Algorithmica (2004) 40: 51–62
DOI: 10.1007/s00453-004-1098-x

Algorithmica

Minimum Cycle Bases for Network Graphs[1]

Franziska Berger,[2] Peter Gritzmann,[2] and Sven de Vries[2]

Abstract. The minimum cycle basis problem in a graph $G = (V, E)$ is the task to construct a minimum length basis of its cycle vector space. A well-known algorithm by Horton of 1987 needs running time $O(|V||E|^{2.376})$. We present a new combinatorial approach which generates minimum cycle bases in time $O(\max\{|E|^3, |E||V|^2 \log|V|\})$ with a space requirement of $\Theta(|E|^2)$. This method is especially suitable for large sparse graphs of electric engineering applications since there, typically, $|E|$ is close to linear in $|V|$.

Key Words. Graph cycle, Minimum cycle basis, Matroid, Electrical network.

1. Introduction. Cycle bases in graphs have been an object of study for more than 150 years, since they have proved a useful tool in Electric Network Theory and various other applications. Ever since Kirchhoff, cycle bases obtained from a spanning tree in the network graph serve to check or enforce the Kirchhoff Voltage Law [Ki], [Re], [Mu], [IM]. Other fields include Static Analysis, dealing with the rigidity of framework structures [Re], [Ka], [Sp], chemical ring perception [DGHL], and periodic event scheduling [LM].

A cycle basis obtained from a spanning tree of the graph is generally not of minimum length. To cope with the computational exigencies posed by the large data sets of practical electric networks, minimum cycle bases are of advantage, because the subsequent computation time reduces with decreasing basis length.

Horton's algorithm for finding a minimum cycle basis in an edge-weighted undirected graph can be implemented in time $O(|V||E|^\alpha)$ where α denotes the matrix multiplication constant (currently, $\alpha \sim 2.376$), see [Ho] and [GH]. With our present approach, a minimum cycle basis can be determined in time $O(\max\{|E|^3, |E||V|^2 \log|V|\})$ and space $\Theta(|E|^2)$. Thus, the new method has a lower complexity bound whenever $|E|^{3-\alpha} \leq |V|$, hence currently for $|E| = O(|V|^{1.61})$, which is the case in the aforementioned applications, since the relevant graphs from practical electric networks are very large and particularly sparse.

An algorithm based on the same principle as our approach has been developed independently by Gerards et al. see [GdPS]. A similar algorithm has also been mentioned in [CGH], the latter however with exponential running time.

The paper is organized as follows. After introducing the necessary notation in Section 1.1, we describe a useful preprocessing step for multigraphs in Section 1.2. The

[1] The authors were supported by the DFG Schwerpunktprogramm Nr. 1126, Grant No. GR993/8-1.
[2] Zentrum Mathematik, Technische Universität München, Boltzmannstraße 3, D-85747 Garching bei München, Germany. {berger,gritzman,devries}@ma.tum.de.

Received April 17, 2003; revised February 27, 2004. Communicated by S. Albers.
Online publication May 28, 2004.

algorithm for finding a minimum cycle basis is presented in Section 2. A first indication of its practical behavior is given in Section 2.3.

1.1. *Definitions and Notation.* Let $G = (V, E)$ be an undirected multigraph with m edges and n vertices. Let $E = \{e_1, \ldots, e_m\}$, and let ω: $E \to \mathbb{R}^+$ be a positive weight function on E.

A *cycle* C is a subgraph of G in which all vertices have even degree. It is *simple* if it is connected and if each vertex in C has degree two. A cycle C may be represented by its incidence vector $(b_i(C))_{i=1,\ldots,m}$, with

$$b_i(C) = \begin{cases} 0, & \text{if} \quad e_i \notin E(C), \\ 1, & \text{if} \quad e_i \in E(C), \end{cases}$$

where $E(C)$ denotes the set of edges in C. Usually, we identify a cycle C with its incidence vector. The weight of a cycle C with respect to ω is defined as $\omega(C) := \sum_{e \in C} \omega(e)$. If $\omega(e) = 1$ for all $e \in E$, the weight of a cycle coincides with its *length* $|C|$ defined by $|E(C)|$.

The incidence vectors of cycles span a binary vector space $\mathcal{C}_{GF(2)}(G)$, the *cycle space* of the graph when the addition of two cycles C_1 and C_2 is defined as the binary addition of their incidence vectors. It corresponds to the symmetric difference

$$C_1 \oplus C_2 = (E(C_1) \cup E(C_2)) \backslash E(C_1 \cap C_2).$$

A set of cycles is called *linearly independent* if their incidence vectors are linearly independent over $GF(2)$. The following result is well known (see, e.g., [Di]).

PROPOSITION 1. *The dimension* $\mu(G)$ *of* $\mathcal{C}_{GF(2)}(G)$ *is equal to* $m - n +$ c(G), *where* c(G) *is the number of connected components of* G.

