Informaticae Abstracta

Año 2023, Vol. 1, No. 1 JUL | DIC 2023

Universidad Autónoma del Estado de México

Informaticae Abstracta
Año 2023, Vol. 1, No. 1 JUL | DIC 2023

Informaticae Abstracta, número 1, año 2023, es una revista de publicación continua editada por la Universidad Autónoma del Estado de México, con sede en el Instituto Literario número 100 oriente, Colonia Centro, Toluca, Estado de México, C.P. 50000. Se puede contactar a través del teléfono +(52 722) 2140855 y 21140795 ext. 10 1217, así como en el sitio web http://informaticae.uaemex.mx o mediante el correo electrónico: informaticae@uaemex.mx. La editora responsable es Ruth Hernández Pérez, perteneciente a la Facultad de Ingeniería. La revista cuenta con los derechos de uso exclusivo número 04-2022-111113523100-102 otorgados por el Instituto Nacional del Derecho de Autor. La última edición fue supervisada por Ruth Hernández Pérez, Javier Salas García y Mercedes Lucero Chávez, de la Facultad de Ingeniería, ubicada en Cerro de Coatepec s/n, Ciudad Universitaria, C.P. 50100, Toluca, Estado de México. La fecha de última modificación fue el 29/06/2023. Diseño de portada y diseño editorial: Ruth Hernández Pérez y Javier Salas García.

Informaticae Abstracta
Año 2023, Vol. 1, No. 1 JUL | DIC 2023

Índice

Informaticae Abstracta
Año 2023, Vol. 1, No. 1, pp. 4-20 JUL | DIC 2023

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

López Medina, M. A.; Marcial Romero, J. R.; González Ruiz, J. L.; Hernández Servín, J. A.

Información del Artículo

Recibido: 11-mar-2023
Aceptado: 27-jun-2023

Palabras clave: 2SAT, teoría de la complejidad, teoría de grafos, algoritmos lineales, fórmulas tipo árbol,

Resumen

Este artículo presenta un algoritmo de tiempo lineal para resolver el problema de conteo #2SAT en fórmulas booleanas de dos formas normales conjuntivas (2-CNF) con una estructura específica. El enfoque se centra en fórmulas cuya representación gráfica forma árboles con subgrafos serie-paralelo en sus nodos interiores u hojas.

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

Preliminares

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

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 $s : \nu(F) \to \{0, 1\}$ 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, $s(c) = 1$.

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.

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:

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

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:

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

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:

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

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 $v_i : 1 \leqslant i \leqslant k$, 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.

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

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

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.

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.

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 $\{x_1, x_n\}$.

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 $e_l = \{x_{i_l}, x_{j_l}\}$, $1 \leqslant l \leqslant m$, 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.

Estructuras específicas

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

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:

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

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.

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.

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.

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.

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.

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:

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

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

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

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

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.

Lemma 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* $x_2, \ldots x_{n-1}$*, 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.

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

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

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

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 $(\gamma_i \neq \epsilon_j)$.

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

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:

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

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.

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

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

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

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

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

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.

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, $e'_l = \{x_i, x_j, C_{x_i x_j}\} : 1 \leqslant l \leqslant h$. La unión puede llevarse a cabo con la ecuación:

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

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

Lemma 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* $(x_1, x_2, C_{x_1 x_2})$ *entonces:*

$$\#2SAT(F) = \pi_1(C_{x_1 x_2})\alpha_2 + \pi_2(C_{x_1 x_2})\beta_2 + \pi_3(C_{x_1 x_2})\alpha_2 + \pi_4(C_{x_1 x_2})\beta_2 \tag{12}$$

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 $C_{x_i x_j}$) y su unión en la figura 8.

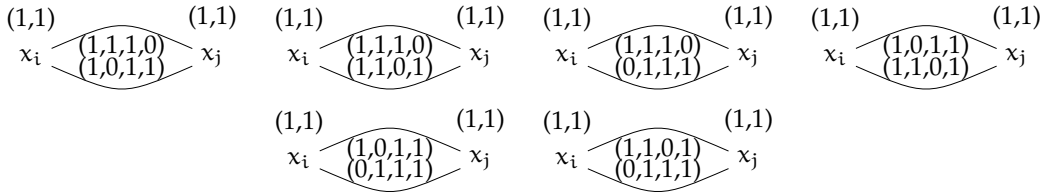

Figura 7: Casos paralelos sobre dos cláusulas.

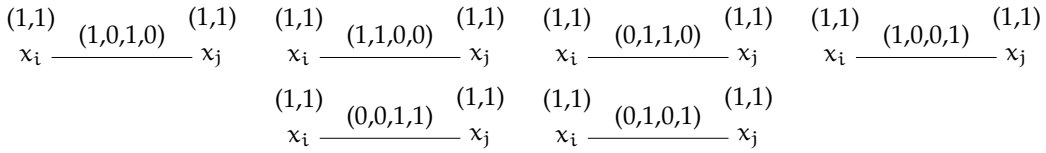

Figura 8: Aplicando conteo sobre unión de aristas en los casos de dos cláusulas.

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.

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 $x_j : 1 \leqslant j \leqslant k$, y el conjunto de aristas con elemento direccional $C_{x_j x_i}$, los valores (α_i, β_i) se calculan usando las ecuaciones 13 y 14;

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

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

Lemma 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.

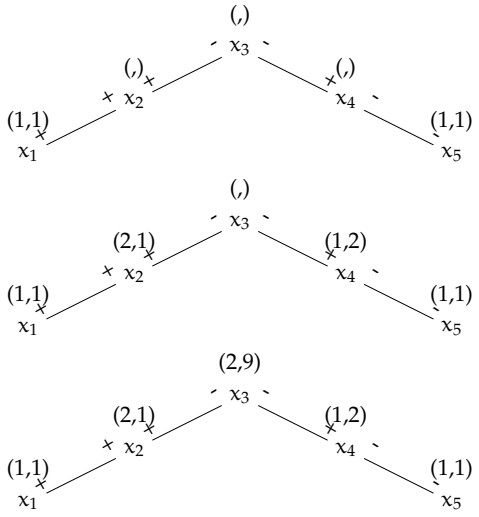

Figura 9: Representación y conteo sobre una gráfica tipo árbol con cinco vértices, tomando como raíz x_3, usando la ecuación 4.

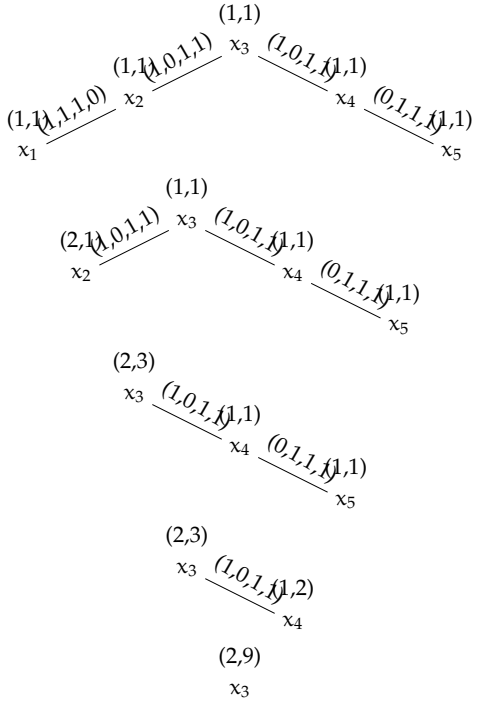

Figura 10: Representación y conteo sobre una gráfica tipo árbol con cinco vértices, tomando como raíz x_3, usando las ecuaciones 13 y 14.

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 y 3.
- Recorrer el árbol y aplicar el algoritmo 2.

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

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 $E' = \{\{x_1, x_3, (1,1,1,0)\}, \{x_{10}, x_1, (1,1,1,0)\},$
$\{x_2, x_3, (1,1,1,0)\}, \{x_{10}, x_2, (1,1,1,0)\},$
$\{x_4, x_3, (1,1,1,0)\}, \{x_3, x_8, (1,1,1,0)\},$
$\{x_{10}, x_4, (1,1,1,0)\}, \{x_9, x_{10}, (1,1,1,0)\},$
$\{x_7, x_{10}, (1,1,1,0)\}, \{x_6, x_5, (1,1,1,0)\},$
$\{x_5, x_{10}, (1,1,1,0)\}, \{x_{12}, x_{13}, (1,1,1,0)\},$
$\{x_{12}, x_{14}, (1,1,1,0)\}, \{x_{12}, x_{11}, (1,1,1,0)\},$
$\{x_{11}, x_8, (1,1,1,0)\}, \{x_{14}, x_8, (1,1,1,0)\},$
$\{x_{13}, x_8, (1,1,1,0)\}\}$:

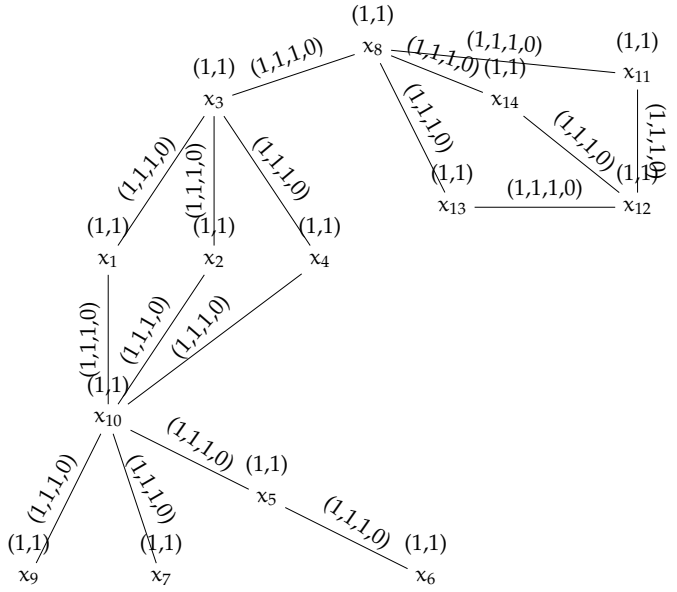

Figura 11: Representación gráfica de F.

