Modeling Work Teams Trough Signed Graphs

Guillermo De Ita Luna, Yolanda Moyao Martínez, Luis Carlos Altamirano Robles

Faculty of Computer Sciences, Universidad Autónoma de Puebla
deita@cs.buap.mx, *ymoyao@cs.buap.mx*, *altamirano@cs.buap.mx*

Abstract. We present a method to count the different ways to form work teams in large corporate companies for projects with specific requirements. The teams are formed by employees and according to the ability of the employees for satisfying the requirements of the project. We model this problem through signed graphs, where each node represents an employee and an edge defines a restriction between the employees. The restrictions of a project is translated as a Boolean Formula Σ in two conjunctive form $(2-CF)$.
The logical one values in a satisfied assignment of Σ determine what employees can be part of an adequate team, while the logical zero values in a satisfied assignments of Σ indicate the employees that they have not to be part of the team.
In this model, $SAT(\Sigma)$ contains the set of different teams that can be formed to develop effectively the project. And $\#SAT(\Sigma)$ will show us how many different teams can be formed to develop the project. In order to compute $\#SAT(\Sigma)$, a signed extended graph is formed and recurrence equations are applied. We show for what class of signed graphs the computation of $\#SAT(\Sigma)$ can be done in polynomial time.
Keywords: Signed graphs, #SAT Problems, Counting Models.

1. Introduction

Signed graphs (graphs whose edges are designated positive or negative). This class of graphs have been very useful for modeling different class of problems in the social sciences [3, 5, 2]. For example, signed graphs are used for modeling the interaction among a group of persons and the type of relationship between certain pair of individuals of the group.

We are interested in computing the different work teams that can be formed to develop certain kind of projects in large corporative companies. This problem can be modeled through signed graphs, considering each node of the graph as an employee and forming edges to join nodes that represent restriction between those employees. A team's leader of the project defines the restrictions of the project, and such constraints form a Boolean Formula Σ in $2-CF$ (conjunction of binary or unary clauses).

Therefore, if we can find $SAT(\Sigma)$ (the satisfy assignments of Σ), we will obtain the adequate teams to develop the project effectively. And $\#\mathrm{SAT}(\Sigma)$

Advances in Computer Science and Engineering
Research in Computing Science 63, 2013 pp. 139–150
Paper Recived 22-09-2013 and Acepted 21-10-2013

(the number of models in Σ) shows how many different teams can be formed to develop the project.

The SAT problem is a classic NP-complete problem [1], while #SAT is relevant in the following issues: for estimating the degree of reliability in a communication network, for computing degree of belief in propositional theories, for the generation of explanations to propositional queries, in Bayesian inference, in a truth maintenance systems and for repairing inconsistent databases [6, 8]. Those previous problems come from several AI applications such as planning, expert systems, reasoning, etc.

$\#SAT$ is as difficult as the SAT problem, but even when SAT can be solved in polynomial time, it is not known an efficient computational method for $\#SAT$. For example, the 2-SAT problem (SAT limited to consider (≤ 2)-CF's) can be solved in linear time. However, the corresponding counting problem #2-SAT is #P-complete.

The aim is to develop a method to count the different teams that can be formed to develop different projects through to count the number of models in the Boolean formula associated to the requirements of the project.

This kind of automatization, will help the companies to save money and time, by making the team work selection process more efficient and dynamic.

2. Notation and Preliminaries

A signed graph Γ (also called sigraph) is an ordered pair $\Gamma = (G, \sigma)$ where $G = (V, E)$ is a graph called the underlying graph of Γ and $\sigma : E \to \{+, -\}$ is a function called a *signature* or *signing*. $E^+(\Gamma)$ denotes the set of edges from E that are mapped by σ to '+', and $E^-(\Gamma)$ denotes the set of edges from E that are mapped by σ to '-'.

The elements of $E^+(\Gamma)$ are called positive edges and those of $E^-(\Gamma)$ are called negative edges of Γ. A signed graph is *all-positive* (respectively, *all-negative*) if all of its edges are positive (negative); further, it is said to be *homogeneous* if it is either all-positive or all-negative, and *heterogeneous* otherwise.