$\mu := \mu(G)$ is also called the *cyclomatic number* of G. A *cycle basis* $\mathcal{B}$ is a basis of $\mathcal{C}_{GF(2)}(G)$. It is called a *minimum cycle basis* if its weight $\omega(\mathcal{B}) := \sum_{C \in \mathcal{B}} \omega(C)$ is minimal. The cycle-edge incidence matrix $A \in GF(2)^{\mu \times m}$ of a cycle basis $\mathcal{B}$ has as rows the (incidence vectors of the) cycles in $\mathcal{B}$.

Adapting an argument of [LS] for graphs with positive edge weights, it is easy to see that the following holds.

REMARK 2. Each cycle in a minimum cycle basis is simple.

Some cycle bases may be constructed from a spanning tree or—if G is not connected—from a spanning forest T of G (*fundamental tree bases*, Kirchhoff bases) by adding edges to T. Any nontree edge e forms a unique *fundamental cycle* $F(e)$ with edges of T. The set of such cycles has cardinality μ and their linear independence is obvious since every nontree edge is contained in exactly one cycle.

A fundamental tree basis $\mathcal{B}_T$ can be computed in time $O(mn)$, more specifically in time $O(\sum_{i=1}^{\mu} |F_i|)$, where F_i, $i = 1, \ldots, \mu$, are the fundamental cycles [Pa].

In Section 2.2 we use the fact that the columns of the incidence matrix A of a fun-

damental tree basis $\mathcal{B}_T$ may be ordered such that $A = (I_\mu | A')$, where I_μ is the $\mu \times \mu$ identity matrix.

Horton's polynomial time algorithm for finding a *minimum cycle basis* of a graph G [Ho], [GH] is based on the fact that the cycle space of G has the structure of a matroid with sets of linearly independent cycles as independent sets. It follows that, if all cycles were explicitly given, the Greedy algorithm could be used to find a minimum cycle basis [Ox]. However, the number of simple cycles in G may be exponential in n, so it is necessary to reduce the number of cycles to be checked. Horton's algorithm generates a polynomially sized set of candidate cycles which is guaranteed to contain a minimum cycle basis. The minimum cycle basis is then extracted from this set by means of the Greedy algorithm. Linear independence of the cycles is checked by Gaussian elimination on the corresponding cycle-edge incidence matrix.

The cycles in Horton's candidate set are of the form $C(x, y, z) = P(z, x) + (x, y) + P(y, z)$ for every triple $\{x, y, z\}$ with $(x, y) \in E$ and $z \in V$, where $P(z, x)$ denotes an *arbitrary* path from z to x that is shortest with respect to the weight ω.

REMARK 3 [Ho]. The number of candidate cycles is $O(mn)$.

In their recent article, Golynski and Horton show that the running time complexity of the algorithm is $O(|E|^\alpha |V|)$ where α denotes the matrix multiplication constant [GH].

In planar graphs a minimum cycle basis can be found in time $O(n^2 \log n + m)$, see [HM]. Faster algorithms exist also for special graph classes like Halin-graphs, outerplanar graphs, and series-parallel graphs, see [LS] and [St].

1.2. *Preprocessing*. Clearly, the cycle space of a graph is the direct product of the cycle spaces of its connected components. Similarly, the cycle space of a connected but not 2-connected graph G is the direct product of the cycle spaces of its 2-connected components. Therefore, a minimum cycle basis of G can be found by concatenating minimum cycle bases of the 2-connected components. The 2-connected components can be computed in linear time, see, e.g., [Ta]; consequently we henceforward consider only 2-connected graphs.

Multigraphs from electric engineering applications often have many parallel edges and loops. However, as we will see, these can be removed in a preprocessing step. In practice, this often leads to a significant reduction of the problem size.

The following statement shows that there are classes of cycles, a representative of which must be contained in every minimum cycle basis.

PROPOSITION 4 [HS]. *Let $e \in E$ be an edge of a graph G through which there exists a shortest cycle $C(e)$. Then there is a minimum cycle basis $\mathcal{B}$ containing $C(e)$. Moreover, every minimum cycle basis must contain some shortest cycle through e.*

In general, the set $\{C(e) : C(e)$ is a shortest cycle through $e \in E\}$ does not span $\mathcal{C}_{GF(2)}(G)$. For instance, in Figure 1 every edge e belongs to one of the eight triangles, but according to Proposition 1, $\dim \mathcal{C}_{GF(2)}(G) = 16 - 8 + 1 = 9$.