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.

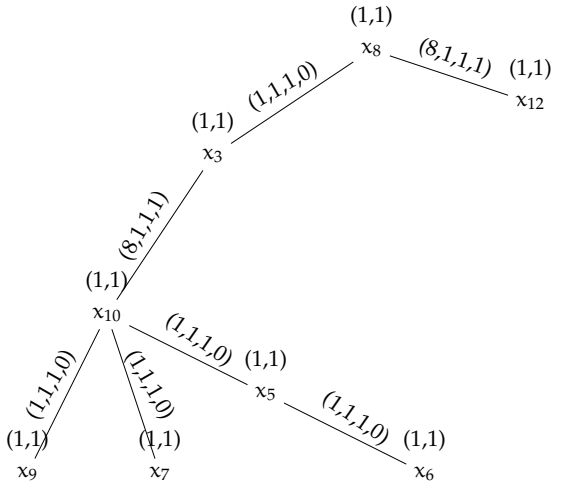

Figura 12: Representación gráfica de F al aplicar la reducción de las subgráficas serial paralelo.

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

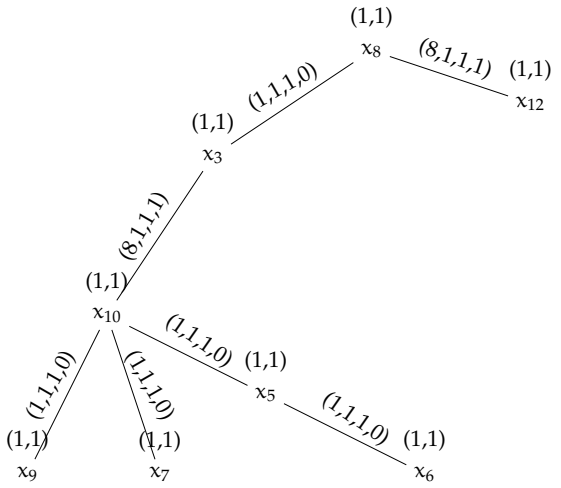

Figura 13: Árbol sobre el que se aplicará el procedimiento de conteo.

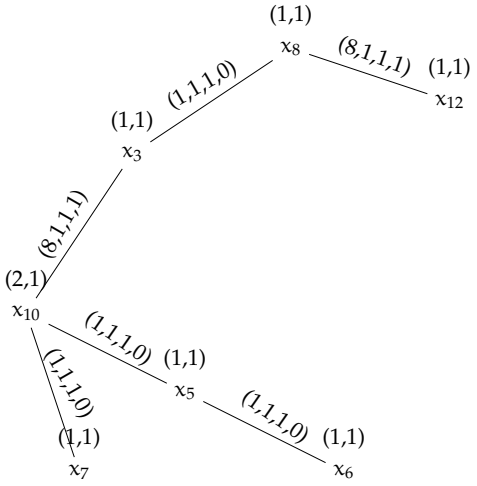

Figura 14: Resultado de la reducción del vértice x_9.

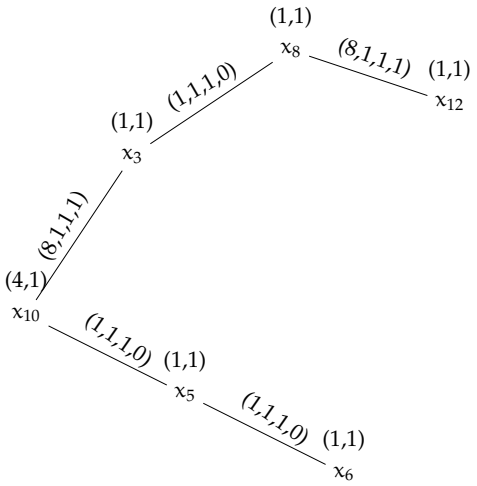

Figura 15: Resultado de la reducción del vértice x_7.

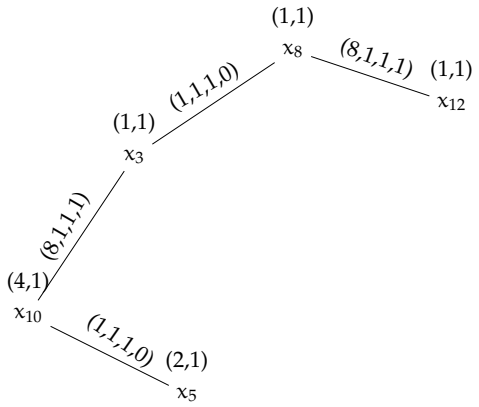

Figura 16: Resultado de la reducción del vértice x_6.

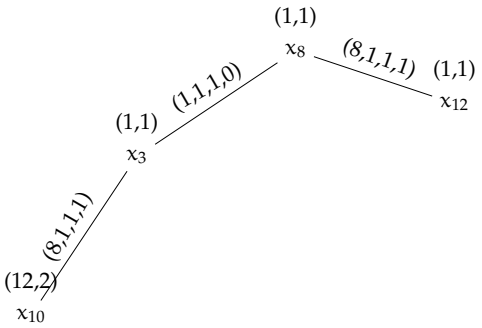

Figura 17: Resultado de la reducción del vértice x_5.

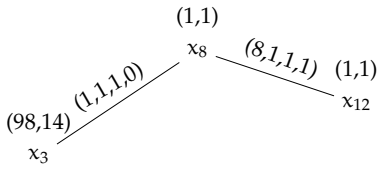

Figura 18: Resultado de la reducción del vértice x_{10}.

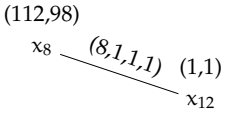

Figura 19: Resultado de la reducción del vértice x_3.

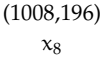

Figura 20: Resultado de la reducción del vértice x_{12}.

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 ($O(n)$).

Bibliografía

[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.

Direcciones de correo de los autores

López Medina, M. A.: malopezme@uaemex.mx

Marcial Romero, J. R.: jrmarcialr@uaemex.mx

González Ruiz, J. L.: jlgonzalezru@uaemex.mx

Hernández Servín, J. A.: xoseahernandez@uaemex.mx

Informaticae Abstracta
Año 2023, Vol. 1, No. 1, pp. 20-29 JUL | DIC 2023

Independent sets on n-polygonal regular arrays

Hernández-Servín, J.A.; Marcial-Romero, J.R.; De Ita Luna, G.

Article info

Recibido: 10-feb-2023
Aceptado: 15-jun-2023

Keywords: Merrifield-Simmons index, Polygonal arrays, Independent sets, Graph theory

Abstract

On this paper, we present a closed formula to compute the Merrifield-Simmons index on polygonal arrays at distance 1. The proof of our main theorem establishes a method to solve the recurrences derived from a decomposition of the original polygonal array at any distance. Hence, closed formulas for any polygonal array can be obtained. Closed formulas had previously been exhibited when polygons were vertex's joined however for edge's joined it has not been previously presented.

1 Introduction

Hexagonal chain graphs have been widely investigated. They represent a relevant area of interest in mathematical chemistry, since they are molecular graphs used to represent the structural formula of chemical benzenoid compounds. Thus, the unbranched catacondensed benzenoid molecules play an important role for the theoretical chemistry of benzenoid hydrocarbons [1].

The Merrifield-Simmons index of a molecular graph G, that is the number of independent sets of G, denoted as $\sigma(G)$, is a typical example of an invariant used in mathematical chemistry for quantifying relevant details of molecular structures. Merrifield and Simmons showed the correlation between $\sigma(G)$ and boiling points for polygonal chain graphs representing chemical molecules.

Hexagonal chains have been widely investigated and represent a relevant area of interest in mathematical chemistry, since they are used for quantifying relevant details of the molecular structure of the benzenoid hydrocarbons [2, 1, 3].

In recent years, a lot of work has been done on the extremal problem for the Merrifield-Simmons index, i. e., on determining the graphs within a prescribed class that minimize or maximize the value of the index. Gutman [4] discussed the extremal hexagonal chains according to three topological invariants: Hosoya index, largest eigenvalue, and Merrified-Simmons index. His work greatly motivated the study of extremal polygonal chains.

On his seminal paper, Gutman showed that the linear chain has the maximum value for the Merrifield-Simmons index for the particular case of hexagonal chains. Afterward, L.Zhang [5] showed that the minimum value for the Merrifield-Simmons index is achieved by the zig-zag polyphenegraph, which we denote as H^1.

These results were further generalized in various directions: the second-smallest (-largest) values were determined [6] as well as the corresponding extremal chains, and it was shown that the linear chain is still optimal if catacondensed benzenoids are considered (which allow branching, as opposed to chains). A very surprising result due to Zhang and Zhang [7] states that the linear chain and the zigzag chain

are even extremal, when the number of independent sets or edges of any fixed size k (and not just the sum over all k) are considered.

