UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIERÍA
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

DOCTORADO EN CIENCIAS DE LA INGENIERÍA
FACULTAD DE INGENIERÍA

DISEÑO DE UN ALGORITMO PARA EL CONTEO DE MODELOS DE FÓRMULAS BOOLEANAS EN 2 FORMA NORMAL CONJUNTIVA

Alumno:	M. en C. Marco Antonio López Medina
Director de tesis:	Dr. José Raymundo Marcial Romero
Co-Director:	Dr. Jacobo Leonardo González Ruíz
Tutores:	Dr. Guillermo De Ita Luna
	Dr. Marcelo Romero Huertas

Índice general

1 Protocolo de investigación **11**

1.1 Marco teórico 11

1.1.1 Lógica proposicional 11

1.1.2 Forma Normal Conjuntiva y Forma Normal Disyuntiva 13

1.1.3 Métodos de conteo 14

1.1.4 Representación de fórmulas mediante gráficas 17

1.1.5 Construcción de una gráfica a partir de una fórmula en 2-FNC . . . 22

1.2 Estado del arte 24

1.3 Planteamiento del problema 27

1.4 Justificación 28

1.5 Hipótesis 28

1.6 Objetivo general 28

1.6.1 Objetivos específicos 28

1.7 Metodología propuesta 28

2 Publicaciones **31**

2.1 A Linear Time Algorithm for Counting #2SAT on Series-Parallel Formulas 31

2.2 Computing the clique-width on series-parallel graphs 31

2.3 A method for counting models on grid Boolean formulas 32

2.4 Un algoritmo lineal para el conteo de #2SAT en fórmulas tipo árbol con subfórmulas serial paralelo 32

2.5 Un algoritmo para el conteo de modelos en fórmulas booleanas cúbicas . . . 42

3 Discusión y Trabajo futuro **43**

3.1 Discusión 43

3.2 Trabajo futuro 46

Summary

The problem of model counting on a two conjunctive normal form (2-CNF) formula, denoted as #2SAT, is analyzed by its representation on series-parallel, grids, and cubic graphs. At present, the model counting problem does not have an efficient method to solve it in general, then the way to approach it is done based on cases that have specific characteristics, such as those mentioned above.

Each one of these graphs represents a formula in 2-CNF, in which the vertices represent the variables and the edges the clauses. In this way, a logic problem such as model counting is translated into a representation that can be worked on from a computational point of view.

In this Thesis, new methods are proposed to count models on these types of graphs using computational representations of them, such as: trees, matrices, vectors, and graphs labeled in vertices and edges.

Theoretical results show that the counting models on formulas, whose graph representation is series-parallel can be done in polynomial time and for grid graphs there is an algorithm whose complexity is lower than the last reported in the literature. On the other hand, the experimental results show that the counting models on formulas whose representation in graphs is cubic, the proposed heuristic algorithm improves the result of the best proposal reported in the literature.

Resumen

El problema de conteo de modelos en dos forma normal conjuntiva (2-FNC), denominado #2SAT se estudia a través de su representación en gráficas de tipo serial-paralelo, mallas y cúbicas. Hasta el momento, este problema de conteo de modelos no tiene un método eficiente que pueda resolverlo de manera general, por lo que la forma de abordarlo se realiza con base en casos que tienen características específicas, como las antes mencionadas.

Cada una de estas gráficas representa una fórmula en 2-FNC, en la que los vértices representan las variables y las aristas las cláusulas. De esta forma, se traduce un problema de la lógica como el conteo de modelos a una representación sobre la que se trabaja desde el punto de vista computacional.

En esta Tesis se proponen nuevos métodos para realizar el conteo de modelos sobre estos tipos de gráficas utilizando representaciones computacionales de ellos, como lo son: árboles, matrices, vectores, gráficas etiquetadas tanto en vértices como en aristas.

Los resultados teóricos muestran que el conteo de modelos de fórmulas cuya representación en gráficas es serial-paralelo se puede realizar en tiempo polinomial y para gráficas tipo malla existen un algoritmo cuya complejidad es menor a la reportada en la literatura. Por otro lado los resultados experimentales muestran que el conteo de modelos sobre fórmulas cuya representación en gráficas es cúbica el algoritmo heurístico propuesto mejora el resultado de la mejor propuesta reportada tambien en la literatura.

Introducción

Dentro de la teoría de la lógica computacional, el problema *SAT(F)*, donde F es una fórmula Booleana, consiste en decidir si F es satisfactible, es decir, tiene una asignación sobre sus variables tal que al ser evaluada con respecto a la lógica booleana clásica devuelve un valor verdadero [1], esta asignación es un modelo de F. Si F está en dos Forma Normal Conjuntiva (2-FNC), entonces *SAT(F)* se resuelve en tiempo polinomial, sin embargo, si F está en *k-FNC*, $k > 2$, entonces *SAT(F)* es un problema NP-Completo [2]. El problema de conteo de modelos para la fórmula F es denotado como *#SAT(F)*. A diferencia de *SAT(F)*, *#SAT(F)* en determinar cuántos modelos tiene la fórmula F. *#SAT(F)* pertenece a la clase #P-Completo, incluso en su restricción en 2-FNC, la cual se denota #2SAT [3]. Aunque el problema en general no tiene una solución eficiente, hay casos cuya solución está dada en tiempo polinomial [4].

En su mayoría los algoritmos que se utilizan para resolver #2SAT en cualquier fórmula F en 2-FNC, descomponen F hasta que se tienen subfórmulas, que representan casos base, y en estas subfórmulas se puede realizar el conteo de modelos en tiempo polinomial. Para el caso general, el algoritmo con la menor complejidad en tiempo hasta el momento fue desarrollado por Wahlström [5], el cuál está dado por $O(1.2377^n)$, donde n representa el número de variables de la fórmula. El algoritmo de Wahlström utiliza el método de descomposición por variable cuyo criterio de elección está dado por el número de veces que aparece en la fórmula (sea la variable o su negación) y considera dos criterios de paro cuando $F = \emptyset$ o cuando $\emptyset \in F$.

Además se han esarrollado las implementaciones para #SAT, denominadas *solvers* (SAT-solver), en las cuales el objetivo es buscar estrategias que permitan resolver el problema en un tiempo de cómputo razonable para instancias en dónde el número de variables es considerablemente alto. Haciendo uso de métodos heurísticos, enfocados sobre las variables, cláusulas o estructura de la fórmula, reduciendo el espacio de búsqueda de soluciones con respecto al nivel de precisión que requieran. Con la propuesta de que problemas reales van a ser resueltos sin usar métodos completos (costosos),y reduciendo el tiempo-esfuerzo requerido para realizarlo.

Por lo antes expuesto, en esta Tesis se estudian los métodos utilizados para resolver el problema #2SAT que se basan en el uso de la gráfica representada por la fórmula en 2-FNC, los cuáles son precisos en el resultado obtenido.

La Tesis tiene la siguiente estructura: El pirmer Capítulo se realiza la descripción general del proyecto considerando el protocolo de investigación actualizado. En tanto en el Capítulo 2 se incluyen las publicaciones obtenidas durante los estudios de Doctorado. Finalmente en el Capítulo 3 se presenta una discusión general a los distintos modelos propuestos, así como estableces el trabajo futuro.

Capítulo 1

Protocolo de investigación

El problema $SAT(F)$, donde F es una fórmula Booleana, consiste en decidir si F tiene un modelo, es decir una asignación a las variables de F tal que al ser evaluadas con respecto a la lógica Booleana clásica devuelva un valor verdadero. Si F esta en dos Forma Normal Conjuntiva (2-FNC) entonces $SAT(F)$ se puede resolver en tiempo polinomial, sin embargo si F esta en k-FNC, $k > 2$, entonces $SAT(F)$ es un problema NP-Completo. Por otro lado, existe el problema de conteo, denotado como $\#SAT(F)$, el cual consiste en contar el número de modelos de F. $\#SAT(F)$, a diferencia de $SAT(F)$ que se puede considerar como un problema de decisión, pertenece a la clase #P-Completo aún cuando F este en 2-FNC [3].

En esta investigación se estudian los principales métodos utilizados para resolver el problema #2SAT con la intención de obtener un método de conteo para fórmulas en 2-FNC, utilizando como auxiliar topologías de fórmulas estudiadas antes ([6] [7] [8]) y durante el proyecto.

1.1. Marco teórico

En esta sección se presentan los conceptos necesarios para realizar el trabajo de investigación.

1.1.1. Lógica proposicional

La lógica proposicional tiene como elemento básico los enunciados o proposiciones, que tienen un significado determinado y que pueden tener un solo valor de verdad *falso* o *verdadero*. Un enunciado o proposición puede ser sustituido por una variable proposicional, que lo representa dentro de una estructura lógica (fórmula).

Para alguna proposición, que llamaremos variable proposicional, se pueden utilizar los símbolos $p_0, p_1, ...$ Para las conectivas 'no', 'y', 'o', 'si ... entonces ...', '... si y solo si ...', se usan los símbolos $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, respectivamente; sus nombres son negación, conjunción, disyunción, condicional, bicondicional. Los paréntesis '(' y ')' se usan como signos de agrupamiento. Las variables proposicionales son también llamadas de manera común proposiciones atómicas o átomos. Entonces el alfabeto que se utilizará en adelante se forma por el conjunto $\{), (, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, p_0, p_1, p_2, ...\}$.

Cualquier expresión sobre este alfabeto es una cadena de símbolos como: "$(\neg p_0 \rightarrow ()\wedge p_1 \vee$", "$\neg p_{100})(\rightarrow \vee$", o "$(\neg p_0 \rightarrow p_1)$".

De este modo, una proposición es una *fórmula bien formada* si se compone basada en las siguientes reglas:

1. Cada p_i es una proposición, con $i \in \mathbb{N}$.
2. Si x es una proposición entonces su negación ($\neg x$ o $\bar{x}$) es una fórmula bien formada.
3. Si x, y son proposiciones, entonces $(x \vee y)$, $(x \wedge y)$, $(x \rightarrow y)$, $(x \leftrightarrow y)$ son fórmulas bien formadas.
4. Si un enunciado no satisface alguna o todas las reglas (1)-(3) ,entonces no es una fórmula bien formada.

Interpretación

El significado asociado con cualquier proposición puede ser de dos tipos, denominados *verdadero* y *falso*. También pueden escribirse como '0' para *falso* y '1' para *verdadero*. Estos dos elementos son llamados *valores de verdad*. Ya que las fórmulas bien formadas son construidas a partir de variables proposicionales y conectivas, y dependiendo de la situación las variables proposicionales pueden recibir cualquiera de los valores de verdad, se debe conocer como tratar con las conectivas.

El significado de sentido común para las conectivas $\neg, \wedge, \vee$, $\Longrightarrow$ y $\Longleftrightarrow$ son respectivamente *no*, *y*, *o*, *implica* y *si y sólo si*. Esto significa, $\neg x$ o $\bar{x}$ invierte el valor de verdad de x, esto es, si x es verdadero, entonces $\bar{x}$ es falso; y si x es falso, entonces $\bar{x}$ es verdadero. Cuando ambas x y y son verdaderas, $x \wedge y$ es verdadera; y cuando al menos una de x o y es falsa, $x \wedge y$ es falsa. Si al menos una de x o y es verdadera, entonces $x \vee y$ es verdadera; y si ambas x y y son falsas, entonces $x \vee y$ es falsa. De manera similar, $x \Longleftrightarrow y$ es verdadera cuando x y y son ambas verdaderas; o cuando x y y son ambas falsas; $x \Longleftrightarrow y$ es falsa si una de x, y es verdadera y la otra falsa. En el caso $x \Longrightarrow y$ es falsa cuando x es verdadera y y es falsa; en cualquier otro caso $x \Longrightarrow y$ es verdadera.

Formalmente a la suposición de asociar valores de verdad a las variables proposicionales se le llama *asignación de verdad*. Esto es, una *asignación de verdad* es una función de

$\{p_0, p_1, ...\}$ a $\{0, 1\}$. Una extensión de una asignación de verdad al conjunto de todas las proposiciones que evalúan las conectivas, con lo descrito en el párrafo anterior, se llama interpretación. Esto es, una *interpretación* es una función $I : Prop \to \{0, 1\}$ que satisface las siguientes propiedades para toda $x, y \in Prop$:

1. $I(\bar{x}) = 1$ si $I(x) = 0$, de otra forma, $I(\bar{x}) = 0$.
2. $I(x \wedge y) = 1$ si $I(x) = I(y) = 1$, de otra forma, $I(x \wedge y) = 0$.
3. $I(x \vee y) = 0$ si $I(x) = I(y) = 0$, de otra forma, $I(x \vee y) = 1$.
4. $I(x \implies y) = 0$ si $I(x) = 1$,$I(y) = 0$, de otra forma, $I(x \implies y) = 1$.
5. $I(x \iff y) = 1$ si $I(x) = I(y)$, de otra forma, $I(x \iff y) = 0$.

Éstas mismas condiciones se pueden representar en una tabla de verdad, los símbolos u, x, y serán las proposiciones. Estas condiciones se llaman condiciones Booleanas; y la tabla se llama tabla de verdad (ver tabla 1.1).

u	$\bar{u}$	x	y	$x \wedge y$	$x \vee y$	$x \implies y$	$x \iff y$
1	0	1	1	1	1	1	1
0	1	1	0	0	1	0	0
		0	1	0	1	1	0
		0	0	0	0	1	1

Tabla 1.1: Tabla de verdad para las condiciones de interpretación.

Una interpretación también es llamada como: evaluación, evaluación booleana, estado, situación, mundo, o lugar.

Modelos

Sea w una proposición. Un *modelo* para w es una interpretación I con $I(w) = 1$. El hecho de que I sea un modelo de w se escribe como $I \models w$. También se lee como I verifica w; I *satisface* w. Si I no satisface a w se escribe como $I \not\models w$; se lee como, I no satisface a w, I no verifica a w, I *falsifica* a w.

1.1.2. Forma Normal Conjuntiva y Forma Normal Disyuntiva

Una literal es una variable proposicional o la negación de una variable proposicional. Para una variable proposicional p, las literales p y $\bar{p}$ son llamadas literales complementarias (complementarias entre ellas). Una cláusula conjuntiva es una conjunción de literales; una cláusula disyuntiva es una disyunción de literales. Una proposición en forma normal con-

juntiva (FNC) será la conjunción de cláusulas disyuntivas, entonces, una proposición en forma normal disyuntiva (FND) será una disyunción de cláusulas conjuntivas.