By Proposition 4, cycles formed by loops in G belong to every minimum cycle basis. Remark 2 implies that it is possible to remove the loops from G, compute a minimum

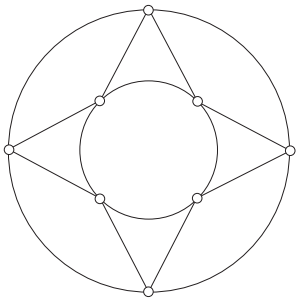


Fig. 1. Example showing that Proposition 4 generally does not provide a cycle basis. We assume here $\omega(e) = 1$ for all $e \in E$.

cycle basis of the remaining graph, and add to it the cycles corresponding to loops. The basis obtained is a minimum cycle basis of G.

Let u, v be two vertices that are joined by k parallel edges with $k \geq 2$. Select an edge e' with minimum weight among them. For each edge e of the other $k-1$ parallel edges compute successively a shortest cycle through e and then remove e from G. The graph that remains after doing this for all pairs of adjacent vertices is simple. We claim that the set of cycles $\mathcal{B}$ obtained by computing a minimum cycle basis $\mathcal{B}_0$ of the reduced, simple graph together with the set of cycles $\mathcal{B}_1$ resulting from the parallel edges and loops described above form a minimum cycle basis of G.

LEMMA 5. *The resulting set of cycles $\mathcal{B} := \mathcal{B}_0 \cup \mathcal{B}_1$ is a minimum cycle basis of G.*

PROOF. $\mathcal{B}$ clearly consists of μ cycles. The cycles in $\mathcal{B}_1$ are linearly independent. In fact, their cycle-edge incidence matrix A, where the edges are numbered according to the order in which the cycles are constructed, has upper triangular form. Further, all cycles in $\mathcal{B}_1$ are also linearly independent from all cycles in $\mathcal{B}_0$.

It remains to show that $\mathcal{B}$ is a minimum cycle basis of G. By choice of the edge e', $\mathcal{B}_0$ is part of a minimum cycle basis of G; no cycle could be replaced by a shorter cycle. The cycles of $\mathcal{B}_1$ are shortest possible cycles in G through the corresponding edges, again by choice of e', and thus belong to a minimum cycle basis by Proposition 4. □

2. A Fast Minimum Cycle Basis Algorithm

2.1. *The Basic Scheme.* In this section we describe our main algorithm for computing a minimum cycle basis of an undirected edge-weighted graph G. By the results of Section 1.2, we assume that G is simple and 2-connected.

It is based on the observation that a cycle basis is minimal when no cycle in it can be replaced by a strictly smaller cycle. Thus, the algorithm begins with a fundamental tree basis and successively exchanges cycles for smaller ones if possible. This is done by constructing specific shortest paths in an auxiliary graph. The basic scheme is given in Algorithm 1.

Algorithm 1. Basic exchange scheme

Input: Undirected 2-connected simple edge-weighted graph G
Output: Minimum cycle basis $\mathcal{B}$ of G

1: Construct a fundamental tree basis $\mathcal{B}_0 = \{F_1, \ldots, F_\mu\}$.
2: **for** $i = 1$ to μ **do**
3: Find a shortest cycle C_i that is linearly independent of $\mathcal{B}_{i-1} \backslash \{F_i\}$ with Subroutine 2.
4: **if** $\omega(C_i) < \omega(F_i)$ **then**
5: $\mathcal{B}_i := (\mathcal{B}_{i-1} \backslash \{F_i\}) \cup \{C_i\}$
6: **end if**
7: **end for**
8: Output $\mathcal{B} := \mathcal{B}_\mu$

Of course, the crucial part is the selection of the new cycle C_i in Subroutine 2, which is given in detail in Section 2.3. In any case, it follows readily from the matroid property that Algorithm 1 produces a minimum cycle basis. Finding a fundamental tree basis takes time $O(mn)$. As we will show in Lemma 13, each of the $O(m)$ steps of the algorithm can be carried out in $O(\max\{m^2, n^2 \log n\})$ time. Hence, we can conclude

THEOREM 6. *Algorithm* 1 *computes a minimum cycle basis with* $O(\max\{m^3, mn^2 \log n\})$ *time*.

PROOF. We show that in each step i the set of the first i cycles can be extended to a minimum cycle basis. For a contradiction, let i_0 be the smallest index such that the first i_0 new cycles in the basis $\mathcal{B}_{i_0}$ cannot be supplemented by other cycles in G such that a minimum cycle basis is obtained. Let $\mathcal{B}$ be a minimum cycle basis which contains the first $i_0 - 1$ cycles of $\mathcal{B}_{i_0}$. As the cycle C_{i_0} cannot be a linear combination of the first $i_0 - 1$ cycles, its basis representation by cycles from $\mathcal{B}$ requires at least one other cycle. If one of these cycles has weight equal to or larger than C_{i_0}, we could replace it by C_{i_0} in $\mathcal{B}$. Hence all are strictly smaller. However, then at least one of these cycles is linearly independent of $\mathcal{B}_{i_0-1} \backslash \{F_i\}$, for otherwise, replacing each such cycle by its basis representation with respect to $\mathcal{B}_{i_0}$ yields a representation of C_{i_0} with cycles of $\mathcal{B}_{i_0}$, a contradiction. Hence, in step i_0, we could have found a strictly shorter cycle, contradicting the choice of C_{i_0}. □

Note that our algorithm differs from Horton's algorithm fundamentally in that no explicit global set of candidate cycles is required. Moreover, a cycle basis is maintained in every step of the algorithm. Thus it can also be used as a heuristic method to obtain a sparse cycle basis, by starting with the longest cycles in $\mathcal{B}_0$ and stopping after some time interval has elapsed.