Several works have been developed for searching closed formulas to compute the Merrifield-Simmon index on different class of graphs. For example, Döslić et.al [9] showed closed formulas based on the independence polynomials of polyphenylene uniform chains. De Ita et.al [8] presented some recurrence equations to compute the Merrifield-Simmons index on polygonal uniform chains.

In this paper, we develop initial recurrence equations expressing the Merrifield-Simmons index on a polygonal uniform chain at distance 1. Our objective is to find closed formulas for computing $\sigma(H^1)$, we also show that the computation of $\sigma(H^1)$ can be done in poly-logarithmic time, and in fact, for any polygonal uniform chain at distance 1.

2 Preliminaries

Let $G = (V, E)$ be an undirected simple graph with set of vertices V and set of edges E. The *neighborhood* of a vertex $v \in V$ is defined as the set of vertices adjacent to v, that is $N(v) = \{y \in V \mid vy \in E\}$. The *closed neighborhood* of V is defined as $N[v] = N(v) \cup \{v\}$.

A simple path of length n between two vertices v and y , denoted as P_n, is a sequence of distinct edges in E of the form $v_0v_1, v_1v_2, \ldots, v_{n-1}v_n$, such that $v_0 = v$ and $v_n = y$. A simple cycle C_n is a non-empy simple path P_n, where the first and last vertices are identical. The distance $d_G(v; y)$ from a vertex v to another vertex y is the minimum number of edges in an $v - y$ path of G.

An *independent set* of V is a subset $S \subseteq V$ such that when $v, y \in S$, then $vy \notin E$. The number of independent sets of a graph G is denoted as $\sigma(G)$. We restrict our attention to particular cases of molecular graphs. An m-gon is a simple cycle C_m. Meanwhile, an n, m-gon chain graph consists of a set of n, m-gons $C^1, C^2, \ldots C^n$ such that $E(C^i) \cap E(C^{i+1}) = \{e_i\}$, e.g. the intersection of two consecutive m-gons is a single edge. Figure 1 shows a 5, 6-gon chain graph, which is also called an hexagonal chain graph. From now on, we will call an nm-gon chain graph as an array of n, m-gons.

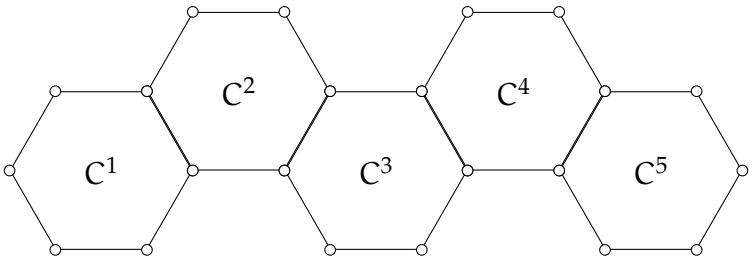

Figure 1: Hexagonal chain graph

Let $\mathcal{H}_n^m$ be an array of n, m-gons. By definition, each par of adjacent m-gons C^i and C^{i+1} are joined by a common edge, lets say e_i. If each pair of consecutive sharing edges e_i and e_{i+1} of the array holds that $d_{H_n}(e_i, e_{i+1}) = r$, then we say that $\mathcal{H}_n^m$ is an array at distance r. For example, for hexagonal chain graphs with $r = 1$, then $\mathcal{H}_n^6$ is known as the zig-zag hexagonal chain (Z_n), Figure 1.

The following rules have been useful to count independent sets on a graph G:

1. Vertex reduction rule: Let $v \in V(G)$,

$$\sigma(G) = \sigma(G - v) + \sigma(G - N[v]) \tag{1}$$

2. Edge division rule : let $e = \{x, y\} \in E(G)$,

$$\sigma(G) = \sigma(G - e) - \sigma(G - N[x] \cup N[y]) \tag{2}$$

Therefore, in order to count the number of independent sets on an $\mathcal{H}_n^m$ array at distance 1, we find a recurrence which represents that number. Afterward, we calculate a closed formula using a matrix method.

3 Posing the problem

In this section, we formally establish the problem we want to solve. The question we want to answer in this paper is how to calculate $\sigma(\mathcal{H}_n^m)$, that is

Question 1. *What's* $\sigma(\mathcal{H}_n^m)$*, where* $\mathcal{H}_n^m$ *is an* n, m*-gon?*

For simplicity, we drop the index m as the proof the main theorem and set of recurrences is independent of m. Part of the answer is to calculate the set of recurrences that are inferred using vertex and edge division rules. For that, we are using the decomposition of a $6,4-$gon, which is shown in Figure 2.

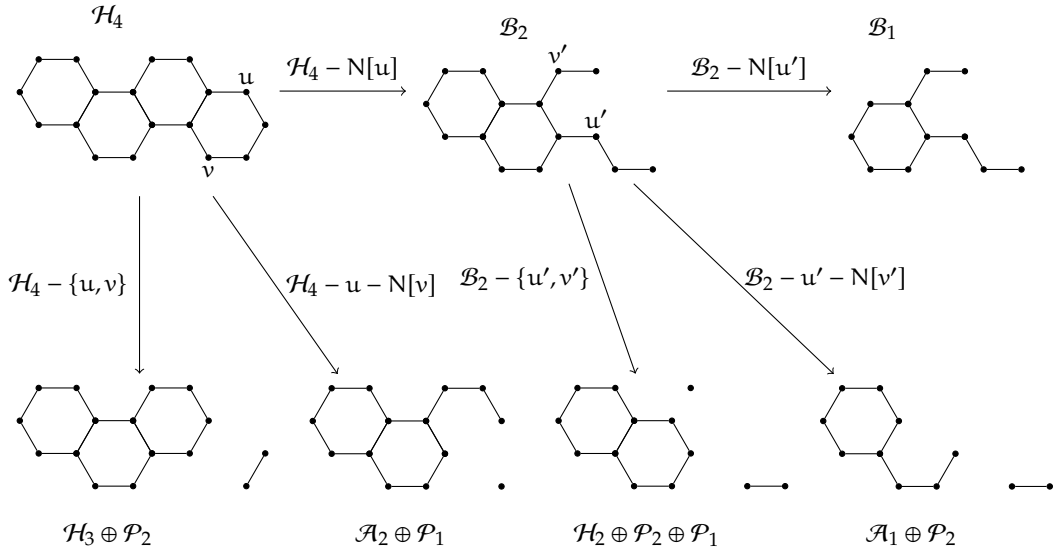

Figure 2: Decomposition of $\mathcal{H}_4$ in order to get the graphs $\mathcal{H}_3$, $\mathcal{A}_2$, $\mathcal{B}_2$, etc., which lead to the set of recurrences.

Let $\mathcal{H}_n$ be and array of n, m-gons, we ommit the superscript in $\mathcal{H}_n$ since it is composed uniquely of m-gons. Let $C^1, C^2, \ldots, C^n$ be the m-gons of $\mathcal{H}_n$. Let e_{n-1} be the edge that joins the m-gons C^{n-1} and C^n. Let v and y be the neighbours of the vertices e_{n-1} in C^n. We use Equation 1 to decompose $\mathcal{H}_n$ in order to calculate $\sigma(\mathcal{H}_n)$.

$$\begin{aligned}\sigma(\mathcal{H}_n) &= \sigma(\mathcal{H}_n - v) + \sigma(\mathcal{H}_n - N[v]) \\ &= \sigma(\mathcal{H}_n - v - y) + \sigma(\mathcal{H}_n - v - N[y]) + \sigma(\mathcal{H}_n - N[v])\end{aligned}$$

It is easy to see that $\sigma(\mathcal{H}_n - v - y) = \sigma(\mathcal{H}_{n-1}) \cdot \sigma(\mathcal{P}_{m-4})$. On the other hand, $\sigma(\mathcal{H}_n - v - N[y])$ can be computed as $\sigma(\mathcal{A}_{n-2}) \cdot \sigma(\mathcal{P}_{m-5})$, where $\mathcal{A}_{n-2}$ is the graph formed by the union of $\mathcal{H}_{n-2}$ and a path of length $m-3$ joined to a vertex of C^{n-2}. Figure 2 shows an example of $\mathcal{A}_{n-2}$ for hexagonal arrays. We call the graph $\mathcal{H}_n - (N[v])$ as $\mathcal{B}_{n-2}$, since it is the union of $\mathcal{H}_{n-2}$ and two paths of length $m-3$ and $m-4$ to the vertices of an edge e of $\mathcal{H}_2$. Figure 2 shows and example of $\mathcal{B}_{n-2}$.

