Increasing the Reliability of a Network via the Number of Edge Covers

Guillermo De Ita, Yolanda Moyao, Meliza Contreras, Pedro Bello

Faculty of Computer Science, Universidad Autónoma de Puebla
{deita,ymoyao,mcontreras,pbello}@cs.buap.mx

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Counting the number of edge covers on graphs, problem denoted as #Edge_Covers, is a #P-complete problem. Knowing the number of edge covers is useful for estimating the relevance of the lines in a communication network, which is an important measure in the reliability analysis of a network. In this paper, we propose a method for increasing the degree of reliability of a network for adding strategely new lines and then to increase the density of edge cover sets of the network. Regarding to the most common physical topologies of a network; Bus, Stars, Trees and Rings, we present efficient algorithms for solving the #Edge_Covers problem for these topologies.

Keywords: Edge Covers problem; communication network; reliability analysis;

1. Introduction

A computer network can be modeled via an undirected graph $G = (V, E)$ where the vertices of the graph V represent sites and the edges of the graph E stand for the links between the sites. There are many types of networks varying in their performance, definitions and therefore with different concepts of reliability [3].

In practice each site or link can fail, if an element (either node or edge) fails, we say that it is *down*, otherwise we say that it is *up*. The problem of checking the connectivity of the network is known to be NP-hard [1, 7]. In our work, we assume that sites (vertices) are perfect, but links may independently fail with similar known probabilities.

We show how to calculate the number of edge cover sets of the network in order to determine a degree of reliability of the network as well as for computing the relevance of the lines which is an important estimation of the 'strategic' value of each line in a network.

Given an initial network $N = (V, E)$ as an undirected Graph, we define the reliability of N in terms of the number of edge covers that N has. And we present, how to estimate the reliability of N when new communication lines are aggregated to N. To add a new line to a network N allow us to increase the *connectivity density* of the original network. Then, our procedure permit us to search for the best two nodes for adding a new line to N and maximizes so the *connectivity density* of N.

Our proposal is based on counting the number of edge covers of a graph (the #Edge_Covers problem) [5, 6], and for solving this late problem, we have developed a novel method for counting edge covers of a graph for the most common topologies of a network; bus, stars, trees and rings. We show that the #Edge_Covers problem is solved in linear time on the size (number of nodes plus number of edges) of any graph without intersecting cycles (any pair of cycles with common edges).

1.1. Preliminaries

We model a communication network by an undirected graph $G = (V, E)$ (i.e. finite, loopless and without parallel edges), where V represents the set of vertices (also known as nodes) and E represents the set of edges. Sometimes $V(G)$ and $E(G)$ are utilized to emphasize the graph G. A vertex and an incident edge are said to *cover* each other.

The neighborhood of a vertex $v \in V$ is the set $N(v) = \{w \in V : \{v, w\} \in E\}$ and its degree, denoted as $\delta(v)$, is the number of neighbors that it has. The cardinality of a set A will be denoted as $|A|$. Thus, $\delta(v) = |N(v)|$, and we say that a vertex v is *pendant* if its neighborhood contains only one vertex; an edge e is *pendant* if one of its endpoints is a pendant vertex [2]. The degree of the graph G is $\Delta(G) = max\{\delta(x) : x \in V\}$.

Given a graph $G = (V, E)$, $S = (V', E')$ is a subgraph of G if $V' \subseteq V$ and E' contains edges $\{v, w\} \in E$ such that $v, w \in V'$. If E' contains every edge $\{v, w\} \in E$ where $v, w \in V'$ then S is called the *subgraph of G induced by S* and is denoted by $G \| S$. $G - S$ denotes the graph $G \| (V - V')$. $G - v$ for any $v \in V$ denotes the induced subgraph $G \| (V - \{v\})$, and $G - e$ for any $e \in E$ denotes the subgraph of G with the set of nodes V and set of edges $E - \{e\}$.

A *connected component* of G is a maximal induced subgraph of G, that is, a connected component is not a proper subgraph of any other connected subgraph of G. Notice that, in a connected component, for every pair of its vertices u, v, there is a path from u to v. A tree graph is an acyclic connected graph.

