New Approximation Algorithms for Minimum Cycle Bases of Graphs

Telikepalli Kavitha[1], Kurt Mehlhorn[2], and Dimitrios Michail[2]

[1] Indian Institute of Science, Bangalore, India
kavitha@csa.iisc.ernet.in
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{mehlhorn,michail}@mpi-inf.mpg.de

Abstract. We consider the problem of computing an approximate minimum cycle basis of an undirected edge-weighted graph G with m edges and n vertices; the extension to directed graphs is also discussed. In this problem, a $\{0,1\}$ incidence vector is associated with each cycle and the vector space over $\mathbb{F}_2$ generated by these vectors is the cycle space of G. A set of cycles is called a cycle basis of G if it forms a basis for its cycle space. A cycle basis where the sum of the weights of the cycles is minimum is called a minimum cycle basis of G. Cycle bases of low weight are useful in a number of contexts, e.g. the analysis of electrical networks, structural engineering, chemistry, and surface reconstruction.

We present two new algorithms to compute an approximate minimum cycle basis. For any integer $k \geq 1$, we give $(2k-1)$-approximation algorithms with expected running time $O(kmn^{1+2/k} + mn^{(1+1/k)(\omega-1)})$ and deterministic running time $O(n^{3+2/k})$, respectively. Here ω is the best exponent of matrix multiplication. It is presently known that $\omega < 2.376$. Both algorithms are $o(m^\omega)$ for dense graphs. This is the first time that any algorithm which computes sparse cycle bases with a guarantee drops below the $\Theta(m^\omega)$ bound.

We also present a 2-approximation algorithm with $O(m^\omega\sqrt{n \log n})$ expected running time, a linear time 2-approximation algorithm for planar graphs and an $O(n^3)$ time 2.42-approximation algorithm for the complete Euclidean graph in the plane.

1 Introduction

Let $G = (V, E)$ be an undirected connected graph with m edges and n vertices. A *cycle* of G is any subgraph of G where each vertex has even degree. Associated with each cycle C is an *incidence vector* x, indexed on E, where for any $e \in E$, x_e is 1 if e is an edge of C and 0 otherwise. The vector space over $\mathbb{F}_2$ generated by the incidence vectors of cycles is called the *cycle space* of G. It is well known that this vector space has dimension $N = m - n + 1$, where m is the number of edges of G and n is the number of vertices. A maximal set of linearly independent cycles is called a *cycle basis*. The edges of G have non-negative weights assigned to them. A cycle basis where the sum of the weights of the cycles is minimum is called a *minimum cycle basis* of G. We use the abbreviation MCB to refer

to a minimum cycle basis. Minimum cycle bases are of considerable practical importance and therefore the problem of computing an MCB has received considerable attention. An early paper is by Stepanec [1]. Horton [2] presented the first polynomial time algorithm. Faster and/or alternative algorithms were later presented by de Pina [3], Golynski and Horton [4], Berger et al. [5], and Kavitha et al. [6]. The current fastest algorithm [6] has running time $O(m^2n+mn^2 \log n)$. Implementations are discussed in [7–9].

An important application of the MCB problem is the construction of sparse systems when solving problems in electrical networks [10, 3, 5]. Other applications are in structural engineering [11], chemistry and biochemistry [12], and surface reconstruction from point clouds [13]. In most applications, the computation of an MCB is a preprocessing step. The use of an MCB ensures sparseness and translates into faster running times of the main algorithm. Unfortunately, even the fastest exact minimum cycle basis algorithm has a running time of $\Theta(m^2n + mn^2 \log n)$. This may dominate the running time of the application.

However, most applications can work with any cycle basis and any constant factor approximate minimum cycle basis may be substituted for a minimum cycle basis without much affect on the application. In [6] an α-approximation algorithm for any $\alpha > 1$ is presented for the MCB problem; its running time is $o(m^2n + mn^2 \log n) + \Theta(m^\omega)$, where ω is the exponent of matrix multiplication. It is known [14] that $\omega < 2.376$. The time bound of $\Theta(m^\omega)$ is still prohibitive for some of the applications. It results from Gaussian elimination on $m \times m$ linear systems.

We present a new approximation approach which leads to vastly improved time bounds. In particular, for any integer $k \geq 1$, we give two $(2k-1)$ approximation algorithms with expected running time $O(kmn^{1+2/k} + mn^{(1+1/k)(\omega-1)})$ and deterministic running time $O(n^{3+2/k})$, respectively. Both algorithms are $o(m^\omega)$ for sufficiently dense graphs, the first algorithm for number of edges $m > \max(n^{1+1/k}, \sqrt[\omega-1]{kn^{1+2/k}})$ and the second algorithm for $m > n^{\frac{3}{\omega}+\frac{2}{k\omega}} = n^{1.26+\frac{0.84}{k}}$. The first algorithm is faster for sparser graphs and the second algorithm for denser graphs. More precisely, the second algorithm is faster for $m > n^{4-\omega+\frac{3-\omega}{k}}$ which with the current upper bound on ω is $m > n^{1.624+\frac{0.624}{k}}$.

Our algorithms work in two phases. The first phase is a very fast computation of a large number of cycles (all but $O(n^{1+1/k})$ cycles) in an approximate MCB. The second part is a more expensive computation of the remaining cycles. We present two different ways for computing these remaining cycles, leading to the above two algorithms, each faster for different graph densities. Only the second phase needs a null space computation; it is a null space computation of a square system of size $O(n^{1+1/k})$. Our new algorithms are fast even when implemented without fast matrix multiplication. Furthermore, by combining the techniques of both the algorithms, we get an even faster algorithm at the expense of a larger approximation factor.