Signed graphs have been very useful for modeling interactions among a group of persons and for representing the type of relationship between certain pair of individuals of the group. Special focus is on the social inequalities. What is it about such characteristics as sex, race, occupation, education, and so on that leads to inequalities in social interactions? [2].

Given a set of de n Boolean variables, $X = \{x_1, x_2, ..., x_n\}$. It's called a *literal* to any variable x or the negation $\overline{x}$ of it. We use $v(l)$ to indicate the variable involved by the literal l.

The disjunction of different literals is called a *clause*. For $k \in \mathbb{N}$, a $k-clause$ is a *clause* consisting of exactly k *literals*. A variable $x \in X$ appears in a clause c if x or $\overline{x}$ is an element of c. Let $v(c) = \{x \in X : x \; appears \; in \; c\}$. A conjunctive form ($CF$) is a conjunction of *clauses*. A $k - CF$ is a CF containing only $k - clausulas$ and, $(\leq k) - CF$ denotes a CF containing *clauses* with at most k literals.

Let Σ be a 2-CF, then an assignment s for Σ is a function $s : v(\Sigma) \to \{0, 1\}$. An assignment can also be considered as a set of non-complementary pairs of literals. If $l \in s$, being s an assignment, then s makes l $true$ and makes $\bar{l}$ $false$. A *clausula* c *is satisfied* if and only if $c \cap s \neq \emptyset$, and if for all $l \in c, \bar{l} \in s$ then s falsifies c.

A CF Σ is satisfied by an assignment s if each *clausula* in Σ is satisfied by s and Σ is contradicted if it is not satisfied. s is a model of Σ if s is a satisfied assignment of Σ.

Let $SAT(\Sigma)$ be the set of models than Σ has over $v(\Sigma)$. Σ is a contradiction or unsatisfiable if $SAT(\Sigma) = \emptyset$. Let $\mu_{v(\Sigma)}(\Sigma) = |SAT(\Sigma)|$, be the cardinality of $SAT(\Sigma)$. Given Σ a CF, the SAT problem consists in determining if Σ has a model. The $\#SAT$ consists of counting the number of models of F defined over $v(\Sigma)$. We will also denote $\mu_{v(\Sigma)}(\Sigma)$ by $\#SAT(\Sigma)$.

Given a signed graph G_Σ, we obtain the associated Boolean formula Σ. Σ can be expressed as a two Conjunctive Form (2-CF) in the following way. Let $G_\Sigma = ((V, E), \sigma)$ be the signed graph, then $v(\Sigma) = V$ and for all positive edge $x \overset{+}{} y$ in G_Σ the clause (x, y) is part of Σ, while for a negative edge $x \overset{-}{} y$ in G_Σ the clause $(\overline{v(x)}, \overline{v(y)})$ is part of Σ. This means that the vertices of G_Σ are the variables of Σ, and each signed edge clause in E there is a clause in Σ.

2.1. Extending Signed Graphs

We consider the following approach which represents a very important application of signed graphs:

Corporate companies working based on projects must dynamically form work teams to solve the projects requirements. These companies have a finite set of "n" employees available, which we denote as: X=$\{x_1, ..., x_n\}$.

Project managers propose the staff selection to form a work team, based on restrictions between pairs of employees, such restrictions are formed according to the capabilities of employees and the requirements that must be accomplished in the project. For example, a restriction could be where both employees have the same abilities, another example could be where no employee covers some job profile for a certain type of project and this difficult or delay the develop of the project that might be accomplis with another employee that match the required profile.

The project manager defines the restrictions (capabilities or requirements)for each project with a pair of employees, where: $i \neq j = 1, 2, ..., n$. The clause is the way to model the restrictions on the employee x_i and the employee x_j. We identified four cases that model the relation between a pair of employees.

Case 1: (x_i). It indicates that it is mandatory for the employee x_i to be part of the project.

Case 2: $(x_i \vee x_j)$. The clause is unsatisfiable when $x_i = 0$ and $x_j = 0$, indicating that either one employees x_i or x_j must participate in the team, or even both can be on the same team.