The computation of $\sigma(\mathcal{A}_{n-2})$ is achieved by applying Equation 1 to the first vertex of the path, lets say v, which joins it with $\mathcal{H}_{n-2}$. The result is:

$$\begin{aligned}\sigma(\mathcal{A}_{n-2}) &= \sigma(\mathcal{A}_{n-2} - v) + \sigma(\mathcal{A}_{n-2-}(N[v])) \\ &= \sigma(\mathcal{P}_{m-4}) \cdot \sigma(\mathcal{H}_{n-2}) + \sigma(\mathcal{P}_{m-5}) \cdot \sigma(\mathcal{A}_{n-3})\end{aligned}$$

The calculation of $\sigma(B_{n-2})$ is done by applying Equation 1 to the first vertices of the paths, lets say v and y, which join them with H_{n-2}. The result is:

$$\begin{aligned}\sigma(\mathcal{B}_{n-2}) &= \sigma(\mathcal{B}_{n-2} - v) + \sigma(\mathcal{B}_{n-2} - N[v]) \\ &= \sigma(\mathcal{B}_{n-2} - v - y) + \sigma(\mathcal{B}_{n-2} - v - N[y]) + \sigma(\mathcal{B}_{n-2} - N[v]) \\ &= \sigma(\mathcal{P}_{m-5}) \cdot \sigma(\mathcal{P}_{m-4}) \cdot \sigma(\mathcal{H}_{n-2}) + \sigma(\mathcal{P}_{m-4}) \cdot \sigma(\mathcal{P}_{m-6}) \cdot \sigma(\mathcal{A}_{n-2}) + \\ &\quad \sigma(\mathcal{P}_{m-5}) \cdot \sigma(\mathcal{B}_{n-2})\end{aligned}$$

It is well known that $\sigma(\mathcal{P}_n) = F_{n+2}$, where F_n is the $n-th$ Fibonacci number. Hence, the final recurrences are as follows:

$$\begin{aligned}H_{n+1} &= F_{m-2} \cdot H_n + F_{m-3} \cdot A_{n-1} + B_{n-1} \\ A_{n+1} &= F_{m-2} \cdot H_{n+1} + F_{m-3} \cdot A_n \\ B_{n+1} &= F_{m-2} \cdot F_{m-3} \cdot H_{n+1} + F_{m-2} \cdot F_{m-4} \cdot A_n + F_{m-3} \cdot B_n\end{aligned}$$

where we have simplified the notation $\sigma(\mathcal{H}_n) = H_n$, $\sigma(\mathcal{A}_n) = A_n$ and $\sigma(\mathcal{B}_n) = B_n$. Now, if we make $p = F_{m-2}$, $q = F_{m-3}$ and $r = F_{m-4}$, then the recurrences for an array of n, m-gons at distance one become a reduced sets of recurrences:

$$\begin{aligned}H_{n+1} &= pH_n + qA_{n-1} + B_{n-1} \\ A_{n+1} &= pH_{n+1} + qA_n \\ B_{n+1} &= pqH_{n+1} + prA_n + qB_n\end{aligned} \tag{3}$$

Thus, Question (1) has been reduced to solve the above set of recurrences, which we are going established as the main theorem of this paper.

Theorem 1. *Let us define* a_n, c_n *two sequences as* $a_{-1} = 0$, $a_0 = 0$, *and* $a_n = n$ *for all* n, *and* $c_0 = 1$ *and* $c_n = -(a_{n-2}P + a_n Q)$, *where* $P = p(pr - q^2)/q^2$ *and* $Q = p/q$. *Let* T *be the upper triangular matrix*

$$\begin{pmatrix} 1 & c_1 & \cdots & c_n \\ 0 & 1 & \cdots & c_{n-1} \\ \vdots & & \cdots & \vdots \\ 0 & 0 & \cdots & c_1 \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

and $\Lambda = (\delta_{ij} t_{ij})$. *If* $y_n = H_n/q^n$, *then we have that*

$$\begin{pmatrix} y_n \\ \vdots \\ y_5 \end{pmatrix} = \left(\sum_{j=0}^{n} \left(-\Lambda^{-1} T_u \right)^j \right) \begin{pmatrix} z_n \\ \vdots \\ z_4 \end{pmatrix} \tag{4}$$

gives a solution to the system of recurrences (3) for H_n, *where* $z_n = x_n - \sum_{s=2}^{4} c_{4-s+1} y_s$ *and the sequence* x_n *is given by*

$$x_n = \frac{A_1}{q^2} + \frac{D}{q} + \frac{A_1(n-2)}{q} + \frac{A_1(pr - q^2)(n-3)}{q^3}$$

with $D = B_1/q + A_1(pr - q^2)/q^2$.

Proof. We want to solve A_n, B_n and the back substitution of (3) after finding out an equation for H_n. Starting with (3), we can apply a well known generalized procedure to solve these type of recurrences by taking as input data the recurrence H_n. Thus, if $f_n = q$ for all n, we have the recurrence $A_{n+1} - f_n A_n = pH_{n+1}$. The recurrence is therefore equivalent to

$$\frac{A_{n+1}}{\prod_{k=0}^{n} f_k} - \frac{A_n}{\prod_{k=0}^{n-1} f_k} = \frac{pH_{n+1}}{\prod_{k=0}^{n} f_k}, \tag{5}$$

which after taking the sums we end up with

$$\begin{aligned} A_{n+1} &= q^{n+1} \left(\frac{A_1}{q} + p \sum_{k=1}^{n} \frac{H_{k+1}}{q^{k+1}} \right) \qquad (6) \\ A_{n+1} &= q^n A_1 + p q^{n+1} \sum_{k=1}^{n} \frac{H_{k+1}}{q^{k+1}} \\ A_{n-1} &= q^{n-2} A_1 + p q^{n-1} \sum_{k=1}^{n-2} \frac{H_{k+1}}{q^{k+1}} \end{aligned}$$

If now we solve for B_{n+1}, using (3), we have the recurrence $B_{n+1} - qB_n = qA_{n+1} - q^2A_n + prA_n$ and by using the same procedure as for A_n, we have

$$\begin{aligned} B_{n+1} &= q^{n+1}\left(\frac{B_1}{q} + \sum_{s=1}^{n}\frac{qA_{s+1} + (pr-q^2)A_s}{q^{s+1}}\right) \\ &= q^nB_1 + q^{n+1}\sum_{s=1}^{n}\frac{qA_{s+1} + (pr-q^2)A_s}{q^{s+1}} \\ B_{n-1} &= q^{n-2}B_1 + q^{n-1}\sum_{s=1}^{n-2}\frac{qA_{s+1} + (pr-q^2)A_s}{q^{s+1}} \end{aligned} \tag{7}$$

For B_{n-1} we obtain,

$$\begin{aligned} B_{n-1} &= q^{n-2}B_1 + q^{n-1}\sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} + (pr-q^2)q^{n-1}\sum_{s=1}^{n-2}\frac{A_s}{q^{s+1}} \\ &= q^{n-2}B_1 + q^{n-1}\sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} + (pr-q^2)q^{n-1}\sum_{s=1}^{n-2}\frac{A_s}{q^{s+1}} \\ &= q^{n-2}B_1 + (pr-q^2)q^{n-3}A_1 + q^{n-1}\sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} + (pr-q^2)q^{n-1}\sum_{s=2}^{n-2}\frac{A_s}{q^{s+1}} \\ &= D + q^{n-1}\sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} + (pr-q^2)q^{n-1}\sum_{s=2}^{n-2}\frac{A_s}{q^{s+1}} \end{aligned}$$

where we have made

$$D(p,q,r,n) = q^{n-2}B_1 + (pr-q^2)q^{n-3}A_1 \tag{8}$$

From Equation (6) we have that,

$$\begin{aligned} \frac{A_{s+1}}{q^s} &= A_1 + pq\sum_{k=1}^{n}\frac{H_{k+1}}{q^{k+1}} \\ \sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} &= A_1(n-2) + pq\sum_{s=1}^{n-2}\sum_{k=1}^{s}\frac{H_{k+1}}{q^{k+1}} \\ q^{n-1}\sum_{s=1}^{n-2}\frac{A_{s+1}}{q^s} &= A_1(n-2)q^{n-1} + pq^n\sum_{s=1}^{n-2}\sum_{k=1}^{s}\frac{H_{k+1}}{q^{k+1}} \end{aligned}$$

and

$$\begin{aligned}
\sum_{s=2}^{n-2} \frac{A_s}{q^{s+1}} &= \frac{A_1(n-3)}{q^2} + \frac{p}{q} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}} \\
q^{n-1} \sum_{s=2}^{n-2} \frac{A_s}{q^{s+1}} &= A_1(n-3)q^{n-3} + pq^{n-2} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}} \\
(pr-q^2)q^{n-1} \sum_{s=2}^{n-2} \frac{A_s}{q^{s+1}} &= A_1(pr-q^2)(n-3)q^{n-3} + p(pr-q^2)q^{n-2} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}}
\end{aligned}$$

Therefore,

$$\begin{aligned}
B_{n-1} &= D + q^{n-1}A_1(n-2) + pq^n \sum_{s=1}^{n-2} \sum_{k=1}^{s} \frac{H_{k+1}}{q^{k+1}} \\
&\quad + A_1(pr-q^2)(n-3)q^{n-3} + p(pr-q^2)q^{n-2} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}}
\end{aligned} \tag{9}$$

Substitution of (8) by using the above equations and also A_{n-1} from (6) into the original recurrence for H_n in the Formula (3). We obtain a recurrence for H_n, as follows:

$$\begin{aligned}
H_{n+1} = pH_n &\\
&+ q^{n-1}A_1 + pq^n \sum_{k=1}^{n-2} \frac{H_{k+1}}{q^{k+1}} \\
&+ D + A_1q^{n-1}(n-2) \\
&+ pq^n \sum_{s=1}^{n-2} \sum_{k=1}^{s} \frac{H_{k+1}}{q^{k+1}} + A_1(pr-q^2)(n-3)q^{n-3} \\
&+ p(pr-q^2)q^{n-2} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}}.
\end{aligned} \tag{10}$$

