Repositorio de grafos para el conteo de conjuntos independientes

Juan Antares Perdomo Flandez

Benemérita Universidad Autónoma de Puebla

kanabos_delid@hotmail.com

Pedro Bello López

Benemérita Universidad Autónoma de Puebla

pbello@cs.buap.mx

Meliza Contreras González

Benemérita Universidad Autónoma de Puebla

mcontreras@cs.buap.mx

Brayan Chavez Benavides

Benemérita Universidad Autónoma de Puebla

brayan120@gmail.com

Resumen

La teoría de grafos es empleada en infinidad de aplicaciones en las áreas de química, comunicaciones, control, modelos de transporte, robótica por lo que resulta indispensable contar con herramientas que permitan el diseño y análisis de problemas que involucren su modelado mediante estas estructuras. Por esta razón se propone un sistema con dos usos, por un lado un editor gráfico que permita diseñar grafos con topologías precisas para grafos ponderados para exportarlos y tenerlos disponibles en el uso de otras aplicaciones y por otro lado una interfaz que con entradas específicas calcule propiedades típicas de los grafos como los conjuntos independientes.

Palabra(s) Clave(s): conjuntos independientes, editor gráfico, grafos, repositorio.

1. Marco Teórico

Un grafo es un conjunto de nodos que son unidos por vértices o aristas [6]. Estos pueden ser representados de muchas formas dependiendo su aplicación. Dentro de este trabajo se estudian grafos con características específicas.

Reinhard Diestel [9] define a un grafo como:

> Un grafo es un par G=(V,E) de conjuntos tal que E ⊆ 〖[V]〗^2; así, los elementos de E son 2 elementos subconjuntos de V. Los elementos de V son los vértices (nodos o puntos) del grafo G, los elementos de E son sus vértices (o líneas).

Mientras que Reinaldo Giudici [7] define en su libro:

> Un grafo finito G es un par (V(G),E(G)), donde V(G) es un conjunto finito, no vacío, cuyos elementos elementos llamaremos vértices, y E(G) es un conjunto de pares de vértices de V(G) x V(G) que definen una relación R, de modo que si los vértices están en la relación, existe al menos una línea que los une.

En la anterior definición también se pueden incluir los multigrafos [3,4] los cuales son aquellos que tienen más de una arista para un par de nodos o aristas que relacionan al mismo nodo. En el presente trabajo solo se estudiarán los grafos simples los cuales por el contrario solo tienen una sola arista para cada par de nodos y no incluyen aristas que relacionen al mismo nodo en sus extremos.

Los grafos no dirigidos son un tipo de grafos donde no importa el orden en que se hagan los recorridos a través de las aristas de los nodos del grafo, por lo que se puede recorrer de igual forma del nodo 1 al nodo 2 como del 2 al 1 mientras exista una arista que una a ambos nodos. Un grafo no dirigido se define como el conjunto de nodos N y vértices V que forman el grafo en donde el orden no importa, es decir que un grafo G(N, V) no será dirigido si para todo vértice v |v ∈V se cumple que $v(n_i,n_j)=v(n_j,n_i)$ ∀ v ∈G.

Al hablar de grafos ponderados se habla de aquellos que atribuyen un peso a la arista, esto es común verlo en el análisis del camino más corto donde se hace una búsqueda de a través de cuáles aristas es necesario pasar para que se vaya de un nodo inicial a un nodo final y que este conjunto de aristas sea el que presente el menor peso. Sin embargo en el estudio de los conjuntos independientes de los grafos es irrelevante la ponderación de las aristas.

Los grafos no ponderados no presentan algún peso dentro de las aristas por lo que no existe diferencia al recorrer los grafos sea cual sea su camino de un nodo a otro.

Dentro del estudio ya se ha mencionado que se incluyen grafos simples, bien estos pueden contener ciclos, lo cual eleva la complejidad en muchos casos en el desarrollo de los algoritmos. También se estudiarán los casos más simples en donde se presenten grafos sin ciclos (arboles).