Case 3: $(x_i \vee \sim x_j)$. The clause is unsatisfiable when $x_i = 0 \wedge x_j = 1$. It indicates that if x_j is on the team, then x_i should't be on the same team, because they block themselves, in the same way if x_i is in the team then x_j must not be in it.
Case 4: $(\sim x_i \vee \sim x_j)$. The clause is unsatisfiable if $x_i = 1 \wedge x_j = 1$ indicating that either one x_i or x_j employees should not participate in the team, even may not be both. But we must avoid having both in the same team.

This kind of restrictions are joined in a formula Σ in $2 - FC$ (conjunction of binary or unary clauses where each variable appears twice at most), and also forms a constraint graph of the formula G_Σ.

G_Σ shows graphically how work teams can be formed to develop a project. Thus, an assignment that satisfies Σ indicates a way of creating the proper work team that will handle a certain project. This means that the variables that take value 1, represents the employee that will form the team, and the value 0 represents the employee that should not be on the team.

Therefore $SAT(\Sigma)$ contains all the possible teams that can be formed to implement the project effectively and $\#SAT(\Sigma)$ will indicate us how many different teams can be formed for the project Σ.

The signed graphs are very useful to represent some kind of relationships among individuals. In this case, each employed is a node of the graph and there is an arc between $\{x, y\}$ if x is in some relation to y. Many of the relationships of interest have natural opposites, for example: likes/dislikes, associates with/avoids, and so on [2]. For this, two different signs $\{+, -\}$ are associated with each edge of the graph.

For example, if we consider that x likes y, then a positive edge is denoted between x and y, but we do not know what about y with x. y might dislike x or maybe like him. In order to be more precise in the kind of relationships between two individuals is better to consider two signs in each edge, one sign associated with each point end of the edge.

Furthermore, in order to consider general Boolean formulas in 2-CF, we consider an extension of the concept of signed graphs. Instead to consider just one sign (+ or -) associated with each edge of a signed graph $G = ((V, E), \sigma)$, we consider here that all edge in E has associated a pair of signs. Then the signed function σ has now the type $\sigma :\to \{(+,+),(+,-),(-,+),(-,-)\}$ which gives a pair of signs to each edge of G.

In this way, all binary clause $\{x, y\}$ can be represented by a signed edge in the graph in a natural way without importance of the signs associated to the variables in the clause. For example the clause $\{\overline{x}, y\}$ will be represented as the signed edge $x \overset{- +}{- -} y$ and in this case, - is called the adjacent sign of x and + the adjacent sign of y in the edge $x \overset{- +}{- -} y$.

Add more, we can consider those extended signed graphs as work team $\Sigma = (G, \sigma)$ where $G = (V, E)$ is the underlying graph of Σ and $\sigma : E \to \{(+,+),(+,-),(-,+),(-,-)\}$ is the signature function. That means that all edge in the graph of work team is signed in its endpoints.

We are interested here, in count the different assignments associated to the nodes of the work team in such a way that all edge in the work team will be satisfied by at least one of its two possible signs of its endpoints.

For this, we start analyzing the way to build satisfy assignment in the work team considering first the most simple topologies associated with the underlying graph of a work team.

3. #SAT Solution Techniques

3.1. If G_Σ is a Linear Path

Let us consider that the graph of the work team, $G_\Sigma = (V, E)$ **is a linear path**. Let us write down its associated formula Σ, without a loss of generality (ordering the clauses and its literals, if it were necessary), as: $\Sigma = \{c_1, ..., c_m\} = \left\{\{x_1^{\epsilon_1}, x_2^{\delta_1}\}, \{x_2^{\epsilon_2}, x_3^{\delta_2}\}, \ldots, \{x_{m-1}^{\epsilon_m}, x_m^{\delta_m}\}\right\}$, where $|\upsilon(c_i) \cap \upsilon(c_{i+1})| = 1, i \in [\![m-1]\!]$, and $\delta_i, \epsilon_i \in \{0, 1\},\ i = 1, ..., m$.

In order to compute the number of signed paths in such graph of the work team, we start with a pair of values (α_i, β_i) associated with each node i of the graph. We call the charge of node i to the pair (α_i, β_i). The value (α_i) indicates the number of times that node i takes the positive value and (β_i) indicate the number of times that node i takes the negative value.

