A Minimum Spanning Tree for the #2SAT Problem

Guillermo De Ita, Meliza Contreras, Pedro Bello

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

Abstract. #2SAT is a classical #P-complete problem. We present here, a novel algorithm for given a 2-CF Σ, to build a minimum spanning tree for its constraint graph G_Σ assuming dynamic weights on the edges of the input graph.
Keywords: Counting the Number of Models, Minimum Spanning Tree.

1. Introduction

Counting combinatorial objects over graphs has been an interesting and important area of research in Mathematics, Physics, and Computer Sciences. Counting problems, being mathematically interesting by themselves, are closely related to important practical problems. For instance, reliability issues are often equivalent to counting problems [2, 3, 5].

The techniques for building minimum spanning trees have been developed assuming static weights on the edges of the graph. But for the #2SAT problem [4, 6], instead of static weights we consider dynamic weights determined by the signs of the edges, as well as the number of partial models associated with the nodes. We address the construction of the minimum spanning tree of a constraint graph considering such dynamic weights.

2. Preliminaries

Let Σ be a 2-CF, the *constraint graph* of Σ is the undirected graph $G_\Sigma = (V(\Sigma), E(\Sigma))$, with $V(\Sigma) = \upsilon(\Sigma)$ and $E(\Sigma) = \{\{\upsilon(x), \upsilon(y)\} : \{x, y\} \in \Sigma\}$, i.e. the vertices of G_Σ are the variables of Σ, and for each clause $\{x, y\}$ in Σ there is an edge $\{\upsilon(x), \upsilon(y)\} \in E(\Sigma)$. Each edge has associated an ordered pair (s_1, s_2) of signs assigned as labels. For example, the signs s_1 and s_2 for the clause $\{\overline{x} \vee y\}$ are related to the signs of the literals x and y respectively, then $s_1 = -$ and $s_2 = +$ and the edge is denoted as: $x \overset{-}{—} \overset{+}{} y$ which is equivalent to $y \overset{+}{—} \overset{-}{} x$.

A *connected component* of G is a maximal induced subgraph of G. We say that the set of *connected components* of Σ are the subformulas corresponding to the connected components of G_Σ. From now on, when we mention a 2-CF Σ, we assume that Σ is a connected component graph.

3. Linear procedures for #2SAT

We present here, some procedures for computing the number of models of a formula for basic topologies of a graph.

Procedure A: If Σ is a path:

The first pair (α_0, β_0) is (1,1) since for any logical value to y_0, f_0 is satisfied. (α_i, β_i) associated with each variable y_i, $i = 1, .., m$ is computed according to the signs: ϵ_i, δ_i of the literals in the clause c_i, by the recurrence (1). As $\Sigma = f_m$ then $\#SAT(\Sigma) = \mu_m = \alpha_m + \beta_m$. We denote with $' \rightarrow'$ the application of one of the four rules in the recurrence.

$$(\alpha_i, \beta_i) = \begin{cases} (\beta_{i-1}, \mu_{i-1}) \text{ if } (\epsilon_i, \delta_i) = (0,0) \\ (\mu_{i-1}, \beta_{i-1}) \text{ if } (\epsilon_i, \delta_i) = (0,1) \\ (\alpha_{i-1}, \mu_{i-1}) \text{ if } (\epsilon_i, \delta_i) = (1,0) \\ (\mu_{i-1}, \alpha_{i-1}) \text{ if } (\epsilon_i, \delta_i) = (1,1) \end{cases} \quad (1)$$

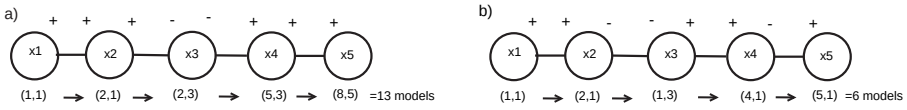

Fig. 1. a) Counting models on a monotone paths b)Counting models on a changing sign path

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 the edges, as it is illustrated in figure (1a). A similar path but with complementary signs on the nodes is shown in (1b).*

Let us consider the class $\mathcal{P}$ of Boolean functions where their constraint graphs are paths. For example, figures (1a) and (1b) represent Boolean path functions. We denote as P_n^+ a monotone path with n nodes and where each variable appears with the same sign (like in (1a)), while P_n^{-+} is the n path where the occurrences of the variables are with complementary signs (like in (1b)), we call to this last Boolean formula a *changing sign path*. Let $\#^+ : I\!N \rightarrow I\!N$ be the function which for a $n \in I\!N$, it considers a monotone path $P_n^+ \in \mathcal{P}$ holding that $\#^+(n) = \#SAT(P_n^+)$. We can estimate the rate of growth of $\#^+$ since it corresponds with the growth of $\#SAT(P_n^+)$ when the number of variables n growth. As $\#SAT(P_n^+) \in O(\phi^n)$ with $\phi = \left(\frac{1+\sqrt{5}}{2}\right) \approx 1.618$ [1], then, $\#^+(n) \in O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$. Let $\#^{-+} : I\!N \rightarrow I\!N$ be a function which for a $n \in I\!N$ it considers a changing sign path $P_n^{-+} \in \mathcal{P}$, and $\#^{-+}$ holds that