Grafos conexos y no conexos

Un grafo es conexo si para cualquier par de vértices existe una ruta por medio de las aristas que pueda unir ambos vértices (fig. 1a), por el contrario si no existen rutas posibles el grafo será no conexo o disconexo (fig. 1b). A pesar de que ambos tipos de grafos se incluyen dentro del estudio de los conjuntos independientes es necesario separarlos en la clasificación debido a que podrían elevar o disminuir la complejidad a la hora de ser procesados.

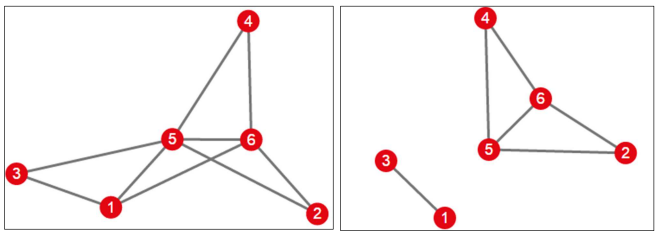


Fig. 1.a grafo conexo Fig. 1.b grafo no conexo

Grafos completos y desconectados

Estos tipos de grafos también son necesarios para tener en cuenta ya que de igual forma pueden elevar o disminuir la complejidad del proceso.

Un grafo simple será completo si cada nodo está conectado a todos los demás nodos del grafo (fig. 2a) por el contrario los grafos desconectados que no tienen absolutamente ninguna arista (fig. 2b). Estos bien pueden presentar tanto el mejor como el peor caso de el algoritmo ya que son descartables las posibilidades a la hora de analizar y contar la cantidad de conjuntos independientes existentes en los grafos.

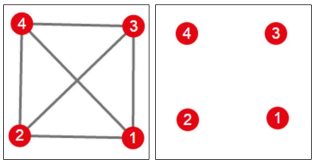


Fig. 2.a grafo completo. Fig. 2.b (derecha) grafo desconectado.

Valencia o grado

La valencia o grado de un nodo es la cantidad de aristas que inciden en él y se representa como δ(x). El grado máximo y mínimo de un grafo se representan respectivamente como Δ(G) y δ(G).

La adyacencia se refiere a la vecindad de unos nodos con otros. Un par de nodos será adyacente si existe una arista que los una.

Grafo complemento

Siendo G un grafo cualquiera, su complemento G será aquel que tenga el mismo conjunto de vértices de G pero presente solo las aristas que no están en el grafo original. (Ver Figura 3)

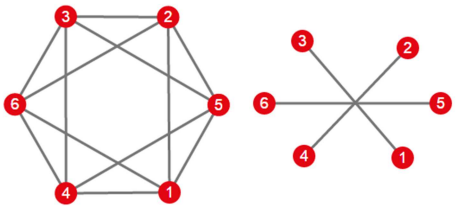


Fig. 3. Complemento de un grafo.

Los conjuntos independientes [5] de un grafo son todos aquellos conjuntos de nodos que se pueden agrupar sin que exista una arista o vértices que una a cualquiera de los nodos agrupados. Los conjuntos pueden tener cualquier tamaño dependiendo del grafo dado que se incluye el conjunto vacío, los conjuntos donde se agrupan cada uno de los nodos, los cuales son de tamaño uno y el resto de conjuntos en donde se incluyen a todos los nodos que no tienen aristas en común.

Algunas definiciones se presentan a continuación:

Un conjunto independiente de G es un subconjunto $S \subseteq V$ de vértices tal que ningún par de vértices en S están conectados por una arista de G.

Dado un grafo G, un conjunto independiente es un subconjunto S de vértices en G tal que no hay dos vértices en S que sean adyacentes (conectados por un vértice).

Otro concepto que se estudia a la par de los conjuntos independientes de un grafo son los conjuntos maximales.

Los maximales son un subconjunto de los conjuntos independientes de un grafo donde ya no se puede añadir ningún otro nodo al conjunto para que este siga siendo independiente. Los conjuntos maximales pueden tener el tamaño de cualquier otro conjunto independiente. Por ejemplo si el nodo de un grafo estuviera conectado a todos los otros nodos, un conjunto maximal del grafo sería aquel que solo incluyera a dicho nodo, ya que no se podría agregar a ningún otro y que el conjunto conserve su propiedad de independencia. De igual forma pueden existir varios conjuntos maximales del mismo tamaño.