An edge cover for a graph $G = (V, E)$ is a subset of edges $\mathcal{E} \subseteq E$ that covers all nodes of G, that is, for each $u \in V$ there is a $v \in V$ such that $e = \{u, v\} \in \mathcal{E}$. Let $C\mathcal{E}(G) = \{\mathcal{E} \subseteq E : \mathcal{E}$ is an edge cover of $G\}$ be the set of edge covers that G has. Let $NE(G) = |C\mathcal{E}(G)|$ be the number of edge covers of G. Computing the number $NE(G)$ for any input graph is known as the #Edge_Covers problem.

The #Twice-SAT problem consists in counting the number of models of a Boolean formula F where variable of F appears at most two times (with any one of its two signs) in F. #Edge_Covers is a #P-complete problem which has been proved via the reduction from #Twice-SAT to #Edge_Covers [2, 8].

2. Estimating the Relevance on Communication Lines

Complex networks, modeled as large graphs, have received much attention during the last years. However, topological information on these networks is only

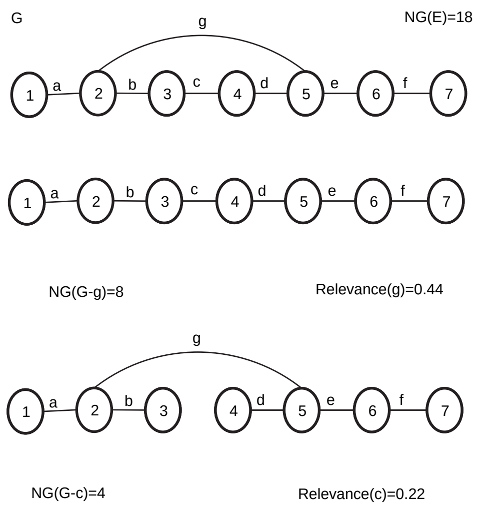


Fig. 1. Estimating the relevance of lines in a network

available through intricate measurement procedures [1]. Nodes and edges of a network could fail (become non-operational). If an element (either node or edge) fails, we say that it is *down*, otherwise we say it is *up*.

If we assume that the communication lines (edges) in a network G have the same failure probability and those failures are independent from each other, whenever an edge $e \in G$ fails we can estimate different classes of reliability of the network. For example, the 'relevance' of a line e in a network G can be estimated by the conditional probability $P_{e/G}$ which can be approximated by the fraction of the number of edge covers which are substracted when the edge e is removed (fails), that is, $P_{e/G} = 1 - \frac{NE(G-e)}{NE(G)}$.

Thus, $P_{e/G}$ gives us the strategic value of an edge e in a network G, i.e. for greater values of $P_{e/G}$ the line e is most relevant than the other lines in G, in order to maintain the connectivity of G in case of failures. $P_{e/G}$ is an proportion which indicates the density of edge covers of G with respect to the line e.

In general, as e is any edge of G then $P_{e/G}$ could be used for estimating the relevance of any edge of G, such as it is shown in Table 1. For example in figure 1, we show the effect over the number of edge covers of the network G when lines g or c of the network G fail. Thus, we are estimating the 'density' of each edge for covering all node of G when some edges fail.

In the table 1, we assume that one of the lines of the set $\{b, c, d, e, f\}$ down, and the the effect of that fails over the reliability of the network of the figure 1. So, in the table 1 we show the relevance of each one of those lines.

Given this measure of reliability and given a network $N = (V, E)$, a relevant practical problem is to determine two non-adjacent nodes in V for connecting such nodes via a new communication line for N and then to increase the density

Table 1. Relevance of each edge of G

Edges	$NE(G-e)$	$P_{e/G}$
g	8	.44
c	4	.22
b	7	.61
d	5	.72
e	7	.61

of edge covers of N. How to select such two nodes, is the objective of the following section.

2.1. Increasing the Density of the Edge Covers of a Networks

Assuming a network $N = (V, E)$, $n = |V|$ and $m = |E|$, there are $n*(n-1)-|E|$ options for selecting two non-adjacent nodes of N. If we know the value $NE(N)$ and adding to E a new line e then the value $NE(N \cup e)$ is a measure about how increase the number of edge covers in N.

