Minimum Cycle Bases and Their Applications

Franziska Berger[1], Peter Gritzmann[2], and Sven de Vries[3]

[1] Department of Mathematics, Polytechnic Institute of NYU, Six MetroTech Center, Brooklyn NY 11201, USA
fberger@poly.edu
[2] Zentrum Mathematik, Technische Universität München, 80290 München, Germany
gritzman@ma.tum.de
[3] FB IV, Mathematik, Universität Trier, 54286 Trier, Germany
devries@uni-trier.de

Abstract. Minimum cycle bases of weighted undirected and directed graphs are bases of the cycle space of the (di)graphs with minimum weight. We survey the known polynomial-time algorithms for their construction, explain some of their properties and describe a few important applications.

1 Introduction

Minimum cycle bases of undirected or directed multigraphs are bases of the cycle space of the graphs or digraphs with minimum length or weight. Their intriguing combinatorial properties and their construction have interested researchers for several decades. Since minimum cycle bases have diverse applications (e.g., electric networks, theoretical chemistry and biology, as well as periodic event scheduling), they are also important for practitioners.

After introducing the necessary notation and concepts in Subsections 1.1 and 1.2 and reviewing some fundamental properties of minimum cycle bases in Subsection 1.3, we explain the known algorithms for computing minimum cycle bases in Section 2. Finally, Section 3 is devoted to applications.

1.1 Definitions and Notation

Let $\boldsymbol{G} = (V, E)$ be a directed multigraph with m edges and n vertices. Let $E = \{e_1, \ldots, e_m\}$, and let $w : E \to \mathbb{R}^+$ be a positive weight function on E. A cycle $\boldsymbol{C}$ in $\boldsymbol{G}$ is a subgraph (actually ignoring orientation) of $\boldsymbol{G}$ in which every vertex has even degree (= in-degree + out-degree). We generally neglect vertices of degree zero. $\boldsymbol{C}$ is called *simple* if it is connected and every vertex has degree two. The weight of a cycle $\boldsymbol{C}$ with respect to w is defined as $w(\boldsymbol{C}) := \sum_{e \in \boldsymbol{C}} w(e)$.

We want to associate F-vector spaces (usually for $F \in \{GF(2), \mathbb{Q}\}$) to the cycles of $\boldsymbol{G}$ and to study in particular their w-minimal bases. Towards this, choose a *cyclic order* of the edges in $\boldsymbol{C}$. The entries of the *incidence vector* $b(\boldsymbol{C})$

J. Lerner, D. Wagner, and K.A. Zweig (Eds.): Algorithmics, LNCS 5515, pp. 34–49, 2009.

of $\boldsymbol{C}$ are defined as follows with respect to this order, for $i = 1, \ldots, m$.

$$b_i(\boldsymbol{C}) = \begin{cases} -1, & \text{if } e_i \in \boldsymbol{C} \text{ and } e_i \text{ occurs in backward direction} \\ 0, & \text{if } e_i \notin \boldsymbol{C} \\ +1, & \text{if } e_i \in \boldsymbol{C} \text{ and } e_i \text{ occurs in forward direction.} \end{cases}$$

For simple cycles, there are two different orders. These orders lead to incidence vectors of opposite sign; as we are interested mainly in the linear spaces spanned by them, this order is inconsequential. For $F = GF(2)$ the orientation of the edges is ignored; so one could alternatively define the incidence vector as the ordinary binary incidence vector of cycles in the underlying graph.

The F-vector space $\mathcal{C}_F(\boldsymbol{G})$, spanned by the incidence vectors of all cycles in $\boldsymbol{G}$ is called *F-cycle space* of $\boldsymbol{G}$. A basis of $\mathcal{C}_F(\boldsymbol{G})$ is called *F-cycle basis*. The weight of an F-cycle basis is defined as the weight of its binary incidence vectors: $w(\mathcal{B}) = \sum_{\boldsymbol{C} \in \mathcal{B}} \sum_{e: C_e \neq 0} w(e)$. If $w(e) = 1$ for all $e \in E$, we also speak of $w(\mathcal{B})$ as the *length* of $\mathcal{B}$. A *minimum F-cycle basis* has minimum weight among all F-cycle bases. We omit the prefix "F" if F is clear from the context.

The idea to consider different fields F is motivated by the applications, see Section 3, some of which are formulated over $\mathbb{Q}$, whereas the cycle-basis problem was originally studied only for $GF(2)$. Additionally, there are interesting differences among elements of the cycle spaces over different fields (see Fig. 1). However, it is well-known that the dimension of the vector spaces always coincides, see, e.g., [1,2].

Proposition 1. *The dimension of the F-cycle space of a digraph $\boldsymbol{G}$ is equal to $\mu(\boldsymbol{G}) := |E| - |V| + c(\boldsymbol{G})$, where $c(\boldsymbol{G})$ denotes the number of connected components of $\boldsymbol{G}$.*

$\mu := \mu(\boldsymbol{G})$ is called the *cyclomatic number* of $\boldsymbol{G}$. It is also possible to consider the *directed cycle space of* $\boldsymbol{G}$, which is the subspace of $\mathcal{C}_{\mathbb{Q}}(\boldsymbol{G})$ generated by all cycles whose incidence vectors contain only edges of the same orientation. This space coincides with $\mathcal{C}_{\mathbb{Q}}(\boldsymbol{G})$ if and only if $\boldsymbol{G}$ is strongly connected. The algorithms in Section 2 can be executed also in this space, by restricting them to work only on cycles with edges of the same orientation.

1.2 Matroids

The most important property of the F-cycle space of a (di)graph $\boldsymbol{G}$ is that it forms a matroid. We review some standard notions for matroids, cf. [3]. A *matroid* $\mathcal{M}$ is a pair $(E, \mathcal{I})$, where E is a finite ground set and $\mathcal{I}$ is a set of subsets of E satisfying the axioms:

1. $\emptyset \in \mathcal{I}$;
2. if $I \in \mathcal{I}$ and $I' \subseteq I$ then $I' \in \mathcal{I}$; and
3. if $I_1, I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $e \in I_2 - I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Subsets of E that belong to $\mathcal{I}$ are called *independent*; all other sets are called *dependent*. Minimal dependent sets of a matroid $\mathcal{M}$ are called *circuits*. A maximal