Algunas definiciones de los conjuntos maximales son:

Un conjunto independiente se llama maximal si no existe otro conjunto independiente en el que esté contenido propiamente. Esto es, un conjunto independiente I es maximal si para todo conjunto H tal que $I \subset H$ con $I \neq H$, se tiene que H no es independiente.

Un conjunto independiente maximal de un grafo es un conjunto independiente que no puede ser expandido a otro conjunto independiente por adición de algún otro vértice en el grafo.

El tamaño máximo de los conjuntos independientes de un grafo se ve limitado por la cantidad de nodos del mismo grafo así como la cantidad de aristas, aunque no existe ninguna regla matemática que obtenga el tamaño máximo de los conjuntos independientes de un grafo. Esto se conoce como el conjunto máximo de un grafo o el conjunto independiente máximo y puede existir más de un conjunto con el tamaño máximo para un grafo dado.

El problema del conjunto independiente máximo y ha sido extensamente estudiado por algoritmos exactos y es un problema NP-Difícil.

Los problemas de conteo, aunque interesantes matemáticamente, tienen relación con importantes problemas prácticos. Por ejemplo, los problemas de confiabilidad en una red son finalmente traducidos a problemas de conteo [12].

Vadhan [12] demostró que el conteo de conjuntos independientes en grafos de grado mayor o igual a 4, es un problema #P-completo. Posteriormente, Greenhill [11] muestra que éste mismo problema es #P-completo aun cuando el grado del grafo se reduzca a 3. Roth [12] presenta algoritmos de complejidad polinomial para el mismo problema pero sobre grafos de grado 2 o menor.

Se han propuesto algoritmos capaces de resolver el problema de los conjuntos independientes[10] para estructuras específicas de grafos como en el caso de grafos del tipo cadena y ciclo [1], en donde se ha creado un algoritmo que usando la secuencia

o serie de Fibonacci como base obtienen los conjuntos independientes de dichos grafos. De igual forma se han desarrollado soluciones para los conjuntos maximales de forma paralela haciendo uso de árboles [2].

Otros trabajos proponen y mejoran algoritmos para grafos circulares enfocándose a la optimización del compilador como en la asignación de registros en "pipelined bucles" de software o bien en diseño de VLSI (Very Large Scale Integration) para sistemas de circuitos integrados basados en transistores [14]. Igualmente se ha llegado a proponer soluciones para grafos con grado o valencia baja que por su simplicidad y siendo soluciones más específicas pueden conseguirse tiempos por debajo de la media como en el propuesto por Mingyu Xiao y Hiroshi Nagamochi [13] que da a conocer una solución de tiempo $1.1736^{n}n^{O(1)}$ para grafos de grado 5.

El estudio de soluciones para grafos con ciertas características constituye un tópico de investigación y aplicación en ciertas áreas como lo son en redes computacionales en donde puede variar la topología y un algoritmo especifico llegará a ser más eficiente, sin embargo la necesidad de tener una solución general es igualmente útil para cuando se desconoce la estructura.

Al momento no se tiene referencia de un algoritmo general que obtenga los conjuntos independientes de un grafo, puesto que solo se tienen resultados teóricos como los antes expuestos, por lo que en este trabajo se estudiaran las estructuras de grafos donde es posible obtener los conjuntos independientes y diseñar un algoritmo exhaustivo y su aplicación computacional para realizar pruebas y determinar los tiempos que tardan en realizar el proceso del cálculo de los conjuntos independientes.

2. Desarrollo

Como se sabe los grafos pueden ser representados de diferentes formas[8], dos de ellas son usando cierto tipo de listas ligadas donde cada nodo apuntará a todos los

nodos adyacentes a él, sin embargo de esta forma no se puede tener un control total de los elementos de un grafo los cuales son nodos y aristas. En esta forma de representación las aristas irían implícitas por lo que otra forma de representar un grafo son con pares de listas una que contenga los nodos y otra que contenga las aristas, esta forma es más útil en especial para el dibujo de los grafos donde es importante saber la posición de los elementos que conforman a la estructura en general. En esta segunda forma de representación las aristas tienen dos apuntadores a los nodos a los que está conectada y los nodos siendo que pueden tener cero o más aristas que incidan en ellos tendrán una lista de todas estas aristas incidentes.