Dividing by q^{n+1} and making $y_n = H_n/q^n$ we have

$$\begin{aligned}
\frac{H_{n+1}}{q^{n+1}} = \frac{p}{q}\frac{H_n}{q^n} &\\
&+ \frac{A_1}{q^2} + \frac{p}{q} \sum_{k=1}^{n-2} \frac{H_{k+1}}{q^{k+1}} \\
&+ \frac{D}{q^{n+1}} + \frac{A_1(n-2)}{q^2} \\
&+ \frac{p}{q} \sum_{s=1}^{n-2} \sum_{k=1}^{s} \frac{H_{k+1}}{q^{k+1}} + \frac{A_1(pr-q^2)(n-3)}{q^4} \\
&+ \frac{p(pr-q^2)}{q^3} \sum_{s=2}^{n-2} \sum_{k=1}^{s-1} \frac{H_{k+1}}{q^{k+1}}.
\end{aligned} \tag{11}$$

In oredr to simplify the equation, we will make

$$\begin{aligned}\frac{D(p,q,r,n)}{q^{n+1}} &= \frac{B_1}{q^3}+\frac{(pr-q^2)A_1}{q^4}\\ x_n &= \frac{A_1}{q^2}+\frac{D}{q^{n+1}}+\frac{A_1(n-2)}{q^2}+\frac{A_1(pr-q^2)(n-3)}{q^4}\\ P &= \frac{p(pr-q^2)}{q^3}\\ Q &= \frac{p}{q}\end{aligned}$$

Equation (11) reduces to

$$y_{n+1} = Qy_n + Q\sum_{k=1}^{n-2} y_{k+1} + Q\sum_{s=1}^{n-2}\sum_{k=1}^{s} y_{k+1} + P\sum_{s=2}^{n-2}\sum_{k=1}^{s-1} y_{k+1} + x_n. \tag{12}$$

Or,

$$y_{n+1} = Q\sum_{k=1}^{n-1} y_{k+1} + Q\sum_{s=1}^{n-2}\sum_{k=1}^{s} y_{k+1} + P\sum_{s=2}^{n-2}\sum_{k=1}^{s-1} y_{k+1} + x_n. \tag{13}$$

And also,

$$y_{n+1} = Q\sum_{s=1}^{n-1}\sum_{k=1}^{s} y_{k+1} + P\sum_{s=2}^{n-2}\sum_{k=1}^{s-1} y_{k+1} + x_n. \tag{14}$$

Let us calculate the double sum,

$$\begin{aligned}\sum_{s=1}^{n-1}\sum_{k=1}^{s} y_{k+1} &= y_2, n=2\\ &\quad + y_2 + y_3, n=3\\ &\quad + y_2 + y_3 + y_4, n=4\\ &\quad + y_2 + y_3 + y_4 + y_5, n=5\\ &\quad + y_2 + y_3 + y_4 + y_5 + y_6, n=6\\ &\quad \vdots\\ &\quad + y_2 + y_3 + y_4 + y_5 + y_6 + \cdots + y_n, n-1\\ &= (n-1)y_2 + (n-2)y_3 + (n-3)y_4 + \cdots +\\ &\quad 4y_{n-3} + 3y_{n-2} + 2y_{n-1} + y_n, n-1\\ &= \sum_{j=1}^{n-1}(n-j)y_{j+1}\end{aligned}$$

Now for the second summation,

$$\sum_{s=2}^{n-2}\sum_{k=1}^{s-1} y_{k+1} = \sum_{j=1}^{n-3}(n-j-2)y_{j+1}$$

Then, we obtain the reduced recurrences,

$$y_{n+1} = Q\sum_{j=1}^{n-1}(n-j)y_{j+1} + P\sum_{j=1}^{n-3}(n-j-2)y_{j+1} + x_n \tag{15}$$

Or after making $s = j+1$

$$y_{n+1} = Q\sum_{s=2}^{n}(n-s+1)y_s + P\sum_{s=2}^{n-2}(n-s-1)y_s + x_n \tag{16}$$

$$\begin{aligned} c_0y_8 + c_1y_7 + c_2y_6 + c_3y_5 + c_4y_4 + c_5y_3 + c_6y_2 &= x_7, \quad n = 7 \\ c_0y_7 + c_1y_6 + c_2y_5 + c_3y_4 + c_4y_3 + c_5y_2 &= x_6, \quad n = 6 \\ c_0y_6 + c_1y_5 + c_2y_4 + c_3y_3 + c_4y_2 &= x_5, \quad n = 5 \\ c_0y_5 + c_1y_4 + c_2y_3 + c_3y_2 &= x_4, \quad n = 4 \end{aligned}$$

Now, if we make $z_n = x_n - \sum_{s=2}^{4} c_{4-s+1}y_s$, then

$$\begin{pmatrix} c_0 & c_1 & \cdots & c_{n-5} \\ 0 & c_0 & \cdots & c_{n-4} \\ \vdots & & & \vdots \\ 0 & \cdots & 0 & c_0 \end{pmatrix}\begin{pmatrix} y_n \\ y_{n-1} \\ \vdots \\ y_5 \end{pmatrix} = \begin{pmatrix} z_{n-1} \\ z_{n-2} \\ \vdots \\ z_4 \end{pmatrix}$$

In order to solve the above system of equations, we use the following lemma, where the proof can be found elsewhere. For completeness, we are reproducing it on its more general form:

Lemma 1. *Let* $T = \Lambda + T_u$ *be an upper triangular matrix, where* $\Lambda = (\delta_{ij}t_{ij})$ *then* $T^{-1} = \left(I + \Lambda^{-1}T_u\right)^{-1}\Lambda^{-1}$ *and*

$$\left(I + \Lambda^{-1}T_u\right)^{-1} = \sum_{j=0}^{m}\left(-\Lambda^{-1}T_u\right)^j \tag{17}$$

Thus from lemma (1), the theorem follows.

Conclusions

We have presented a closed formula for computing the Merrifield-Simmons index of polygonal array graphs at distance 1, and a method to compute the closed formula for a polygonal array at any distance based on its associated recurrence. Although the Merrifield-Simmons index has been previously exhibited for polygonal edge joining, it has not been tackled with its associated closed formula.

Bibliography

[1] W.C. Shiu, Extremal Hosoya index and Merrifield-Simmons index of hexagonal spiders, Discrete Applied Mathematics 156 (2008) 2978–2985.

[2] T. Došli'c, F. Måløy, Chain hexagonal cacti: Matchings and independent sets, Discrete Mathematics, 310 (2010) 1676–1690.

[3] S. Wagner, I. Gutman, Maxima and minima of the Hosoya index and the Merrifield-Simmons index: A survey of results and techniques, Acta Applicandae Mathematicae 112(3) (2010) 323–346.

[4] I. Gutman, Extremal hexagonal chains, J. of Mathematical Chemistry, 12 (1993) 197–210.

[5] L. Zhang, The proof of Gutman's conjectures concerning extremal hexagonal chains, Jour. Systems Sci. Math. Sci., 18(4) (1998) 460–465.

[6] L.-Z. Zhang. On the ordering of a class of hexagonal chains with respect to Merrifield-Simmons index, Systems Sci. Math. Sci., 13(2) (2000) 219–224.

[7] L.-Z. Zhang and F.-J. Zhang. Extremal hexagonal chains concerning k-matchings and k-independent sets, J. Math. Chem., 27(4), (2000) 319–329.

[8] G. De Ita, J.R. Marcial-Romero, P. Bello, M. Contreras, Linear-time algorithms for computing the Merrifield-Simmons index on polygonal trees, MATCH Comm. in Math. Comp. Chem. 69:1 (2018) 55–78.

[9] T. Došli'c, M. S. Litz, Matching and independent sets in polyphenylene chains, MATCH Comm. in Math. Comp. Chem. 67 (2012) 313–330.

email from authors

Hernández-Servín, J.A.: xoseahernandez@uaemex.mx

Marcial-Romero, J.R.: jrmarcialr@uaemex.mx

De Ita Luna, G.: email@example.com

Informaticae Abstracta
Año 2023, Vol. 1, No. 1, pp. 30-42 JUL | DIC 2023

Un modelo de programación entera para el problema de horarios universitarios

Francisco, I.; Marcial Romero, J. R.; Hernández-Servín, J. A.

Información del Artículo

Recibido: 20-mar-2023
Aceptado: 28-may-2023

Palabras clave: Programación entera, Timetabling, Períodos consecutivos

Resumen

Se presenta un modelo de programación entera para resolver un caso de estudio para el problema de horarios universitarios de la Facultad de Ingeniería de la Universidad Autónoma del Estado de México. El modelo toma en consideración las preferencias de días y horas que el personal académico puede elegir para impartir clases dentro de una escala de 0 a 5, donde 0 representa la mayor preferencia y 5 representa la menos preferida para dar clases. Se presentan dos formulaciones para la generación de horarios con clases en períodos consecutivos, estas formulaciones se comparan con las propuestas de otros autores.

1. Introducción

El proceso de generación de horarios escolares se conoce como Carga Horaria o Timetabling, se dice que un horario es eficiente si presenta la menor cantidad de solapamientos entre clases y si además cumple con una serie de condiciones impuestas por la institución académica [1].

La generación de horarios puede variar dependiendo del nivel educativo y de las condiciones que se deben cumplir, es por ello que en [2] se propone la siguiente clasificación: - Carga horaria escolar: es la programación semanal de todas las clases de una escuela, evitando que los profesores tenga dos clases o más al mismo tiempo. - Carga horaria universitaria: es la programación semanal de todas las clases de un conjunto de cursos de alguna universidad, minimizando los conflictos entre clases que tienen alumnos en común. - Carga horaria de exámenes: esta es la programación de un conjunto de exámenes de una universidad, evitando programar al mismo tiempo dos exámenes con alumnos en común, además se debe dar el tiempo necesario entre exámenes para que los alumnos tengan un periodo de descanso y preparación.

