i,t}$. This arc a is either (v,w) or (w,v). In the first case, that is, when $a = (v,w)$, the change in layer numbers is $S_i(a) \bmod p_t$. And in the second case, that is, when $a = (w,v)$, the change in layer numbers is

$-S_i(a) \bmod p_t$. So the change in layer numbers is exactly $(S_i(a) \bmod p_t) \cdot (\pm 1)$, where the ± 1 depends on the direction of traversal of the arc a by the edge (v_ℓ, w_k).

The total change in layer numbers caused by taking the path ρ (which we know is $s-r$) is exactly the sum of changes in layer numbers caused by traversing each edge. Let $c_1, \ldots, c_k$ be all the arcs in the cycle C. So we have

$$s - r = \sum_{j=1}^{k} (\text{the direction of traversal of } c_j \text{ by } \rho) \cdot S_i(c_j) \pmod{p_t}.$$

The inner product $\langle C, S_i \rangle$ is $\sum_{j=1}^{k}$ (the direction of traversal of c_j by C) $\cdot S_i(c_j)$. The direction that we had chosen for traversing C either corresponds to traversing ρ from u_r to u_s or from u_s to u_r. In the first case, for each c_j, the direction of traversal of c_j by $C =$ the direction of traversal of c_j by ρ. So in this case $s - r = \langle C, S_i \rangle \bmod p_t$.

In the second case, for each arc c_j, (the direction of traversal of c_j by C)$=$ $-$(the direction of traversal of c_j by ρ). Then $r - s = \langle C, S_i \rangle \bmod p_t$. Thus we have $\langle C, S_i \rangle = \pm(s - r) \pmod{p_t}$. ∎

Lemma 4 *Let $q = \min_v \min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path in the graph $U_{i,t}$.*

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

Then q corresponds to a shortest cycle B_t in G such that $\langle B_t, S_i \rangle \neq 0 \pmod{p_t}$.

Proof: The path q is a shortest (v_0, v_ℓ) path over $\ell = 1, \ldots, p_t - 1$ and all $v \in V$, in $U_{i,t}$. We will first have to show that q has no repeated edges. It is to help us here that we had this extra condition that in case of a tie, q is a shortest such path that has the least number of edges. If q has repeated edges, say, $e_1 = (u_x, s_{x'})$ and $e_2 = (u_y, s_{y'})$ then there is a subpath q' of q from u_x to u_y (or possibly, from $s_{x'}$ to $s_{y'}$) which avoids traversing either the edge $(u_x, s_{x'})$ or the edge $(u_y, s_{y'})$. Since all edges have nonnegative lengths, the length of q' is at most that of q and it has fewer edges. By replacing each vertex w_r in q' with the vertex $w_{r'}$ where $r' = r - x \pmod{p_t}$, we get a path from u_0 to u_h, where $h = y - x \bmod p_t$, whose length is identical to the length of q'. This contradicts the definition of q as the minimum over all v of shortest (v_0, v_ℓ) paths, for all $\ell \neq 0$, that has the least number of edges.

Hence, q has no repeated edges and so it corresponds to a cycle C in G. Since q is a (v_0, v_ℓ) path for some $\ell \neq 0$, by Lemma 3, C satisfies $\langle C, S_i \rangle \neq 0 \pmod{p_t}$.

It is easy to see that C is a shortest such cycle. If there is a shorter cycle C' in G with $\langle C', S_i \rangle = r \neq 0$, then there would be a shorter path than q in $U_{i,t}$ from (w_0, w_r), where w is any vertex of C'. ∎

Remark. Whenever S_i mod 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 $\ell \neq 0$, since each layer of $U_{i,t}$ is connected. If S_i mod 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 mod p_t is the zero vector.

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 layer 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 $U_{i,t}$ takes $O(p_t m + p_t n \log p_t n)$ 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 $O(m \log m)$. Hence we have shown the following lemma.

Lemma 5 *We can compute a shortest cycle B_t such that $\langle B_t, S_i \rangle \neq 0$ modulo p_t in $\tilde{O}(m^2 n)$ time.*

4.1 The entire algorithm

A summary of Algorithm-MCB, i.e., our algorithm to compute a minimum cycle basis in $G = (V, A)$, is given in Figure 5.

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.