Let f_i be a family of clauses of Σ built as follows: $f_0 = \emptyset$,$f_i = \{c_j\}_{j \leq i}$, $i \in [\![m]\!]$. Note that $f_i \subset f_{i+1}$, $i \in [\![m-1]\!]$. Let $SAT(f_i) = \{s : s\ satisfies\ f_i\}$, $A_i = \{s \in SAT(f_i) : x_i \in s\}$, $B_i = \{s \in SAT(f_i) : \overline{x}_i \in s\}$. Let $\alpha_i = |A_i|$; $\beta_i = |B_i|$ and $\mu_i = |SAT(f_i)| = \alpha_i + \beta_i$. From the total number of models in $\mu_i, i \in [\![m]\!]$, there are α_i of which x_i takes the logical value 'true' and β_i models where x_i takes the logical value 'false'.

For example, $c_1 = (x_1^{\epsilon_1}, x_2^{\delta_1})$, $f_1 = \{c_1\}$, and $(\alpha_1, \beta_1) = (1, 1)$ since x_1 can take one logical value 'true' and one logical value 'false' and with whichever of those values satisfies the subformula f_0 while $SAT(f_1) = \{x_1^{\epsilon_1} x_2^{\delta_1}, x_1^{1-\epsilon_1} x_2^{\delta_1}, x_1^{\epsilon_1} x_2^{1-\delta_1}\}$, and then $(\alpha_2, \beta_2) = (2, 1)$ if δ_1 were 1 or rather $(\alpha_2, \beta_2) = (1, 2)$ if δ_1 were 0.

In general, we compute the values for (α_i, β_i) associated to each node x_i, $i = 2, .., m$, according to the signs (ϵ_i, δ_i) of the literals in the clause c_i, by the next recurrence equation:

$$(\alpha_i, \beta_i) = \begin{cases} (\beta_{i-1} & , \alpha_{i-1} + \beta_{i-1}) \ if(\epsilon_i, \delta_i) = (-, -) \\ (\alpha_{i-1} + \beta_{i-1} & , \beta_{i-1}) \ if(\epsilon_i, \delta_i) = (-, +) \\ (\alpha_{i-1} & , \alpha_{i-1} + \beta_{i-1}) \ if(\epsilon_i, \delta_i) = (+, -) \\ (\alpha_{i-1} + \beta_{i-1} & , \alpha_{i-1}) \ if(\epsilon_i, \delta_i) = (+, +) \end{cases} \tag{1}$$

We denote with $' \rightarrow '$ the application of one of the four rules of the recurrence (1), so, the expression $(2, 3) \rightarrow (5, 2)$ denotes the application of one of the rules (in this case, the rule 4), over the pair $(\alpha_{i-1}, \beta_{i-1}) = (2, 3)$ in order to obtain $(\alpha_i, \beta_i) = (\alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}) = (5, 3)$.

In recurrence (1) the (ϵ_i, δ_i) represents the signs associated with the edge joining a child node joining with the father node. These recurrence equations

allow to carry the count to indicate the different ways to conform work teams for the project. In fact, the sum $\alpha_i + \beta_i$ obtained from the root node of the graph, indicate the total number of Boolean formula models associated with the graph and it is as well as the number of different ways to conform work teams for the project.

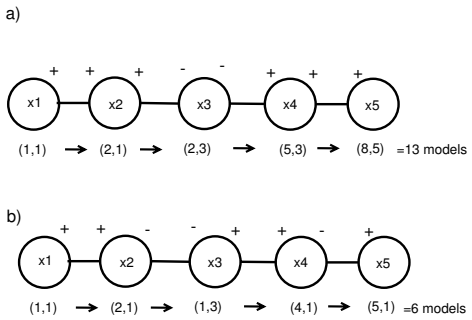

Fig. 1. a) Counting models over a positive path. b) Counting models over a general path.