This trade-off property is particularly important in dealing with very large scale electric networks.

2.2. *A Criterion for Linear Independence*. Now, we describe a criterion for constructing a cycle that is linearly independent of a set of $\mu - 1$ basis cycles efficiently. Such a cycle is called *feasible*. We show in the following three subsections that a shortest

feasible cycle needed in Line 3 of Algorithm 1 can be computed by combinatorial means in time $O(\max\{m^2, n^2 \log n\})$.

Note that testing linear independence by Gaussian elimination requires that all cycles to be checked are explicitly given; but this is not the case here.

Our approach utilizes the fact that a cycle C_i is linearly independent of the row space of the cycle-edge incidence matrix A of a set of cycles if and only if there exists a vector $u \in \ker A$ such that $\langle C_i, u\rangle = 1$, where $\langle \cdot, \cdot \rangle$ denotes the standard inner product over $GF(2)$. As this statement plays a crucial role in the following, we include it as an explicit lemma.

LEMMA 7. *Let $\mathcal{C}$ be a set of cycles, let A be its cycle-edge incidence matrix, and let $C_i \notin \mathcal{C}$ be a cycle. C_i is linearly independent of the row space of A if and only if there exists a vector $u \in \ker A$ such that $\langle C_i, u\rangle = 1$.*

PROOF. ($\Rightarrow$) If no vector $u \in \ker A$ with the claimed property exists, we have $\ker \binom{A}{C_i} = \ker A$. However, by linear independence, $\operatorname{rank}\binom{A}{C_i} = \operatorname{rank}(A) + 1$, a contradiction.

($\Leftarrow$) If $C_i = \bigoplus_{j\in J} a_j$ for some index set J, then $\langle C_i, u\rangle = \langle \bigoplus_{j\in J} a_j, u\rangle = \bigoplus_{j\in J} \langle a_j, u\rangle = 0$ for all $u \in \ker A$, again a contradiction. □

In Subroutine 2 we first compute the kernel of the incidence matrix of the cycles of $\mathcal{B}_{i-1}\backslash\{F_i\}$ for $i \in \{1, \ldots, \mu\}$ by transforming the matrix into upper triangular form by Gaussian elimination. Since the bases $\mathcal{B}_i$ and $\mathcal{B}_{i-1}$ for $i = 1, \ldots, \mu$ differ in at most a single cycle, the corresponding new matrices can be calculated from the previous ones very efficiently.

In the following, A_0 denotes the cycle-edge incidence matrix of $\mathcal{B}_0$. Since $\mathcal{B}_0$ is a fundamental tree basis we may assume that A_0 is of the form $(I_\mu | A')$, where I_μ denotes the $\mu \times \mu$ unit matrix. In general, A_i denotes the upper triangular matrix obtained after completion of step i and corresponding elimination. For $i = 1, \ldots, \mu$, the matrix resulting from A_{i-1} by the removal of row i is denoted by $A_{i-1}^{(i)}$. For instance, $A_0^{(1)}$ is the cycle-edge incidence matrix of $\mathcal{B}_0\backslash\{F_1\}$.

If in step i of Algorithm 1, F_i is replaced by a new cycle C_i, the matrix A_i is obtained from $A_{i-1}^{(i)}$ by adding the incidence vector of C_i as row i and subsequent Gaussian elimination on this row.

By construction, each matrix $A_{i-1}^{(i)}$ has upper triangular form. Therefore, only after insertion of a new cycle C_i into $A_{i-1}^{(i)}$, row i might have nonzero entries below the diagonal. Hence it suffices to perform Gaussian elimination steps only on row i; rows with index $j \neq i$ remain unchanged.

LEMMA 8. *The matrices A_i, $i = 1, \ldots, \mu$, are the cycle-edge incidence matrices of the sets $\mathcal{B}_i$ after Gaussian elimination.*

It is clear that the row operations do not change the kernel of the corresponding matrix. Further, since the dimension μ of the cycle space coincides with the rank of A_{i-1}, it follows that $\langle F_i, u\rangle = \langle C_i, u\rangle$ for each $u \in \ker A_{i-1}^{(i)}$, hence $\ker A_{i-1} = \ker A_i$.