We want to select the two non-adjacent nodes such that the edge e between such node permit us to maximize the value $NE(N \cup e)$ into the set of all possible new edges of N. And for this, we need first how to compute $NE(N \cup e)$ for any new edge e.

Let $e = \{u, v\} \notin E$ and $u, v \in V$. e is a new possible edge for N and we want to know the difference $NE(N \cup e) - NE(N)$. When an edge cover $\mathcal{E}$ of G is being built, we distinguish between two different states of a node u; we say that u is *free* when it has not still been covered by any edge of $\mathcal{E}$, otherwise the node is *covered*.

In order to compute $NE(N \cup e)$ we conform a new network $N_1 = (V_1, E_1)$ obtained from $N = (V, E)$ in the following way: $V_1 = V - \{u, v\}$ plus new virtual covered node: u_x for each edge in $E - e$ type $\{x, u\}$, and new virtual covered node: v_x for each incident edge type $\{x, v\}$. When we compute $NE(N)$ we will consider that the new virtual nodes u_x and v_x are covered nodes, that means, any adyacent edge to them could (or not) appear into the set of edge covers of N.

In order to form E_1, the edge e is deleted from $E(N)$ and for each edge type $e_j = \{x, u\} \in E(N)$ a new pendant edge $e'_j = \{x, u_j\}$ appears in E_1, or when $e_j = \{x, v\} \in E(N)$ it changes in E_1 as $e'_j = \{x, v_j\}$, u_j and v_j being one of the new covered nodes. Such edges e'_jś are pendant edges in N_1 as well as the new covered nodes u_x and v_x are pendant nodes, see figure 2. Furthermore, we can show that $NE(N \cup e) = NE(N_1)$.

Notice that in this case, all cycle of N containing any one of the original nodes u and v is not more a cycle in N_1. Notice also that the main problem here, consists in count the number of edge cover sets for different class of physical topologies networks. In the following chapter, we present linear time exact procedures for computing the number of edge cover sets of a network, for the most common physical topologies [9]. The degree of the graph is irrelevant in our methods.

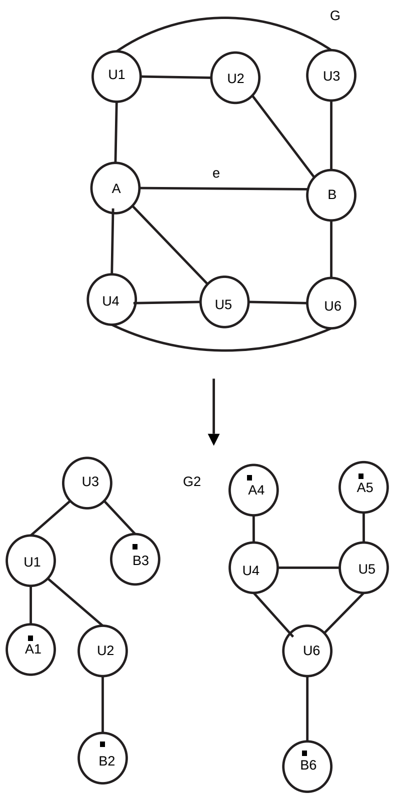


Fig. 2. Estimating the number of edge covers for adding a new line (e)

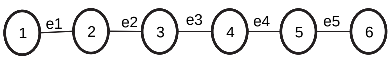
3. Linear time Procedures for Counting Edge Covers

$NE(G)$ for any graph G, including the case when G is a disconnected graph, is computed as: $NE(G) = \prod_{i=1}^{k} NE(G_i)$, where k is the cardinality of the set of connected components of G and each G_i represents an element of this set. The set of connected components of G can be computed in linear time [1].

The edges of G appearing in all edge cover sets are called *fixed edges*. We start designing procedures for counting edge covers, considering the most common topologies of a network.