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 $\tilde{O}(m^2 n)$ time (by Lemma 5), the time required to compute C_i is $\tilde{O}(m^3 n)$. That is, the i-th iteration takes $\tilde{O}(m^3 n)$ time since $m^{\omega+1} < m^3 n$ (because $m \leq n^2$). Thus Theorem 2 immediately follows.

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

– 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 $p_0, \ldots, p_{f(d)}$, where each prime $\geq m$. {This can be done by a sieving algorithm.}

for $i = 1, \ldots, d$ **do**
  Compute vector $S_i = (s_{i1}, \ldots, s_{ii}, 0, \ldots, 0) \in \mathbb{Z}^m$ such that $s_{ii} \neq 0$, $\|S_i\|_\infty \leq i^{i/2}$ 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.
    – 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 $\{C_1, \ldots, C_d\}$.

Figure 5: Algorithm-MCB: An algorithm to compute a minimum cycle basis in a digraph.

5 An $\tilde{O}(m^4)$ algorithm

In this section we present a faster implementation of our basic idea given in Section 3.1. First we consider the problem of computing the vectors $S_1, \ldots, S_d$ more efficiently. Then we will use an improved subroutine given in [13] to compute C_i from S_i.

5.1 The improved algorithm

Let us recall the basic idea presented in Section 3.1. The basic idea was:
For $i = 1, \ldots, d$ 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 $j < i$.

2. compute C_i to be a shortest cycle in G such that $\langle S_i, C_i \rangle \neq 0$.

Recall that we computed S_i in the i-th iteration as a solution to a linear system given by Equation (2). The role played by the vectors $S_1, \ldots, S_d$ above will now be played by vectors $T_1, \ldots, T_d$. But now we will compute the vectors T_i using *updates* and *normalizations*. However, in spite of this difference in the manner that they are computed, we will show that finally $T_i = S_i$ for each i. We present our algorithm below.

As in Algorithm-MCB (Figure 5) we first reorder the arcs so that $a_{d+1}, \ldots, a_n$ are the arcs of a spanning tree of the underlying undirected graph and we initialize the vectors $T_1, \ldots, T_d$ of $\mathbb{Q}^m$ to the first d vectors $e_1, \ldots, e_d$ of the standard basis of $\mathbb{Q}^m$, where the vector e_i has 1 in the i-th position and 0's elsewhere. Then the algorithm runs as follows.

1. For $i = 1$ to d do
   - compute C_i to be a shortest cycle such that $\langle C_i, T_i \rangle \neq 0$.
   - for $j = i+1$ to d do
     - update T_j as: $T_j = \langle C_i, T_i \rangle T_j - \langle C_i, T_j \rangle T_i$
     - normalize T_j as:

       $$T_j = T_j / y_j \text{ where } y_j = (j\text{-th coordinate of } T_j)/\langle C_i, T_i \rangle.$$

     (note that y_j is always defined since $\langle C_i, T_i \rangle \neq 0$)
2. return $\{C_1, \ldots, C_d\}$.

In the above algorithm we start with $T_1, \ldots, T_d$ initialized to the first d vectors of the standard basis. And each T_i is updated in iterations $1, 2, \ldots$ till the $(i-1)$-th iteration and T_i is finally used in the i-th iteration to compute the cycle C_i. We will show that at this stage, T_i is orthogonal to $C_1, \ldots, C_{i-1}$. The update step ensures this property.

Lemma 6 *At the end of the $(i-1)$-th iteration, the vectors $T_i, \ldots, T_d$ are orthogonal to the cycles $C_1, \ldots, C_{i-1}$. Moreover, in T_j, where $i \leq j \leq d$, the j-th coordinate is non-zero and besides the j-th coordinate, only the first $i-1$ coordinates may be non-zero.*

Proof: We will prove this by induction on i. At the end of the first iteration $T_j = \langle C_1, T_1 \rangle T_j - \langle C_1, T_j \rangle T_1$ for $j = 2, \ldots, d$. The inner product

$$\begin{aligned} \langle C_1, T_j \rangle &= \langle C_1, T_1 \rangle \langle C_1, T_j \rangle - \langle C_1, T_j \rangle \langle C_1, T_1 \rangle \\ &= 0. \end{aligned}$$

The update step takes a scalar multiple of T_j and adds a scalar multiple of T_1 to it. Since $\langle C_1, T_1\rangle \neq 0$ and $T_1 = (1,0,\ldots,0)$, the update step makes T_j look like $(x_{j1}, 0, \ldots, \langle C_1, T_1\rangle, 0, \ldots)$ for some x_{j1}. The normalization step just multiplies T_j with a non-zero scalar. So at the end of the first iteration each T_j is orthogonal to C_1 and has the above structure. This proves our base case.