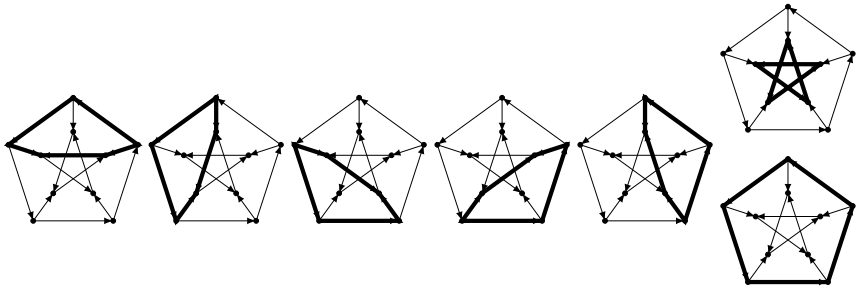


Fig. 1. Cycle bases (in bold) of an orientation of the Petersen graph (with unit edge-weights); the cycle spaces over $\mathbb{Q}$ and $GF(2)$ have dimension $\mu = 15 - 10 + 1 = 6$. The top six cycles clearly form a $\mathbb{Q}$-basis. They are not a $GF(2)$ basis, since every edge is covered twice: thus, the cycles are linearly dependent. The lower six cycles form a cycle basis over $GF(2)$ and therefore over $\mathbb{Q}$. As the Petersen graph has girth 5 and as all cycles in the bases above have length 5, the cycle bases are minimal.

independent set is called a *basis* of the matroid, and all bases have the same cardinality.

Basis exchange is a very important property of matroids: If B_1, B_2 are bases of a matroid and $x \in B_1 \setminus B_2$, then there exists $y \in B_2 \setminus B_1$ such that $(B_1 \setminus \{x\}) \cup \{y\}$ is again a basis. Also, if B is a basis and $e \in E \setminus B$ then $B \cup \{e\}$ contains a unique circuit C; furthermore, $(B \cup \{e\}) \setminus \{f\}$ is a basis for any $f \in C$.

Matroids are further characterized by the property that for any additive weighting $w\colon E \mapsto \mathbb{R}$, the greedy algorithm finds an optimal (maximum-weight) basis.

The F-cycle space of $\boldsymbol{G}$ has the structure of a matroid with sets of F-linearly independent cycles as independent sets. F-cycle bases correspond to bases of this matroid. Thus, cycles in an F-cycle basis may be replaced by certain other cycles without destroying linear independence. More precisely, let $\mathcal{B}$ be a cycle basis and $\boldsymbol{C} \in \mathcal{B}$ a cycle. Then $\boldsymbol{C}$ can be replaced by any linear combination of cycles from the basis in which $\boldsymbol{C}$ has a non-vanishing coefficient.

Because of the basis-exchange property of matroids, every F-cycle basis can be obtained from a given F-cycle basis by a series of such replacements. For minimum F-cycle bases, this implies the following lemma.

Lemma 1. *Let $\mathcal{B}$ be an F-cycle basis. If no cycle $\boldsymbol{C}$ in $\mathcal{B}$ can be exchanged for a shorter cycle $\boldsymbol{D} \notin \mathcal{B}$, then $\mathcal{B}$ is a minimum F-cycle basis.*

Consequently, if the cycles of two minimum F-cycle bases are ordered by increasing weight, their *(sorted) weight vectors*, $\vec{w} := (w(\boldsymbol{C}_1), \ldots, w(\boldsymbol{C}_\mu))$, coincide.

1.3 Basic Properties of Cycle Bases

There are several classes of special cycle bases which are of practical interest. Most important are the *fundamental tree bases* that are constructed from a spanning tree or—if $\boldsymbol{G}$ is not connected—from a spanning forest T of $\boldsymbol{G}$ by

adding edges to T. Any non-tree edge e forms a unique *fundamental cycle* $\boldsymbol{F}(e)$ with some edges of T. The set of such cycles has cardinality μ and their linear independence is obvious since every non-tree edge is contained in exactly one cycle. Hence, fundamental tree bases are F-cycle bases.

A fundamental tree basis $\mathcal{B}_T$ can be computed in time $O(mn)$, more specifically in time $O(\sum_{i=1}^{\mu} |\boldsymbol{F}_i(e)|)$, where $\boldsymbol{F}_i(e), i = 1, \ldots, \mu$, are the fundamental cycles [4]. Fundamental tree bases are not necessarily minimal among all cycle bases. Moreover, examples show that a minimum cycle basis need not be a fundamental tree basis [5,6]. See Subsection 2.5 for further discussion.

We may assume that $\boldsymbol{G}$ is simple and two-connected. If it is not, a preprocessing step can be added which takes care of multiple edges and loops and then the 2-blocks can be treated independently, see [7].

Every $GF(2)$-cycle basis is also a $\mathbb{Q}$-cycle basis. The converse is not true, see Fig. 1, and there are examples of graphs where a minimum $\mathbb{Q}$-cycle basis is strictly smaller than a minimum $GF(2)$-cycle basis, see [8,9].

Minimum F-cycle bases have important properties which are used in the algorithms for their construction. For instance, they have to contain certain cycles.

Lemma 2 ([10]). *Let $e \in E$ be an edge of a graph $\boldsymbol{G}$ through which there exists a shortest cycle $\boldsymbol{C}(e)$ with respect to w. Then there is a minimum F-cycle basis $\mathcal{B}$ containing $\boldsymbol{C}(e)$. Moreover, every minimum F-cycle basis must contain some shortest cycle through e.*

In general, the set $\{\boldsymbol{C}(e) : \boldsymbol{C}(e)$ is a shortest cycle through $e \in E\}$ does not span $\mathcal{C}_F(\boldsymbol{G})$, however. The next lemma shows that minimum F-cycle bases contain shortest paths between any two vertices.

Lemma 3 ([11,12]). *Let $v, w \in V$ and $\mathcal{B}$ be an F-cycle basis of $\boldsymbol{G}$. Let P be a shortest path from v to w. Then any cycle $\boldsymbol{C}$ of $\mathcal{B}$ that contains v and w can be exchanged for either a cycle that includes P or a cycle that excludes v or w.*

Lemma 4 ([12]). *Let $\boldsymbol{C}$ be a cycle in a minimum F-cycle basis, and let $u \in \boldsymbol{C}$ be an arbitrary vertex. Then there is an edge $(v, w) \in \boldsymbol{C}$ such that $\boldsymbol{C}$ consists of a shortest u–v path, a shortest u–w path and (v, w).*

Cycles that are contained in at least one minimum cycle basis of $\boldsymbol{G}$ are called *relevant cycles* [11,13] (defined there only for $GF(2)$). Vismara [13] showed the following result for $GF(2)$; it is valid for arbitrary fields.