A continuación se muestra la interfaz del editor de gráficos, en la Figura 4 se muestra cómo se construye el grafo a partir de la ubicación de los nodos y la conexión por medio de las aristas, en la Figura 5 se muestra un grafo complejo generado en formato png a partir de la opción exportar:

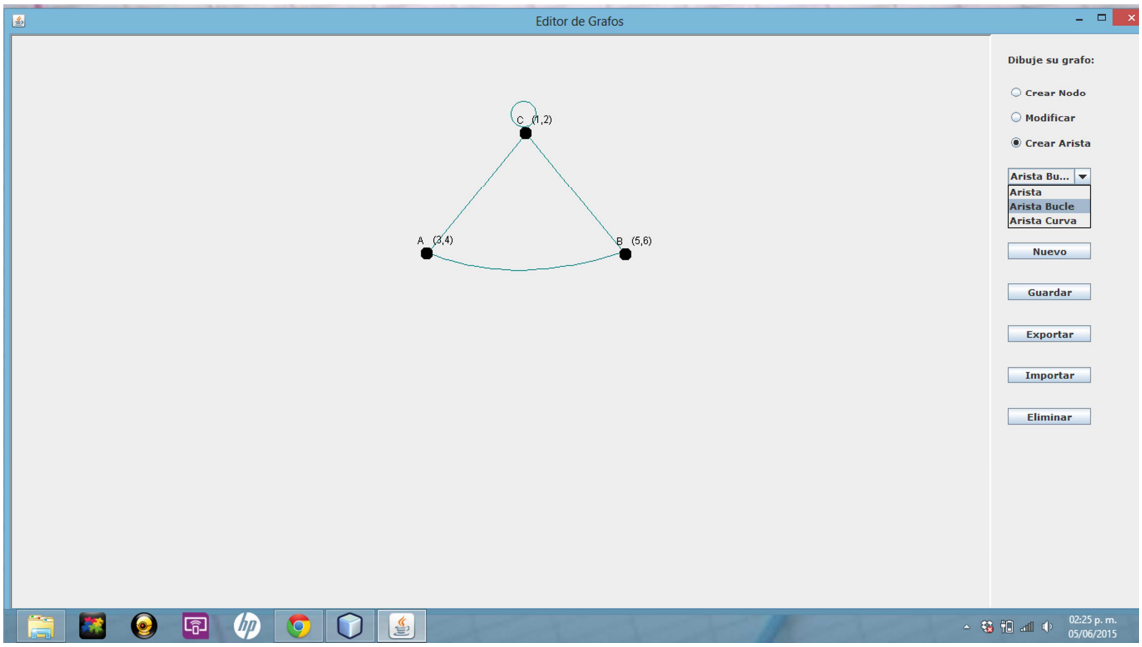


Fig. 4. Editor de gráficos diseño de nodos y aristas.