LEMMA 9. *For each $i = 1, \ldots, \mu$, $\ker A_0 = \ker A_i \subset \ker A_{i-1}^{(i)}$.*

The dimension of $\ker A_0$ is $m - \mu = n - 1$. Since $A_0 = (I_\mu | A')$, a basis for $\ker A_0$ is given by the columns of $\binom{A'}{I_{m-\mu}} =: U$. We index the kernel vectors in U as $u_{\mu+1}, \ldots, u_m$ to point out that, in this basis, every kernel vector corresponds to a free variable of A_0 with entry 1 in the row corresponding to its index.

The kernel of each matrix $A_{i-1}^{(i)}$, $i = 1, \ldots, \mu$, is spanned by the vectors u_j, $j = \mu + 1, \ldots, m$, and an additional basis vector u_i. u_i can be chosen such that $u_{ij} = 0$ for $j > i$ and $u_{ii} = 1$. Its entries for $j < i$ are then determined by solving the upper triangular $(i-1) \times i$ part of the system $A_{i-1}^{(i)} u_i = 0$ which requires at most $O(m^2)$ time. Note that u_1 is the binary vector with one as the first entry and $m - 1$ zeros.

By Lemmas 7 and 9, every cycle C_i that is linearly independent of the row space of $A_{i-1}^{(i)}$ must fulfill $\langle C_i, u_i \rangle = 1$. Therefore it is sufficient to examine only the additional kernel basis vector u_i constructed for $A_{i-1}^{(i)}$ in order to find all feasible cycles.

Lemma 8 ensures that a shortest cycle C_i with $\langle C_i, u_i \rangle = 1$ will be linearly independent not only from the rows of $A_{i-1}^{(i)}$ but also from $\mathcal{B}_{i-1} \backslash \{F_i\}$ in each step $i = 1, \ldots, \mu$. To construct a shortest linearly independent cycle note that any cycle with odd parity with respect to u_i will be linearly independent of $\mathcal{B}_{i-1} \backslash \{F_i\}$ and we simply select a shortest one with respect to ω. Details are given in the next subsection.

2.3. *Constructing a Shortest Linearly Independent Cycle.* Consider a fixed step $i \in \{1, \ldots, \mu\}$ of Algorithm 1. To find a shortest feasible cycle C with $\langle C, u \rangle = 1$ for the kernel vector $u := u_i$ constructed above we consider an auxiliary graph G_u in which this can be achieved by solving a shortest path problem.

Subroutine 2. Constructing a shortest feasible cycle C

Input: Index $i \geq 1$, Matrix A_{i-1}
Output: Shortest feasible cycle C

Form $A_{i-1}^{(i)}$ by removing row i from A_{i-1}.
Construct kernel vector u by setting $u_i = 1$, $u_j = 0$ for $j > i$ and solving $A_{i-1}^{(i)} u = 0$.
Form graph G_u as described in Section 2.3.
Let $\mathcal{C}$ be an empty set of cycles.
for all vertices v incident to an edge e in G with $u_e = 1$ **do**
    Find a shortest v–v' path P_v in G_u with respect to ω_u.
    if P_v does not contain a vertex pair $\{x, x'\}$ with $x \neq v$ **then**
        $\mathcal{C} := \mathcal{C} \cup W(P_v)$
    end if
end for
if $\min_{D \in \mathcal{C}} \omega(D) < \omega(F_i)$ **then**
    Let $C := \operatorname{argmin}_{D \in \mathcal{C}} \omega(D)$
    Form A_i by inserting C into row i of $A_{i-1}^{(i)}$ and performing Gaussian elimination on row i.
else
    $C := F_i$, $A_i := A_{i-1}$
end if
return C.

The graph G_u is constructed from G as follows (for an example see Figure 2): For every vertex v of G make two vertices v and v' in G_u. For each edge $e = (x, y)$ in G add two edges to G_u: If entry u_e is one, add edge (x, y') and edge (x', y), if it is zero, add edges (x, y) and (x', y'). This way, G_u displays graph G on two levels. Define a weight function ω_u: $G_u \to \mathbb{R}^+$ by assigning each edge the weight $\omega(e)$ of the corresponding edge e in G it comes from.

G_u has $2n$ vertices and $2m$ edges; it can be constructed in time $O(m)$ and needs space $\Theta(m)$.

