UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

ALGORITMO PARA CALCULAR EL ÍNDICE MERRIFIELD-SIMMONS SOBRE MALLAS POLIGONALES

TESIS PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

HERLINDA GONZÁLEZ VÁZQUEZ

BAJO LA DIRECCIÓN DE:

DRA. CRISTINA LÓPEZ RAMÍREZ

EN CODIRECCIÓN:

DR. PEDRO BELLO LÓPEZ

CUNDUACÁN, TABASCO, A: MARZO 2026

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO

DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

ALGORITMO PARA CALCULAR EL ÍNDICE MERRIFIELD-SIMMONS SOBRE MALLAS POLIGONALES

TESIS PARA OBTENER EL GRADO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

HERLINDA GONZÁLEZ VÁZQUEZ

BAJO LA DIRECCIÓN DE:

DRA. CRISTINA LÓPEZ RAMÍREZ

EN CODIRECCIÓN:

DR. PEDRO BELLO LÓPEZ

CUNDUACÁN, TABASCO, A: MARZO 2026

Declaración de Autoría y Originalidad

En la Ciudad de Cunduacán el día Dos del mes de Marzo del año 2026, el que suscribe **Herlinda González Vázquez**, alumno del Programa de **Doctor en Ciencias de la Computación** con número de matrícula **231H18004**, adscrito a la **División Académica de Ciencias y Tecnologías de la Información**, de la Universidad Juárez Autónoma de Tabasco, como autor de la Tesis presentada para la obtención del Grado de Doctor en Ciencias de la Computación y titulada **Algoritmo para calcular el Índice Merrifield-Simmons sobre mallas poligonales**, dirigida por la Dra. Cristina López Ramírez y Dr. Pedro Bello López.

DECLARO QUE: La Tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la LEY FEDERAL DEL DERECHO DE AUTOR (Decreto por el que se reforman y adicionan diversas disposiciones de la Ley Federal del Derecho de Autor del 01 de Julio de 2020 regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita. Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad o contenido de la Tesis presentada de conformidad con el ordenamiento jurídico vigente.

Cunduacán, Tabasco a 2 de Marzo de 2026.

Estudiante: Herlinda González Vázquez

Cunduacán, Tabasco, a 03 de marzo de 2026
Oficio No. 0427/DACYTI/D

Asunto: Autorización de impresión de Tesis

C. Herlinda González Vázquez
Egresada del Doctorado en Ciencias de la Computación

En virtud de que cumple satisfactoriamente los requisitos establecidos en el Reglamento General de Estudios de Posgrado vigente en la Universidad, informo a Usted que se autoriza la impresión del trabajo recepcional **"Algoritmo para calcular el índice Merrifield-Simmons sobre mallas poligonales"**, para presentar examen y obtener el Grado de Doctora en Ciencias de la Computación.

Sin otro particular, aprovecho la ocasión para enviarle un afectuoso saludo.

Atentamente

Dr. Óscar Alberto González González
Director

UNIVERSIDAD JUÁREZ AUTÓNOMA DE TABASCO
DIVISIÓN ACADÉMICA DE CIENCIAS Y TECNOLOGÍAS DE LA INFORMACIÓN

C.c.p. Mtra. Yenny Lorena Dussán Rojas. – Encargada del despacho de la Coordinación de Posgrado.
Archivo.
Consecutivo.

DR.*OAGG/YLDR

Carta de Cesión de Derechos

Villahermosa, Tabasco a 2 de Marzo de 2026.

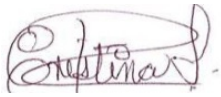
Por medio de la presente manifiesto haber colaborado como AUTOR en la producción, creación y/o realización de la obra denominada: **Algoritmo para calcular el Índice Merrifield-Simmons sobre mallas poligonales.**

Con fundamento en el artículo 83 de la Ley Federal del Derecho de Autor y toda vez que, la creación y/o realización de la obra antes mencionada se realizó bajo la comisión de la Universidad Juárez Autónoma de Tabasco; entendemos y aceptamos el alcance del artículo en mención de que tenemos el derecho al reconocimiento como autores de la obra, y a la Universidad Juárez Autónoma de Tabasco mantendrá en un 100% la titularidad de los derechos patrimoniales por un período de 20 años sobre la obra en la que colaboramos, por lo anterior, cedemos el derecho patrimonial exclusivo en favor de la Universidad.

COLABORADOR

Estudiante: Herlinda González Vázquez

TESTIGOS



Dra. Cristina López Ramírez

Dr. Pedro Bello López

Índice general

Índice de figuras

Índice de tablas

Resumen

Contar conjuntos independientes en grafos es un problema computacional dentro de la teoría de grafos: incluso grafos de grado máximo tres, se clasifican como #P-completo, por lo que los enfoques clásicos suelen crecer de forma exponencial. En este contexto se propone un algoritmo orientado a calcular el índice de Merrifield-Simmons (MS) en mallas poligonales, estructuras que modelan grafos moleculares asociados al grafeno y a compuestos aromáticos bencenoides. La idea central del método es construir una trayectoria hamiltoniana que visite cada vértice de una cuadrícula hexagonal isomórfica exactamente una vez. El cómputo de los conjuntos independientes se realiza aplicando tres reglas: (1) una recurrencia tipo Fibonacci, cuando la arista visitada pertenece al árbol de búsqueda; (2) una regla de sustracción, al detectar bordes o aristas frontales o posteriores; y (3) una regla de camino de retroceso, que aprovecha las aristas implicadas en retrocesos para recortar de manera importante el número de pasos. La metodología completa incluye generar listas de adyacencias, construir la matriz de adyacencias y ejecutar DFS para localizar ciclos y bordes de retroceso. Aunque el problema sigue siendo exponencial, el algoritmo reduce drásticamente las operaciones del conteo del MS en mallas poligonales, por lo que la estimación del orden de complejidad en el peor caso se expresa como $O(2^{min(r,c)+2}$ x *Polinomial(n,m)*), mejorando frente a técnicas clásicas de transferencia de matrices.

Palabras clave: Índice de Merrifield-Simmons, conjuntos independientes, trayectoria hamiltoniana, grafos hexagonales, orden de complejidad.

Abstract

Counting independent sets in graphs is a computational problem within graph theory: even graphs with a maximum degree of three are classified as #P-complete, so classical approaches tend to grow exponentially. In this context, an algorithm is proposed to calculate the Merrifield-Simmons (MS) index in polygonal meshes, structures that model molecular graphs associated with graphene and benzenoid aromatic compounds. The central idea of the method is to construct a Hamiltonian trajectory that visits each vertex of an isomorphic hexagonal grid exactly once. The computation of independent sets is performed by applying three rules: (1) a Fibonacci recurrence, when the visited edge belongs to the search tree; (2) a subtraction rule, when front or rear edges are detected; and (3) a backtracking path rule, that takes advantage of the edges involved in backtracking to significantly reduce the number of steps. The complete methodology includes generating adjacency lists, constructing the adjacency matrix, and executing DFS to locate cycles and backtracking edges. Although the problem remains exponential, the algorithm drastically reduces MS counting operations in polygonal meshes, so the worst-case complexity order estimate is expressed as $O(2 \wedge min_{r,c} + 2 \times Polinomial(n,m))$, an improvement over classical matrix transfer techniques.

Keywords: Merrifield-Simmons index, independent sets, Hamiltonian path, hexagonal graphs, order of complexity.

Capítulo 1

Protocolo de tesis

1.1. Introducción

El índice de Merrifield-Simmons, introducido por Merrifield-Simmons en 1989, es un índice topológico utilizado en química matemática, según (R. E. Merrifield y Simmons, 1989a) es el número de conjuntos independientes de un grafo G . Éste índice en teoría de grafos es denominado número Fibonacci de un grafo *i(G)* (Hosoya, 1973b). Este índice ha demostrado tener propiedades matemáticas aplicables en diversas cuestiones de química molecular. El índice MS y el índice de Hosoya son algunos de los índices topológicos más populares en química matemática (De Ita et al., 2023b). El índice de Merrifield-Simmons *i(G)* de un grafo G se define como el número de subconjuntos del conjunto de vértices, en el que dos vértices cualesquiera no son adyacentes, es decir, el número de vértices independientes del conjunto de G. Por otra parte, el índice de Hosoya *z(G)* de un grafo G (Hosoya, 1971) , denotado por *z(G)*, fue introducido por Hosoya en 1971; y se define como el total de número de subconjuntos de aristas, es decir, el número total de subconjuntos de clique del grafo tal que los vértices son vecinos 2 a 2 (Dai y Zhang, 2012).

1.2. Marco teórico

1.2.1. Grafos

El origen de la teoría de grafos surge en el siglo XVII con el problema de los puentes de Königsberg, de tal manera que se recorrieran todos los puentes pasando una sola vez por cada uno de ellos (Barrero et al., 2010). Los grafos representan una forma simple de plantear y modelar diversos problemas que pueden resolverse computacionalmente, de esta forma resolvemos el problema del cálculo de conjuntos independientes.

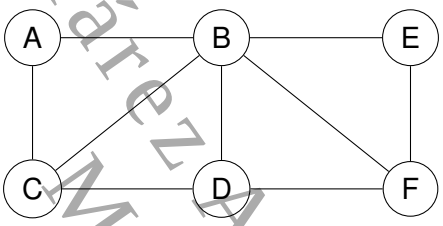


Figura 1.1. Grafo simple

Un grafo está compuesto por dos conjuntos finitos: un conjunto de $|V|$ vértices y un conjunto de $|E|$ aristas. Un grafo, es un conjunto de puntos (vértices o nodos), unidos por líneas (aristas). Dos vértices son adyacentes si están unidos por una misma arista, en la Figura 1.1 se muestra un ejemplo de un grafo con 6 nodos $\{A, B, C, D, E, F\}$ y 9 aristas (líneas) $\{AB, AC, BC, BD, BE, BF, CD, DF, EF\}$.

1.2.2. Conjuntos independientes

En teoría de grafos, un conjunto independiente es un conjunto de nodos del grafo, tal que ninguno de sus nodos es adyacente a otro (Bracho y Sánchez, 2000a). Los conjuntos independientes pueden ser de diferente tamaño iniciando con el conjunto vacío, después de tamaño uno, dos, tres, etc. (López-Ramírez et al., 2020).

En la Tabla 1.1, se muestran los conjuntos independientes que se generan del grafo de

la Figura 1.1 . El total de conjuntos independientes es de 14, siendo el conjunto formado por los nodos A, D, E el conjunto más grande que se puede obtener ya que los nodos A, D, E no se tocan entre ellos. La Figura 1.2 muestra un conjunto independiente A, D, E.

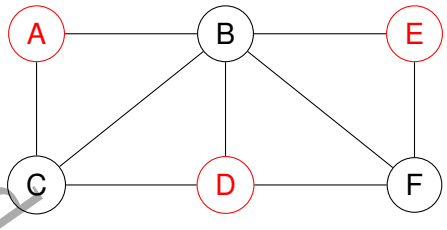


Figura 1.2. Conjunto independiente

Los conjuntos independientes son una parte fundamental de la teoría de grafos y la combinatoria matemática. Los desafíos de contar no solo capturan la atención desde una perspectiva matemática, sino que también tienen numerosas aplicaciones prácticas. En particular, contar el número de conjuntos independientes en un grafo G, designado como $I(G)$, es esencial para establecer la distinción entre algoritmos de conteo eficientes y aquellos que son intratables. En la actualidad, son escasos los problemas de conteo en grafos que admiten soluciones en tiempo polinómico. Se considera que la tarea de contar conjuntos independientes en un grafo es un problema de difícil resolución.

Tabla 1.1. Conjuntos independientes

Tamaño	Conjunto Independiente	Total
0	$\{\}$	1
1	$\{A\},\{B\},\{C\},\{D\},\{E\},\{F\}$	6
2	$\{A,D\},\{A,E\},\{A,F\},\{C,E\},\{C,F\},\{D,E\}$	6
3	$\{A,D,E\}$	1

1.2.3. Matriz de adyacencia

Una matriz de adyacencia es una representación de un grafo mediante una matriz bidimensional. Esta matriz es utilizada especialmente en grafos no dirigidos. La idea básica es asignar a cada fila y columna de la matriz un vértice del grafo y usar las entradas de la matriz para representar si hay o no una arista entre los vértices correspondientes. Si tienes un grafo con n vértices, la matriz de adyacencia será una matriz cuadrada de tamaño *n x n*. Cada fila y columna en la matriz representa un vértice, y la entrada en la posición *(i,j)* indica si hay una arista entre el vértice *i* y el vértice *j*. Si hay una arista entre los vértices *i* y *j*, la entrada *(i,j)* y *(j,i)* son marcadas como 1. Si no hay una arista entre *i* y *j*, las entradas *(i,j)* y *(j,i)* son marcadas como 0 (Perdomo Flández et al., 2015).

1.2.4. Complejidad computacional

Es muy importante estudiar la calidad de los algoritmos, ya no en busca de la solución a un problema, sino más bien de la mejor solución posible. La calidad de un algoritmo puede definirse según diferentes criterios; como el tiempo y el tamaño o cualquier otra métrica que implique economizar recursos. Nos interesa diseñar algoritmos que encuentren la solución en el menor y mejor tiempo posible (Mañas, 1997).

La complejidad temporal de un algoritmo es una medida que describe cómo aumenta el tiempo de ejecución de un algoritmo a medida que aumenta el tamaño de los datos de entrada. Se expresa comúnmente usando la notación Big O, que nos permite analizar el peor caso de un algoritmo en términos de su eficiencia. Se enfoca en el crecimiento del tiempo de ejecución en relación con el tamaño de la entrada (n), ignorando constantes y términos de menor orden. Esto se debe a que, para entradas suficientemente grandes, los términos de mayor orden dominan el crecimiento.

1.2.5. Ordenes de complejidad algoritmica

Desde una perspectiva matemática, cuando el número *N* tiende a infinito, su comportamiento se describe como asintótico. A un conjunto de funciones que exhiben un comportamiento asintótico similar se le conoce como un orden de complejidad, comúnmente representado por la notación O. Indican que el tiempo de ejecución del algoritmo aumenta de forma polinomial a medida que aumenta el tamaño del problema.

- Cota superior **O**. Se adopta una notación especial llamada O-grande (big-Oh), por ejemplo *O(f(n))* para indicar que la cota superior del algoritmo es *f(n)*. La cota superior de un algoritmo, indica una cota o la máxima razón de crecimiento que un algoritmo puede tener.

- Cota inferior ■ . Existe una notación similar para indicar la mínima cantidad de recursos que un algoritmo necesita para alguna clase de entrada. La cota inferior de un algoritmo, denotada por el símbolo ■, pronunciado “Gran Omega” u “Omega”.

- Notación Θ. Cuando las cotas superior e inferior son la misma, indicamos esto usando la notación Θ (big-Theta).

1.2.6. SAT Satisfacibilidad booleana

Stephen Arthur Cook (Cook, 1971) presentó el problema que ha impulsado diversas investigaciones en el campo de la complejidad computacional. El desarrollo de técnicas efectivas para resolver problemas computacionales que ha sido un desafío duradero para los investigadores, y entre los problemas más desafiantes desde el punto de vista computacional se destaca la satisfactibilidad de restricciones.

Dentro del ámbito de los problemas de satisfactibilidad *SAT*, involucra determinar si una fórmula booleana es capaz de satisfacerse o no (si al menos existe una asignación que la haga verdadera), estas asignaciones se denominan modelos de las fórmulas.

A partir del desafío de decisión *SAT* se origina el problema relacionado con el conteo: este problema implica calcular la cantidad de modelos de una fórmula booleana, el cual es comúnmente conocido como *#SAT*. Si se desea calcular el número de conjuntos independientes de un grafo cadena sería equivalente a calcular el número de modelos del grafo (Perez Barrios et al., 2015).

1.2.7. P, NP, NP Completo

El origen de los problemas *P* vs *NP* se encuentran en la lógica en general, y más exactamente, el conjunto de problemas relativos a la computación y la teoría de la información. El tema fue formulado por tres investigadores: (Cook, 1971;Karp, 1972 y Levin, 1973) . Es tal la importancia del problema que ha sido identificado como uno de los problemas del milenio *(The milleniun problems) por el Instituto Clay* (Jaffe, 2006). La motivación aquí es la teoría de la información y más la teoría de la computación. Y lo que mas interesa son temas y problemas de análisis combinatorios. Por lo que el Instituto Clay incluyó a este problema entre los más importantes de nuestra época *P* vs *NP* son la columna vertebral para el estudio en la Teoría de la Complejidad.

- **NP**. Conjunto de problemas en los que podemos comprobar en un tiempo razonable si una respuesta al problema es correcta o no. Problemas que se comprueban que algo, es solución en tiempo polinomial.

- **P**. Conjunto de problemas en los que podemos encontrar una respuesta al problema en un tiempo razonable. Problemas qué se resuelven con algoritmo polinomial.

- **NP Completo**. Reducir cualquier NP. Demostrar que está en NP y luego transformar a este, en tiempo polinómico, en un problema que ya esté en NP-completo. Algunos NP Completos:

    - Problema de satisfacibilidad booleana *(SAT)*

- Problema de la mochila *(knapsack)*
- Problema del ciclo hamiltoniano
- Problema del vendedor viajero
- Problema de la clique ...

1.2.8. Aplicaciones de los conjuntos independientes

Los conjuntos independientes se estudian principalmente en el campo de la informática porque los grafos son una potente herramienta para modelar problemas de la vida real.

La asignación eficiente de aulas y recursos en entornos universitarios es un desafío que implica coordinar horarios, cantidad de alumnos, profesores y requisitos específicos para cada clase, garantizando que se cumplan criterios de calidad en la asignación (Bracho et al., 2003a) .

Esta necesidad de organización y asignación efectiva de recursos se extiende a diversas áreas, como la organización de equipos de trabajo, la conectividad en redes de computadoras, la planificación de actividades; por mencionar algunos.

En Física (Bonilla García, 2019a) estudia la medición de la entropía del gas, donde dos partículas no pueden estar en la misma arista; y en Química los conjuntos independientes pueden ser utilizados para modelar ciertos aspectos de la estructura del sistema bencenoide (Gutman, 1982a).

Los conjuntos independientes tienen aplicaciones potenciales en diversas áreas como las industrias, desde las telecomunicaciones y la logística hasta las finanzas y la planificación estratégica (Wurtz et al., 2022a), además de mejorar la colaboración y la gestión de datos. En la Computación Cuántica Adiabática, resolver el problema de los conjun-

tos independientes implica encontrar todos los estados básicos computacionales de un Hamiltoniano de muchos cuerpos $HG(V,E)$ (Yin et al., 2023a).

Además, los conjuntos independientes mejoran la asignación de recursos y la distribución de tareas en las redes informáticas. Un algoritmo exacto, como branch-and-bound, puede resolver grandes redes con precisión, como señala (Lamm et al., 2017a). Por lo que son elementos esenciales de la teoría de grafos y combinatoria matemática, que ofrecen soluciones desde la educación hasta la ciencia y la nutrición, demostrando su importancia en diferentes contextos.

1.2.9. Estrategias para calcular el Índice Merrifield-Simmons

1.2.9.1. Busqueda en profundidad

DFS, o algoritmo de búsqueda en profundidad, es reconocido por proporcionar los pasos para explorar todos los nodos de un grafo sin repetir ninguno. DFS (G,v) recorrido en profundidad con origen v. La idea es:

- Se marca el nodo v.
- Si todos los nodos adyacentes a v están marcados, entonces TERMINAR, sino se elige un nodo w, adyacente a v que no está marcado.
- Se ejecuta el proceso DFS(G,w). La Figura 1.3 muestra cual sería el orden de visita al aplicar el algoritmo DFS(G,v)

1.2.9.2. Recorrido hamiltoniano

Como paso inicial del método para calcular el número de conjuntos independientes de un grafo, se lleva a cabo la construcción de un camino Hamiltoniano (Hc) sobre el grafo de entrada. Este camino visita cada vértice del grafo exactamente una vez. En el ámbito

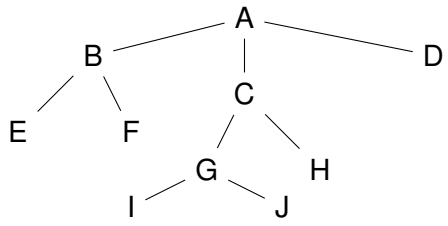


Figura 1.3. DFS (A, B, E, F, C, G, I, J, H, D)

matemático de la teoría de grafos, un camino hamiltoniano se define como una secuencia de aristas consecutivas en un grafo, donde se visitan todos los vértices exactamente una vez. En caso de que el último vértice visitado sea adyacente al primero, se denomina ciclo hamiltoniano.

1.2.10. Conteo de conjuntos independientes en estructuras básicas

1.2.10.1. Grafo en cadena lineal

Sea *G=(V,E)* con $\Delta(G) = 2$ siendo *G* un grafo cadena con *n* nodos, por tanto $|V| = n$ y $|E| = m = n-1$. Ordenemos los nodos en G como *V= {v_1,v_2,...,v_n}* de tal forma que se cumpla *E={ {c_1,...,c_{n-1}}= { {v_1,v_2},{v_2,v_3},...,{v_{n-2},v_{n-1}}, {v_{n-1},v_n} }* es decir $|v(c_i) \bigcap v(c_{i+1})|$=1, i=1,...n-2. G_i, *i={ 1, . . . n }* siendo G_i el subgrafo inducido de *G*. La técnica básica para contar el número de conjuntos independientes $I(G_i)$ sobre G_i de *G*, es una estrategia incremental (Morales Alcántara et al., 2019) que se basa en asociar un par de valores (α_i, β_i) a cada nodo $v_i \in V, i = \{1, \ldots n\}$.

- El recorrido inicia en uno de los extremos (del nodo v_1 al nodo v_n) visitando a su nodo adyacente en forma lineal.
- El primer par de valores inicia con (α_i, β_i) = (1, 1). La relación de recurrencia considera que se conoce el valor (α_i, β_i) asociado al subgrafo inducido G_i de *G* tal que *I(G_i)=(α_i, β_i)*.

- El nuevo par de valores (α_{i+1}, β_{i+1}) se construye en base a (α_i, β_i) al aplicar la recurrencia que genera a los números de Fibonacci α_{i+1}=α_i +β_i, β_{i+1}=α_i.

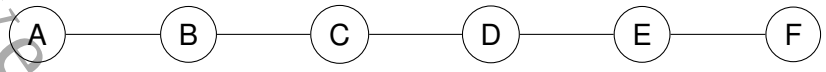


Figura 1.4. Cadena lineal

En la Tabla 1.2 se muestra el conteo de los conjuntos independientes, aplicando la secuencia Fibonacci en el grafo de cadena de la Figura 1.4. Por lo tanto : $(\alpha_i, \beta_i) = (F_{i+1}, F_i)$ entonces $I(G_i) = \alpha_i + \beta_i = F_{i+1} + F_i = F_{i+2}, i = \{0, ..., n\}$ es decir para $n = 6$, $I(G) = I(G_6) = F_7 + F_6 = F_8 = 21$.

Tabla 1.2. Conjuntos independientes del grafo-cadena

A	B	C	D	E	F
(α_1, β_1) $\rightarrow$	(α_2, β_2) $\rightarrow$	(α_3, β_3) $\rightarrow$	(α_4, β_4) $\rightarrow$	(α_5, β_5) $\rightarrow$	(α_6, β_6)
(1,1) $\rightarrow$	(2,1)$\rightarrow$	(3,2) $\rightarrow$	(5,3) $\rightarrow$	(8,5) $\rightarrow$	(13,8)
$(F_2, F_1)\rightarrow$	$(F_3, F_2)\rightarrow$	$(F_4, F3)\rightarrow$	$(F_5, F4) \rightarrow$	$(F6, F_5)\rightarrow$	(F_7, F_6)

1.2.10.2. Ciclo simple de un grafo

La Figura 1.5 muestra un grafo que contiene un ciclo simple. Contando conjuntos independientes se inicia un hilo principal (Lp); el par de valores comienza con (α_i, β_i) = (1,1), y el hilo secundario (Ls) inicia la recurrencia de la secuencia con (α_i, β_i) = (0, 1), donde i= 1,. . . , m. El último par (α_m, β_m) = (0, β_m) de modo que la serie (α_1, β_1=(0,1) $\rightarrow$ (α_2, β_2) = (1,0) $\rightarrow$ (α_3, β_3)= (1,1). . . $\rightarrow$ ((α_m, β_m)) $\rightarrow$ (0, β_m) y donde el valor β_m nos da el valor de : $|S\varepsilon I(G') : v_1 \varepsilon S \wedge v_m \varepsilon S|$ La serie anterior se corresponde a la siguiente serie considerando los números de Fibonacci: $(F_0, F_1) \rightarrow (F_1, F_0) \rightarrow (F_2, F_1) \rightarrow ... \rightarrow (F_{m-1}, F_{m-2})$.

$(\alpha_m, \beta_m) = (0, \beta_m)$; $i(G_c) = ((\alpha_m, \beta_m) - (0, \beta_m))$.

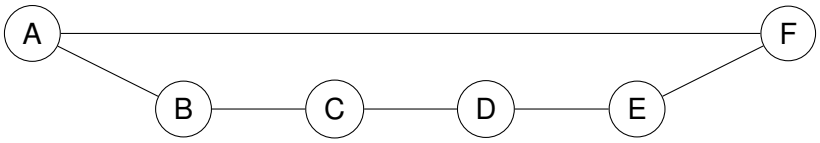


Figura 1.5. Ciclo simple

Tabla 1.3. Conjuntos independientes del grafo ciclo

Hilo	A	B	C	D	E	F
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)
Lp	(1,1) $\rightarrow$	(2,1)$\rightarrow$	(3,2) $\rightarrow$	(5,3) $\rightarrow$	(8,5) $\rightarrow$	(13,8)
Ls	(0,1) $\rightarrow$	(1,0)$\rightarrow$	(1,1) $\rightarrow$	(2,1) $\rightarrow$	(3,2) $\rightarrow$	(5,3)
F0=0, F1=1	(F2, F1)$\rightarrow$	(F3, F2)$\rightarrow$	(F4, F3)$\rightarrow$	(F5, F4) $\rightarrow$	(F6, F5)$\rightarrow$	(F7, F6)

De acuerdo con De Ita et al., 2020a para calcular el cierre del ciclo aplicamos la regla (2) de sustracción ; el último par (α_m, β_m) del hilo secundario (L_s) es $(\alpha_m, \beta_m) = (0, \beta_m)$, donde $v_m, v_0 \in E(G)$ representa la arista de retroceso que cierra el ciclo de G y se resta del último par del hilo primario (Lp). Por tanto, $(\alpha_m, \beta_m) - (0, \beta_m) = ((\alpha_6, \beta_6) - (0, \beta_6)) = (13,8) - (0,3) = (13,5)$, obtenemos que $i(G_c) = 13+5$, es decir, 18 es el número total de conjuntos independientes. Este cómputo se muestra en detalle en la Tabla 1.3, donde podemos ver el conteo de conjuntos independientes de G.

1.2.10.3. Ciclo y cadena lineal de un grafo

En la Figura 1.6 se presenta un grafo con 9 nodos, donde los nodos (A,B,C,D,E,F,A) forman un ciclo, para calcular los conjuntos independientes en un ciclo, se realizan los mismos pasos que en un grafo de ciclo simple, el cálculo se muestra en detalle en la Tabla 1.4. El nodo F cierra el ciclo, obteniendo como resultado el par $(13,5)$ y continuando con el cálculo de la secuencia Fibonacci hasta terminar el recorrido del grafo. En total los conjuntos independientes $I(G)$ es igual a la suma del último par $(49,31) = 49+31 = 80$ conjuntos independientes.

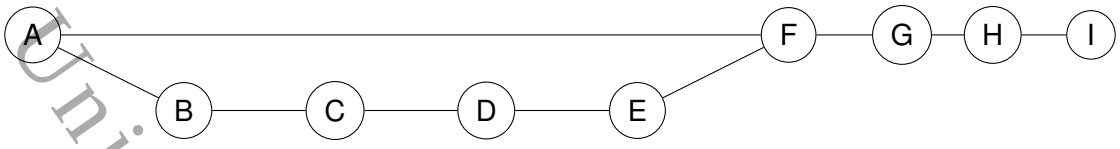


Figura 1.6. Ciclo simple y cadena lineal

Tabla 1.4. Conjuntos independientes del ciclo y cadena

Hilo	A	B	C	D	E	F	G	H	I
	(α_1,β_1)	(α_2,β_2)	(α_3,β_3)	(α_4,β_4)	(α_5,β_5)	(α_6,β_6)	(α_7,β_7)	(α_8,β_8)	(α_9,β_9)
Lp	(1,1)→	(2,1)→	(3,2)→	(5,3)→	(8,5)→	(13,8)→			
Ls	(0,1)→	(1,0)→	(1,1)→	(2,1)→	(3,2)→	(5,3)→			
						(13,5)→	(18,13)→	(31,18)→	(49,31)

1.2.10.4. Hexágonos unidos por aristas

La Figura 1.7 muestra el caso de un grafo con dos ciclos conectados por una arista puente. Aquí se realiza el mismo proceso para los ciclos simples y, una vez encontrada una arista de retroceso, se cierra el primer ciclo, se aplica la ecuación (2) y se continúa calculando los conjuntos independientes según la regla de Fibonacci.