Por ejemplo, si p y q son variables proposicionales. Entonces:

- p, $\bar{p}$, q, $\bar{q}$ son literales.
- $p \wedge q$, $\bar{p} \wedge q$, $p \wedge \bar{q}$, $\bar{p} \wedge \bar{q}$, son cláusulas conjuntivas.
- $p \vee q$, $\bar{p} \vee q$, $p \vee \bar{q}$, $\bar{p} \vee \bar{q}$, son cláusulas disyuntivas.
- $(p \vee q) \wedge (\bar{p} \vee q)$, es una forma normal conjuntiva.
- $(p \wedge \bar{q}) \vee (\bar{p} \wedge \bar{q})$, es una forma normal disyuntiva.

El objetivo de estudio son las fórmulas en 2-FNC, en las que cualquier cláusula tiene a lo más dos literales. Así mismo el conteo de modelos en estas gráficas se denomina *#2SAT* que pertenece a la clase #P completo [2], lo que indica que no existe algoritmo que resuelva el problema general de manera eficiente.

Las instancias en 2-FNC pueden ser representadas mediante una gráfica como se verá en la sección 1.1.4.

1.1.3. Métodos de conteo

Se presentarán algunos métodos usados para resolver este problema de conteo. Entre ellos se pueden encontrar las tablas de verdad, un árbol de eliminación o el proceso de descomposición de una fórmula en casos más simples (basado en su estructura).

Tablas de verdad

El método más simple de poder contar el número de modelos de una fórmula en general, con n variables, es realizar el cómputo de todas las posibles interpretaciones sobre la fórmula en unas tablas en las que:

- Las primeras n columnas representan el conjunto de combinaciones de interpretaciones sobre las n variables, lo que genera un número de renglones igual a 2^n.
- La siguiente columna representa la fórmula sobre la que se quiere contar, en cada uno de sus renglones se representa el valor de verdad obtenido al aplicar la interpretación de ese renglón sobre ella.

De esta forma, aunque es un método exhaustivo, se tiene una forma de obtener el número de modelos de cualquier fórmula bien formada dentro de la lógica proposicional. Por ejemplo para la fórmula $(p_1 \vee p_2 \vee \bar{p_3}) \wedge (p_2 \vee p_3)$, la tabla de verdad que permite contar los modelos se muestra en la tabla 1.2, en esta fórmula el número de modelos es 6.

p_1	p_2	p_3	$(p_1 \vee p_2 \vee \bar{p_3}) \wedge (p_2 \vee p_3)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Tabla 1.2: Conteo de modelos mediante una tabla de verdad.

Los métodos que se explican en las siguientes secciones permiten resolver el mismo problema realizando un menor número de operaciones.

Árbol de eliminación

En este método lo que se propone es poder tener un orden dentro de las variables de la fórmula, y que cada una de ellas forme un nivel dentro de un árbol binario, para poder realizar un conjunto de asignaciones parciales.

La estructura de árbol binario utilizada necesita que cada nodo dentro del árbol represente una fórmula, y cada nivel del árbol representa la variable a eliminar de la fórmula, utilizando los dos posibles valores de verdad para cada variable se generan dos nuevos nodos por cada nodo en el nivel de la variable a eliminar.

Por ejemplo dada la FNC $F = (x \vee \bar{y} \vee z)$, el árbol de eliminación generado tendrá cuatro niveles, la raíz del árbol contiene la FNC original, y si se tiene un orden de eliminación $\{y, z, x\}$, el segundo nivel contiene las fórmulas generadas al aplicar las dos posibles interpretaciones, sobre y en la FNC F, obteniendo F_y que asume un valor de verdad *verdadero* para y y $F_{\bar{y}}$ que asume un valor de verdad *falso* para la variable y. En el primer caso $F_y = (x \vee z)$, debido a que al asignar *verdadero* a y, $\bar{y}$ se convierte en falso, para el segundo caso $F_{\bar{y}} = \emptyset$, debido que al asignar *falso* a y, $\bar{y}$ se convierte en verdadero, al ser una cláusula con disyunciones obtener un valor verdadero en alguna de las variables que contiene satisface la cláusula sin importar las otras variables. Se muestra un ejemplo de árbol de eliminación por variable en la figura 1.1.

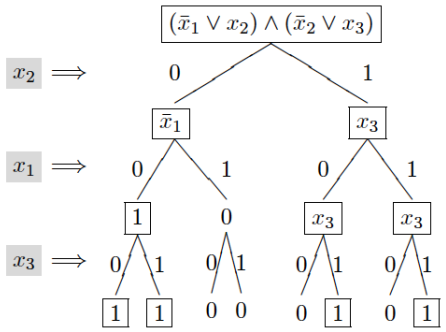


Figura 1.1: Árbol de eliminación de la fórmula $(\bar{x_1} \vee x_2) \wedge (\bar{x_2} \vee x_3)$ con el orden de eliminación $\{x_2, x_1, x_3\}$.

Éste método de árbol de eliminación puede ser aplicado utilizando cláusulas en lugar de variables, de esta forma en lugar de ser un árbol binario en el que cada rama toma valor *verdadero* o *falso*, se toman todas las posibles combinaciones de valores de verdad para las variables involucradas dentro de la cláusula, excepto la combinación única que la falsifica. En este caso se reduce el número de niveles en el árbol a costa de la cantidad de nodos hijo en cada nodo interno. Se muestra un ejemplo de árbol de eliminación por cláusula en la figura 1.2.

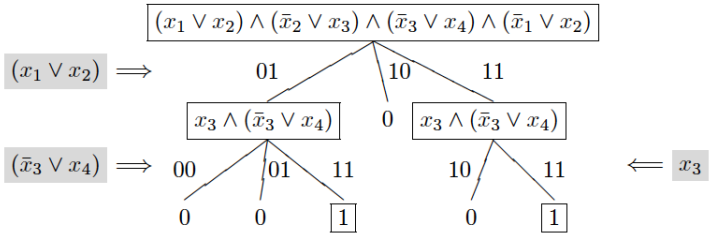


Figura 1.2: Árbol de eliminación de la fórmula $(x_1 \vee x_2) \wedge (\bar{x_2} \vee x_3) \wedge (\bar{x_3} \vee x_4) \wedge (\bar{x_1} \vee x_2)$ con el orden de eliminación $\{(x_1 \vee x_2), (\bar{x_3} \vee x_4)\}$.

Descomposición de una fórmula

Este método está relacionado con el concepto de árbol de eliminación ya que busca un orden en la eliminación de las variables de la fórmula, aunque en este caso se busca la mejor opción de eliminación por cada subfórmula obtenida en la descomposición, por lo que no es posible generar un árbol equilibrado como en el método anterior, ya que se busca obtener fórmulas con una estructura que permita contar sus modelos sin necesidad de realizar un siguiente paso de descomposición.

Este método permite el reconocer características en la estructura de una fórmula durante el proceso de descomposición, reduciendo en mayor medida la cantidad de pasos necesarios para realizar el conteo de modelos de una fórmula.

1.1.4. Representación de fórmulas mediante gráficas

Definición 1. *Una gráfica no dirigida G se representa por el par ordenado $(V(G), E(G))$, $V(G)$ es el conjunto de vértices y $E(G)$ es el conjunto de aristas, una arista de G se representa como $\{x, y\} : x, y \in V(G)$ [9].*

Cuando una arista conecta dos vértices se dice que son adyacentes y uno incide en el otro, si son vértices distintos entonces se dice que son vecinos. El grado de un vértice es el número de aristas incidentes en él. La representación de una arista que conecta al vértice u con el vértice v es:

$$u \text{ — } v$$

Si los vértices conectados por una arista son idénticos, entonces la arista es un ciclo simple o loop($e = \{w, w\}$). Si dos o más aristas unen al mismo par de vértices, entonces son aristas paralelas($e_i = \{u, v\}$, $e_j = \{u, v\}$). Se muestra un ejemplo en la Figura 1.3. Se denomina simple a una gráfica que no contiene ninguno de los dos tipos de aristas mencionados.

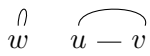


Figura 1.3: Un loop en w y aristas paralelas en u, v.

Un *camino* está formado por vértices ordenados secuencialmente y las aristas en la gráfica unen solo a vértices que son consecutivos.La longitud de un camino o un ciclo está dada por el número de aristas que lo forman. La Figura 1.4 muestra un camino de longitud tres y un ciclo de longitud cinco.

Se dice que una gráfica es conectada si para cualquier par de vértices, dentro de la gráfica, existe al menos un camino entre ellos(Figura 1.5) [9].

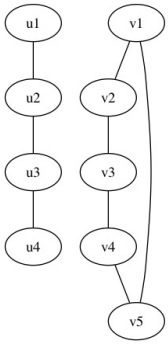
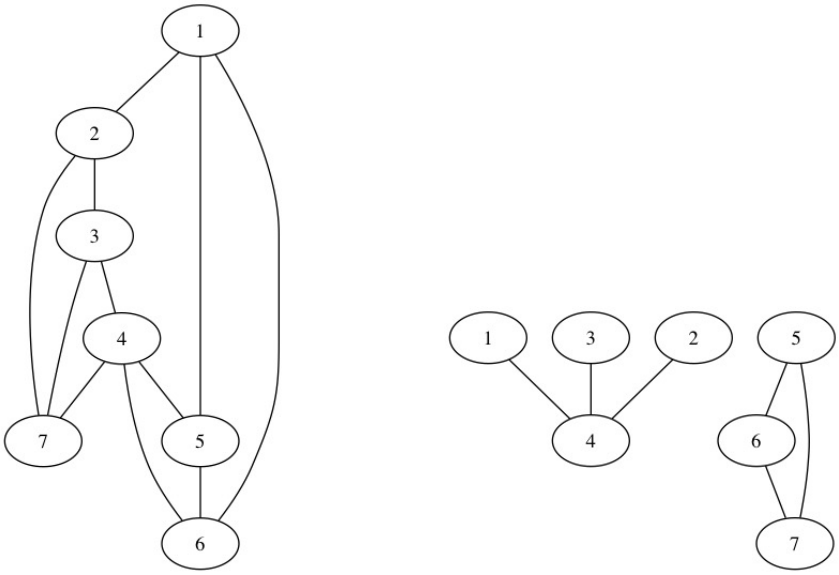


Figura 1.4: Un camino y un ciclo.



(a) (b)

Figura 1.5: (a) Una gráfica conectada, y (b) una gráfica no conectada.

Operaciones en gráficas y subgráficas

Para poder derivar a una gráfica G en gráficas más pequeñas, se tienen dos formas naturales. La primera es obtener una gráfica con $m-1$ aristas, esto quiere decir que se elimina una arista e de la gráfica, $G\backslash e$, en este caso se mantienen todos los demás elementos dentro de G. La segunda es eliminar un vértice v en la gráfica, obteniendo $n-1$ vértices, $G-v$, en este caso se eliminan todas las aristas incidentes a v(Figura 1.6) [9].

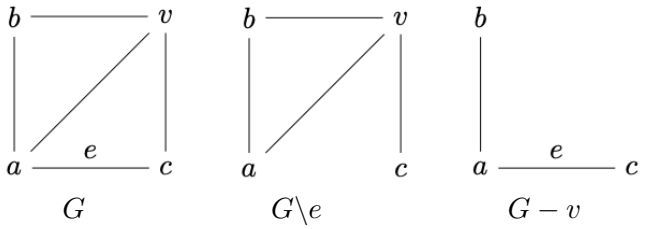


Figura 1.6: G, $G\backslash e$ y $G - v$.

Una gráfica conectada, se dice que es acíclica si entre cualquier par de sus vértices sólo existe un camino, por lo que debe contener al menos un vértice de grado menor a dos.

Una arista de corte es una arista dentro de una gráfica conectada que al ser eliminada, la convierte en una gráfica desconectada [9].

Una subgráfica de expansión de una gráfica G se obtiene eliminando únicamente aristas, por lo que mantiene al conjunto completo de vértices en G. Si el conjunto de aristas eliminadas es S, entonces la subgráfica de expansión se denota $G\backslash S$ [9].

Una gráfica acíclica conectada es llamada un árbol, en una gráfica acíclica desconectada cada componente es acíclico, por esta razón son llamados bosques.

Un árbol de expansión T de una gráfica conectada G es el conjunto de aristas que conectan a los vértices de G de tal forma que solo existe un camino entre cualquier par de vértices, teniendo que $E(T) \subseteq E(G)$ entonces el número de aristas es $E(T) = |V(G)| - 1$.

Si G es una gráfica conectada y no es acíclica, si e es una arista que forma parte en un ciclo de G, entonces si eliminamos a e de la gráfica obtenemos una subgráfica de expansión, porque e no es una arista de corte en G. Si se repite este procedimiento hasta mantener sólo aristas de corte en las subgráficas, entonces el resultado final será un árbol de expansión T de G.

Al conjunto de aristas eliminadas para generar un árbol de expansión se le llama coárbol, al ser $E\backslash T$ se denomina complemento del árbol.

Gráficas cactus

Este tipo de gráficas son simples, conectadas y se caracterizan porque sus ciclos cumplen con las siguientes condiciones (Fig 1.7):

1. Cualquier arista en la gráfica se encuentra a lo más en un ciclo.
2. Cualquier par de ciclos comparten a lo más un vértice.

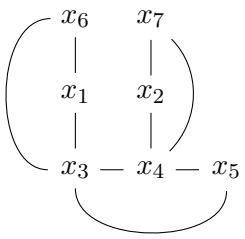


Figura 1.7: Gráfica tipo cactus.

Gráficas outerplanar

Definición 2. *Completa: Una gráfica G completa es aquella en la que todos los vértices tienen grado $d(v) = |V(G)| - 1$ y no existen arístas múltiples o loops, es decir, cada vértice se conecta con todos los demás una sola vez.*

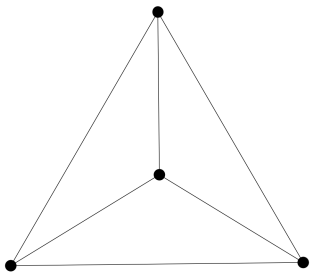


Figura 1.8: Gráfica completa de cuatro vértices.

Definición 3. *Bipartita: Una gráfica G bipartita es aquella en la que sus vértices pueden ser organizados dentro de dos conjuntos X y Y, de tal forma que solo pueden existir aristas que unan al conjunto X con el conjunto Y y no pueden existir aristas que unan vértices dentro de un solo conjunto.*

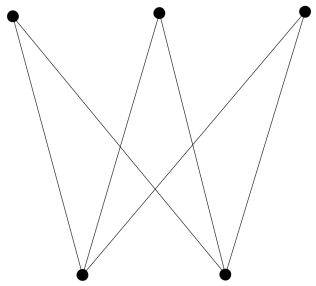


Figura 1.9: Gráfica bipartita con partes de tamaño 2 y 3.