Se sabe que el problema de la carga horaria es NPCompleto [1] [2] [3], este es el motivo por el cual encontrar una solución sin conflictos puede llevar un tiempo exponencial, ya que para ello se requiere examinar las combinaciones posibles entre los conjuntos de asignaturas, profesores, alumnos, salones y periodos de tiempo, para obtener aquella combinación en la cual no exista ningin conflicto o en su caso aquella que contenga la menor cantidad. Debido a esto, el problema de la carga horaria se ha abordado desde diferentes áreas y enfoques, se ha tomado como un problema de búsqueda, ya que el objetivo es encontrar una solución que cumpla con las condiciones impuestas por la institución, para ello se ha aplicado técnicas como búsqueda tabú, temple simulado, algoritmos genéticos, y búsquedas basadas en heurísticas [2], las cuales garantizan encontrar una solución cercana a la óptima. Otro enfoque que se ha dado, es tratar al problema de la carga horaria como un problema de optimización, ya que no únicamente se busca encontrar una

solución, sino aquella que tenga la menor cantidad de conflictos, por lo cual se puede abordar por medio de Programación Lineal y resolver utilizando los algoritmos desarrollados para el área de Investigación de operaciones como Algoritmo de Ramificación y Acotamiento [4], Algoritmo de Ramificación y Corte [4] [5].

Para probar la propuesta que se presenta en este articulo, se ha tomado como caso de estudio la Facultad de Ingeniería (FI) de la Universidad Autónoma del Estado de México (UAEM) que ofrece las carreras de Ingenieria Civil, Ingeniería en Computación, Ingeniería en Electrónica, Ingeniería Mecánica y Sistemas Energéticos Sustentables. El proceso de generación de horarios escolares consiste en lo siguiente, cada profesor de cada carrera plantea el horario en que desea impartir sus clases, estos horarios se compilan por cada coordinación y son enviados a Control Escolar donde se autorizan siempre y cuando no entren en conflicto con horarios de otras carreras, ya sea porque no existen suficientes salones disponibles o porque los dias o periodos ya han sido asignados, en caso contrario se reasignan.

La FI, a diferencia de las demás Facultades pertenecientes a la UAEM, oferta en cada semestre todas las asignaturas dentro del plan de estudios de cada carrera, los alumnos que se inscriben deben elegir qué materias cursarán, esto permite que los alumnos puedan adelantar asignaturas de otros semestres siempre y cuando no estén serializadas con asignaturas que aún no han cursado. Debido a esto la FI crea grupos de alumnos en base a las asignaturas, por lo cual un alumno puede pertenecer a un grupo a en la asignatura A y su vez pertenecer al grupo b en la asignatura B, donde A y B se intersectan lo cual dificulta la generación de horarios. Otra característica que presenta la FI, es que existen asignaturas que son de tronco común para las cinco carreras, los grupos de estas asignaturas pueden estar formados por alumnos de distintas carreras y distintos semestres.

En este artículo se presenta un modelo de programación entera para la generación de horarios escolares de la FI que toma en cuenta las preferencias de los profesores, y plantea los horarios en base a la tupla grupo-alumno.

Este modelo es un prototipo para el desarrollo de un sistema automatizado para la generación de horarios escolares de la FI.

2. Modelo para el caso de estudio

Daskalaki en [1] propone un modelo de programación entera 0-1 para el problema de carga horaria de la Universidad de Patras, en ésta se ofrecen cursos de dos tipos: obligatorios y opcionales. Los cursos obligatorios están diseñados para los alumnos de los primeros años, por el contrario los cursos opcionales son para los alumnos que están en los últimos años de la carrera, además en el modelo se toma en cuenta que algunos de los cursos ofrecidos requieren de laboratorios en lugar de un salón clases. En [1] se toman en cuenta las siguientes restricciones: - Cada profesor debe ser asignado a lo más a un curso, un grupo de estudiantes y a un salón a la vez. - Para cada grupo de estudiantes a lo más un curso, un profesor y un salón debe ser asignado en cada periodo. - Cada salón debe ser asignado a lo más un curso, un profesor y a un grupo de estudiantes a la vez. - Cada curso en el plan de estudios de cada estudiante debe ser programado en la cantidad exacta de periodos de tiempo. - Cada curso debe ser programado tantos periodos como el plan de estudios de cada grupo de estudiantes requiera. - Cada profesor debe

ser asignado a tantos periodos a la semana como su carga de trabajo requiera. - Un curso c que requiere una sesión de $h_\nu \in H_c$ periodos consecutivos, deberia ser asignado exactamente h_ν periodos en el día especificado. - Si h_ν periodos de un día en especifico se asignan al curso c, estos deben ser consecutivos. - Las h_ν sesiones deben ser tantas como se requieran semanalmente y deben tener por lo menos una sesión al día.

El modelo Daskalaki et al es uno de los más completos ya que, toma en consideración la mayoría de restricciones al problema: cursos, alumnos, profesores, salones, dias y periodos de tiempo y su objetivo es minimizar los conflictos entre las clases, Para Daskalaki et. al. un horario debe cumplir cuatro condiciones: - El horario no debe tener colisiones tanto para cursos, alumnos, profesores y salones. - El horario debe ser completo para ello todos los cursos deben ser programados la cantidad requerida por el plan de estudios - El horario debe ser flexible tomando en consideración si el profesor desea impartir su curso en varias sesiones consecutivas en un dia o repartidas durante la semana. - El horario debe permitir la programación de una sesión en dos o más clases, por ejemplo si un curso requiere de un laboratorio cuya capacidad sea menor a la cantidad de alumnos inscritos al curso, el horario debe ser capaz de programar dos o más sesiones a la semana.

Se puede desarrollar un modelo especifico para la Facultad de Ingenieria de la Universidad Autónoma del Estado de México, tomando como base el modelo de Daskalaki.

Restricciones de la Facultad de Ingeniería

Los horarios generados para la Facultad de Ingenieria deben cumplir con las siguientes restricciones: - Ningún profesor puede impartir más de una clase al mismo tiempo

- Ningún salón puede ser ocupado por más de una clase al mismo tiempo - Ningún salón puede ser ocupado por una clase cuyo grupo de alumnos supere su capacidad - Cada asignatura debe ser programa tantas horas como se especifique en el plan de estudios - Una asignatura puede ser programa a lo máximo 2 horas diarias - Si una asignatura se programa en más de un periodo en un dia estos deben ser consecutivos

3. Modelo para la FI

En esta sección se presenta el modelo de programación entera para la Facultad de Ingenieria de la Universidad Autónoma del Estado de México.

Notación

Conjuntos del modelo: - $D = (1, \cdots, 6)$, dias en que la facultad está disponible para impartir clases. - $H = \{1, \cdots, 30\}$, periodos de tiempo en que la facultad puede impartir clases, estos periodos están compuestos por 30 minutos. - $P = \{1, \cdot\cdot, |P|\}$, profesores que imparten clases en la facultad. - $C = \{1, \cdots, |C|\}$, grupo-asignatura quienes reciben clases. - $S = \{1, \cdots, |S|\}$, salones en los cuales se imparten clases. Subconjuntos: - P_c, profesores que imparten clases al grupo-asignatura c. - P_{dh},

profesores disponibles en el dia d y la hora h. - C_p, grupo-asignatura a los que imparte clase el profesor p. - C_{ps}, clases que imparte el profesor p y puede ser albergada en el salón s. - S_c, salones en que la clase c no supera su capacidad. - D_p, dias en que el profesor p está disponible. - H_{pd}, horas del dia den que el profesor p está disponible. - HF_{pd}, hora inicial en el día d en que el profesor p está disponible. - h_c, horas semanales requeridas por la asignatura c. - h_d, horas diarias máximas

Variables

Para el modelo se tiene la siguiente variable de decisión x_{dhpes} La cual toma el valor de 1 si se programa un clase en el dia d en el periodo h, y es impartida por el profesor p al grupoasignatura c en salón s, en caso de que no se programe alguna clase, la variable toma el valor de 0 .

Función Objetivo

La función objetivo está orientada a tomar en consideración las preferencias de que dia y hora el profesor prefiere impartir sus clases, para ello se hace uso de la constante $T_{dhpc} \in (0, 1, 2, 3, 4, 5)$ donde 0 indica mayor preferencia y 5 la menor.

$$\min \sum_{d\oplus} \sum_{h\in H} \sum_{p\in P_{dh}} \sum_{c\in C_p} \sum_{s\in S_c} T_{dhpc} x_{dhpcs}$$

Restricciones Básicas

Ningún profesor puede impartir más de un clase al mismo tiempo.

$$\forall d \in D, \forall h \in H, \forall p \in P_{dh} \quad \sum_{c\in C_p} \sum_{s\in S_c} x_{dhpcs} \leqslant 1$$

Ningún salón puede ser ocupado por más de una clase al mismo tiempo.

$$\forall s \in S, \forall d \in D, \forall h \in H \quad \sum_{p\in P_{dh}} \sum_{c\in C_{ps}} x_{dhpcs} \leqslant 1$$

Cada asignatura debe ser programa tantas horas semanales como se especifique en el plan de estudios.

$$\forall c \in C, \forall p \in P_c \sum_{d\in D_p} \sum_{n\in H_{pd}} \sum_{s\in S_c} x_{dhpcs} = h_c$$

Cada asignatura debe programarse a lo mucho 2 horas dias.

$$\forall d \in D, \forall p \in P_d, \forall c \in C_p \sum_{s\in s_c} \sum_{h\in H_{pd}} x_{dhpcs} \leqslant h_d$$

4. Restricciones clases consecutivas

En esta sección se muestra cuatro técnicas para obtener clases consecutivas, una basada en una extensión del modelo propuesto por Daskalaki en [1], otra en una extensión del modelo de Barki y Aksop propuesto en [6], y dos restantes ideadas en este artículo con la finalidad de obtener la mejor solución posible.

