Low–exponential algorithm for counting the number of edge cover on simple graphs

J. A. Hernández-Servín[1], J. Raymundo Marcial-Romero[1], Guillermo De Ita Luna[2]

[1] Facultad de Ingeniería, UAEM
[2] Facultad de Ciencias de la Computación, BUAP
xoseahernandez@uaemex.mx, jrmarcialr@uaemex.mx, deita@cs.buap.mx

Abstract. A procedure for counting edge covers of simple graphs is presented. The procedure splits simple graphs into non-intersecting cycle graphs. This is the first "low exponential" exact algorithm to count edge covers for simple graphs whose upper bound in the worst case is $O(1.465575^{(m-n)} \times (m+n))$, where m and n are the number of edges and nodes of the input graph, respectively.

Keywords: Edge covering, Graph theory, Integer partition.

1 Introduction

A graph is a pair $G = (V, E)$, where V is a set of *vertices* and E is a set of *edges* that associates pairs of vertices. The number of vertices and edges is denoted by $v(G)$ and $e(G)$, respectively. A *simple graph* is an unweighted, undirected graph containing no graph *loops* or multiple edges. Through the paper only *simple finite graphs* will be considered, where G is a *finite graph* if $|v(G)| < \infty$ and $|e(G)| < \infty$. An *edge cover* of a graph G is a set of edges $C \subseteq E_G$, such that meets all vertices of G. That is for any $v \in V$, it holds that $E_v \cap C \neq \emptyset$ where E_v is the set of edges incident to v. The family of *edge covers* for the graph G wil be denoted by $\mathcal{E}_G$. The problem of computing the cardinality of $\mathcal{E}_G$ is well known to be $\sharp$P-complete problem. In [4], however, have designed to what they call a fully polynomial time approximation scheme (FPTAS) for counting edge covers of a simple graphs; same authors have extended the technique to tackle the weighted edge cover problem in [3]. In this paper, we study a novel algorithm for counting edge covers taking into account that there exists a time polynomial algorithm for non-intersecting cyclic graphs [2]. Having said that, the technique is basically to reduce a simple graph into a sequence of non-intersecting cyclic graphs. The complexity of the algorithm is also studied. Although, the complexity of our proposed algorithm remains exponential its complexity is comparatively low to the ones reported in the literature.

2 Splitting simple graphs

A *subgraph* of a graph G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$. If $e \in E$, e can simply be removed from graph G, yielding a subgraph denoted by $G \backslash e$; this is obviously the graph $(V, E - e)$. Analogously, if $v \in V$, $G \backslash v$ is the graph $(V - v, E')$ where $E' \subseteq E$ consists of the edges in E except those incident at v. A *spanning subgraph* is a subgraph computed by deleting a number of edges while keeping all its vertices covered, that is if $S \subset E$ is a subset of E, then a spanning subgraph of $G = (V, E)$ is a graph $(V, S \subseteq E)$ such that for every $v \in V$ it holds $E_v \cap S \neq \emptyset$.

A *path* in a graph is a linear sequence of adjacent vertices, whereas a *cycle* in a graph G is a simple graph whose vertices can be arranged in a cyclic sequence in such a way that two vertices are adjacent if they are consecutive in the sequence, and are nonadjacent otherwise [1]. The length of a path or a cycle is the number of its edges.

An *acyclic* graph is a graph that does not contain cycles. The connected acyclic graphs are called *trees*, and a connected graph is a graph that for any two pair of vertices there exists a path connecting them. The number of connected components of a graph G is denoted by $c(G)$. It is not difficult to infer that in a tree there is a unique path connecting any two pair of vertices. Let $T(v)$ be a tree T with root vertex v. The vertices in a tree with degree equal to one are called *leaves*.