3.1. Case A: The Bus Topology

Let $P_n = G = (V, E)$ be a linear bus (a path graph). We assume an order between vertices and edges in P_n, i.e. let $V = \{v_0, v_1, \ldots, v_n\}$ be the set of $n+1$ vertices and let $e_i = \{v_{i-1}, v_i\}$, $1 \leq i \leq n$ be the n edges of P_n.



$$\begin{array}{ccccc} e_1 & e_2 & e_3 & e_4 & e_5 \\ (1,0) \rightarrow & (1,1) \rightarrow & (2,1) \rightarrow & (3,2) \mapsto & (5,0) = 5 \end{array}$$

Fig. 3. Counting edge covers on a bus

Let $G_i = (V_i, E_i)$, $i = 0, \ldots, n$ be the subgraphs induced by the first i nodes of V, i.e. $G_0 = (\{v_0\}, \emptyset), G_1 = (\{v_0, v_1\}, \{e_1\}), \ldots, G_n = P_n$. $G_i, i = 0, \ldots, n$ is the family of induced subgraphs of G formed by the first i nodes of V. Let $\mathcal{CE}(G_i)$ be the set of edge covers of each subgraph G_i, $i = 0, \ldots, n$.

Each edge $e_i, i = 1, \ldots, n$ in the bus has associated an ordered pair (α_i, β_i) of integer numbers where α_i carries the number of edge cover sets of $\mathcal{CE}(G_i)$ where the edge e_i appears in order to cover the node v_{i-1}, while β_i conveys the number of edge cover sets in $\mathcal{CE}(G_i)$ where the edge e_i does not appear.

By traversing P_n in depth-first search [1], each pair $(\alpha_i, \beta_i), i = 1, \ldots, n$ is computed in accordance with the type of edge that e_i is. P_n has two fixed edges: e_1 and e_n. The pair (1,0) is assigned to (α_1, β_1) because e_1 has to appear in all edge cover of P_n.

If we know the values $(\alpha_{i-1}, \beta_{i-1})$ for any $0 < i < n$, then we know the number of times where the edge e_{i-1} appears or does not appear into the set of edge covers of G_i. When the edge e_i is being visited, the vertex v_{i-1} has to be covered considering its two incident edges: e_{i-1} and e_i. Any edge cover of $\mathcal{CE}(G_i)$ containing the edge e_{i-1} (α_{i-1} cases) has already covered v_{i-1} then the ocurrence of e_i is optional. But for the edge covers where e_{i-1} does not appear (β_{i-1} cases) e_i must appear in order to cover v_{i-1}. This simple analysis shows that the number of edge covers where e_i appears is $\alpha_{i-1} + \beta_{i-1}$ and that just in α_{i-1} edge covers the edge e_i does not appear. Thus, we compute (α_i, β_i) associated with the edge e_i, applying the Fibonacci recurrence relation.

$$\alpha_i = \alpha_{i-1} + \beta_{i-1}; \quad \beta_i = \alpha_{i-1} \tag{1}$$

When the search arrives to the last edge e_n of the linear bus, we have already computed the pair $(\alpha_{n-1}, \beta_{n-1})$; since e_n is a fixed edge, it has to appear in all edge covers of P_n. We call $\alpha_n = \alpha_{n-1} + \beta_{n-1}$ and $\beta_n = 0$ *the recurrence for processing fixed edges* (RPFE).

The pair associated with e_n is $(\alpha_n, \beta_n) = (\alpha_{n-1} + \beta_{n-1}, 0)$. The sum of the elements of this pair (α_n, β_n) yields the number of edge covers: $NE(P_n) = \alpha_n + \beta_n$. Notice that $NE(P_n)$ is computed in linear time over the number of edges in P_n. In figure 3 we present an example where $\rightarrow$ denotes the application of recurrence (1), and $\mapsto$ denotes the application of RPFE.

Recall that each Fibonacci number F_i can be bounded from above and from below by $\phi^{i-2} \geq F_i \geq \phi^{i-1}, i \geq 1$, where $\phi = \frac{1}{2} \cdot (1 + \sqrt(5))$.

Theorem 1 *The number of edge cover sets of a path of* n *edges, is:* $F_n = \texttt{ClosestInteger}\left[\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\right]$.

proof: The series $(\alpha_i, \beta_i), i = 1, \ldots, n$ used for computing $NE(P_n)$, coincides with the Fibonacci numbers: $(F_1, F_0) \rightarrow (F_2, F_1) \rightarrow (F_3, F_2) \rightarrow \ldots \rightarrow (F_{n-1}, F_{n-2}) \mapsto (F_n, 0)$. Then, we infer that $(\alpha_i, \beta_i) = (F_i, F_{i-1})$ for $i = 1, \ldots, n-1$ and $\alpha_n = F_n, \beta_n = 0$. Thus, $NE(P_n) = \alpha_n + \beta_n = F_n$.