Lemma 5. *Let $\boldsymbol{G}$ be a graph. A cycle $\boldsymbol{C} \in \mathcal{C}(\boldsymbol{G})$ is relevant if and only if there do not exist simple cycles, $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$, with the property that $\boldsymbol{C} = \boldsymbol{C}_1 + \cdots + \boldsymbol{C}_k$ and $w(\boldsymbol{C}_i) < w(\boldsymbol{C})$ for all $i = 1, \ldots, k$.*

For $GF(2)$, the addition of two cycles $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ corresponds to the symmetric difference

$$\boldsymbol{C}_1 + \boldsymbol{C}_2 = \boldsymbol{C}_1 \oplus \boldsymbol{C}_2 = (E(\boldsymbol{C}_1) \cup E(\boldsymbol{C}_2)) \setminus E(\boldsymbol{C}_1 \cap \boldsymbol{C}_2)$$

of the underlying edge sets.

2 Algorithms for Computing Minimum Cycle Bases

Some applications require only $GF(2)$-cycle bases while others (such as the electric networks in Subsection 3.1) can at least utilize $GF(2)$-cycle bases, although their structure is over $\mathbb{Q}$. Therefore, we will describe first the $GF(2)$-case in Subsections 2.1–2.3 and then mention the necessary modifications for $\mathbb{Q}$ in Subsection 2.4.

Due to the matroid structure of $\mathcal{C}_F(\boldsymbol{G})$, all known algorithms are based on some sort of greedy argument. However, since the number of simple cycles in a graph can be exponential in n, direct application of the greedy algorithm would be too costly. There are two main approaches to deal with this issue.

The first polynomial algorithm, due to Horton [12], addresses the problem by constructing an $O(mn)$-sized set of cycles that is guaranteed to contain a minimum cycle basis.

The second methodological approach, by de Pina [14] and independently by Berger et al. [7], contains an oracle that creates a cycle which—depending on the point of view—replaces a long cycle in a given basis or, is added to a set of cycles that is already part of a minimum cycle basis.

2.1 Horton's Algorithm

Let us start with Horton's Algorithm 1. It follows from Lemmas 3 and 4 that the cycles of the form $\boldsymbol{C}(x, y, z) = P(z, x) + (x, y) + P(y, z)$ for every triple $\{x, y, z\}$ with $(x, y) \in E$ and $z \in V$, where $P(z, x)$ denotes an *arbitrary* shortest z–x path, contain a minimum cycle basis.

A 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.

Input: Two-connected simple edge-weighted digraph $\boldsymbol{G}$
Output: Minimum $GF(2)$-cycle basis $\mathcal{B}$
1. For all $x, y \in G$, find shortest paths, $P(x, y)$, between x and y.
2. For all triples (z, x, y) with $e = \{x, y\} \in E$ construct the cycle, $\boldsymbol{C}(z, e) := P(z, x) + e + P(y, z)$, if it is simple.
3. Order all cycles, $\boldsymbol{C}(z, e)$, by increasing weight.
4. Extract greedily a cycle basis, $\mathcal{B}$, from this set by checking linear independence with Gaussian elimination.

Algorithm 1. Horton's algorithm to construct a minimum cycle basis

Since the number of candidate cycles is $O(mn)$, a direct implementation has running time $O(m^3 n)$.

2.2 An Exchange Algorithm

The algorithm begins with a fundamental tree basis and successively exchanges cycles for smaller ones if possible. The basic scheme is given in Algorithm 2.

Input: Two-connected simple edge-weighted digraph $\boldsymbol{G}$
Output: Minimum $GF(2)$-cycle basis $\mathcal{B}$
1. Construct a fundamental tree basis $\mathcal{B}_0 = \{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_\mu\}$.
2. **for** $i = 1$ to μ **do**
3. Find a shortest cycle $\boldsymbol{C}_i$ that is linearly independent of $\mathcal{B}_{i-1} \setminus \{\boldsymbol{F}_i\}$.
4. **if** $w(\boldsymbol{C}_i) < w(\boldsymbol{F}_i)$ **then**
5. $\mathcal{B}_i := (\mathcal{B}_{i-1} \setminus \{\boldsymbol{F}_i\}) \cup \{\boldsymbol{C}_i\}$
6. **end if**
7. **end for**
8. Output $\mathcal{B} := \mathcal{B}_\mu$

Algorithm 2. Basic exchange scheme

Of course, the crucial part is the oracle in Line 3 which selects a new cycle $\boldsymbol{C}_i$ in each iteration of the algorithm.

Let $\mathcal{B}$ be a cycle basis. The cycle-edge incidence matrix $A \in GF(2)^{\mu \times m}$ of $\mathcal{B}$ has as rows the (incidence vectors of the) cycles in $\mathcal{B}$. We denote the rows of A as well as the cycles in $\mathcal{B}$ by $\boldsymbol{C}_i, i = 1, \ldots, \mu$.

When looking at linear (in)dependence of cycles, it suffices to restrict attention to the entries corresponding to the non-tree-edges of a spanning tree of the graph: a set of cycles is linearly independent if and only if the submatrix of A corresponding to the non-tree-edges has full rank. Henceforth we will assume that we have fixed a spanning tree for this purpose.

We will now describe Line 3 of Algorithm 2 for a fixed step i. Let U be the inverse matrix of A (restricted to the non-tree edges) and let u_i be the i-th column of U padded with $m - \mu$ zeros (in place of the tree-edges). Denote by $A^{(-i)}$ the matrix that results from A after removing the i'th row $\boldsymbol{C}_i$. Clearly u_i is in the kernel of $A^{(-i)}$. The vector u_i can be computed using standard matrix operations. Alternatively, it is also possible to determine the entire matrix U in each step; this leads to a simple update possibility for step $i + 1$, see [14].

The following trivial remark gives an easy-to-check criterion for whether a cycle $\boldsymbol{C}$ can replace a basis cycle $\boldsymbol{C}_i \in \mathcal{B}$, using u_i:

Remark 1. Let $\mathcal{B}$ be a cycle basis and let $\boldsymbol{C} \notin \mathcal{B}$ be a cycle. $\boldsymbol{C}$ is linearly independent of the row-space of $A^{(-i)}$ if and only if $\langle \boldsymbol{C}, u^i \rangle \neq 0$ holds with the vector u^i defined above.