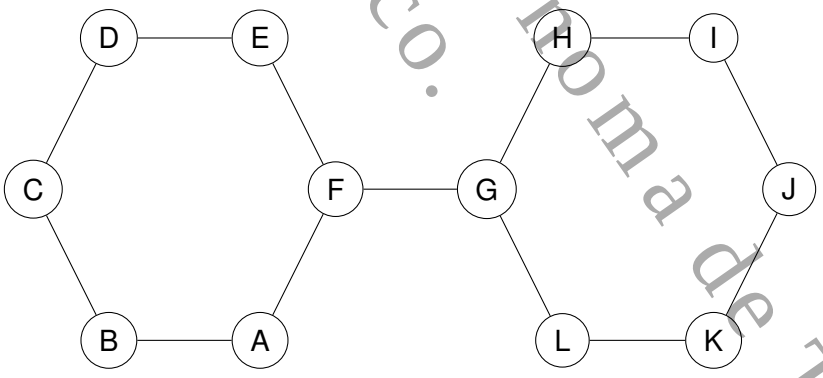


Figura 1.7. Dos hexágonos unidos por una arista

Las tablas 1.5 y 1.6 muestran los cálculos del total de los conjuntos independientes $i(G)$ que es igual a la suma del último par $(209,90) = 209+90 = 299$ conjuntos independientes.

Tabla 1.5. Ciclo 1

Hilo	A	B	C	D	E	F
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)
Lp	(1,1) $\rightarrow$	(2,1)$\rightarrow$	(3,2) $\rightarrow$	(5,3) $\rightarrow$	(8,5) $\rightarrow$	(13,8)
Ls	(0,1) $\rightarrow$	(1,0)$\rightarrow$	(1,1) $\rightarrow$	(2,1) $\rightarrow$	(3,2) $\rightarrow$	(5,3)

Tabla 1.6. Ciclo 2

Hilo	G	H	I	J	K	L
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)
Lp	(18,13) $\rightarrow$	(31,18)$\rightarrow$	(49,31) $\rightarrow$	(80,49) $\rightarrow$	(129,80) $\rightarrow$	(209,129))
Ls	(0,13) $\rightarrow$	(13,0)$\rightarrow$	(13,13) $\rightarrow$	(26,13) $\rightarrow$	(39,26) $\rightarrow$	(65,39)
						(209,129) -(0,39)
						(209,90)

1.2.10.5. Hexágonos unidos por vértices

Se han desarrollado algoritmos para el recuento de conjuntos independientes en estructuras reticulares planas que facilitan el recuento, aplicando el recorrido por filas y columnas o viceversa (De Ita Luna et al., 2023a). Nuestra propuesta algorítmica procesa las aristas *frond* en dos fases. Sea v,w una arista *frond* de un grafo G. En la primera fase, cuando un hilo principal de paseo (L_p) visita el primer vértice v de la arista *frond*, el número de hilos activos y de hilos de cálculo (L_s) debe duplicarse. Supongamos que $(\alpha_v, \beta_v)i$ es la carga asociada al hilo activo Li cuando visita el vértice v. Se crea un nuevo hilo L_{vwi} , subordinado al hilo maestro L_i, y con un par inicial asociado $(0, \beta_v)_{vw}$. Por lo tanto, la etiqueta vwi de L_{vwi} es también un puntero a su hilo maestro L_i . La segunda fase en el procesamiento del borde *frond* (v,w) es cuando el paseo visita el vértice w, y v ya ha sido etiquetado como vértice visitado. Así, el control se mantiene en el vértice w. Supongamos que $(\alpha_w, \beta_w)i$ es la carga del vértice w en el hilo maestro L_i . Mientras tanto, $(\alpha_{vw}, \beta_{vw})_{vw}$ es la carga de w en el hilo subordinado L_{vwi} . A continuación, la regla de sus-

tracción de la ecuación (2) actualiza la carga de w en L_i. Tras la aplicación de esta regla, el hilo subordinado L_{vwi} queda cerrado. De este modo, cualquier hilo en el que aparezca la partícula vw debe cerrarse, lo que disminuye el número de hilos activos.

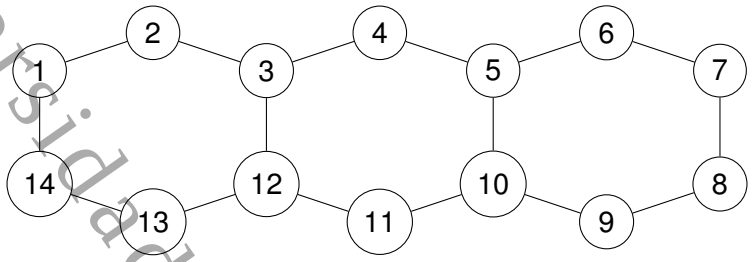


Figura 1.8. Tres hexagonos conectados por vertices

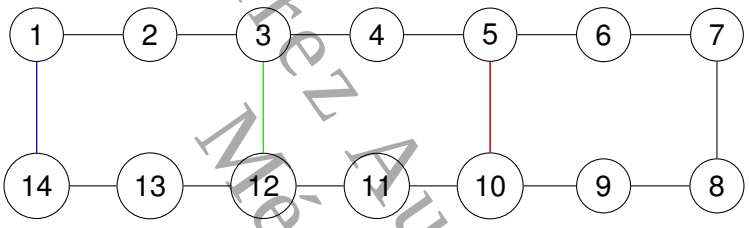


Figura 1.9. Cuadrícula de Tres hexágonos unidos por vértices

Calcular el número de conjuntos independientes de una malla es posible tomando como guía la trayectoria hamiltoniana H_c, aunque la complejidad temporal es de orden exponencial sobre el número máximo de aristas de *frond* en cualquier fila de la malla, dado el número de ciclos que se mantienen abiertos durante la trayectoria H_c. Las reglas de Fibonacci y de sustracción son suficientes para procesar una fila de mosaicos de una cuadrícula hexagonal, como se muestra en las Figuras 1.8, 1.9 y en la Tabla 1.7.

1.3. Justificación

El índice de Merrifield-Simmons que en la teoría de grafos es conocido como el número de Fibonacci de un grafo $i(G)$, es un índice topológico utilizado en química matemática, el cual facilita la visualización y comprensión del problema de los conjuntos independientes

Tabla 1.7. Conteo de conjuntos independientes de la Figura 1.9

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Lp (1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	(21,13)	(34,21)	(55,34)	(89,55)	-(0,15)=	(89,40)		
Ls⊢(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	(21,13)	(34,21)	-(0,6)=	(34,15)		
	c2Lp	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	(16,10)	(26,16)	-(0,6)=	(26,10)		
	c2Ls	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	-(0,3)=	(13,5)		
			c3Lp	(0,5)	(5,0)	(5,5)	(10,5)	(15,10)	(25,15)	x			
			c3Ls	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	x			
			c3c2Lp	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	x			
			c3c2Lp	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	x			
										(129,89)	(218,129)	(311,218)	(529,311)
											-(0,36)		-(0,114)
											=(218,93)		= (529,197) ⊣
										(49,34)	(83,49)	(114,83)	(197,114)
											-(0,18)		x
											=(83,31)		
										(36,10)	(46,36)	x	
										(18,13)	=(31,18)	x	

(CI). Contar los conjuntos independientes, con la estrategia de cálculo basada en la serie de Fibonacci simplifica su implementación al explorar grafos.

Éste índice abarca un amplio campo de investigación con numerosas aplicaciones prácticas en diversos ámbitos, como en el área médica, nutrición, logística empresarial, planeación de tareas y horarios, organización de actividades académicas en las universidades, entre otros. Los conjuntos independientes nos permiten crear combinaciones de diferentes tamaños y disponer de múltiples opciones para elegir, lo cual resulta de gran utilidad.

Los algoritmos exactos son fundamentales para el conteo preciso de conjuntos independientes $i(G)$, un problema #P-completo para grafos con grado máximo $\Delta(G) \geq 3$, ya que garantizan resultados absolutos sin errores, esenciales en aplicaciones donde las aproximaciones fallan (ej. grados ≥ 6), en donde la precisión es no negociable como en aplicaciones científicas, como en la química matemática (De Ita Luna et al., 2026).

1.4. Preguntas de investigación

1.4.1. General

- ¿Cómo diseñar un algoritmo eficiente para contar el índice de Merrifield-Simmons (MS) en una cuadrícula hexagonal isomórfica en una estructura de malla poligonal?. . . .

1.4.2. Específicas

- ¿Cómo diseñar un algoritmo eficiente para contar los conjuntos independientes en una estructura basada en grafos?
- ¿Cuál es la metodología más eficiente para contar los conjuntos independientes?

- ¿Cómo calcular el tiempo de ejecución y el orden de complejidad del algoritmo?

1.5. Hipótesis

Al Diseñar un algoritmo exacto para contar el índice Merrifield-Simmons (MS) en una cuadrícula hexagonal isomórfica de malla poligonal, modelada como una estructura de grafos,se establecerá el análisis de la complejidad temporal de los algoritmos propuestos, en contraste con un problema que originalmente tenía una complejidad exponencial en tiempo.

1.6. Objetivos

1.6.1. Objetivo general

Diseñar un algoritmo para contar el índice Merrifield-Simmons (MS) en cuadrícula hexagonal de una estructura de malla poligonal modelada con grafos.

1.6.2. Objetivos específicos

- Diseñar un algoritmo exacto para contar los conjuntos independientes en grafos de malla hexagonal.
- Realizar pruebas exhaustivas en estructuras poligonales para el conteo de conjuntos independientes.

1.7. Metodología

1.7.1. Cálculo de fibonacci en grafos

Para calcular el índice Merrifield-Simmons, se utiliza un método que involucra un recorrido hamiltoniano en una cuadrícula hexagonal isomórfica, aplicada a sistemas bencenoides (ver Figura 1.10). La técnica a utilizar es la serie de números Fibonacci 0,1,1,2,3,5,8... que nos permite contar los conjuntos independientes para ciertas estructuras en grafos.

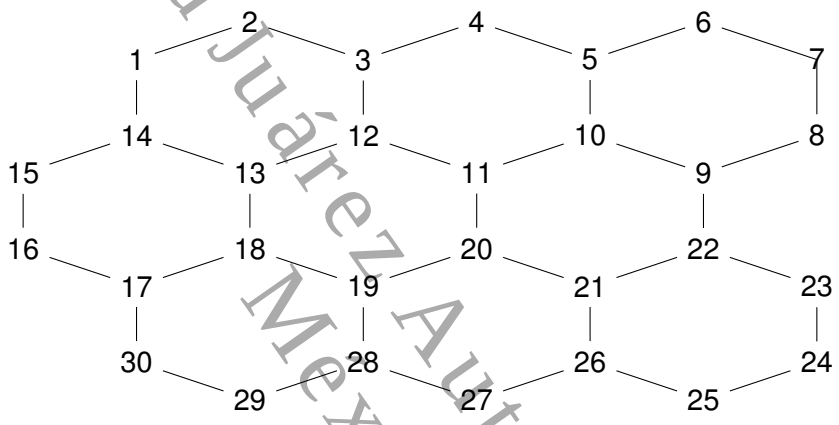


Figura 1.10. Sistema bencenoide $H_{r,t}$ en malla hexagonal

Contar el número de conjuntos independientes $i(G)$ en un grafo G implica recorrer el camino hamiltoniano Hc de G según (El-Basil, 1988a). Durante este recorrido, el primer par de valores comienza con (α_i, β_i) = $(1,1)$. Para las siguientes aristas identificadas, se aplica una de dos reglas. La regla de recurrencia de Fibonacci (1.1) cuando la arista visitada es una arista de árbol o una regla de sustracción (1.2) cuando se reconoce una arista *frond* (o arista de retroceso si nos referimos a búsquedas en profundidad). Si consideramos que v_i pertenece al camino P_n, y v_{i+1} es el siguiente vértice que visitará una arista del árbol (v_i, v_{i+1}), entonces se aplica la siguiente ecuación de recurrencia para calcular la carga $(\alpha_{v_i+1}, \beta_{v_i+1})$ en función de la carga $(\alpha_{v_i}, \beta_{v_i})$:

$$(\alpha_{v_{i+1}}, \beta_{v_{i+1}}): \quad \alpha_{v_{i+1}} = \alpha_{v_i} + \beta_{v_i}; \qquad \beta_{v_{i+1}} = \alpha_{v_i} \tag{1.1}$$

Llamamos al anterior par de recurrencias, recurrencia de la regla de Fibonacci, porque cuando se aplican sobre un camino P_n obtenemos la identidad de conocimiento $i(P_n) = \alpha_{v_i+1} + \beta_{v_i+1} = F_n + F_{n+1} = F_{n+2}$, donde F_n es el enésimo número de Fibonacci.

$$(\alpha_w, \beta_w)_i = (\alpha_w, \beta_w) - (0, \beta_{vw}) = \qquad (\alpha_w, \beta_w - \beta_{v,w}) \tag{1.2}$$

Ilustremos nuestra propuesta con un ejemplo; en la Figura 1.11, mostramos cómo calcular el $M-S$ del grafo. El camino hamiltoniano utilizado para recorrer G es $A-B-C-D-E$. Se abre un hilo primario L_p y se aplica la función de recurrencia (1) al inicio del cálculo de $I(G_5)$: L_p : (1,1) $\rightarrow$ (2,1) $\rightarrow$ (3,2) $\rightarrow$ (5,3) $\rightarrow$ (8,5). Obsérvese que las cargas temporales $(\alpha_{v_i}, \beta_{v_i})$ pueden almacenarse en el vértice v_i y marcarse como visitadas. Para cualquier par de vértices $(v,w) \in S$, donde S es un conjunto independiente v y w son adyacentes, y si la arista (v,w) se identifica como arista *frond*, en este caso (E,B) es la arista *frond*, se abre un hilo secundario L_s paralelo a L_p; es decir, L_s : L_s: (0,1) $\rightarrow$ (1,0) $\rightarrow$ (1,1) $\rightarrow$ (2,1) cuando la caminata visita el vértice v y w ya ha sido visitado, entonces se aplica la regla de resta (2), que permite calcular la carga del vértice v a partir de una carga del vértice u; y para cada arista *frond* que se encuentre. El apartado 3.2.5 lo explica con detalle. Por lo tanto, *I(G)= {∅}, {A}, {B}, {C}, {D}, {E}, {A,C}, {B,D}, {C,E}, {E,A}, {D,A}, {A,C,E}} = 12. S* $\subseteq$ *V(G)* es un conjunto independiente. Si $\Delta x, y \in S, \{x,y\} \notin E$. $I(G) = S/S$. En la Tabla 1.8 se puede ver el cálculo de los conjuntos independientes respecto a la Figura 1.11.

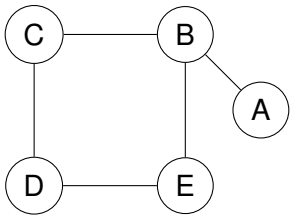


Figura 1.11. Grafo G con 5 vértices

Tabla 1.8. Conteo de Conjuntos independientes por la secuencia fibonacci

Hilo	A	B	C	D	E	i(G)	
	(α_1, β_1)	(α_2, β_2)	(α_3, β_3)	(α_4, β_4)	(α_5, β_5)		
Lp	(1,1) $\rightarrow$	(2,1)$\rightarrow$	(3,2) $\rightarrow$	(5,3) $\rightarrow$	(8,5)	(8,5)	
Ls		(0,1) $\rightarrow$	(1,0)$\rightarrow$	(1,1) $\rightarrow$	(2,1)	-(0,1)	(8,4)=8+4=12

Siguiendo con nuestra sistemática, el árbol binario de la Figura 1.12 presenta todas las combinaciones posibles formadas con los cinco vértices del grafo G, considerando que ninguno de sus vértices es adyacente a otro (Bracho y Sánchez, 2000a). El símbolo negativo indica que el vértice no puede formar parte de la combinación. Al final de la construcción del árbol binario, podemos ver que hay 12 conjuntos independientes (y eliminamos los vértices B y E (vértices en rojo), que forman la arista *frond*. Además, aplicando el recuento de la secuencia de Fibonacci de la Tabla 1.8, confirmamos que los resultados, que tienen una importancia significativa, coinciden.

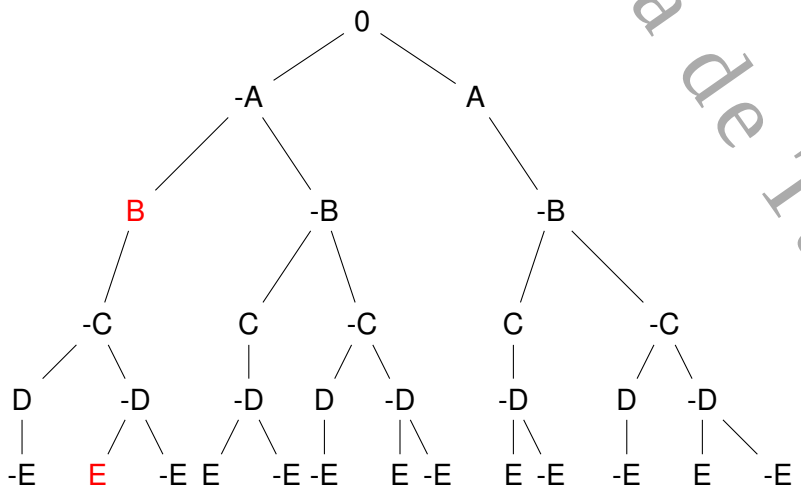


Figura 1.12. Árbol binario de conjuntos independientes

1.8. Cronograma de actividades

Tabla 1.9. Lista de Actividades

		2023		2024		2025	
No	Actividad	1er	2do	1er	2do	1er	2do
1	Recopilación y revisión de la Información	X	X	-	-	-	-
2	Estado del arte	X	X	-	-	-	-
3	Desarrollo del protocolo de tesis	X	X	-	-	-	-
4	Diseño del Algoritmo	-	X	X	-	-	-
5	Análisis de complejidad	-	X	X	-	-	-
6	Construcción del prototipo	-	-	X	X	-	-
7	Validación exhaustiva del algoritmo	-	-	-	-	X	X
8	Presentación de avances	X	X	X	X	X	X
8	Entrega de proyecto final	-	-	-	-	-	X
9	Actividad de retribución social	X	X	-	-	-	-
10	Redacción de artículos	X	-	X	-	X	-
12	Estudio de Cursos alternos	-	X	X	X	X	X
13	Presentación final de la tesis de grado	-	-	-	-	-	X

Referencias

Barrero, A. C., de García, G. W., & Parra, R. M. M. (2010). *Introducción a la Teoría de Grafos*. Elizcom sas.

Bonilla García, A. N. (2019a). ¿ Podría ser la gravedad una fuerza entrópica?

Bracho, R. L., Gutiérrez-Andrade, M. A., Ortuño-Sánchez, M. P., & Ramírez-Rodríguez, J. (2003a). El problema del conjunto independiente en la selección de horarios de cursos. *Revista de Matemática: Teoría y Aplicaciones*, *10*(1-2), 156-167.

Bracho, R. L., & Sánchez, M. P. O. (2000a). Un algoritmo paralelo para el problema del conjunto independiente. *Revista de Matemática: Teoría y Aplicaciones*, *7*(1-2), 125-134.

Cook, S. A. (1971). The complexity of theorem-proving procedures.

Dai, S., & Zhang, R. (2012). The Merrifield-Simmons Index and Hosoya Index of C (n, k, λ) Graphs. *Journal of Applied Mathematics*, *2012*, 1-8.

De Ita, G., Bello, P., & Contreras, M. (2023b). A Method for Computing the Merrifield-Simmons Index on Benzenoid Systems. *Match-communications in mathematical and in computer chemistry*, *89*(1), 245-270.

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020a). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets, 57-66.

De Ita Luna, G., Marcial Romero, R., Bello Lopez, P., & Meliza, C. G. (2026). An Exact Algorithm for Counting the Number of Independent Sets of a Graph. *Mathematics*, *14*(2), 275. https://doi.org/https://doi.org/10.3390/math14020275

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023a). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1).

El-Basil, S. (1988a). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29.

Gutman, I. (1982a). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73.

Hosoya, H. (1971). Topological index. A newly proposed quantity characterizing the topological nature of structural isomers of saturated hydrocarbons. *Bulletin of the Chemical Society of Japan*, *44*(9), 2332-2339.

Hosoya, H. (1973b). Topological index and Fibonacci numbers with relation to chemistry. *Fibonacci Quart*, *11*(3), 255-266.

Jaffe, A. M. (2006). The millennium grand challenge in mathematics. *Notices of the AMS*, *53*(6), 652-660.

Karp, R. M. (1972). Reducibility among combinatorial problems. *University of California at Berkeley*.

Lamm, S., Sanders, P., Schulz, C., Strash, D., & Werneck, R. F. (2017a). Finding near-optimal independent sets at scale. *Journal of Heuristics*. https://doi.org/https://doi.org/10.1137/1.9781611974317.12

Levin, L. A. (1973). Universal sequential search problems. *Problemy peredachi informatsii*, *9*(3), 115-116.

López-Ramírez, C., Gómez, J. E. G., & Luna, G. D. I. (2020). Building a maximal independent set for the vertex-coloring problem on planar graphs. *Elecronic Notes in Theoretical Computer Science*, *354*, 75-89.

Mañas, J. A. (1997). Análisis de algoritmos: Complejidad.

Merrifield, R. E., & Simmons, H. E. (1989a). Topological methods in chemistry.

Morales Alcántara, L. D., et al. (2019). *Sistema web para el conteo eficiente de Conjuntos Independientes en estructuras jerárquicas (Árboles)* [B.S. thesis].

Perdomo Flández, J. A., et al. (2015). *Análisis del cómputo exhaustivo de conjuntos independientes* [B.S. thesis].

Perez Barrios, O., et al. (2015). *Construcción de un algoritmo para contar modelos de fórmulas en 2-FC* [Tesis de maestría].

Wurtz, J., Lopes, P. L., Gemelke, N., Keesling, A., & Wang, S. (2022a). Industry applications of neutral-atom quantum computing solving independent set problems. *arXiv preprint arXiv:2205.08500*.

Yin, X.-F., Yao, X.-C., Wu, B., Fei, Y.-Y., Mao, Y., Zhang, R., Liu, L.-Z., Wang, Z., Li, L., Liu, N.-L., et al. (2023a). Solving independent set problems with photonic quantum circuits. *Proceedings of the National Academy of Sciences*, *120*(22), e2212323120. https://doi.org/https://doi.org/10.1073/pnas.2212323120

Capítulo 2

Ramificación y poda para el conteo del Índice Merrifield-Simmons en mallas poligonales

Ramificación y poda para el conteo del Índice Merrifield-Simmons en mallas poligonales

[1] Herlinda González Vázquez $231H18004@alumno.ujat.mx$, [1]Cristina López Ramírez $cristina.lopez@ujat.mx$, [2]Pedro Bello López $pedro.bello@correo.buap.mx$, [2]Guillermo De Ita Luna $deitaluna63@gmail.com$

Resumen: En este artículo, presentamos la estructura molecular del benceno como un anillo hexagonal regular y una malla hexagonal isomórfica, una estructura molecular introducida por Merrifield-Simmons (MS) en 1989. El índice MS, un índice topológico utilizado en química matemática, es el número de conjuntos independientes de un grafo G. Este índice desempeña un papel crucial en nuestra comprensión de las estructuras moleculares complejas. La estrategia para calcular el índice MS implica la construcción de una trayectoria hamiltoniana en el grafo de entrada. Estas estructuras se discuten en campos como la química teórica y computacional. Por lo tanto, proponemos un nuevo método de ramificación y límite para contar conjuntos independientes en mallas bencenoides poligonales, destacando su importancia en el estudio del análisis de estructuras moleculares complejas y su aplicabilidad en campos científicos interdisciplinarios.

Palabras clave: Benzenoid, Independent Sets, Hamiltonian Path.

Institución de adscripción: [1]Universidad Juárez Autónoma de Tabasco, [2]Benemérita Universidad Autónoma de Puebla.

2.1. Introducción

La forma molecular del benceno se representa como un anillo hexagonal regular, tal y como se muestra en la figura 2.1, en el que cada vértice contiene un átomo de carbono. Estos átomos de carbono están unidos entre sí por enlaces simples y dobles alternos. Como resultado, la molécula de benceno es plana y presenta simetría hexagonal (Gutman, 1982b). Una rejilla hexagonal isomórfica es una estructura de rejilla formada por hexágonos regulares conectados entre sí. En una red cuadrada o rectangular, donde los elementos de la red son cuadrados o rectángulos, en una red hexagonal isomórfica, los hexágonos son las unidades básicas de la red, formando una red poligonal, donde cada polígono comparte un borde con los polígonos adyacentes, lo que crea una estructura regular y simétrica, como se muestra en la figura 2.2. Por lo tanto, si consideramos la estructura molecular como un esqueleto, podemos representarla como un grafo G, donde los átomos de carbono representan los vértices y las adyacencias atómicas son los bordes del grafo o aristas (R. E. Merrifield y Simmons, 1980).

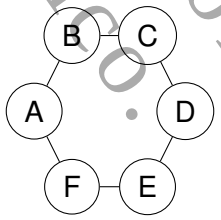


Figura 2.1. Forma molecular del benceno

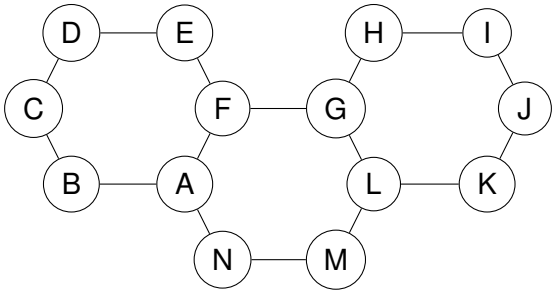


Figura 2.2. Malla hexagonal

El índice Merrifield-Simmons (MS) (Simmons y Merrifield, 1977), que en teoría de grafos

se conoce como el número de Fibonacci de un grafo $i(G)$, es un índice topológico utilizado en química matemática (Hosoya, 1971; Vesel, 2013) que representa el número de conjuntos en el grafo G. Los conjuntos independientes se estudian principalmente en el campo de la informática, ya que los grafos son una herramienta poderosa para modelar problemas de la vida real. El objetivo es contar los conjuntos independientes en un conjunto de bencenos que forman una malla poligonal. Los conjuntos independientes tienen aplicaciones potenciales en diversas áreas, como la industria, desde las telecomunicaciones y la logística hasta las finanzas y la planificación estratégica (Wurtz et al., 2022b), además de mejorar la colaboración y la gestión de datos. En la computación cuántica adiabática, resolver el problema del conjunto independiente implica encontrar todos los estados fundamentales computacionales de un hamiltoniano de muchos cuerpos $HG(V,E)$ (Yin et al., 2023b). Además, los conjuntos independientes mejoran la asignación de recursos y la distribución de tareas en las redes informáticas. Un algoritmo exacto, como el de ramificación y límite, puede resolver con precisión redes grandes, como señala (Lamm et al., 2017b).

2.2. Notación

Sea $G = (V,E)$ un grafo simple no dirigido con un conjunto de vértices V y aristas E. $I(G) = \{S/S$ es un conjunto independiente en $G\}$, entonces $I(G)$ es el conjunto de todos los conjuntos independientes de G. $i(G) = |I(G)|$, entonces $i(G)$ es el número de conjuntos independientes de G. La conexión entre los vértices u y v se representa como uv. También utilizamos la notación $\{u,v\}$ para denotar el borde uv. Definimos *N(x) ={ y* $\in$ *V : {x, y}* $\in$ *E}* como el vecindario de *x* $\in$ *V*.

Definamos algunas notaciones. *N[x] = N(x)* $\cup$ *{x}* representa el vecindario cerrado de x. Usamos $|A|$ para denotar la cardinalidad de un conjunto A. De manera similar, el grado de un vértice x se denota como $\delta(x) = |N(x)|$, mientras que el grado del grafo G es $\Delta(G) =$

$max\{\delta(x) : x \in V\}$. Un conjunto $S \subset V(G)$ de vértices de G es un conjunto independiente en G si, para cualquier par de vértices $(u,v) \in S$, entonces (u,v) no son adyacentes en G. Sea $v \in V(G)$. $I_v(G) = \{ S \in I(G) : v \in S \}$, $I_{-v}(G) = \{ S \in I(G) : v \notin S \}$. Para el cálculo del índice MS, se han desarrollado diferentes algoritmos para contar conjuntos independientes en estructuras de cuadrícula plana que facilitan el conteo aplicando el recorrido por filas y columnas o viceversa (De Ita Luna et al., 2023b).