3.2. Case B: The Tree Topology

Let $T = (V, E)$ be a rooted tree. *Root-edges* in T are the edges with one endpoint in the root node; *leaf-edges* in T are the edges with one endpoint in a leaf node of T.

Given any intermediate node $v \in V$, we call a *child-edge* of v to the edge connecting v with any of its children nodes, and the edge connecting v with its father node is called the *father-edge* of v. $NE(T)$ is computed by traversing T in post-oder and associating (α_e, β_e) with each edge $e \in E$, except for the leaf edges.

Theorem 2 *If T is a tree without covered nodes and T' is the leaves-pruned tree of T with leaves labelled as covered nodes then $NE(T) = NE(T')$.*

proof: Suppose the tree T has leaf-edges $e_1, e_2, \ldots, e_k$. Since each leaf-edge covers a leaf-node of T and there is not other edge which covers such a leafn-ode then, every leaf-edge of T should be included in each set cover for T. Let $CE(T) = \{A_0 \cup \{e_1, e_2, \ldots, e_k\}, A_1 \cup \{e_1, e_2, \ldots, e_k\}, ..., A_n \cup \{e_1, e_2, \ldots, e_k\}\}$ be the set of edge covers for T where neither of $A_0, A_1, \ldots, A_n$ contains elements of the set $\{e_1, e_2, \ldots, e_k\}$. By removing from each element of the set $CE(T)$ the following set $e_1, e_2, \ldots, e_k$ we end up with $L = \{A_0, A_1, \ldots, A_n\}$, whose cardinality obviously coincides with the cardinality of $CE(T)$, that is $|CE(T)| = |L|$. It will be shown that $|L| = |CE(T')|$. It is obvious that each element of L is an edge cover for T' since each element of L covers all the vertexes of T, except the leaves. Now, suppose that A_i is a cover for T' then $A_i \cup \{e_1, e_2, ..., e_k\}$ should cover T for every $i = 1, \ldots, k$. Hence $|CE(T')| = |CE(T)| = |L|$ or $NE(T) = NE(T')$, which concludes the proof.

Procedure for computing #Edge Covers for Trees(T)

1. Reduce the input tree T to other tree T' by prunning all leaf nodes and leaf-edges from T, and by labeling as *covered* all father nodes of the original leaf nodes of T (see figure 4).
2. Traverse T' in post-order and associate a pair (α_e, β_e) with each edge e in T'. Such pairs are computed in the following way:
   (a) $(\alpha_e, \beta_e) = (1, 1)$ if e is a leaf-edge of T', since its children nodes have been covered.
   (b) if an internal node v is being visited and it has a set of child-edges, e.g. $u_1, u_2, ..., u_k$, as we have already visited all child-edges of v, then each pair $(\alpha_{u_j}, \beta_{u_j})$, $j = 1, \ldots, k$ has been computed. Assume α_u carries the number of different combinations of the child-edges of v for covering v,

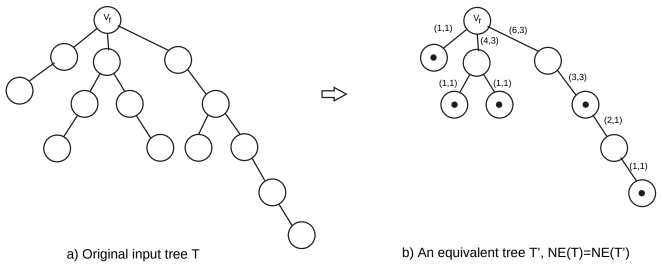


Fig. 4. Computing the number of edge covers for a tree

while β_u gives the number of combinations among the child-edges of v which do not cover v. The pair (α_u, β_u), which we assume represents an imaginary child-edge e_u of v, is computed as:

$$\alpha_u = \prod_{j=1}^{k}(\alpha_{u_j} + \beta_{u_j}) - \prod_{j=1}^{k}\beta_{u_j} \ ; \quad \beta_u = \prod_{j=1}^{k}\beta_{u_j} \qquad (2)$$