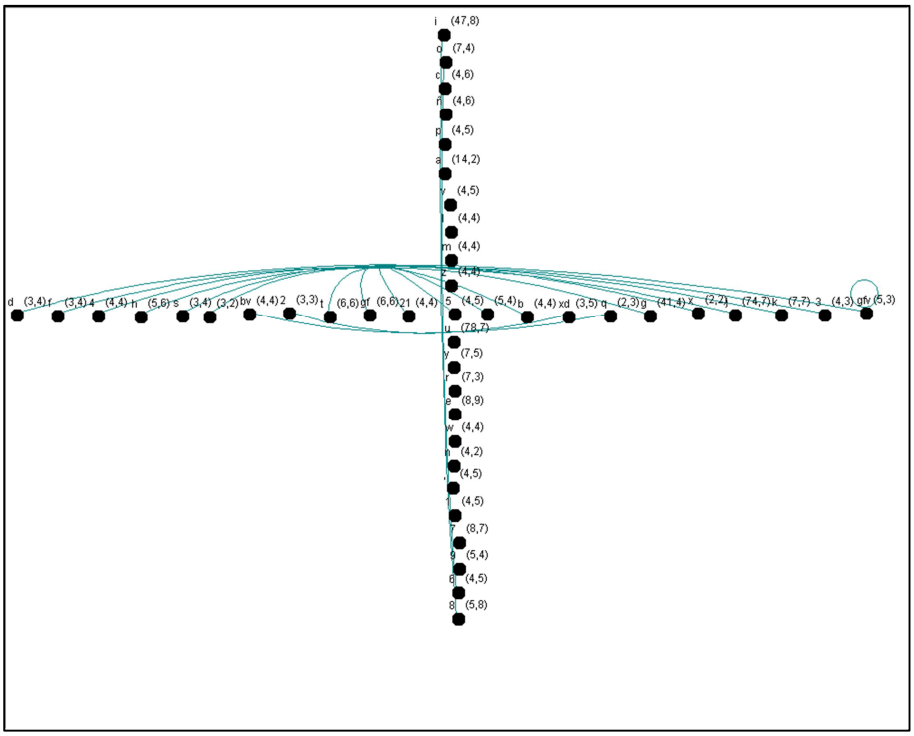


Fig. 5. Grafo obtenido en formato png.

Para encontrar todos los conjuntos independientes de un grafo es necesario hacer un análisis exhaustivo sobre cada nodo verificando si alguno del resto de los nodos es adyacente, en caso de no serlo se puede continuar la búsqueda de algún nodo que no sea adyacente ni al primer nodo ni al segundo y así continuar en busca de conjuntos de mayor tamaño hasta encontrar un maximal, un conjunto independiente cuya adición de cualquier otro nodo hará que este deje de ser independiente. En la Figura 6 se muestra la interfaz que calcula los conjuntos independientes de un grafo. En este caso el usuario debe introducir en una caja de texto el total de vértices así como las aristas del grafo, posteriormente pulsará el botón **Calcular y Dibujar** y la aplicación desplegará el grafo correspondiente que puede expandirse para apreciar la topología y muestra el total de conjuntos independientes así como lista todos los conjuntos dependiendo de su cardinalidad y el tiempo que duro el cálculo.