Suppose the above invariant is maintained till the end of iteration k. So at the beginning of the $(k+1)$-th iteration, $T_{k+1},\ldots,T_d$ are orthogonal to $C_1,\ldots,C_k$. After the update step of the $(k+1)$-th iteration, it is easy to see that $T_{k+2},\ldots,T_d$ are orthogonal to C_{k+1}. And each of the new $T_{k+2},\ldots,T_d$ is a linear combination of the corresponding old $T_{k+2},\ldots,T_d$ and T_{k+1} - but all these vectors are already orthogonal to $C_1,\ldots,C_k$. Hence their linear combinations are also orthogonal to $C_1,\ldots,C_k$.

Also, by the induction hypothesis, at the end of the k-th iteration, each T_j is of the form $(t_1,\ldots,t_k,0,\ldots,t_j,0,\ldots)$ where $t_j \neq 0$. In the $(k+1)$-th iteration we add a scalar multiple of T_{k+1} to a non-zero scalar multiple of T_j, where $j > k+1$. By induction hypothesis, T_{k+1} has non-zero coordinates in only its first $k+1$ coordinates. So adding a scalar multiple of such a vector to $\langle C_{k+1}, T_{k+1}\rangle T_j$ will make the new T_j look like $(t'_1,\ldots,t'_{k+1},0,\ldots,t'_j,0,\ldots)$, where $t'_j \neq 0$. The normalization step only scales values by a non-zero scalar. Thus we maintain our invariant at the end of the $(k+1)$-th iteration. ∎

Let us try to understand better the structure of the vector T_i when it is used to compute C_i. We know that T_i is of the form $(t_{i1},\ldots,t_{ii},0,\ldots,0)$ with $t_{ii} \neq 0$. Write T_i as

$$T_i = t_{ii}(u_1,\ldots,u_{i-1},1,\ldots,0).$$

Since T_i is orthogonal to $C_1,\ldots,C_{i-1}$ and $t_{ii} \neq 0$, $(u_1,\ldots,u_{i-1})$ is a solution to the set of equations:

$$\begin{pmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_{i-1}^T \end{pmatrix} \vec{x} = \begin{pmatrix} -c_{1i} \\ \vdots \\ -c_{(i-1)i} \end{pmatrix} \tag{4}$$

But it follows from the analysis in Section 3.3 that this set of equations has a unique solution. Hence $(u_1,\ldots,u_{i-1})$ is the same vector as $S'_i = (r_1,\ldots,r_{i-1})$ that we computed in Section 3.3. So this implies that $T_i = t_{ii}S'_i$ is some rational multiple of $S_i = \det(\mathcal{M}_i)S'_i$, where $\mathcal{M}_i$ is the $(i-1)\times(i-1)$ matrix of $\tilde{C}$'s on the left of Equation (4). We actually want T_i to be the same as the vector S_i. The normalization step ensures this. In the $(i-1)$-th iteration we normalize T_i as:

$$\begin{aligned} T_i &= \frac{\langle C_{i-1}, T_{i-1}\rangle}{t_{ii}} \cdot t_{ii}(r_1, \ldots, r_{i-1}, 1, \ldots, 0, \ldots, 0) \\ &= \langle C_{i-1}, T_{i-1}\rangle (r_1, \ldots, r_{i-1}, 1, \ldots, 0, \ldots, 0) \end{aligned}$$

We will show that $\langle C_{i-1}, T_{i-1}\rangle = \det(\mathcal{M}_i)$. Then this proves that after the normalization step, $T_i = S_i$.

Lemma 7 $\det(\mathcal{M}_i) = \langle C_{i-1}, T_{i-1}\rangle$.