En nuestra propuesta, el elemento principal consiste en asignar una carga (α_v, β_v) a cada vértice v del grafo. Esta carga (α_v, β_v) se calculará al visitar v durante un recorrido de G, lo que garantiza un proceso sencillo y eficiente. $(\alpha_v, \beta_v) = (|I_{-v}(G)|, |I_v(G)|)$ denota la carga del vértice $v \in V(G)$. Para procesar el número de conjuntos independientes en cualquier camino P_n, utilizaremos un hilo de cálculo o, simplemente, un hilo. Un hilo de cálculo es una secuencia de pares $(\alpha_{v_i}, \beta_{v_i})$, $i = 1, ..., n$, que se utiliza durante el cálculo incremental de $i(P_n)$, $i = 1, ..., n$, donde los n bordes de P_n son bordes de árbol, y cada par $(\alpha_{v_i}, \beta_{v_i})$ indica la carga del vértice v_i.

Simbolizamos con $\rightarrow$ cuando se aplica la regla de Fibonacci (2.1). En la ruta hamiltoniana de un grafo, simbolizamos el inicio del recorrido con |-, y el final con $>|$.

En el campo matemático de la teoría de grafos, una *ruta hamiltoniana* se define como una secuencia de aristas consecutivas en un grafo en la que se visitan todos los vértices exactamente una vez. Si el último vértice visitado es adyacente al primero, se denomina ciclo hamiltoniano.

2.3. Topología poligonal para contar conjuntos independientes

Proponemos crear un algoritmo innovador diseñado para calcular el índice de Merrifield-Simmons en sistemas bencenoides regulares $H_{r,t}$, que consisten en cuadrículas hexagonales con r filas y t columnas, como se muestra en la figura 2.3. Este algoritmo comienza con un recorrido hamiltoniano de tiempo lineal en una cuadrícula hexagonal isomórfica de $H_{r,t}$, mientras que simultáneamente calcula de forma incremental el número de conjuntos independientes. La complejidad temporal de este enfoque es menor que la del método de la matriz de transferencia cuando se aplica al cálculo del índice MS en grafos de rejilla (Euler, 2005a).

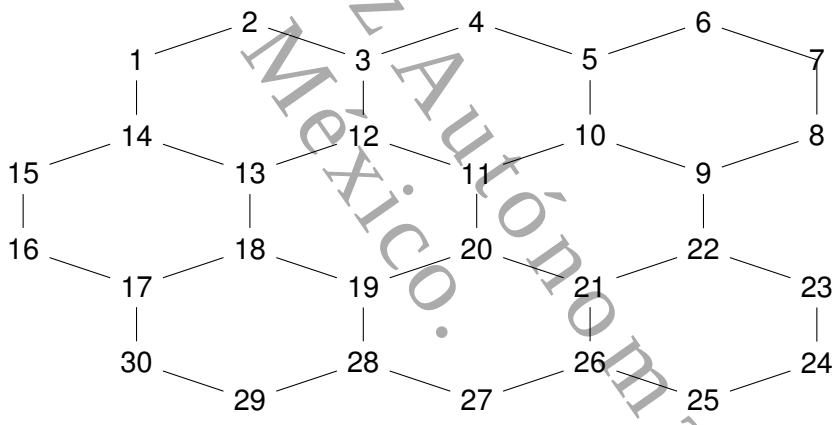


Figura 2.3. Sistema bencenoide $H_{r,t}$

2.3.1. Estrategias para calcular el índice Merrifield-Simmons

2.3.1.1. Búsqueda en profundidad

El algoritmo DFS, o algoritmo de búsqueda en profundidad, es conocido por proporcionar los pasos necesarios para explorar todos los nodos del grafo sin repetir ninguno de ellos. La figura 2.4 muestra la ruta $DFS(G,v)$ en profundidad con el nodo de origen A. La idea es

- Se marca el nodo v.
- Si todos los nodos adyacentes a v están marcados, entonces TERMINAR; de lo contrario, se elige un nodo w adyacente a v que no esté marcado.
- Se ejecuta el proceso $DFS(G, w)$.

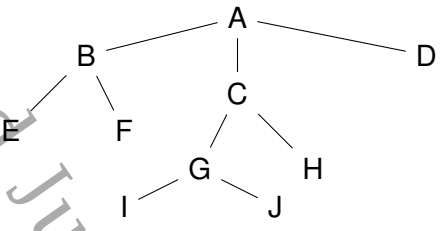


Figura 2.4. *DFS* Ruta (A, B, E, F, C, G, I, J, H, D)

2.3.1.2. Camino hamiltoniano

El primer paso de nuestro método para contar conjuntos independientes en un grafo consiste en construir una ruta hamiltoniana (H_c) sobre el grafo de entrada G. Durante este recorrido, H_c visita cada vértice del grafo exactamente una vez, identificando cada arista del grafo como una arista de árbol o una arista de *frond*. Aunque encontrar un ciclo hamiltoniano en cualquier grafo es un problema clásico $NP-completo$, en este caso las restricciones se relajan al considerar caminos en lugar de ciclos, lo que evita volver al mismo punto de partida. Esta relajación es más evidente cuando se examinan topologías de grafos como las mallas, donde la búsqueda de un H_c se convierte en un problema de complejidad temporal lineal. Un ejemplo es el caso de las mallas regulares, donde H_c puede seguir una ruta a través de filas, alternando direcciones en filas pares e impares.

2.3.1.3. Cálculo de Fibonacci en grafos

Contar el número de conjuntos independientes $i(G)$ en un grafo G implica recorrer la ruta hamiltoniana H_c de G (El-Basil, 1988b). Durante este recorrido, el primer par de valores comienza con $(\alpha_i,\ \beta_i) = (1,1)$. Para los siguientes bordes identificados, se aplica una

de dos reglas. La regla de recurrencia de Fibonacci: (2.1) cuando el borde visitado es un borde de árbol o una regla de sustracción, (2.2) cuando se reconoce un borde de *frond* (o borde posterior si se refiere a búsquedas en profundidad). Si consideramos que vi pertenece a la ruta P_n, y v_{i+1} es el siguiente vértice que visitará un borde del árbol (v_i, v_{i+1}), entonces se aplica la siguiente ecuación de recurrencia para calcular la carga $(\alpha_{v_i+1},\ beta_{v_i+1})$ en función de la carga $(\alpha_{v_i}, \beta_{v_i})$:

$$(\alpha_{v_{i+1}}, \beta_{v_{i+1}}): \quad \alpha_{v_{i+1}} = \alpha_{v_i} + \beta_{v_i}; \qquad \beta_{v_{i+1}} = \alpha_{v_i} \tag{2.1}$$

Llamamos al anterior par de recurrencias "la recurrencia de la regla de Fibonacci"porque cuando se aplican a una ruta P_{n-th}, obtenemos la identidad de conocimiento $i(P_n) = \alpha_{v_i+1} + \beta_{v_i+1} = F_n + F_{n+1} = F_{n+2}$, donde F_n es el número de Fibonacci n-ésimo.

$$(\alpha_w, \beta_w)_i = (\alpha_w, \beta_w) - (0, \beta_{vw}) = \qquad (\alpha_w, \beta_w - \beta_{vw}) \tag{2.2}$$

Ilustremos nuestra propuesta con un ejemplo; en la figura 2.5, mostramos cómo calcular el índice $M-S$ del grafo. La ruta hamiltoniana utilizada para recorrer G es $A-B-C-D-E$. Se abre un hilo principal L_p y se aplica la función de recurrencia (2.1) al comienzo del cálculo de $i(G_5) : L_p : (1,1) \to (2,1) \to (3,2) \to (5,3) \to (8,5)$. Tenga en cuenta que las cargas temporales $(\alpha_{v_i}, \beta_{v_i})$ se pueden almacenar en el vértice v_i y marcar como visitadas. Para cualquier par de vértices $(v,w) \in S$, donde S es un conjunto independiente y v,w son adyacentes, y si el borde (v,w) se identifica como un borde

de *frond*, en este caso (E,B) es el borde de *frond*, se abre un hilo secundario Ls paralelo a Lp; es decir, $L_s : (0,1) \to (1,0) \to (1,1) \to (2,1)$ cuando el recorrido visita el vértice v y w ya han sido visitados, se aplica la regla de sustracción (2.2), que permite calcular la carga del vértice v basándose en la carga del vértice u; y para cada arista de *frond* que se encuentra. La sección 3.2.5 explica esto en detalle. Por lo tanto, $i(G) = |\{\emptyset\},\{A\},\{B\},\{C\},\{D\},\{E\},\{A,C\},\{B,D\},\{C,E\},\{E,A\},\{D,A\},\{A,C,E\}\}| = 12$. $S \subseteq V(G)$ es un conjunto independiente. Si $\Delta x,y \in S$, $\{x,y\} \notin E$. $i(G) = S/S$.

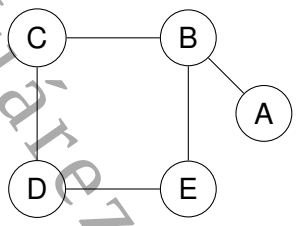


Figura 2.5. Grafo G con 5 vertices

Tabla 2.1. Contando conjuntos independientes mediante la secuencia de Fibonacci

Thread	A	B	C	D	E	i(G)	
	(α_1, β_1)	(α_2, β_2)	(α_3, β_3)	(α_4, β_4)	(α_5, β_5)		
Lp	|-(1,1) $\to$	(2,1)$\to$	(3,2) $\to$	(5,3) $\to$	(8,5)	(8,5)	
Ls		(0,1) $\to$	(1,0)$\to$	(1,1) $\to$	(2,1)	-(0,1)	(8,4) $\to$ $i(G)$ =8+4=12 >|

Continuando con nuestro enfoque sistemático, el árbol binario de la figura 2.6 presenta todas las combinaciones posibles formadas con los cinco vértices del grafo G, considerando que ninguno de sus vértices es adyacente a otro (Bracho y Sánchez, 2000b). El signo negativo indica que el vértice no puede formar parte de la combinación. Al final de la construcción del árbol binario, podemos ver que hay 12 conjuntos independientes (y eliminamos los vértices B y E, que forman el borde frontal). Además, aplicando el conteo de la secuencia de Fibonacci de la tabla 2.1, confirmamos que los resultados, que son de gran importancia, coinciden.

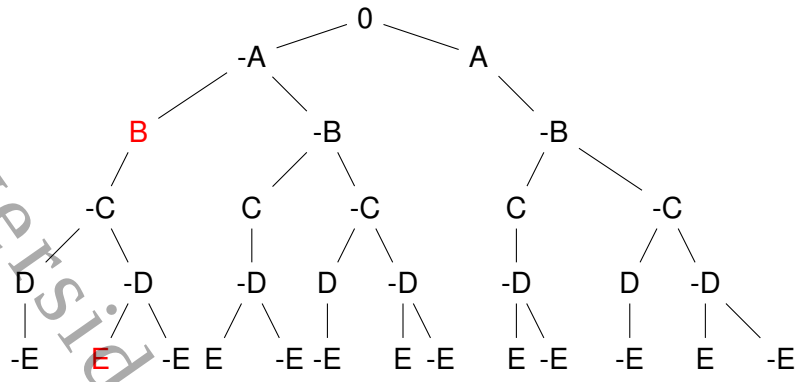


Figura 2.6. Árbol binario de conjuntos independientes

2.3.2. Conteo de conjuntos independientes en topologías básicas

2.3.2.1. Grafo de cadena lineal

Sea $G=(V,E)$ con $\Delta(G)=2$, siendo G un grafo en cadena con n nodos, por lo tanto $|V|=n$ y $|E|=m=n-1$. Ordenemos los nodos en G como $V=v_1,v_2,...,v_n$ de tal manera que se cumpla $E=c_1,...,c_{n-1}=v_1,v_2,v_2,v_3,...,v_{n-2},v_{n-1},v_{n-1},v_n$ es decir, $|v(c_i)\bigcap v(c_{i+1})|=1, i=1,...,n-2$. G_i, $i=1,...,n$ siendo G_i el subgrafo inducido de G. La técnica básica para contar el número de conjuntos independientes $i(G_i)$ en G_i de G es una estrategia incremental (Samotij, 2015; Okamoto et al., 2005) que se basa en asociar un par de valores (α_i,β_i) a cada nodo $v_i\in V$, $i=1,...,n$. La ruta comienza en uno de los extremos del nodo v_1 o del nodo v_n, visitando su nodo adyacente de forma lineal. El primer par de valores comienza con $(\alpha_i,\beta_i)=(1,1)$. La relación de recurrencia considera que se conoce el valor (α_i,β_i) asociado al subgrafo inducido G_i de G tal que $i(G_i)=(\alpha_i,\beta_i)$. El nuevo par de valores $(\alpha_{i+1},\beta_{i+1})$ se construye basándose en (α_i,β_i) aplicando la recurrencia que genera a los números de Fibonacci $\alpha_{i+1}=\alpha_i+\beta_i$, $\beta_{i+1}=\alpha_i$.

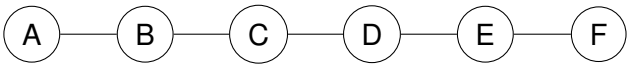


Figura 2.7. Cadena lineal

$(\alpha_1,\beta_1)\rightarrow$ (1,1) $\rightarrow$ $(\alpha_2,\beta_2)\rightarrow$ (2,1)$\rightarrow$ $(\alpha_3,\beta_3)\rightarrow$ (3,2) $\rightarrow$ $(\alpha_4,\beta_4)\rightarrow$ (5,3) $\rightarrow$ $(\alpha_5,\beta_5)\rightarrow$ (8,5) $\rightarrow$ (α_6,β_6) (13,8)

La tabla 2.2 muestra el conteo de conjuntos independientes obtenidos al aplicar la se-

cuencia de Fibonacci en el grafo en cadena que se muestra en la figura 2.7. Por lo tanto, (α_i,β_i)=(F_{i+1},F_i) entonces $i(G_i)$=α_i+β_i=F_{i+1}+F_i=F_{i+2}, $i = \{0,...,n\}$ es decir, para $n = 6$, $i(G) = i(G_6)$ =F_7+F_6=$F_8 = 21$.

Tabla 2.2. Conjuntos independientes del grafo de cadena lineal

A	B	C	D	E	F
$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)
$(1,1) \rightarrow$	$(2,1) \rightarrow$	$(3,2) \rightarrow$	$(5,3) \rightarrow$	$(8,5) \rightarrow$	$(13,8)$
$(F_2,F_1) \rightarrow$	$(F_3,F_2) \rightarrow$	$(F_4,F_3) \rightarrow$	$(F_5,F_4) \rightarrow$	$(F_6,F_5) \rightarrow$	$(F_7,F_6)^a$

2.3.2.2. Ciclo simple de un grafo

La figura 2.8 muestra un grafo que contiene un ciclo simple. El conteo de conjuntos independientes inicia un hilo principal (L_p); el par de valores comienza con $(\alpha_i,\beta_i) = (1,1)$, y el hilo secundario (L_s) inicia la recurrencia de la secuencia con $(\alpha_i,\beta_i) = (0,1)$, donde $i = 1,...,m$. El último par es $(\alpha_m,\beta_m) = (0,\beta_m)$, de modo que la serie $((\alpha_1,\beta_1) = (0,1) \rightarrow (\alpha_2,\beta_2) = (1,0) \rightarrow (\alpha_3,\beta_3) = (1,1)... \rightarrow (\alpha_m,\beta_m) \rightarrow (0,\beta_m))$ nos da el valor de $|S \in i(G') : v_1 \in S \wedge v_m \in S|$. La serie anterior corresponde a la siguiente serie considerando los números de Fibonacci: $F_0 = 0$ y $F_1 = 1$, $(F_0,F_1) \rightarrow (F_1,F_0) \rightarrow (F_2,F_1) \rightarrow ... \rightarrow (F_{m-1},F_{m-2})$.

$$(\alpha_m,\beta_m) = (0,\beta_m); \qquad i(G_c) = ((\alpha_m,\beta_m) - (0,\beta a_m))$$

Para calcular el cierre del ciclo (De Ita et al., 2020b), aplicamos la regla (2.2); el último par (α_m,β_m) del hilo secundario (Ls) es $(\alpha_m, \beta_m) = (0,\beta_m)$, donde $\{v_m,v_0\} \in E(G)$ representa el borde hacia atrás que cierra el ciclo de G y se resta del último par del hilo primario (Lp). Por lo tanto, $((\alpha_m,\beta_m) - (0,\beta_m)) = ((\alpha_6,\beta_6) - (0,\beta_6))$=(13,8) – (0,3) = (13,5), obtenemos que $i(Gc)$ = 13 + 5, es decir, 18 es el número total de conjuntos independientes. Este conteo se muestra en detalle en la Tabla 2.3, donde podemos ver el conteo de conjuntos independientes de G.

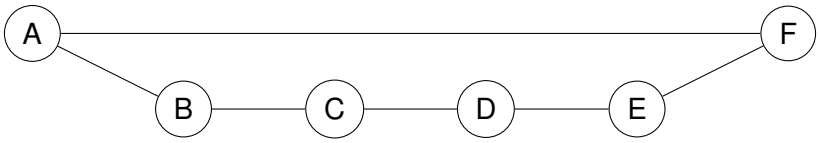


Figura 2.8. Ciclo simple

Tabla 2.3. Conjuntos independientes de grafos de ciclo simple

Hilo	A	B	C	D	E	F	
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)	
Lp	(1,1) $\rightarrow$	(2,1)$\rightarrow$	(3,2) $\rightarrow$	(5,3) $\rightarrow$	(8,5) $\rightarrow$	(13,8)	(13,8)*
Ls	(0,1) $\rightarrow$	(1,0)$\rightarrow$	(1,1) $\rightarrow$	(2,1) $\rightarrow$	(3,2) $\rightarrow$	(5,3)	-(0,3)**
$F_0 = 0, F_1 = 1$	$(F_2, F_1) \rightarrow$	$(F_3, F_2) \rightarrow$	$(F_4, F_3) \rightarrow$	$(F_5, F_4) \rightarrow$	$(F_6, F_5) \rightarrow$	(F_7, F_6)	$(13, 5)$

2.3.2.3. Ciclo y cadena lineal de un grafo

La figura 2.9 muestra un grafo con nueve nodos, donde los nodos A, B, C, D, E, F y A forman un ciclo. Se sigue el proceso exacto para contar los conjuntos independientes en un ciclo para un grafo de ciclo simple, estudiado anteriormente en la sección 2.3.2.2; el cálculo se muestra en detalle en la tabla 2.4. Cuando el nodo F cierra el ciclo, se aplica la regla de arista inversa o sustracción (2.2), obteniéndose como resultado el par $(13, 5)$ $\rightarrow$ y continuando con el cálculo de la secuencia lineal aplicando la regla de Fibonacci (2.1) en los nodos G, H, I hasta terminar el recorrido del gráfico. El número total de conjuntos independientes $i(G)$ es igual a la suma del último par (49,31)= 49+31, es decir, 80 conjuntos independientes.

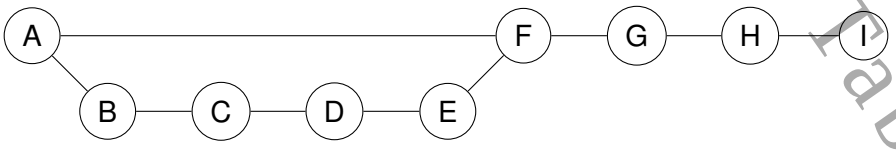


Figura 2.9. Ciclo simple y cadena lineal

Tabla 2.4. Conjuntos independientes del ciclo simple y la cadena lineal del grafo

Hilo	A	B	C	D	E	F	G	H	I
	(α_1, β_1)	(α_2, β_2)	(α_3, β_3)	(α_4, β_4)	(α_5, β_5)	(α_6, β_6)	(α_7, β_7)	(α_8, β_8)	(α_9, β_9)
Lp	(1,1) →	(2,1)→	(3,2) →	(5,3) →	(8,5) →	(13,8)→			
Ls	(0,1) →	(1,0)→	(1,1) →	(2,1) →	(3,2) →	(5,3)→			
						(13,5)→	(18,13)→	(31,18)→	(49,31)*

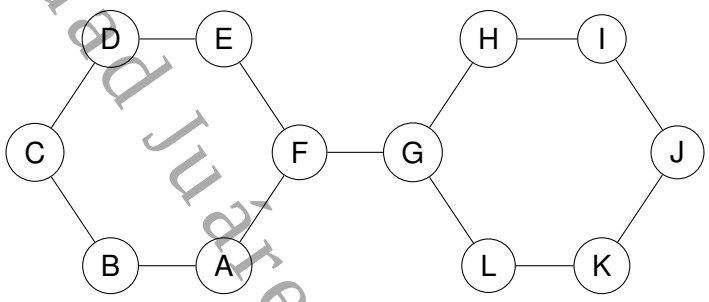


Figura 2.10. Dos hexágonos unidos por una arista

Tabla 2.5. Ciclo 1

Hilo	A	B	C	D	E	F	
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)	
Lp	(1,1) →	(2,1)→	(3,2) →	(5,3) →	(8,5) →	(13,8)	(13,8)
Ls	(0,1) →	(1,0)→	(1,1) →	(2,1) →	(3,2) →	(5,3)	-(0,3)*
							(13,5)**

Tabla 2.6. Ciclo 2

Hilo	G	H	I	J	K	L
	$(\alpha_1, \beta_1) \rightarrow$	$(\alpha_2, \beta_2) \rightarrow$	$(\alpha_3, \beta_3) \rightarrow$	$(\alpha_4, \beta_4) \rightarrow$	$(\alpha_5, \beta_5) \rightarrow$	(α_6, β_6)
Lp	(18,13) →	(31,18)→	(49,31) →	(80,49) →	(129,80) →	(209,129))
Ls	(0,13) →	(13,0)→	(13,13) →	(26,13) →	(39,26) →	(65,39)
						(209,129)
						-(0,39)*
						(209,90)

2.3.2.4. Hexagonos connecteados por aristas

La figura 2.10 muestra el caso de un grafo con dos ciclos conectados por un borde puente. Aquí se realiza el mismo proceso para los ciclos simples y, una vez encontrado un borde posterior, se cierra el primer ciclo, se aplica la ecuación (2.2) y se continúa calculando los conjuntos independientes según la regla de Fibonacci. Las tablas 2.5 y 2.6 muestran que los cálculos del total de los conjuntos independientes $i(G)$ son iguales a la suma del último par (209, 90)= 209+90=299 conjuntos independientes.

2.3.2.5. Hexagonos conectados por vertices

Se han desarrollado algoritmos para contar conjuntos independientes en estructuras de cuadrícula plana que facilitan el conteo, aplicando el recorrido por filas y columnas o viceversa (De Ita Luna et al., 2023b). Nuestra propuesta algorítmica procesa las aristas *frond* en dos fases. Sea v,w un arista *frond* de un grafo G. En la primera fase, cuando un hilo principal del recorrido (L_p) visita el primer vértice v de la arista *frond*, se debe duplicar el número de hilos activos y de hilos de cálculo (L_s).

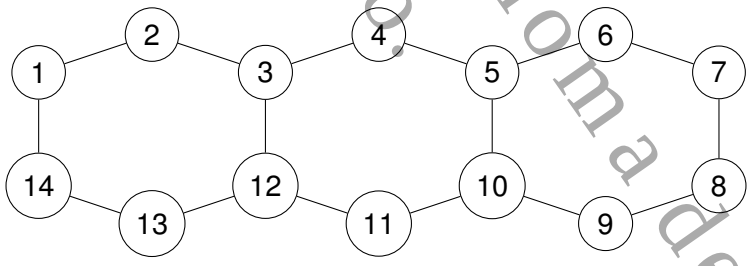


Figura 2.11. Tres hexágonos unidos por vertices

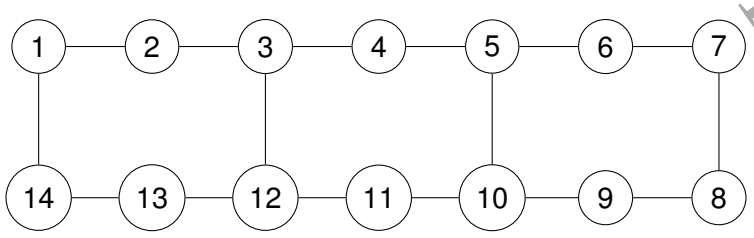


Figura 2.12. Malla de tres hexágonos unidos por vertices

Supongamos que $(\alpha_v, \beta_v)_i$ es la carga asociada al hilo activo L_i al visitar el vértice v. Se

crea un nuevo hilo L_{vw_i}, que está subordinado al hilo maestro L_i, y con un par inicial asociado $(0, \beta_v)_{vw}$. Por lo tanto, la etiqueta vwi de L_{vwi} es también un puntero a su hilo maestro L_i. La segunda fase en el procesamiento de la arista *frond* (v, w) es cuando el recorrido visita el vértice w, y v ya ha sido etiquetado como un vértice visitado.

Por lo tanto, el control se mantiene en el vértice w. Supongamos que $(\alpha_w, \beta_w)_i$ es la carga del vértice w en el hilo maestro L_i. Mientras tanto, $(\alpha_{vw}, \beta_{vw})_{vw}$ es la carga de w en el hilo subordinado L_{vw_i}. Entonces, la regla de sustracción de la ecuación (2.2) actualiza la carga de w en L_i. Tras la aplicación de esta conteo, el hilo subordinado L_{vw_i} se cierra. De esta forma, cualquier hilo en el que aparezca la partícula vw debe cerrarse, lo que disminuye el número de hilos activos.

Es posible calcular el número de conjuntos independientes de una cuadrícula tomando como guía la ruta hamiltoniana H_c, aunque la complejidad temporal es de orden exponencial sobre el número máximo de aristas *frond* en cualquier fila de la cuadrícula, dado el número de ciclos que se mantienen abiertos durante la ruta Hc. Las conteos de Fibonacci y de sustracción son suficientes para procesar una fila de fichas de una rejilla hexagonal, como los de las las figuras 2.11 y 2.12.

2.4. Algoritmo de ramificación y poda

2.4.1. Sistema bencenoide

La primera etapa de nuestro método para calcular el índice de Merrifield-Simmons en bencenoides $I(H_{r,t})$ consiste en insertar el sistema bencenoide $H_{r,t}$ de la figura 2.3 en una malla hexagonal regular HG como se muestra en la figura 2.13, donde en lugar de cuadrados convencionales, consideramos hexágonos.

Tabla 2.7. Conjuntos independientes de la Figura 2.12

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Lp (1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	(21,13)	(34,21)	(55,34)	(89,55)	-(0,15)=	(89,40)		
Ls⊣(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	(21,13)	(34,21)	-(0,6)=	(34,15)		
	c2Lp	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	(16,10)	(26,16)	-(0,6)=	(26,10)		
	c2Ls	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(13,8)	-(0,3)=	(13,5)		
			c3Lp	(0,5)	(5,0)	(5,5)	(10,5)	(15,10)	(25,15)	x			
				c3Ls	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	x		
					c3c2Lp	(0,2)	(2,0)	(2,2)	(4,2)	(6,4)	(10,6)	x		
					c3c2Lp	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(5,3)	x		
										(129,89)	(218,129)	(311,218)	(529,311)
											-(0,36)		-(0,114)
											=(218,93)		=(529,197)
										(49,34)	(83,49)	(114,83)	(197,114) ⊣
											-(0,18)		x
											=(83,31)		
										(36,10)	(46,36)	x	
										(18,13)	=(31,18)	x	

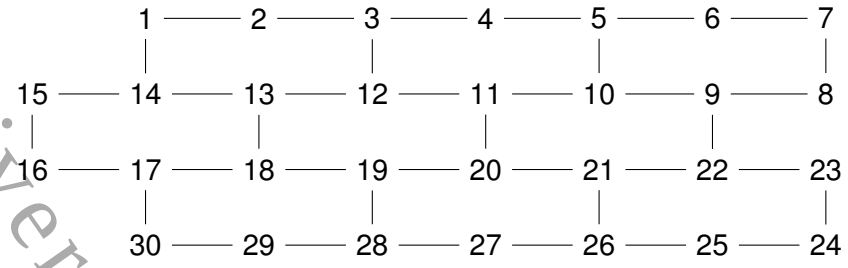


Figura 2.13. Malla hexagonal regular $HG_{m,n}$

Ambos grafos, el sistema bencenoide $H_{r,t}$ y la malla hexagonal regular HG, son isomorfos y poseen el mismo número de hexágonos. Designamos el número de filas en HG como m y el número de columnas en la primera fila como n.

Por lo tanto, $HG_{m,n}$ representa una malla hexagonal con m filas, donde cada fila tiene n o $n+1$ columnas. La primera fila en $HG_{m,n}$ tiene n vértices, mientras que las filas 2 y 3 tienen $n+1$ vértices. La fila 4 vuelve a tener n vértices, y las dos filas siguientes tienen $n+1$ vértices, y así sucesivamente. El número de hexágonos es el mismo en cada fila de $HG_{m,n}$, por lo que los hexágonos de $H_{r,t}$ están relacionados uno a uno con los hexágonos de $HG_{m,n}$.