We also present a 2-approximation algorithm with $O(m^\omega \sqrt{n \log n})$ expected running time. For sparse graphs, this is subcubic. Moreover, we develop very fast approximation algorithms for some special graph classes. For planar graphs

we give a linear time 2-approximation algorithm and for the complete Euclidean graph in the plane we give a 2.42-approximation algorithm with running time $O(n^3)$. In higher dimensions we give a k-approximation algorithm for any $k > 1$ with running time $O(s^d n^3 \log n)$ where $s = 4(k+1)/(k-1)$ and d is the fixed dimension.

The minimum cycle basis problem for directed graphs is less studied. Polynomial time algorithms are given in [15–18]. The fastest deterministic algorithm [17] has running time $O(m^3 n + m^2 n^2 \log n)$ and there is a Monte Carlo algorithm [18, 17] with running time $O(m^2 n + mn^2 \log n)$. Some of our algorithms generalize to directed graphs. We give a deterministic $(2k-1)$-approximation algorithm with running time $O(n^{4+3/k})$, a Monte Carlo $(2k-1)$-approximation algorithm with running time $O(n^{3+2/k})$, and a 2-approximation algorithm with expected running time $O(m^\omega \sqrt{n \log n})$.

Preliminaries. Let T be any spanning tree in $G(V, E)$, let $e_1, \ldots, e_N$ be the edges of $E \setminus T$ in some arbitrary but fixed order, and let $e_{N+1}, \ldots, e_m$ be the edges in T in some arbitrary but fixed order. We frequently view cycles in terms of restricted incidence vectors[3], that is, each cycle is a vector in $\{0,1\}^N$.

We use S and R to denote subsets of $E \setminus T$. Each such subset gives rise to an incidence vector in $\{0,1\}^N$. We use $\langle C, S \rangle$ to denote the standard inner product of vectors C and S. We say that a vector S is *orthogonal* to C if $\langle C, S \rangle = 0$. In the field $\mathbb{F}_2$, $\langle C, S \rangle = 1$ if and only if C contains an odd number of edges of S.

For a cycle C, we use $w(C) = \sum_{e \in C} w(e)$ to denote its weight. We use $w_G(\text{MCB})$ to denote the weight of a minimum cycle basis of graph G. When it is clear by the context we omit G and write $w(\text{MCB})$. The following lemma gives us a lower bound on $w(\text{MCB})$. See [6] for a proof.

Lemma 1 (de Pina [3]). *Let $R_1, \ldots, R_N$ be linearly independent vectors in $\{0,1\}^N$ and let A_i be a shortest cycle in G such that $\langle A_i, R_i \rangle = 1$. Then $\sum_{i=1}^N w(A_i) \leq w(\text{MCB})$.*

The sets $R_i = \{e_i\}$, $1 \leq i \leq N$, are clearly independent. The shortest cycle C with $\langle C, R_i \rangle = 1$ consists of the edge e_i plus the shortest path in $G \setminus \{e_i\}$ connecting its endpoints. We use SC_i to denote this cycle. The cycle exists, since there is always the spanning tree path in $E \setminus \{e_i\}$ connecting the endpoints of e_i. Let $\mathsf{SC} = \{SC_i \mid 1 \leq i \leq N\}$ be the shortest cycle multiset and let $w(\mathsf{SC}) = \sum_{C \in \mathsf{SC}} w(C)$ be its weight. By applying Lemma 1 to the cycles in SC, we obtain Lemma 2.

Lemma 2. $w(\mathsf{SC}) \leq w(\text{MCB})$.

[3] For a cycle C, use C to denote its incidence vector in $\{0,1\}^N$ (restricted to $e_1, \ldots, e_N$) and C^* to denote its incidence vector in $\{0,1\}^m$. Consider a set of cycles $C_1, \ldots, C_k$. Clearly, if the vectors C_1^* to C_k^* are dependent, then so are the vectors C_1 to C_k. Conversely, assume that $\sum_i \lambda_i C_i = 0$. Then $C = \sum_i \lambda_i C_i^*$ contains only edges in T. Moreover, since C is a sum of cycles, each vertex has even degree with respect to C. Thus, $C = 0$ and hence linear dependence of the restricted incidence vectors implies linear dependence of the full incidence vectors. Thus we may restrict attention to the restricted incidence vectors when discussing questions of linear independence.

2 The new approach

Our approximation algorithms are motivated by the shortest cycle multiset lower bound (Lemma 2). We fix a parameter $\lambda \leq N$ and construct a set of linearly independent cycles $C_1, \ldots, C_\lambda$ such that $w(C_i) \leq (2k-1) \cdot w(SC_i)$ for $1 \leq i \leq \lambda$. In the second phase, we extend the partial basis to a full basis. We offer two alternatives for the second phase. Let $t = 2k - 1$.

Now we give the details of the first phase. We construct the cycles $C_1, \ldots, C_\lambda$ using a sparse t-spanner of G. A *multiplicative* t-spanner of a graph G is a subgraph $G'(V, E')$, $E' \subseteq E$ such that for any $u, v \in V$ we have $w(\mathrm{SP}_{G'}(u, v)) \leq t \cdot w(\mathrm{SP}_G(u, v))$ where $\mathrm{SP}_G(u, v)$ denotes a shortest path in G from u to v. When it is clear from the context we omit the subscript G and write $\mathrm{SP}(u, v)$. Let $G'(V, E')$ be such a t-spanner of G. Since we can always add the edges in the spanning tree T to E', we may assume $T \subseteq E'$. We also assume that the edges are indexed such that $E \setminus E' = \{e_1, \ldots, e_\lambda\}$.