Restricción Daskalaki

Daskalaki en [1] propone una serie de restricciones para obtener clases consecutivas, adaptadas a la Facultad de Ingenieria quedando como:

$$\forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall f \in HF_{pd},$$
$$\forall t \in \{1, \cdots, h_v - 1\}\, x_{dfpcs} - x_{d(f+t)pcs} \leqslant 0$$
$$\forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall h \in H_{pdt}$$
$$\forall t \in \{2, \cdots, h_v\}$$
$$-x_{dhpcs} + x_{d(h+1)pcs} - x_{d(h+\varepsilon)pcs} \leqslant 0$$

Donde HF_{pd} es el periodo inicial del conjunto de los periodos en que el profesor p está disponible para impartir clases en el día d, h_v es el número de periodos consecutivos requeridos para la clase c.

Ahora supóngase que existan 4 periodos $X = \{x_1, x_2, x_3, x_4\}$, donde se desea programar una clase con $h_v = 2$ periodos consecutivos. Las restricciones generadas en base a Daskalaki son:

$$x_1 - x_2 \leqslant 0$$
$$-x_1 + x_2 - x_3 \leqslant 0$$
$$-x_2 + x_3 - x_4 \leqslant 0$$

Uno de los posibles valores para X que cumpla con las restricciones anteriores es $X = \{1, 1, 0, 1\}$, es decir se pueden programar clases consecutivas permitiendo periodos vacios. En dado caso que únicamente se requiere programar una única clase, las restricciones de Daskalaki únicamente permiten que sea en el periodo final, por lo cual $X = \{1, 0, 0, 0\}$, $X = \{0, 1, 0, 0)$ y $X = \{0, 0, 1, 0)$ no son aceptadas.

Restricción Barki y Aksop

Barki y Aksop en [6] agregan la siguiente restricción a las propuestas por Daskalaki en [1]. $\forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall l \in HL_{pd}, \forall t \in \{1, \cdots, h_p - 1\}\, x_{dhpcs} - x_{d(1-t)pes} \leqslant 0$ Donde HL_{pd} es el periodo final del conjunto de los periodos en que el profesor p está disponible para impartir clases en el dia d.

Al igual que Daskalaki, Barki y Aksop permiten huecos y únicamente permite la programación de clases consecutivas de h_v o más horas consecutivas.

Formulación 1

La siguiente restricción esta ideada para programar las clases desde 0 hasta el número total de periodos disponibles de los profesores. Para el desarrollo de esta restricción se tomó como base la propuesta por Avella en [6]. Supóngase que existen únicamente 3 periodos consecutivos $X = \{x_1, x_2, x_3\}$, una clase se puede programar de la siguiente manera: - No se programe en ningún periodo - Se programe en solo uno de los periodos - Se programe en dos periodos que sean consecutivos - Se programe en los 3 periodos disponibles Lo anterior se puede cumplir con la siguiente restricción:

$$x_1 - x_2 + x_3 \leqslant 1$$

Para cualquier cantidad de periodos, la restricción puede generalizase como:

$$\forall j \in J, \forall k \in \{j+2, \cdots, |J|\} \quad x_j - x_{j+1} + x_k \leqslant 1$$

Donde J es el conjunto de periodos. La idea radica en tomar tres periodos y determinar que el segundo periodo al tomar el valor de 0 no provoque huecos, ver Tabla I.

TABLA I. FORMULACIÓN 1 PARA 3 PERÍODOS DE TIEMPO

Petiodos			Restrioción
x_j	x_{j+1}	x_k	$x_j - x_{j+1} + x_k \leqslant 1$
0	0	0	V
0	0	1	V
0	1	0	V
0	1	1	V
1	0	0	V
1	0	1	F
1	1	0	V
1	1	1	V

Para el modelo, las restricciones que determinan que las clases se programen en periodos consecutivos en base a esta formulación queda como:

$$\forall d \in D, \forall p \in P_d, \forall c \in C_p \sum_{s \in S_c} y_{dcps} \leqslant 1$$

$$y \in \{0,1\}$$

$$\forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall h \in H$$

$$x_{dhpcs} \leqslant y_{dpcs}$$

$$\forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall h \in H_{pd},$$

$$\forall t \in \left\{h+2, \cdots, \left|H_{pd}\right|\right\}$$

$$x_{dhpcs} - x_{d(h+1)pcs} + x_{dtpcs} \leqslant 1$$

Donde (13) determina el salón en que se programará la clase, la restricción (14) asegura que no se programen clases en más de un salón y la restricción (15) determina los periodos consecutivos. D. Formulación 2 Se puede obtener una segunda formulación utilizando el siguiente enfoque, supongamos que se tienen dos periodos cualquiera x_j y x_{j+1}, donde x_j es el periodo en el cual deben iniciar las clases en periodos consecutivos, por lo cual una clase se puede programar de la siguiente manera: - No se programe en ningún periodo. - Se programe en solo uno de los periodos. - Se programe en los dos periodos. Estos casos se pueden obtener por medio de la siguiente restricción, ver Tabla II:

$$x_i - x_{i+1} \geqslant 0$$

Para el caso de 3 periodos $x_j, x_{j+1} y x_{j+2}$, las siguientes restricciones son necesarias para obtener clases consecutivas con el periodo inicial en x_j :

$$x_j - x_{j+1} \geqslant 0$$

$$x_{j+1} - x_{j+2} \geqslant 0$$

TABLA II FORMULACIOŃ 2 PARA 2 PERIODOS DE TIEMPO

Periodos		Restricción
x_j	x_{j+1}	$x_j - x_{j+1} \geqslant 0$
0	0	V
0	1	F
1	0	V
1	1	V

Si el periodo inicial es en x_{j+1}, se requieren la restricción (18) y la siguiente:

$$x_i \leqslant 0$$

En cambio si el periodo inicial es en x_{j+2} se requieren las restricciones (17) y (19). Se pueden agrupar las tres restricciones (17), (18) y (19), y dependiendo del periodo inicial desactivar la restricción correspondiente. Para x_j la restricción (19) se desactiva, para x_{j+1} la restricción (17) y para x_{j+2} la restricción (18), ver la Tabla III.

TABLA III FORMULACIOŃ 2 PARA 3 PERIODOS DE TIEMPO

Periodos			Restricciones		
x_j	x_{j+1}	x_{j+2}	$x_j - x_{j+1} \geqslant 0$	$x_{j+1} - x_{j+2} \geqslant 0$	$x_j \leqslant 0$
0	0	0	V	V	V
0	0	1	V	F	V
0	1	0	F	V	V
0	1	1	F	V	V
1	0	0	V	V	F
1	0	1	V	F	F
1	1	0	V	V	F
1	1	1	V	V	F

Para desactivar restricciones se hace uso de la técnica K out of N Constraints Must Hold [7] [8] [9], que establece que dadas N posibles restricciones:

$$\begin{gathered} f_1(x_1, x_2, \cdots, x_n) \leqslant d_1 \\ f_2(x_1, x_2, \cdots, x_n) \leqslant d_2 \\ \vdots \\ f_N(x_1, x_2, \cdots, x_n) \leqslant d_N \end{gathered}$$

Se pueden tener N restricciones activas haciendo uso de la siguiente formulacion:

$$\begin{gathered} f_1(x_1, x_2, \cdots, x_n) \leqslant d_1 + My_1 \\ f_2(x_1, x_2, \cdots, x_n) \leqslant d_2 + My_2 \\ \vdots \\ f_N(x_1, x_2, \cdots, x_n) \leqslant d_N + My_N \\ \sum_{i=1}^{N} y_i = N - K \end{gathered}$$

Donde y_i es una variable auxiliar binaria que toma el valor de 1 si la restricción esta desactivada o 0 si esta activada. M es un entero positivo lo suficientemente grande de tal manera que la restricción se satisfaga para cualquier valor de las variables, en este caso $M = 1$. De esta manera se obtiene la siguiente formulación para 3 periodos:

$$\begin{aligned} & x_j \leqslant y_j \\ & -x_j + x_{j+1} \leqslant y_{j+1} \\ & -x_{j+1} + x_{j+2} \leqslant y_{j+2} \\ & y_j + y_{j+1} + y_{j+2} = 1 \end{aligned}$$

Si se sustituye y_j por x_j en (25), se puede eliminar la restricción (22) y obtener el mismo resultado. Para el modelo de la Facultad de Ingenieria, la formulación que determina que las clases se programen de manera consecutivas queda como:

$$\begin{aligned} & \forall d \in D, \forall p \in P_d, \forall c \in C_p, \forall s \in S_c, \forall h \in H_{pd} \\ & -x_{dhpcs} + x_{d(h+1)pcs} \leqslant y_{d(h+1)pcs} \\ & \forall d \in D, \forall p \in P_d, \forall c \in C_p \\ & \sum_{s \in S_c} \sum_{f \in HF_{pd}} x_{dhpcs} + \sum_{s \in S_c} \sum_{h \in H_{pd}} y_{d(h+1)pcs} \leqslant 1 \end{aligned}$$

Con la restricción (27) además de determinar el periodo inicial, también se determina el salón en que se programen las clases.