Las gráficas outerplanar son cíclicas, simples y no contienen una subdivisión de las gráficas k_4, que es la gráfica completa de 4 vértices, y la gráfica $k_{2,3}$, que es la gráfica bipartita completa con partes de tamaño 2 y 3. Además de que puede dibujarse sobre un plano con todas las aristas incidentes a una cara exterior de la gráfica(Figura 1.10).

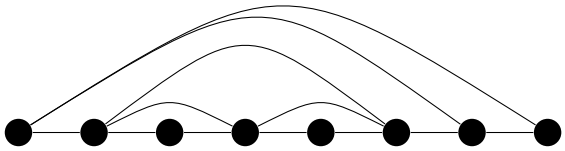


Figura 1.10: Gráfica Outerplanar.

En teoría de gráficas el menor de una gráfica está definido como la subgráfica obtenida mediante contracción de aristas o eliminación de vértices o aristas. Estas últimas dos operaciones se vieron en la sección 1.1.4 como $G\backslash e$ y $G - v$. La operación de contracción de aristas consiste en contraer dos vértices, unidos por una arista, en uno nuevo que mantendrá todas las aristas de unión de los dos vértices anteriores.

Las gráficas outerplanar pueden ser representadas por medio de árboles poligonales definidos como gráficas en las cuales los polígonos que la conforman comparten a lo más un lado con otro polígono, es decir dos polígonos no pueden estar unidos por dos o más lados, esto daría como resultado que contengan a la subgráfica $k_{2,3}$ como un menor de la gráfica. Este tipo de representación permite recorrer todos los vértices de la gráfica si se sigue el contorno de ella (Fig 1.11).

Estas gráficas se pueden representar también como un conjunto de ciclos sobre un camino en el cual no existe cruce de aristas entre cualquier par de ciclos.

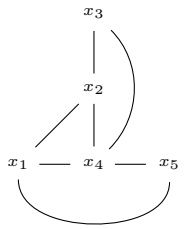


Figura 1.11: Gráfica outerplanar.

Gráfica serial-paralelo

Este tipo de gráficas contienen como subconjunto a las gráficas tipo cactus y outerplanar, debido a que una de sus características es que no contienen como menor a la gráfica k_4, como las gráficas outerplanar, sin embargo si pueden contener a $k_{2,3}$ como menor.

Este tipo de gráficas son construidas bajo dos operaciones:

- Construcción serial: dados dos vértices unidos por una arista, se inserta un vértice nuevo en la arista, generando así dos aristas uniendo a los vértices originales y al vértice agregado.
- Construcción paralela: dados dos vértices unidos por una arista, se agrega una arista nueva uniendo a los vértices originales. Con esta operación es posible generar gráficas con aristas múltiples.

Dadas las operaciones de construcción se observa que este tipo de gráficas no siempre son simples, ya que es posible generar aristas múltiples utilizando la operación de construcción paralela.

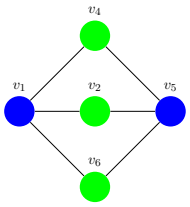


Figura 1.12: Una gráfica serial-paralelo de 5 vértices

1.1.5. Construcción de una gráfica a partir de una fórmula en 2-FNC

Existen algunas representaciones de una fórmula en Forma Normal Conjuntiva mediante su gráfica representativa. En específico para las fórmulas en 2-FNC una de sus representaciones permite relacionar las variables de la fórmula con los vértices de la gráfica y las cláusulas

con las aristas, además de incluir información sobre las literales en cada cláusula. Por ejemplo la fórmula de una sola cláusula $(x \vee \bar{y})$, se representa con los vértices de la gráfica $V(G) = \{x, y\}$ y el conjunto de aristas $E(G) = \{\{x, y\}\}$, además de asociar cada arista en $E(G)$ con un par de signos (s_1, s_2) que son etiquetas en la arista que une a los vértices dentro de la gráfica.

Por ejemplo, la fórmula $(x_1 \vee x_2) \wedge (x_2 \vee \bar{x_4}) \wedge (\bar{x_1} \vee x_3) \wedge (\bar{x_3} \vee \bar{x_4})$ puede representarse mediante la gráfica:

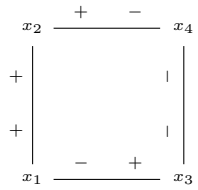


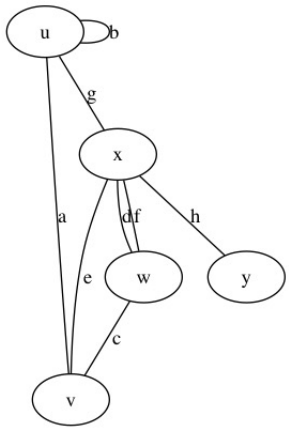
Figura 1.13: Representación de una fórmula mediante una gráfica asociada.

Al poder tener una representación de la gráfica que representa a una 2-FNC es posible aplicar métodos usados en teoría de gráficas para reconocimiento de estructuras. Uno de los métodos más simples para reconocer casos como caminos o árboles, dentro de una gráfica, es la búsqueda primero en profundidad.

Búsqueda primero en profundidad

Este tipo de búsqueda es utilizado para construir un árbol de expansión T dentro de una gráfica y que al construirlo las aristas de retroceso, o ciclos, que se encuentren se pueden organizar en una estructura llamada coárbol $\bar{T}$.

Un auxiliar para poder realizar este procedimiento es obtener la matriz de adyacencia de G, que es de tamaño $n \times n$, y se denota como $A_G := (a_{uv})$, donde a_{uv} sería el número de aristas que unen a los vértices u y v, cada loop cuenta como dos aristas dentro de la matriz(Figura 1.14).



G

	u	v	w	x	y
u	2	1	0	1	0
v	1	0	1	1	0
w	0	1	0	2	0
x	1	1	2	0	1
y	0	0	0	1	0

A

Figura 1.14: Gráfica G y su matriz de adyacencia.

Búsqueda primero en profundidad es un método que se basa en el concepto de árbol de expansión. El proceso consiste en añadir un vértice al árbol T y en cada etapa posterior agregar un vértice vecino al más recientemente agregado a T. Con este método es posible decidir si una arista va a ser agregada a T o a $\bar{T}$.

1.2. Estado del arte

Una propuesta para resolver el problema de satisfactibilidad(SAT) está basada en la aplicación de un método de *s-resolución*. El objetivo en éste método es utilizar dos cláusulas C_1 y C_2 que contienen a la misma literal de forma complementaria, $v \in C_1$ y $\bar{v} \in C_2$, de tal forma que se puede obtener una nueva cláusula $C_3 = C_1 \backslash v \vee C_2 \backslash \bar{v}$. El parámetro s define el tamaño máximo que pueden tener las tres cláusulas involucradas. Este método es presentado por Paturi et. al [10] para aplicarse al problema de k-SAT, con una complejidad en tiempo dada para un valor de $k = 3$, $O(1.3067^n)$, y para $k = 4$, $O(1.4768^n)$.

Sobre ésta idea Hertli et al. [11] propone una mejora al algoritmo presentado por Paturi et al, para resolver k-SAT. En éste método se presenta el uso de la *s-implicación*, la literal v o su complemento $\bar{v}$ es $s-implicada$ si existen al menos s número de cláusulas que impliquen lógicamente a esa literal, por lo que a partir de s cláusulas todas las soluciones obtenibles para una fórmula dada van a contener a la literal v o su complemento. La complejidad en el método que presenta para un valor de $k = 3$ es $O(1.30704^n)$, y para $k = 4$ se tiene $O(1.46899^n)$

Para resolver k-SAT, Thurley [12] presenta un método que se divide en dos partes. Si la fórmula de entrada tiene pocas soluciones, es decir un número de variables que Thurley considera pequeño, realiza la enumeración de cada una de las soluciones utilizando un algoritmo que se basa en el árbol de eliminación. En otro caso utiliza un algoritmo de aproximación para el que indica una probabilidad de $\frac{3}{4}$ de dar el número de modelos de manera correcta. También proporciona la complejidad en tiempo en el caso de $k = 3$ es $O(1.5366^n)$, y para $k = 4$ se tiene $O(1.6155^n)$

Posteriormente Schmitt et al. [13] plantea una mejora en el método de Thurley, con respecto al uso de los árboles de eliminación, en este caso realiza una reducción en los niveles de los árboles generados haciendo uso de las cláusulas en los nodos del árbol y no sólo las variables. La complejidad en tiempo que presenta para $k = 3$ es $O(1.51426^n)$, y para $k = 4$ es $O(1.60816^n)$.

Burchard et al. [14] presenta una herramienta la cual aplica un conjunto de algoritmos exactos para calcular el número de modelos basado en el número de ramas que se satisfacen en el árbol de eliminación. Aplica técnicas como: descomposición en subfórmulas, cache de fórmulas, preprocesado, deducción rápida y conflicto de cláusulas. La mejora que propone es que, al realizar decisiones dentro del árbol y utilizar hilos en cada decisión pueden surgir errores en los valores al hacer uso de cache y por lo tanto propone reglas para el acceso y administración al cache, con lo que soluciona este problema y permite resolverlo de manera paralela. Sin embargo, al ser desarrollado únicamente como una herramienta no presenta la complejidad computacional de ésta, pero se considera el uso de este tipo de técnicas como un aporte sustancial al desarrollo de algoritmos para este problema.

Particularizando el problema general a #2SAT, el algoritmo con la menor complejidad reportada es presentado por Wahlström [5] que realiza el conteo del número de soluciones de peso máximo para fórmulas en 2-FNC. Lo que busca en este algoritmo es descomponer la gráfica hasta poder obtener un grado máximo en la fórmula F, $d(F) = \{3\}$, al obtener estas condiciones busca las variables con una mayor conectividad en la gráfica y que permitan generar gráficas desconectadas durante la descomposición de la fórmula. Al utilizar lo que denomina medidas multivariadas obtiene una complejidad para el algoritmo de $O(1.2377^n)$.

En la caracterización de instancias de MaxSAT y #SAT que puedan ser resueltas en tiempo polinomial, Sæther [15] propone el uso de un parámetro para fórmulas en FNC. Utilizando éste parámetro obtenido mediante el procesamiento de la fórmulas de entrada, y una función de corte sobre la fórmula que la divide en conjuntos independientes de cláusulas y variables, presenta un conjunto de algoritmos para resolver algunas clases de *fórmulas estructuradas* en FNC. Para un conjunto de fórmulas estructuradas F, con n variables m cláusulas y tamaño s, y que puede tener un ordenamiento lineal en sus variables y cláusulas, la complejidad que maneja su método es $O(m^2(m + n)s)$. El ordenamiento lineal que se busca en estas fórmulas proviene de un tipo de gráficas llamadas *bigráficas de intervalo*,

estas gráficas cumplen con la característica de ser *bipartitas* y cada uno de sus vértices representan un intervalo en una línea. Este tipo de gráficas al ser bipartitas tienen una estructura definida en la que no pueden existir ciclos de más de tres vértices.

Para las fórmulas en 2-FNC se puede generar una representación mediante una gráfica en la que las variables son los vértices y las cláusulas son representadas por aristas, permitiendo que el problema de conteo de modelos sea abordado desde el punto de vista de teoría de gráficas, buscando una caracterización en las fórmulas que se representan de esta manera.

Bajo esta perspectiva, en la que se pueden estudiar las fórmulas utilizando un acercamiento basado una estructura diferente, se pueden clasificar fórmulas utilizando parámetros propios de las gráficas, así como relacionar problemas análogos como la cubierta de vértices o el cálculo de conjuntos independientes.

Así como existe el método de conteo de modelos utilizando la enumeración de soluciones mediante un árbol de eliminación, un método basado en la representación de una fórmula mediante su gráfica es presentado por Marcial et al. [16], en dónde muestra un algoritmo exacto para calcular el número de modelos de una 2-FNC utilizando el concepto de árbol de expansión, el algoritmo obtiene un conjunto de aristas que pertenecen a un árbol T de la gráfica y otro conjunto de aristas que pertenecen al coárbol $\bar{T}$. Para determinar el número de modelos utiliza a las aristas de $\bar{T}$ para aplicar un método de descomposición sobre la gráfica. la complejidad de este método se da por $O(2^{|\bar{T}|})$.

López et al. [6] muestra un algoritmo para calcular el número de modelos, para fórmulas en 2-FNC, utilizando la representación en gráfica de la fórmula. El algoritmo divide la fórmula de entrada en particiones a partir del uso del árbol de expansión T y coárbol $\bar{T}$, es decir, subfórmulas que contienen ciclos intersectados (elementos de $\bar{T}$), por lo que el número de modelos se puede calcular de forma independiente en cada partición. La complejidad del algoritmo se mejora con respecto a [16] para fórmulas dispersas, que son fórmulas con un número reducido de ciclos intersectados.

Guillen et al. [17] presenta un método para realizar el conteo de modelos para fórmulas en 2-FNC que pueden representarse mediante una gráfica con topología de malla, con una propuesta que se basa en la matriz de transferencia de Golin et al. [18] para el conteo de conjuntos independientes. El algoritmo presentado para el conteo de modelos mantiene la complejidad en tiempo que se tiene para los conjuntos independientes, y es $O(2^{n^4 \cdot m})$, considerando que en una gráfica tipo malla n y m representan las dos dimensiones de la malla.

Aunque el problema #2SAT es #P-Completo, existen instancias que se pueden resolver en tiempo polinomial [4]. Por ejemplo, si la gráfica que representa la fórmula es acíclica, entonces el número de modelos se puede calcular en tiempo polinomial. Actualmente, los algoritmos que se utilizan para resolver el problema para cualquier fórmula F en 2-FNC,

descomponen F en subfórmulas hasta que se tienen casos base en los que se puede contar de forma eficiente [5] [19] [13].

En el aspecto práctico, están las implementaciones para #SAT (SAT solvers) en donde el objetivo es buscar estrategias que permitan resolver el problema de forma eficiente para instancias en donde el número de variables es considerable. Las herramientas reportadas en la literatura que hasta la fecha se consideran eficientes son relsat [20] y sharpSAT [12] ambas bajo el paradigma secuencial y countAntom [14] utilizando paralelismo. Dentro de estas herramientas existen aquellas que se enfocan en resolver el problema utilizando una forma de descomponer la fórmula para obtener a casos más sencillos de resolver, por ejemplo markSAT [8].

La herramienta relsat esta basada en el algoritmo DPLL (Davis-Putnam algorithm) cuya ventaja es poder procesar rápidamente fórmulas disjuntas, es decir cuyos conjuntos de variables no se intersectan.