Proof: For $k \le i-1$, define $\tilde{T}_k$ to be the restriction of the vector T_k to its first $i-1$ coordinates. Similarly let $\tilde{C}_k$ be the restriction of C_k to its first $i-1$ coordinates. Then we have $\langle \tilde{C}_k, \tilde{T}_k\rangle = \langle C_k, T_k\rangle$ for all $k < i$ since $\tilde{T}_k$ has all the non-zero values of T_k when $k < i$. Putting all these relations in the form of a matrix, we have

$$\begin{pmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_{i-1}^T \end{pmatrix} \cdot \left(\tilde{T}_1 \ldots \tilde{T}_{i-1}\right) = \begin{pmatrix} \langle C_1, T_1\rangle & 0 & 0 & \ldots & 0 \\ * & \langle C_2, T_2\rangle & 0 & \ldots & 0 \\ * & * & \langle C_3, T_3\rangle & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ * & * & * & \ldots & \langle C_{i-1}, T_{i-1}\rangle \end{pmatrix}$$

The matrices above are square matrices - the determinant of the matrix of $\tilde{C}$'s is $\det(\mathcal{M}_i)$. The matrix of $\tilde{T}$'s is an upper triangular matrix and so its determinant is the product of its diagonal entries which is $t_{11}t_{22}\ldots t_{(i-1)(i-1)}$. So we have the relation:

$$\det(\mathcal{M}_i) t_{11} \ldots t_{(i-1)(i-1)} = \langle C_1, T_1\rangle \cdots \langle C_{i-1}, T_{i-1}\rangle. \tag{5}$$

But each $t_{kk} = \langle C_{k-1}, T_{k-1}\rangle$ since the vector T_k got updated for the final time in the $(k-1)$-th iteration and the normalization step of the $(k-1)$-th iteration set t_{kk} to the value $\langle C_{k-1}, T_{k-1}\rangle$. Hence $\prod_{k=2}^{i-1} t_{kk} = \langle C_1, T_1\rangle \cdots \langle C_{i-2}, T_{i-2}\rangle$. And $t_{11} = 1$ since $T_1 = (1, 0, \ldots, 0)$. Hence we have:

$$\det(\mathcal{M}_i)\langle C_1, T_1\rangle \cdots \langle C_{i-2}, T_{i-2}\rangle = \langle C_1, T_1\rangle \cdots \langle C_{i-1}, T_{i-1}\rangle.$$

Since none of the $\langle C_k, T_k\rangle$ values is zero, it follows from the above equation that $\det(M_i) = \langle C_{i-1}, T_{i-1}\rangle$. ∎

From Lemma 7 and the discussion preceding it, we can conclude the following lemma.

Lemma 8 *At the beginning of the i-th iteration, the vector* $T_i = S_i = \det(\mathcal{M}_i) \cdot (r_1, \ldots, r_{i-1}, 1, 0, \ldots, 0)$.

So though we construct the vector T_i differently from the vector S_i in Section 3.3, at the beginning of the i-th iteration, when we compute C_i from T_i, the vector T_i is exactly the same as the vector S_i.

5.2 Cost of update and normalization

In each iteration we do at most d update and d normalization steps. The cost of updating (similarly, normalizing) the vector T_j (for $i+1 \leq j \leq d$) in iteration i depends on how large are the infinity norms of T_j and T_i.

Lemma 9 *At the beginning of iteration i,* $\|T_j\|_\infty \leq i^{i/2}$ *for all* $j \geq i$.

Proof: Consider any $j \geq i$. The normalization step of iteration $i-1$ sets T_j to the vector $\det(\mathcal{M}_i)(v_{j1}, \ldots, v_{j(i-1)}, 0, \ldots, 1, 0, \ldots)$ where $(v_{j1}, \ldots, v_{j(i-1)})$ is the solution to the set of equations:

$$\begin{pmatrix} \tilde{C}_1^T \\ \vdots \\ \tilde{C}_{i-1}^T \end{pmatrix} \vec{x} = \begin{pmatrix} -c_{1j} \\ \vdots \\ -c_{(i-1)j} \end{pmatrix} \tag{6}$$