Comparación entre formulaciones

Para eyey y Nemhauser en [10] no basta únicamente con el número de restricciones y el número de variables para determinar la calidad de una formulación, sino que influye desde el tamaño de los números en las operaciones básicas hasta el algoritmo que se usa para resolverlo. Sin embargo Wolsey y Nemhauser establecen que para un programa entero, la complejidad en tiempo al resolverse por medio de fuerza bruta es de:

$$O\left(2^n mn\right)$$

Donde n es el número de variables y m el número de restricciones.

Las restricciones generadas, para una única clase, por las diferentes formulaciones es de: - Daskalaki

$$\begin{cases} \frac{(h_v-1)\left(2\left|H_{pd}\right|-h_v\right)}{2} & 2 \leqslant h_v < \left|H_{pd}\right| \\ \frac{\left|H_{pd}\right|\left(\left|H_{pd}\right|-1\right)}{2} & h_v = \left|H_{pd}\right| \end{cases}$$

- Bakir

$$\begin{cases} \frac{(h_v-1)\left(2\left|H_{pd}\right|-h_v+2\right)}{2} & 2 \leqslant h_v < \left|H_{pd}\right| \\ \frac{\left(\left|H_{pd}\right|-1\right)\left(\left|H_{pd}\right|+2\right)}{2} & h_v = \left|H_{pd}\right| \end{cases}$$

- Formulación 1

$$\frac{\left(\left|H_{pd}\right|-2\right)\left(\left|H_{pd}\right|-1\right)}{2}+2$$

- Formulación 2

$$\left|H_{pd}\right|$$

Las variables generadas por las diferentes formulaciones es de: - Daskalaki

$$\left|H_{pd}\right|$$

- Bakir

$$\left|H_{pd}\right|$$

- Formulación 1

$$\left|H_{pd}\right| + 1$$

- Formulación 2

$$2\left|H_{pd}\right| - 1$$

Con base en la complejidad de Wolsey y Nemhauser, la formulación 1 es la más fácil de resolver a comparación de la formulación 2 debido a la cantidad de variables que maneja.

5. Solución del Modelo

Para probar el modelo se uso el solucionador CBC (Coin-or branch and cut) en su versión 2.8 en un equipo con procesador i7 y 8 GB de memoria ram.

Control Escolar de la Facultad de Ingeniería proporcionó datos de sus profesores y sus cursos para el periodo escolar 2015 B, estos se dividieron en base a las carreras que ofrece la facultad, tomando las asignaturas de tronco común como una instancia adicional, ver la Tabla IV.

Tabla 4 Instancias

	PROFESORES	CLASES	SALONES
CIVIL	78	112	22
COMPUTACIÓN	81	130	22
ELECTRÓNICA	42	61	22
MECÁNICA	54	80	22
ENERGÉTICOS	25	41	22
COMÚN	137	230	22

Se tomaron en cuentan las siguientes consideraciones

1. La disponibilidad de los Profesores se asignaron de manera aleatoria.
2. La cantidad de horas semanales de las clases se asignó de manera aleatoria.
3. Las preferencias de los Profesores se asignaron de manera aleatoria.
4. Se utilizó la instancia Energéticos.
5. Se dio un tiempo máximo de ejecución de 2 horas.
6. Para las formulaciones Daskalaki y Bakir se uso $h_v = 2$.

6. Resultados

Los resultados de la ejecución del modelo utilizando Coin CBC, con la instancia Energéticos se observan en la Tabla V.

En la Tabla V se observa que la formulación de Bakir y la Formulación 1 propuesta en este documento no pueden dar ninguna solución entera en un periodo menor a 2 horas. Tanto Daskalaki y la Formulación 2, resuelven el modelo en menos de 2 horas sin embargo Daskalaki requiere de mayor cantidad de iteraciones y con ello de mayor tiempo. Basándonos en el valor de objetivo, la Formulación 2 toma en cuenta mejor las preferencias de los profesores.

TABLA V RESULTADOS DE LA EJECUCION DE INSTANCIA ENERGETICOS

	Daskalaki	Bakir	Formulación #1	Formulación #2
Valor objetivo lineal	252.33	209.20	128.00	209.50
Valor objetivo entero	259.00	-	-	213.00
Nados enumerados	2486	-	-	50
Iteraciones	131289	-	-	2764
Tiempo (seg)	205.74	-	-	94.91

Para comparar los horarios generados por las diferentes formulaciones se tomó el profesor con identificador 177, que imparte las clases Ciclos Avanzados con identificador 1053 de 7 horas semanales, Aplicaciones No Eléctricas de la Geotermia con identificador 474 y 9 horas semanales, y Sistemas Geotérmicos con identificador 994 y 11 horas semanales.

Aunque con la formulación de Daskalaki genera horarios en poco tiempo, presenta huecos, esto se puede ver con la clase 474, el dia lunes donde existe una separación de 13 periodos entre el segundo y tercer periodo de la clase. Otro problema que presenta Daskalaki es que logra programar clases consecutivas pero en diferentes salones tal es el caso de la clase 1053 en el día viernes. En el horario generado por la formulación 2, las clases se programan correctamente pero tiende a programar clases de un solo período de tiempo en distintos dias de la semana.

TABLA VI Horario generado utilizando Daskalaki

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
7 : 00			994			1053
7 : 30	474.		994			1053
8 : 00	474					
8,30		474				
9 : 00		474			1053	
9 : 30	994.			994	1053	
10 : 00	994	474		994		
10 : 30	994.	474				
11 : 00						
11 : 30						
12 : 00						
12 : 30						
13 : 00						
13 : 30						
14 : 00					994	
14 : 30					994	1053
15 : 00	474			474		1053
15 : 30				474		1053

TABLA VII Horario generado utilizando Formulación 2

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
7 : 00			994			
7 : 30	474		994			
8 : 00	474					
8 : 30					474	
9 : 00					1053	
9 : 30	994				1053	994
10 : 00	994	474				
10 : 30	994	474				
11 : 00						
11 : 30						
12 : 00						
12 : 30		1053				
13 : 00		994		994		
13 : 30		994				474
14 : 00	1053		474		994	
14 : 30					994	1053
15 : 00				474		1053
15 : 30				474		1053

Se realizó una segunda prueba con la instancia Electrónica, la segunda instancia con la menor cantidad de datos, comparando únicamente a Daskalaki y la Formulación 2, se obtienen los resultados mostrados en la Tabla VIII.

En este caso la Formulación Daskalaki, no fue capaz de resolver la instancia de Electrónica en menos de 2 horas.

TABLA VIII RESULTADOS DE LA EJECUCIÓN DE LA INSTANCIA ELECTRÓNICA

	Daskalaki	Formulacion 2
Valor objetivo real	418.41	337.88
Valor objetivo entero	-	346.00
Nodos enumerados	-	0
Iteraciones	-	1328
Tiempo (seg)	-	327.45

7. Conclusiones

Como se observa en los resultados la formulación 2 obtiene mejores resultados al programar las clases en periodos consecutivos en un mismo salón, esta formulación además presenta el mejor valor objetivo, es decir el modelo toma más en consideración las preferencias de los profesores. Las restricciones de Daskalaki aunque da soluciones en un tiempo similar a la formulación 2, son malas ya que presenta huecos además de cambios de salón. La formulación 2 sin embargo en ocasiones puede programar clases de un solo periodo por día, pero esto es preferible a tener clases consecutivas en diferentes salones como los horarios generados por Daskalaki, además de generar horarios en menor cantidad de tiempo. Con los resultados actuales se puede afirmar que la formulación 2 es la adecuada para aplicarse a la Facultad de Ingeniería de la Universidad Autónoma del Estado de México ya que genera soluciones adecuadas a las restricciones de la facultad.

Bibliografía

[1] S. Daskalaki, T. Bitbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.

[2] A. Schaerf. A Survey of Automated Timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[3] D. de Werra. The combinatorics of timetabling. *European Journal of Operational Research*, 96:504–513, 1997.

[4] E. K. Burke, J. Marecek, A. J. Parkes, and H. Rudova. A branch-and-cut procedure for the Udine Course Timetabling problem. *Annals of Operations Research*, 194:71–87, 2011.

[5] P. Avella and I. Vasil'Ev. A Computational Study of a Cutting Plane Algorithm for University Course Timetabling. *Journal of Scheduling*, 8:497–514, 2005.

[6] M. A. Bakir and C. Aksop. A 0-1 integer programming approach to a university timetabling problem. *Hacettepe Journal of Mathematics and Statistics*, 37(1), 2008.

[7] F. Hillier and G. Lieberman. *Introduction to Operations Research*. McGraw-Hill International Editions, McGraw-Hill, 2009.

[8] D.-S. Chen, R. G. Batson, and Y. Dang. *Applied Integer Programming*. Wiley-Blackwell, 2009.

[9] H. P. Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[10] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*. Wiley-Interscience, 1999.

Direcciones de correo de los autores

Francisco, I.: malopezme@uaemex.mx

Marcial Romero, J. R.: jrmarcialr@uaemex.mx

Hernández-Servín, J. A.: xoseahernandez@uaemex.mx