$\#^{-+}(n) = \#SAT(P_n^{-+})$. Thus, the rate of growth of the function $\#^{-+}$ is the same that the growth of $\#SAT(P_n^{-+})$ when the number of variables n growth, and as $\#SAT(P_n^{-+}) \in O(n)$ [1], then $\#^{-+} \in O(n)$. Comparing the rates of growth for P^+ and P^{-+}, we obtain $O(\#SAT(P_n^+)) > O(\#SAT(P_n^{-+}))$.

Procedure B: If Σ is a tree:
Let Σ be a 2-CF where its associated constraint graph G_Σ is a tree. We denote with (α_v, β_v) the pair associated with the node v ($v \in G_\Sigma$). We compute $\#SAT(\Sigma)$ considering the methodology used in [1].

Example 2 *Let B_7^+ a monotone balanced tree, like in (2a). And B_7^{+-} the binary balanced tree where each internal node has complementary signs on its incident child-edges. Applying Count_Models_for_trees to both trees, it starts assigning the pair (1,1) to the leaf nodes. The number of models at each level of the tree is shown in Figure 2. At the level of the root node, the procedure obtains $\#SAT(B_n^+) = 25 + 16 = 41$. And for the second tree, $\#SAT(B_7^{+-}) = 8 + 8 = 16$.*

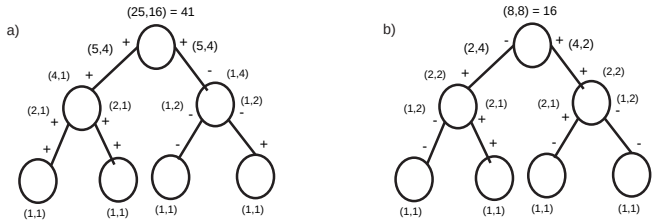

Fig. 2. a) Counting models over a monotone balanced tree, b) Counting models over a non-monotone balanced tree

For monotone Boolean formulas whose constraint graph is a balanced binary tree, like in example 2, we can express how the number of models changes according with the number of nodes of the tree. E.g. for a monotone binary tree with 3 nodes $(\alpha_3, \beta_3) = (2^2, 1^2) = (4, 1)$. The following balanced binary tree has 7 nodes and $(\alpha_7, \beta_7) = (5^2, 4^2) = (25, 16)$. The following tree has 15 nodes and $(\alpha_{15}, \beta_{15}) = (41^2, 25^2) = (1681, 625)$. In general, we obtain the following recurrence relation for the balanced monotone binary trees B_n^+ with n nodes: $\#\text{SAT}(B_n^+) = F_{n/2}^2 + F_{n/4}^4$ being F_i the i-th Fibonacci number. This recurrence relation has an order of growth of $O(2^{(3/4)n})$.

On the other hand, the upper bound for the class of binary formulas where there are complementary signs on the child-edges on each internal node, tree denoted as B_n^{+-} is given by $\#SAT(B_n^{+-}) \in O(2^{\frac{n}{2}})$. The previous upper bounds establish a hierarchy for #SAT according to the topology of the constraint graphs, given as: $O(\#SAT(P_n^{-+})) \leq O(\#SAT(B_n^{+-})) \leq O(\#SAT(P_n^+)) \leq O(\#SAT(B_n^+))$.

4. Building the Minimum Spanning Tree

Let $A_1, \ldots, A_n$ be the sequence of initial charges obtained by the procedures (A) or (B) for computing #SAT(Σ). Now, we build a new sequence of pairs which represent the final charges (or just the charges) $B_n, \ldots, B_1$, being B_i the charge of the variable $x_i \in v(\Sigma)$, computed as:

$$\begin{array}{l} B_n = \quad A_n \\ B_{n-i} = balance(A_{n-i}, B_{n-i+1}),\ i = 1, ..., n-1 \end{array} \tag{2}$$