and $\mathcal{M}_i$ is the $(i-1) \times (i-1)$ matrix of $\tilde{C}$'s on the left of Equation (6). Since the entries of this matrix are $-1, 0, 1$, by Hadamard's inequality we can bound the absolute value of $\det(\mathcal{M}_i)$ by $i^{i/2}$. Similarly by Cramer's rule, each $v_{jk} = n_{jk}/\det(\mathcal{M}_i)$ where n_{jk} is the determinant of the $(i-1) \times (i-1)$ matrix obtained by replacing the k-th column of $\mathcal{M}_i$ by the vector on the right hand side of Equation (6). Since the entries of this matrix again are $-1, 0, 1$, this determinant has absolute value at most $i^{i/2}$. Hence for each k, $|\det(\mathcal{M}_i) \cdot v_{jk}| \leq i^{i/2}$. So the infinity norm of T_j is at most $i^{i/2}$. ∎

So each coordinate of T_j and T_i has bit size at most $\tilde{O}(i)$. Hence arithmetic on such numbers can be implemented in $\tilde{O}(i)$ time. So the update step for any T_j in iteration i can be implemented in $\tilde{O}(i^2)$ time. If the infinity norm of each T_j before the update step is $O(i^{i/2})$, then the infinity norm of any T_j after the update step is at most $O(i^i)$. So each coordinate of T_j still has bit size $\tilde{O}(i)$ and so the arithmetic for the normalization step also can be implemented in $\tilde{O}(i^2)$ time.

So the cost of updating and normalizing $T_{i+1}, \ldots, T_d$ in iteration i is $\tilde{O}((d-i)i^2)$ or $\tilde{O}(m^3)$.

5.3 Computing B_t more efficiently

Since $T_i = S_i$ at the beginning of iteration i, Lemma 1 tells us that we can compute the cycle C_i as $\min(B_0, B_1, \ldots, B_{f(i)})$, where B_t is a shortest cycle whose inner product with S_i is non-zero modulo the prime p_t and $f(i) = \lceil i/2 \rceil$. Each cycle B_t can be computed by the subroutine given in Section 4. However this subroutine has a running time of $\tilde{O}(m^2 n)$ which would make the computation of C_i take $\tilde{O}(m^3 n)$ time. This wastes the improvement that we achieved in computing T_i more efficiently.

An improved algorithm for computing the cycle B_t was given in [13]. It computes B_t in $\tilde{O}(mn)$ time as follows. We will not build the graph $U_{i,t}$ explicitly. Whenever we are at a vertex u_j, we know its neighborhood as follows. If there is an arc a between two vertices u and w in G, then in $U_{i,t}$, the vertex w_k is a neighbor of u_j where $k = (j + S_i(a)) \bmod p_t$ if $a = (u, w)$ (directed from u to w), and $k = (j - S_i(a)) \bmod p_t$ if $a = (w, u)$ (directed from w to u).

Recall that by Lemma 4, the $\min_v \min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path in $U_{i,t}$ corresponds to the cycle B_t in G. The key observation here is that in order to compute $\min_{\ell \neq 0}$ shortest (v_0, v_ℓ) path in $U_{i,t}$, 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 $\{u_0, u_1, \ldots, u_{p_t - 1}\}$. Note that we can impose a strict ordering on the vertices in $U_{i,t}$ (with respect to their distance from v_0) according to the order Dijkstra's algorithm extracts vertices from the priority queue. So if u_i is extracted before u_j, then we say that u_i is closer to v_0.

Since we will process only 2 vertices of "type" u for each u in the subroutine below, 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 $O(p_t n)$ iterations and the number of edges that we look at is also $O(m)$ and not $\Theta(p_t m)$.

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

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

and update predecessor of w to x if necessary.

– else do nothing.

Remark. Recall that there is always a (v_0, v_ℓ) path for some $\ell \neq 0$ in the graph $U_{i,t}$ for each $v \in G$ whenever S_i mod p_t is not the zero vector.

Running time of the above subroutine. 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 $\{u_0, \ldots, u_{p_t-1}\}$ 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 $O(\log(p_t n)) = O(\log n)$ 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 $O(p_t n)$ time to build the binary heap. So the total time to compute $\min_{\ell \neq 0}(v_0, v_\ell)$ path is $O(m \log n)$. Once we compute $\min_{\ell \neq 0}$shortest(v_0, v_ℓ) path for all $v \in V$, we have determined B_t. This takes time $O(n \cdot m \log n)$. So the total amount of time to compute all the cycles $B_0, \ldots, B_{f(i)}$ is $O(f(i)(n \cdot m \log n))$ which is $\tilde{O}(m^2 n)$. All we need to show now is the following lemma.