Example 1 *Let* $\Sigma = \{(x_1, x_2), (x_2, \overline{x}_3), (\overline{x}_3, x_4), (x_4, x_5)\}$ *be a path, the series* $(\alpha_i, \beta_i), i \in [\![5]\!]$*, is computed according to the signs of each clause, as it is illustrated if the figure (1a). A similar path with 5 nodes but with different signs in the edges is shown in figure (1b).*

If Σ is a path , we apply (1) in order to compute $\mu(\Sigma)$. The procedure has a linear time complexity over the number of variables of Σ, since (1) is applied while we are traversing the chain, from the initial node y_0 to the final node y_m.

3.2. If G_Σ is a Tree

The charge of the nodes (α_i, β_i) for $i = 1...n$, are calculated as the nodes are visited applying a post-order traversals. The first pair of values (α_i, β_i) for the terminal nodes of the network (tree leaves) are initialized with the value $(1, 1)$. So, to traverse the tree in post-order, we start by the left tree first, followed by the right tree, and then finally the root node. At each visit of a child node $i-1$ to a father node i, the new values (α_i, β_i) are computed for the father node as shown in the algorithm 1.

This procedure returns the number of signed paths as the ways to conform the work teams for the project in time $O(n+m)$ which is the necessary time for traversing G_Σ in depth-first [7]

Input: A_Σ the tree defined by the depth-search over connectivity graph of the electrical network G_Σ
Output: The number of signed paths of an electrical network.
Procedure: Traversing A_Σ in depth-fist, and when a node $v \in A_\Sigma$ is left (all of its edges have been processed) assign:
1. $(\alpha_v, \beta_v) = (1, 1)$, If v is a leaf node in A_Σ
2. If v is a father node with a list of child nodes associated, i.e., $u_1, u_1, ...u_k$, are the child nodes of v, then as we have already visited all the child nodes, then each pair $(\alpha_{u_j}, \beta_{u_j})$ $j = 1, ..., k$ has been defined based on (1), $(\alpha_{v_i}, \beta_{v_i})$ is obtained by apply (2) over $(\alpha_{i-1}, \beta_{i-1}) = (\alpha_{u_j}, \beta_{u_j})$. This step is iterated until computes all the values $(\alpha_{v_j}, \beta_{v_j})$, $j = 1, , k$. And finally, let $\alpha_v = \prod_k^{j=1} \alpha_{v_j}$ y $\beta_v = \prod_k^{j=1} \beta_{v_j}$
3. If v is the root node of $A\Sigma$ the return $(\alpha_v + \beta_v)$

Algorithm 1: Algorithm Count_Models (A_Σ)

3.3. If G_Σ have a Single Cycle or Parallel Edges

The method now proposed is based on the recurrence equations in Table 1 and set theory.

Parallel Edges The parallel edges are those that join two nodes more than once, to be exact up to 4 times. In this way can be created only 4 cases of edges $\{ \overset{+\,+}{\longrightarrow}, \overset{+\,-}{\longrightarrow}, \overset{-\,+}{\longrightarrow}, \overset{-\,-}{\longrightarrow}\}$, a fifth case would be redundant of any of the above. Hence the idea to combine the recurrence equations with set theory.

The method consists of:
1.We must know which is the source node and destination node, so we choose the direction of the edges.
2. Values are taken Greek letters source node (α_i, β_i) to obtain the final result of the values of the destination node $(\alpha_{i+1}, \beta_{i+1})$. Each value of the destination node is a unitary set. If there is a sum, then they will join Greek letters and form a set of two elements.
3. For each ordered pair value of their corresponding sets intersect. The resulting set will be the value that the destination node. If the intersection is empty set then the value is zero.
4. In case of quadruple parallel edges automatically eliminates the possibility of finding a model or mapping that satisfies the Boolean formula.

Example 2 *Let* $\Sigma = \{(x_1, x_2), (x_1, x_4), (x_1, x_8), (\overline{x}_1, \overline{x}_8), (x_1, \overline{x}_8), (\overline{x}_1, x_8), (x_2, \overline{x}_3), (\overline{x}_2, x_3), (x_4, x_5), (x_4, x_6), (x_6, x_7), (x_6, \overline{x}_7), (\overline{x}_6, x_7)\}$

Step 1. In this example, Σ contains quadruple parallel edges which we give the result that there is no satisfiable assignment for it. We note that the leaf nodes should take values from $(1, 1)$. To resolve double parallels edges that join x_3, x_2, we apply the recurrence (1) where in this case, we have the cases of edges ($\overset{+\,-}{\longrightarrow}$, $\overset{-\,+}{\longrightarrow}$). In Table 1, we show the step 3, with the help of recurrence (1) for the cases 2 and 3.

