ISSN 2007-9737

Low-Exponential Algorithm for Counting the Number of Edge Cover on Simple Graphs

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

[1] Universidad Autónoma del Estado de México, Toluca, Mexico

[2] Benemérita Universidad Autónoma de Puebla, Puebla, Mexico

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 a “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

Counting problems, although intrinsically interesting, have applications in a wide range of different areas. For instance, when estimating the probability that a given graph remains connected (graph connectivity is fundamental in network reliability theory) or when a propositional formula needs to be probabilistically tested to be true. The estimation of such probabilities can be seen as a counting problem. Counting problems also arise naturally in Artificial Intelligence Research, when some methods are used in reasoning areas, such as computing the ‘degree of belief’ when ‘Bayesian belief networks’ are used [6, 5, 17, 18].

Counting has become an important area in theoretical computer science, even though it has received less attention than decision problems. There is a handful of counting problems in graph theory that can be solved exactly in polynomial time, and indeed an important line of research is to determine low-exponential upper bounds for the time complexity of hard counting problems.

An *edge cover* of a graph G is a subset of edges covering all nodes of G. The problem of counting the number of edge cover sets of a graph, denoted as #Edge_Covers, is a #P complete problem proven via the reduction from #Twice-SAT to #Edge_Covers [4]. Most of the research on the subject has focused on approximate algorithms. In [2], an approximation algorithm for counting edge covers on 3 regular randomized graphs was presented.

More recently, in [13] a fully polynomial time approximation scheme (FPTAS) for counting edge covers of simple graphs was proposed; same authors have extended the technique to tackle the weighted edge cover problem in [14]. Additionally, the edge covering property was found interesting and relevant in various domains [7, 15, 16]. The edge cover polynomial of a graph G is presented in [1].

The edge cover problem is related to (perfect) matching, k-factor problems among others. The previous problems involve a set of edges satisfying local vertex constraints. For matching, it is at most one incident edge should be chosen compared to the edge cover problem in which at least one edge is chosen. For generic constraints, it is the Holant setting [10, 11], which has been studied for exact counting [9, 11, 8].

In this paper, we present an exact algorithm for counting edge covers taking into account that there exists a polynomial time algorithm for non-intersecting cyclic graphs [15]. Having said that, the technique is basically to reduce a simple graph into a sequence of non-intersecting cyclic graphs. The time complexity of the algorithm is also studied. Although, the complexity of our proposal remains exponential its complexity is smaller than the trivial $2^{(m-n)}$, where m and n are the

ISSN 2007-9737

number of edges and vertices respectively of the input graph.

2 Preliminaries

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 $n = |v(G)| < \infty$ and $m = |e(G)| < \infty$. A *simple cycle* in a simple graph is a set of vertices that 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 [3]. A *cycle basis* is a minimal set of simple cycles such that any cycle can be written as the sum of the cycles in the basis. A graph is said to be *non-intersecting cyclic* if any pair of simple cycles are edge disjoints.

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 will be denoted by $\mathcal{E}_G$. The problem of computing the cardinality of $\mathcal{E}_G$ is well known to be ♯P-complete problem.

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 [3]. 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 [12]. 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 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.

3 Splitting Simple 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) [3]. The process of getting a spanning tree for G by DFS algorithm will be denoted by $\langle G \rangle$. By using depth first search a spanning tree or forest $T = \langle G \rangle$ can be constructed for any graph G. The cycle basis is the set of all cycles such that each of them consists of an edge in $\bar{T} = G \backslash T$ and the simple path in T connecting its two end vertices. The dimension of $\mathcal{C}_G$ is therefore $|\bar{T}|$.

3.1 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

$$B_p = \bigcup_{i_1 \neq \cdots \neq i_p} [E_{i_1} \cap \cdots \cap E_{i_p}] \tag{3.1}$$

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 symmetric group of permutations. The number of different terms to consider in Equation (3.1) 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, $n_e = \max\{p | B_p \neq \emptyset\}$.