For each edge $e_i = (u, v) \in E \setminus E'$, let C_i be formed by e_i and $\mathrm{SP}_{G'}(u, v)$. The cycles C_i, $1 \leq i \leq \lambda$ are clearly independent since e_i is contained in precisely C_i. We have $w(C_i) = w(e_i) + w(\mathrm{SP}_{G'}(u, v)) \leq w(e_i) + t \cdot w(\mathrm{SP}_G(u, v)) \leq t \cdot w(SC_i)$.

The running time of phase 1 is easily estimated. As pointed out by Althöfer et al. [19] every weighted undirected graph on n vertices has a $(2k-1)$-spanner with $O(n^{1+1/k})$ edges where $k \geq 1$ is an integer. Such a spanner can be constructed using an algorithm similar to Kruskal's algorithm for constructing minimum spanning trees. In order to build the spanner, consider all edges of the graph in non-decreasing order of weight, adding each edge to the spanner if its endpoints are not already connected, in the spanner, by a path using at most $2k - 1$ edges. At any stage, the spanner is a $(2k - 1)$-spanner of the edges already considered, and its unweighted girth is at least $2k + 1$, so it has only $O(n^{1+1/k})$ edges. The above procedure can be implemented in $O(mn^{1+1/k})$ time.

In the above spanner we are going to perform λ shortest path computations, one for each edge of G that is not in the spanner. Using Dijkstra's algorithm we need $O(\lambda \cdot (n^{1+1/k} + n \log n))$ time and since $\lambda \leq m$ we can compute both the spanner and the λ linearly independent cycles in time $O(mn^{1+1/k})$. We should mention that there are faster algorithms to construct similar spanners, see for example [20]. However, the construction by Althöfer et al. suffices for our purposes.

3 The remaining cycles

In the preceding section we computed most of the cycles of an approximate minimum cycle basis. We are left with computing the remaining cycles. The number of additional cycles is $N - \lambda$. Note that this is exactly the dimension of the cycle space of the spanner G'. We present two different algorithms. The first approach uses all the edges in G to construct the remaining cycles while the second approach uses only the edges $e_{\lambda+1}, \ldots, e_m$ of the spanner.

3.1 The first approach

We first need to briefly review the algorithm in [6] in order to compute a minimum cycle basis; it refines a previous algorithm by de Pina [3]. The algorithm is recursive. We immediately describe the modification of the algorithm needed for our purposes.

The general step adds some number k of cycles to a partial basis PB of size α. This step takes as input an integer $k \geq 1$, and k linearly independent vectors $S_{\alpha+1}, \ldots, S_{\alpha+k}$ orthogonal to the cycles in PB. These vectors, viewed as sets, have the additional property that $S_{\alpha+i} \cap \{e_{\alpha+1}, \ldots, e_N\} = \{e_{\alpha+i}\}$ for $1 \leq i \leq k$. The step updates $S_{\alpha+1}, \ldots, S_{\alpha+k}$ and returns k cycles $Z_{\alpha+1}, \ldots, Z_{\alpha+k}$ such that $\langle Z_i, S_j \rangle = \delta_{ij}$ for $\alpha + 1 \leq i \leq j \leq \alpha + k$ (here δ_{ij} is 1 if $i = j$ and 0 otherwise). The update has the additional property that it does not affect the orthogonality w.r.t the partial basis PB. Observe, that the cycles $\mathsf{PB} \cup \{Z_{\alpha+1}, \ldots, Z_{\alpha+k}\}$ are linearly independent. To see this note that for any $1 \leq i \leq k$, $\langle Z_{\alpha+i}, S_{\alpha+i} \rangle = 1$ while any cycle C in the span of $\mathsf{PB} \cup \{Z_{\alpha+1}, \ldots, Z_{\alpha+i-1}\}$ has $\langle C, S_{\alpha+i} \rangle = 0$.

The top level call: We call the recursive procedure with the partial basis of phase 1, namely $\mathsf{PB} = \{C_1, \ldots, C_\lambda\}$ and ask it to compute $\mu = N - \lambda$ additional cycles. Let us write the $C_1, \ldots, C_\lambda$ in the form of a $\lambda \times N$ matrix with one row per cycle. Then

$$\begin{pmatrix} C_1 \\ \vdots \\ C_\lambda \end{pmatrix} = \begin{pmatrix} I_\lambda \; B \end{pmatrix} \tag{1}$$

where I_λ is the $\lambda \times \lambda$ identity matrix and B is a $\lambda \times \mu$ matrix. The matrix has this form since each of the edges e_i for $1 \leq i \leq \lambda$ belongs only to the cycle C_i. Set

$$\begin{pmatrix} S_{\lambda+1} \ldots S_{\lambda+\mu} \end{pmatrix} = \begin{pmatrix} B \\ I_\mu \end{pmatrix}. \tag{2}$$

Then the product of the matrix of C's on the left side of Equation (1) and the matrix of S's on the left side of Equation (2) is $B + B = 0$, i.e., the S's are orthogonal to the C's. Moreover, $S_{\lambda+i} \cap \{e_{\lambda+1}, \ldots, e_N\} = e_{\lambda+i}$ for $1 \leq i \leq \mu$. The running time required to compute this null space basis is the time required to output the already known matrix B. By using some sparse representation of the vectors we need at most $O(\lambda \cdot \mu)$ time. In the general case $\lambda \leq m$ and $\mu = N - \lambda \in O(n^{1+1/k})$. Thus, initialization of phase 2 needs $O(mn^{1+1/k})$ time.