LEMMA 10. *Let C be a simple cycle in G. Then the following two statements are equivalent*: (a) $\langle C, u\rangle = 1$. (b) *For every vertex $v \in C$ there exist two simple v–v' paths in G_u which do not contain any pair of vertices $\{x, x'\}$ with $x \neq v$.*

PROOF. ($\Leftarrow$) Trivial. ($\Rightarrow$) Let $C = \{e_1, \ldots, e_r\}$ be a simple cycle in G that fulfills $\langle C, u\rangle = 1$. Let $e_1 = (v, v_1), \ldots, e_k = (v_{k-1}, v_k), \ldots, e_r = (v_{r-1}, v)$. We construct a path in G_u according to the assertion. For e_1 choose edge (v, v_1') in G_u if $u_{e_1} = 1$, (v, v_1) otherwise. For every subsequent edge e_k in C with $u_{e_k} = 0$ choose edge (v_{k-1}, v_k) or (v_{k-1}', v_k'), depending on the end vertex of edge e_{k-1}. If $u_{e_k} = 1$, take edge (v_{k-1}, v_k') or (v_{k-1}', v_k), again depending of the end vertex reached in the previous step. Since $\langle C, u\rangle = 1$, this procedure must end with v' in the last step. C is simple, therefore, for every vertex $x \in C\backslash\{v\}$, either x or x' appears in G_u, but not both.

For every simple cycle C with $v \in C$, there are two v–v' paths of the same weight which represent C, the v–v' path P_1 described above, and a v'–v path P_2 which uses x' wherever P_1 uses x for all $x \in C$. □

While, according to Lemma 10, simple odd parity cycles in G lead to simple paths in G_u, without the condition on the pairs $\{x, x'\}$ not every simple v–v' path P in G_u corresponds to a simple cycle in G. It need not even correspond to a cycle at all (see Figure 2). However, then it contains a subpath Q corresponding to a strictly smaller simple cycle in G.

LEMMA 11. *For a vertex v of G and a simple v–v' path P_v in G_u let $W(P_v)$ denote the closed walk in G obtained by replacing each vertex x' of P_v by x. Then $W(P_v)$ contains a simple odd parity cycle C in G.*

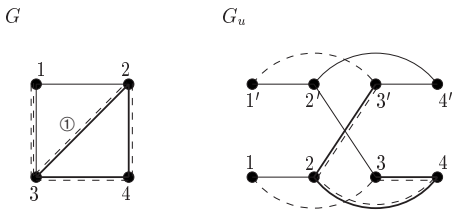


Fig. 2. The dashed 1–1′ path in G_u does not correspond to a cycle in G. However, it contains the bold 3–3′ subpath corresponding to cycle $\{2, 3, 4\}$. Here, u contains only one nonzero entry, namely for edge $(2, 3)$, displayed as ① in G.

PROOF. Let P_v be a simple v–v' path in G_u. If $W(P_v)$ is no simple cycle, there must be a vertex $x \in W(P)$ different from v which is encountered at least twice. This means that, for this vertex, both copies x and x' must be contained in P_v, for otherwise P_v would not be simple. Consider the strictly shorter x–x' subpath P_x. If P_x does not correspond to a simple cycle either, the argument may be repeated and eventually, since the edge weights ω_u are positive, a shortest subpath Q is found that corresponds to a simple cycle $C \in G$. □

Lemma 10 enables us to find a shortest odd parity cycle containing v if it exists, by constructing a shortest v–v' path in G_u and checking whether it contains a pair of vertices $\{x, x'\}$ with $x \neq v$. This is done in Subroutine 2 for each vertex v incident to an edge e whose corresponding component in the kernel vector u is 1. By Lemma 11, a simple path that contains such a pair can still be used since it contains a subpath corresponding to a shortest odd parity cycle for a different pair of vertices. The existence of at least one odd parity cycle is guaranteed, since the removed fundamental cycle F_i has odd parity with respect to u by construction. We obtain up to n paths which all correspond to simple feasible cycles. Among these, we select a shortest one with respect to ω_u and discard the rest. Hence, we conclude with the following lemma.

LEMMA 12. *The cycle returned by Subroutine* 2 *in each step* $i \in \{1, \ldots, \mu\}$ *is a shortest cycle with respect to* ω *that is linearly independent of all cycles in* $\mathcal{B}_{i-1} \setminus \{F_i\}$.

2.4. *Time Bounds*. We analyze the time complexity of Algorithm 1.

LEMMA 13. *Subroutine* 2 *runs in time* $O(\max\{m^2, n^2 \log n\})$.

PROOF. The kernel vector u may be constructed in time $O(m^2)$ by solving in step $i \in \{1, \ldots, \mu\}$ a linear system $A_{i-1}^{(i)} u = 0$, with $A_{i-1}^{(i)}$ an upper triangular matrix. Construction of the graph G_u takes linear time, and the solution of at most n shortest path problems time $O(n(m + n \log n))$. Further, the operations for updating matrix A_i require linear time and Gaussian elimination has to be performed on row i, if a new cycle was inserted, so only the first $(i - 1)$ entries of row i might have to be annihilated; note that no other elements are affected; rows of index j with $j > i$ still contain the original cycles from the fundamental tree basis $\mathcal{B}_0$. Rows of index j with $j < i$ have already been examined. As cycles have at most n nonzero entries, this adds another $O(mn)$ term. □

LEMMA 14. *The space requirement of Algorithm* 1 *is* $\Theta(m^2)$.