3.2 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 as presented in Figure 1.*

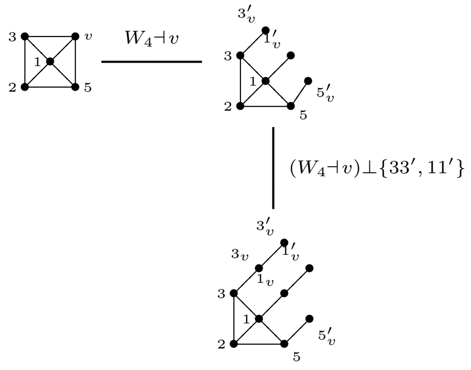

Fig. 1. The split and subdivision operation

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 labeling 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 relabeling 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 or basic graphs.*

3.3 Edge Covering Sets for Simple Graphs

The following results summarize 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

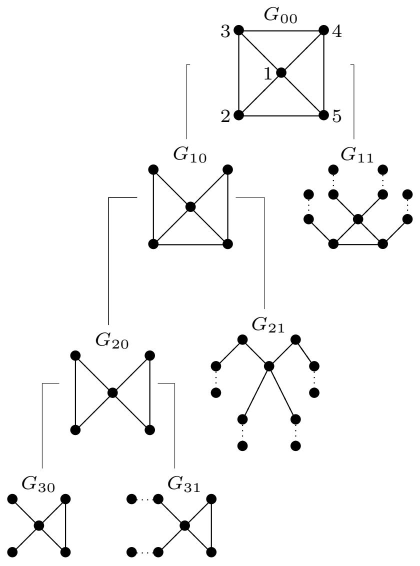

Fig. 2. Example of the splitting process provided by Definiton (2) applied to the star graph W_4 with five spokes. It clearly shows the process step by step and how the family G_{ij} of acyclic or basic graphs is built from Definiton (2)

allow us to conclude that the operator $\sqcup$ will split the graph G into non-intersecting cyclic graph.

Proposition 3.1 *Let G be a simple graph, $\mathcal{B}$ a 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 $b_e + \dim \mathcal{C}_{G/e} = \dim \mathcal{C}_G$*

Proof 1 *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 connected components, thus we must consider a spanning forest for G instead of its spanning tree.

Lemma 1 *Let $G = (V, E)$ be a simple graph, F, $\overline{F}$ are a 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, $|\mathcal{E}_G| = |\mathcal{E}_{G \backslash e}| + |\mathcal{E}_{G/e}|$.

Proof 2 *(i) For any forest F, $|E_F| = |V_F| - c(F)$. If $G = \oplus_s G_s$, where G_s are the connected components of G, a spanning forest F for G is a disjoint union $\oplus T_s$ such that T_s is an spanning tree for every component G_s. If $e \in \overline{F} = \cup_s \overline{T}_s$, where $\overline{T}_s = E_{G_s} - E_{T_s}$, there exist q such that $e \in \overline{T}_q$, $e \notin E_{G_s}$ and $G_s \backslash e = G_s$ with $s \neq q$ and therefore $c(G_q \backslash e) = c(G_q)$ which immediately implies that $c(G \backslash e) = c(G)$.*

The operation G/e on G is clearly adding vertices and edges as follows: $|V_{G/e}| = |V_G| + 2(d_G(u) + d_G(v)) - 6$ for vertices and $|E_{G/e}| = |E_G| + d_G(u) + d_G(v) - 3$ for edges. It is not difficult to check that for any graph G, $c(G) = \dim \mathcal{C}_G - |E_G| + |V_G|$, as a consequence we also have $c(G/e) = \dim \mathcal{C}_{G/e} - |E_{G/e}| + |V_{G/e}|$. Since T_q is a spanning tree of G_q, by Proposition (3.1) $\dim \mathcal{C}_{G_q/e} = \dim \mathcal{C}_{G_q} - b_e$ where $\mathcal{B}$, the basis that defines b_e, is the basis for the cycle space $\mathcal{C}_{G_q}$; clearly we also have that $d_G(u) + d_G(v) = d_F(u) + d_F(v) + a_e$ then

$$\begin{aligned} c(G/e) &= \sum_{s \neq q} \dim \mathcal{C}_{G_s/e} + \dim \mathcal{C}_{G_q/e} \\ &\quad - |E_{G/e}| + |V_{G/e}| \\ &= c(G) - b_e + d_G(u) + d_G(v) - 3 \\ &= c(G) + d_F(u) + d_F(v) \\ &\quad + a_e - b_e - 3. \end{aligned}$$