The recursive case, $k \geq 2$: Let $\ell = \lceil k/2 \rceil$. We first call the algorithm recursively with ℓ and $S_{\alpha+1}$ to $S_{\alpha+\ell}$. The call will return cycles $Z_{\alpha+1}$ to $Z_{\alpha+\ell}$ and updated sets $S_{\alpha+1}$ to $S_{\alpha+\ell}$. We next update the sets $S_{\alpha+\ell+1}$ to $S_{\alpha+k}$. The set $S_{\alpha+j}$, $\ell + 1 \leq j \leq k$, is replaced by a sum $S_{\alpha+j} + \sum_{1 \leq i \leq \ell} \beta_{ji} S_{\alpha+i}$ where the β_{ji} are chosen such that the updated $S_{\alpha+j}$ becomes orthogonal to the cycles $Z_{\alpha+1}$ to $Z_{\alpha+\ell}$. Observe that orthogonality to the cycles in PB is not affected. The update step is implemented using fast matrix multiplication and takes time $O(mk^{\omega-1})$. The final step is to call the algorithm recursively for the remaining

cycles. We therefore have the following recursion for the running time: $T(k) = T(\lceil k/2 \rceil) + T(\lfloor k/2 \rfloor) + O(mk^{\omega-1})$ for $k \geq 2$. This solves to $T(k) = k \cdot T(1) + O(mk^{\omega-1})$. We call the algorithm with $k = \mu$ and hence have total running time $\mu \cdot T(1) + O(m\mu^{\omega-1})$.

The base case, $k = 1$: The algorithm computes a t-approximate shortest[4] cycle C with $\langle C, S_{\alpha+1} \rangle = 1$. The shortest cycle C with $\langle C, S_{\alpha+1} \rangle = 1$ can be computed as follows. We set up an auxiliary graph $G^\dagger$ with two copies, say v' and v'', for each vertex v, and two copies e' and e'' for each edge $e = (u, v) \in E$. If $e \in S_{\alpha+1}$, the copies are (u', v'') and (u'', v') and if $e \notin S_{\alpha+1}$, the copies are (u', v') and (u'', v''). Then a shortest cycle C with $\langle C, S_{\alpha+1} \rangle = 1$ corresponds to a shortest path connecting the two copies of some vertex v minimized over all v. Such a path can be found by n shortest path computations in the auxiliary graph. In order to compute a t-approximate shortest cycle C with $\langle C, S_{\alpha+1} \rangle = 1$ we compute t-approximate single source shortest paths between n pairs of vertices.

We need to perform a total of μn approximate shortest path computations. Therefore, we require a faster algorithm than constructing a spanner. We use an *approximate distance oracle*. Thorup and Zwick [20] constructed a structure which answers $(2k-1)$-approximate shortest path queries in time $O(k)$. The structure requires space $O(kn^{1+1/k})$ and can be constructed in expected time $O(kmn^{1/k})$.

Using such a construction, we bound $T(1)$ by the cost of computing the approximate distance oracle ($O(kmn^{1/k})$ expected time) and the cost of performing n queries to the oracle. Each query costs $O(k)$ and thus a total cost of $O(nk)$. Forming the actual cycle can be done in time linear to its length which is $O(n)$. Thus, $T(1) = O(kmn^{1/k})$ and therefore $T(\mu) = O(\mu kmn^{1/k} + m\mu^{\omega-1})$. Since $\mu \in O(n^{1+1/k})$ we get a bound of $O(kmn^{1+2/k} + mn^{(1+1/k)(\omega-1)})$.

Approximation guarantee. We prove that the computed set of cycles is a t-approximation of the MCB. Consider the vectors $S_{\lambda+1}, \ldots, S_N$ at the end of the algorithm and define $S_i = \{e_i\}$ for $1 \leq i \leq \lambda$. Then each C_i, $1 \leq i \leq N$, is a t-approximation of the shortest cycle in G having odd intersection with S_i. All we need to show is Lemma 3. Then the approximation guarantee follows from Lemma 1.

Lemma 3. *The vectors $S_1, \ldots, S_N$ are linearly independent.*

Proof. Consider any i. We have $\langle C_i, S_i \rangle = 1$ and $\langle C_i, S_j \rangle = 0$ for all $j \geq i+1$. The latter holds for $j > \lambda$ by the invariants of the recursive procedure and it holds for $i < j \leq \lambda$ since C_i consists of edge e_i and edges in the spanner (which have index greater than λ) and $S_j = \{e_j\}$. Thus, S_i is independent of the $S_{i+1}, \ldots, S_N$ and the lemma follows.

Theorem 1. *For any integer $k \geq 1$, a $(2k-1)$-approximate* MCB *can be computed in expected time $O(kmn^{1+2/k} + mn^{(1+1/k)(\omega-1)})$ in undirected weighted graphs. An $O(\log n)$-approximate* MCB *in expected time $O(mn^{\omega-1} + mn \log n)$.*

[4] The original algorithm in [6] constructs a shortest cycle.

3.2 The second approach

Our second algorithm to compute the remaining cycles of our cycle basis, just computes a minimum cycle basis of the t-spanner G'. The dimension of the cycle space of G' is $\mu = N - \lambda$ and thus we have the right number of cycles. Let $C_{\lambda+1}, \ldots, C_N$ be an MCB of G'. Cycles $\{C_1, \ldots, C_\lambda\} \cup \{C_{\lambda+1}, \ldots, C_N\}$ are by definition linearly independent and we are also going to prove that they form a t-approximation of an MCB of G.