The vectors $u^i, i = 1, \ldots, \mu$, have the property that $\langle \boldsymbol{C}_j, u^i \rangle = 0$, for $j \neq i$. According to the criterion of Remark 1, it suffices to compute a shortest cycle $\boldsymbol{C}$ for which $\langle \boldsymbol{C}, u^i \rangle = 1$ in step i of the algorithm. This is achieved by computing a shortest path in an auxiliary undirected graph $\boldsymbol{G_u}$. For $u \in \{0,1\}^E$ the graph $\boldsymbol{G_u}$ is constructed from $\boldsymbol{G}$ as follows (for an example see Figure 2): The vertex set of $\boldsymbol{G_u}$ is $V \times GF(2)$. For each edge $e = \{x, y\}$ in $\boldsymbol{G}$ add the two edges $\{(x, 0), (y, 0 \oplus u_e)\}$ and $\{(x, 1), (y, 1 \oplus u_e)\}$ to $\boldsymbol{G_u}$. Define a weight function $w_u : \boldsymbol{G_u} \to \mathbb{R}^+$ by assigning each edge the weight $w(e)$ of the corresponding edge e in $\boldsymbol{G}$ it comes from. $\boldsymbol{G_u}$ has $2n$ vertices and $2m$ edges. For a vertex v of $\boldsymbol{G}$ and a simple

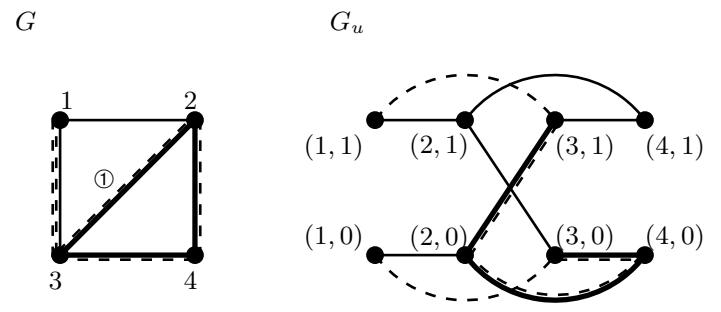


Fig. 2. The dashed $(1,0)$–$(1,1)$ path in $\boldsymbol{G_u}$ does not correspond to a cycle in $\boldsymbol{G}$. However, it contains the bold $(3,0)$–$(3,1)$ sub-path corresponding to cycle $\{2,3,4\}$. Here, u contains only one nonzero entry, namely for edge 2–3, displayed as ① in $\boldsymbol{G}$.

$(v,0)$–$(v,1)$ path $\boldsymbol{P_v}$ in $\boldsymbol{G_u}$ let $\boldsymbol{W(P_v)}$ denote the closed walk in $\boldsymbol{G}$ obtained by replacing each vertex (x,u) of $\boldsymbol{P_v}$ by x. For the next observation see [7].

Lemma 6. *Let v be a vertex of $\boldsymbol{G}$, and let $\boldsymbol{P_v}$ be a simple $(v,0)$–$(v,1)$ path in G_u. Then the closed walk $\boldsymbol{W(P_v)}$ contains a simple cycle $\boldsymbol{C}$ in $\boldsymbol{G}$ with $\langle \boldsymbol{C}, u\rangle = 1$.*

It is not difficult to show that among all paths $\boldsymbol{P_v}$, a shortest one always corresponds to a simple cycle $\boldsymbol{C} = \boldsymbol{W(P_v)}$ in $\boldsymbol{G}$. Hence, to find a shortest cycle $\boldsymbol{C}$ with $\langle \boldsymbol{C}, u\rangle = 1$, one computes a shortest $(v,0)$–$(v,1)$ path in $\boldsymbol{G_u}$ for each v and retains only a shortest one.

Theorem 1 ([7]). *Algorithm 2 computes a minimum cycle basis of $\boldsymbol{G}$.*

A direct implementation runs in time $O(\max\{m^3, mn^2 \log n\})$.

2.3 Speed-Ups

There are two major bottlenecks in both algorithms. On the one hand there is a large number of shortest-path computations (n^2 and μn of them, respectively), and on the other hand, for the linear independence tests or the computation of the vectors u^i, Gaussian elimination on an $O(mn) \times \mu(\boldsymbol{G})$ or a $\mu(\boldsymbol{G}) \times \mu(\boldsymbol{G})$ matrix, respectively, is needed.

There are ways to improve on both. Golynski and Horton [15] use a recursive procedure to check linear independence of the cycle-edge incidence matrix of the candidate cycles: the matrix is partitioned into blocks and the pivot operations are performed block-wise. This allows the use of fast matrix multiplication and accelerates the algorithm. Consequently, the performance improves from $O(m^3 n)$ to $O(m^\omega n)$, where ω denotes the matrix multiplication constant (it is presently known, that $\omega < 2.376$).

Kavitha, Mehlhorn, Michail, and Paluch [16] found a similar way to apply fast matrix multiplication and a recursive computation of the vectors u^i with delayed updates to Algorithm 2, which reduces its running time from $O(m^3 + mn^2 \log n)$ to $O(m^2 n + mn^2 \log n)$; for dense graphs, this is an improvement.

Finally, Mehlhorn and Michail [17] have recently improved the running time to $O(m^2 n/\log n + mn^2)$. The idea here is that instead of solving n shortest path problems in each step of Algorithm 2 for constructing the next cycle, one may obtain that cycle by searching through a suitable set of candidate cycles, for which the linear products $\langle \boldsymbol{C}, u^i \rangle$ are computed efficiently from a shortest path tree structure. This approach only requires $O(n)$ shortest path computations. The candidate cycles are a subset of Horton's candidate set from Subsection 2.1.

2.4 The Case of $\mathbb{Q}$

Horton's algorithm from Subsection 2.1 can be used without modification also for computing a $\mathbb{Q}$-minimum cycle basis. Note however, that for Gaussian elimination over $\mathbb{Q}$, it is necessary to pay for the calculations on rational numbers with up to $O(m \log m)$ bits by an additional factor of $\tilde{O}(m)$ (which means $O(m \log^k(m))$ for some k) in asymptotic running time, representing the cost of each arithmetic operation. Hence the algorithm takes time $\tilde{O}(m^4 n)$ if implemented directly. With the divide and conquer strategy sketched in Subsection 2.3, this may be reduced to $\tilde{O}(m^{\omega+1} n)$, see [9].