(ii) By definition of the cycle space $\mathcal{C}_{G_s}$, $\overline{T}_s$ forms a vector basis for every s, thus $\dim \mathcal{C}_{G_s} = |\overline{T}|$ and since $|E_{T_s}| = |V_{G_s}| - 1$ we have that $\dim \mathcal{C}_{G_s} = |E_{G_s}| - |V_{G_s}| + 1$. Now, if $e \in \overline{F}$ then $e \in \overline{T}_q$ for some q; thus $\dim \mathcal{C}_{G_q \backslash e} = |E_{G_q \backslash e}| - |V_{G_q \backslash e}| + 1$, but $|E_{G_q \backslash e}| = |E_{G_q}| - 1$ and $|V_{G_q \backslash e}| = |V_{G_q}|$ so we have that $\dim \mathcal{C}_{G_q \backslash e} = \dim \mathcal{C}_{G_q} - 1$. On the other hand, since $G = \oplus_s G_s$ we have that $G \backslash e =$

$[G_q\backslash e] \oplus \bigoplus_{s\neq q} G_s$ and so a sum of cycle spaces $\mathcal{C}_{G\backslash e} = \mathcal{C}_{G_q\backslash e} \oplus \bigoplus_{s\neq q} \mathcal{C}_{G_s}$ such that

$$\begin{aligned}\dim \mathcal{C}_{G\backslash e} &= \dim \mathcal{C}_{G_q\backslash e} + \sum_{s\neq q}^{c(G)} \dim \mathcal{C}_{G_s} \\ &= \dim \mathcal{C}_G - 1.\end{aligned}$$

It readily follows from Proposition (3.1) that $\dim \mathcal{C}_{G_q/e} = \dim \mathcal{C}_{G_q} - b_e$ where b_e as in Proposition (3.1) with $\mathcal{B}$ replaced by $\mathcal{B}^q$, the fundamental basis for graph G_q. Now, b_e is always different form zero since $\mathcal{B}^q_u \neq \emptyset$ and $\mathcal{B}^q_v \neq \emptyset$ because they always contain the fundamental cycle formed by the edge e. Thus,

$$\begin{aligned}\dim \mathcal{C}_{G/e} &= \dim \mathcal{C}_G - b_e \\ &< \dim \mathcal{C}_G.\end{aligned}$$

If $e \in T$ for some T in the forest then there must exists $C \in \mathcal{B}$ such that $e \in C$. The cycle C is destroyed under the operation $\sqcup_e G$ therefore $\dim \mathcal{C}_{\sqcup_e G} < \dim \mathcal{C}_G$.