La herramienta sharpSAT es una mejora de relsat en donde se elige una literal heurísticamente, así mismo, utiliza descomposición de componentes y cache de componentes para agilizar el cálculo.

La herramienta CountAntom es una implementación paralela basada en sharpSAT cuyos resultados muestran que se mejora el tiempo en las instancias de prueba con respecto a sharpSAT.

La herramienta markSAT es una implementación para resolver #2SAT utilizando descomposición de la fórmula, identificación de particiones de aristas, métodos de resolución polinomial para subfórmulas con estructura árbol, cactus u outerplanar.

1.3. Planteamiento del problema

Dentro de la teoría de la computación el problema $SAT(F)$, donde F es una fórmula booleana, consiste en decidir si F tiene un modelo, es decir, una asignación a las variables de F tal que al ser evaluada con respecto a la lógica booleana clásica devuelva un valor verdadero [1]. Si F esta en 2-FNC entonces $SAT(F)$ se puede resolver en tiempo polinomial, sin embargo, si F esta en k-FNC, $k > 2$, entonces $SAT(F)$ es un problema NP-Completo [2]. Por otro lado, existe el problema de conteo de modelos denotado como $\#SAT(F)$. En este caso $\#SAT(F)$, a diferencia de $SAT(F)$, consiste en contar el número de modelos de la fórmula F. $\#SAT(F)$ pertenece a la clase #P-Completo aún cuando F este en 2-FNC, este último denotado como #2SAT [3].

Dada la dificultad de obtener un algoritmo eficiente para resolver el problema #SAT, es necesario buscar fórmulas cuya representación mediante una gráfica pueda ser caracterizada en una estructura.

1.4. Justificación

Dentro del área de la computación, la complejidad computacional permite el estudio de la "dificultad" en problemas de importancia teórica o práctica. El esfuerzo necesario para el diseño de una solución eficiente a un problema puede variar, ésto depende de cómo se piensa abordar el problema y qué parte de él se pretende resolver.

Algunas de las aplicaciones que tiene el problema de #SAT es poder estimar el nivel de veracidad en teorías proposicionales [21], poder generar alguna explicación a preguntas proposicionales, reparar inconsistencias en bases de datos, inferencia bayesiana, entre otras [1, 22, 23, 21, 24, 25].

1.5. Hipótesis

Existen algoritmos en #2SAT que para gráficas del tipo serial-paralelo y malla reducen la complejidad computacional así como un algoritmo heurístico para gráficas cúbicas que experimentalmente mejora al algoritmo de Walstrom.

1.6. Objetivo general

Obtener un algoritmo que permita utilizar las características de una fórmula para contar sus modelos de manera eficiente.

1.6.1. Objetivos específicos

- Obtener un algoritmo para el conteo de modelos en fórmulas tipo serial paralelo.
- Obtener un algoritmo para el conteo de modelos en fórmulas tipo malla.
- Obtener un algoritmo para el conteo de modelos en fórmulas booleanas en 2-FNC que se consideren cúbicas.

1.7. Metodología propuesta

Para el logro del objetivo planteado, fue necesario seguir el siguiente proceso motodológico:

1. Estudiar los procedimientos actuales para resolver instancias de #2-SAT.
2. Analizar la aplicación de métodos basados en la estructura de gráficas simples, que representan el problema de #2SAT.

3. Proponer un algoritmo para el conteo de modelos en fórmulas tipo serial paralelo en 2-FNC.
4. Proponer un algoritmo para el conteo de modelos en fórmulas tipo malla en 2-FNC.
5. Proponer un algoritmo para conteo de modelos de fórmulas booleanas en 2-FNC que se consideren cúbicas.
6. Establecer la complejidad para cada uno de los algoritmos para el conteo de modelos desarrollados durante la investigación.

Capítulo 2

Publicaciones

Atendiendo a los requisitos establecidos en el Reglamento de Estudios Avanzados vigente (Artículo 60 BIS), para fines de obtención de grado, en este capítulo se presentan los artículos obtenidos y derivados de la investigación realizada.

Es importante mencionar que en este capítulo se incluye únicamente el abstract del artículo publicado.

2.1. A Linear Time Algorithm for Counting #2SAT on Series-Parallel Formulas

Este artículo fue enviado y presentado en el MICAI 2020 (Mexican International Conference on Artificial Intelligence 2020) y publicado como parte de Lecture Notes in Computer Science (LNAI,volume 12468). DOI: https://doi.org/10.1007/978-3-030-60884-2_33. ISBN impreso: 978-3-030-60883-5. ISBN en línea: 978-3-030-60884-2.

Abstract An $O(m+n)$ time algorithm is presented for counting the number of models of a two Conjunctive Normal Form Boolean Formula whose constrained graph is represented by a Series-Parallel graph, where n is the number of variables and m is the number of clauses. To the best of our knowledge, no linear time algorithm has been developed for counting in this kind of formulas.

2.2. Computing the clique-width on series-parallel graphs

Este artículo fue enviado y presentado en el LANMR 2020 (Thirteenth Latin American Workshop on New Methods of Reasoning 2020) y fue publicado en Computación y Sistemas. ISSN: 2007-9737. DOI: 10.13053/cys-26-2-4250.

Abstract The clique-width (cwd) is an invariant of graphs which, similar to other invariants like the tree-width (twd) establishes a parameter for the complexity of a problem. For example, several problems with bounded clique-width can be solved in polynomial time. There is a well known relation between tree-width and clique-width denoted as $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$. Serial-parallel graphs have tree-width of at most 2, so its clique–width is at most 6 according to the previous relation. In this paper, we improve the bound for this particular case, showing that the clique-width of series-parallel graphs is smaller or equal to 5.

2.3. A method for counting models on grid Boolean formulas

Este artículo fue enviado y presentado en el LKE 2021 (8th International Symposium on Language & Knowledge Engineering) y publicado como parte del Journal of Intelligent & Fuzzy Systems, vol. 42, no. 5.(2021). DOI: 10.3233/JIFS-219259. ISSN 1064-1246 (P). ISSN 1875-8967 (E)

Abstract We present a novel algorithm based on combinatorial operations on lists for computing the number of models on two conjunctive normal form Boolean formulas whose restricted graph is represented by a grid graph $G_{m,n}$. We show that our algorithm is correct and its time complexity is $O(\sqrt{t} \cdot 1.618^{\sqrt{t}+2} + t \cdot 1.618^{2\sqrt{t}+4})$, where $t = n \cdot m$ is the total number of vertices in the graph.

For this class of formulas, we show that our proposal improves the asymptotic behavior of the time-complexity with respect of the current leader algorithm for counting models on two conjunctive form formulas of this kind.

2.4. Un algoritmo lineal para el conteo de #2SAT en fórmulas tipo árbol con subfórmulas serial paralelo

Este artículo no fué enviado a evaluación, se realizará un rediseño del método para una generalización sobre algunas topologías.

Un algoritmo lineal para el conteo de #2SAT en fórmulas tipo árbol con subfórmulas serial paralelo

López Medina Marco Antonio[1], Marcial Romero José Raymundo[2], González Ruiz Jacobo Leonardo[3], Hernández Servín José Antonio[4]

Facultad de Ingeniería, Universidad Autónoma del Estado de México. Toluca, México.
[1]mlopezm851@gmail.com, {[2]jrmarcialr, [3]jlgonzalezru, [4]xoseahernandez}@uaemex.mx

Resumen– **Se presenta un algoritmo lineal para realizar el conteo de modelos sobre fórmulas en 2 Forma Normal Conjuntiva cuya representación mediante gráficas esta dada por árboles cuyos nodos interiores u hojas pueden ser subgráficas serial-paralelo.**
Palabras clave: *#2SAT; teoría de la complejidad; teoría de gráficas*

Abstract– **A linear time algorithm is presented for counting the number of models of a two Conjunctive Normal Form Boolean Formula whose constrained graph represents trees whose interior nodes or leaves can be series-parallel subgraphs.**
Keywords: *#2SAT; complexity theory; graph theory*

1 Introducción

$SAT(F)$ es el problema de decisión que consiste en determinar si la fórmula Booleana F tiene una asignación de valores de verdad sobre las variables de F, que al ser evaluada utilizando la lógica Booleana clásica de como resultado $verdadero$. Si la fórmula F está en dos forma normal conjuntiva entonces $SAT(F)$ puede resolverse en tiempo polinomial, sin embargo si F está en k forma normal conjuntiva, $k > 2$, entonces $SAT(F)$ es un problema NP-Completo. $\#SAT(F)$ es la versión de conteo del problema SAT, que consiste en determinar el número de asignaciones sobre las variables de F que al ser evaluadas dan como resultado $verdadero$. $\#SAT(F)$ pertenece a la clase de problemas #P-Completo incluso cuando F está en dos forma normal conjuntiva ($\#2SAT$) [1].

Aunque el problema $\#2SAT$ es #P-Completo, hay casos que pueden resolverse en tiempo polinomial [2] [3]. Por ejemplo, si la representación gráfica de la fórmula es acíclica, entonces el número de modelos se puede calcular en tiempo lineal. Actualmente los métodos para resolver #2SAT se basan en descomponer la fórmula a través de sus variables o cláusulas para obtener fórmulas más simples. El algoritmo reportado con mejor complejidad en tiempo fue presentado por Wahlström [4], la cual es $O(1{,}2377^n)$ (dónde n es el número de variables en la fórmula).

Sobre gráficas serial paralelo el trabajo más cercano al conteo de modelos está relacionado con el reconocimiento de este tipo de gráficas y algunos problemas de decisión.

Schoenmakers [5] muestra un algoritmo de tiempo lineal para reconocimiento de gráficas serial paralelo, calculando una representación *origen-sumidero* de esta gráfica usando un árbol de expansión construido por anchura. La complejidad de construir esta representación es $O(n\sqrt{n})$. Así mismo Eppstein [6] propone un algoritmo para reconocer gáficas serial paralelo dirigidas y no dirigidas, basado en una caracterización llamada *descomposición en oreja*. Por otro lado Takamizawa [7] dice que si una gráfica es restringida a la clase serial paralelo, entonces existen algoritmos de tiempo lineal para problemas de decisión y combinatorios como: cubierta de vértices mínima, conjunto de vértices independientes maximal, subgráfica de linea máxima (inducida), subgráfica outerplanar máxima (inducida), conjunto mínimo de vértices de *retro alimentación*, subgráfica máxima en escalera (inducida), cobertura de trayectoria mínima, entre otros.

En este artículo se presenta un algoritmo para el conteo de modelos en arboles de gráficas serial paralelo, el cual tiene una complejidad lineal.

2 Metodología

2.1 Preliminares

En esta sección se describen conceptos básicos utilizados en el desarrollo del artículo.

2.1.1 Forma Normal Conjuntiva

Siendo $X = \{x_1, ..., x_n\}$ un conjunto de n variables Booleanas (que sólo pueden tener dos posibles valores falso (0) o verdadero (1)). Una literal puede ser la variable x_i, denotada también como x_i^1, o su negación $\bar{x_i}$, denotada como x_i^0.

Una fórmula en forma normal conjuntiva es una conjunción de cláusulas, y las cláusulas son una disyunción de variables.

Una asignación s sobre F es una función Booleana $s : v(F) \to \{0, 1\}$ y definida como:

$$s(x^0) = 1 \text{ si } s(x^1) = 0, \text{ de otra forma }, s(x^0) = 0.$$

Esta asignación puede usarse con conjunciones y disyunciones:

- $s(x \wedge y) = 1$ si $s(x) = s(y) = 1$, de otra forma, $s(x \wedge y) = 0$
- $s(x \vee y) = 0$ si $s(x) = s(y) = 0$, de otra forma, $s(x \vee y) = 1$

Sea F una fórmula Booleana en forma normal conjuntiva, y s una asignación sobre las variables de F, s satisface a F si se cumple que para toda cláusula c en F, $s(c) = 1$.

2.1.2 La gráfica restringida de una 2 Forma Normal Conjuntiva

Existen algunas representaciones gráficas de una forma normal conjuntiva, en este caso se describe la gráfica primal signada (gráfica restringida). Sea F una dos forma normal conjuntiva, su gráfica restringida se denota por $G_F = (V(F), E(F))$, donde los vértices de la gráfica son las variables (V(F) = v(F)) y el conjunto de aristas $E(F)$ se conforma por todas las cláusulas en F, esto es para cada cláusula $\{x_i^\epsilon, x_j^\gamma\} \in F$ existe la arista $\{x_i^\epsilon, x_j^\gamma\} \in E(F)$. Para $x \in V(F)$, $\delta(x)$ denota su grado, el número de aristas incidentes en x. Cada arista $\{x_i^\epsilon, x_j^\gamma\}$ tiene asociado un par (ϵ, γ), que representa cuando las variables x_i o x_j aparecen negadas o no.

2.1.3 Métodos de conteo para #2SAT

En artículos relacionados al conteo de modelos sobre fórmulas en dos forma normal conjuntiva se considera como idea básica el uso de una tupla (α_i, β_i) sobre cada vértice x_i en la gráfica G, dónde α_i representa el número de veces que x_i aparece con una asignación positiva en los modelos de G y β_i el número de veces que aparece negaiva.