balance(A, B) is a binary operator between two pairs, e.g. if $x \frac{s_1\ s_2}{y}$ is an edge of the constraint graph G_Σ, and assuming $A = (\alpha_x, \beta_x)$ be the initial charge of the variable x, $B = (a_y, b_y)$ be the final charge of the variable y, then *balance* produces a new pair (a_x, b_x) which will be the final charge for x, i.e. #SAT(Σ) $= a_x + b_x$. Let $\mu_x = \alpha_x + \beta_x$ and $\mu_y = a_y + b_y$. Let $P_1 = \frac{\alpha_x}{\mu_x}$ and $P_0 = \frac{\beta_x}{\mu_x}$ be the proportion of the number of 1's and 0's in the initial charge of the variable x. The final charge (a_x, b_x) is computed, as:

$$\begin{array}{l} a_x = a_y \cdot P_1 + b_y;\ b_x = \mu_y - a_x \text{ if}(s_1, s_2) = (+, +) \\ b_x = b_y \cdot P_0 + a_y;\ a_x = \mu_y - b_x \text{ if}(s_1, s_2) = (-, -) \\ b_x = b_y \cdot P_0 + a_y;\ a_x = \mu_y - b_x \text{ if}(s_1, s_2) = (+, -) \\ a_x = a_y \cdot P_1 + b_y;\ b_x = \mu_y - a_x \text{ if}(s_1, s_2) = (-, +) \end{array} \tag{3}$$

In the case that the coefficients P_0 or P_1 are not integer numbers, the following formulas are applied for computing the charge of x.

$$\begin{array}{l} a_x = (b_y - a_y);\ b_x = (\mu_y - a_x) \text{ if}(s_1, s_2) = (-, -) \\ a_x = (a_y - b_y);\ b_x = (\mu_y - a_x) \text{ if}(s_1, s_2) = (-, +) \\ b_x = (b_y - a_y);\ a_x = (\mu_y - b_x) \text{ if}(s_1, s_2) = (+, -) \\ b_x = (a_y - b_y);\ a_x = (\mu_y - b_x) \text{ if}(s_1, s_2) = (+, +) \end{array} \tag{4}$$

Note that the essence of the rules in *balance* consists in applying the inverse operation utilized via recurrence (1) during the computation of #SAT(Σ). Furthermore, in the case of bifurcations from a father node to a list of child nodes, the application of recurrence (3) or (4) remains valid since each branch edge has its respective pair of signs.

4.1. A Procedure for Building Spanning Trees for #2SAT

Given a 2-CF Σ and its constraint graph G_Σ, a *minimum spanning tree*, denoted by T_Σ, is a tree containing all vertices of G_Σ and such that $\#SAT(A_\Sigma) \geq \#SAT(\Sigma)$ and $\#SAT(A_\Sigma)$ is minimum into the set of all spanning trees of G_Σ.

When G_Σ contains cycles, our proposal works like the well known Kruskal's algorithm. An initial spanning tree $A_\Sigma = (V(G_\Sigma), P\_Edges)$ is formed by all vertices of G_Σ because all vertices are connected components by themselves, and all pendant edge of G are edges of the spanning tree (if there are not pendant edges then an empty set is initially assigned to P_Edges).

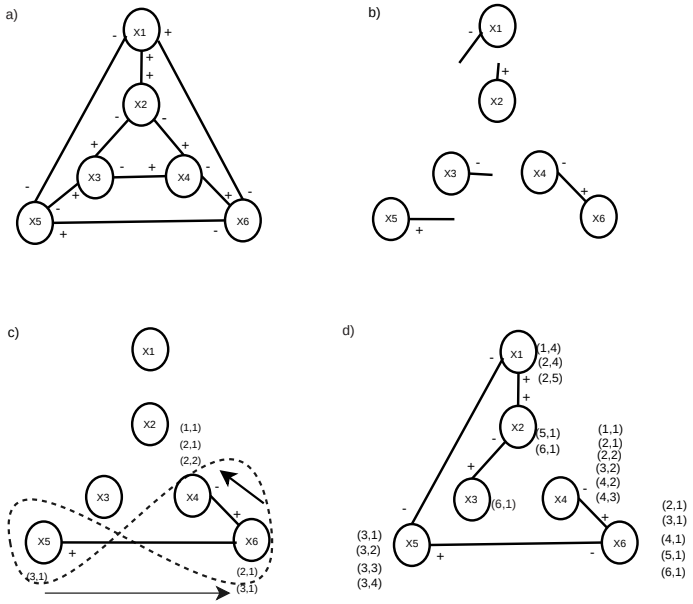

Fig. 3. Phases on the construction of the minimum spanning tree from an initial constraint graph

.