(iii) *The family $\mathcal{E}_G$ can be partitioned into two disjoint subfamilies of edge covering sets, that is $\mathcal{R} = \{S \in \mathcal{E}_G | e \in S\}$ then $\mathcal{E}_G = \mathcal{R} \cup \mathcal{R}^c$. To build up the map ε we proceed as follows: $S \in \mathcal{R}^c$ if and only if $S \in \mathcal{E}_{G\backslash e}$; this is because $G\backslash e \subseteq G$ thus any $S \in \mathcal{R}^c$ must be a subset of $E_{G\backslash e}$ and vice versa. Therefore, we define $\varepsilon|_{\mathcal{R}^c} = id$, where id is the identity map. Let $N_z = \{z_i\}_{i\in I_z}$ be the set of adjacent vertices to z, I_z a set of indices of cardinality $d_{G\backslash e}(z)$ where z is either u or v. Let us define $Q_z = \{z_i z_i'\}$ and $Q_z' = \{z_i' z_i''\}$, thus any $S \in \mathcal{E}_{G/e}$ must necessarily contain the set Q_z'; if any $f \in Q_z'$ is not in S then S would not be an edge covering set because z_i'' will be an isolated vertex for some $i \in I_z$. So, $S = Q_u' \cup Q_v' \cup S'$ for some $S' \subseteq E_{G/e}$ such that $S' \cap Q_z' = \emptyset$ for $z = u, v$. Now if $R \in \mathcal{R}$ then $R = \{e\} \cup R'$ such that $e \notin R'$. Since $|E_{G/e}| = |E_{G\backslash e}| + d_{G\backslash e}(u) + d_{G\backslash e}(v)$, and $|E_{G/e} - Q_u' \cup Q_v'| = |E_{G/e}| - |Q_u'| - |Q_v'|$ which implies that $|E_{G/e} - Q_u' \cup Q_v'| = |E_{G\backslash e}|$ and therefore there exist a bijection ϕ between sets $\mathcal{P}(E_{G/e} - Q_u' \cup Q_v')$ and $\mathcal{P}(E_{G\backslash e})$ since they are both finite. In fact, ϕ can be chosen in such a way that if $Q_u' \cup Q_v' \cup S' \in \mathcal{E}_{G/e}$ then $\{e\} \cup \phi(S') \in \mathcal{R}$ and vice versa. Therefore, $\varepsilon(Q_u' \cup Q_v' \cup S') = \{e\} \cup \phi(S')$ and $\varepsilon^{-1}(\{e\} \cup R') = Q_u' \cup Q_v' \cup \phi^{-1}(R')$.*

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 $0 \leq j \leq \phi(i)$ we have therefore,

$$\begin{aligned}\sqcup_S G &= G_{|S|} = \bigsqcup_{0\leq j\leq\phi(|S|)} \Big[G_{|S|j}\Big] \\ &= \bigsqcup_{0\leq j\leq\phi(|S|)} \Big[G_{(|S|-1)j}\backslash e_{|S|} \sqcup G_{(|S|-1)j}/e_{|S|}\Big]\end{aligned} \quad (3.2)$$

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 a 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 $0 \leq j \leq \phi(t)$.*

(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, $0 \leq j \leq \phi(t)$ 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, $1 \le q \le t$, there exist bijections ε_q

$$\mathcal{E}_q \xrightarrow{\varepsilon_q} \mathcal{E}_{q-1} \xrightarrow{\varepsilon_{q-1}} \cdots \xrightarrow{\varepsilon_2} \mathcal{E}_1 \xrightarrow{\varepsilon_1} \mathcal{E}_G \qquad (3.3)$$

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

$$\begin{aligned} |\mathcal{E}_G| &= \sum_j |\mathcal{E}_{tj}| \\ &= \sum_j \Big[\prod_{s\in S_{tj}} |\mathcal{E}_{tjs}|\Big]. \end{aligned} \qquad (3.4)$$

Proof 3 *(i) Let T be a spanning tree of G, $\overline{T} = E\backslash T$ its cotree and let I be an index set of integers. Let us consider $e_i \in \overline{T}$, $u_i, v_i \in V_G$ such that $e_i = u_iv_i$; it is well known that e_i and the path of T joining u_i to v_i forms a basic cycle. Let $\mathcal{B} = \{C_i\}_{i\in I}$ be that set of basic cycles then $\mathcal{B}$ is a basis for the cycle space $\mathcal{C}_G$ where C_i is the basic cycle corresponding to edge e_i [12]. Let us define the family $\{G_{tj}\}$ as in Equation (3.2) of G_t of subgraphs such that $G_{00} = G$, $G^*_{tj} = G_{(t-1)j*e_t}$, $j \in J_i$ and $i \in I$ where $J_i = \{j|0 \le j \le \phi(i), i \in I\}$. By making $\mathcal{C}^*_{ij} = \mathcal{C}^*_{G_{ij}}$, it follows from Lemma (1), $\dim\mathcal{C}^{\backslash}_{(i)(J_i)} = \dim\mathcal{C}^{\backslash}_{(i-1)(J_i)} - 1$ and $\dim\mathcal{C}^{/}_{(i)(J_i)} < \dim\mathcal{C}^{/}_{(i-1)(J_i)}$ for all $i \in I$. Therefore at some point in the decomposition process of graph G we must have $\dim\mathcal{C}^*_{tj} = 0$ or $\dim\mathcal{C}^*_{tj} > 0$ and $B_2(G^*_{tj}) \ne \emptyset$, which means that graphs G^*_{tj} are acyclic or non-intersecting cycle graphs for all $j \in J_t$.*