Existen métodos reportados para el conteo de modelos de una dos forma normal conjuntiva F [8], por ejemplo:

Si la gráfica representa un camino de la forma $P_n = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_2^{\epsilon_2}, x_3^{\gamma_2}\}, \cdots \{x_{n-1}^{\epsilon_{n-1}}, x_n^{\gamma_{n+1}}\}\}$ de n vértices, el número de modelos está dado por la suma de los elementos (α_n, β_n). Donde $(\alpha_1, \beta_1) = (1, 1)$ y la tupla para los demás vértices se calcula según los signos de acuerdo a la siguiente recurrencia:

$$(\alpha_i, \beta_i) = \begin{cases} (\alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}) & if\ (\epsilon_{i-1}, \gamma_{i-1}) = (1, 1) \\ (\alpha_{i-1}, \alpha_{i-1} + \beta_{i-1}) & if\ (\epsilon_{i-1}, \gamma_{i-1}) = (1, 0) \\ (\alpha_{i-1} + \beta_{i-1}, \beta_{i-1}) & if\ (\epsilon_{i-1}, \gamma_{i-1}) = (0, 1) \\ (\beta_{i-1}, \alpha_{i-1} + \beta_{i-1}) & if\ (\epsilon_{i-1}, \gamma_{i-1}) = (0, 0) \end{cases} \quad (1)$$

Si $\{x_i^{\epsilon_1}, x_j^{\gamma_1}\}$ y $\{x_i^{\epsilon_2}, x_j^{\gamma_2}\}$ son dos cláusulas paralelas en la fórmula F, el número de modelos para estas cláusulas es dado por:

$$(\alpha_j, \beta_j) = \begin{cases} (\alpha_i, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (1, 0) \\ (\alpha_i + \beta_i, 0) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (0, 1) \\ (\beta_i, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (0, 0) \\ (\alpha_i, \beta_i) & if\ (\epsilon_1, \gamma_1) = (1, 0)\ and\ (\epsilon_2, \gamma_2) = (0, 1) \\ (0, \alpha_i + \beta_i) & if\ (\epsilon_1, \gamma_1) = (1, 0)\ and\ (\epsilon_2, \gamma_2) = (0, 0) \\ (\beta_i, \beta_i) & if\ (\epsilon_1, \gamma_1) = (0, 1)\ and\ (\epsilon_2, \gamma_2) = (0, 0) \end{cases} \quad (2)$$

Si $\{x_i^{\epsilon_1}, x_j^{\gamma_1}\}$, $\{x_i^{\epsilon_2}, x_j^{\gamma_2}\}$ y $\{x_i^{\epsilon_3}, x_j^{\gamma_3}\}$ son tres cláusulas paralelas en la fórmula F, el número de modelos para las cláusulas se calcula:

$$(\alpha_j, \beta_j) = \begin{cases} (0, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (1, 0) \\ & and\ (\epsilon_3, \gamma_3) = (0, 0) \\ (\alpha_i + \beta_i, 0) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (1, 0) \\ & and\ (\epsilon_3, \gamma_3) = (0, 1) \\ (\beta_i, \alpha_i) & if\ (\epsilon_1, \gamma_1) = (1, 1)\ and\ (\epsilon_2, \gamma_2) = (0, 1) \\ & and\ (\epsilon_3, \gamma_3) = (0, 0) \\ (\alpha_i, \beta_i) & if\ (\epsilon_1, \gamma_1) = (0, 1)\ and\ (\epsilon_2, \gamma_2) = (1, 0) \\ & and\ (\epsilon_3, \gamma_3) = (0, 0) \end{cases} \quad (3)$$

Para realizar el conteo de modelos en gráficas tipo árbol se necesita hacer un recorrido primero en profundidad del árbol, es decir, el proceso de conteo de modelos se realiza de las hojas a la raíz, por eso es necesario hacer el recorrido en postorden, asegurando que al llegar a la raíz se han contado los modelos sobre todos los demás vértices. El procedimiento durante el recorrido es como sigue:

Algorithm 1: Algoritmo para conteo de modelos sobre árboles, utilizando los valores asignados a los vértices.

Result: #SAT(F) en v_i

1. Si el vértice v_j es hoja, los valores asignados son $(\alpha_j, \beta_j) = (1, 1)$.
2. Dado el vértice v_j, teniendo un número k de hijos $v_i : 1 \leq i \leq k$, Los modelos se calculan con respecto a las aristas que unen a cada v_i con v_j utilizando la ecuación 1 y se multiplican los valores (α_i, β_i) obtenidos en cada operación.

$$(\alpha_j, \beta_j) = (\Pi_{i=1}^k \alpha_i, \Pi_{i=1}^k \beta_i) \quad (4)$$

Además de asociar una tupla (α, β) a cada vértice, se asociará un elemento direccional a cada arista de la gráfica.

2.2 Gráficas serial paralelo

Las gráficas serial paralelo cumplen con las siguientes características:

- No contienen como subdivisión a la gráfica completa de cuatro vértices k_4.
- Si es una gráfica simple, debe contener al menos un vértice de grado 2.
- Se construyen a través de dos operaciones:

- Composición serial: Se divide una arista en la gráfica en dos, incluyendo un vértice nuevo en la gráfica. Dada la arista $e_i = (x, y)$, al dividirse, y agregar el nuevo vértice z, se generan las aristas $e_j = (x, z)$ y $e_k = (z, y)$, sustituyendo la arista original en la gráfica.
- Composición paralela: Se duplica una arista en la gráfica. Dada la arista $e_i = (x, y)$, se genera una nueva arista $e_j = (x, y)$ y se agrega a la gráfica.

2.3 Descomposición de gráficas serial paralelo

Una gáfica serial paralelo se puede deconstruir usando las operaciones de composición de manera inversa como se demuestra en [9] y se resume a continuación.

2.3.1 Descomposición serial

La operación de descomposición serial la definimos como una *contracción* de aristas, en la cual al tener un vértice x_j de grado 2 en la gráfica, $\delta(x_j) = 2$, se toman las dos aristas que lo contienen $e_1 = \{x_i, x_j\}$ y $e_2 = \{x_j, x_k\}$, y se construye la nueva arista $e_3 = \{x_i, x_k\}$. Esta operación permite contraer caminos de n vértices, $P_n = \{\{x_1, x_2\}, \{x_2, x_3\}, ..., \{x_{n-1}, x_n\}\}$, a una sola arista $\{x_1, x_n\}$.

2.3.2 Descomposición paralela

La operación de descomposición paralela la definimos como una *unión* de aristas, y consiste en reducir un número finito de aristas paralelas uniéndolas en una sola arista, que las represente dentro de la gráfica. Las gráficas serial paralelo permiten que existan m caminos, entre un mismo par de vértices, y al aplicar la descomposición serial se obtienen un conjunto de m aristas uniendo dos vértices. Al contraer un conjunto de aristas $e_l = \{x_{i_l}, x_{j_l}\}$, $1 \leq l \leq m$, se busca obtener una arista que las representa $\{x_i, x_j\}$.

3 Resultados

En esta sección se muestran las estructuras, métodos y ecuaciones desarrolladas para poder llevar a cabo las operaciones y el conteo de modelos.

3.1 Estructuras específicas

A continuación se definen la estructuras necesarias para llevar a cabo el método de conteo.

3.1.1 Elemento direccional

Para una cláusula $\{x_i^{\epsilon}, x_j^{\gamma}\}$, una cuádrupla $C_{x_i x_j}$ es asociada a la arista que la representa en la gráfica G. Una cuádrupla $C_{x_i x_j} = \{a_{ij}, b_{ij}, a'_{ij}, b'_{ij}\}$ representa los modelos para las asignaciones posibles de la literal x_j. Inicialmente los valores para esta cuádrupla son tres elementos unitarios y un cero, los cuales representan los tres posibles modelos que una cláusula de dos variables tiene asociada, como se presenta en la ecuación 5:

$$C_{x_i x_j} = \begin{cases} (1,1,1,0) \;\; if \;\; (\epsilon,\gamma) = (1,1) \\ (1,0,1,1) \;\; if \;\; (\epsilon,\gamma) = (1,0) \\ (1,1,0,1) \;\; if \;\; (\epsilon,\gamma) = (0,1) \\ (0,1,1,1) \;\; if \;\; (\epsilon,\gamma) = (0,0) \end{cases} \tag{5}$$

Esto se puede observar directamente con un ejemplo. Dada la cláusula $(x_1 \vee \bar{x_2})$, la variable x_1 en disyunción con la negación de la variable x_2, la tabla de verdad que la representa es:

x_1	x_2	$(x_1 \vee \bar{x_2})$
0	0	1
0	1	0
1	0	1
1	1	1

Figura 1: Valores de verdad dados para la cláusula $(x_1 \vee \bar{x_2})$.

Para las posibles combinaciones de signos restantes se puede crear su tabla de manera similar.

3.1.2 Aristas extendidas

El conjunto de aristas extendidas se contruye a partir de las aristas originales de la gráfica, para toda arista $e = \{x_i, x_j\} \in E(G)$ con signos (ϵ, γ), existe la arista extendida $e' = \{x_i, x_j, C_{x_i x_j}\} \in E'(G)$, donde $C_{x_i x_j}$ se calcula con la ecuación 5.

3.1.3 Vértices extendidos

El conjunto de vértices extendidos se construye con los vértices originales de la gráfica. para todo vértice $x_i \in V(G)$, existe $x_i \in V'(G)$ con una tupla asociada (α_i, β_i). Los valores de α_i y β_i deben ser inicializados con 1.

3.2 Descomposición de gráficas tipo árbol

El objetivo de la descomposición de un árbol es poder reducirlo desde sus hojas hacia la raíz, en este caso se busca poder realizar esta reducción durante la construcción post orden árbol.

3.2.1 Reducción de vértices *hoja*

Dentro de las gráficas tipo árbol o camino, siempre existen vértices de grado 1, y esta operación de *reducción* busca poder eliminar este tipo de vértices de la gráfica. Dado un vértice $x_j \in V(G)$, $\delta(x_j) = 1$, y la arista incidente a el $e_r = \{x_i, x_j\}$, esta operación busca eliminar e_r y x_j de G, $E(G)\backslash e_r$ y $V(G)\backslash x_j$. El objetivo de esta operación es poder llegar a tener un solo elemento en el conjunto de vértices $V(G)$, y que el conjunto $E(G)$ sea vacío.

3.3 Conteo sobre una contracción de aristas

Como primera operación para resolver el conteo en las gráficas serial paralelo tenemos la *contracción* de aristas. Esto quiere decir si se tienen dos aristas extendidas, $e'_1 = \{x_i, x_j, C_{x_i x_j}\}$ y $e'_2 = \{x_j, x_k, C_{x_j x_k}\}$, además de un vértice extendido x_j con la tupla asociada (α_j, β_j). Al realizar la contracción con las aristas e'_1, e'_2 se generará una nueva arista e'_{1-2} uniendo x_i con x_k, $e'_{1-2} = \{x_i, x_k, C_{x_i x_k}\}$, en la cual se calcula el nuevo elemento direccional $C_{x_i x_k}$ con las siguientes ecuaciones:

$$\pi_1(C_{x_i x_k}) = \pi_1(C_{x_j x_k})\pi_1(C_{x_i x_j})\alpha_j + \pi_2(C_{x_j x_k})\pi_3(C_{x_i x_j})\beta_j \quad (6)$$

$$\pi_2(C_{x_i x_k}) = \pi_1(C_{x_j x_k})\pi_2(C_{x_i x_j})\alpha_j + \pi_2(C_{x_j x_k})\pi_4(C_{x_i x_j})\beta_j \quad (7)$$

$$\pi_3(C_{x_i x_k}) = \pi_3(C_{x_j x_k})\pi_1(C_{x_i x_j})\alpha_j + \pi_4(C_{x_j x_k})\pi_3(C_{x_i x_j})\beta_j \quad (8)$$

$$\pi_4(C_{x_i x_k}) = \pi_3(C_{x_j x_k})\pi_2(C_{x_i x_j})\alpha_j + \pi_4(C_{x_j x_k})\pi_4(C_{x_i x_j})\beta_j \quad (9)$$

Donde $\pi_l(C_{x_i x_j})$ es la función de proyección sobre el elemento l de la cuádrupla.

Por ejemplo, dadas las aristas
$e'_1 = \{x_1, x_2, \{1, 0, 1, 1\}\}$ y $e'_2 = \{x_2, x_3, \{1, 1, 1, 0\}\}$ sobre el vértice x_2 con la tupla asociada $(\alpha_2, \beta_2) = (1, 1)$

Generar la arista e'_{1-2} da el elemento direccional $C_{x_1 x_3} = \{2, 1, 1, 0\}$, que como se vió en la sección anterior, $\pi_1(C_{x_1 x_3})$ y $\pi_2(C_{x_1 x_3})$ representan los modelos en los que x_3 aparece positiva, y $\pi_3(C_{x_1 x_3})$ y $\pi_4(C_{x_1 x_3})$ en los que aparece negativa. El total de modelos en la cláusula e'_{1-2} está dado por la suma de los elementos de la cuádrupla $C_{x_1 x_3}$. Tomando la tabla de verdad de estas dos cláusulas:

x_1	x_2	x_3	$(x_1 \vee \bar{x_2}) \wedge (x_2 \vee x_3)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figura 2: Tabla de verdad de la fórmula $(x_1 \vee \bar{x_2}) \wedge (x_2 \vee x_3)$.

Al aplicar el procedimiento de *contracción* en las dos aristas, tenemos como resultado la siguiente tabla:

x_1	x_3	$e_{1-2} = \{(x_1 \vee \bar{x_2}) \wedge (x_2 \vee x_3)\}$
0	0	0
0	1	1
0	0	0
0	1	0
1	0	0
1	1	1
1	0	1
1	1	1

Figura 3: Tabla de verdad de la fórmula $(x_1 \vee \bar{x_2}) \wedge (x_2 \vee x_3)$ eliminando a x_2 de ella.

Se puede observar que la contracción de las cláusulas permite omitir la columna que representa a la variable en común x_2. Al poder realizar esta *simplificación* de la fórmula, eliminando variables de ella, se reduce su tamaño y por lo tanto también se reduce el tiempo de procesamiento en algoritmos posteriores.

Lema 1. *Sea F una fórmula que representa un camino $P_n = \{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_2^{\epsilon_2}, x_3^{\gamma_2}\} \cdots \{x_{n-1}^{\epsilon_{n-1}}, x_n^{\gamma_{n-1}}\}\}$, dónde la tupla asociada (α_i, β_i) corresponde al elemento x_i, si se aplica la regla de contracción sobre los vértices $x_2, ...x_{n-1}$, hasta obtener una tripleta $\{x_1, x_n, C_{x_1 x_n}\}$, entonces:*

$$\#2SAT(F) = \pi_1(C_{x_1 x_n})\alpha_n + \pi_2(C_{x_1 x_n})\beta_n + \pi_3(C_{x_1 x_n})\alpha_n + \pi_4(C_{x_1 x_n})\beta_n \quad (10)$$

Demostración. Inicialmente $\alpha_i = \beta_i = 1$, comparándolo con el método de conteo conocido en 1 y de manera inductiva. Teniendo la cláusula simple $\{x_1^{\epsilon}, x_2^{\gamma}\}$ el número de modelos corresponde a los calculados en la ecuación 1 y como se muestra en la figura 4 y en la figura 5 la cuádrupla correspondiente, en ambos casos los modelos calculados son tres.

$$\overset{(1,1)}{x_1^{\epsilon}} \text{ ——————— } \overset{(2,1)}{x_2^{\gamma}}$$

Figura 4: Camino P_2 usando la ecuación 1, $\#SAT(F) = \alpha_2 + \beta_2 = 2 + 1 = 3$.

$$\overset{(1,1)}{x_1} \overset{C_{x_1x_2}(1,1,1,0)}{\text{———————}} \overset{(1,1)}{x_2}$$

Figura 5: Camino P_2 usando conteo sobre contracción de aristas, $\#SAT(F) = \pi_1(C_{x_1x_2}) + \pi_2(C_{x_1x_2}) + \pi_3(C_{x_1x_2}) + \pi_4(C_{x_1x_2}) = 1 + 1 + 1 + 0$.