La inserción de un sistema bencenoide $H_{r,t}$ en una malla hexagonal regular $HG_{m,n}$ se puede realizar en tiempo lineal en $r \cdot t$, gracias a la existencia de algoritmos de tiempo lineal que crean la inserción de un grafo plano en una malla (Vadhan, 2001a). Por lo tanto, podemos representar las aristas de $HG_{m,n}$ como segmentos de línea recta. $HG_{m,n}$ solo tiene vértices de grado dos o tres. Las aristas se dividen en dos tipos: aristas horizontales con vértices $\{(i,j),(i,j+1)\}$, donde $i = 1,...,m$, y $j = 1,...,n$; y aristas verticales con vértices $\{(i,j),(i+1,j)\}$ o $\{(i,j),(i+1,j+1)\}$, o $\{(i,j),(i+1,j-1)\}$ dependiendo de la fila considerada.

2.4.2. Ramificación y poda

1. **Camino hamiltoniano**

   La ruta hamiltoniana Se puede formar basándose en una ruta a través de las filas de la cuadrícula cambiando la dirección de la ruta de izquierda a derecha en las filas impares y de derecha a izquierda en las filas pares. Comenzamos construyendo una ruta hamiltoniana $I_{\_v}(G)$ que recorre cada vértice de $HG_{m,n}$ una vez. Para los vértices de grado 2, $\delta(v) = 2$, el caminante visita ambos bordes y continúa la ruta; para los vértices de grado 3, $\delta(v) = 3$, el caminante visita los tres aristas y comprueba si el grado de sus vecinos es igual a tres. Si $\delta(N(v)) = 3, N(v) = 3$, entonces v es el vértice seleccionado para aplicar la poda, como se muestra en la 2.14, subsección (a).

2. **conteos de conteo**

   Conjunto de conteos de conteo que se aplicarán a los aristas (o vértices) durante el camino (Morrison et al., 2016). La ruta hamiltoniana Se puede formar basándose en una ruta a través de las filas de la cuadrícula cambiando la dirección de la ruta de izquierda a derecha en las filas impares y de derecha a izquierda en las filas pares. Comenzamos construyendo una ruta hamiltoniana $I_{\_v}(G)$ que recorre cada vértice de $HG_{m,n}$ una vez. Para los vértices de grado 2, $\delta(v) = 2$, el caminante visita ambas aristas y continúa la ruta; para los vértices de grado 3, $\delta(v) = 3$, el caminante visita los tres aristas y comprueba si el grado de sus vecinos es igual a tres. Si $\delta(N(v)) = 3, N(v) = 3$, entonces v es el vértice seleccionado para aplicar la poda, como se muestra en la figura 2.14, subsección (a).

3. **conteos de conteo**

   Conjunto de conteos de conteo que se aplicarán a los aristas (o vértices) durante el recorrido. (Morrison et al., 2016).

a) conteo de ramificación:

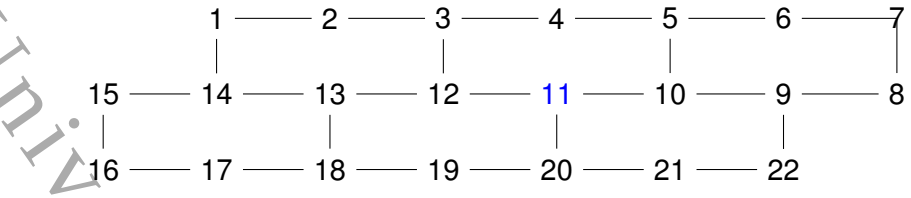
$$i(G) = i(G-v) + i(G-N[v]) \tag{2.3}$$

b) El vértice a seleccionar debe ser de grado 3, es decir, δ(v)=3; por ejemplo, en la figura 2.14, sección (a), se muestra el vértice que se ha seleccionado.

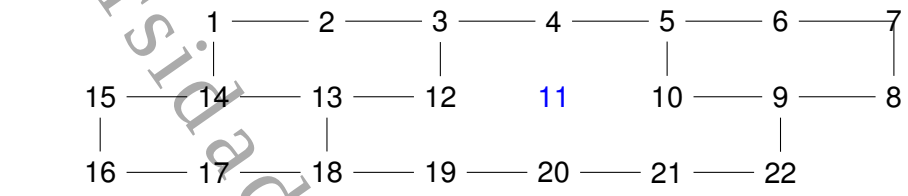
c) Sus vecinos también deben ser de grado 3, es decir, δ(N(v))=3; en la figura 2.14, parte (b), se pueden ver cuáles son los vecinos de v, en este caso, son $\{12, 10, 20\}$.

Al implementar la regla de ramificación (2.3) en el vértice v, se generan dos nodos, v_1 y v_2, a partir del nodo actual en el árbol de cálculo. El subgrafo asociado con v_1 se establece como $G_1 = (G-v)$, mientras que el subgrafo asociado con v_2 se establece como $G_2 = (G-N[v])$. Tenemos un problema similar para cada subgrafo $G_i, i = 1,2$ que teníamos con la cuadrícula original HG. Si resolvemos el problema de forma recursiva y r_i es su solución, entonces la solución completa para $r = i(HG)$ es $r = r_1 + r_2$. El procedimiento anterior determina un árbol enumerativo cuyas hojas corresponden a las instancias de subgrafos denominadas casos base del árbol enumerativo. El proceso de ramificación se repite hasta que se completa la ruta de todo el grafo y el caminante de la ruta hamiltoniana no encuentra otro vértice que satisfaga las conteos. Es el momento de aplicar el conteo de conjuntos independientes a los casos básicos resultantes, que se muestran en la subsección (d) de la figura 2.14, utilizando la serie de Fibonacci.

Una vez definidos los casos base, aplicamos las estrategias computacionales para contar

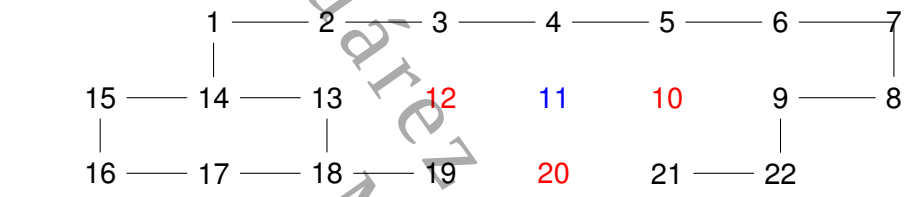


(a) Vértice seleccionado $(v = 11)$ en $HG_{m,n}$



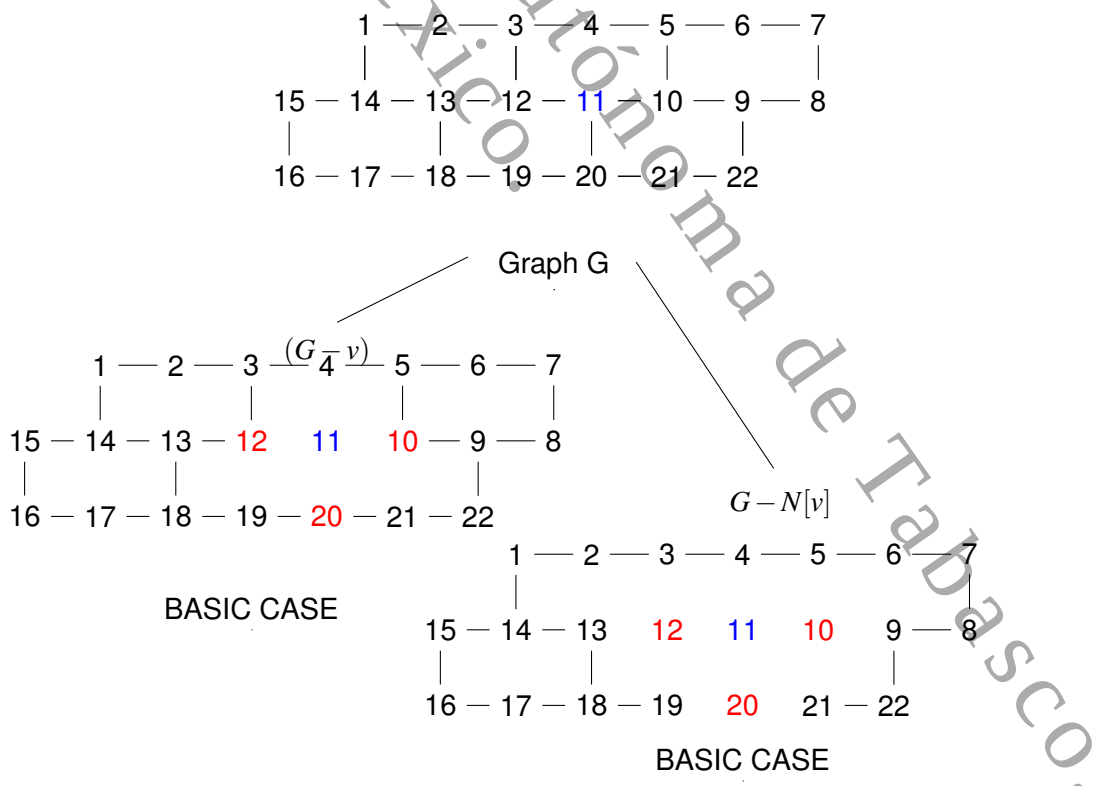
(b) Vecinos $N(v)$

$$N(v) = \{(12/\{11,12\}), (10/\{11,10\}), (20/\{11,20\}) \in E\}$$



(c) Vecindad cerrada

$$N[v] = \{12, 10, 20\} \cup \{11\}$$



(d) Caso base

Figura 2.14. Ramificación y poda

conjuntos independientes en estructuras básicas simples y estructuras de malla, como la cadena lineal, el ciclo simple, el ciclo simple y la cadena lineal, los hexágonos conectados por aristas y los hexágonos conectados por vértices, tal y como se presentó en la sección 2.3.2 La complejidad temporal del algoritmo de ramificación y poda viene determinada por la recurrencia $i(G) = i(G-v) + i(G-N[v])$. Nuestra propuesta sigue presentando una complejidad temporal exponencial, pero no muestra el carácter combinatorio explosivo que tiene el método clásico de la matriz de transferencia para calcular $i(HG_{m,n})$.

2.5. Conclusion y trabajo futuro

Hemos presentado una propuesta de ramificación y poda para calcular el número de conjuntos independientes en mallas hexagonales, denotadas como $HG_{m,n}$, donde m representa el número de filas y n el número de columnas. En nuestro enfoque, utilizamos la conteo de ramificación para dividir el grafo en subgrafos más simples. Nuestra estrategia consiste en descomponer el grafo original en subgrafos planos externos, tratándolos como casos básicos. La complejidad temporal de nuestro enfoque para este cálculo es significativamente menor que la requerida por el método clásico de la matriz de transferencia para lograr el mismo resultado.

En el futuro, tenemos previsto investigar técnicas de optimización combinatoria para reducir la complejidad del algoritmo de ramificación y poda para calcular conjuntos independientes en mallas hexagonales, y mejorar aún más nuestra comprensión y capacidad para abordar este reto computacional de forma más eficaz. Esto nos permitirá tratar de forma más eficiente grafos más grandes. Por lo tanto, esta propuesta podría aplicarse como una solución eficaz para procesar BIG DATA y otras redes complejas mediante la optimización de la selección de nodos y la reducción de redes.

Referencias

Bonilla García, A. N. (2019a). ¿ Podría ser la gravedad una fuerza entrópica?

Bracho, R. L., & Sánchez, M. P. O. (2000b). Un algoritmo paralelo para el problema del conjunto independiente. *Revista de Matemática: Teoría y Aplicaciones*, *7*(1-2), 125-134.

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020b). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets, 57-66. https://doi.org/https://doi.org/10.1007/978-3-030-49076-8_6

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023b). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1). https://research.ebsco.com/linkprocessor/plink?id=4318a715-0f3e-30fc-9741-35999983df7e

El-Basil, S. (1988b). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29.

Euler, R. (2005a). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Gutman, I. (1982b). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73. https://doi.org/https://doi.org/10.1515/zna-1982-0115

Hosoya, H. (1973b). Topological index and Fibonacci numbers with relation to chemistry. *Fibonacci Quart*, *11*(3), 255-266.

Lamm, S., Sanders, P., Schulz, C., Strash, D., & Werneck, R. F. (2017b). Finding near-optimal independent sets at scale. *Journal of Heuristics*. https://doi.org/https://doi.org/10.1137/1.9781611974317.12

Merrifield, R. E., & Simmons, H. E. (1980). The structures of molecular topological spaces. *Theoretica chimica acta*, *55*, 55-75. https://doi.org/https://doi.org/10.1007/BF00551410

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79-102. https://doi.org/https://doi.org/10.1016/j.disopt.2016.01.005

Okamoto, Y., Uno, T., & Uehara, R. (2005). Linear-time counting algorithms for independent sets in chordal graphs. *Graph-Theoretic Concepts in Computer Science: 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers 31*, 433-444.

Samotij, W. (2015). Counting independent sets in graphs. *European Journal of Combinatorics*, *48*, 5-18. https://doi.org/https://doi.org/10.1016/j.ejc.2015.02.005

Simmons, H., & Merrifield, R. (1977). Mathematical description of molecular structure: Molecular topology. *Proceedings of the National Academy of Sciences*, *74*(7), 2616-2619. https://doi.org/https://doi.org/10.1073/pnas.74.7.2616

Vadhan, S. P. (2001a). The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, *31*(2), 398-427. https://doi.org/10.1137/S0097539797321602

Vesel, A. (2013). Fibonacci dimension of the resonance graphs of catacondensed benzenoid graphs. *Discrete Applied Mathematics*, *161*(13), 2158-2168. https://doi.org/https://doi.org/10.1016/j.dam.2013.03.019

Wurtz, J., Lopes, P. L., Gemelke, N., Keesling, A., & Wang, S. (2022b). Industry applications of neutral-atom quantum computing solving independent set problems. *arXiv preprint arXiv:2205.08500*.

Yin, X.-F., Yao, X.-C., Wu, B., Fei, Y.-Y., Mao, Y., Zhang, R., Liu, L.-Z., Wang, Z., Li, L., Liu, N.-L., et al. (2023b). Solving independent set problems with photonic quantum circuits. *Proceedings of the National Academy of Sciences*, *120*(22), e2212323120. https://doi.org/https://doi.org/10.1073/pnas.2212323120

Capítulo 3

Algoritmo de detección de caras en grafos hexagonales: un enfoque para el conteo de conjuntos independientes

Algoritmo de detección de caras en grafos hexagonales: un enfoque para el conteo de conjuntos independientes

[1] Herlinda González Vázquez $231H18004@alumno.ujat.mx$, [1]Cristina López Ramírez $cristina.lopez@ujat.mx$, [2]Pedro Bello López $pedro.bello@correo.buap.mx$, [2]Guillermo De Ita Luna $deitaluna63@gmail.com$

Resumen: En este trabajo de investigación se propone un algoritmo para el cálculo del índice Merrifield-Simmons (MS) sobre mallas de caras hexagonales. Este tipo de mallas se usan para modelar grafos moleculares de grafenos, o de compuestos bencenoides aromáticos, entre otros. El algoritmo utiliza una trayectoria hamiltoniana que recorre cada vértice una sola vez en una cuadrícula hexagonal isomórfica. El proceso incluye la creación de una lista de adyacencia, la construcción del grafo, su matriz de adyacencia, y la realización de una búsqueda en profundidad para identificar ciclos y aristas de retroceso. De esta manera, se calcula de manera incremental el número de conjuntos independientes existentes en las mallas hexagonales, obteniéndose, además una reducción de la complejidad temporal en tiempo del recorrido DFS que es de orden lineal sobre el número de aristas que tiene el grafo en el proceso de conteo en comparación con los métodos tradicionales usados para este fin.

Palabras clave: Mallas hexagonales, Conteo de Conjuntos independientes, Índice Merrifield-Simmons, Grafos moleculares

Institución de adscripción: [1]Universidad Juárez Autónoma de Tabasco, [2]Benemérita Universidad Autónoma de Puebla.

3.1. Introducción

La teoría de grafos tiene su origen en el siglo XVIII con el problema de los puentes de Königsberg. Este problema consistía en encontrar un recorrido que atraviese los siete puentes sobre el río Pregel, pasando por cada uno de ellos una única vez. Este dilema se puede representar mediante grafos, lo que permite modelar de forma simple diversos problemas que pueden resolverse computacionalmente. De manera análoga, el cálculo de conjuntos independientes (CI) puede abordarse utilizando la teoría de grafos. El índice Merrifeld-Simmons i(G) de un grafo G introducido por Merrifeld-Simmons en 1989, es un índice topológico utilizado en química matemática (R. E. Merrifield y Simmons, 1989a); se define como el número de subconjuntos del conjunto de vértices, en el que dos vértices cualesquiera no son adyacentes entre sí, es decir, el número de vértices independientes del conjunto de G. Sin embargo, éste índice en el área de la teoría de grafos es denominado como el número de Fibonacci de un grafo i(G) (Hosoya, 1973a).

El presente trabajo introduce algoritmos que buscan mejorar la complejidad de conteo de conjuntos independientes (CI). Estas técnicas incluyen el recorrido de filas y columnas (De Ita Luna et al., 2023c) y transformaciones que pueden ser similares a problemas de recorridos sobre un tablero de ajedrez (Deng et al., 2017), lo cual sugiere una aplicación de métodos combinatorios y geométricos sobre mallas hexagonales para identificar y enumerar ciclos, lo cual es fundamental para analizar y optimizar redes, mejorar el recorrido de grafos, seleccionar nodos estratégicos y reducir redes complejas. Estas técnicas son valiosas para el análisis de grafos, ya que ayudan a simplificar estructuras complejas, mejorar la eficiencia del procesamiento de datos y resolver problemas de conectividad y flujo. La tarea de contar conjuntos independientes es relevante porque está asociada a aplicaciones en problemas de optimización, como la planificación y la asignación de recursos, además de permitir conocer la temperatura a la cual los enlaces de compuestos químicos se rompen y se determinan los puntos de ebullición de dichos compuestos

(índice Merrifeld-Simmons del compuesto). Esta tarea es esencial para reconocer propiedades fisicoquímicas de compuestos formados por mallas hexagonales, como las placas de grafeno y los compuestos aromáticos, tales como los bencenos.

3.2. Preliminares

Sea *G=(V, E)* un grafo simple no dirigido con un conjunto de vértices *V* y aristas *E*. La conexión entre los vértices *u* y *v* se representa como *uv*. También usamos la notación *{u, v}* para denotar la arista *uv*. Dos vértices son adyacentes si están unidos por una misma arista, y no son adyacentes cuando no lo están.

Definamos $N(x) = y \in V : x, y \in E$ como la vecindad de $x \in V$. $N[x] = N(x) \cup x$ representa la vecindad cerrada de *x*. Usamos *|A|* para denotar la cardinalidad de un conjunto *A*. Del mismo modo, el grado de un vértice *x* se denota como $\delta(x) = |N(x)|$, mientras que el grado del grafo G es $\Delta(G) = max\delta(x) : x \in V$. Un conjunto $S \subset V(G)$ de vértices de *G* es un conjunto independiente en *G*, si para cualquier par de vértices $u, v \in S$, se cumple que $\{u, v\}$ no está en *E*. Se denota por *I(G)=*{*S/S* es un conjunto independiente en *G* - el conjunto de todos los conjuntos independientes de *G*, mientras que *i(G)=|I(G)|* - es el número de conjuntos independientes de *G*. Sea $v \in V(G)$, se denota por $Iv(G) = \{S \in I(G) : v \in S\}, I - v(G) = \{S \in I(G) : v$ no está en $S\}$.

Para el cómputo de conjuntos independientes se han desarrollado diferentes algoritmos en estructuras reticulares planas que facilitan el conteo, aplicando el recorrido por filas y columnas o viceversa (De Ita Luna et al., 2023c) . Deng et al., 2017 transforman el problema de formar los conjuntos independientes de un grafo en un problema equivalente al movimiento de las piezas en un tablero de ajedrez. Proponemos un algoritmo para calcular el índice Merrifield-Simmons (MS) R. E. Merrifield y Simmons, 1981a en sistemas bencenoides regulares *Hr,t* , que consisten en mallas hexagonales con *r* filas y

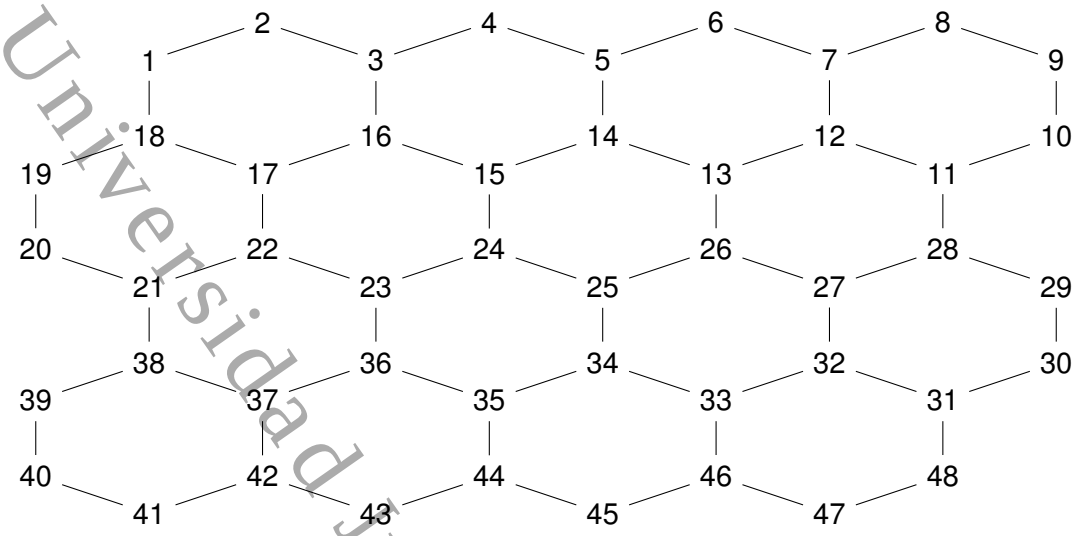


Figura 3.1. Sistema bencenoide $H_{r,t}$

t columnas, como se muestra en la Figura 3.1. El algoritmo inicia realizando un recorrido hamiltoniano (lo que consume un tiempo lineal sobre una rejilla hexagonal isomórfica de *Hr,t)* mientras se va calculando de forma simultánea el número de conjuntos independientes que hay en la malla. La complejidad temporal de este enfoque es menor que la del método de la matriz de transferencia (Euler, 2005b) cuando se aplica al cálculo del índice MS en grafos reticulares (De Ita et al., 2023a).

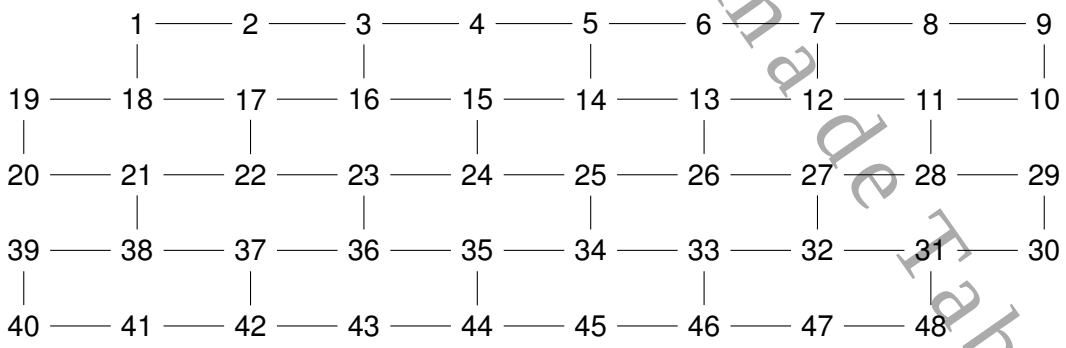


Figura 3.2. Malla hexagonal regular $HG_{m,n}$

La primera fase de nuestro método para calcular el índice de MS en bencenoides *I(Hr,t)* consiste en redibujar el sistema bencenoide *Hr,t* que se muestra en la Figura 3.1 como malla hexagonal regular *HG* como se ilustra en la Figura 3.2. En *HG* se dibujan hexágo-

nos en lugar de cuadrados convencionales de las mallas cuadriculadas. Ambos grafos, el sistema bencenoide *Hr,t* y la malla hexagonal regular *HG*, son grafos isomorfos y contienen el mismo número de hexágonos. Definimos como *m* el número de filas de *HG* y como *n* el número de columnas de la primera fila. Así, *HGm,n* representa una malla hexagonal con *m* filas, donde cada fila tiene *n* o *n + 1* columnas. La primera fila de *HGm,n* tiene *n* vértices, mientras que las filas 2 y 3 tienen *n + 1* vértices cada una. La fila 4 vuelve a tener *n* vértices, y las dos filas siguientes tienen *n+1* vértices, y así sucesivamente. El número de hexágonos es constante en cada fila de *HGm,n*, por lo que hay una correspondencia uno a uno entre los hexágonos de *Hr,t* y los de *HGm,n*. La inserción de un sistema bencenoide *Hr,t* en una malla hexagonal regular *HGm,n* se puede realizar en tiempo lineal *O(r*t)*, gracias a la existencia de algoritmos de tiempo lineal que permiten reducir un grafo plano en una malla (Vadhan, 2001b). Por lo tanto, podemos representar las aristas del *HGm,n* como segmentos de línea recta. *HGm,n* sólo tendrá vértices de grado dos o tres. Las aristas se dividen en dos tipos: aristas horizontales con vértices $(i, j), (i, j+1)$, donde $i = 1, ..., m$, y $j = 1, ..., n$; y aristas verticales con vértices $(i, j), (i+1, j)$ o $(i, j), (i+1, j+1)$, o bien $(i, j), (i+1, j-1)$ según la fila considerada (De Ita et al., 2020c).

3.3. Métodos para determinar el índice Merrifield-Simmons

Como paso inicial del método para calcular el Índice Merrifield-Simmons en una malla hexagonal del grafo G, primero debemos identificar las caras construyendo un camino Hamiltoniano (Hc) sobre el grafo de entrada, para luego seguir con el conteo de los conjuntos independientes aplicando el cálculo de Fibonacci en grafos, que se explica a detalles en el apartado 3.3.3. Iniciamos identificando las caras con los datos de entrada, luego se representan usando una lista de adyacencia y una matriz de adyacencia. El grafo se construye a partir de la lista de adyacencia, y se aplica el algoritmo de búsqueda en profundidad (DFS, por sus siglas en inglés) para recorrer el grafo. A lo largo del recorrido

DFS, se rastrean los vértices visitados y se marcan en la matriz de adyacencia como visitados (con el valor 0); los vértices que quedaron marcados (con el valor 1) son los no visitados y representan las aristas de retrocesos en el grafo. En base al recorrido DFS se crea una lista enlazada e inicia la ejecución del Algoritmo conteo de caras y aquí se van añadiendo y enlazando las aristas de retroceso encontradas en la matriz de adyacencia, que luego es utilizada para identificar las caras (ciclos) del grafo. Finalmente, se grafica el recorrido DFS como las aristas de retroceso y los ciclos encontrados.

Por otro lado, la complejidad en tiempo del recorrido DFS es de orden lineal sobre el número de aristas que tiene el grafo en HGm,n que es un factor constante del número de hexágonos en *Hr,t*, que es del orden *O(r*t)* (De Ita et al., 2023a). Mientras que la parte de mayor complejidad es el procesamiento de las aristas de retroceso que existen en el grafo, ya que lleva a aplicarse la regla de sustracción (3.2) lo que puede llevar a una complejidad exponencial en términos del número de aristas de retroceso que sean encontradas.

3.3.1. Trayectoria hamiltoniana

En el campo matemático de la teoría de grafos, un camino hamiltoniano se define como una secuencia de aristas consecutivas en un grafo en el que todos los vértices se visitan exactamente una vez. Si el último vértice visitado es adyacente al primero, se denomina ciclo hamiltoniano (Vegi Kalamar, 2023). El primer paso de nuestro método para contar conjuntos independientes en un grafo consiste en construir un camino hamiltoniano (Hc) sobre el grafo de entrada G. Este recorrido visita cada vértice del grafo una sola vez, clasificando cada arista como arista de árbol o arista de retroceso. Aunque encontrar un ciclo hamiltoniano en cualquier grafo es un problema NP-completo, en este caso, la restricción se relaja al considerar caminos en lugar de ciclos, evitando la necesidad de regresar al punto de partida. Esta relajación es más evidente en topologías de grafos

como las mallas, donde la construcción del camino Hamiltoniano Hc se convierte en un problema de complejidad lineal en tiempo. Por ejemplo, en mallas regulares, Hc puede seguir una ruta a través de las filas, alternando direcciones entre filas pares e impares.