Lemma 10 *In order to compute* $\min_{\ell \neq 0}$ *shortest* (v_0, v_ℓ) *path in* $U_{i,t}$*, 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 $\{v_1,\ldots,v_{p_t-1}\}$. 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 $U_{i,t}$, we know that if there is a (u_k, v_r) path, then there is also a $(u_j, v_{(r+k-j) \bmod p_t})$ path which is just a translation of the (u_k, v_r) path, for every $j = 0,1,\ldots,p_t-1$. (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 $(u_f, v_{(r+k-f) \bmod p_t})$ path described above to the shortest-path(v_0, u_f). This gives us a path from v_0 to $v_{(r+k-f) \bmod p_t}$, which is shorter than q. So $v_{(r+k-f) \bmod p_t}$ is closer than v_r to v_0. But there is one catch: it could be the case that $(r+k-f) \bmod p_t = 0$.

Then consider the path which is obtained by adding the analogous path from u_s to $v_{(r+k-s) \bmod p_t}$ 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 $(r+k-f) \bmod p_t$ and $(r+k-s) \bmod p_t$ are 0 because that would mean that $f-s = 0(\bmod p_t)$ but f and s are distinct numbers in $\{0,\ldots,p_t-1\}$. 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 above lemma also shows that the shortest paths to vertices which are closest or second closest among their type to v_0 are correctly computed by the subroutine. This completes its proof of correctness and we can conclude the following lemma.

Lemma 11 *A shortest cycle C_i whose inner product with T_i is non-zero can be computed in $\tilde{O}(m^2 n)$ time.*

Now we can analyse the total running time of our algorithm presented in Section 5.1 to compute a minimum cycle basis. Right at the beginning of the algorithm, we reorder the arcs so that $a_{d+1},\ldots,a_m$ are the arcs of a spanning tree and let us also compute the primes $p_0, p_1, \ldots, p_{f(d)}$ using a sieving algorithm. These steps take $\tilde{O}(m)$ time. From Lemma 11 we know that the time taken to compute the cycle C_i is $\tilde{O}(m^2 n)$. We saw in Section 5.2 that the running time of the i-th iteration of the main For loop, excluding the time to compute the cycle C_i, is $\tilde{O}(m^3)$. Since there are d iterations, the total running time of the algorithm is $\tilde{O}(m^4)$. Thus we have shown the following theorem.

Theorem 3 *A minimum cycle basis of a directed graph G with m arcs and non-negative weights on its arcs can be computed in $\tilde{O}(m^4)$ time.*

6 A verification algorithm

Let us consider the following problem. We are given a directed graph G and set of d cycles $C_1, \ldots, C_d$ in G and we are asked to check if these cycles form a minimum cycle basis of G. It follows from Theorem 3 and the result in [18] that this can be done in $\tilde{O}(\min(m^4, m^{\omega+1}n))$ time, where $\omega < 2.376$. Here we present an $\tilde{O}(m^3 n)$ algorithm for this problem.

The verification algorithm.

1. Compute a spanning tree T of the underlying undirected graph of G.
2. Form the $d \times d$ matrix $\mathcal{C} = \left(C_1^T \ldots C_d^T\right)$, where the i-th row of $\mathcal{C}$ is the incidence vector of C_i over the edges of $G \setminus T$.
3. If $\mathcal{C}$ is singular, then declare no and quit. ($C_1, \ldots, C_d$ are linearly dependent and so they do not form a cycle basis.)

   Else, compute the adjoint $\mathcal{S}$ of $\mathcal{C}$. (The matrix $\mathcal{S} = \det(\mathcal{C}) \cdot \mathcal{C}^{-1}$.)
4. Call the columns of $\mathcal{S}$ as $S_1, \ldots, S_d$. For $i = 1, \ldots, d$ do:
   - append S_i with $n-1$ 0's so that S_i is now an m-vector.
   - if C_i is not a shortest cycle such that $\langle C_i, S_i \rangle \neq 0$, then declare no and quit.
5. Declare yes.