En un camino P_3 con aristas $\{\{x_1^{\epsilon_i}, x_2^{\gamma_i}\}, \{x_2^{\epsilon_j}, x_3^{\gamma_j}\}\}$ el número de modelos es 5 si $(\gamma_i = \epsilon_j)$ o 4 si $(\gamma_i \neq \epsilon_j)$.

$$\overset{(1,1)}{x_1} \overset{(1,1,1,0)}{\text{———}} \overset{(1,1)}{x_2} \overset{(1,1,1,0)}{\text{———}} \overset{(1,1)}{x_3} \rightarrow \overset{(1,1)}{x_1} \overset{(2,1,1,1)}{\text{———}} \overset{(1,1)}{x_3}$$

Figura 6: Camino P_3 aplicando contracción sobre las aristas de x_2.

Por hipótesis de inducción $\alpha_n + \beta_n = \pi_1(C_{x_1x_n}) + \pi_2(C_{x_1x_n}) + \pi_3(C_{x_1x_n}) + \pi_4(C_{x_1x_n})$, por demostrar que $\alpha_{n+1} + \beta_{n+1} = \pi_1(C_{x_1x_{n+1}}) + \pi_2(C_{x_1x_{n+1}}) + \pi_3(C_{x_1x_{n+1}}) + \pi_4(C_{x_1x_{n+1}})$.

Si $\epsilon = 1$ y $\gamma = 1$, entonces al aplicar la ecuación 1 $\alpha_{n+1} = \alpha_n + \beta_n$ y $\beta_{n+1} = \alpha_n$. Por otro lado $\pi_1(C_{x_nx_{n+1}}) = 1$, $\pi_2(C_{x_nx_{n+1}}) = 1$, $\pi_3(C_{x_nx_{n+1}}) = 1$, $\pi_4(C_{x_nx_{n+1}}) = 0$, y $\pi_1(C_{x_1x_n}) + \pi_2(C_{x_1x_n}) = \alpha_n$ $\pi_3(C_{x_1x_n}) + \pi_4(C_{x_1x_n}) = \beta_n$.

Por lo tanto al aplicar las ecuaciones 6, 7, 8 y 9, el valor de

$\pi_1(C_{x_1x_{n+1}}) = \pi_1(C_{x_1x_n}) + \pi_3(C_{x_1x_n}) =$
$\alpha_n - \pi_2(C_{x_1x_n}) + \beta_n - \pi_4(C_{x_1x_n})$ y para

$\pi_2(C_{x_1x_{n+1}}) = \pi_2(C_{x_1x_n}) + \pi_4(C_{x_1x_n}) =$
$\alpha_n - \pi_1(C_{x_1x_n}) + \beta_n - \pi_3(C_{x_1x_n})$, ahora el valor

$\alpha_{n+1} = \pi_1(C_{x_1x_{n+1}}) + \pi_2(C_{x_1x_{n+1}}) =$
$\alpha_n - \pi_2(C_{x_1x_n}) + \beta_n - \pi_4(C_{x_1x_n}) + \alpha_n - \pi_1(C_{x_1x_n}) + \beta_n - \pi_3(C_{x_1x_n}) = 2\alpha_n + 2\beta_n - (\pi_1(C_{x_1x_n}) + \pi_2(C_{x_1x_n})) - (\pi_3(C_{x_1x_n}) + \pi_4(C_{x_1x_n})) = 2\alpha_n + 2\beta_n - \alpha_n - \beta_n = \alpha_n + \beta_n$.

Al aplicar la ecuaciones el valor de

$\pi_3(C_{x_1x_{n+1}} = \pi_1(C_{x_1x_n}) = \alpha_n - \pi_2(C_{x_1x_n})$ y para $\pi_4(C_{x_1x_{n+1}}) = \pi_2(C_{x_1x_n}) = \alpha_n - \pi_1(C_{x_1x_n})$, ahora el valor de $\beta_{n+1} = \pi_3(C_{x_1x_{n+1}} + \pi_4(C_{x_1x_{n+1}}) = \alpha_n - \pi_2(C_{x_1x_n}) + \alpha_n - \pi_1(C_{x_1x_n}) = 2\alpha_n - (\pi_1(C_{x_1x_n} + \pi_2(C_{x_1x_n}) = 2\alpha_n - \alpha_n = \alpha_n$. En resumen $\alpha_{n+1} = \alpha_n + \beta_n$ y $\beta_{n+1} = \alpha_n$.

El paso inductivo es un análisis sobre los valores de ϵ y γ

□

Ejemplo, sea F=$\{\{x_1^1, x_2^1\}, \{x_2^1, x_3^0\}, \{x_3^0, x_4^1\}, \{x_4^0, x_5^0\}\}$ una fórmula cuya gráfica restringida es un camino. Las tripletas para cada cláusula son: $\{x_1, x_2, (1,1,1,0)\}, \{x_2, x_3, (1,0,1,1)\}, \{x_3, x_4, (1,1,0,1)\}$, $\{x_4, x_5, (0,1,1,1)\}$.

La representación que usaremos para las cuádruplas es la siguiente:

$$\overset{(1,1)}{x_1} \overset{(1,1,1,0)}{\text{———}} \overset{(1,1)}{x_2} \overset{(1,0,1,1)}{\text{———}} \overset{(1,1)}{x_3} \overset{(1,1,0,1)}{\text{———}} \overset{(1,1)}{x_4} \overset{(0,1,1,1)}{\text{———}} \overset{(1,1)}{x_5}$$

Primero necesitamos contraer $\{x_1^1, x_2^1\}$ y $\{x_2, x_3^0\}$ en una arista nueva de x_1 a x_3. Usando las ecuaciones (6-9) obtenemos las nuevas cuádruplas.

$$\overset{(1,1)}{x_1} \overset{(1,1,2,1)}{\text{———}} \overset{(1,1)}{x_3} \overset{(1,1,0,1)}{\text{———}} \overset{(1,1)}{x_4} \overset{(0,1,1,1)}{\text{———}} \overset{(1,1)}{x_5}$$

Ahora se hace la contracción sobre las aristas de x_3, obteniendo:

$$\overset{(1,1)}{x_1} \overset{(3,2,2,1)}{\text{———}} \overset{(1,1)}{x_4} \overset{(0,1,1,1)}{\text{———}} \overset{(1,1)}{x_5}$$

Finalmente con la contracción en las aristas de x_4 se obtiene:

$$\overset{(1,1)}{x_1} \overset{(2,1,5,3)}{\text{———}} \overset{(1,1)}{x_5}$$

El número de modelos de la fórmula original se obtiene sumando los cuatro elementos de la cuádrupla, multiplicando por (α_5, β_5):

$\#2SAT(F) = 2 * 1 + 1 * 1 + 5 * 1 + 3 * 1$, dando como resultado 11 modelos.

3.4 Conteo en una unión de aristas

Si $e_1, ..., e_h$ son las aristas paralelas sobre el mismo par de vértices x_i, x_j, $e'_l = \{x_i, x_j, C_{x_ix_j}\} : 1 \leq l \leq h$. La unión puede llevarse a cabo con la ecuación:

$$\pi_k(C_{x_ix_j}) = \prod_{l=1}^{h} \pi_k(C_{x_ix_j}) \tag{11}$$

Donde $k = 1, 2, 3, 4$, y se puede formar la nueva arista $e_{1-h} = \{x_i, x_j, C_{x_ix_j}\}$

Lema 2. *Sea F una fórmula representada por una gráfica con dos o más cláusulas paralelas e.g. F=$\{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_1^{\epsilon_2}, x_2^{\gamma_2}\}\}$ o la fórmula*

F=$\{\{x_1^{\epsilon_1}, x_2^{\gamma_1}\}, \{x_1^{\epsilon_2}, x_2^{\gamma_2}\}, ..., \{x_1^{\epsilon_m}, x_2^{\gamma_m}\}\}$*, se aplica la unión de aristas hasta obtener la arista que las represente a todas* $(x_1, x_2, C_{x_1x_2}\}$ *entonces:*

$$\#2SAT(F) = \pi_1(C_{x_1x_2})\alpha_2 + \pi_2(C_{x_1x_2})\beta_2 + \pi_3(C_{x_1x_2})\alpha_2 + \pi_4(C_{x_1x_2})\beta_2 \quad (12)$$

Demostración. Un caso de análisis simple comparando los resultados de los algoritmos conocidos de las ecuaciones 2 y 3.El resultado para dos cláusulas paralelas se muestra en la figura 7 (mostrando sólo el elemento direccional $C_{x_ix_j}$) y su unión en la figura 8.

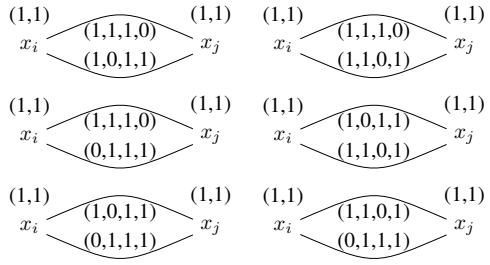


Figura 7: Casos paralelos sobre dos cláusulas.

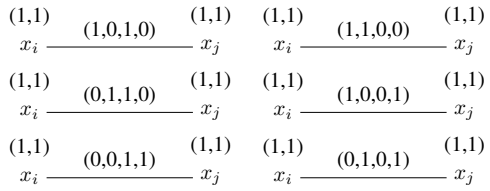


Figura 8: Aplicando conteo sobre unión de aristas en los casos de dos cláusulas.

De la ecuación 2 el par (α_i, β_i) obtenido en casos de dos cláusulas paralelas $(x_i^{\epsilon_1}, x_j^{\gamma_1})$ y $(x_i^{\epsilon_2}, x_j^{\gamma_2})$ son $(2, 0)$ si $(\epsilon_1, \gamma_1) = (1, 1)$ y $(\epsilon_2, \gamma_2) = (0, 1)$, Es $(0, 2)$ si $(\epsilon_1, \gamma_1) = (1, 0) y (\epsilon_2, \gamma_2) = (0, 0)$, y $(1, 1)$ en los demás casos. Para el análisis sobre tres cláusulas es similar.

□

3.5 Conteo sobre una reducción de vértice

Para realizar el conteo de modelos en gráficas tipo árbol se necesita hacer un recorrido post orden del árbol (no necesariamente binario), es decir, el proceso de conteo de modelos se realiza de las hojas a la raíz. El procedimiento durante el recorrido es como sigue:

Algorithm 2: Algoritmo para conteo de modelos sobre árboles, utilizando aristas y vértices extendidos.

Result: #SAT(F) en x_i

1. Si el vértice asociado es hoja los valores asociados al vértice son $(\alpha_j, \beta_j) = (1, 1)$;
2. Dado el vértice x_i, teniendo un número k de hijos $x_j : 1 \leq j \leq k$, y el conjunto de aristas con elemento direccional $C_{x_jx_i}$, los valores (α_i, β_i) se calculan usando las ecuaciones 13 y 14;

$$\alpha_i = \Pi_{j=1}^{k}(\pi_1(C_{x_jx_i})\alpha_j + \pi_2(C_{x_jx_i})\beta_j) \quad (13)$$

$$\beta_i = \Pi_{j=1}^{k}(\pi_3(C_{x_jx_i})\alpha_j + \pi_4(C_{x_jx_i})\beta_j) \quad (14)$$

Lema 3. *Sea F una fórmula representada por una gráfica tipo árbol, con raíz* x_i*, entonces: #SAT(F) se calcula con el algoritmo 2*

Demostración. Ya que el algoritmo 1 es correcto, la prueba consiste en comparar que la salida del algoritmo 2 es equivalente a la salida del algoritmo 1.

Ya que los signos en la recurrencia 1 para el algoritmo 1, se reemplazan por las cuádruplas en el algoritmo 2, entonces las operaciones son equivalentes.

□

Ejemplo, un caso de análisis sobre una gráfica tipo árbol con cinco vértices usando la ecuación 4, y las ecuaciones 13 y 14 se muestran en las figuras 9 y 10.

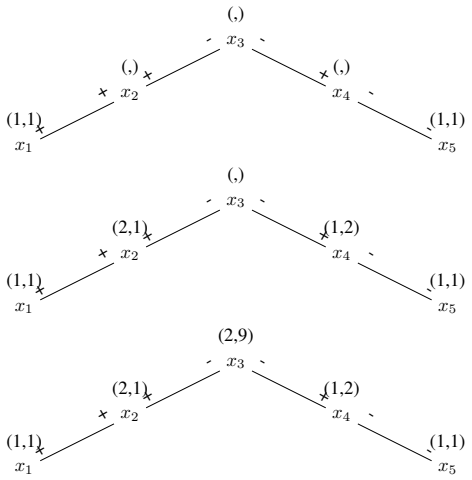


Figura 9: Representación y conteo sobre una gráfica tipo árbol con cinco vértices, tomando como raíz x_3, usando la ecuación 4.

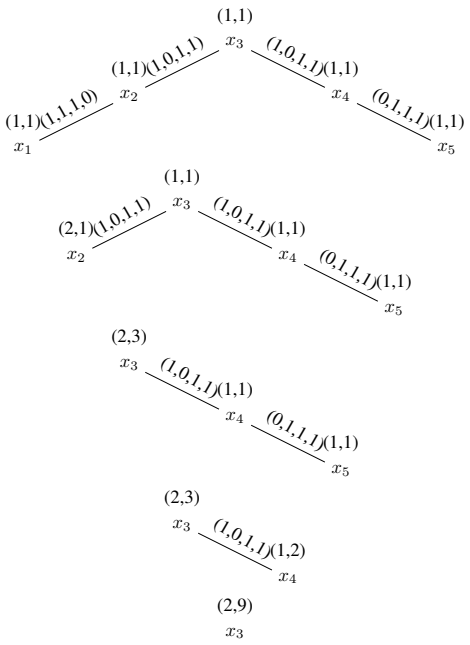


Figura 10: Representación y conteo sobre una gráfica tipo árbol con cinco vértices, tomando como raíz x_3, usando las ecuaciones 13 y 14.

3.6 Conteo sobre árboles con subgráficas serial paralelo

El método para realizar el conteo sobre árboles con subgráficas tipo serial paralelo, sigue los siguientes pasos:

- Identificar el árbol de la gráfica, para saber cuál será el vértice raíz.
- Identificar los puntos *origen* y *sumidero* de las subgráficas serial paralelo dentro del árbol.
- Reducir las subgráficas serial paralelo con los métodos presentados en las secciones 3.3 y 3.4.
- Recorrer el árbol y aplicar el algoritmo 2.