2.1 The cycle space

A *cycle basis* is a minimal set of basic cycles such that any cycle can be written as the sum of the cycles in the basis [5]. The sum of cycles C_i is defined as the subgraph $C_1 \oplus \cdots \oplus C_k$, where the edges are those contained in an odd number of C_i's, $i \in \{1, ..., k\}$ with $k \in \mathbb{N}$ arbitrary; the sum is again a cycle. The aforementioned sum gives to the set of cycle or cycle space $\mathcal{C}$ the structure of a vector space under a given field $\mathtt{k}$. The dimension of the cycle space $\mathcal{C}$ is $\dim_{\mathtt{k}} \mathcal{C} = |\mathcal{B}|$ where $\mathcal{B}$ is a basis for $\mathcal{C}$. In particular, if G is a simple graph the field is taken to be $\mathtt{k} = \mathbb{F}_2$, which is the case concerning this paper. Thus the field $\mathbb{F}_2$ will be used through the entire paper to describe cycle space of graphs. The technique proposed in this paper, assumes that if the graph has cycles, it is always possible to calculate a cycle basis for the cycle space of the graph. The cycle basis to be considered in this paper can easily be constructed by using the well known depth first search algorithm (DFS) [1]. The proccess of getting a spanning tree for G by DFS algorithm will be denoted by $\langle G \rangle$. By using depth first search an spanning tree or forest $T = \langle G \rangle$ can be constructed for any graph G. The cycle basis are the unique cycle formed with edges in the cotree $\bar{T} = G \backslash T$ and edges in T. The dimension of $\mathcal{C}_G$ is therefore $|\bar{T}|$.

2.2 Non-intersecting cycle graph or basic graphs

In this paper we define *basic graphs* or *non-intersecting cycle graphs* as those simple graphs G with $\dim \mathcal{C}_G \neq 0$ in such a way that any pair of basic cycles are edge disjoints. Let $\mathcal{B} = \{C_1, ..., C_k\}$ a basis for the cycle space $\mathcal{C}_G$; if $C_k = (V_k, E_k)$ let us define the sequence of intersections of the edge sets $\{E_i\}$ as

$$B_p = \bigcup_{i_1 \neq \cdots \neq i_p} [E_{i_1} \cap \cdots \cap E_{i_p}] \tag{2.1}$$

for any $I_p = \{i_1, ..., i_p\} \subseteq \{1, ..., k\}$, it is clear that $E_{i_1} \cap \cdots \cap E_{i_p} = E_{i_{\sigma(1)}} \cap \cdots \cap E_{i_{\sigma(p)}}$ for any permutation $\sigma \in S_p$; where S_p is the symetric group of permutations. The number of different terms to consider in Equation (2.1) is given by $k!/(k-p)!p!$ which is the number of different combinations of the index set I_p. Thus, we can establish the conditions of whether a graph G has not an intersecting cycle basis. If the graph G is not acyclic and $B_2 = \emptyset$ then $\dim G \geq 1$ so G is called a basic graph. Let $e \in E$ be an edge and define n_e as the maximum integer such that e belongs to as many as n_e edge sets E_i. In other words, $n_e = \max\{p | B_p \neq \emptyset\}$.

2.3 Splitting a graph into basic graphs

Computing edge covers for simple graphs lies on the idea of splitting a given graph G into acyclic graphs or basic graphs. It will be shown, that calculating edge covers for simple graphs can be reduced to the computation of edge covers for acyclic graphs or basic graphs thus being able to fully compute $|\mathcal{E}_G|$. The definition below describes in detail the process of splitting a graph into smaller graphs, which eventually leads to a decomposition of simple graphs into acyclic graphs or basic graphs.

Definition 1. *For a given graph $G = (V, E)$,*

1. *the split at vertex $v \in V$ is defined as the graph $G \dashv v = (V', E')$ where $V' = (V - \{v\}) \cup \bigcup_{w \in N_v} \{w'\}$ and $E' = (E - E_v) \cup \bigcup_{w \in N_v} \{ww'\}$ with $w' \notin V$,*
2. *the subdivision operation at edge $e = uv \in E$ is defined as the graph $G \perp e = (V \cup \{z\}, (E - e) \cup \{uz, zv\})$ with $z \notin V$, and z can be written either as u_v or v_u.*
3. *If $e = uv \in E$ is an edge, G/e will denote the resulting graph after performing the following operation*
$$(((G \backslash e) \perp S) \dashv u) \dashv v$$
where $S = E_u \cup E_v - \{e\}$. The subdivide operation $(G \backslash e) \perp S$ means tacitly $(\cdots((G \backslash e) \perp f) \perp g \cdots))$ where $f, g, ... \in S$.