3.3.2. Búsqueda en profundidad

El algoritmo de búsqueda en profundidad (DFS, por sus siglas en inglés) es conocido por ofrecer una metodología para explorar todos los vértices de un grafo sin pasar dos veces por un mismo vértice. La Figura 3.3 ilustra el recorrido *DFS(G, v)* comenzando desde el nodo 1. La estrategia consiste en: 1) Marcar el nodo *v*. 2) Si todos los nodos adyacentes a v ya están marcados, se finaliza el proceso; de lo contrario, se selecciona un nodo adyacente w a v que no esté marcado. 3) Se aplica el algoritmo *DFS(G, w)* al nodo seleccionado.

Nótese que en la Figura 3.3 las aristas verticales no marcadas corresponden a las aristas de retroceso que la búsqueda DFS va detectando y que a su vez, indican la formación de un ciclo hexagonal.

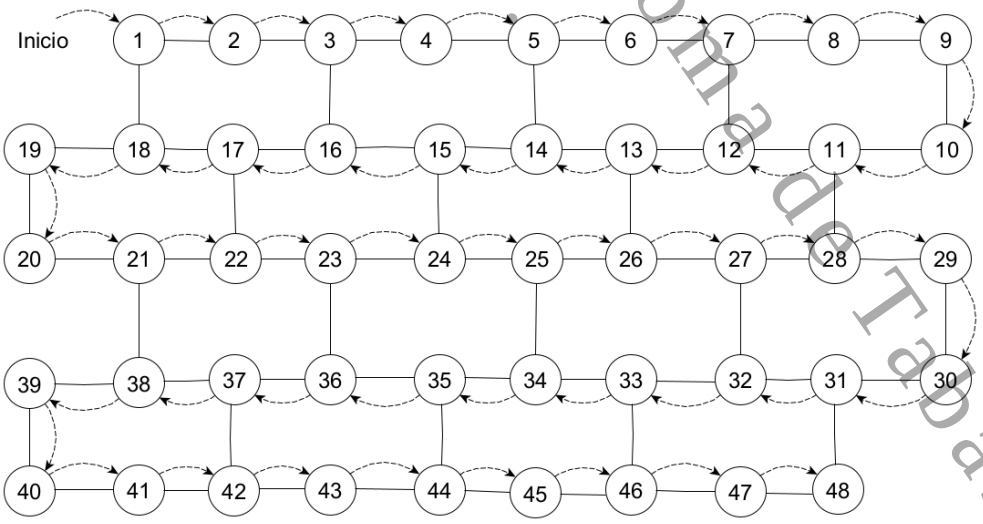


Figura 3.3. Recorrido DFS, el camino *(Hc)* queda como sigue (1,2,3,4,5,6,7,8,9,. . . ,47,48)

La función DFS inicializa listas para los $nodos\_visitados$ y las $aristas\_de\_retroceso$, así co-

mo un diccionario para la tabla de ruteo, que registra la relación entre los nodos hijos y nodos padres, es decir, $tabla\_de\_ruteo[nodo] = padre$. Seguidamente, realiza la búsqueda recursiva, marcando los nodos visitados y agregándolos a una lista enlazada. Al encontrar un vecino no visitado; se busca la arista $(nodo\_actual, vecino)$ en la matriz de adyacencia *M(m*n)+1* donde, *m* es el número de filas, *n* el número de columnas y el +1 indica que se añade una fila y una columna más para asignar las etiquetas de los vértices; y se actualiza la matriz de adyacencia, siendo *i* la posición de la fila y *j* la posición de la columna, estableciendo M[i][j]=0 y M[j][i]=0 para indicar que la arista ha sido visitada. Luego, se llama la función recursivamente para continuar la búsqueda. Al finalizar, la función DFS retorna las listas de nodos visitados y aristas de retroceso, junto con la matriz de adyacencia actualizada.

3.3.3. Cálculo de fibonacci en grafos

Contar el número de conjuntos independientes *i(HGm,n)* sobre la malla hexagonal *HGm,n* implica construir el camino hamiltoniano *Hc* de *HGm,n* según El-Basil (El-Basil, 1988c). Para llevar un conteo parcial del número de conjuntos independientes mientras se visitan los vértices de la malla *HGm,n* se asigna la carga computacional o de procesamiento a ese vértice específico dentro de la malla. La carga para el primer vértice que se visita se inicializa como: $(\alpha_i, \beta_i) = (1, 1)$. Para las siguientes aristas identificadas, se aplica una de dos reglas. La regla de recurrencia de Fibonacci (3.1) que es aplicada cuando la arista visitada corresponde a una arista de árbol. En otro caso, se aplicará la regla de sustracción (3.2) cuando se reconoce que la arista corresponde a una arista *frond* (o arista de retroceso, si nos referimos a búsquedas en profundidad). Si consideramos que v_i pertenece a la ruta P_n, y v_{i+1} es el siguiente vértice que visitará a través de una arista de árbol (v_i, v_{i+1}), entonces se aplica la siguiente ecuación de recurrencia para calcular la carga $(\alpha_{v_{i+1}}, \beta_{v_{i+1}})$ en función de la carga $(\alpha_{v_i}, \beta_{v_i})$:

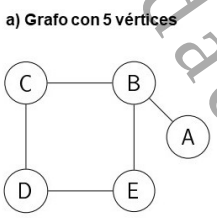
$$(\alpha_{v_{i+1}}, \beta_{v_{i+1}}): \quad \alpha_{v_{i+1}} = \alpha_{v_i} + \beta_{v_i}; \qquad \beta_{v_{i+1}} = \alpha_{v_i} \tag{3.1}$$

Llamamos al anterior par de ecuaciones, la recurrencia Fibonacci, porque cuando se aplican a un camino *Pn*, obtenemos la identidad reconocida como los números de Fibonacci: *i(Pn)=αvi+1+βvi+1=Fn+Fn+1=Fn+2*, donde *Fn* es el n-ésimo número de Fibonacci. Por otro lado, la regla que se aplica cuando se reconoce una arista *frond* en el recorrido es la llamada regla de sustracción, que se expresa como:

$$(\alpha_w, \beta_w)i = \quad (\alpha_w, \beta_w) - (0, \beta_{vw}) = \qquad (\alpha_w)\beta_w \tag{3.2}$$

Ilustremos con un ejemplo en la Figura 3.4 cómo calcular el índice MS del grafo *G*, que es lo mismo que el conteo de los conjuntos independientes (CI). El camino hamiltoniano utilizado para recorrer *G* es *A – B – C – D – E*. Al inicio del conteo de CI se abre un hilo primario *Lp* (un hilo es una secuencia de cargas formadas al visitar los vértices en un camino), y se aplica la regla (1) de Fibonacci a *i(G5):Lp:(1,1)→(2,1)→(3,2)→(5,3)→(8,5)*. Obsérvese que las cargas temporales $(\alpha_v i, \beta_v i)$ pueden almacenarse en el vértice *vi* y marcarse como visitadas. Para cualquier par de vértices $(v, w) \in S$, donde S es un conjunto independiente y *v,w* son adyacentes, y si la arista *(v,w)* se identifica como arista *frond* (en este caso *(E,B)* es la arista *frond*), se abre un hilo secundario *Ls* paralelo a *Lp*; es decir, *Ls : (0,1) → (1,0) → (1,1) → (2,1)*. Cuando la caminata visita el vértice *v* y *w* ya ha sido visitado, entonces se aplica la regla (3.2) de sustracción, que

permite calcular la carga del vértice v a partir de una carga del vértice *u*; y esto se hace para cada arista *frond* que se encuentre. En el caso del grafo G representado por la Figura 3.4, que consiste en un ciclo de longitud 4 y una arista adicional a uno de los vértices del cuadrado, se tiene que su número de conjuntos independientes es: $i(G) = |\{\{\},\{A\},\{B\},\{C\},\{D\},\{E\},\{A,C\},\{B,D\},\{C,E\},\{E,A\},\{D,A\},\{A,C,E\}\}| = 12\}$.



b) Conteo de conjuntos independientes

Hilo	A	B	C	D	E	i(G)	
	(α_1, β_1)	(α_2, β_2)	(α_3, β_3)	(α_4, β_4)	(α_5, β_5)		
Lp	(1,1) →	(2,1)→	(3,2) →	(5,3) →	(8,5)	(8,5)	
Ls		(0,1) →	(1,0)→	(1,1) →	(2,1)	-(0,1)	(8,4)=8+4=12

Figura 3.4. Conteo de los conjuntos independientes de G

3.4. Algoritmo conteo de caras en un grafo hexagonal

En teoría de grafos, las caras y los ciclos están relacionados, pero no son conceptos equivalentes. Las caras en un grafo se refieren a las regiones encerradas por ciclos, que son trayectorias cerradas del grafo. La relación entre caras y ciclos es crucial en problemas de incrustación de grafo (Teimoori Faal, 2023; Lozzo y Rutter, 2016). Según Bonsma y Breuer, 2012, los hexágonos en un grafo plano están formados por caras internas que miden 6 unidades. Los vértices internos tienen grado 3, mientras que los vértices del borde pueden tener grado 2 o 3. Ellos abordan el problema del conteo de conjuntos independientes en grafos circulares.

A continuación, presentamos el algoritmo general para el conteo de caras en una malla:

1. Crear la lista de adyacencia a partir de los datos de entrada.

2. Se crea el grafo *G* y la matriz de adyacencia *M(m*n)+1* a partir de la lista de adyacencia, marcando con 1 los vértices adyacentes. Las etiquetas de los vértices *(k)*

se asignan en las filas y columnas, respectivamente, como M[0][i]=k y M[i][0]=k.

3. Aplicar el algoritmo DFS(G) desde un nodo inicial. A partir del recorrido en profundidad, se crea una lista enlazada de nodos visitados y se marca el nodo visitado en la matriz de adyacencia con el valor 0.

4. A continuación, se realiza un recorrido en la matriz de adyacencia. Un par *(u, v)* de vértices adyacentes marcado con el valor 1 indica la presencia de una arista de retroceso. Por lo tanto, se busca el par de vértices *(u, v)* en la lista enlazada simple de nodos visitados y se enlazan en la lista.

5. Finalmente, se recorre la lista enlazada simple buscando ciclos. Al encontrar un ciclo, todos los vértices que forman parte de él se añaden a una lista de caras para conocer el total de caras de G. Durante el proceso DFS, se crea un árbol de expansión T que nos permitirá identificar las aristas de retroceso (e) o también conocido como *frond edge* , cuando se realice la revisión de la matriz de adyacencia se marca como visitada. Supongamos que el grafo G contiene ciclos. Una arista $e \in E(G) - E(TG)$ se conoce como arista de retroceso con respecto al DFS - búsqueda en profundidad. El camino en T entre los extremos de la arista e forman un ciclo simple Ce; este ciclo se denomina ciclo básico de G con respecto a T, como se muestra en la Figura 3.5 (De Ita Luna et al., 2024a).

En base, al camino hamiltoniano *Hc* el Algoritmo 1 (conteo de caras en grafo hexagonal), busca identificar aristas de retroceso *(e)*, iterando a través de la matriz de adyacencia *M*, es decir, esto implica buscar las conexiones (representada por un 1) entre los nodos adyacentes *u* y *v* de la arista *u,v*, donde *uv* es igual a *e* en *E(TG)* del grafo *G*. Si la conexión encontrada, no ha sido registrada previamente en la lista de aristas de retroceso, se añade el par *(u,v)* y se establece una enlace auxiliar *(aux)* entre los nodos para mantener la estructura de *E(G)*. Este proceso se repite recursivamente hasta completar el recorrido

de *Hc* obteniendo finalmente el total de caras en el grafo y los vértices que lo conforman. Una vez identificadas todas las caras y los nodos que la forman, iniciamos el cálculo de *i(G)* para contar los conjuntos independientes.

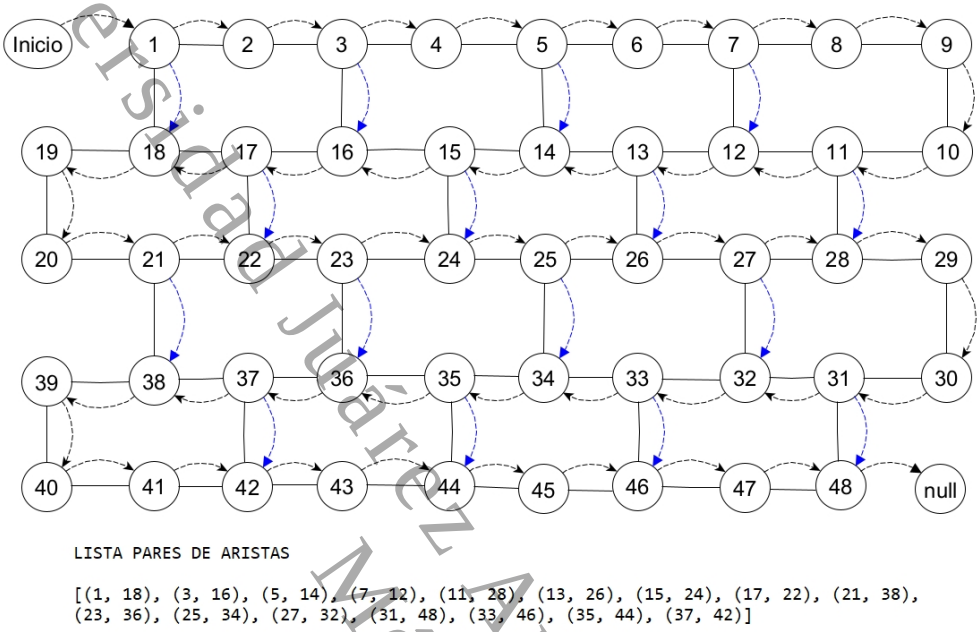


Figura 3.5. Conteo de caras, el ciclo simple lo forman los nodos *1,2,3,16,17,18* y la flecha de color azul es la *frond edge* (aristas de retroceso)

Algoritmo 1: Caras

```
Input: lista_arista
Output: lista_caras   Función Caras(lista_arista)
    if CABEZA == NULL then
        return [];
    lista_caras ← [];
    for ini, fin ∈ lista_arista do
        par_arista_retroceso ← [];
        ini_nodo ← buscar(ini);
        fin_nodo ← buscar(fin);
        actual ← ini_nodo;
        while actual ≠ NULL do
            par_arista_retroceso.agregar(actual.dato);
            if actual ≠ ini_nodo y actual.aux ≠ NULL then
                if actual.siguiente == fin_nodo y actual.aux ≠ NULL then
                    actual ← actual.siguiente;
                else
                    actual ← actual.aux;
            else
                if actual == fin_nodo then
                    actual ← NULL;
                else
                    actual ← actual.siguiente;
        lista_caras.agregar(par_arista_retroceso);
    return lista_caras;
```

Para el cierre de una cara aplicamos la regla de sustracción (3.2) con el par $(\alpha_m, \beta_m) = (0, \beta_m)$ de modo que la serie $((\alpha_1, \beta_1) = (0,1) \rightarrow (\alpha_2, \beta_2) = (1,0) \rightarrow (\alpha_3, \beta_3) = (1,1)... \rightarrow (\alpha_m, \beta_m) \rightarrow (0, \beta_m))$ calcula el valor para $i(Cs) = |S \in i(G') : v1 \in S \wedge vm \in S|$. La serie anterior corresponde a la serie que inicia considerando los números de Fibonacci: $F_0 = 0$ y $F_1 = 1$, $(F_0, F_1) \rightarrow (F_1, F_0) \rightarrow (F_2, F_1) \rightarrow ... \rightarrow (F_{m-1}, F_{m-2})$, $(\alpha_m, \beta_m) = (0, \beta_m)$; $i(Gc) = ((\alpha_m, \beta_m) - (0, \beta_a))m$.

Por lo que, el último par (α_m, β_m) del hilo secundario (C_{s+1}) es $(\alpha_m, \beta_m) = (0, \beta_m)$, donde

$vm, v0 \in E(G)$ representa la arista hacia atrás que cierra la cara C_{s+1} de G y se resta al último par del hilo primario de (C_s). Y así, de forma recursiva se van cerrando cada una de las caras conforme se van encontrando las aristas de retroceso de C_s hasta terminar el recorrido de *G*. En la Tabla 3.1, se muestra el cálculo de *i(G)* hasta el nodo (18). Al mismo tiempo, se observa que al encontrar la arista (11, 28), la cual indica el inicio de una nueva cara, se abren hilos, duplicando así el número de hilos activos.

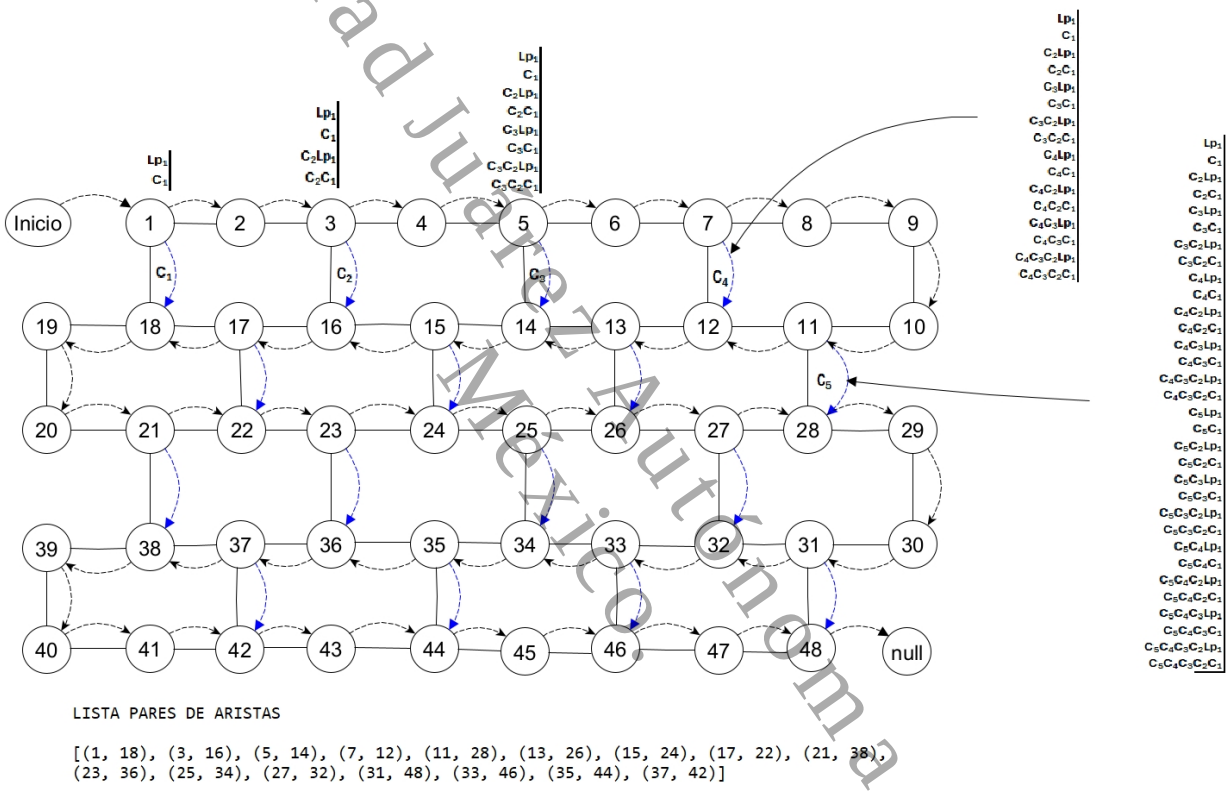


Figura 3.6. Recorrido para el conteo de los conjuntos independientes

Tabla 3.1. Cálculo del conteo de los conjuntos independientes 3.6

Fila	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Lp_1	1,1	2,1	3,2	5,3	8,5	13,8	21,13	34,21	55,34	89,55	144,89	233,144	0,39	233,105
C_1	0,1	1,0	1,1b	2,1	3,2	5,3	8,5	13,8	21,13	34,21	55,34	89,55	0,15	89,40
		C_2Lp_1	0,2b	2,0	2,2	4,2	6,4	10,6	16,10	26,16	42,26	68,42	0,12	68,30
		C_2C_1	0,1b	1,0	1,1	2,1	3,2	5,3	8,5	13,8	21,13	34,21	0,6	34,15
				C_3Lp_1	0,5	5,0	5,5	10,5	15,10	25,15	40,25	65,40	0,15	65,25
				C_3C_1	0,2	2,0	2,2	4,2	6,4	10,6	16,10	26,16	0,6	26,10
				$C_3C_2Lp_1$	0,2	2,0	2,2	4,2	6,4	10,6	16,10	26,16	0,6	26,10
				$C_3C_2C_1$	0,1	1,0	1,1	2,1	3,2	5,3	8,5	13,8	0,3	13,5
						C_4Lp_1	0,13	13,0	13,13	26,13	39,26	65,39	X	
						C_4C_1	0,5	5,0	5,5	10,5	15,10	25,15	X	
						$C_4C_2Lp_1$	0,4	4,0	4,4	8,4	12,8	20,12	X	
						$C_4C_2C_1$	0,2	2,0	2,2	4,2	6,4	10,6	X	
						$C_4C_3Lp_1$	0,5	5,0	5,5	10,5	15,10	25,15	X	
						$C_4C_3C_1$	0,2	2,0	2,2	4,2	6,4	10,6	X	
						$C_4C_3C_2Lp_1$	0,2	2,0	2,2	4,2	6,4	10,6	X	
						$C_4C_3C_2C_1$	0,1	1,0	1,1	2,1	3,2	5,3	X	

Tabla 3.2. ... Continuación del conteo de los conjuntos independientes

Fila	15	16	17	18	19	20	21	22	23	24	25	26
Lp_1	338,233	571,338	0,90	571,248	819,571	1390,819	0,228	1390,591	1981,1390	3371,1981	0,726	3371,1255
C_1	129,89	218,129	0,36	218,93	311,218	529,311	0,114	529,197	726,529	1255,726	X	$CI = 4626$
C_2Lp_1	98,68	166,98	0,36	166,62	228,166	394,228	X					
C_2C_1	49,34	83,49	0,18	83,31	114,83	197,114	X					
C_3Lp_1	90,65	155,90	X									
C_3C_1	36,26	62,36	X									
$C_3C_2Lp_1$	36,26	62,36	X									
$C_3C_2C_1$	18,13	31,18	X									

Calcular el número de conjuntos independientes de una malla es posible tomando como guía la trayectoria hamiltoniana *Hc*, aunque la complejidad temporal es de orden exponencial sobre el número máximo de aristas de *frond* en cualquier fila de la malla, dado el número de ciclos que se mantienen abiertos durante la trayectoria Hc. Las reglas de Fibonacci (3.1) y de sustracción (3.2) son suficientes para procesar una fila de mosaicos de una cuadrícula hexagonal.

3.5. Conclusión y trabajo futuro

En este artículo, se presenta un algoritmo para el cálculo del número de conjuntos independientes sobre mallas hexagonales. El algoritmo se basa en la construcción de un camino Hamiltoniano sobre los vértices de la malla, al mismo tiempo que se calcula de forma incremental el número de conjuntos independientes existentes en las mallas hexagonales. Nuestra propuesta para calcular *i(HGm,n)* sigue teniendo una complejidad temporal exponencial, pero no muestra el carácter combinatorio explosivo que tiene métodos clásicos como el método de la matriz de transferencia. Conocer el número de conjuntos independientes de un grafo molecular (como en el caso de compuestos aromáticos como los bencenos) permite determinar las temperaturas de ebullición de los compuestos, lo que permite modelar más y mejores compuestos, o generar nuevas moléculas con diferentes propiedades químicas.

Referencias

Bonsma, P., & Breuer, F. (2012). Counting hexagonal patches and independent sets in circle graphs. *Algorithmica*, *63*(3), 645-671. https://doi.org/10.1007/s00453-011-9561-y

De Ita, G., Bello, P., & Contreras, M. (2023a). A method for computing the Merrifield–Simmons index on benzenoid systems. *Match Communications in Mathematical and in Computer Chemistry*, *89*(1), 245-270. https://doi.org/10.46793/match.89-1.245I

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020c). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets. *Pattern Recognition*, 57-66. https://doi.org/10.1007/978-3-030-49076-8_6

De Ita Luna, G., Bello López, P., & Marcial-Romero, R. (2024a). Counting Rules for Computing the Number of Independent Sets of a Grid Graph. *Mathematics*, *12*(6). https://doi.org/10.3390/math12060922

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023c). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1). https://doi.org/10.61467/2007.1558.2023.v14i1.334

Deng, Z., Ding, J., Heal, K., & Tarokh, V. (2017). *The number of independent sets in hexagonal graphs*. https://doi.org/10.1109/ISIT.2017.8007062

El-Basil, S. (1988c). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29. https://doi.org/10.1007/BF01166466

Euler, R. (2005b). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Hosoya, H. (1973a). Topological Index and Fibonacci Numbers with Relation to Chemistry. *The Fibonacci Quarterly*, *11*(3), 255-265. https://doi.org/10.1080/00150517.1973.12430822

Lozzo, G. D., & Rutter, I. (2016). On the Complexity of Realizing Facial Cycles. https://doi.org/10.48550/arXiv.1607.02347

Merrifield, R. E., & Simmons, H. E. (1981a). Enumeration of structure-sensitive graphical subsets: Theory. *Proceedings of the National Academy of Sciences*, *78*(2), 692-695. https://doi.org/10.1073/pnas.78.2.692

Merrifield, R. E., & Simmons, H. E. (1989a). Topological methods in chemistry.

Teimoori Faal, H. (2023). Face-Counting Identities and Derivatives of Face Polynomials. *Turkish Journal of Mathematics and Computer Science*, *15*(2), 443-448. https://doi.org/10.47000/tjmcs.1120399

Vadhan, S. P. (2001b). The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, *31*(2), 398-427. https://doi.org/10.1137/S0097539797321602

Vegi Kalamar, A. (2023). Counting Traversing Hamiltonian Cycles in Tiled Graphs. *Mathematics*, *11*(12). https://doi.org/10.3390/math11122650

Capítulo 4

De lo exponencial a lo lineal: caminos de retroceso para el conteo de conjuntos independientes en polígonos

De lo Exponencial a lo lineal: Caminos de retroceso para el conteo de conjuntos independientes en polígonos

[1] Herlinda González Vázquez $231H18004@alumno.ujat.mx$, [1]Cristina López Ramírez $cristina.lopez@ujat.mx$, [2]Pedro Bello López $pedro.bello@correo.buap.mx$, [2]Guillermo De Ita Luna $deitaluna63@gmail.com$

Resumen: Este trabajo presenta una comparación entre dos enfoques con diferentes complejidades temporales para el conteo de conjuntos independientes en arreglos de estructuras poligonales. Por un lado, se analiza un método con complejidad exponencial $O(2_n)$, basado en un array de hexágonos en donde por cada ciclo que se abre y por cada arista de retroceso encontrada se duplican las líneas de cómputo para el conteo de los conjuntos independientes. Por otro lado, se propone un enfoque optimizado basado en la trayectoria hamiltoniana Hc, que combina las reglas de Fibonacci y de camino de retroceso, logrando una complejidad temporal de orden lineal $O(4*n)$. Este nuevo método no solo reduce drásticamente la complejidad computacional, sino que también demuestra su aplicabilidad práctica en el modelado y análisis de estructuras hexagonales, como los compuestos bencenoides, facilitando estimaciones precisas de propiedades moleculares y posibilitando el diseño de materiales innovadores.

Palabras clave: hexágonos, aristas de retroceso, caminos de retroceso.

Institución de adscripción: [1]Universidad Juárez Autónoma de Tabasco, [2]Benemérita Universidad Autónoma de Puebla.

4.1. Introducción

El índice Merrifeld-Simmons *i(G)* de un grafo *G* introducido por (R. E. Merrifield y Simmons, 1989b), es un índice topológico utilizado en química matemática. La química matemática utiliza la teoría de grafos para modelar estructuras químicas, donde las moléculas se representan como grafos, asignando a los vértices los átomos y a las aristas los enlaces. Este enfoque facilita el análisis de propiedades moleculares mediante índices topológicos, como *i(G)*, que funcionan como descriptores numéricos al correlacionar las estructuras químicas con sus propiedades físicas y químicas. En una publicación de American Journal of Mathematics, (Sylvester, 1878) introdujo el término chemicograph (graph = grafo) para referirse a la notación gráfica empleada en Química.

Así, *i(G)* representa el número de subconjuntos dentro del conjunto de vértices donde no existe adyacencia entre dos vértices, es decir, el número de vértices independientes en el conjunto *G*. En el ámbito de la teoría de grafos, este índice también se conoce como el número de Fibonacci de un grafo, *i(G)* (Sylvester, 1878; Yurttas Gunes et al., 2020).

El conteo de conjuntos independientes es importante debido a su relación con problemas de optimización, como la planificación y asignación de recursos. Además, permite determinar la temperatura a la cual los enlaces de ciertos compuestos químicos se rompen, así como sus puntos de ebullición (índice Merrifield-Simmons). Este proceso es esencial para identificar propiedades fisicoquímicas de compuestos con estructuras hexagonales, como las placas de grafeno y compuestos aromáticos, tales como los bencenos.