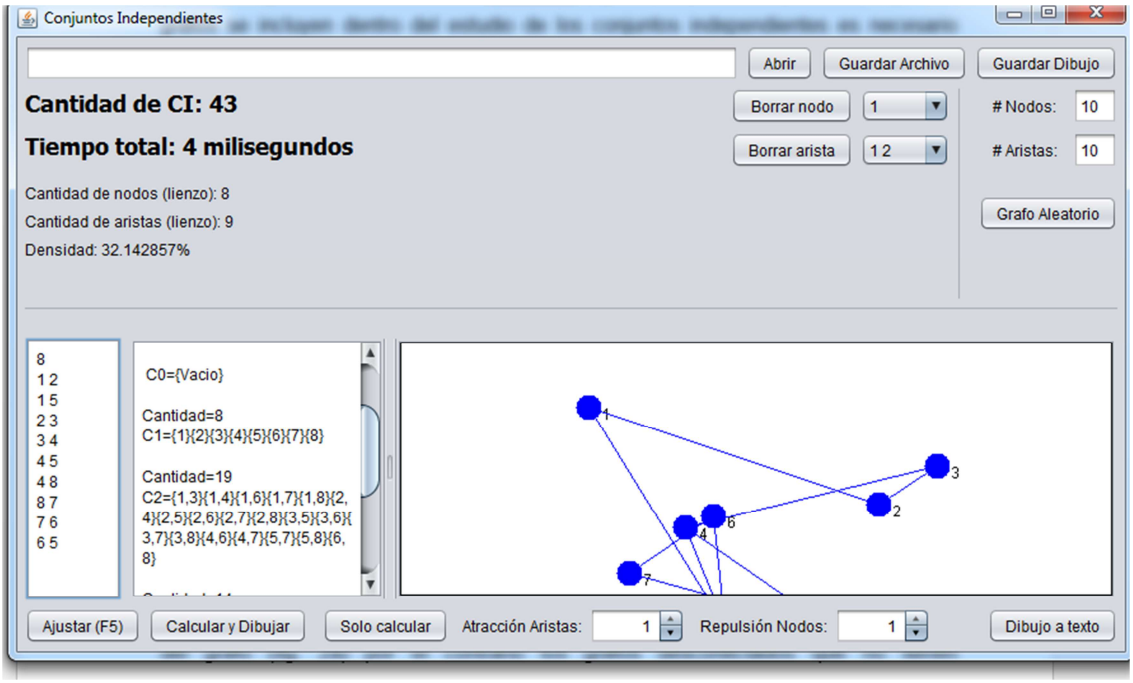


Fig. 6. Interfaz del contador de la propiedad de conjuntos independientes.

Usando la forma de representación de grafos de la que se habló con anterioridad se pueden hacer estas búsquedas pero sin embargo termina teniendo cierta ineficacia debido a la naturaleza de las listas ligadas que tienen que indexar en diferentes partes de memoria para acceder a sus elementos y que también para ello es necesario recorrer los anteriores puestos.

Para este problema se optó mejor por el uso de otra forma de representación de los grafos que es usando una tabla de $n \times n$ (n igual a la cantidad de nodos en el grafo) donde se enlistan a lo alto y a lo ancho todos los nodos y en las intersecciones en la tabla de un nodo N_1 con un nodo N_2 se colocará un 1 o true si este par de nodos son adyacentes entre si y un 0 o false en caso contrario.

Esta forma a pesar de tampoco puede tener un gran control sobre ni las aristas ni los nodos resulta más beneficiosa para la búsqueda de conjuntos independientes debido a que todo se hace en un área de la memoria y que esta búsqueda se basa en encontrar las adyacencias de los nodos.

Con esta última forma de representación se ha trabajado el algoritmo que se presenta a continuación.

Una vez se haya leído un grafo y se tenga en la tabla se pueden encontrar los primeros conjuntos, los cuales no necesitan ser procesados dentro del algoritmo. El vacío es el primer conjunto en contarse debido a que siempre existirá sin importar el tamaño del grafo ni la cantidad de conjuntos que se puedan encontrar. El vacío es un conjunto con cero elementos y se clasifica como un conjunto de tamaño igual a cero.

A continuación también sin necesidad de entrar a los ciclos del algoritmo se pueden encontrar los conjuntos de tamaño igual a uno. Estos conjuntos serán cada uno de los nodos tomados en cuenta individualmente $\{1\},\{2\},\{3\},\ldots\{n\}$. Para este punto hay que aclarar que solo se toman en cuenta grafos simples (que no tienen dos aristas conectadas al mismo par de nodos ni aristas conectadas al mismo nodo en ambos extremos). En caso de que se aceptaran usar grafos que contengan bucles se tendrían que eliminar del conteo todos los conjuntos que contengan nodos con bucles debido a que estos nodos no son independientes de sí mismos y por tanto tampoco podrían pertenecer a conjuntos de mayor tamaño. En caso de aristas múltiples (si se tomarán en cuenta) bastaría tomar una sola arista para cada par de nodos adyacentes ya que no habría diferencia si un par de nodos está conectado por una o varias aristas, dicho par no será independiente ni será un subconjunto de un CI.

Otro tipo de grafos no simples que usualmente no se toman en cuenta dentro del estudio de los conjuntos independientes son los grafos dirigidos. Estos tipos de grafos se caracterizan por tener un orden por tanto el conjunto $\{a,b\} \neq \{b,a\}$ por lo que un par de nodos $a \rightarrow b$ se podría decir que $\{b,a\}$ es un conjunto independiente debido a que b no tiene adyacencia con a, sin embargo el caso contrario $\{a,b\}$ no sería independiente ya que a es dependiente de b.