The first step of our procedure consists in detecting the potential edges which could generate change of signs when a node will be visited,(see fig. 3b). In each step, each edge with one of its end-points in a node of A_Σ and the other end-point in a node not included in A_Σ. While the procedure detects an edge $e \in E$ which generate change of signs on the incident edges of a same node, such edge e is added to A_Σ. When there are more than one edge E generating change of signs or there are not such edges the procedure reviews how increase the number of models when an edge $e \in E$ would be added to A_Σ, for this in each node an inverse counting is calculated ,(see fig. 3c). The resulting spanning tree (see fig. 3d) with a total number of models of 7=6+1.

The edge $e \in E$ which infers a minimal increment on $\#\mathrm{SAT}(A_\Sigma)$ with respect to any other edge in E, is selected to be added to A_Σ. Notice that the increment on the number of models depends mainly of the signs of e as well as the charge of the two-endpoints of e. There are a set of strategies used for detecting the edges in $(E(G_\Sigma) - E(A_\Sigma))$ which infer minimal increment on $\#\mathrm{SAT}(A_\Sigma)$. Such strategies are: In general if two edges e_1 and e_2 generate the same increment on the number of models of A_Σ e_1 is preferred over e_2 if e_1 could bring about a change of signs, in the following steps, on its incident node, if the edge e connects A_Σ with one extra-node and generates complementary signs on one of its end-points, then e is an optimal selection, and it is preferable to obtain a path than a tree when the signs on the edges are the same.

Algorithm 1 Procedure $Spanning\_Tree(G_\Sigma)$ (Input: $G_\Sigma = (V(G), E(G))$)

Let $P\_Edges = \{e \in E(G_\Sigma) : e$ is a pendant edge $\}$;
$All\_Edges := E(G) - P\_Edges$; $Cs := \emptyset$; {Set of potential edges}
$A_\Sigma := (V(G), P\_Edges)$; {all pendant edges are part of the Tree}
$CandEdge = Select_Candidate Edges(All\_Edges)$ {Set of candidate semi-edges }
$FirstEdge = Union_Candidate Edges(CandEdge)$ {looking for a complete edge}
if $FirstEdge <> \emptyset$ **then**
  $E(A_\Sigma) := E(A_\Sigma) \cup \{FirstEdge\}$; $Initialize(Vect\_Models, 1, 1)$;
  $Count\_Models(A_\Sigma, Vect\_Models)$; $InverseOrderCount(A_\Sigma, Vect\_Models)$;
end if
while $(All\_Edges <> \emptyset)$ **do**
  $Count\_Models(All\_Edges, Vect\_Models)$; {count models by each potential edge}
  $Sel\_Edge = min\{Vect\_Models\}$; {select the edge which minimum incremental}
  if $|Sel\_Edge| > 1$ **then**
    $Cs = Find(Test, A_\Sigma)$; {looking for edges which could generate change of signs}
    $Test = complete(Cs)$; {choose edges generating change of sign on the nodes}
    $Sel\_Edge = First(Test)$; {Select an edge keeping potential change of sign}
  end if
  $All\_Edges := All\_Edges - \{Sel\_Edge\}$; $E(A_\Sigma) := E(A_\Sigma) \cup \{Sel\_Edge\}$;
  $All\_Edges := All\_Edges - Edges\_Cycles(A_\Sigma, All\_Edges)$; {delete cycles}
end while

5. Conclusions

We consider a new line of researching of building spanning trees with dynamic weights determined by the signs of the edges and the partial number of models associated with the endpoints of the edges. This consideration allows us, given a 2-CF Σ, to build its minimum spanning tree A_Σ such that $\#\mathrm{SAT}(A_\Sigma)$ is a minimum upper bound for $\#\mathrm{SAT}(\Sigma)$. To build efficiently the minimum spanning tree of a 2-CF is very helpful in the research for determining frontiers between efficient and exponential counting procedures for the #2SAT problem.

References

1. De Ita G, Bello P, Contreras M, New Polynomial Classes for #2SAT Established Via Graph-Topological Structure, *Jour. Engineering Letters*, Vol. 15 (2), (2007), pp.250-258.
2. Dyer M., Greenhill C., Some #P-completeness Proofs for Colourings and Independent Sets, Research Report Series, University of Leeds, 1997.
3. Greenhill Catherine, The complexity of counting colourings and independent sets in sparse graphs and hypergraphs", *Computational Complexity*, 9(1): 52-72, 2000.
4. Russ B., *Randomized Algorithms: Approximation, Generation, and Counting*, Distinguished dissertations Springer, 2001.
5. Roth D., On the hardness of approximate reasoning, *Artificial Intelligence 82*, (1996), pp. 273-302.
6. Wahlström M., A Tighter Bound for Counting Max-Weight Solutions to 2SAT Instances, *LNCS 5018*, Springer-Verlag (2008), pp. 202-213.