Teorema 1. *Si F es una fórmula tipo árbol con subfórmulas tipo serial paralelo, entonces el método de conteo calcula $\#2SAT(F)$ de manera correcta.*

Demostración. Por los lemmas 1 y 2 se demuestra el conteo sobre gráficas serial paralelo y por el lemma 3 el conteo sobre gráficas tipo árbol. □

3.7 Ejemplo

Ahora usaremos un ejemplo para aplicar nuestro algoritmo.

Dada la fórmula F=$\{\{x_1^1, x_3^1\}, \{x_{10}^1, x_1^1\}, \{x_2^1, x_3^1\}, \{x_{10}^1, x_2^1\}, \{x_4^1, x_3^1\}, \{x_3^1, x_8^1\}, \{x_{10}^1, x_4^1\}, \{x_9^1, x_{10}^1\}, \{x_7^1, x_{10}^1\}, \{x_6^1, x_5^1\}, \{x_5^1, x_{10}^1\}, \{x_{12}^1, x_{13}^1\}, \{x_{12}^1, x_{14}^1\}, \{x_{12}^1, x_{11}^1\}, \{x_{11}^1, x_8^1\}, \{x_{14}^1, x_8^1\}, \{x_{13}^1, x_8^1\}\}$

Usaremos la representación gráfica de la fórmula usando vértices extendidos y el conjunto de aristas extendidas $E' = \{\{x_1, x_3, (1,1,1,0)\}, \{x_{10}, x_1, (1,1,1,0)\}, \{x_2, x_3, (1,1,1,0)\}, \{x_{10}, x_2, (1,1,1,0)\}, \{x_4, x_3, (1,1,1,0)\}, \{x_3, x_8, (1,1,1,0)\}, \{x_{10}, x_4, (1,1,1,0)\}, \{x_9, x_{10}, (1,1,1,0)\}, \{x_7, x_{10}, (1,1,1,0)\}, \{x_6, x_5, (1,1,1,0)\}, \{x_5, x_{10}, (1,1,1,0)\}, \{x_{12}, x_{13}, (1,1,1,0)\}, \{x_{12}, x_{14}, (1,1,1,0)\}, \{x_{12}, x_{11}, (1,1,1,0)\}, \{x_{11}, x_8, (1,1,1,0)\}, \{x_{14}, x_8, (1,1,1,0)\}, \{x_{13}, x_8, (1,1,1,0)\}\}$:

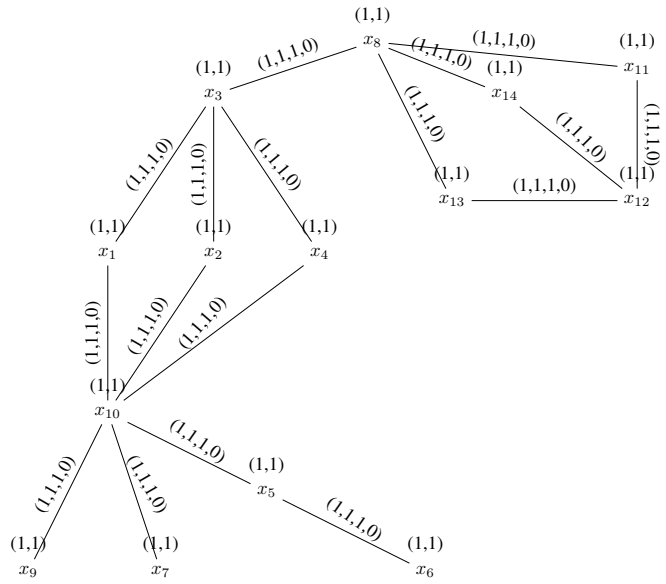


Figura 11: Representación gráfica de F.

Tomando la representación gráfica de la figura 11, tomaremos como raíz de la gráfica a x_8 y tenemos dos subgráficas serial paralelo, la primera tiene como vértices *origen* y *sumidero* a los vértices x_{10} y x_3 respectivamente, y la segunda tiene como vértices *origen* y *sumidero* a x_{12} y x_8.

El proceso de reducción de estas subgráficas serial paralelo se muestra en los lemas 1 y 2, y el resultado se ilustra en la figura 12.

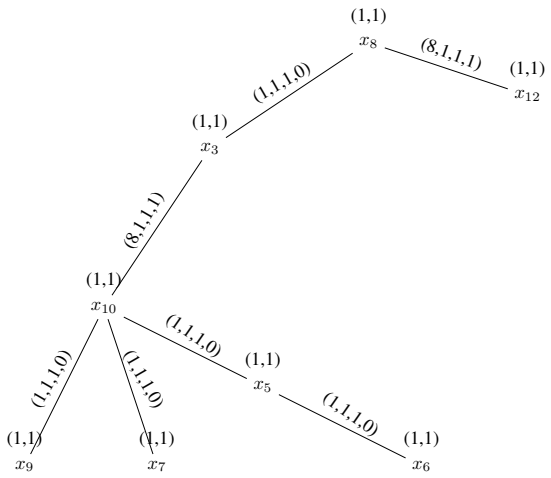


Figura 12: Representación gráfica de F al aplicar la reducción de las subgráficas serial paralelo.

Posteriormente se realiza el recorrido del árbol aplicando el conteo por reducción de vértice, como se muestra en las siguientes figuras.

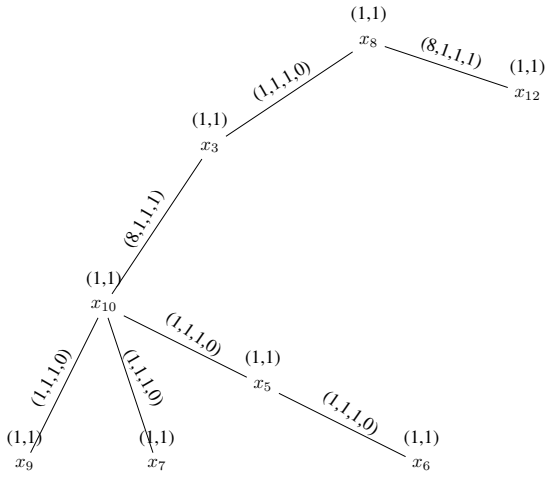


Figura 13: Árbol sobre el que se aplicará el procedimiento de conteo.

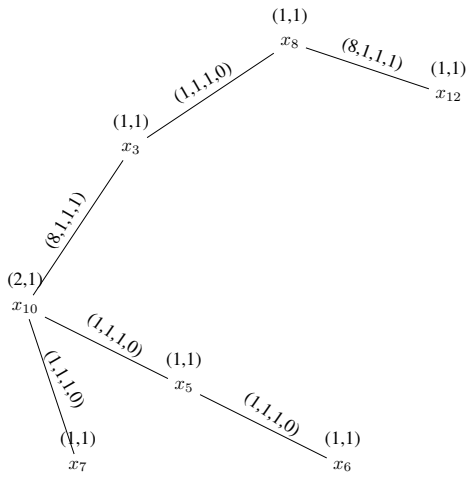


Figura 14: Resultado de la reducción del vértice x_9.

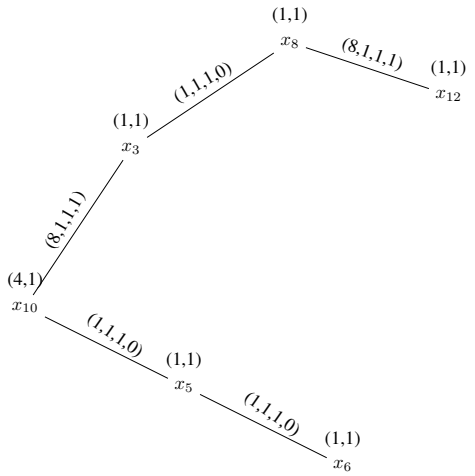


Figura 15: Resultado de la reducción del vértice x_7.

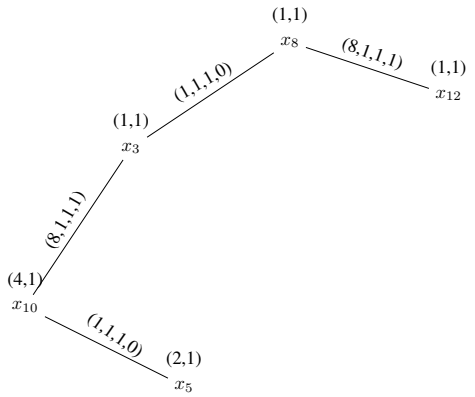


Figura 16: Resultado de la reducción del vértice x_6.

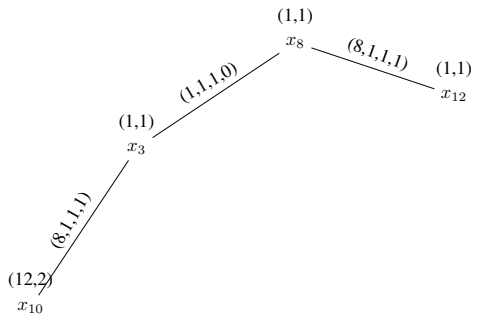


Figura 17: Resultado de la reducción del vértice x_5.

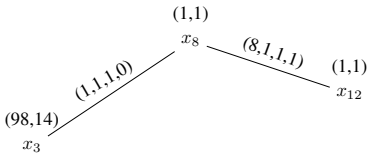


Figura 18: Resultado de la reducción del vértice x_{10}.

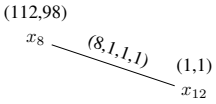


Figura 19: Resultado de la reducción del vértice x_3.

Figura 20: Resultado de la reducción del vértice x_{12}.

Como resultado del ejemplo se obtiene un total de 1204 modelos en la fórmula F.

Aunque ya extistían métodos para contar sobre árboles como los presentados en [8], este método permite conservar los modelos que representan subgráficas tipo serial paralelo, en una sola arista, con lo que se puede aplicar la operación de reducción por vértice y obtener el conteo de modelos correspondiente. Con esto se logra extender el tipo de fórmulas sobre las que se puede contar modelos en tiempo lineal.

4 Conclusiones

En este artículo se presento una nueva forma de contar modelos sobre gráficas tipo camino y árbol. El nuevo método permite contar sobre gráficas tipo árbol que contiene subgráfias serial-paralelo dentro de él. Adicionalmente, el procedimiento presentado se puede realizar en tiempo lineal, ya que como demostraron en [9] contar sobre gráficas serial paralelo se puede hacer en tiempo lineal debido a que el procedimiento utiliza el número de aristas y vértices en la gráfica serial paralelo ($O(n+m)$), y el conteo sobre gráficas tipo árbol también es lineal ya que se hace un recorrido en post orden sobre sus vértices ($O(n)$).

Referencias

[1] P. Winkler G. Brifhtwell. Counting linear extensions. *Order*, 8(e):225–242, 1991.

[2] M. A. López-Medina, J. R. Marcial-Romero, G. De Ita Luna, H. A. Montes-Venegas, and R. Alejo. A linear time algorithm for solving #2SAT on cactus formulas. *CoRR, ams/1702.08581*, 2017.

[3] M. A. López-Medina, J. R. Marcial-Romero, G. De Ita, and Y. Moyao. A Linear Time Algorithm for Computing #2SAT for Outerplanar 2-CNF Formulas. *Lecture Notes in Computer Science*, 10880:72–81, 2018.

[4] M. Wahlström. A tighter bound for counting max-weight solutions to 2sat instances. *Springer Berlin Heidelberg*, pages 202–213, 2008.

[5] B. Schoenmakers and L. A.M. A new algorithm for the recognition of series parallel graphs. *CWI (Centre for Mathematics and Computer Science)*, 1995.

[6] D. Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98:41 – 55, 1992.

[7] K. Takamizawa, T. Nishizeki, and N. Saito. Linear-time computability of combinatorial problems on series-parallel graphs. *Journal of the Association for Computing Machinery*, 29(3):623 – 641, 1982.

[8] J. R. Marcial-Romero, G. De Ita, J. A. Hernández, and R. M. Valdovinos. A parametric polynomial deterministic algorithm for #2sat. *Lecture Notes in Computer Science*, 9413:202–213, 2015.

[9] Marco A. López Medina, José Raymundo Marcial-Romero, Guillermo De Ita Luna, and José A. Hernández. A linear time algorithm for counting #2sat on series-parallel formulas. 12468:437–447, 2020.

2.5. Un algoritmo para el conteo de modelos en fórmulas booleanas cúbicas

Este artículo se envió y presentó en la Mexican Conference on Pattern Recognition(MCPR) 2023. Fue publicado como parte de Lecture Notes in Computer Science, vol. 13902 (2023). DOI: 10.1007/978-3-031-33783-3_7

Abstract We present an algorithm based on heuristic variable selection for computing the number of models on two conjunctive normal form Boolean formulas whose restricted graph is represented by a cubic graph.

For this class of formulas, we show that in most of the cases our proposal improves the time-complexity with respect of the current leader algorithm for counting models on two conjunctive form formulas of this kind.

Capítulo 3

Discusión y Trabajo futuro

En esta Tesis se presentan nuevos algoritmos para topologías de gráficas en las que la complejidad computacional en tiempo es de orden polinomial, con la finalidad de poder utilizarlos en un algoritmo general de conteo y de este modo reducir el tiempo necesario para llevar a cabo el conteo de modelos.

3.1. Discusión

Para validar la hipótesis planteada se realizaron una serie de propuestas que fueron publicadas en revistas arbitradas de alcance internaional. A continuación se brinda una discusión general de cada una de ellas.

El primer estudio que se realizó consideró las gráficas tipo serial-paralelo, que tienen una forma específica para llevar a cabo su construcción, además de tener una restricción dentro de sus gráficas menores a la gráfica k_4.

Para llevar a cabo el proceso de conteo en este tipo de gráficas y demostrar su correctitud, se crearon nuevas representaciones para el conteo de modelos, para poder comprobar equivalencia con las operaciones de lógica booleana utilizadas para la representación de una 2-FNC.

Las operaciones para construir una gráfica serial-paralelo son:

- Composición serial: consiste en tomar una arista en la gráfica que conecte dos vértices (u, v) e incluir un vértice adicional w entre ellos que sustituye a la arista que los une, para conectarlo posteriormente a los vértices de la arista eliminada, generando dos aristas (u, w) y (w, v).

- Composición paralela: consiste en duplicar una arista dentro de la gráfica, para poder realizar una composición serial en ellas.