Above definition can be easily extended to subsets $V' = \{v_1, ..., v_r\} \subseteq V$, $E' = \{e_1, ..., e_s\} \subset E$, that is $G \dashv V' = (\cdots((G \dashv v_1) \dashv v_2) \cdots) \dashv v_r$; analogously, $G \perp E' = (\cdots((G \perp e_1) \perp e_2) \cdots) \perp e_s$. It must be noted, that the order on which the operations to obtain G/e are performed is unimportant as long as one keeps track of the labels used during the process.

Example 1. Consider the star graph W_4, therefore

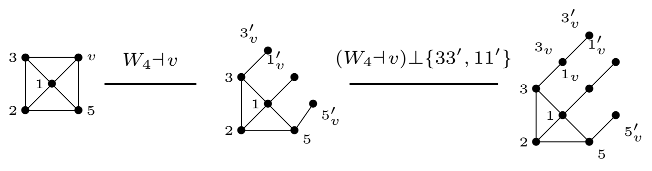

Fig. 1: The split and subdivision operation

If H, Q are graphs not necessarily edge or vertex disjoint we define a formal union of H and Q as the graph $H \sqcup Q$ by properly relabelling V_H and V_Q; this can be accomplished by defining $V_{H \sqcup Q} = \{(u,1) | u \in H\} \cup \{(u,2) | u \in Q\}$. The edge set $E_{H \sqcup Q}$ could be defined as follows: if $a, b \in V_{H \sqcup Q}$ such that $a = (u,i)$, $b = (v,j)$ for some $i, j \in \{1,2\}$ and $u, v \in V_H \cup V_Q$ then $ab \in E_{H \sqcup Q}$ if and only if $uv \in E_H \cup E_Q$ and $i = j$. Others labelling systems might work out just fine, as long as they respect the integrity of graphs H and Q. To recover the original graphs, we define the projections π_X, as $\pi_H(H \sqcup Q) = H$ and $\pi_Q(H \sqcup Q) = Q$; that is the projections π_X revert the relabelling process to the original for both graphs H and Q.

Definition 2. *Let $G = (V, E)$ be a simple graph. Let us define the split operator $\sqcup_e$ as*

(i) the graph $\sqcup_e G = G \backslash e \sqcup G/e$, that is the graph G splits into the graph $G \backslash e$ and the graph G/e if $e \in E$. If $e \notin E$ then $\sqcup_e G = \pi_G(G \backslash e \sqcup G/e) = \pi_G(G \sqcup G) = G$.
(ii) if H, Q are arbitrary graphs then $\sqcup_e(H \sqcup Q) = \sqcup_e H \sqcup \sqcup_e Q$ with $e \in E_H \cup E_Q$.

Example 2. Figure (2) shows an example of how a simple graph can be decomposed into acyclic graphs.

2.4 Edge covering sets for simple graphs

The following lemma and propositions summarizes the main properties of the split operator $\sqcup_e$, necessary for the calculation of the edge covering sets for simple graphs. The first result is regarding the dimension of the cycle spaces $\mathcal{C}_{G/e}$ and $\mathcal{C}_G$ for an edge e in the cotree of G. The proposition is rather simple in the sense that the resulting graph after applying G/e its dimension must diminishes certain amount which allow us to conclude that the operator $\sqcup$ will split the graph G into non-intersecting cyclic graph.

Proposition 21 *Let G be a simple graph, $\mathcal{B}$ a fundamental cycle base, $e = uv \in \overline{T}$ an edge in the cotree of G. If $b_e = |\mathcal{B}_u \cup \mathcal{B}_v|$ where $\mathcal{B}_u = \{C \in \mathcal{B} | u \in C\}$ and $\mathcal{B}_v = \{C \in \mathcal{B} | v \in C\}$ then*

1. *If $e \in \bar{T}$ then $b_e + \dim \mathcal{C}_{G/e} = \dim \mathcal{C}_G$.*
2. *If $e \in T$ we have that $(G/e) \backslash \langle G/e \rangle \subset \bar{T}$.*

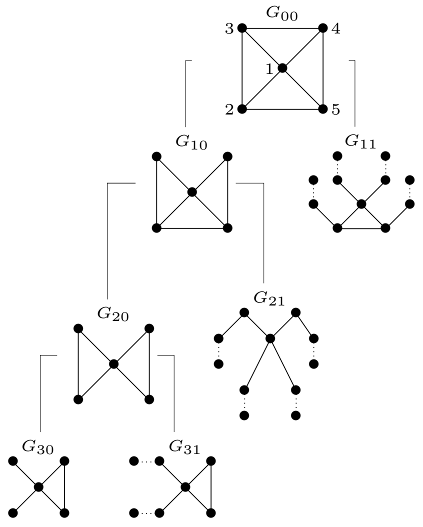

Fig. 2: Example of the splitting process provided by Definiton (2) applied to the star graph W_4 with five spokes. It clearly shows the process step by step and how the family G_{ij} of acyclic graphs is built from Definiton (2).