The pair associated to the father-edge e_v of v is computed as:
$(\alpha_v, \beta_v) =$
$$\begin{cases} (\alpha_u + \beta_u, \alpha_u) & \text{if } v \text{ is a free node or,} \\ (\alpha_u + \beta_u, \alpha_u + \beta_u) & \text{if } v \text{ is a covered node} \end{cases}$$
This step is iterated until it computes the pairs (α_e, β_e) for all edge $e \in T'$. If there are more than one root-edges then one iteration more of this step is applied in order to obtain a final pair $(\alpha_{e_r}, \beta_{e_r})$ associated with just one root-edge e_r.

3. $NE(T)$ is computed in accordance with the status of the root node v_r of T; $NE(T) = \alpha_{e_r} + \beta_{e_r}$ if v_r is a covered node, otherwise $NE(T) = \alpha_{e_r}$.

The above procedure returns $NE(T)$ in time $O(n + m)$ which is the necessary time for traversing T in post-order. Notice that this case includes the star topology network.

Example 1 *Let T be the tree of figure 4a. T' is the reduced tree from T where its covered nodes are marked by a black point inside of the nodes (figure 4b). When T' is traversed in post-order a pair (α_e, β_e) is associated with each edge. The pairs for the child-edges of v_r, are: (1,1), (4,3) and (6,3). Those three edges are combined in only one edge e_r applying recurrence (2): $\alpha_{e_r} = (1 + 1) * (4 + 3) * (6 + 3) - 1 * 3 * 3 = 117$ and $\beta_{e_r} = 1 * 3 * 3 = 9$. Since v_r is the root node and it is free, then $NE(T) = \alpha_{e_r} = 117$*

3.3. Case C: The Ring Topology

Let $C_n = (V, E)$ be a simple ring with n edges. We assume an order over the nodes and edges of C_n given by $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_n\}, e_i =$

$\{v_i, v_{i+1}\}, i = 1, \ldots, n-1,\ e_n = \{v_n, v_1\}$. We call a *computing thread* or just a *thread* to the series $(\alpha_1, \beta_1) \rightarrow (\alpha_2, \beta_2) \rightarrow \cdots \rightarrow (\alpha_k, \beta_k)$ obtained for counting in incremental way, applying the recurrence (1), the number of edge covers of a path with k edges.

Let L_p be the thread used for computing the series of pairs associated to the n edges of C_n. $(\alpha_1, \beta_1) = (1, 1)$ is associated with e_1 since C_n has not fixed edges. Traversing in depth first search, the new pairs in L_p are computed applying the Fibonacci recurrence (1) since all nodes in C_n have degree two and they are free. After n applications of recurrence (1), the pair $(\alpha_n, \beta_n) = (F_{n+1}, F_n)$ is obtained, F_i being the i-th Fibonacci number.

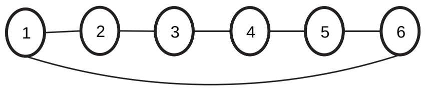
Let NC_n be the number of edge sets counted by L_p, i.e. $NC_n = \alpha_n + \beta_n = F_{n+2}$. L_p counted the edge sets where neither e_1 nor e_n appear, since $\beta_1 = 1$ and $\beta_n > 0$. Due to e_1 or e_n or both have to be included in the edge cover sets of C_n in order to cover v_1, we have to substract from NC_n the number of sets which not cover v_1.

Let Y be the number of edge sets which cover all nodes of C_n except v_1, then $NE(C_n) = NC_n - Y$. In order to compute Y a new thread $L'_p = (\alpha'_1, \beta'_1) \rightarrow \cdots \rightarrow (\alpha'_n, \beta'_n)$ is computed. L'_p begins with the pair $(\alpha'_1, \beta'_1) = (0, 1)$, i.e. it begins counting the edge sets where e_1 does not appear. After n applications of recurrence (1) the last pair (α'_n, β'_n) of L'_p obtains (F_{n-1}, F_{n-2}).