For $1 \leq i \leq \lambda$, we have $C_i = e_i + p_i$, where p_i is a shortest path in G' between the endpoints of e_i. In order to show that cycles $C_1, \ldots, C_N$ are a t-approximation of the MCB, we again define appropriate linearly independent vectors $S_1, \ldots, S_N \in \{0,1\}^N$ and use Lemma 1. Consider the exact algorithm in [6] executing with the t-spanner G' as its input. Other than the cycles $C_{\lambda+1}, \ldots, C_N$, the algorithm also returns vectors $R_{\lambda+1}, \ldots, R_N \in \{0,1\}^{N-\lambda}$ such that $\langle C_i, R_j \rangle = 0$ for $\lambda + 1 \leq i < j \leq N$ and C_i is a shortest cycle in G' such that $\langle C_i, R_i \rangle = 1$ for $\lambda+1 \leq i \leq N$. Moreover, the $(N-\lambda) \times (N-\lambda)$ matrix whose j-th row is R_j is lower triangular with 1 in its diagonal. This implies that the R_j's are linearly independent. Given any vector $S \in \{0,1\}^N$ let $\tilde{S}$ be the projection of S onto its last $N - \lambda$ coordinates. In other words, if S is an edge set of G, then let $\tilde{S}$ be the edge set restricted only to the edges of G'. We define S_j for $1 \leq j \leq N$ as follows. Let $S_1, \ldots, S_\lambda$ be the first λ unit vectors of $\{0,1\}^N$. For $\lambda + 1 \leq j \leq N$ define S_j as:

$$S_j = (-\langle \tilde{C}_1, R_j \rangle, \ldots, -\langle \tilde{C}_\lambda, R_j \rangle, R_{j,1}, R_{j,2}, \ldots, R_{j,(N-\lambda)}),$$

where $R_{j,1}, \ldots, R_{j,(N-\lambda)}$ are the coordinates of the vector $R_j \in \{0,1\}^{N-\lambda}$. Note that the vectors S_j for $1 \leq j \leq N$, defined above, are linearly independent. This is because the $N \times N$ matrix whose j-th row is S_j is lower triangular with 1's in its diagonal. The above definition of S_j's is motivated by the property that for each $1 \leq i \leq \lambda$, we have $\langle C_i, S_j \rangle = -\langle \tilde{C}_i, R_j \rangle + \langle \tilde{C}_i, R_j \rangle = 0$, since the cycle C_i has 0 in all first λ coordinates, except the i-th coordinate, which is 1. Lemma 4, shown below, together with Lemma 1, implies the correctness of our approach.

Lemma 4. *Consider the above defined S_j for $1 \leq j \leq N$ and let D_j be the shortest cycle in G such that $\langle D_j, S_j \rangle = 1$. Cycle C_j has weight at most t times the weight of D_j.*

Proof. This is obvious for $1 \leq j \leq \lambda$ since D_j is a shortest cycle in G which uses edge e_j and $C_j = e_j + p_j$, where p_j is a t-approximate shortest path between the endpoints of e_j. Consider now D_j for $\lambda + 1 \leq j \leq N$. If D_j uses any edge e_i for $1 \leq i \leq \lambda$ we replace it with the corresponding shortest path in the spanner. This is the same as saying consider the cycle $D_j + C_i$ instead of D_j. Let $D'_j = D_j + \sum_{1 \leq i \leq \lambda} (e_i \in D_j) C_i$ where $(e_i \in D_j)$ is 1 if $e_i \in D_j$ and 0 if $e_i \notin D_j$. Then

$$\langle D'_j, S_j \rangle = \langle D_j, S_j \rangle + \sum_{1 \leq i \leq \lambda} (e_i \in D_j) \langle C_i, S_j \rangle.$$

But recall that our definition of S_j ensures that $\langle C_i, S_j \rangle = 0$ for $1 \leq i \leq \lambda$. This implies that $\langle D'_j, S_j \rangle = \langle D_j, S_j \rangle = 1$. But D'_j by definition has 0 in the first λ coordinates and $\tilde{S}_j = R_j$, which in turn implies that $\langle \tilde{D}'_j, R_j \rangle = 1$.

C_j is a shortest cycle in G' such that $\langle C_j, R_j \rangle = 1$. Thus, C_j has weight at most the weight of $\tilde{D}'_j$, and by construction, D'_j has weight at most t times the weight of D_j.

Thus, we have shown that the cost of our approximate basis is at most t times the cost of an optimal basis. As a t-spanner we will again use a $(2k-1)$-spanner. The best time bound in order to compute an MCB is $O(m^2n + mn^2 \log n)$ and since a $(2k-1)$-spanner has at most $O(n^{1+1/k})$ edges the total running time becomes $O(n^{3+2/k})$.

Theorem 2. *A $(2k-1)$-approximate* MCB*, for any integer $k \geq 1$, in a weighted undirected graph can be computed in time $O(n^{3+2/k})$. An $O(\log n)$-approximate* MCB *can be computed in time $O(n^3 \log n)$.*

Further results. By combining the two approaches we can get even faster algorithms in the expense of an increased approximation ratio. Due to space restrictions, details of this and several following results can be found in the full version of this paper.

Our techniques for $2k-1$ approximate minimum cycle basis can also be applied to the minimum cycle basis problem in directed graphs. The problem definition is described in Section 4.1. We simply state our results here.