(ii) It is clear from the definition of operator $\sqcup_{\overline{T}}$ that all graphs G^*_{tj} are vertex and edge disjoint. The connected components G^*_{ijs} are all subgraphs of G^*_{tj} thus $G^*_{ij} = \oplus_s G^*_{ijs}$ make sense.

(iii) It follows from Lemma (1)(iii).

(iv) It follows from Lemma (1)(iii) and (i)-(iii) of this theorem.

Algorithm 1 decomposes the input graph into basic or acyclic graphs as Definition 1 establishes.

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 $e \in B_p(G) \ne \emptyset$ for some $p > 1$. {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 $H = G\backslash e$ and $Q = G/e$
7:     if $B_2(H) \ne \emptyset$ then
8:         $A = $ SPLIT($H$)
9:     end if
10:    if $B_2(Q) \ne \emptyset$ then
11:        $B = $ SPLIT($Q$)
12:    end if
13: end if
14: return $A \sqcup B$ {the set of edges where the splitting process 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 [15] for basic and acyclic graphs.}
```

4 Time Complexity of the SPLIT Algorithm

Let $G = (V, E)$ be a simple graph, $m = |E|$, $n = |V|$. The time complexity of Algorithm 1 is given by the recursive calls over G (steps 9 and 12) which can be established by the following theorem.

Theorem 2 *Let $G = (V, E)$ be a simple connected graph with $m = |E|$, $n = |V|$ and $nc = m - n + 1$ the basic cycles of G. The recurrence which represent the complexity of Algorithm 1 is given by:*

$$T(nc) = T(nc - 1) + T(nc - 3) \qquad (4.1)$$

whose solution is ≈ 1.46557

Proof 4 *Since e is part of at least one pair of intersecting cycles, then $G \setminus e = (V_1, E_1)$ is still a connected graph. $|V_1|$=n_1=n, $|E_1| = m_1 = m - 1$. The number of base cycles in H_1 is $nc_1 = m_1 - n_1 = m - n - 1 = nc - 1$. Then, $G \setminus e$ contains at least one pair less of intersecting cycles than G.*

Let $G/e = (V_2, E_2)$, $n_2 = |V_2|$ and $m_2 = |E_2|$. By lemma 1-(i) $nc-3$. This recurrence has the characteristic polynomial $p(r) = r^3 - r^2 - 1$ which has the maximum real root $r \approx 1.46557$.

Remark 1 *Finding e such that $e = uv \in E$, $e \in B_p(G) \neq \emptyset$ for some p has complexity $O(m + n)$.*

Corollary 1 *The time complexity for splitting a simple graph G is given by:*

$$O(r^{(m-n)} * (m + n)) \approx O(1.465571^{(m-n)} * (m + n)).$$

A polynomial procedure for computing edge covers for basic graphs (acyclic or non-intersecting graphs) can be consulted at [15], so the complexity of counting edge covers is given by the splitting process.

5 Conclusions

A sound procedure has been presented to decompose a graph in order to compute the number of edge covers for the resulting subgraphs.

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). Since polynomial time procedures are known for basic graphs, the computational complexity of the edge cover problem resides on intersecting cycle graphs.

Additionally, a recurrence relation has 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. **Akbari, S. & Oboudi, M. R. (2013).** On the edge cover polynomial of a graph. *Eur. J. Comb.*, Vol. 34, No. 2, pp. 297–321.
2. **Bezáková, I. & Rummler, W. A. (2009).** Sampling edge covers in 3-regular graphs. **Královic, R. & Niwinski, D.**, editors, *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, Springer, pp. 137–148.