Algorithm 2 can also be adapted to work over $\mathbb{Q}$, see [18]. First, it has to be determined how to compute the vectors u^i from Remark 1. Kavitha and Mehlhorn show that such vectors with size bounded by $||u^i|| \leq i^{i/2}$ can be computed efficiently. Second, one has to construct a shortest cycle $\boldsymbol{C}$ with $\langle \boldsymbol{C}, u^i \rangle \neq 0$ in each step. This can be done by working in $GF(p)$, for a set of small primes p and applying the Chinese remainder theorem. The shortest path calculation is done in a graph with p levels, but otherwise analogously. The above mentioned block-updates lead to a running time of $O(m^3 n + m^2 n^2 \log n)$. Finally, the variant proposed in [17] gives a running time of $O(m^3 n)$.

2.5 Other Algorithms and Variants

In many cases, it is not required that the F-cycle basis be minimal. Cycle bases of low but not necessarily minimal weight can be computed more quickly. Several different heuristic algorithms exist, see [6,12,19,20,21] and the references therein.

Algorithms for special graphs are also widely discussed, but references will be omitted here. We only mention [22] for planar graphs with running time $O(n^2 \log n + m)$. Approximation algorithms are proposed in [16,23,17], with a constant approximation factor. Randomized algorithms are considered in [24,17].

Cycle bases with additional properties are also of interest. For a discussion of different classes, see, e.g., [20,25]. The problem to find a minimum cycle basis among all fundamental tree bases is APX-hard [26]. The same holds for so-called weakly fundamental cycle bases [27]. However, the existence of an unweighted fundamental tree basis with a length of $O(n^2)$ has been proved recently, see [28].

3 Applications

Cycle bases appear as tools in several areas of engineering. We limit ourselves to three which reflect their main functions. These can be summarized as follows:

- The cycles of the basis are explicitly needed; the basis need not be minimal, but a shorter cycle basis may speed up an ensuing algorithm.
- The cycles are needed explicitly; it is essential that the basis be minimal.
- The cycle basis itself is not needed explicitly; but information is derived from it. It is necessary that the basis is minimal.

3.1 Electric Networks

Cycle bases can be used to model the second Kirchhoff law in the design of electric networks. Electric circuits consist of interconnected network elements whose properties can be described by non-linear differential equations in terms of currents and voltages at time t.

The structure of the interconnection is given by a directed unweighted graph, $\boldsymbol{G} = (V, E)$ whose edges correspond to network elements. Two edges meet in a vertex if the corresponding elements are joined by a wire. More complicated network elements with several ports can be represented by an equivalent circuit diagram consisting of simple circuit elements [29]. The direction of an edge represents the voltage drop with respect to a specified ground node.

In addition to the properties of the circuit elements, the two time-independent Kirchhoff laws govern the behavior of the network:

Current law: The sum of the currents across any edge cut of the network is zero at all times.

Voltage law: The sum of the voltages along any mesh (directed cycle) in the network is zero at all times. If the matrix A is the cycle-edge incidence matrix of a $\mathbb{Q}$-cycle basis, this law can be written as $AU = 0$, where $U \in \mathbb{R}^{|E|}$ denotes the unknown edge voltages (at time t).

The combination of the Kirchhoff laws and the differential equations which describe the circuit elements is called the *network equation*. It completely describes the network and can be used to determine the unknown voltages and currents during the design process. There are different formulations of the network equation; the sparse tableau analysis [30] and the extended modified nodal analysis model, see, e.g., [31] both contain explicitly the equations modeling the Kirchhoff voltage law.

Since the graphs of electric networks can be very large, subsequent computations such as a generic solvability check and error detection algorithms are influenced by the length of the cycle basis used. A short or minimum $\mathbb{Q}$-cycle basis is therefore desirable, in particular if the computations need to be done repeatedly. In practice, short $\mathbb{Q}$-cycle bases will be used rather than minimal ones to improve the running time.

3.2 Chemistry and Biology

For the investigation of functional properties of a chemical compound, it is of interest to study the cycles (or rings) of the molecular graph. The goal is to determine how the ring structure of a molecule influences its chemical and physical

properties. One of the most commonly used sets of rings is a minimum $GF(2)$-cycle basis of the graph (called Smallest Set of Smallest Rings, see, e.g., [32]). It is also possible to use the set of relevant cycles instead, for instance.

In this case, a minimum $GF(2)$-cycle basis needs to be explicitly determined. For algorithms which compute all relevant cycles, see [13,33,34].

A different application is the use of a minimum $GF(2)$-cycle basis of the molecular graph for the derivation of graph invariants: Structural information about molecules is available in molecular databases. Most databases offer search and retrieval functionality for molecules also by structure, i.e., the user draws or uploads a molecular graph $\boldsymbol{G}$, and then the database is searched for molecules whose structure is identical to $\boldsymbol{G}$ or which contain substructures identical to $\boldsymbol{G}$.

This identification requires solving the notorious (sub-)graph isomorphism problem for $\boldsymbol{G}$ for each molecule in the database. A common strategy to avoid solving many graph isomorphism problems exactly is to eliminate as many candidate structures as possible by comparing *graph invariants* known in this context as *molecular descriptors*. The most popular descriptors involve the number of atoms and bonds (vertices and edges) and the different atom types. Graphs from the database whose invariants do not coincide with those of $\boldsymbol{G}$ (or do not allow a substructure isomorphic to $\boldsymbol{G}$) are immediately rejected.

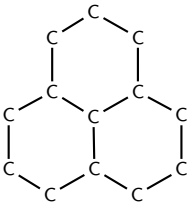
As a consequence of the matroid property and Lemma 1, the weight of a minimum cycle basis as well as the weight vector $\vec{w}(\boldsymbol{G})$ are graph invariants. In [34], other invariants are considered which, taken together, provide a strong description of the structure of the cycle space.

The idea is to identify the 'functionality' of a relevant cycle in a minimum $GF(2)$-cycle basis, i.e., stated informally, to characterize the part of the cycle space it spans. Cycles with the same function can in principle be exchanged for each other without violating linear independence; this redundancy defines an equivalence relation, the *interchangeability relation* $\sim_\kappa$, [33], that can be characterized as follows.

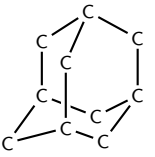
Lemma 7. *Let $\boldsymbol{C}, \boldsymbol{C}'$ be two relevant cycles of weight κ. Then $\boldsymbol{C} \sim_\kappa \boldsymbol{C}'$ if and only if there is a representation $\boldsymbol{C} = \boldsymbol{C}' \oplus \bigoplus_{\boldsymbol{D} \in \mathcal{I}} \boldsymbol{D}$, where $\{\boldsymbol{C}'\} \cup \mathcal{I}$ is a (linearly) independent subset of the relevant cycles of weight smaller than or equal to κ.*

This equivalence relation can also be interpreted as a refined form of the connectivity concept in the cycle matroid of the graph. The $\sim_\kappa$ equivalence classes are called *interchangeability classes*.