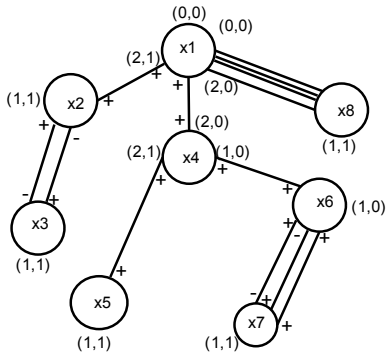

Fig. 2. Graph with parallel edges from a formula in 2-CF.

Table 1. Recurrence Equations expressed in sets

3. (+)⟶(-)	$\alpha_{i+1} = \{\alpha_i\} \quad \beta_{i+1} = \{\alpha_i, \beta_i\}$
2. (-)⟶(+)	$\alpha_{i+1} = \{\alpha_i, \beta_i\ \} \quad \beta_{i+1} = \{\beta_i\}$

In Table 2, we show the step 4, intersections are performed of $(\alpha_{i+1}, \beta_{i+1})$.

Table 2. Recurrence Equations expressed in sets of triple edges

(x_3, x_2)	$\alpha_{i+1} = \{\alpha_i\} \bigcap \{\alpha_i, \beta_i\} = \{\alpha_i\}$ $\beta_{i+1} = \{\alpha_i, \beta_i\} \bigcap \{\beta_i\} = \{\beta_i\}$

Table 3. Intersection of the sets of $(\alpha_{i+1}, \beta_{i+1})$

1. (+)⟶(+)	$\alpha_{i+1} = \{\alpha_i, \beta_i\ \} \quad \beta_{i+1} = \{\alpha_i\}$
2. (+)⟶(-)	$\alpha_{i+1} = \{\alpha_i\} \quad \beta_{i+1} = \{\alpha_i, \beta_i\}$
3. (-)⟶(+)	$\alpha_{i+1} = \{\alpha_i, \beta_i\ \} \quad \beta_{i+1} = \{\beta_i\}$

In Table 2, we show the step 4, intersections are performed of $(\alpha_{i+1}, \beta_{i+1})$.

In Table 4, we obtain an empty set for β_{i+1}. In quadruple edges of (x_8, x_1), the end result for x_1 is $(0, 0)$, to prove this only sets must intersect all four cases, this is shown in Table 5.

Table 4. Intersection of the sets of $(\alpha_{i+1}, \beta_{i+1})$ with triple edges

(x_7, x_6)	$\alpha_{i+1} = \{\alpha_i, \beta_i\} \bigcap \{\alpha_i\} \bigcap \{\alpha_i, \beta_i\} = \{\alpha_i\}$ $\beta_{i+1} = \{\alpha_i\} \bigcap \{\alpha_i, \beta_i\} \bigcap \{\beta_i\} = \emptyset$

Table 5. Intersection of the sets of $(\alpha_{i+1}, \beta_{i+1})$ whit quadruple edges

(x_8, x_1)	$\alpha_{i+1} = \{\alpha_i, \beta_i\} \bigcap \{\alpha_i\} \bigcap \{\alpha_i, \beta_i\} \bigcap \{\beta_i\} = \emptyset$ $\beta_{i+1} = \{\alpha_i\} \bigcap \{\alpha_i, \beta_i\} \bigcap \{\beta_i\} \bigcap \{\alpha_i, \beta_i\} = \emptyset$

Simple Cycles So far we have only count models of Boolean formulas that are converted to tree-like graphs. But what happens if there is a simple cycle? This mystery will be solved below.

The cycles are formed by back edges [4], which is why we subtract the value of the path to that node, the negative value of the ordered pair that makes the back edge unsatisfiable.