Los conjuntos independientes son objeto de estudio principalmente en el ámbito de la informática, ya que los grafos representan una herramienta poderosa para modelar problemas del mundo real. Un ejemplo de esto es la asignación eficiente de aulas y recursos en instituciones educativas, un desafío que requiere coordinar horarios, número de estudiantes, profesores y requisitos específicos de cada clase, asegurando que se cumplan

criterios de calidad en la distribución (Bracho et al., 2003b). La necesidad de una organización y asignación eficiente de recursos también se aplica en diversos campos, como la gestión de equipos de trabajo, la conectividad en redes de computadoras o la planificación de actividades, entre otros. En el ámbito de la Física, (Bonilla García, 2019b) aborda la medición de la entropía de gas lo que podría implicar que, en un sistema como los bencenos, las contribuciones entrópicas podrían influir en las interacciones moleculares y la estabilidad, donde las partículas no pueden compartir la misma arista; mientras que en Química los conjuntos independientes se emplean para modelar ciertos aspectos estructurales de los sistemas bencenoides (Gutman, 1982c). Por lo tanto, la teoría de grafos y la combinatoria matemática brindan soluciones aplicables en ámbitos que abarcan desde la educación hasta la ciencia, destacando su relevancia en una variedad de contextos.

4.2. Preliminares

Sea $G = (V,E)$ un grafo simple no dirigido con un conjunto de vértices V y aristas E. La conexión entre los vértices u y v se representa como uv. También usamos la notación u,v para denotar la arista uv. Dos vértices son adyacentes si están unidos por una misma arista, y no son adyacentes cuando no lo están.

Definamos $N(x) = y \in V : x,y \in E$ como la vecindad de $x \in V.N[x] = N(x) \cup x$ representa la vecindad cerrada de x. Usamos $|A|$ para denotar la cardinalidad de un conjunto A. Del mismo modo, el grado de un vértice x se denota como $\delta(x) = |N(x)|$, mientras que el grado del grafo G es $\Delta(G) = max\delta(x) : x \in V$. Un conjunto $S \subset V(G)$ de vértices de G es un conjunto independiente en G, si para cualquier par de vértices $u,v \in S$, se cumple que u,v no está en E. Se denota por $I(G) = S/S$ es un conjunto independiente en G - el conjunto de todos los conjuntos independientes de G, mientras que $i(G) = |I(G)|$ - es el número de conjuntos independientes de G. Sea $v \in V(G)$, se denota por $Iv(G) = S \in I(G) : v \in S, I - v(G) = S \in I(G) : v$ no está en S.

Por otro lado, se han desarrollado diversos métodos para el cómputo de conjuntos independientes en estructuras reticulares planas, los cuales simplifican el proceso de conteo mediante recorridos organizados por filas y columnas, o en el orden inverso (De Ita Luna et al., 2023d) . En este artículo, nuestra propuesta es un método para calcular el índice Merrifield-Simmons (MS) (R. E. Merrifield y Simmons, 1981b) en arreglo unidimensional de polígonos H[1...n] conformado por n hexágonos como se muestra en la Figura 4.1. Diferenciaremos dos procesos de cómputo de conjuntos independientes: conteo de CI (conjuntos independientes) con recorrido hamiltoniano Hc en H[1...n] con aristas de retroceso versus con caminos de retroceso.

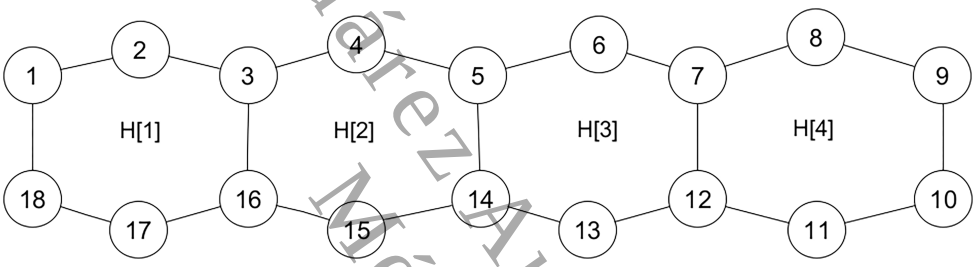


Figura 4.1. Arreglo unidimensional hexagonal H[1...n]

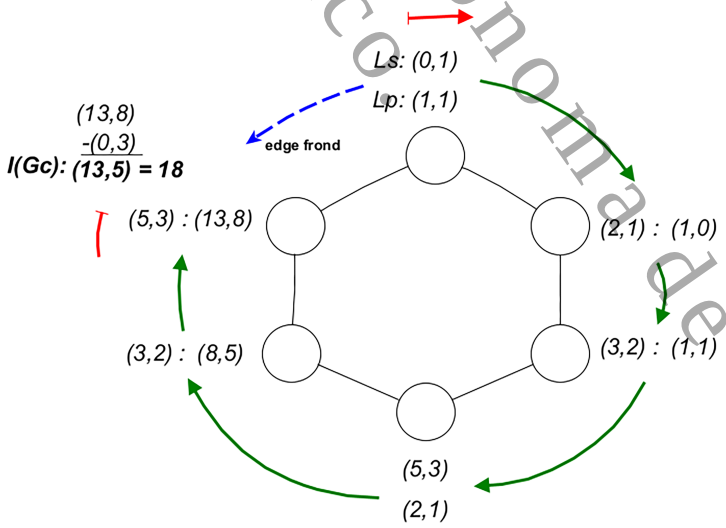


Figura 4.2. Conteo de CI en un hexágono.

El proceso de conteo de conjuntos independientes para un arreglo unidimensional de polígonos, inicia la recurrencia con el hilo principal (L_p), donde el par de valores inicial

es: $(\alpha_i, \beta_i) = (1,1)$. De igual forma, el hilo secundario (L_s) con $(\alpha_i, \beta_i) = (0,1)$, para $i = 1,...,m$; aplicando la regla de Fibonacci (1) en cada paso del camino, hasta encontrar el último par $(\alpha_m, \beta_m) = (0, \beta_m)$, formando la serie $((\alpha_1, \beta_1) = (0,1) \rightarrow (\alpha_2, \beta_2) = (1,0) \rightarrow (\alpha_3, \beta_3) = (1,1)... \rightarrow (\alpha_m, \beta_m) \rightarrow (0, \beta_m))$, que proporciona el valor de $|S \in I(G') : v1 \in S \wedge vm \in S|$. Esta serie corresponde a la sucesión de Fibonacci, definida como $F_0 = 0$ y $F_1 = 1$, $(F_0, F_1) \rightarrow (F_1, F_0) \rightarrow (F_2, F_1) \rightarrow ... \rightarrow (F_{m-1}, F_{m-2})$. Para calcular el cierre del ciclo, se aplica la regla de sustracción (2) por la arista de retroceso encontrada. El último par del hilo secundario $(\alpha_m, \beta_m) = (0, \beta_m)$ corresponde a la arista posterior $v_n, v_0 \in E(G)$, que cierra el ciclo. Este valor se resta del último par del hilo principal. Así, $((\alpha_m, \beta_m) - (0, \beta_m)) = ((\alpha_6, \beta_6) - (0, \beta_6)) = (13,8) - (0,3) = (13,5)$. Por tanto, $I(G_c) = 13 + 5 = 18$, lo que indica que hay 18 conjuntos independientes en total. Este resultado se presenta con detalle en la Figura 4.2, donde se ilustra el conteo de conjuntos independientes del grafo G. El inicio del conteo se indica con una flecha ($\rightarrow$) (De Ita et al., 2020c).

4.3. Método de cálculo de conjuntos independientes con aristas de retroceso

En la Figura 4.3 se muestra un recorrido hamiltoniano Hc en el arreglo unidimensional de hexágonos, entiéndase por recorrido hamiltoniano, que la arista es visitada solo 1 vez. Durante la trayectoria del recorrido por cada ciclo encontrado, marcamos una arista de retroceso, misma que por cada una de ellas se duplican las líneas de proceso del cómputo de los conjuntos independientes, más adelante en la Figura 4.4 se ilustra un ejemplo.

El cómputo de los conjuntos independientes en un arreglo unidimensional se puede realizar siguiendo la trayectoria hamiltoniana (Hc). Sin embargo, la complejidad temporal crece de forma exponencial con respecto al número máximo de aristas de retroceso en

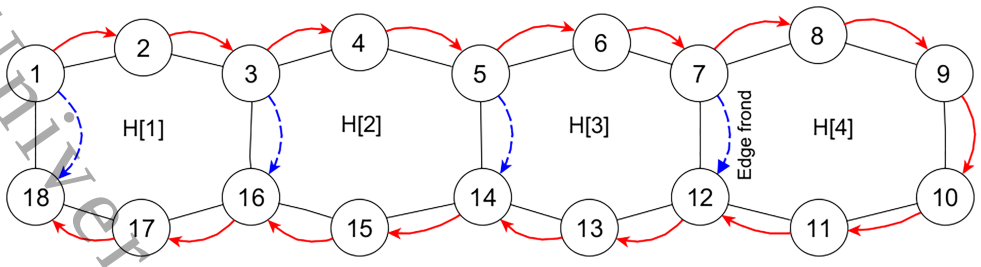


Figura 4.3. Recorrido hamiltoniano Hc en H[1...n] con aristas de retroceso

la fila, debido a la cantidad de ciclos que permanecen abiertos a lo largo de dicha trayectoria. Para procesar una fila de mosaicos en una cuadrícula hexagonal, son suficientes las reglas de Fibonacci (4.1) y de sustracción (4.2).

$$(\alpha_{v_{i+1}}, \beta_{v_{i+1}}): \quad \alpha_{v_{i+1}} = \alpha_{v_i} + \beta_{v_i}; \qquad \beta_{v_{i+1}} = \alpha_{v_i} \tag{4.1}$$

$$(\alpha_w, \beta_w)_i = (\alpha_w, \beta_w) - (0, \beta_{vw}) = \quad (\alpha_w, \beta_w - \beta_{vw}) \tag{4.2}$$

El conteo de los conjuntos independientes en un arreglo unidimensional hexagonal se ilustra en la Figura 4.4 , en este caso el arreglo está formado por 4 hexágonos y el conteo se inicia en el vértice 1. En este proceso, se inicia un hilo principal (Lp) con el par de valores iniciales $(\alpha_i, \beta_i) = (1,1)$, mientras que el hilo secundario (Ls) señala el comienzo de cada una de las caras con la secuencia $(\alpha_i, \beta_i) = (0,1)$, donde $i = 1, ..., m$. Ambos hilos, (Lp) y (Ls), avanzan simultáneamente en el cálculo siguiendo la regla de recurrencia de Fibonacci de la ecuación (4.1). Al detectar el inicio de un nuevo hexágono, se generan nuevos hilos $(Lph(1...n))$ y $(Lsh(1...n))$, lo que duplica el número de hilos activos de forma paralela por cada bifurcación encontrada, continuando con la recurrencia de Fibonacci.

Al identificar una arista *frond* o de retroceso, se procede al cierre del hexágono (Ls) correspondiente mediante la aplicación de la regla de sustracción de la ecuación (4.2), lo que permite actualizar la carga de los hilos secundarios y reducir el número de hilos activos. Para cerrar un hexágono en el arreglo, el último par (α_m, β_m) del hilo secundario $(Lsh(1..n))$, se define como $(\alpha_m, \beta_m) = (0, \beta_m)$, donde la arista $v_m, v_0 \in E(G)$ representa la arista de retroceso que cierra el hexágono $(Lsh(1..n))$, en el arreglo G. Este par se resta del último par del hilo primario de $(Lsh(1..n))$. Este proceso se repite de manera recursiva, cerrando cada hexágono del arreglo unidimensional a medida que se encuentran las aristas de retroceso de (Ls), hasta completar el recorrido del grafo. En la Tabla 4.1 y 4.2 se muestra el cómputo de $i(G)$ hasta el vértice 18. Además, se puede observar que, por cada inicio de un nuevo hexágono, se generan nuevos hilos, duplicando el número de hilos activos durante el proceso de cálculo.

El cálculo del número de conjuntos independientes en un arreglo unidimensional de hexágonos se puede realizar utilizando como referencia la trayectoria hamiltoniana Hc. No obstante, la complejidad temporal de este proceso es exponencial, ya que depende del número máximo de aristas de retroceso, debido a los ciclos que permanecen abiertos durante el recorrido. Para procesar un arreglo hexagonal, la aplicación de las reglas de Fibonacci (4.1) y de sustracción (4.2) resulta suficiente. Aunque calcular $i(G)$ sigue teniendo una complejidad temporal exponencial, este enfoque, a diferencia de los métodos clásicos como el de la matriz de transferencia, no presenta un crecimiento combinatorio tan explosivo. Determinar el número de conjuntos independientes en un grafo molecular, como en el caso de los compuestos aromáticos como los bencenos, ayuda a estimar sus temperaturas de ebullición. Esta información permite no solo modelar compuestos más eficientes, sino también diseñar nuevas moléculas con diferentes propiedades químicas.

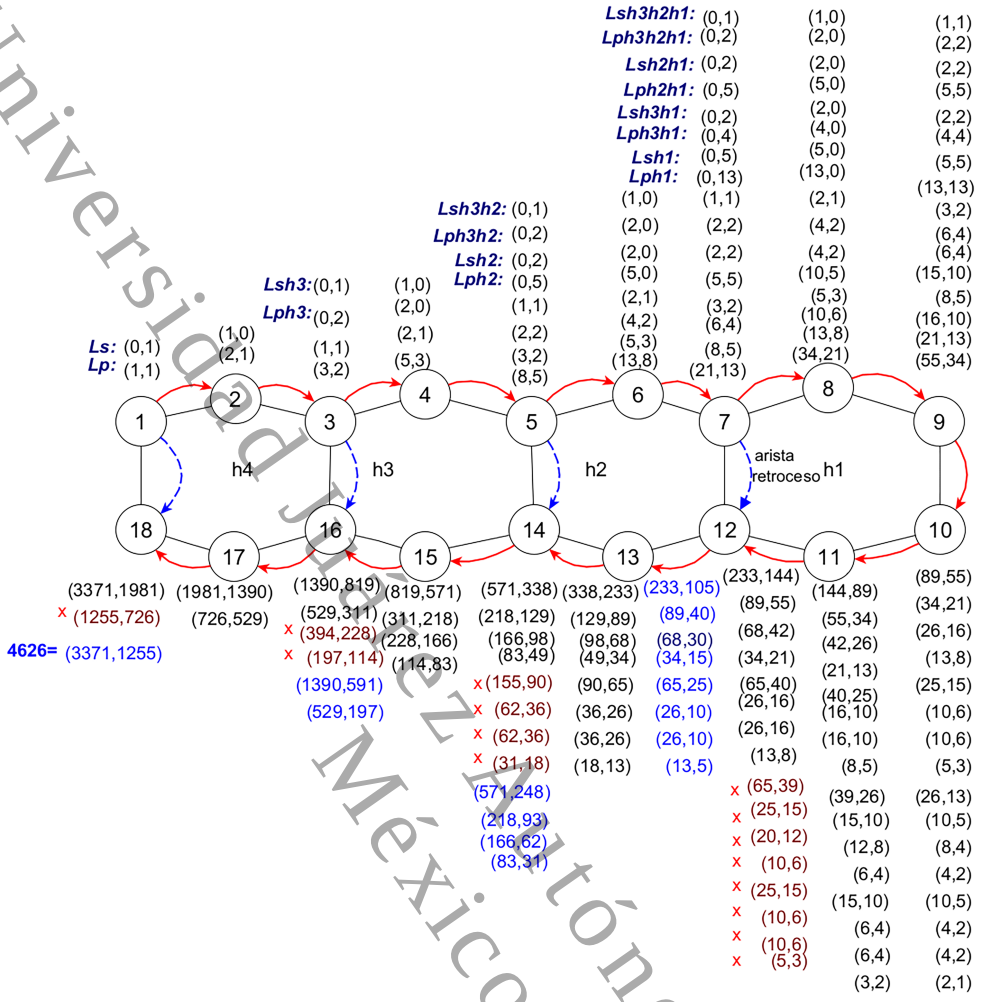


Figura 4.4. Conteo de conjuntos independientes con orden exponencial

Tabla 4.1. Cálculo del conteo de los conjuntos independientes 4.4

Fila	1	2	3	4	5	6	7	8	9	10	11	12	13	14
*Lph*1	1,1	2,1	3,2	5,3	8,5	13,8	21,13	34,21	55,34	89,55	144,89	233,144	0,39	233,105
*Lsh*1	0,1	1,0	1,1b	2,1	3,2	5,3	8,5	13,8	21,13	34,21	55,34	89,55	0,15	89,40
		*Lph*2	0,2b	2,0	2,2	4,2	6,4	10,6	16,10	26,16	42,26	68,42	0,12	68,30
		*Lsh*2	0,1b	1,0	1,1	2,1	3,2	5,3	8,5	13,8	21,13	34,21	0,6	34,15
				*Lph*3	0,5	5,0	5,5	10,5	15,10	25,15	40,25	65,40	0,15	65,25
				*Lsh*3	0,2	2,0	2,2	4,2	6,4	10,6	16,10	26,16	0,6	26,10
				*Lph*2*h*3	0,2	2,0	2,2	4,2	6,4	10,6	16,10	26,16	0,6	26,10
				*Lsh*2*h*3	0,1	1,0	1,1	2,1	3,2	5,3	8,5	13,8	0,3	13,5
						*Lph*1*h*4	0,13	13,0	13,13	26,13	39,26	65,39	*X*	
						*Lsh*1*h*4	0,5	5,0	5,5	10,5	15,10	25,15	*X*	
						*Lph*2*h*4	0,4	4,0	4,4	8,4	12,8	20,12	*X*	
						*Lsh*2*h*4	0,2	2,0	2,2	4,2	6,4	10,6	*X*	
						*Lph*3*h*4	0,5	5,0	5,5	10,5	15,10	25,15	*X*	
						*Lsh*3*h*4	0,2	2,0	2,2	4,2	6,4	10,6	*X*	
						*Lph*2*h*3*h*4	0,2	2,0	2,2	4,2	6,4	10,6	*X*	
						*Lsh*2*h*3*h*4	0,1	1,0	1,1	2,1	3,2	5,3	*X*	

Tabla 4.2. ... Continuación del conteo de los conjuntos independientes 4.4

Fila	15	16	17	18	19	20	21	22	23	24	25	26
*Lph*1	338,233	571,338	0,90	571,248	819,571	1390,819	0,228	1390,591	1981,1390	3371,1981	0,726	3371,1255
*Lsh*1	129,89	218,129	0,36	218,93	311,218	529,311	0,114	529,197	726,529	1255,726	X	$CI = 4626$
*Lph*2	98,68	166,98	0,36	166,62	228,166	394,228	X					
*Lsh*2	49,34	83,49	0,18	83,31	114,83	197,114	X					
*Lph*3	90,65	155,90	X									
*Lsh*3	36,26	62,36	X									
*Lph*2*h*3	36,26	62,36	X									
*Lsh*2*h*3	18,13	31,18	X									

4.4. Método de cálculo de conjuntos independientes con camino de retroceso

La Figura 4.5 muestra cómo se procesan los caminos de retroceso identificados a medida que avanza el recorrido, el procedimiento comienza con un recorrido hamiltoniano Hc, ejecutado en tiempo lineal sobre el arreglo unidimensional hexagonal, durante el cual se realiza simultáneamente el cómputo del número de conjuntos independientes y las dos aristas del camino de retroceso. Este enfoque permite reducir la complejidad temporal en comparación con el método de matriz de transferencia (Euler, 2005c), aplicado a grafos reticulares (Hosoya, 1973c) y supera la complejidad exponencial asociada con el número máximo de aristas de retroceso en cualquier fila de la malla, considerando los ciclos abiertos a lo largo de la trayectoria Hc en una matriz hexagonal regular HGm,n (De Ita Luna et al., 2024b).

Sin embargo, para completar el cálculo y cerrar el grafo, es necesario adicionar una operación de resta correspondiente al camino de retroceso. Esta resta permite ajustar adecuadamente el cálculo y asegurar la precisión en la contabilización final de los conjuntos independientes. Como resultado, el método opera con una complejidad de orden lineal O(4*n), - siendo n el número de vértices a visitar, multiplicado por 4 que son las líneas en ejecución- lo que representa una solución más manejable y eficiente para este tipo de

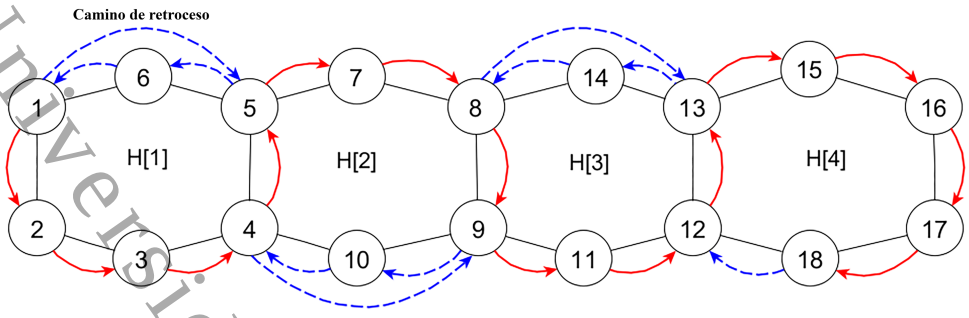


Figura 4.5. Recorrido hamiltoniano Hc en H[1...n] con caminos de retroceso

problemas en estructuras reticulares hexagonales.

Para iniciar el conteo del índice de Merrifield-Simmons, tomamos en cuenta las dos reglas anteriores (4.1) Regla Fibonacci, (4.2) Regla de arista de retroceso y se agrega una nueva regla: la Regla de camino de retroceso (4.3) .

$$(\alpha_i, \beta_i): \quad (2\alpha_i, \beta_i): 2\alpha_i = 2(\alpha_i + \beta_i); \qquad \beta_i = \beta_i \tag{4.3}$$

En este método, el cálculo del número de conjuntos independientes se simplifica al enfocarse en dos aspectos principales: las aristas del camino de retroceso y el proceso necesario para regresar al punto de inicio de dicho camino. Este enfoque se fundamenta en reglas derivadas del método del árbol binario para la construcción de la regla del camino de retroceso, como se muestra en la Figura 4.6. En los árboles binarios, se pueden definir estados basados en la inclusión o exclusión de un nodo en el conjunto independiente, optimizando así el cálculo del número total de conjuntos independientes (Aleid et al., 2015; Law, 2010). El proceso inicia con la construcción del árbol binario en $-x1$ y $x1$, que representan el par inicial $(\alpha_1, \beta_1) = (1, 1)$. Este par establece el punto de partida para el cómputo de los conjuntos independientes en Lp del fragmento del grafo mostrado

en la Figura 4.6. Dicho inicio significa que el conteo arranca con el conjunto vacío y un único vértice. Este árbol binario facilita la construcción de combinaciones de pares, considerando que un conjunto independiente está compuesto por vértices no adyacentes. El símbolo negativo (-) se utiliza para indicar que un vértice no puede estar presente en un conjunto. Por ejemplo, al llegar al vértice $x5$, el conteo en la línea principal Lp con $(\alpha_5, \beta_5) = (8,5)$ señala que 8 pares de nodos no pueden formar parte del conjunto, lo que se denota como $-x5$, mientras que 5 pares de nodos sí pueden incluirse, denotado como $x5$. Este método asegura un proceso sistemático y eficiente para enumerar y calcular las combinaciones de los conjuntos independientes en el grafo analizado.

La implementación de estas reglas permite una drástica reducción en el número de líneas de cómputo necesarias, disminuyendo considerablemente el orden de complejidad en comparación con métodos tradicionales. Este procedimiento se caracteriza por seguir una trayectoria lineal que cubre un renglón de polígonos, como se ilustra en la Figura 4.5, lo que elimina la necesidad de manejar la complejidad exponencial observada en enfoques previos, como el presentado en la Figura 4.4. En el peor de los escenarios, este método requiere únicamente la construcción de 4 líneas de proceso, evitando el crecimiento exponencial a 8, 16, 32, etc., característico de otros métodos menos eficientes. Este ahorro computacional se logra al restringir las operaciones a lo estrictamente necesario, lo que convierte a este enfoque en una alternativa práctica y eficiente para el conteo de conjuntos independientes en grafos hexagonales.

En la Figura 4.7, utilizamos una sección del grafo presentado en la Figura 4.5 para ilustrar el conteo de conjuntos independientes mediante la aplicación de la regla del camino de retroceso (4.3). Iniciamos el cómputo del número de conjuntos independientes del grafo G, a partir del vértice 1. Este punto de inicio marca el comienzo del recorrido, permitiendo aplicar las reglas establecidas para identificar y contar los conjuntos independientes a medida que se exploran los vértices y aristas del grafo. El conteo se inicia con un hilo

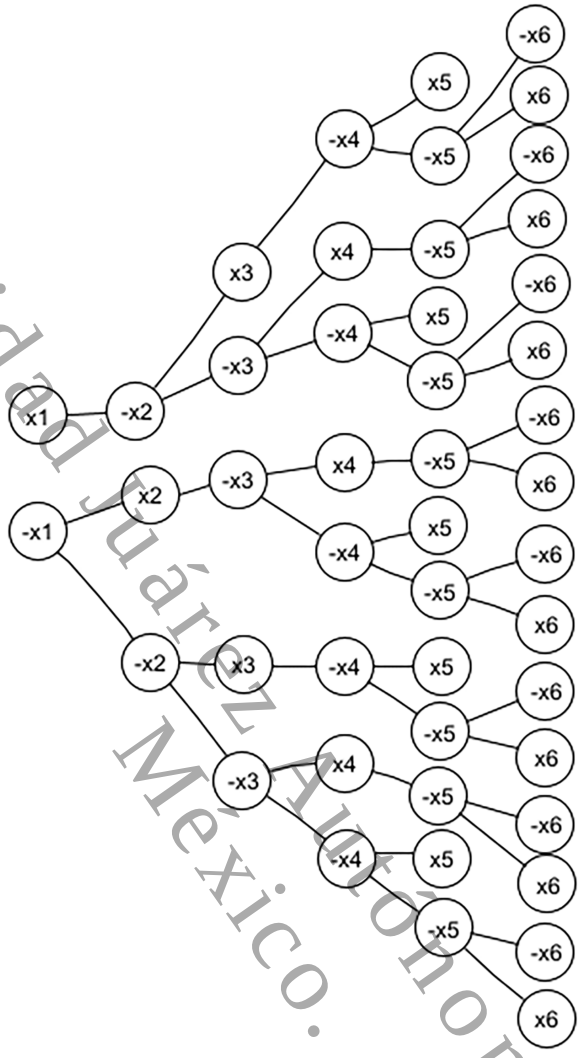


Figura 4.6. Árbol binario para la construcción de la regla de camino de retroceso

principal (Lp) comenzando con el par de valores con $(\alpha_i, \beta_i) = (1,1)$, y el hilo secundario $(Lh1)$ marcando el inicio del hexágono $h1$ con la secuencia $(\alpha_i, \beta_i) = (0,1)$, donde $i = 1,...,n$. Ambos hilos (Lp) y $(Lh1)$ avanzan de manera paralela aplicando la regla de recurrencia de Fibonacci (4.1) sobre las aristas Fibonacci, Este proceso continúa hasta que se encuentra una arista de retroceso que da inicio a un nuevo hexágono $h_{2..n}$. En este punto, se generan dos nuevas líneas de ejecución $(Lph1)$ y $(Lh1h2)$ iniciando cada línea con el par $(0, \beta_i)$ derivadas de las dos líneas principales anteriores. El cómputo continua con la aplicación de la regla de recurrencia hasta llegar al vértice donde comienza

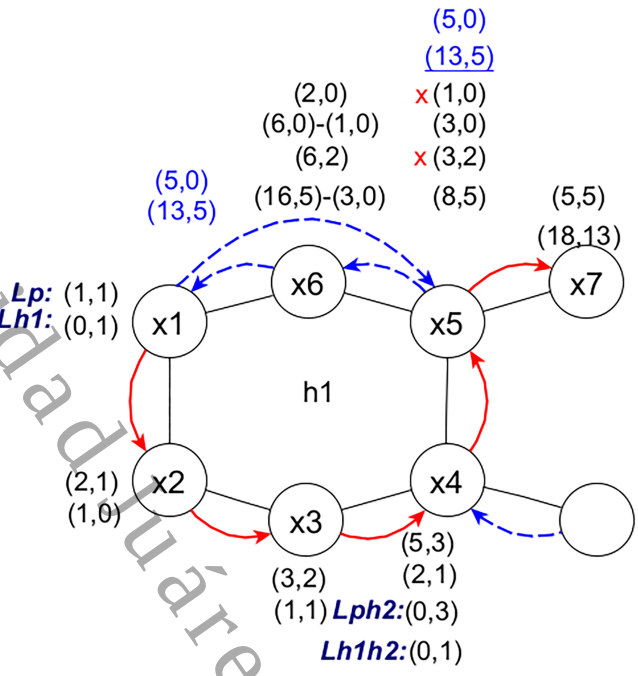


Figura 4.7. Conteo de CI con camino de retroceso

el camino de retroceso que cierra el hexágono h1 y entonces aplicar la regla de camino de retroceso (4.3), cerrando el hexágono y regresando el control de conteo al nodo en el que ha iniciado el camino; para luego continuar el recorrido al siguiente hexágono que forma parte del arreglo unidimensional de polígonos, y así mismo, repetir nuevamente la aplicación de las mismas reglas; ya sea la regla de Fibonacci o la regla de camino de retroceso, según sea el caso durante el recorrido del grafo G hasta finalizar, los resultados obtenidos se detallan en la Tabla 4.3.