If a graph contains a unique shortest cycle $\boldsymbol{C}(e)$ for some edge e, this cycle is necessarily contained in every minimum cycle basis, and it cannot be replaced by any other cycle. Hence it forms its own equivalence class with respect to $\sim_\kappa$ where $\kappa = w(\boldsymbol{C})$. On the other hand, there often exist several relevant cycles which are $\sim_\kappa$ interchangeable for $\kappa = w(\boldsymbol{C})$. Minimum cycle bases of unweighted complete graphs, for instance, have only one interchangeability class: suitable $\frac{1}{2}(n^2 - 3n + 2)$ of the $\binom{n}{3}$ triangles need to be contained in a minimum cycle basis; they are all interchangeable.



G_1



G_2

Fig. 3. Molecules $\boldsymbol{G}_1$ (Perhydrophenalene) $C_{13}H_{22}$ and $\boldsymbol{G}_2$ (Adamantane) $C_{13}H_{16}$. All edge weights are assumed to be equal to one. The minimum cycle basis of $\boldsymbol{G}_1$ consists of the three hexagons. Since they are unique, each hexagon represents its own $\sim_6$ equivalence class. In $\boldsymbol{G}_2$, any three of the four visible hexagons add up to the fourth, hence they are all interchangeable. Any three of them form a minimum cycle basis.

It is possible to compute the generally exponential interchangeability classes explicitly. However, the following observation leads to a graph invariant which can be computed in polynomial time:

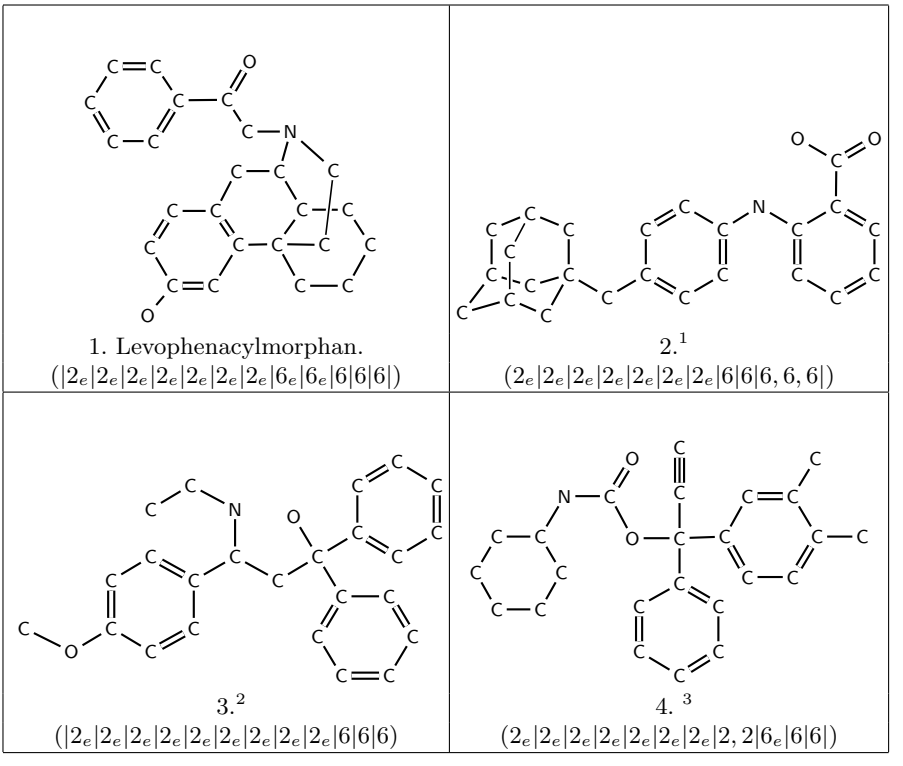
Lemma 8 ([33]). *Let $\kappa > 0$ be the weight of some relevant cycle, let $\mathcal{B}, \mathcal{B}'$ be two different minimum cycle bases and let $\mathcal{W}^\kappa$ be an equivalence class for $\sim_\kappa$. Then $|\mathcal{B} \cap \mathcal{W}^\kappa| = |\mathcal{B}' \cap \mathcal{W}^\kappa|$.*

Therefore we may call $|\mathcal{B} \cap \mathcal{W}^\kappa|$ the *relative rank* of $\mathcal{W}^\kappa$. This rank describes how many cycles of equal weight in a minimum cycle basis are related with respect to interchangeability. For each weight κ of a relevant cycle, there may be several equivalence classes. (Graph $\boldsymbol{G}_1$ in Fig. 3 has three $\sim_6$ equivalence classes, each with relative rank 1, Graph $\boldsymbol{G}_2$ has one $\sim_6$ equivalence class of relative rank 3.) The ordered vector $\vec{\beta}(\boldsymbol{G})$ containing the relative ranks of the $\sim_\kappa$ equivalence classes is a graph invariant. Thus, intuitively, $\vec{\beta}(\boldsymbol{G})$ measures the degree of interconnectivity or local density of $\boldsymbol{G}$'s cyclic structure (compare again the two graphs in Fig. 3).

Theorem 2 ([34]). *$\vec{\beta}(\boldsymbol{G})$ can be computed in time $O(m^4 n)$.*

We may encode the information gained from a minimum cycle basis and the relative ranks of the interchangeability classes within one vector: Vertical lines separate the entries in $\vec{w}(\boldsymbol{G})$ according to the $\sim_\kappa$ equivalence classes. The relative rank of each class corresponds to the number of entries between two vertical lines, sorted by increasing rank. A subscript e means that the corresponding equivalence class has cardinality 1.

In Figure 4, compare in particular graphs 1 and 2, and 3 and 4, respectively. For each pair, the vector $\vec{w}(\boldsymbol{G})$ is equal, but the difference can be noticed by looking at the information about the relative ranks.