Theorem 3. *For any integer $k \geq 1$, a $(2k-1)$ approximate* MCB *of a directed graph with non-negative edge weights can be computed in time $O(n^{4+3/k})$. If we allow randomization it can be computed, with high probability, in time $O(n^{3+2/k})$.*

For some classes of graphs which admit better spanners, our approaches lead to very fast approximation algorithms. For the complete Euclidean graph in two dimensions we get a 2.42 approximation in time $O(n^3)$. Similar results can be obtained in higher (but fixed) dimensions. For planar graphs we get a linear time 2 approximation by just returning the list of bounded faces.

Practical considerations. Both approaches (Section 3.1 and 3.2) use fast matrix multiplication. However, they are also efficient even when used without fast matrix multiplication. This fact has high practical value since high performance fast matrix multiplication libraries are difficult to implement. Instead of the $O(m^2n + mn^2 \log n)$ algorithm to compute an MCB in G', use the $O(m^3 + mn^2 \log n)$ algorithm from [3], which is the fastest algorithm to compute an MCB without fast matrix multiplication.

Theorem 4. *A $(2k-1)$-approximate* MCB*, for any integer $k \geq 1$, can be computed in an undirected weighted graph without fast matrix multiplication in expected $O(kmn^{1+2/k} + mn^{2+2/k})$ and deterministic $O(n^{3+3/k})$ time respectively.*

Both our algorithms are $o(m^\omega)$ for sufficiently dense graphs and appropriate values of k. Moreover, they are easy to implement efficiently. Preliminary experiments suggest a significant speedup in practice.

4 A 2-approximation algorithm

For any undirected (connected) graph $G = (V, E)$ with n vertices and m edges, Horton [2] defined a set of $O(mn)$ cycles and proved that it contains an MCB. An MCB can be found by determining the least weight $N = m - n + 1$ linearly independent cycles from this set, using Gaussian elimination. We define a set of $O(m\sqrt{n \log n})$ cycles and show that it contains a 2-approximate minimum cycle basis; our set is a subset of Horton's set. Again, the basis is extracted from the set by determining the least weight N linearly independent cycles in it.

For a vertex $x \in V$ and an edge $e = (u, v) \in E$, let $C[x, e] = \mathrm{SP}(x, u) + e + \mathrm{SP}(v, x)$ be the cycle consisting of the edge e and the shortest paths from x to its endpoints. Horton's collection consists of the cycles $C[x, e]$ for all $x \in V$ and $e \in E$. We use a subset of Horton's collection.

Definition 1. *For $v, x \in V$ and $S \subset V$, $bunch(v, S)$ consists of all vertices closer to v than to any vertex in S and $cluster(x, S)$ consists of all vertices v with $x \in bunch(v, S)$.*

Lemma 5 (Thorup and Zwick [21]). *Given a weighted graph $G = (V, E)$ and $0 < q < 1$, one can compute a set $S \subset V$ of size $O(nq \log n)$ in expected time $O(m/q \log n)$ such that $|cluster(x, S)| = 1/q$ for all $x \in V$.*

We take a value $q = 1/\sqrt{n \log n}$ here and first compute in expected time $O(m\sqrt{n} \log^{3/2} n)$ a set $S \subset V$ of $O(\sqrt{n \log n})$ vertices as given in Lemma 5. This ensures that $cluster(v, S)$ has size $\sqrt{n \log n}$ for all $v \in V$. Also, $bunch(v, S)$ for all v can be computed in expected time $O(m/q)$ [20], which is $O(m\sqrt{n \log n})$. We use two types of cycles:

- the $O(m\sqrt{n \log n})$ cycles $C[s, e]$ for all $s \in S$ and $e \in E$,
- the cycles $C[u, e]$ for each $u \in V$ and $e = (v, w) \in E$ and either v or w in $bunch(u, S)$. The number of such cycles is $\sum_{u \in V} \sum_{v \in bunch(u,S)} \deg(v)$. This is the same as $\sum_{v \in V} \deg(v) \cdot |cluster(v, S)|$, which is $\sqrt{n \log n} \sum_{v \in V} \deg(v) = m\sqrt{n \log n}$.

Thus, our collection has $O(m\sqrt{n \log n})$ cycles. We need to show that it contains a 2-approximate cycle basis. Let $B_1, \ldots, B_N$ be the minimum cycle basis of G determined by Horton's algorithm in order of increasing weight, i.e., $w(B_1) \leq w(B_2) \leq \cdots \leq w(B_N)$. We show that each $B_i = \sum_{C \in \mathcal{C}_i} C$ where $\mathcal{C}_i$ is a subset of our collection and each cycle in $\mathcal{C}_i$ has cost at most $2w(B_i)$. This implies that our collection contains N linearly independent cycles $A_1, \ldots, A_N$ with $w(A_i) \leq 2 \cdot w(B_i)$ for $i = 1, \ldots, N$. Assume otherwise and let j be minimal such that $\cup_{i \leq j} \mathcal{C}_i$ contains less than j linearly independent vectors with $w(A_i) \leq 2 \cdot w(B_i)$ for $i = 1, \ldots, j$. Then $j \geq 1$ and $\cup_{i \leq j-1} \mathcal{C}_i$ contains at least $j-1$ linearly independent vectors with $w(A_i) \leq 2 \cdot w(B_i)$ for $i = 1, \ldots, j-1$. Also, $\cup_{i \leq j} \mathcal{C}_i$ spans $\{B_1, \ldots, B_i\}$ and hence contains at least i linearly independent vectors. Thus, it contains a vector A_j linearly independent from $\{A_1, \ldots, A_{j-1}\}$. Furthermore, $A_j \in \mathcal{C}_i$ for some $i \leq j$ and hence $w(A_j) \leq 2w(B_i) \leq 2w(B_j)$, a contradiction.