The correctness of our algorithm relies on the following lemma. It was originally shown by De Pina [4] for the case of undirected graphs, where the base field is $\mathbb{GF}(2)$.

Lemma 12 *Let $S_1, \ldots, S_d$ be linearly independent vectors in $\mathbb{Z}^m$. Let A_i be a shortest cycle such that $\langle A_i, S_i \rangle \neq 0$. Then $\sum_{i=1}^d \mathrm{weight}(A_i) \leq$ weight of a minimum cycle basis.*

Proof: We can rearrange the vectors S_i's so that the cycles A_i's are in sorted order i.e., $\mathrm{weight}(A_1) \leq \mathrm{weight}(A_2) \leq \cdots \leq \mathrm{weight}(A_d)$. Let $\{B_1, \ldots, B_d\}$ be a minimum cycle basis and let $\mathrm{weight}(B_1) \leq \mathrm{weight}(B_2) \leq \cdots \leq \mathrm{weight}(B_d)$. We will show that for each $i, \mathrm{weight}(A_i) \leq \mathrm{weight}(B_i)$. That will prove the lemma.

We will first show that $\langle B_j, S_k \rangle \neq 0$ for some j and k with $j \leq i \leq k$. Otherwise, the $d-i+1$ linearly independent vectors $S_i, S_{i+1}, \ldots, S_d$ belong to the subspace orthogonal to $B_1, \ldots, B_i$; however, this subspace has dimension only $d-i$. Hence we have $\langle B_j, S_k \rangle \neq 0$ for some j and k with $j \leq i \leq k$. This means that weight$(A_k) \leq$ weight(B_j) since A_k is a shortest cycle such that $\langle A_k, S_k \rangle \neq 0$.

By the sorted order, weight$(A_i) \leq$ weight(A_k) and weight$(B_j) \leq$ weight(B_i). This implies that weight$(A_i) \leq$ weight(B_i). ∎

Correctness of the algorithm. The correctness of our algorithm follows directly from the above lemma. The algorithm says yes only when $\{C_1, \ldots, C_d\}$ are linearly independent and for each $1 \leq i \leq d$, C_i is a shortest cycle such that $\langle C_i, S_i \rangle \neq 0$. Then the weight of $\{C_1, \ldots, C_d\}$ is at most the weight of a minimum cycle basis (by Lemma 12). Since $\{C_1, \ldots, C_d\}$ are linearly independent, it has to be a minimum cycle basis.

When $\{C_1, \ldots, C_d\}$ are linearly independent, the algorithm says no only when for some i, there is a cycle B_i with weight$(B_i) <$ weight(C_i) and $\langle B_i, S_i \rangle \neq 0$. Then $\{C_1, \ldots, C_d\}$ is indeed not a minimum cycle basis because the set $\{C_1, \ldots, C_d\} \setminus \{C_i\} \cup \{B_i\}$ is linearly independent. To show this we replace the i-th row of the matrix $\mathcal{C}$ with the first d coordinates of B_i and multiply the resulting matrix with $\mathcal{S}$. The determinant of the product matrix is $\langle B_i, S_i \rangle \cdot \det(\mathcal{C})^{d-1} \neq 0$. So $\{C_1, \ldots, C_d\} \setminus \{C_i\} \cup \{B_i\}$ is linearly independent and its weight is smaller than the weight of $\{C_1, \ldots, C_d\}$. Hence $\{C_1, \ldots, C_d\}$ is not a minimum cycle basis.

Running Time of the algorithm. We need to account for two jobs:

- computing the matrix $\mathcal{S}$.
- checking that C_i is a shortest cycle such that $\langle C_i, S_i \rangle \neq 0$, for each i.

We first compute the determinant of the $d \times d$ matrix $\mathcal{C}$ to check if it is nonsingular. If it is nonsingular, then we compute the inverse of $\mathcal{C}$ and multiply each element of the inverse with the determinant of $\mathcal{C}$. This gives us the matrix $\mathcal{S}$. Computing the determinant and inverse of $\mathcal{C}$ takes $O(d^{\omega})$ steps. We also need to account for doing arithmetic since the numbers grow large here. Since entries in $\mathcal{C}$ are $-1, 0, 1$, these numbers are bounded by $d^{d/2}$. Assuming that arithmetic on such numbers takes $\tilde{O}(d)$ amount of time, the total time required to compute $\mathcal{S}$ is $\tilde{O}(d^{\omega+1})$.