The method consists of:

1. Make to traverse in a graph until arriving at the back edge. The first node is the destination of the back edge, then follow the normal course of recurrence equations, until the source node of the back edge.
2. You must find the values that unsatisfiable the clause that represents the back edge.
3. Is again make to traverse from the destination node to the source node of the back edge, but taking into account only the value of the ordered pair that contains the destination node which unsatisfied the back edge. The other side will get a zero.
4. When you reach the the source node back edge only subtract the negative part of the ordered pair of the new tour route to normal.
5. This new ordered pair used to traverse the whole graph.
6. Should you find another cycle or back edge repeat the process.

Example 3 *Let* $\Sigma = \{(x_1, \overline{x}_2), (\overline{x}_2, x_3), (x_2, x_4), (x_3, x_4), (\overline{x}_4, \overline{x}_5)\}$ *be a boolean formula in 2-CF.*

Step 1. The graph obtained of Σ is shown in Figure 3a. The values of recurrence equations are below the graph, without regard to cycle or back edge. In this way, there are 11 assignments that satisfy Σ.

Step 2. The back edge that forms a simple cycle is (x_2, x_4). Previous clause is unsatisfied with $x_2 = 0$ and $x_4 = 0$, if x_4 is source node and x_2 is destination node.

Step 3. Make to traverse from the node x_2, but its value will $(0, 2)$, as this node unsatisfiable in the negative part and therefore the positive part is converted in zero. This is shows in Figure 3b.

Step 4. Substraction is made of the negative part obtained in the steps 1 and 3,

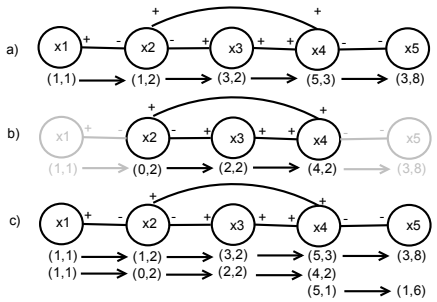

Fig. 3. a) Graph with simple cycle. Normal tour. b) Traversing the cycle. c)Count models of a graph with a simple cycle.

respectively, $(5,3)-(0,2)=(5,1)$.
Step 5. We ends the tour which is shown in figure 3c. The result is 7 assignments that satisfy Σ.

4. Conclusions and Future Work

Most companies today use work teams to reach a particular purpose, which obtains effectively results for certain type of project under a certain time.

The task of project managers is to consolidate staff to create a team that can achieve the desired results, under certain types of restrictions, that might be empathy, capabilities, abilities and knowledge to accomplish the project. These relations or restrictions are often complex for the project manager and he need to take the best decision.

Our solution is based on graph theory and the satisfiability solutions of boolean formulas that represent the graphs. They can solve the problem by finding the number of ways to form work teams and the individuals that are part of those teams, eliminating errors and reducing time in selecting the right staff.

Until now only we solve graphs with simple edges, parallel edges, simple cycles and combinations between them. The solution of graphs with nested loops is in process and will help to have a complete scene and can model any job relation.

References

1. D.J.M. Garey: Computers and Intractability a Guide to the Theory of NP-Completeness (1979)
2. Fred S. Roberts: Graph Theory and Its Applications to Problems of Society, Edit. SIAM Collection, (1978)
3. Germina K.A.: Hameed S.K.: On signed paths, signed cycles and their energies, Applied Mathematical Sciences, Vol. 4:(70),pp. 3455-3466 (2010)

4. Johnsonbaugh Richard: Matemticas Discretas, Editorial Prentice Hall. 4ta edicin (1993)
5. Kota P. S., Subramanya M.S.: Note on path signed graphs, Notes on Number Theory and Discrete Mathematics 15:(4), pp. 1-6 (2009)
6. Roth, D.: "On the hardness of approximate reasoning", Artificial Intelligence 82,pp 273-302 (1996)
7. R. Tarjan: Depth-First Search and linear Graph Algorithms, SIAM Journal on Computing 1:146-160 (1972)
8. Vadhan, S. P.: "The complexity of Counting in Sparse, Regular, and Planar Graphs", *SIAM Journal on Computing*,pp. 398-427, V31, N2 (2001)