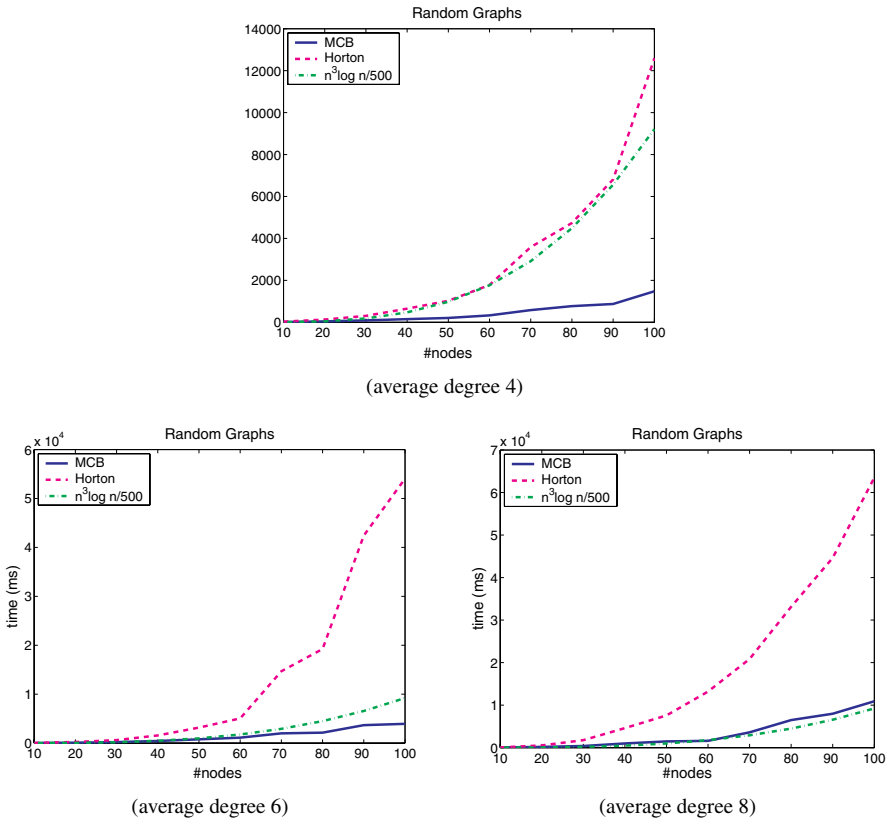
PROOF. The matrices A_i for $i = 1, \ldots, \mu$ can have at most $\Theta(m^2)$ entries. n additional paths have to be stored temporarily in every step i for $i = 1, \ldots, \mu$, with a space requirement of $\Theta(n^2)$. The graph G_u needs linear space. The bases $\mathcal{B}_i$, $i = 0, \ldots, \mu$, require space $\Theta(mn)$. □

For practical instances from electric engineering with up to 2000 vertices, Algorithm 1 took less than an hour on an SGI Octane, with 2 × 250-MHz MIPS R10000 CPUs

(Data Cache: 32 kB; Instruction Cache: 32 kB; L2-Cache: 1 MB) and 640 MB RAM. A forthcoming contribution [BGdV2] gives fast heuristic approaches appropriate for sparse graphs with $O(10^5)$ vertices. Applications in Computational Chemistry can be found in [BGdV1].

To compare the practical performance of Algorithm 1 with Horton's original algorithm from [Ho], we conducted experiments on sparse real-world graphs. It turned out that Horton's algorithm took a prohibitively long time for such small graphs from the applications. For larger randomly generated graphs with small average degree, Algorithm 1 outperforms Horton's by orders of magnitude.

Figure 3 depicts the average running time of Algorithm 1 for graphs with average degree close to 4, 6, and 8, taken over a sample of 20 graphs with 10–100 vertices. The graphs were created by randomly choosing edges between all pairs of vertices, and then adding further edges to make the graph 2-connected if this was not already the case.



(average degree 4)

(average degree 6)

(average degree 8)

Fig. 3. The average running time of a standard implementation of Horton's algorithm and Algorithm 1 (MCB). For comparison, we include the curve $n^3 \log n$ (scaled by some constant factor). This is the theoretical worst-case complexity of Algorithm 1 for our sample graphs.

2.5. *Minimum Cycle Bases of Regular Matroids*. Golynski and Horton also show that a minimum cycle basis for weighted regular matroids can be computed in time $O(k^{\alpha+1})$ where k denotes the cardinality of the ground set and α is the matrix multiplication constant, see [GH].