Proof. Every fundamental cycle containing either u or v must disappear under the operation G/e then those edges in $\overline{T}$ that do not contain u or v are the only ones that contribute to the dimension of the graph G/e, therefore $\dim \mathcal{C}_{G/e} = \dim \mathcal{C}_G - b_e$.

The proposition above allow us to explicitly calculate the number of connected components into which the graph G/e is being decomposed, on the other hand is providing a rather efficient way of testing whether or not G/e is a non-intersecting cyclic graph. The lemma below assumes that the graph in consideration is not connected, that is G has multiple conected components, thus we must consider an spanning forest for G instead of its spanning tree.

Lemma 1. *Let $G = (V, E)$ be a simple graph, F, $\overline{F}$ are an spanning forest and its corresponding coforest for G, respectively; let us choose an edge $e = uv \in E_G$ and consider the split $\sqcup_e G$ of G. If $a_e = |(E_u \cup E_v) \cap \overline{F}|$ then*

(i) If $e \in \overline{F}$ then $c(G\backslash e) = c(G)$ and $c(G/e) = c(G)+d_F(u)+d_F(v)+a_e-b_e-3$.
(ii) It holds that $\dim \mathcal{C}_{G\backslash e} = \dim \mathcal{C}_G - 1$ and $\dim \mathcal{C}_{G/e} < \dim \mathcal{C}_G$.
(iii) There exists a bijective map $\varepsilon : \mathcal{E}_{\sqcup_e G} \to \mathcal{E}_G$. That is, if S is an edge covering set for $G\backslash e$ or G/e then $\varepsilon(S)$ is an edge covering set for G, which is equivalent to $\mathcal{E}_G = \mathcal{E}_{G\backslash e} \cup \varepsilon(\mathcal{E}_{G/e})$. Thus, $|\mathcal{E}_G| = |\mathcal{E}_{G\backslash e}| + |\mathcal{E}_{G/e}|$.

The family $\mathcal{E}_{G\backslash e}$ accounts for those edge covering sets S for G on which $e \notin S$ whereas $\mathcal{E}_{G/e}$ stands for those edge covering sets where e is always a member.

Let $S \subseteq E$ be a subset of edges, from Definiton (2)(i) the split of G along S, denoted by $\sqcup_S G$, is recursively defined in terms of the sequence of splits $G_{ij} = \sqcup_{e_i} G_{(i-1)j} = G_{(i-1)j}\backslash e_i \sqcup G_{(i-1)j}/e_i$, $i \in \{1, ..., |S|\}$ in particular for $i = 1$ we define $G_{11} = G_{00}\backslash e_1 \sqcup G_{00}/e_1$ where $G_{00} = G$. By setting $\phi(i) = 2^{i-1} - 1$ and for $0 \le j \le \phi(i)$ we have therefore,

$$\begin{aligned} \sqcup_S G = G_{|S|} &= \bigsqcup_{0 \le j \le \phi(|S|)} \Big[G_{|S|j} \Big] \\ &= \bigsqcup_{0 \le j \le \phi(|S|)} \Big[G_{(|S|-1)j}\backslash e_{|S|} \sqcup G_{(|S|-1)j}/e_{|S|} \Big] \end{aligned} \tag{2.2}$$