El hilo principal (Lp) comienza $Lp : ((\alpha_1,\beta_1) = (1,1) \rightarrow (2\alpha_2,\beta_2) = (2,1) \rightarrow (3\alpha_3,2\beta_3) = (3,2) \rightarrow (5\alpha_4,3\beta_4) = (5,3) \rightarrow (8\alpha_5,5\beta_5) = (8,5) \rightarrow ... \rightarrow (F\alpha_+1\alpha, F\alpha\beta) = (\alpha_m,\beta_m))$. Mientras tanto, el hilo secundario (Ls) calcula el valor de $\|S \in i(G') : v_1 \in S \wedge v_m \in S\|$ en G. $Ls : (\alpha_s,\beta_s) = (0,1) \rightarrow (1\alpha_s+\beta_s,\alpha_s) = (1,0) \rightarrow (2\alpha_s+\beta_s,\alpha_s+\beta_s) = (1,1) \rightarrow (3\alpha_s+2\beta_s,2\alpha_s+\beta_s) = (2,1) \rightarrow (5\alpha_s+3\beta_s,3\alpha_s+2\beta_s) = (3,2) \rightarrow ... \rightarrow (F\alpha_+1\alpha_-F\alpha_-1\alpha_s, F\alpha_\beta - F\alpha_-2\beta) = (\alpha_m,0))$, para cerrar los ciclos abiertos en cada polígono. Finalmente, la serie general del proceso para contar conjuntos independientes considerando un camino de retroceso de longitud

Tabla 4.3. Conteo de conjuntos independientes con la regla de caminos de retroceso 4.7

Hilo	x1	x2	x3	x4	x5	x6	x5	x1 →x5	x7
	(α_1, β_1)	(α_2, β_2)	(α_3, β_3)	(α_4, β_4)	(α_5, β_5)	$(2\alpha_5, \beta_5)$	$(\alpha_5, 0)$		(α_7, β_7)
Lp	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)	(16,5)	-(3,0)	(13,5)	(18,13)
Lh_1	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)	(6,2) X			
			Lph_2	(0,3)	(3,0)	(6,0)	-(1,0)	(5,0)	(5,5)
			Lh_1h_2	(0,1)	(1,0)	(2,0) X			

k es $(\alpha, \beta - \beta_s) \to (2\alpha - \alpha_s, \beta) \to (3\alpha - \alpha_s, 2\beta - \beta_s) \to (5\alpha - 2\alpha_s, 3\beta - \beta_s) \to (8\alpha - 3\alpha_s, 5\beta - 2\beta_s) \ldots \to (F_{k+1}\alpha - F_{k-1}\alpha_s, F_k\beta - F_{k-2}\beta_s)$ donde $k = 1, 2, 3, \ldots, n$ aristas de retroceso. Ver en Figura 4.8 y Tabla 4.4 el conteo de los conjuntos independientes aplicando la regla de caminos de retroceso.

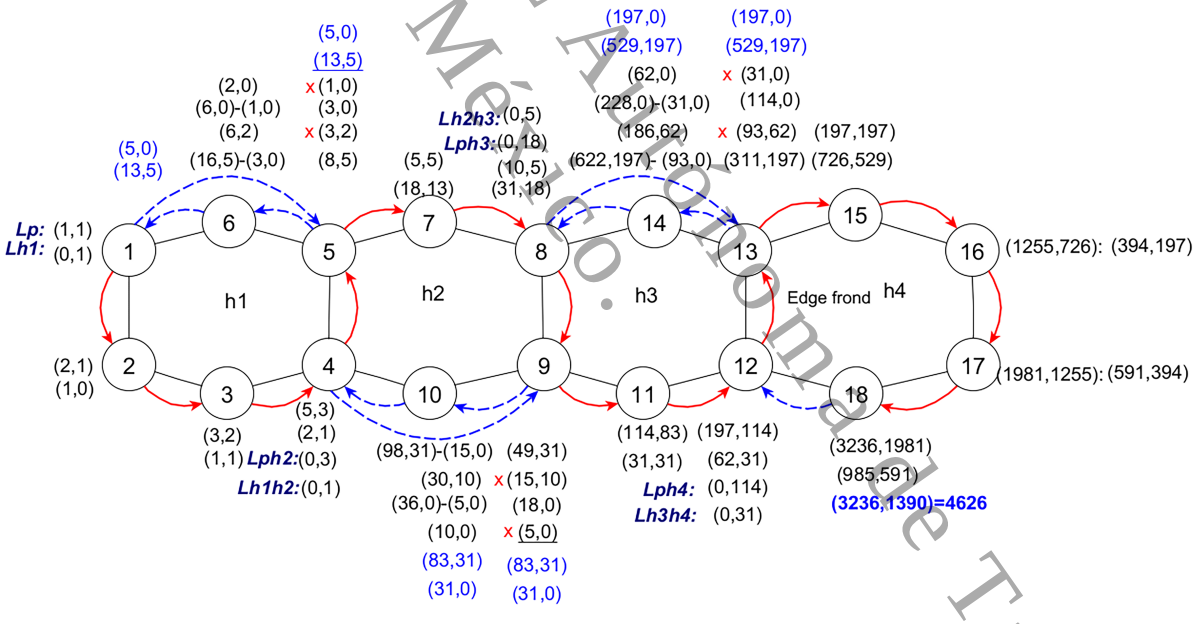


Figura 4.8. Conteo de conjuntos independientes de orden lineal

Tabla 4.4. Conteo de conjuntos independientes con caminos de retroceso 4.8

Fila	1	2	3	4	5	$5\to6\to1\to5$		7	8	9	$9\to10\to4\to9$	
Lp	$1,1$	$2,1$	$3,2$	$5,3$	$8,5$	$16-3,5$	$13,5$	$18,13$	$31,18$	$49,31$	$98-15,31$	$83,31$
Lph_1	$0,1$	$1,0$	$1,1$	$2,1$	$3,2$	$6,2$	X					
			Lph_2	$0,3$	$3,0$	$6-1,0$	$5,0$	$5,5$	$10,5$	$15,10$	$30,10$	X
			Lph_1h_2	$0,1$	$1,0$	$2,0$	X					
								Lph_3	$0,18$	$18,0$	$36-5,0$	$31,0$
								Lh_2h_3	$0,5$	$5,0$	$10,0$	X
Fila	11	12	13	$13\to14\to8\to13$	15	16	17	18	19	20		
Lp	$114,83$	$197,114$	$311,197$	$622-93,197$	$529,197$	$726,529$	$1255,726$	$1981,1255$	$3236,1981$	$3236,1390$	$CI=4626$	
Lph_3	$31,31$	$62,31$	$93,62$	$186,62$	X							
	Lph_4	$0,114$	$114,(228-31,0)$	$228-31,0$	$197,0$	$197,197$	$394,197$	$591,394$	$985,591$			
	Lh_3h_4	$0,31$	$31,(62,0)$	$62,0$	X							

Esto evidencia el gran potencial de este método, no solo para simplificar cálculos complejos, sino también para adaptarse a diversas aplicaciones en la teoría de grafos y en otros campos relacionados, como la Química. En particular, su utilidad se destaca al permitir determinar el número de conjuntos independientes en un compuesto con estructuras específicas, como los sistemas bencenoides. Este tipo de análisis resulta crucial en el estudio de compuestos aromáticos, ya que los conjuntos independientes están directamente relacionados con propiedades fundamentales de estas moléculas, como la estabilidad de sus enlaces y las condiciones necesarias para su descomposición. En el caso de los compuestos aromáticos, como el benceno, conocer el número de conjuntos independientes brinda información clave sobre cómo se comportan los enlaces moleculares bajo diferentes condiciones físicas y químicas. Por ejemplo, este análisis permite establecer la relación entre los conjuntos independientes y las temperaturas críticas a las que se rompen los enlaces químicos, lo que a su vez está vinculado a las temperaturas de descomposición y ebullición de estos compuestos. Esto es especialmente relevante en contextos industriales y científicos donde se busca optimizar procesos relacionados con materiales químicos, como la creación de nuevos compuestos o el diseño de materiales más eficientes.

El conteo de caras en grafos hexagonales está directamente relacionado con la identi-

ficación de ciclos cerrados que delimitan las regiones internas del grafo, denominadas caras. Estos ciclos son esenciales para resolver problemas de incrustación de grafos y analizar mallas hexagonales. En este caso de estudio, se considera un arreglo unidimensional de hexágonos, donde cada hexágono se define como una cara o ciclo. Por lo tanto, para llevar a cabo un recorrido en orden lineal, es imprescindible identificar primero las caras que conforman el arreglo de polígonos.

A continuación, se detalla el algoritmo para el conteo de conjuntos independientes en un arreglo de hexágonos utilizando caminos de retroceso, basado en el método de cálculo previamente descrito:

El algoritmo general sigue los siguientes pasos: 1°. Construcción de la lista de adyacencia: Se genera la lista de adyacencia a partir de los datos de entrada para representar las conexiones entre los vértices del grafo. 2°. Creación del grafo y matriz de adyacencia: Se construye el grafo G y su matriz de adyacencia M, asignando etiquetas a los vértices en filas y columnas, y marcando con 1 las conexiones adyacentes. 3°. Aplicación de DFS (Depth-First Search): Desde un nodo inicial, se realiza un recorrido en profundidad. Los nodos visitados se registran en una lista enlazada y se marcan como 0 en la matriz de adyacencia. 4°. Identificación de los caminos de retroceso: Durante el recorrido de la matriz, los pares de vértices adyacentes marcados con un 1 identifican las aristas de retroceso. Estas aristas se enlazan en la lista enlazada simple de nodos visitados, y el vértice que precede al par de la arista de retroceso se incluye como un elemento adicional en el camino de retroceso. Este elemento adicional, agregado como parte del camino de retroceso, también se enlaza en la lista enlazada para su recorrido posterior. 5°. Detección de Ciclos y Conteo de Caras: Se analiza la lista enlazada para identificar ciclos cerrados. Los vértices que forman parte de cada ciclo se agregan a una lista de caras, permitiendo determinar el número total de caras del grafo G. 6°. Recorrido del grafo para el conteo de los conjuntos independientes: Se construye un camino siguiendo la lista

enlazada que se construyó en el paso 3°, utilizando punteros para recorrer los nodos del grafo.

Los pasos 1°, 2° y 3° se describen en el algoritmo de detección de caras en grafos de mallas hexagonales desarrollado por Vázquez et al., 2024. Sin embargo, en este caso, el algoritmo de detección de ciclos y conteo de caras está diseñado específicamente para un arreglo unidimensional de hexágonos, lo que genera diferencias entre ambos pseudocódigos. El paso 4° se detalla en el Algoritmo 1: Caminos de Retroceso; el paso 5° se aborda en el Algoritmo 2: Conteo de Caras en el Arreglo Unidimensional Hexagonal, y finalmente, el paso 6° se desarrolla en el Algoritmo 3: Recorrido de Orden Lineal para el Conteo de los Conjuntos Independientes.

Algoritmo 2: Caminos de retroceso

```
Input: matrix, list
Output: retro_path
Input: matrix
Output: list   Function backward_path(matrix, list):
    retro_path ← empty_list;
    list_edge ← empty_list;
    for i ∈ filas de matrix do
        for j ∈ columnas de matrix[i] do
            if matrix[i][j] == 1 then
                start ← list.search(i);
                end ← list.search(j);
                if start ≠ none and end ≠ none then
                    if (start.data, end.data) ∉ list_edge and (end.data, start.data) ∉ list_edge
                     then
                        list_edge.add((end.data, start.data));
                        previous.aux ← end.next;
                        end.aux ← start;
                        start.aux ← previous;
                        retro_path.add((previous.data, end.data, start.data));
    return retro_path;
```

Algoritmo 3: Conteo de caras en matriz hexagonal

```
Function caras(list_edges):
    Input  : list_edges (lista de aristas)
    Output: list_faces (lista de caras)
    if head of list is null then
        return empty_list;
    global list_faces;
    list_faces ← empty_list;
    foreach (end, start) ∈ list_edges do
        start_node ← find_node(start);
        end_node ← find_node(end);
        current ← start_node;
        fin2 ← null;
        face ← empty_list;
        while current ≠ null do
            add current.data to face;
            if fin2 == null and not (current.next == end_node and current.aux ==
              end_node.next) and not (current == start_node.next) then
                current ← current.next;
            else if current.aux == null then
                current ← current.next;
            else if current.next == end_node then
                fin2 ← current;
                current ← current.next;
            else if current == start_node.next then
                current ← current.aux;
            else
                current ← null;
        add face to list_faces;
    return list_faces;
```

Algoritmo 4: Camino de orden lineal para el conteo de conjuntos independientes

```
Input: list_edges
Output: path_ham
Function route(list_edges):
    if head_of_list is null then
        return empty_list;
    end
    global path_ham;
    start_node ← head_of_list;
    end_node ← null;
    current ← ini_node;
    path_ham ← empty_list;
    i ← 0;
    while current is not null and i < length(retro_path) do
        add current.data to path_ham;
        if current.data ≠ retro_path[i][0] and end_node is null then
            current ← current.next;
        else if current.data = retro_path[i][0] and end_node is null then
            fin_node ← current.next;
            current ← current.next;
        else if current.data = retro_path[i][2] and end_node is not null then
            current ← current.aux;
        else if current.data = retro_path[i][2] and end_node is null then
            current ← current.next;
        else if current.data = retro_path[i][1] and end_node is not null then
            current ← current.aux;
        else
            current ← current.aux;
            end_node ← null;
            i ← i + 1;
        end
    end
    print path_ham;
    return path_ham;
```

El algoritmo 1, “Backward paths”; identifica y construye caminos de retroceso en un grafo a partir de una matriz de adyacencia y una lista enlazada de nodos visitados. El proceso se lista a continuación: • Inicializa listas para almacenar caminos de retroceso y aristas del grafo. • Recorre la matriz de adyacencia, buscando pares de nodos conectados (i y j) donde exista una arista ($matriz[i][j] == 1$). • Verifica que los nodos sean válidos y que la arista no haya sido previamente registrada. • Si cumple las condiciones, agrega la arista a la lista, actualiza los enlaces auxiliares (aux) en la lista enlazada y guarda el camino de retroceso. • Finalmente, devuelve la lista de caminos de retroceso. Este algoritmo es útil para procesar grafos y encontrar caminos de retroceso.

El algoritmo 2, “Hexagonal array face count”; tiene como objetivo identificar y recopilar las caras (regiones cerradas) de un grafo basándose en una lista de aristas proporcionada $(list\_edges)$. Este pseudocódigo es útil para detectar y almacenar las regiones cerradas o ciclos de un grafo, siendo relevante en el análisis de grafos y problemas relacionados con estructuras poligonales, como mallas hexagonales.

El algoritmo 3, "Linear Order Path for Counting Independent Sets"; está diseñado para construir y devolver un camino (almacenado en $camino\_ham$) en un grafo, utilizando una lista de aristas y los caminos de retroceso definidos previamente en $camino\_retro$. Primero, la función recorrido verifica si la lista está vacía. Si es el caso, retorna una lista vacía. Luego, inicializa las siguientes variables clave: ini_node: apunta al inicio de la lista enlazada; fin_node: marca el final de un subcamino (inicialmente NULL); current: funciona como un puntero para recorrer la lista; $camino\_ham$: es la lista destinada a almacenar el camino construido. Con las variables inicializadas, comienza el proceso de recorrido. Mientras existan nodos por procesar y no se hayan completado todos los caminos de retroceso, el algoritmo realiza las siguientes acciones: 1. Agrega el dato del nodo actual ($current.data$) a la lista $camino\_ham$. 2. Según las condiciones relacionadas con el nodo actual ($current.data$) y los caminos de retroceso definidos en $camino\_retro$, ajusta el pun-

tero current para avanzar al siguiente nodo en la lista enlazada o seguir el enlace auxiliar (aux). 3. Controla el estado de fin_node para detectar el inicio o fin de un subcamino, y, si corresponde, incrementa el índice i para pasar al siguiente camino de retroceso. Finalmente, imprime la lista $camino\_ham$ y devuelve el camino construido como resultado.

4.5. Conclusión

Los conjuntos independientes trascienden disciplinas al simplificar y optimizar cálculos en sistemas químicos complejos, permitiendo el modelado de reacciones químicas y la predicción de propiedades moleculares en compuestos con estructuras poligonales. Este análisis no solo enriquece la comprensión de los sistemas químicos, sino que también abre nuevas oportunidades en áreas como la nanotecnología, el diseño de fármacos y el desarrollo de materiales avanzados. En particular, el cálculo de conjuntos independientes en arreglos hexagonales mediante la trayectoria hamiltoniana Hc representa un avance significativo frente a métodos tradicionales, como el de la matriz de transferencia. Aunque la complejidad temporal permanece de orden exponencial $O(2n)$, este enfoque resulta más práctico y eficiente. Su aplicación en compuestos bencenoides permite estimar propiedades moleculares clave, como la temperatura de ebullición, posibilitando el diseño de nuevos materiales con propiedades innovadoras. Asimismo, se introduce un método eficiente que combina las reglas de Fibonacci y de camino de retroceso para calcular conjuntos independientes en estructuras hexagonales. Este enfoque, con una complejidad temporal reducida a un orden lineal $O(4 * n)$ supera las limitaciones de los métodos tradicionales de crecimiento exponencial, demostrando su importancia en la modelación química y la optimización de procesos relacionados con estructuras hexagonales.

Referencias

Aleid, S., Caceres, J., & Puertas, M. L. (2015). On independent $[1,2]$-sets in trees. *arXiv preprint arXiv:1511.09262*. https://arxiv.org/pdf/1511.09262

Bonilla García, A. N. (2019b). ¿ Podría ser la gravedad una fuerza entrópica?

Bracho, R. L., Gutiérrez-Andrade, M. A., Ortuño-Sánchez, M. P., & Ramírez-Rodríguez, J. (2003b). El problema del conjunto independiente en la selección de horarios de cursos. *Revista de Matemática: Teoría y Aplicaciones*, *10*(1-2), 156-167.

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020c). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets. *Pattern Recognition*, 57-66. https://doi.org/10.1007/978-3-030-49076-8_6

De Ita Luna, G., Bello López, P., & Marcial-Romero, R. (2024b). Counting Rules for Computing the Number of Independent Sets of a Grid Graph. *Mathematics*, *12*(6). https://doi.org/10.3390/math12060922

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023d). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1). https://doi.org/10.61467/2007.1558.2023.v14i1.334

Euler, R. (2005c). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Gutman, I. (1982c). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73. https://doi.org/https://doi.org/10.1515/zna-1982-0115

Hosoya, H. (1973c). Topological index and Fibonacci numbers with relation to chemistry. *Fibonacci Quart*, *11*(3), 255-266. https://doi.org/10.1080/00150517.1973.12430822

Law, H.-F. (2010). On the number of independent sets in a tree. *The Electronic Journal of Combinatorics*, N18-N18.

Merrifield, R. E., & Simmons, H. E. (1981b). Enumeration of structure-sensitive graphical subsets: Theory. *Proceedings of the National Academy of Sciences*, *78*(2), 692-695. https://doi.org/10.1073/pnas.78.2.692

Merrifield, R. E., & Simmons, H. E. (1989b). Topological methods in chemistry.

Sylvester, J. J. (1878). On an Application of the New Atomic Theory to the Graphical Representation of the Invariants and Covariants of Binary Quantics, with Three Appendices. *American Journal of Mathematics*, *1*(1), 64-104. https://doi.org//10.2307/2369436

Vázquez, H. G., Ramírez, C. L., Lopez, P. B., & Luna, G. D. I. (2024). Algoritmo de detección de caras en grafos hexagonales: un enfoque para el conteo de conjuntos independientes Face detection algorithm for hexagonal graphs: an approach to counting independent sets. *Abstraction & Application*, *47*, 74-86.

Yurttas Gunes, A., Delen, S., Demirci, M., Cevik, A. S., & Cangul, I. N. (2020). Fibonacci Graphs. *Symmetry*, *12*(9). https://doi.org/10.3390/sym12091383

Capítulo 5

Conclusiones y recomendaciones generales

5.1. Algoritmo

Los resultados obtenidos en esta investigación permiten presentar diversos casos de estudio que reflejan los avances logrados durante su desarrollo. En primera instancia, se reconoció que el índice de Merrifield-Simmons, conocido también como el número de Fibonacci de un grafo, es un índice topológico ampliamente utilizado en Química Matemática. Este índice corresponde al número total de conjuntos independientes presentes en el grafo.

Bajo este marco, se trabajó considerando la estructura molecular del benceno que se modela gráficamente -como un anillo hexagonal regular-, en el cual cada vértice representa un átomo de carbono y los enlaces simples y dobles se interpretan como las aristas del grafo. Los hexágonos funcionan como unidades básicas que, al unirse compartiendo aristas, conforman una malla poligonal regular y simétrica. Esta estructura puede verse

como un "esqueleto molecular" que se representa adecuadamente mediante una malla hexagonal *G*.

Finalmente, para procesar una malla poligonal, se ocuparon las siguientes reglas: regla Fibonacci (5.1), regla de sustracción (5.2), y regla de camino de retroceso (5.3).

$$(\alpha_{v_{i+1}}, \beta_{v_{i+1}}): \quad \alpha_{v_{i+1}} = \alpha_{v_i} + \beta_{v_i}; \qquad \beta_{v_{i+1}} = \alpha_{v_i} \tag{5.1}$$

$$(\alpha_w, \beta_w)_i: \qquad (\alpha_w, \beta_w) - (0, \beta_{vw}) = \qquad (\alpha_w, \beta_w - \beta_{v,w}) \tag{5.2}$$

$$(\alpha_i, \beta_i): (2\alpha_i, \beta_i): \qquad 2\alpha_i = 2(\alpha_i + \beta_i); \qquad \beta_i = \beta_i \tag{5.3}$$

En la tabla 5.1, se presenta como se va evaluando y construyendo la regla de camino de retroceso, esta regla permite una reducción en el número de líneas de cómputo, disminuyendo considerablemente el orden de complejidad en comparación con los métodos tradicionales.

Tabla 5.1. Recurrencia de la regla de caminos de retroceso

Tamaño de camino	1	2	3	4	5	...	K
Recurrencia	$(\alpha_p,\ \beta_p - \beta_s)$	$(2\alpha_p - \alpha_s,\ \beta_p)$	$(3\alpha_p - \alpha_s,\ 2\beta_p - \beta_s)$	$(5\alpha_p - 2\alpha_s,\ 3\beta_p - \beta_s)$	$(8\alpha_p - 3\alpha_s,\ 5\beta_p - 2\beta_s)$	...	$(F_{k+1}\alpha_p - F_{k-1}\alpha_s,$ $F_k\beta_p - F_{k-2}\beta_s)$
	(1,1)	(2,1)	(3,2)	(5,3)	(8,5)		$(\alpha_p,\ \beta_p)$
	(0,1)	(1,0)	(1,1)	(2,1)	(3,2)		$(\alpha_s,\ \beta_s)$

El algoritmo final que se ha diseñado es un algoritmo exacto que cuenta el índice Merrifield-Simmons (MS) –conteo de conjuntos independientes– en una cuadrícula hexagonal isomórfica tipo malla, modelada como una estructura de grafos, que se presenta a continuación:

Algoritmo 5: Conteo del índice M-S

Input : grafo, lista_arista

Output: Total_CI

Function `conteo_CI(`*grafo, start*`)`**:**

```
dfs(grafo, start);
lista.aristas_retroceso(matriz, lista);
lcaras() ← lista.caras(lista_arista);
camino_retroceso() ← lista.camino_retroceso(matriz, lista);
switch regla do
    case fibo do
        Total_CI ← Total_CI + regla_Fibonacci();
    case retroceso do
        Total_CI ← Total_CI + regla_aristas_retroceso();
    case camino do
        Total_CI ← Total_CI + regla_camino_retroceso();
    otherwise do
        :default
return Total_CI;
```

En la figura 5.1 se muestra un caso de malla poligonal de dimensión 2 x 2 y en la tabla 5.2, se presentan los cálculos de las líneas de cómputo generadas.

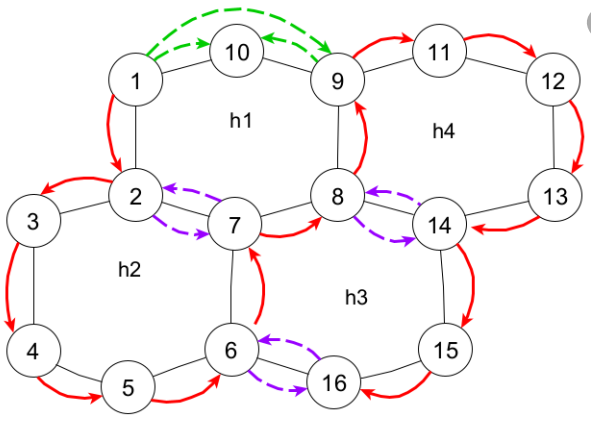


Figura 5.1. Malla poligonal de dimensión 2 x 2.

Tabla 5.2. Conteo de conjuntos independientes con caminos de retroceso

Fila	1	2|2-7	3	4	5	6|6-16	7	7-2	8|8-14	9	9→10→1→9	
Lp	1,1	2,1	3,2	5,3	8,5	13,8	21,13	21,10	31,21	52,31	(104,31)(21,0)	83,31
Lph_1	0,1	1,0	1,1	2,1	3,2	5,3	8,5		13,8	21,13	X	
	Lph_2	0,1	1,0	1,1	2,1	3,2	5,3	X				
					Lph_3	0,8	8,0		8,8	16,8	(32,8)(6,0)	26,8
					Lph_1h_3	0,3	3,0		3,3	6,3	X	
					Lph_2h_3	0,2	2,0		2,2	4,2	(8,2)(0,0)	8,2
								Lph_4	0,21	21,0	(42,0)(8,0)	34,0
								Lph_1h_4	0,8	8,0	X	
								Lph_3h_4	0,8	8,0	(16,0)(3,0)	13,0
								$Lph_1h_3h_4$	0,3	3,0	X	
								$Lph_2h_3h_4$	0,2	2,0	(4,0)(0,0)	4,0

Fila	11	12	13	14	14-8	15	16	16-6	TOTAL CI			
Lp	114,83	197,114	311,197	508,311	508,209	717,508	1225,717	1225,508	1733			
Lph_3	34,26	60,34	94,60	154,94	154,55	209,154	363,209	X				
Lph_2h_3	10,8	18,10	28,18	46,28	28,16	44,28	72,44	X				
Lph_4	34,34	68,34	102,68	170,102	X							
Lph_3h_4	13,13	26,13	39,26	65,39	X							
$Lph_2h_3h_4$	4,4	8,4	12,8	20,12	X							

Para procesar una malla, el recorrido puede iniciarse desde un vértice ubicado en una de sus esquinas. La malla está formada por columnas y renglones de polígonos —en este caso, hexágonos—, por lo que el primer paso consiste en identificar cuál de las dos dimensiones (entre columnas y renglones) es menor. Conviene iniciar la trayectoria en la dirección correspondiente al valor más pequeño. Lo fundamental durante el recorrido es asegurarse de que cada vértice o arista se visite una sola vez y aplicar la regla (5.1). Por ello, cuando se detecta un punto de choque, es decir, cuando el recorrido intenta avanzar hacia un vértice que ya fue visitado, no es posible continuar por esa arista. Si el camino de retroceso está formado por solo una arista, se denomina arista *frond* y aplicamos la regla de sustracción (5.2), y si está formado de más de una arista de retroceso en ese momento se generan los caminos de retroceso, en donde se deberá aplicar la recurrencia (5,3) denominada regla para caminos de retroceso.

La ecuación (5.3) considera el número de aristas involucradas en el camino de retroceso, y permite mantener la posición en el nodo actual, para así, posteriormente continuar con

la trayectoria principal. En la malla de la figura 5.2 se muestra el recorrido de una trayectoria, iniciando en el vértice 1, las líneas en color rojo muestran la trayectoria, las líneas en color lila son las aristas de retroceso y las de color verde los caminos de retroceso que está formado con más de una arista de retroceso, y en las tablas 5.3, 5.4 y 5.5 se desglosa el cómputo de los conjuntos independientes, de la malla de dimensión 3 x 3.