Regular matroids are matroids M which are representable over every field, i.e., for every field $\mathcal{F}$ there is a mapping from the ground set of M into a finite-dimensional vector space W over $\mathcal{F}$ such that a set of elements of M is independent if and only if its image is linearly independent in W, see [Ox] for details.

Golynski and Horton's [GH] algorithm uses the fact that regular matroids can be decomposed into graphic matroids, cographic matroids, and copies of the special matroid R_{10} [Se], [Tr]. It cleverly combines minimum cycle bases of these components to obtain a minimum cycle basis of the matroid itself. Algorithm 1 can be used alternatively as a subroutine for graphic matroids instead of Horton's algorithm, to some advantage if the graphs corresponding to the graphic matroids are sparse.

Note Added in Proof. Recently, T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch used fast all pairs shortest path algorithms to derive an $O(m^2n + mn^2 \log n)$ algorithm for the minimum cycle basis problem.

References

[BGdV1] F. Berger, P. Gritzmann, and S. de Vries. Computing cyclic invariants for molecular graphs. Manuscript, 2004.

[BGdV2] F. Berger, P. Gritzmann, and S. de Vries. New heuristics for the sparse cycle basis problem. Manuscript, 2004.

[CGH] D.M. Chickering, D. Geiger, and D. Heckerman. On finding a cycle basis with a shortest maximal cycle. *Inform. Process. Lett.*, 54:55–58, 1995.

[DGHL] G.M. Downs, V.J. Gillet, J.D. Holliday, and M.F. Lynch. Review of ring perception algorithms for chemical graphs. *J. Chem. Inform Comput. Sci.*, 29:172–187, 1989.

[Di] R. Diestel. *Graph Theory*, volume 173 of Graduate Texts in Mathematics. Springer-Verlag, Berlin, 1997. 2nd Edn. 2000.

[GdPS] A.M.H. Gerards, J.C. de Pina, and A. Schrijver. Shortest circuit bases of graphs. Manuscript, 2002.

[GH] A. Golynski and J.D. Horton. A polynomial time algorithm to find the minimal cycle basis of a regular matroid. In *Algorithm Theory - SWAT* 2002, *Turku*, *Finland*, volume 2368 of Lecture Notes in Computer Science, pages 551–570. Springer-Verlag, Berlin, 2002.

[HM] D. Hartvigsen and R. Mardon. The all-pairs min-cut problem and the min cycle basis problem on planar graphs. *SIAM J. Discrete Math.*, 7:403–418, 1994.

[Ho] J.D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM J. Comput.*, 16:358–366, April 1987.

[HS] E. Hubicka and M.M. Syslo. Minimal bases of cycles of a graph. In M. Fiedler, editor, *Recent Advances in Graph Theory*, *Proc.* 2*nd Czechoslovak Conference on Graph Theory*, pages 283–293. Academia, Prague, 1975.

[IM] S. Iwata and K. Murota. Combinatorial relaxation algorithm for mixed polynomial matrices. *Math. Programming Ser. A*, 90:353–371, 2001.

[Ka] A. Kaveh. On minimal and optimal cycle bases of graphs for sparse flexibility matrices. *Z. Angew. Math. Mech.*, 69:T212–T214, 1989.

[Ki] G. Kirchhoff. Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Ann. Physik Chem.*, 72(12):497–508, 1847.

[LM] C. Liebchen and R. Möhring. A case study in periodic timetabling. In *Proceedings of ATMOS 2002*, volume 66, No. 6 of Electronic Notes in Theoretical Computer Science (ENTS), 2002.

[LS] J. Leydold and P. F. Stadler. Minimal cycle basis of outerplanar graphs. *Electron. J. Combin.*, 5:R16, 1998.

[Mu] K. Murota. *Matrices and Matroids for Systems Analysis*, volume 20 of Algorithms and Combinatorics. Springer-Verlag, Berlin, 2000.

[Ox] J.G. Oxley. *Matroid Theory*. Oxford Gradutae Texts in Mathematics. Oxford University Press, Oxford, 1992.

[Pa] K. Paton. An algorithm for finding a fundamental set of cycles of a graph. *Comm. ACM*, 12(9):514–519, 1969.

[Re] A. Recski. *Matroid Theory and its Applications*, volume 6 of Algorithms and Combinatorics. Springer-Verlag, Berlin, 1989.

[Se] P. Seymour. Decomposition of regular matroids. *J. Combin. Theory Ser. B*, 28:305–359, 1980.

[Sp] K.V. Spiliopoulos. On the automation of the force method in the optimal plastic design of frames. *Comput. Methods Appl. Mech. Engrg.*, 141:141–156, 1997.

[St] P.F. Stadler. Minimum cycle bases of Halin graphs. *J. Graph Theory*, 43:150–155, 2003.

[Ta] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160, 1972.

[Tr] K. Truemper. *Matroid Decomposition*. Academic Press, Boston, MA, 1992.