1. Levophenacylmorphan.
$(|2_e|2_e|2_e|2_e|2_e|2_e|2_e|6_e|6_e|6|6|6|)$

2.[1]
$(2_e|2_e|2_e|2_e|2_e|2_e|2_e|6|6|6, 6, 6|)$

3.[2]
$(|2_e|2_e|2_e|2_e|2_e|2_e|2_e|2_e|2_e|6|6|6)$

4.[3]
$(2_e|2_e|2_e|2_e|2_e|2_e|2_e|2_e|2, 2|6_e|6|6|)$

Fig. 4. Different molecules with the molecular formula $C_{24}H_{27}NO_2$ and their invariants

[1] N-(4-(1-adamantylmethyl)phenyl)anthranilic acid
[2] 1-,1-diphenyl-3-(ethylamino)-3-(p-methoxyphenyl)-1-propanol
[3] N-cyclohexyl-,1-phenyl-1-(3,4-xylyl)-2-propynyl ester

3.3 Periodic Event Scheduling

In periodic event scheduling problems, the events $v_1, \ldots, v_n$ have to be scheduled to occur periodically with period $T > 0$ subject to certain constraints [35,36,37,38,39].

A typical example is the construction of a bus or subway schedule in which a bus or subway services each station on its line at regular intervals of T minutes. A valid schedule will consist of instructions for each driver on each of the lines, at what time to arrive at and to leave from each station. There may be constraints on such a schedule, for instance, interconnection constraints between different lines. These could provide passengers with transfer options or prevent trains from arriving at the same time at a station for security reasons. Further, it is of course of interest to find a schedule which is also optimal with respect to a linear or nonlinear objective function, as for instance the average customer connection time.

A periodic event v consists of an infinite number of individual events $v = (v^i)_{i\in\mathbb{N}}$ such that

$$t(v^i) - t(v^{i-1}) = T$$

holds for the starting times $t(v^i)$ of the event v^i. Clearly, the starting time of a single individual event determines that of all other events in v. Thus, we define the *starting time* of v by

$$\pi(v) = \min\{t(v^i) + pT : i \in \mathbb{Z}, t(v^i) + pT \geq 0\} \in [0, T).$$

Spans allow the modeling of constraints: trivial constraints prevent, e.g., a train from leaving a station before it arrives; interconnection constraints may restrict the schedules of different lines: Given $l, u \in \mathbb{R}$ such that $0 \leq u - l < T$, a *span* is the union $[l, u]_T = \bigcup_{k\in\mathbb{Z}}[l + kT, u + kT]$. A span constraint $e = (v_i, v_j)$ is a relation between an ordered pair of periodic events and a span $[l_{ij}, u_{ij}]_T$ so that

$$\pi(v_j) - \pi(v_i) \in [l_{ij}, u_{ij}]_T.$$

The periodic event scheduling problem (PESP) is given by the tuple (T, V, E) where E is a set of span constraints. The task is to construct an (optimal) schedule $\pi : V \to \mathbb{R}$ so that all span constraints in E are fulfilled. The problem is known to be NP-hard [35]. There exist various mixed integer programming (MIP) formulations. We describe one that is based on the event graph $G = (V, E)$, with vertex set $V = \{v_1, \ldots, v_n\}$ and oriented edges between v_i and v_j whenever $e = (v_i, v_j) \in E$. The edges are labeled with the spans $[l_{ij}, u_{ij}]_T$. A valid schedule can then be seen as a labeling of the vertices of the graph by $\pi(v_i)$ so that for each edge e and $p_{ij} \in \mathbb{Z}$,

$$\pi(v_j) + p_{ij} \cdot T \in \{\pi(v_i) + x_{ij} : x_{ij} \in [l_{ij}, u_{ij}]\}$$

for x_{ij} in the regular interval $[l_{ij}, u_{ij}]$. The label $\pi(v_i)$ is also called *potential* of v_i. With $x_{ij} = \pi(v_j) - \pi(v_i)$, the expression $x_{ij} + p_{ij} \cdot T$ is referred to as *periodic voltage*. Exploiting this analogy to electric networks and assuming that the underlying oriented graph is two-connected, a schedule is valid if and only if $\langle \boldsymbol{C}, x\rangle = 0$ for any cycle $\boldsymbol{C}$ in $\boldsymbol{G}$, where the vectors $x = (x_{ij}) \in \mathbb{R}^m, p = (p_{ij}) \in \mathbb{Z}^m$ fulfill the corresponding box constraints $l \leq x + pT \leq u$ with $l = l_{ij}$ and $u = (u_{ij})$. This gives rise to the MIP-formulation:

$$\begin{aligned} \min \quad & c(x + pT) \\ Ax &= 0 \\ l \leq x + pT &\leq u \\ p &\in \mathbb{Z}^m, \end{aligned}$$

where A denotes the cycle-edge incidence matrix of a $\mathbb{Q}$-cycle basis of $\boldsymbol{G}$ and $c(x + pT)$ is an objective function in terms of $x + pT$. The problem can be reformulated in more condensed form [40]; then A has to be the incidence matrix of an *integral* $\mathbb{Q}$-cycle basis, a cycle basis which uses only integer coefficients to generate all cycles in $\boldsymbol{G}$, see [41,20].

Experiments show that the problem can be solved faster when the chosen cycle basis is small. Several real-world case studies exist. For instance, the Berlin subway network and part of the Deutsche Bahn network was studied in [42,40], the Munich subway network was analyzed in [43].

4 Conclusion

Minimum and other cycle bases over different fields are interesting objects. The known algorithms all exploit the matroid property.

An open problem is to understand the exact difficulty frontier between minimum cycle bases and minimum cycle bases with an additional property: minimum cycle bases can be computed in polynomial time. Minimum fundamental tree cycle bases are hard to compute and to approximate, and minimum weakly fundamental cycle bases are hard to approximate, but it is not clear what happens in between: fundamental cycle bases are a proper subset of integral cycle bases, for instance, but currently, no polynomial time algorithm is known for computing a minimum integral cycle basis.

Acknowledgments. The authors were supported by the DFG Schwerpunktprogramm Nr. 1126, grants GR 933/8-1, GR 933/8-2. We thank the anonymous referees for some helpful remarks.