To short up the notation, we make $G^*_{tj} = G_{(t-1)j} * e_t$, $\mathcal{E}_{tj} = \mathcal{E}_{G_{tj}}$ and $\mathcal{E}^*_{tj} = \mathcal{E}_{G_{(t-1)j}*e_t} = \mathcal{E}_{G^*_{tj}}$ with $* \in \{\backslash, /\}$; under this notation we have that $G_{tj} = G^{\backslash}_{tj} \sqcup G^{/}_{tj}$. In general, the graph G_{tj} is disconnected, if we denote by G_{tjs} its connected components then $\mathcal{E}_{tjs}$ will denote the family of edge covering sets for each graph G_{tj}. For any given spanning tree T for a simple graph G, let us make $t = |\overline{T}| = \dim \mathcal{C}_G$, $\mathcal{H}_j = \coprod_{s \in S_{tj}} \mathcal{E}_{tjs}$ for some set $S_{tj} \subseteq \mathbb{N}$, where $\coprod$ denotes the cartesian product and the projections will be denoted by π_{sj}, for every s, j. Every edge covering set of a graph G induces a subgraph; if $S \subseteq E_Q$ by definition S meets all vertices of G then the induce graph of S becomes (V_G, S). For the rest of the paper the family of edge covering sets, like $\mathcal{E}_{ijs}$ will also denote the family of induced graphs by this sets. Therefore, the calculation of edge cover for a graph G is equivalent to calculate the induced graphs by edge covering sets, since most of the operations to be performed are graph operation like vertex splitting and edge subdivision.

Theorem 1. *Let G be a finite, connected simple graph, T an spanning tree for G and $\overline{T}$ denotes its cotree and $t = |\overline{T}|$ then*

(i) the family $\{G^{\backslash}_{tj}, G^{/}_{tj}\}$, appearing in the expansion $\sqcup_{\overline{T}} G$, are all acyclic graphs or non-intersecting cycle graphs for all j satisfying $0 \le j \le \phi(t)$.

(ii) If G^*_{tjs} denotes the connected components of G^*_{tj}, then G^*_{tjs} are edge and vertex disjoint for every j, and $G^*_{tj} = \bigoplus_{s \in S_{tj}} G^*_{tjs}$ for some set of indices $S_{tj} \in \mathbb{N}$.

(iii) For every j, $0 \le j \le \phi(t)$ there exist bijections $\varepsilon_t : \bigcup_j \mathcal{E}_{tj} \longrightarrow \mathcal{E}_t$, $\varepsilon_{tj} : \mathcal{E}_{tj} \longrightarrow \mathcal{E}_{(t-1)j}$ such that $\mathcal{E}_{(t-1)j} = \mathcal{E}^{\backslash}_{(t-1)j} \cup \varepsilon_{tj}[\mathcal{E}^{/}_{(t-1)j}]$ and therefore $\mathcal{E}_t = \varepsilon_t\Big[\bigcup_j \mathcal{E}_{tj}\Big]$.

(iv) Let $H = (H^1, ..., H^{|S_t|j}) \in \mathcal{H}_j$ be a vector of graphs of the cartesian product of family $\mathcal{E}_{tjs}$ of induced graphs by edge covering sets, then $\mathcal{E}_{tj} = \bigcup_{H \in \mathcal{H}_j} [\bigoplus_{s \in S_{tj}} \pi_{js}(H)]$ and

$$\mathcal{E}_t = \varepsilon_t \Big(\bigcup_{0 \le j \le \phi(t)} \bigcup_{H_j \in \mathcal{H}_j} \Big[\bigoplus_{s \in S_{tj}} \pi_{js}(H_j) \Big] \Big).$$

For every q, $1 \leq q \leq t$, there exist bijections ε_q

$$\mathcal{E}_q \xrightarrow{\varepsilon_q} \mathcal{E}_{q-1} \xrightarrow{\varepsilon_{q-1}} \cdots \xrightarrow{\varepsilon_2} \mathcal{E}_1 \xrightarrow{\varepsilon_1} \mathcal{E}_G \tag{2.3}$$

in such a way that if $\varepsilon = \varepsilon_1 \circ \cdots \circ \varepsilon_q$ then $\mathcal{E}_G = \varepsilon(\mathcal{E}_t)$ and

$$|\mathcal{E}_G| = \sum_j |\mathcal{E}_{tj}| = \sum_j \Big[\prod_{s \in S_{tj}} |\mathcal{E}_{tjs}| \Big]. \tag{2.4}$$

3 The splitting algorithm

The algorithm for counting edge covers is based on the reduction Definiton (1). The operation of computing an spanning forest of a given graph G is denoted by $\langle G \rangle$; by abuse of notation $\langle G \rangle$ also denotes the spanning forest. Its co-forest is therefore $\bar{F} = E - \langle G \rangle$. Algorithm 1 decompose a graph into non-intersecting independent graphs (a forest).