Consider any B_i. If B_i belongs to our collection, we set $\mathcal{C}_i = \{B_i\}$. Otherwise, $B_i = C[u, e]$ where $e = (v, w)$ and neither v nor w is in $bunch(u, S)$. Let $s \in S$ be the nearest vertex in S to u. Then, $w(\mathrm{SP}(s, u)) \leq w(\mathrm{SP}(u, v))$ and $w(\mathrm{SP}(s, u)) \leq w(\mathrm{SP}(u, w))$.

For any edge $f \in B_i$, the cycle $C(s, f)$ is in our collection and furthermore $B_i = \sum_{f \in B_i} C(s, f)$ since the paths from s to the endpoints of the edges in B_i appear twice in this sum and cancel out. We set $\mathcal{C}_i = \{C(s, f) \mid f \in B_i\}$. It remains to show $w(C(s, f)) \leq 2w(B_i)$ for all $f \in B_i$. We distinguish cases.

Assume first that $f \neq e$. Then $f \in \mathrm{SP}(u, v)$ or $f \in \mathrm{SP}(u, w)$. We may assume w.l.o.g. that the former is the case. Then $w(C(s, f)) \leq w(\mathrm{SP}(s, u)) + w(\mathrm{SP}(u, v)) + w(\mathrm{SP}(v, s))$ since $C(s, f)$ consists of f and the shortest paths from s to the endpoints of f and $w(\mathrm{SP}(v, s)) \leq w(\mathrm{SP}(s, u)) + w(\mathrm{SP}(u, v))$ by the triangle inequality. Thus $w(C(s, f)) \leq 2(w(\mathrm{SP}(s, u)) + w(\mathrm{SP}(u, v))) \leq 2w(B_i)$ since $w(\mathrm{SP}(s, u)) \leq w(\mathrm{SP}(u, w))$.

Assume next that $f = e$. Then $w(C(s, f)) = w(\mathrm{SP}(s, v)) + c(e) + w(\mathrm{SP}(w, s)) \leq w(\mathrm{SP}(s, u)) + w(\mathrm{SP}(u, v)) + c(e) + w(\mathrm{SP}(s, u)) + w(\mathrm{SP}(u, w)) \leq 2w(\mathrm{SP}(u, v)) + c(e) + 2w(\mathrm{SP}(u, w)) \leq 2w(B_i)$.

We sort our collection in non-decreasing order of weight and do Gaussian elimination on their incidence vectors, restricted to the N edges $e_1, \ldots, e_N$. This determines the least weight N linearly independent cycles in our collection. The time taken for the Gaussian elimination step, which is the most expensive step in our algorithm, is $O(m^\omega \sqrt{n \log n})$ using fast matrix multiplication.

Theorem 5. *A 2-approximate* MCB *in an undirected graph G with non-negative edge weights can be computed in expected time $O(m^\omega \sqrt{n \log n})$.*

4.1 Extension to directed graphs

The above algorithm also holds for directed graphs. A cycle in a directed graph is a cycle in the underlying undirected graph with edges traversable in both directions. A $\{-1, 0, 1\}$ edge incidence vector is associated with each cycle: edges traversed by the cycle in the right direction get 1 and edges traversed in the opposite direction get -1. The cycle space is the space generated by these cycle vectors over $\mathbb{Q}$. Note that the weight of a cycle is simply the sum of the weight of its edges, independent of the orientation of these edges. Let $C = (e_1, \ldots, e_k)$ be a cycle in a directed graph and let $e_i = (u_i, u_{i+1})$. Then we can write $C = \sum_{i=1}^{k} \mathrm{SP}(s, u_i) + e_i + \mathrm{SP}(u_{i+1}, s)$ where $u_{k+1} = u_1$, since $\mathrm{SP}(s, u_i)$ cancels $\mathrm{SP}(u_i, s)$. Note that $\mathrm{SP}(a, b)$ for us here need not be a directed path - it is a shortest path in the underlying undirected graph between a and b. However, the incidence vector of this path in the directed graph would contain -1's corresponding to edges which are traversed in the reverse direction. All the steps in the above construction go through for directed minimum cycle bases too and we have a collection of $O(m\sqrt{n \log n})$ cycles which is a superset of a 2-approximate directed minimum cycle basis.

However, when we do Gaussian elimination, we are no longer over $\mathbb{F}_2$ and so the numbers could grow large. So we can no longer claim that the time taken for

Gaussian elimination is $O(m^\omega\sqrt{n\log n})$. But if we choose a prime p uniformly at random from a collection of small primes and do the arithmetic in Gaussian elimination modulo p, then our cost remains $O(m^\omega\sqrt{n\log n})$ and we will show that with high probability we determine the cycles of a 2-approximate MCB.

Arithmetic modulo p. The problem with doing arithmetic modulo any number p is that the least weight N linearly independent cycles in our collection could turn out to be *linearly dependent* modulo p. That is, the determinant of the $N \times N$ matrix M, defined by incidence vectors of these N cycles, is a multiple of p. In that case, our algorithm is not guaranteed to return a 2-approximate minimum cycle basis.

Now we will use the property that all the entries in the matrix M are $-1, 0, 1$, to show a bounded error when p is a prime chosen uniformly at random from a collection $P = \{p_1, \ldots, p_{N^2}\}$ of N^2 distinct primes, where each $p_i \geq \sqrt{N}$. It follows from Hadamard's inequality that the absolute value of the determinant of M is at most $N^{N/2}$, since each of the N rows is a vector in $\{-1, 0, 1\}^N$. Thus, at most N elements of P can be divisors of $\det(M)$. So the probability that a random element of P divides $\det(M)$ is $\leq N/N^2 = 1/N$. So with probability $1 - 1/N$, arithmetic modulo p yields the least weight N linearly independent cycles from the collection of $O(m\sqrt{n\log n})$ cycles.