3. **Bondy, J. A. & Murty, U. S. R. (2008).** *Graph theory*, volume 244. Springer, New York.
4. **Bubley, R. & Dyer, M. E. (1997).** Graph orientations with no sink and an approximation for a hard case of #sat. **Saks, M. E.**, editor, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana.*, ACM/SIAM, pp. 248–257.
5. **Dahllöf, V., Jonsson, P., & Wahlström, M. (2002).** Counting satisfying assignments in 2-sat and 3-sat. **Ibarra, O. H. & Zhang, L.**, editors, *Computing and Combinatorics, 8th Annual International Conference, COCOON 2002, Singapore, August 15-17, 2002, Proceedings*, volume 2387 of *Lecture Notes in Computer Science*, Springer, pp. 535–543.
6. **Darwiche, A. (2000).** On the tractable counting of theory models and its application to belief revision and truth maintenance. *CoRR*, Vol. cs.AI/0003044.
7. **Grohe, M. & Marx, D. (2006).** Constraint solving via fractional edge covers. *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, ACM Press, pp. 289–298.
8. **Huang, S. & Lu, P. (2012).** A dichotomy for real weighted holant problems. *IEEE Conference on Computational Complexity*, pp. 96–106.
9. **Jin-Yi Cai, H. G. & Williams, T. (2013).** A complete dichotomy rises from the capture of vanishing signatures: extended abstract. *STOC*, pp. 635–644.
10. **Jin-YI Cai, P. L. & Xia, M. (2009).** Holant problems and counting csp. *Proceeding of the 41st annual ACM symposium on THeory of computing, STOC 09*, ACM, New York, USA, pp. 715–724.
11. **Jin-YI Cai, P. L. & Xia, M. (2011).** Computational complexity of holant problems. *SIAM J. Comput.*, Vol. 40, No. 4, pp. 1101–1132.
12. **Kavitha, T., Liebchen, C., Mehlhorn, K., Michail, D., Rizzi, R., Ueckerdt, T., & Zweig, K. A. (2009).** Cycle bases in graphs characterization, algorithms, complexity, and applications. *Computer Science Review*, Vol. 3, No. 4, pp. 199–243.
13. **Lin, C., Liu, J., & Lu, P. (2013).** A simple FPTAS for counting edge covers. *CoRR*, Vol. abs/1309.6115.
14. **Liu, J., Lu, P., & Zhang, C. (2014).** FPTAS for counting weighted edge covers. **Schulz, A. S. & Wagner, D.**, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737

ISSN 2007-9737

of *Lecture Notes in Computer Science*, Springer, pp. 654–665.

15. **Luna, G. D. I., Marcial-Romero, J. R., & Venegas, H. A. M. (2010).** Estimating the relevance on communication lines based on the number of edge covers. *Electronic Notes in Discrete Mathematics*, Vol. 36, pp. 247–254.
16. **Padó, S. & Lapata, M. (2006).** Optimal constituent alignment with edge covers for semantic projection. **Calzolari, N., Cardie, C., & Isabelle, P.**, editors, *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, Sydney, Australia, 17-21 July 2006*, The Association for Computer Linguistics, pp. 1161–1168.
17. **Roth, D. (1996).** On the hardness of approximate reasoning. *Artif. Intell.*, Vol. 82, No. 1-2, pp. 273–302.
18. **Vadhan, S. P. (2001).** The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, Vol. 31, No. 2, pp. 398–427.

Article received on 14/05/2015; accepted on 15/09/2015. Corresponding author is José A. Hernández-Servín.