The number of edge sets where neither e_1 nor e_n appear is $\beta'_n = F_{n-2}$, hence $Y = F_{n-2}$. Finally, $NE(C_n) = NC_n - Y = F_{n+2} - F_{n-2}$. Then, we deduce the following theorem.

Theorem 3 *The number of edge cover sets of a simple cycle C_n with n edges, expressed in terms of Fibonacci numbers, is: $NE(C_n) = F_{n+2} - F_{n-2}$.*

With ' $\curvearrowleft$ ' we denote the binary operation $(\alpha_n, \beta_n - \beta'_n)$ between two pairs, and the result is assocciated with the last edge e_n of the ring C_n (fig. 4). Notice that the computation of $NE(C_n)$ is the order $O(n)$ since we compute the two threads: Lp and L'_p in parallel while the depth-first search is applied.



$$\begin{array}{rllllll}
 & (\alpha_1, \beta_1) & \rightarrow (\alpha_2, \beta_2) & \rightarrow (\alpha_3, \beta_3) & \rightarrow (\alpha_4, \beta_4) & \rightarrow (\alpha_5, \beta_5) & \rightarrow (\alpha_6, \beta_6) \\
Lp: & (1,1) & \rightarrow (2,1) & \rightarrow (3,2) & \rightarrow (5,3) & \rightarrow (8,5) & \rightarrow (13,8) \\
Lp': & (0,1) & \rightarrow (1,0) & \rightarrow (1,1) & \rightarrow (2,1) & \rightarrow (3,2) & \rightarrow (5,3) \\
 & & & \Rightarrow (13,8) \curvearrowleft (5,3) = (13,5) & & &
\end{array}$$

Fig. 5. Counting the number of edges covers for a ring

Example 2 *Let C_6 be the ring illustrated in figure 5. Applying the above theorem, we have that $NE(C_6) = F_{6+2} - F_{6-2} = F_8 - F_4 = 21 - 3 = 18$.*

The graphs which hold the topologies of the above cases (A) to (C) englobe the most common topologies of a communication network. The linear time procedures designed here can be included into a branch and bound algorithm which processes any kind of topology of a network [4].

4. Conclusion

We have determined different recurrence relations for counting the number of edge cover sets for different physical topologies of a network. If the topology of a network G is a bus, a star, a tree or a ring, then the number of edge covers of G can be computed in linear time on the size (number of nodes plus number of edges) of the network.

Knowing the number of edge covers of a network is helpful for estimating its reliability. For example, we have shown how to estimate the 'relevance' of any line e of the network based on the proportion of the number of edge covers where e does not appear with respect to the total number of edge covers in the network. Such proportion is an indicative of the density of edge covers of the newtwork with respect to the line e.

Furthermore, we present a method for selecting the best two non adjacent nodes on the network which allow us maximize the density of edge covers sets, adding a new line connecting such nodes in the network.

References

1. D. J. M. Garey., Computers and Intractability a Guide to the Theory of NP-Completeness, (1979).
2. M. D. R. Bubley., Graph orientations with no sink and an approximation for a hard case of #sat., *Proc. of the Eight Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 248–257, 1997.
3. Y. Shpungin., Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges., *PInt. Jour. of Computer Science Vol. 1*, 1(3):177–191, 2006.
4. R. Tarjan., Depth-First Search and Linear Graph Algorithms., *SIAM Journal on Computing*, 1:146–160, 1972.
5. S. P. Vadhan., The complexity of counting in sparse, regular, and planar graphs., *SIAM Journal on Computing*, 31(2):398–427, 2001.
6. E. M. V.E. Levit., The independence polynomial of a graph - a survey., *Holon Academic Inst. of Technology*, 2005.
7. B. Russ.,*Randomized Algorithms: Approximation, Generation, and Counting, Distinguished dissertations.*, 2001.
8. L. Kou, C. Stockmeyer, C. Wong., Covering edges by cliques with regard to keyword conflicts and intersection graphs., *Communications of the ACM*, pages 136–139, 1978.
9. G. De Ita, P. Bello, M. Contreras., Applying Fibonacci Recurrences for Counting Combinatorial Objects in Graphs., *Advances in Computer Science and Engineering Vol. 34*, :3–14, 2008.