Also, $\|S_i\|_{\infty} \leq d^{d/2}$ since each coordinate of S_i is the value of the determinant of some $(d-1) \times (d-1)$ submatrix of $\mathcal{C}$. Since $\|S_i\|_{\infty} \leq d^{d/2}$, Lemma 1 tells us that a shortest cycle D_i whose inner product with S_i is non-zero can be computed as $\min(B_0, B_1, \ldots, B_{\lceil d/2 \rceil + 1})$, where B_t is a shortest cycle whose inner product with S_i is non-zero modulo the prime p_t. From the subroutine given in Section 5.3 and Lemma 11, we have an $\tilde{O}(m^2 n)$ time algorithm to check if the weight of C_i is the same as the weight of D_i. So the total running time of our verification algorithm is $\tilde{O}(m^{\omega+1} + m^3 n)$, which is $\tilde{O}(m^3 n)$ since we can assume that $m \leq n^2$ which implies $m^{\omega+1} < m^3 n$. Hence we can conclude the following theorem.

Theorem 4 *Given a directed graph G and a set of d cycles in G, it can be verified in $\tilde{O}(m^3 n)$ time if these cycles form a minimum cycle basis of G.*

7 Conclusions

In this paper we present an $\tilde{O}(m^4 n)$ algorithm and an $\tilde{O}(m^4)$ algorithm to compute a minimum cycle basis in a directed graph with nonnegative weights on its arcs. We also present an $\tilde{O}(m^3 n)$ algorithm to verify if a given cycle basis is a minimum cycle basis. An open problem was to design an $\tilde{O}(m^3 n)$ algorithm to compute a minimum cycle basis in a directed graph. Such an algorithm was recently given in [9].

The approaches used so far to compute minimum cycle bases in directed and undirected graphs are essentially greedy. The greedy approach works for these problems since the cycle bases of both directed and undirected graphs form a matroid. Liebchen and Rizzi [18] showed that the greedy approach cannot be used for minimizing over more specialized subclasses of directed cycle bases, like integral cycle bases and generalized fundamental cycle bases[2]. No polynomial time algorithm is known for computing a minimum cycle basis in these specialized classes. It is an open problem to settle the complexity status of these problems.

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.

[2] A cycle basis $\{C_1, \ldots, C_d\}$ is called generalized fundamental if there is a permutation σ, such that $C_{\sigma(i)} \setminus \{C_{\sigma(1)}, \ldots, C_{\sigma(i-1)}\} \neq \emptyset$ for all $i = 2, \ldots, d$.

[3] B. Bollobás. *Modern Graph Theory,* volume 184 of *Graduate Texts in Mathematics*, Springer, Berlin, 1998.

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

[5] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, 1982.

[6] G. M. Downs, V. J. Gillet, J. D. Holliday, and M. F. Lynch. Review of ring perception algorithms for chemical graphs. *J. Chem. Inf. Comput. Sci.*, 29: 172-187, 1992.

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

[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] R. Hariharan, T. Kavitha, and K. Mehlhorn. A Faster Deterministic Algorithm for Minimum Cycle Basis in Directed Graphs, preprint.

[10] D. Hartvigsen and R. 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.

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

[12] A. Kaveh. *Structural Mechanics: Graph and Matrix Methods.* Research Studies Press, Exeter, UK, 1992.

[13] T. Kavitha. An $\tilde{O}(m^2n)$ Randomized Algorithm to compute a Minimum Cycle Basis of a Directed Graph. *In Proc. of ICALP*, LNCS 3580: 273-284, 2005.

[14] 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.

[15] 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.

[16] C. Liebchen. Finding Short Integral Cycle Bases for Cyclic Timetabling. *In Proc. of ESA*, LNCS 2832: 715-726, 2003.

[17] C. Liebchen and L. Peeters. On Cyclic Timetabling and Cycles in Graphs. Technical Report 761/2002, TU Berlin.

[18] C. Liebchen and R. Rizzi. A greedy approach to compute a minimum cycle basis of a directed graph. Information Processing Letters, 94(3): 107-112, 2005.