The value of $\pi(r)$, the number of primes less than r, is given by $r/6\log r \leq \pi(r) \leq 8r/\log r$ [22]. So all the primes $p_1, \ldots, p_{N^2}$ are $\tilde{O}(N^2)$, and computing them takes $\tilde{O}(N^2)$ time using a sieving algorithm. Arithmetic modulo p ensures that all numbers are $\tilde{O}(N^2)$ and we can assume that arithmetic on $O(\log N)$ bit numbers takes $O(1)$ time. It follows that addition, subtraction and multiplication in $\mathbb{Z}_p$ can be implemented in unit time since p is $\tilde{O}(N^2)$. However, we also need to implement division efficiently. Once p is chosen, we compute the multiplicative inverses of all elements in $\mathbb{Z}_p^*$ by the extended Euclid's gcd algorithm by solving $ax = 1(\mathrm{mod}p)$ for each $a \in \mathbb{Z}_p^*$. This takes time $O(\log p)$ for each element and hence $O(p\log p) = \tilde{O}(N^2)$ for all the elements. Thus, we have shown the following theorem.

Theorem 6. *A 2-approximate minimum cycle basis can be computed with high probability in expected time $O(m^\omega\sqrt{n\log n})$ in an directed graph G with n vertices, m edges and non-negative edge weights.*

5 Conclusions

In this paper we design faster algorithms for computing approximate minimum cycle basis of undirected graphs. To the best of our knowledge it is the first time that sparse cycle bases with a guarantee are computed in $o(m^\omega)$ time. Our techniques extend also to the directed version of the minimum cycle basis problem in which the base field is $\mathbb{Q}$ instead of $\mathbb{F}_2$. We present very fast approximate algorithms for this version as well.

References

1. Stepanec, G.F.: Basis systems of vector cycles with extremal properties in graphs. Uspekhi Mat. Nauk **19** (1964) 171–175
2. Horton, J.D.: A polynomial-time algorithm to find a shortest cycle basis of a graph. SIAM Journal of Computing **16** (1987) 359–366
3. de Pina, J.: Applications of Shortest Path Methods. PhD thesis, University of Amsterdam, Netherlands (1995)
4. Golynski, A., Horton, J.D.: A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In: SWAT. (2002)
5. Berger, F., Gritzmann, P., de Vries, S.: Minimum cycle basis for network graphs. Algorithmica **40**(1) (2004) 51–62
6. Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.E.: A faster algorithm for minimum cycle basis of graphs. In: 31st International Colloquium on Automata, Languages and Programming, Finland. (2004) 846–857
7. Huber, M.: Implementation of algorithms for sparse cycle bases of graphs. Technical report, Technische Universität München (2002) http://www-m9.ma.tum.de/dm/cycles/mhuber.
8. Kreisbasenbibliothek CyBaL. http://www-m9.ma.tum.de/dm/cycles/cybal (2004)
9. Mehlhorn, K., Michail, D.: Implementing minimum cycle basis algorithms. In: WEA. Volume 3503 of LNCS. (2005) 32–43
10. Swamy, M.N.S., Thulasiraman, K.: Graphs, Networks, and Algorithms. John Wiley & Sons, New York (1981)
11. Cassell, A.C., Henderson, J.C., Ramachandran, K.: Cycle bases of minimal measure for the structural analysis of skeletal structures by the flexibility method. In: Proc. Royal Society of London Series A. Volume 350. (1976) 61–70
12. Gleiss, P.M.: Short Cycles, Minimum Cycle Bases of Graphs from Chemistry and Biochemistry. PhD thesis, Fakultät Für Naturwissenschaften und Mathematik der Universität Wien (2001)
13. Tewari, G., Gotsman, C., Gortler, S.J.: Meshing genus-1 point clouds using discrete one-forms. Computers and Graphics (2006) to appear.
14. Coppersmith, D., Winograd, S.: Matrix multiplications via arithmetic progressions. Journal of Symb. Comput. **9** (1990) 251–280
15. Kavitha, T., Mehlhorn, K.: A polynomial time algorithm for minimum cycle basis in directed graphs. In: STACS 2005. Volume 3404 of LNCS. (2005) 654–665
16. Liebchen, C., Rizzi, R.: A greedy approach to compute a minimum cycle basis of a directed graph. Inf. Process. Lett. **94**(3) (2005) 107–112
17. Hariharan, R., Kavitha, T., Mehlhorn, K.: A faster deterministic algorithm for minimum cycle basis in directed graphs. In: Proceedings of ICALP. Volume 4051 of LNCS. (2006) 250–261
18. Kavitha, T.: An $\tilde{O}(m^2n)$ randomized algorithm to compute a minimum cycle basis of a directed graph. In: Proceedings of ICALP, LNCS 3580. (2005) 273–284
19. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete Comput. Geom. **9**(1) (1993) 81–100
20. Thorup, M., Zwick, U.: Approximate distance oracles. In: ACM Symposium on Theory of Computing. (2001) 183–192
21. Thorup, M., Zwick, U.: Compact routing schemes. In: Proceedings of 13th ACM Symposium on Parallel Algorithms and Architecture. (2001) 1–10
22. Apostol, T.M.: Introduction to Analytic Number Theory. Springer-Verlag (1997)