References

1. Biggs, N.: Algebraic Graph Theory. Cambridge University Press, Cambridge (1993)
2. Diestel, R.: Graph Theory. Graduate Texts in Mathematics, vol. 173. Springer, Heidelberg (1997)
3. Oxley, J.G.: Matroid Theory. Oxford University Press, Oxford (1992)
4. Paton, K.: An algorithm for finding a fundamental set of cycles of a graph. Comm. ACM 12, 514–519 (1969)
5. Hartvigsen, D., Zemel, E.: Is every cycle basis fundamental? J. Graph Theory 13, 117–137 (1989)
6. Deo, N., Prabhu, G., Krishnamoorthy, M.: Algorithms for generating fundamental cycles in a graph. ACM Trans. Math. Softw. 8, 26–42 (1982)
7. Berger, F., Gritzmann, P., de Vries, S.: Minimum cycle bases for network graphs. Algorithmica 40, 51–62 (2004)
8. Horton, J., Berger, F.: Minimum cycle bases of graphs over different fields. Electronic Notes in Discrete Mathematics 22, 501–505 (2005)
9. Liebchen, C., Rizzi, R.: A greedy approach to compute a minimum cycle basis of a directed graph. Information Processing Letters 94, 107–112 (2005)
10. Hubicka, E., Sysło, M.: Minimal bases of cycles of a graph. In: Fiedler, M. (ed.) Recent Advances in Graph Theory, Proc. 2nd Czechoslovak Conference on Graph Theory, Academia, pp. 283–293 (1975)
11. Plotkin, M.: Mathematical basis of ring-finding algorithms in CIDS. J. Chem. Documentation 11, 60–63 (1971)
12. Horton, J.: A polynomial-time algorithm to find the shortest cycle basis of a graph. SIAM J. Comput. 16, 358–366 (1987)

13. Vismara, P.: Union of all the minimum cycle bases of a graph. Electronic J. Comb. 4, R9 (1997)
14. de Pina, J.: Applications of shortest path methods. Ph.D thesis, University of Amsterdam, The Netherlands (2002)
15. Golynski, A., Horton, J.D.: A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In: Penttonen, M., Schmidt, E.M. (eds.) SWAT 2002. LNCS, vol. 2368, p. 200. Springer, Heidelberg (2002)
16. Kavitha, T., Mehlhorn, K., Michail, D., Paluch, K.: A faster algorithm for minimum cycle bases of graphs. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 846–857. Springer, Heidelberg (2004)
17. Mehlhorn, K., Michail, D.: Minimum cycle bases: Faster and simpler. ACM Trans. Algorithms (2009) (to appear)
18. Kavitha, T., Mehlhorn, K.: Algorithms to compute minimum cycle basis in directed graphs. Theory of Computing Systems 40, 485–505 (2007)
19. Hartvigsen, D., Mardon, R.: The prism-free planar graphs and their cycles bases. J. Graph Theory 15, 431–441 (1991)
20. Berger, F.: Minimum cycle bases in graphs. Ph.D thesis, TU München (2004)
21. Amaldi, E., Liberti, L., Maculan, N., Maffioli, F.: Efficient edge-swapping heuristics for finding minimum fundamental cycle bases. In: Ribeiro, C.C., Martins, S.L. (eds.) WEA 2004. LNCS, vol. 3059, pp. 14–29. Springer, Heidelberg (2004)
22. Hartvigsen, D., Mardon, R.: When do short cycles generate the cycle space? J. Comb. Theory, Ser. B 57, 88–99 (1993)
23. Kavitha, T., Mehlhorn, K., Michail, D.: New approximation algorithms for minimum cycle bases of graphs. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 512–523. Springer, Heidelberg (2007)
24. Hariharan, R., Kavitha, T., Mehlhorn, K.: Faster deterministic and randomized algorithms for minimum cycle basis in directed graphs. SIAM J. Comp. 38, 1430–1447 (2008)
25. Liebchen, C., Rizzi, R.: Classes of cycle bases. Discrete Appl. Math. 55, 337–355 (2007)
26. Galbiati, G., Amaldi, E.: On the approximability of the minimum fundamental cycle basis problem. In: Solis-Oba, R., Jansen, K. (eds.) WAOA 2003. LNCS, vol. 2909, pp. 151–164. Springer, Heidelberg (2004)
27. Rizzi, R.: Minimum weakly fundamental cycle bases are hard to find. Algorithmica (2009), doi:10.1007/s00453–007–9112–8
28. Elkin, M., Liebchen, C., Rizzi, R.: New length bounds for cycle bases. Information Processing Letters 104, 186–193 (2007)
29. Chen, W.K.: Active Network Analysis. Advanced series in electrical and computer engineering, vol. 2. World Scientific, Singapore (1991)
30. Hachtel, G., Brayton, R., Gustavson, F.: The sparse tableau approach to network analysis and design. IEEE Transactions of Circuit Theory CT-18, 111–113 (1971)
31. Estévez-Schwarz, D., Feldmann, U.: Diagnosis of erroneous circuit descriptions. Slides of a presentation, Paderborn (March 2003)
32. Downs, G.M., Gillet, V.J., Holliday, J.D., Lynch, M.F.: Review of ring perception algorithms for chemical graphs. J. Chem. Inf. Comput. Sci. 29, 172–187 (1989)
33. Gleiss, P., Leydold, J., Stadler, P.: Interchangeability of relevant cycles in graphs. Electronic J. Comb. 7, R16 (2000)
34. Berger, F., Gritzmann, P., de Vries, S.: Computing cyclic invariants for molecular graphs (2008) (manuscript)
35. Serafini, P., Ukovich, W.: A mathematical model for periodic scheduling problems. SIAM J. Discrete Math. 2, 550–581 (1989)

36. Odijk, M.: Railway timetable generation. Ph.D thesis, TU Delft, The Netherlands (1997)
37. Nachtigall, K.: Periodic network optimization with different arc frequencies. Discrete Appl. Math. 69, 1–17 (1996)
38. Nachtigall, K., Voget, S.: Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks. Europ. J. Oper. Res. 103, 610–627 (1997)
39. Liebchen, C., Möhring, R.: A case study in periodic timetabling. In: Proceedings of ATMOS 2002. Elec. Notes in Theor. Comput. Sci (ENTS), vol. 66.6 (2002)
40. Liebchen, C., Proksch, M., Wagner, F.: Performance of algorithms for periodic timetable optimization. In: Computer-aided Systems in Public Transport. Lect. Notes in Economics and Mathematical System, vol. 600, pp. 151–180 (2008)
41. Liebchen, C.: Finding short integral cycle bases for cyclic timetabling. In: Di Battista, G., Zwick, U. (eds.) ESA 2003. LNCS, vol. 2832, pp. 715–726. Springer, Heidelberg (2003)
42. Liebchen, C., Möhring, R.: The modeling power of the periodic event scheduling problem: railway timetables – and beyond. In: Computer-aided Systems in Public Transport, pp. 117–150 (2008)
43. Kaminski, A.: Erzeugung und Optimierung zyklischer Zeitpläne. Diploma Thesis (in German), TU München (2004)