3.1 Time Complexity of the splitting Algorithm

Let $G = (V, E)$ be the input graph of the algorithm SPLIT, $m = |E|$, $n = |V|$. It can be said that the maximum number of intersecting cycles in G is not greater than $nc = m - n$ because, at most, only one cycle is not an intersecting cycle.

The step 3 has time complexity $O(nm \log m)$. The step 6 has time complexity $O(m + n)$. The recursive application of the algorithm (steps 9 and 12) generates an enumerative tree E_G. This reduction generates two new graphs $H = (V_1, E_1)$ and $Q = (V_2, E_2)$ from G.

The time behaviour of the algorithm resides on steps 9 and 12, which corresponds with the number of intersecting cycles on the graph associated with each node of E_G. If nc denotes the maximum number of intersecting cycles in a graph associated with a node of E_G, then the time complexity of the algorithm can be expressed by the recurrence:

$$T(nc) = T(nc - 1) + T(nc - 3) \tag{3.1}$$

such recurrence ends when $nc = 1$.

This recurrence has the characteristic polynomial $p(r) = r^3 - r^2 - 1$ which has the maximum real root $r \approx 1.46557$. This leads to a worst-case upper bound of $O(r^{(m-n)} * (m + n)) \approx O(1.465571^{(m-n)} * (m + n))$. We believe that it is the first non trivial upper bound obtained for the #Edge_Covers problem.

Conclusions

Sound algorithms have been presented to compute the number of edge covers for graphs. It has been shown that if a graph G has simple topologies: paths, trees or only independent cycles appear in the graph; then the number of edge covers can be computed in polynomial time over the graph size.

Algorithm 1 Procedure that decompose a graph G into $\sqcup G$ compose of basic or acyclic graphs.

```
1: procedure SPLIT($G$) {Decomposition of $G$ into basic or acyclic graphs}
2: Input: $G = (V, E)$
3: Output: $\sqcup_S G$
4: Select an edge $e = uv \in E$ such that $e \in B_p(G) \neq \emptyset$ for some $p$. {Notice that if the
   edge $e$ exists can be found in $O(nm \log m)$.}
5: if $e$ exists then
6:     Calculate $\sqcup_e G = G \backslash e \sqcup G/e$ by applying the splitting reduction rule over $e$
       generating $H$ and $Q$
7: end if
8: if $B_2(H) \neq \emptyset$ then
9:     SPLIT($H$)
10: end if
11: if $B_2(Q) \neq \emptyset$ then
12:     SPLIT($Q$)
13: end if
14: return $\sqcup_S G$ {$S$ is the set of edges where the splitting proccess is applied. By
    Theorem (1)(iv) we have that $|\mathcal{E}_G| = \sum_j |\mathcal{E}_{tj}|$ and $|\mathcal{E}_{tj}|$ can be calculated by the
    procedures presented in [2] for basic and acyclic graphs.}
```

Regarding the cyclic graphs with intersecting cycles, a branch and bound procedure has been presented, it reduces the number of intersecting cycles until basic graphs are produced (subgraphs without intersecting cycles).

Additionally, a pair of recurrence relations have been determined that establish an upper bound on the time to compute the number of edge covers on intersecting cycle graphs. It was also designed a "low-exponential" algorithm for the #Edge_Covers problem whose upper bound is $O(1.465571^{(m-n)} * (m+n))$, m and n being the number of edges and nodes of the input graph, respectively.

References

1. Bondy J. A. and Murty U.S.R. *Graph Theory.* Springer Verlag, Graduate Texts in Mathematics, 2010.
2. De Ita G., Marcial-Romero J. Raymundo, Montes-Venegas Héctor., Estimating the relevance on Communication lines based on the number of edge covers, *Electronic Notes in Discrete Mathematics*, Vol. 36, pp. 247-254, 2010.
3. Jincheng Liu and Pinyan Lu and Chihao Zhang FPTAS for counting Weighted Edge Covers *Lectures notes in computer science*, 8737:654–665, 2014
4. Jincheng Liu and Pinyan Lu and Chihao Zhang. A Simple FPTAS for Counting Edge Covers, Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms
5. Telikepalli Kavitha, Christian Liebchen, Kurt Mehlhorn, Dimitrios Michail, Romeo Rizzi, Torsten Ueckerdt, and Katharina A. Zweig. Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, 3: 199–243, 2009.