Los métodos presentados por Marcial et.al. [16] utilizan un método de conteo actualizando valores sólo sobre los vértices en una construcción, de la gráfica que representa a una fórmula en 2-FNC, que se basa en un sentido de dirección hacia la raíz de la gráfica. En esta tesis se plantea la generación de un *elemento direccional* que permite recorrer las aristas de la gráfica en la dirección que se necesite considerando los siguientes criterios:

- Mantener el conjunto de operaciones, sobre los modelos, entre dos vértices dentro del elemento direccional.
- Un elemento direccional representa operaciones generadas por una reducción serial o una unión de aristas paralelas en la gráfica.
- Al aplicar los modelos calculados sobre un vértice de la gráfica en las operaciones de un elemento direccional genera como resultado el total de modelos representados por el elemento direccional.

  Éste procedimiento para calcular los nuevos valores sobre una cuádrupla y el conteo de modelos correspondiente fue validado y demostrado [26].

El procedimiento de conteo de modelos para las gráficas serial-paralelo se lleva a cabo realizando de manera iterativa ambas operaciones, contracción de aristas en serie y unión de aristas paralelas, el algoritmo termina cuando se tiene una sóla arista dentro de la gráfica que contendrá todas las operaciones necesarias en sus elementos direccionales y se obtendrán los modelos al aplicar los valores de uno de los vértices a las operaciones incluidas en el elemento.

El segundo artículo 2.2 presenta el uso de las operaciones para construir una gráfica serial paralelo con el objetivo de calcular el *clique-width*, demostrando que el *clique-width* para este tipo de gráficas es menor o igual a 5, mejorando la relación definida usando el valor de *tree-width* de estas gráficas que es 2, $cwd(G) \leq 3 \cdot 2^{twd(G)-1}$ que da un valor de 6.

En este artículo el proceso para realizar el cálculo del $cwd(G)$ se realiza bajo la misma relación del conteo de modelos, primero se resuelven de manera individual todos los caminos, demostrado en el primer lema, y mediante los siguientes lemas se demuestra el cálculo del *cwd* dobre las distintas relaciones posibles en la composición paralela en este tipo de gráficas.

Demostrando que *cwd* en gráficas serial paralelo es menor o igual a 5, con un algoritmo cuya complejidad es polinomial $O(n^2)$.

En el tercer artículo 2.3 se presenta un algoritmo basado en operaciones combinatorias para realizar el conteo de modelos en gráficas tipo malla. Este tipo de gráficas han sido estudiadas y se han propuesto algoritmos basados en la matriz de transferencia [27] [18] [28].

En este artículo se propone una representación de los modelos de cada columna de la gráfica mediante una lista, cada elemento representa un modelo, y utilizando las aristas entre columnas como una serie de restricciones adicionales se plantea realizar la aplicación de las restricciones de manera secuencial entre columnas.

El método trabaja de manera sencilla, primero se calculan todas las listas de modelos para cada columna, posteriormente desde la columna 1 hasta la columna m se realiza la validación de cada una de las n restricciones entre cada par de columnas. Generando una complejidad para el conteo de modelos en este tipo de gráficas de $O(t \cdot 1.618^{2\sqrt{t}})$.

El cuarto artículo 2.4, es la presentación de un algoritmo para mejorar el método de conteo sobre gráficas tipo serial-paralelo, en este caso son fórmulas tipo árbol que incluyen subgráficas tipo serial-paralelo y que no son posibles resolver sólo con las dos operaciones mostradas en el segundo artículo 2.2. En este artículo, siguiendo la representación y uso de los elementos en gráficas serial-paralelo, se presenta una operación adicional para reducir vértices hoja dentro del árbol.

La operación adicional presentada realiza la reducción de un vértice de grado uno y la arista que lo conecta con el resto de la gráfica, es decir, que hace uso de las operaciones almacenadas en el elemento direccional de la arista y los valores que fueron calculados para el vértice a eliminar, la operación presentada y el conteo de modelos que lleva acabo se encuentran demostradas y ejemplificadas en el artículo. Mediante esta operación se tienen dos resultados. El primero es el conteo de modelos sobre cualquier fórmula que se representa mediante una gráfica acíclica, que es $O(n)$, ya que no es necesario realizar construcción de la gráfica que representa a la fórmula y realizar su conteo. El segundo es el método de conteo para las fórmulas con una representación de árbol con subgráficas serial-paralelo, que es $O(n+m)$, obteniendo un algoritmo lineal para el conteo de modelos en éste tipo de fórmulas.

El quinto artículo 2.5 presenta un algoritmo lineal para la selección de variables de *descomposición* para realizar el conteo de modelos en fórmulas que se representan por gráficas cúbicas. El algoritmo para la selección de variables se basa en encontrar a los vértices con un mayor nivel de intersección en una construcción de la gráfica, es decir que al construir un árbol de expansión de la gráfica y generar las aristas de retroceso, el nivel de intersección de una variable es el número de veces que aparece en el camino formado por una arista de retroceso en el árbol, por ejemplo si existen dos aristas de retroceso que contengan a la raíz, su nivel de intersección es dos.

El funcionamiento del algoritmo para realizar la construción del árbol de expansión realiza previamente una selección de las aristas que van a conformar el árbol utilizando un algoritmo para transformar gráficas outerplanar a su forma de ciclos embebidos [29], de esta forma se plantea asegurar que los vértices con mayor intersección *separen* la gráfica más fácilmente.

Debido a que existen distintas representaciones de una gráfica con ciclos embebidos, se plantea necesario el uso de elegir de manera aleatoria sobre los vértices de mayor intersección al vértice que se elegirá para la descomposición de la gráfica, agregándolo a una lista L'_T. Realizando este procedimiento a partir de la gráfica original un número de veces determinado por un parámetro *limite*, de tal forma que todas las listas generadas en cada aplicación del procedimiento se comparan con la lista con menos elementos generada hasta el momento L_T, lo que significa que si $|L'_T| < |L_T|$ entonces se sustituye la lista L_T actual por L'_T. Al final se mantiene la lista con un menor número de elementos.

Para mantener la complejidad de los algoritmos más eficientes presentados para este tipo de gráficas se respetan las consideraciones para las variables que son generadas durante la descomposición, se simplifican las variables de grado uno y dos 2.4, por lo que nuestro algoritmo para gráficas cúbicas toma una lista de variables L_T, con $|L_T| \leq \frac{n}{4}$ lo que da una complejidad de $O(2^{|L_T|})$. Si $O(2^{\frac{n}{4}}) \approx O(1.1892^n)$, then $O(2^{|L_T|}) \leq O(1.1892^n)$.

En resumen se plantearon nuevos métodos de conteo de modelos para gráficas para las que no se tenía un procedimiento como lo son las gráficas serial-paralelo, se mejoró la complejidad que se tenía para algoritmos que realizan conteo de modelos en gráficas tipo malla. Se realizó la demostración para acotar el *clique-width* en las gráficas serial paralelo a menor o igual a 5, mejorando la relación dada por $cdw(G) \leq 3 \cdot 2^{twd(G)-1}$ [30], que en gráficas serial paralelo es 6. Se presenta una primer mejora en el algoritmo para conteo en gráficas tipo árbol que contienen subgráficas serial paralelo, y finalmente un algoritmo para realizar una mejor selección de variables de *descomposición* para una gráfica cúbica.

3.2. Trabajo futuro

Las propuestas para extender los resultados obtenidos en la investigación son listadas a continuación.

- Existe un gran volumen de literatura orientado al conteo de estructuras dentro de las gráficas tipo malla, por ejemplo, árboles de expansión, ciclos Hamiltonianos, conjuntos independientes, k-coloreo, etc. [27, 31, 18, 17]. El conteo de modelos sobre este tipo de gráficas se ha presentado mediante una modificación al método de matriz de transferencia [17], o el uso de listas combinatorias para reducir su complejidad [32]. En todos los casos se han utilizado estructuras regulares, celdas de cuatro o séis vértices, sin embargo dadas las características conocidas de este tipo de gráficas que pueden contener diagonales o una malla conformada por polígonos irregulares, se plantea el determinar si es posible agrupar en una categoría a gráficas sobre las que se pueda aplicar el método de conteo sobre este tipo de mallas.
- Existen varios artículos dirigidos a resolver instancias del problema de conteo #2SAT, sobre las que se ha demostrado una complejidad polinomial, como lo son las topo-

logías de árbol [16], cactus [6], outerplanar [8, 29] y serial-paralelo [26]. Por lo que al tratarse de gráficas con características definidas, se plantea el desarrollo de un método de conteo aplicable a todas estas instancias y que permita ser extendido para nuevas topologías, demostrando la aplicabilidad del algoritmo independientemente de la topología con la que trabaje o la forma de la gráfica que se construya a partir de una fórmula. Para llevar a cabo el desarrollo de este algoritmo se necesitará realizar la demostración de él para cada topología presentada.

Bibliografía

[1] Darwiche A. On the tractability of counting theory models and its application to belief revision and truth maintenance. *Journal of Applied Non-classical Logics*, 11(1-2):11–34, 2001.

[2] Garey M. and Johnson D. *Computers and Intractability a Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., 1979.

[3] Winkler Peter Brifhtwell G. Counting linear extensions. *Order*, 8(e):225–242, 1991.

[4] Szeider S. Paulusma D., Slivovsky F. Model counting for cnf formulas of bounded modular treewidth. *Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik*, 20:55–66, 2013.

[5] Magnus Wahlström. A tighter bound for counting max-weight solutions to 2sat instances. *Springer Berlin Heidelberg*, pages 202–213, 2008.

[6] Marco A. López, José Raymundo Marcial-Romero, Guillermo De Ita Luna, Héctor A. Montes Venegas, and Roberto Alejo. A linear time algorithm for solving #2SAT on cactus formulas. *CoRR, ams/1702.08581*, 2017.

[7] Marco A. López, J. Raymundo Marcial-Romero, Guillermo De Ita, and Rosa M. Valdovinos. A fast and efficient method for #2sat via graph transformations. *Advances in Soft Computing*, pages 95 – 106, 2017.

[8] Marco A. López, J. Raymundo Marcial-Romero, Guillermo De Ita, and Yolanda Moyao. A Linear Time Algorithm for Computing #2SAT for Outerplanar 2-CNF Formulas. *Lecture Notes in Computer Science*, 10880:72–81, 2018.

[9] Murty U. S. R. Bondy J. A. *Graph Theory*. Springer Verlag, Graduate Texts in Mathematics, 2010.

[10] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of ACM*, 52:337 – 364, 2005.

[11] Timpon Hertli. 3-sat faster and simpler - unique-sat bounds for ppsz hold in general. *52nd IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 277 – 284, 2011.

[12] Thurley M. sharpsat - counting models with advanced component caching and implicit bcp. *The 9th International Conference on Theory and Applications of Satisfiability Testing (SAT 2006)*, pages 424–429, 2006.

[13] Schmitt M. and Wanka R. Exploiting independent subformulas: A faster approximation scheme for #k-sat. *Information Prcocessing Letters*, 113(9):337 – 344, 2013.

[14] Bernd Becker Jan Burchard, Tobias Schubert. Laissez-faire caching for parallel #sat solving. *SAT 2015*, pages 46–61, 2015.

[15] Sæther Sigve Hortemo, Telle Jane Arne, and Vatshelle Martin. Solving maxsat and #sat on structured cnf formulas. *Theory and Applications of Satisfiability Testing SAT 2014*, 8561:16 – 31, 2014.

[16] J. Raymundo Marcial-Romero, G. De Ita, J. Antonio Hernández, and R. M. Valdovinos. A parametric polynomial deterministic algorithm for #2sat. *Lecture Notes in Computer Science*, 9413:202–213, 2015.

[17] C. Guillen, A. Lopez Lopez, and G. De Ita. Computing #2-sat of grids, grid-cylinders and grid-tori boolean formulas. In *Proceedings of the 15th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion*, volume 451. CEUR-WS.org, 2008.

[18] Mordecai Golin, Yiu Cho Leung, Yajun Wang, and Xuerong Yong. Counting structures in grid graphs, cylinders and tori using transfer matrices: Survey and new results. In *ALENEX/ANALCO*, pages 250–258, 01 2005.

[19] Martin Fürer and Shiva Prasad Kasiviswanathan. Algorithms for counting 2-sat solutions and colorings with applications. In Ming-Yang Kao and Xiang-Yang Li, editors, *Algorithmic Aspects in Information and Management*, pages 47–57, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[20] R. Schrag Bayardo R. Using csp look-back techniques to solve real-world sat instances. *Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 203–208, 1997.

[21] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, April 1996.

[22] Dahllöf V., Jonsson P., and Wahlström M. Counting models for 2sat and 3sat formulae. *Theoretical Computer Sciences*, 332(1-3):265 – 291, 2005.

[23] Khardon R. and Roth D. Reasoning with models. *Artificial Intelligence*, 87(1):187 – 213, 1996.

[24] Russ B. Randomized algorithms: Approximation, generation, and counting. *Distinguished dissertations Springer*, 2001.

[25] Vadhan Salil P. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal of Computing*, 31(2):398 – 427, 2001.

[26] Marco A. López-Medina, J. Raymundo Marcial-Romero, Guillermo De Ita-Luna, and José A. Hernández. A linear time algorithm for counting #2sat on series-parallel formulas. In *Advances in Soft Computing*, pages 437–447. Springer International Publishing, 2020.

[27] Neil Calkin and Herbert Wilf. The number of independent sets in a grid graph. *SIAM Journal on Discrete Mathematics*, 11, 02 1998.

[28] Carlos Guillén, Guillermo De Ita, and Aurelio López-López. A novel method for counting models on grid boolean formulas. In José Francisco Martínez-Trinidad, Jesús Ariel Carrasco-Ochoa, and Josef Kittler, editors, *Advances in Pattern Recognition*, pages 322–331, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[29] Marco A. López, J. Raymundo Marcial-Romero, José A. Hernández, and Guillermo De Ita. Model counting for #2sat problem in outerplanar graphs. In *Proceedings of the Eleventh Latin American Workshop on Logic/Languages, Algorithms and New Methods of Reasoning*, volume 2264, pages 76–87. CEUR Workshop Proceedings, 2018.

[30] Bruno Courcelle and Stephan Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101:77 – 114, 2000.

[31] Reinhardt Euler. The fibonacci number of a grid graph and a new class of integer sequences. *Journal of Integer Sequences*, 8, 05 2005.

[32] Marco A. López-Medina, J. Raymundo Marcial-Romero, Guillermo De Ita Luna, and José A. Hernández. A method for counting models on grid boolean formulas1. *Journal of Intelligent & Fuzzy Systems*, 42(5):4719–4726, mar 2022.