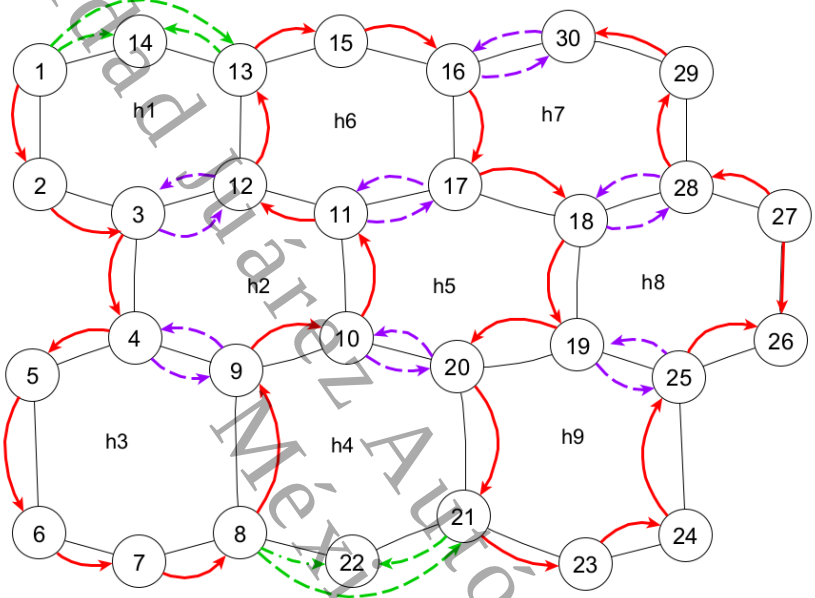


Figura 5.2. Malla poligonal de dimensión 3 x 3.

Tabla 5.3. Conteo de conjuntos independientes con caminos de retroceso de la malla poligonal de dimensión 3x3 (Figura 5.2)

Fila	1|1-14	2	3 |3-12	4|4-9	5	6	7	8|8-22	9	9-4	10|10-20	11|11-17	12	12-3	13	13→14→1→13
Lp	1,1	2,1	3,2	5,3	8,5	13,8	21,13	34,21	55,34	55,25	80,55	135,80	215,135	215,93	308,215	502,215
Lph_1	0,1	1,0	1,1	2,1	3,2	5,3	8,5	13,8	21,13	21,10	31,21	52,31	83,52	83,31	114,83	X
		Lph_2	0,2	2,0	2,2	4,2	6,4	10,6	16,10		26,16	42,26	68,42	X		
		Lph_1h_2	0,1	1,0	1,1	2,1	3,2	5,3	8,5		13,8	21,13	34,21	X		
			Lph_3	0,3	3,0	3,3	6,3	9,6	15,9	X						
			Lph_1h_3	0,1	1,0	1,1	2,1	3,2	5,3	X						
							Lph_4	0,21	21,0		21,21	42,21	63,42	63,30	93,63	152,63
							Lph_1h_4	0,8	8,0		8,8	16,8	24,16	24,10	34,24	X
							Lph_2h_4	0,6	6,0		6,6	12,6	18,12	X		
							$Lph_1h_2h_4$	0,3	3,0		3,3	6,3	9,6	X		
							Lph_3h_4	0,6	6,0	X						
							$Lph_1h_3h_4$	0,2	2,0	X						
										Lph_5	0,55	55,0	55,55	55,39	94,55	154,55
										Lph_1h_5	0,21	21,0	21,21	21,13	34,21	X
										Lph_2h_5	0,16	16,0	16,16	X		
										$Lph_1h_2h_5$	0,8	8,0	8,8	X		
										Lph_4h_5	0,21	21,0	21,21	21,15	36,21	59,21
										$Lph_1h_4h_5$	0,8	8,0	8,8	8,5	13,8	X
										$Lph_2h_4h_5$	0,6	6,0	6,6	X		
										$Lph_1h_2h_4h_5$	0,3	3,0	3,3	X		
										Lph_6	0,80	80,0	80,0	80,80	129,80	
										Lph_1h_6	0,31	31,0	31,0	31,31	X	
										Lph_2h_6	0,26	26,0	X			
										$Lph_1h_2h_6$	0,13	13,0	X			
										Lph_4h_6	0,21	21,0	21,0	21,21	34,21	
										$Lph_1h_4h_6$	0,8	8,0	8,0	8,8	X	
										$Lph_2h_4h_6$	0,6	6,0	X			
										$Lph_1h_2h_4h_6$	0,3	3,0	X			

Tabla 5.4. Conteo de conjuntos independientes con caminos de retroceso de la malla poligonal de dimensión 3x3 (Figura 5.2)

Fila	15	16|16–30	17	17→11	18|18–28	19|19–25	20	20→10	21	21–22–8–21	23	24
Lp	717,502	1219,717	1936,1219	1936,681	2817,1936	4753,2817	7570,4753	7570,3246	10816,7570	18465,7570	26035,18465	44500,26035
Lph_4	215,152	367,215	582,367	582,278	860,582	1442,860	2302,1442	2302,865	3167,2302	X		
Lph_5	209,154	363,209	572,363		935,572	1507,935	2442,1507	X				
Lph_4h_5	80,59	139,80	219,139		358,219	577,358	935,577	X				
Lph_6	209,129	338,209	547,338	X								
Lph_4h_6	55,34	89,55	144,89	X								
	Lph_7	0,717	717,0		717,717	1434,717	2151,1434	2151,1016	3167,2151	5419,2151	7570,5419	12989,7570
	$Lph4h7$	0,215	215,0		215,215	430,215	645,430	645,270	915,645	X		
	$Lph5h7$	0,209	209,0		209,209	418,209	627,418	X				
	$Lph4h5h7$	0,80	80,0		80,80	160,80	240,160	X				
	$Lph6h7$	0,209	209,0	X								
	$Lph4h6h7$	0,55	55,0	X								
				$Lph8$	0,1936	1936,0	1936,1936	1936,1364	3300,1936	5655,1936	7591,5655	13246,7591
				$Lph4h8$	0,582	582,0	582,582	582,363	945,582	X		
				$Lph5h8$	0,572	572,0	572,572	X				
				$Lph4h5h8$	0,219	219,0	219,219	X				
				$Lph7h8$	0,717	717,0	717,717	717,508	1225,717	2100,717	2817,2100	4917,2817
				$Lph4h7h8$	0,215	215,0	215,215	215,135	350,215	X		
				$Lph5h7h8$	0,209	209,0	209,209	X				
				$Lph4h5h7h8$	0,80	80,0	80,80	X				
					$Lph9$	0,2817	2817,0	2817,0	2817,2817	4774,2817	7591,4774	12365,7591
					$Lph4h9$	0,860	860,0	860,0	860,860	X		
					$Lph5h9$	0,935	935,0	X				
					$Lph4h5h9$	0,358	358,0	X				
					$Lph7h9$	0,717	717,0	717,0	717,717	1219,717	1936,1219	3155,1936
					$Lph4h7h9$	0,215	215,0	215,0	215,215	X		
					$Lph5h7h9$	0,209	209,0	X				
					$Lph4h5h7h9$	0,80	80,0	X				

Tabla 5.5. Conteo de conjuntos independientes con caminos de retroceso de la malla poligonal de dimensión 3x3 (Figura 5.2)

Fila	25	25→19	26	27	28	28→18	29	30	30→16	TOTAL CI
Lp	70535, 44500	70535, 32135	102670, 70535	173205, 102670	275875, 173205	275875, 118285	394160, 275875	670035, 394160	670035, 282248	952283
Lph_7	20559, 12989	20559, 9834	30393, 20559	173205, 102670	81345, 50952	81345, 30567	111912, 81345	193257, 111912	X	
Lph_8	20837, 13246		34083, 20837	34083, 20837	54920, 34083	89003, 54920	X			
Lph_7h_8	7734, 4917		12651, 7734	20385, 12651	33036, 20385	X				
Lph_9	19956, 12365	X								
Lph_7h_9	5091, 3155	X								

5.2. Análisis del orden de complejidad

El recorrido de la trayectoria tiene una complejidad lineal *O(V+E)*, porque visita cada vértice *(V)* y cada arista *(E)* una sola vez mientras que la complejidad del algoritmo para calcular el índice de Merrifield-Simmons depende del número máximo de líneas de computación, esto es porque el número de línea es dinámico y va cambiando conforme va abriendo cada polígono, que se deben tener en cualquier momento del calculo *O(MaxLineasComputacion)*.

Si la malla contiene *r* renglones $r \leq c$, existirán $(r+2)$ puntos de inicio de aristas de retroceso de forma contigua, esto da el número máximo de líneas de computación; luego los índices 2 indican que se van cerrando las líneas de ejecución.

Esto genera:

- 2^{r+2} líneas de computación en total, este se multiplica por el orden lineal de *Polinomial(n,m)* que absorbe el orden del recorrido que es de *n*m*.
- Por lo que finalmente la complejidad temporal es de *O*(2^{r+2} x *Polinomial(n,m)*). O de manera equivalente: *O*($2^{min(r,c)+2}$ x *Polinomial(n,m)*).

Para el análisis de la complejidad del algoritmo del cómputo sobre la malla poligonal, se interpreta de la siguiente forma: En la figura 5.3 se puede observar que se ocupan dos

índices el 1 y el 2. El 1 indica inicio del arista de retroceso y el 2 el fin del camino de retroceso; estos nos indican la complejidad del algoritmo. La complejidad exponencial está en el número máximo de 1s que tengamos en las columnas. Cuando los 1s son contiguos, se duplican las líneas beta (aunque para los valores con betas en cero no se generan nuevas líneas de computación), como pasa en el nodo 2 de la figura 1. Lo anterior se puede visualizar en la tabla. En los índices *Lp*, *Lph1* se duplican al momento de abrir una arista de retroceso (2 | 2-7) al duplicarse quedaría como *Lph2*, *Lph1h2* pero como el beta de *Lph1* es igual a 0, entonces ya no se genera la duplicación de la línea *Lph1h2* y esto permite la reducción de líneas de cómputo.

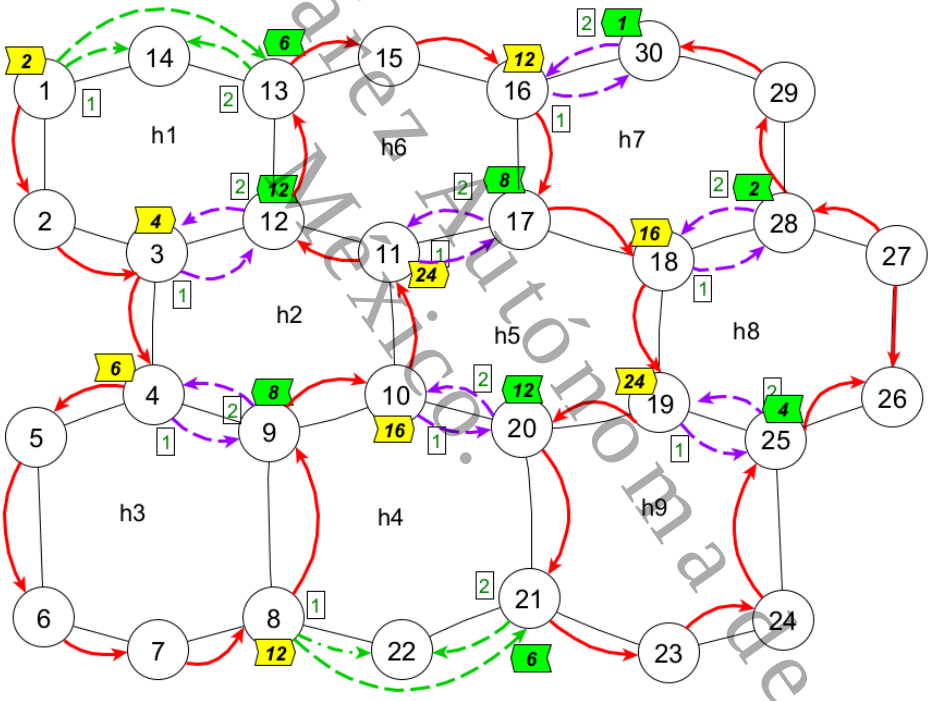


Figura 5.3. Análisis del orden de complejidad.

Al contar el número máximo de etiquetas amarillas (estas suman sobre los números 1's que se escriben como exponente), mientras que las etiquetas verdes restan uno al exponente, considerando en el peor de los casos, por ejemplo: las primeras 4 etiquetas amarillas (que corresponden a que se visitaron 4 inicios de caminos de retroceso), luego

una verde que resta un uno al 4, luego dos amarillas (3+2 =5) y este es el máximo de etiquetas amarillas en toda la cadena de etiquetas (amarillas y verdes) que corresponden a 3 hexágonos por columna + 2 puntos de caminos de retroceso que hay que abrir antes de empezar a cerrar al visitar los puntos finales de los caminos de retroceso.

5.3. Conclusión

En lo referente al cumplimiento de los objetivos planteados, se resumen a continuación las conclusiones obtenidas: El desarrollo de un algoritmo exacto para calcular el índice Merrifield-Simmons (MS) en cuadrículas hexagonales modeladas como grafos permitió establecer un análisis preciso de su complejidad temporal. La propuesta demostró que es posible transformar un problema tradicionalmente asociado con tiempos de ejecución exponenciales en un enfoque algorítmico más eficiente y estructurado.

Por lo anterior, es que la estimación en el peor caso sería: $O(2^{min(r,c)+2}$ x *Polinomial(n,m)*) donde c es el número de columnas y r el número de renglones, y n vértices y m aristas en el mallado. La complejidad es de orden exponencial con respecto al número de renglones cuando $r \leq c$. El término *Polinomial(n,m)* surge de la aplicación de la regla de Fibonacci y la regla del camino de retroceso, aunque la complejidad del método sigue siendo de carácter exponencial, esta depende directamente del número máximo de líneas de computación requeridas en cada polígono, permitiendo un mejor control sobre el rendimiento en comparación con otros métodos existentes.

Se lograron avances significativos en la optimización del proceso computacional a través de la creación de la regla (3) de recurrencia de caminos de retroceso para el conteo de conjuntos independientes en mallas poligonales, ésta es una nueva aportación a las ciencias de la computación en el área de optimización combinatoria. Estos resultados no solo validan la hipótesis planteada, sino que también evidencian que la modelación

mediante grafos es una herramienta poderosa para abordar problemas combinatorios complejos en estructuras poligonales. Se establecen bases sólidas para futuras investigaciones orientadas a mejorar aún más la eficiencia computacional en conteo de conjuntos independientes en configuraciones moleculares no sólo en mallas hexagonales, sino en cualquier polígono (de 4 o más lados) que corresponden a diferentes compuestos, análisis estructural y otras aplicaciones donde el índice Merrifield-Simmons desempeña un papel fundamental. Al igual que existe la posibilidad de explorar otras metodologías en trabajos futuros, como la metodología de *Branch-and-Bound* , ya que justo son las mismas bases de la presente investigación.

Alojamiento de la Tesis en el Repositorio Institucional	
Título de la tesis:	Algoritmo para calcular el índice Merrifield-Simmons sobre mallas poligonales
Autor:	Herlinda González Vázquez
ORCID:	https://orcid.org/0009-0002-4416-6822
Resumen:	Contar conjuntos independientes en grafos es un problema computacional dentro de la teoría de grafos: incluso grafos de grado máximo tres, se clasifican como #P-completo, por lo que los enfoques clásicos suelen crecer de forma exponencial. En este contexto se propone un algoritmo orientado a calcular el índice de Merrifield-Simmons (MS) en mallas poligonales, estructuras que modelan grafos moleculares asociados al grafeno y a compuestos aromáticos bencenoides. La idea central del método es construir una trayectoria hamiltoniana que visite cada vértice de una cuadrícula hexagonal isomórfica exactamente una vez. El cómputo de los conjuntos independientes se realiza aplicando tres reglas: (1) una recurrencia tipo Fibonacci, cuando la arista visitada pertenece al árbol de búsqueda; (2) una regla de sustracción, al detectar bordes o aristas frontales o posteriores; y (3) una regla de camino de retroceso, que aprovecha las aristas implicadas en retrocesos para recortar de manera importante el número de pasos. La metodología completa incluye generar listas de adyacencias, construir la matriz de adyacencias y ejecutar DFS para localizar ciclos y bordes de retroceso. Aunque el problema sigue siendo exponencial, el algoritmo reduce drásticamente las operaciones del conteo del MS en mallas poligonales, por lo que la estimación del orden de complejidad en el peor caso se expresa como $O(2^{min(r,c)+2} xPolinomial(n,m))$, mejorando frente a técnicas clásicas de transferencia de matrices.
Palabras clave:	Índice de Merrifield-Simmons, conjuntos independientes, trayectoria hamiltoniana, grafos hexagonales, orden de complejidad.
Referencias citadas:	En la siguiente página se muestran las referencias.

Bibliografía

Aleid, S., Caceres, J., & Puertas, M. L. (2015). On independent $[1,2]$-sets in trees. *arXiv preprint arXiv:1511.09262*. https://arxiv.org/pdf/1511.09262

Barrero, A. C., de García, G. W., & Parra, R. M. M. (2010). *Introducción a la Teoría de Grafos*. Elizcom sas.

Bonilla García, A. N. (2019a). ¿ Podría ser la gravedad una fuerza entrópica?

Bonilla García, A. N. (2019b). ¿ Podría ser la gravedad una fuerza entrópica?

Bonsma, P., & Breuer, F. (2012). Counting hexagonal patches and independent sets in circle graphs. *Algorithmica*, *63*(3), 645-671. https://doi.org/10.1007/s00453-011-9561-y

Bracho, R. L., Gutiérrez-Andrade, M. A., Ortuño-Sánchez, M. P., & Ramírez-Rodríguez, J. (2003a). El problema del conjunto independiente en la selección de horarios de cursos. *Revista de Matemática: Teoría y Aplicaciones*, *10*(1-2), 156-167.

Bracho, R. L., Gutiérrez-Andrade, M. A., Ortuño-Sánchez, M. P., & Ramírez-Rodríguez, J. (2003b). El problema del conjunto independiente en la selección de horarios de cursos. *Revista de Matemática: Teoría y Aplicaciones*, *10*(1-2), 156-167.

Bracho, R. L., & Sánchez, M. P. O. (2000a). Un algoritmo paralelo para el problema del conjunto independiente. *Revista de Matemática: Teoría y Aplicaciones*, *7*(1-2), 125-134.

Bracho, R. L., & Sánchez, M. P. O. (2000b). Un algoritmo paralelo para el problema del conjunto independiente. *Revista de Matemática: Teoría y Aplicaciones*, *7*(1-2), 125-134.

Cook, S. A. (1971). The complexity of theorem-proving procedures.

Dai, S., & Zhang, R. (2012). The Merrifield-Simmons Index and Hosoya Index of C (n, k, λ) Graphs. *Journal of Applied Mathematics*, *2012*, 1-8.

De Ita, G., Bello, P., & Contreras, M. (2023a). A method for computing the Merrifield–Simmons index on benzenoid systems. *Match Communications in Mathematical*

and in Computer Chemistry, *89*(1), 245-270. https://doi.org/10.46793/match.89-1.245I

De Ita, G., Bello, P., & Contreras, M. (2023b). A Method for Computing the Merrifield-Simmons Index on Benzenoid Systems. *Match-communications in mathematical and in computer chemistry*, *89*(1), 245-270.

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020a). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets, 57-66.

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020b). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets, 57-66. https://doi.org/ https://doi.org/10.1007/978-3-030-49076-8_6

De Ita, G., Rodríguez, M., Bello, P., & Contreras, M. (2020c). Basic Pattern Graphs for the Efficient Computation of Its Number of Independent Sets. *Pattern Recognition*, 57-66. https://doi.org/10.1007/978-3-030-49076-8_6

De Ita Luna, G., Bello López, P., & Marcial-Romero, R. (2024a). Counting Rules for Computing the Number of Independent Sets of a Grid Graph. *Mathematics*, *12*(6). https://doi.org/10.3390/math12060922

De Ita Luna, G., Bello López, P., & Marcial-Romero, R. (2024b). Counting Rules for Computing the Number of Independent Sets of a Grid Graph. *Mathematics*, *12*(6). https://doi.org/10.3390/math12060922

De Ita Luna, G., Marcial Romero, R., Bello Lopez, P., & Meliza, C. G. (2026). An Exact Algorithm for Counting the Number of Independent Sets of a Graph. *Mathematics*, *14*(2), 275. https://doi.org/https://doi.org/10.3390/math14020275

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023a). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1).

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023b). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1). https://research.ebsco.com/linkprocessor/plink?id=4318a715-0f3e-30fc-9741-35999983df7e

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023c). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization Problems & Informatics*, *14*(1). https://doi.org/10.61467/2007.1558.2023.v14i1.334

De Ita Luna, G., Vidal, M. T., & Loranca, B. B. (2023d). A Novel Method for Counting Independent Sets in a Grid Graph. *International Journal of Combinatorial Optimization*

Problems & Informatics, *14*(1). https://doi.org/10.61467/2007.1558.2023.v14i1.334

Deng, Z., Ding, J., Heal, K., & Tarokh, V. (2017). *The number of independent sets in hexagonal graphs*. https://doi.org/10.1109/ISIT.2017.8007062

El-Basil, S. (1988a). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29.

El-Basil, S. (1988b). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29.

El-Basil, S. (1988c). Theory and computational applications of Fibonacci graphs. *Journal of Mathematical Chemistry*, *2*(1), 1-29. https://doi.org/10.1007/BF01166466

Euler, R. (2005a). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Euler, R. (2005b). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Euler, R. (2005c). The Fibonacci number of a grid graph and a new class of integer sequences. *J. Integer Seq*, *8*(2).

Gutman, I. (1982a). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73.

Gutman, I. (1982b). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73. https://doi.org/https://doi.org/10.1515/zna-1982-0115

Gutman, I. (1982c). Topological Properties of Benzenoid Systems. IX*. On the Sextet Polynomial. *Zeitschrift für Naturforschung A*, *37*(1), 69-73. https://doi.org/https://doi.org/10.1515/zna-1982-0115

Hosoya, H. (1971). Topological index. A newly proposed quantity characterizing the topological nature of structural isomers of saturated hydrocarbons. *Bulletin of the Chemical Society of Japan*, *44*(9), 2332-2339.

Hosoya, H. (1973a). Topological Index and Fibonacci Numbers with Relation to Chemistry. *The Fibonacci Quarterly*, *11*(3), 255-265. https://doi.org/10.1080/00150517.1973.12430822

Hosoya, H. (1973b). Topological index and Fibonacci numbers with relation to chemistry. *Fibonacci Quart*, *11*(3), 255-266.

Hosoya, H. (1973c). Topological index and Fibonacci numbers with relation to chemistry. *Fibonacci Quart*, *11*(3), 255-266. https://doi.org/10.1080/00150517.1973.12430822

Jaffe, A. M. (2006). The millennium grand challenge in mathematics. *Notices of the AMS*, *53*(6), 652-660.

Karp, R. M. (1972). Reducibility among combinatorial problems. *University of California at Berkeley*.

Lamm, S., Sanders, P., Schulz, C., Strash, D., & Werneck, R. F. (2017a). Finding near-optimal independent sets at scale. *Journal of Heuristics*. https://doi.org/https://doi.org/10.1137/1.9781611974317.12

Lamm, S., Sanders, P., Schulz, C., Strash, D., & Werneck, R. F. (2017b). Finding near-optimal independent sets at scale. *Journal of Heuristics*. https://doi.org/https://doi.org/10.1137/1.9781611974317.12

Law, H.-F. (2010). On the number of independent sets in a tree. *The Electronic Journal of Combinatorics*, N18-N18.

Levin, L. A. (1973). Universal sequential search problems. *Problemy peredachi informatsii*, *9*(3), 115-116.

López-Ramírez, C., Gómez, J. E. G., & Luna, G. D. I. (2020). Building a maximal independent set for the vertex-coloring problem on planar graphs. *Elecronic Notes in Theoretical Computer Science*, *354*, 75-89.

Lozzo, G. D., & Rutter, I. (2016). On the Complexity of Realizing Facial Cycles. https://doi.org/10.48550/arXiv.1607.02347

Mañas, J. A. (1997). Análisis de algoritmos: Complejidad.

Merrifield, R. E., & Simmons, H. E. (1981a). Enumeration of structure-sensitive graphical subsets: Theory. *Proceedings of the National Academy of Sciences*, *78*(2), 692-695. https://doi.org/10.1073/pnas.78.2.692

Merrifield, R. E., & Simmons, H. E. (1981b). Enumeration of structure-sensitive graphical subsets: Theory. *Proceedings of the National Academy of Sciences*, *78*(2), 692-695. https://doi.org/10.1073/pnas.78.2.692

Merrifield, R. E., & Simmons, H. E. (1980). The structures of molecular topological spaces. *Theoretica chimica acta*, *55*, 55-75. https://doi.org/https://doi.org/10.1007/BF00551410

Merrifield, R. E., & Simmons, H. E. (1989a). Topological methods in chemistry.

Merrifield, R. E., & Simmons, H. E. (1989b). Topological methods in chemistry.

Morales Alcántara, L. D., et al. (2019). *Sistema web para el conteo eficiente de Conjuntos Independientes en estructuras jerárquicas (Árboles)* [B.S. thesis].

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning.

Discrete Optimization, *19*, 79-102. https://doi.org/https://doi.org/10.1016/j.disopt.2016.01.005

Okamoto, Y., Uno, T., & Uehara, R. (2005). Linear-time counting algorithms for independent sets in chordal graphs. *Graph-Theoretic Concepts in Computer Science: 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers 31*, 433-444.

Perdomo Flández, J. A., et al. (2015). *Análisis del cómputo exhaustivo de conjuntos independientes* [B.S. thesis].

Perez Barrios, O., et al. (2015). *Construcción de un algoritmo para contar modelos de fórmulas en 2-FC* [Tesis de maestría].

Samotij, W. (2015). Counting independent sets in graphs. *European Journal of Combinatorics*, *48*, 5-18. https://doi.org/https://doi.org/10.1016/j.ejc.2015.02.005

Simmons, H., & Merrifield, R. (1977). Mathematical description of molecular structure: Molecular topology. *Proceedings of the National Academy of Sciences*, *74*(7), 2616-2619. https://doi.org/https://doi.org/10.1073/pnas.74.7.2616

Sylvester, J. J. (1878). On an Application of the New Atomic Theory to the Graphical Representation of the Invariants and Covariants of Binary Quantics, with Three Appendices. *American Journal of Mathematics*, *1*(1), 64-104. https://doi.org//10.2307/2369436

Teimoori Faal, H. (2023). Face-Counting Identities and Derivatives of Face Polynomials. *Turkish Journal of Mathematics and Computer Science*, *15*(2), 443-448. https://doi.org/10.47000/tjmcs.1120399

Vadhan, S. P. (2001a). The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, *31*(2), 398-427. https://doi.org/10.1137/S0097539797321602

Vadhan, S. P. (2001b). The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, *31*(2), 398-427. https://doi.org/10.1137/S0097539797321602

Vázquez, H. G., Ramírez, C. L., Lopez, P. B., & Luna, G. D. I. (2024). Algoritmo de detección de caras en grafos hexagonales: un enfoque para el conteo de conjuntos independientes Face detection algorithm for hexagonal graphs: an approach to counting independent sets. *Abstraction & Application*, *47*, 74-86.

Vegi Kalamar, A. (2023). Counting Traversing Hamiltonian Cycles in Tiled Graphs. *Mathematics*, *11*(12). https://doi.org/10.3390/math11122650

Vesel, A. (2013). Fibonacci dimension of the resonance graphs of catacondensed benzenoid graphs. *Discrete Applied Mathematics*, *161*(13), 2158-2168. https://doi.org/https://doi.org/10.1016/j.dam.2013.03.019

Wurtz, J., Lopes, P. L., Gemelke, N., Keesling, A., & Wang, S. (2022a). Industry applications of neutral-atom quantum computing solving independent set problems. *arXiv preprint arXiv:2205.08500*.

Wurtz, J., Lopes, P. L., Gemelke, N., Keesling, A., & Wang, S. (2022b). Industry applications of neutral-atom quantum computing solving independent set problems. *arXiv preprint arXiv:2205.08500*.

Yin, X.-F., Yao, X.-C., Wu, B., Fei, Y.-Y., Mao, Y., Zhang, R., Liu, L.-Z., Wang, Z., Li, L., Liu, N.-L., et al. (2023a). Solving independent set problems with photonic quantum circuits. *Proceedings of the National Academy of Sciences*, *120*(22), e2212323120. https://doi.org/https://doi.org/10.1073/pnas.2212323120

Yin, X.-F., Yao, X.-C., Wu, B., Fei, Y.-Y., Mao, Y., Zhang, R., Liu, L.-Z., Wang, Z., Li, L., Liu, N.-L., et al. (2023b). Solving independent set problems with photonic quantum circuits. *Proceedings of the National Academy of Sciences*, *120*(22), e2212323120. https://doi.org/https://doi.org/10.1073/pnas.2212323120

Yurttas Gunes, A., Delen, S., Demirci, M., Cevik, A. S., & Cangul, I. N. (2020). Fibonacci Graphs. *Symmetry*, *12*(9). https://doi.org/10.3390/sym12091383