Con lógica similar a estos casos foráneos al estudio se pueden estudiar los conjuntos independientes para grafos no simples. Continuando con el algoritmo siendo que los

grafos simples no contienen bucles la cantidad de conjuntos independientes con tamaño igual a uno será la misma cantidad de nodos que existan en el grafo.

Hasta este punto el conteo de conjuntos no ha entrado en ningún ciclo. Para el resto de CI ya es necesario recorrer los nodos buscando las no adyacencias. Para explicar más sencillamente el funcionamiento del algoritmo se usará un grafo de ejemplo.

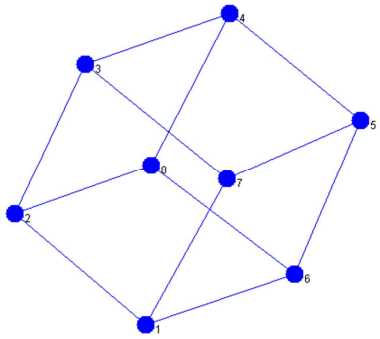


Fig. 7. Representación de un cubo en forma de grafo.

Este grafo será representado por la siguiente tabla:

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	1	1	0
2	1	0	1	0	0	0	0	1
3	0	1	0	1	0	0	1	0
4	0	0	1	0	1	0	0	1
5	0	0	0	1	0	1	1	0
6	1	0	0	0	1	0	0	1
7	1	0	1	0	1	0	0	0
8	0	1	0	1	0	1	0	0

Tabla 1. Representación del grafo.

Como se puede ver la tabla representativa es simétrica, esto ocurre para cualquier tipo de grafo simple por lo que se puede solo recorrer la mitad y así reducir el procesamiento. También es innecesario tomar en cuenta a la diagonal debido a que no habrá conexión alguna y como se mencionó ya sabemos que los conjuntos con tamaño igual a uno será la cantidad de nodos que tenga el grafo.

	1	2	3	4	5	6	7	8
1	0	1	0	0	0	1	1	0
2	1	0	1	0	0	0	0	1
3	0	1	0	1	0	0	1	0
4	0	0	1	0	1	0	0	1
5	0	0	0	1	0	1	1	0
6	1	0	0	0	1	0	0	1
7	1	0	1	0	1	0	0	0
8	0	1	0	1	0	1	0	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
2		1	0	0	0	0	1
3			1	0	0	1	0
4				1	0	0	1
5					1	1	0
6						0	1
7							0

Tablas 2. Seguimiento del algoritmo para la obtención de los CI.

La búsqueda de los CI empieza entonces de la parte superior de la tabla y se ejecuta recorriendo fila por fila en busca de las no adyacencias las cuales están representadas por ceros. Hasta este punto el conteo es de 9 CI (8 con tamaño 1 más el vacío). Cuando encuentra un cero se suma el contador en uno y salta a la fila del nodo donde se encontró el cero.

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
3			1	0	0	1	0

Tablas 3. Conteo de Ceros para la obtención de los CI.

En este segundo nivel se recorre igualmente en busca de ceros sin embargo una vez se encuentre una no adyacencia en este nivel no quiere decir que junto con 1 y 3 el nodo encontrado pueda ser independiente. Entonces una vez que se encuentre un cero en la fila del segundo nivel pasará a verificar si es independiente de los nodos de los niveles anteriores.

Para verificar que el nodo en cuestión puede formar un CI con los nodos de ambos niveles de la tabla se realiza una suma de los elementos de la columna de este nodo. Si

el resultado de esta suma es igual a cero entonces es un conjunto independiente y salta a la fila que representa a este nuevo nodo y el proceso se repite buscando nodos no adyacentes y verificando que estos sean independientes de los niveles anteriores

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
3			1	0	0	1	0
5					1	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
3			1	0	0	1	0
5					1	1	0

Tabla 4. Verificación de los CI.

Una vez en algún nivel se haya alcanzado el final de la fila, regresará a el nivel anterior en la posición donde se había quedado y continuará la búsqueda de no adyacencias para el resto de nodos. En este caso una vez que se encontró el conjunto $\{1,3,5,8\}$ se retornará al nivel tres y continuará la búsqueda hasta encontrar otro nodo no adyacente y como anteriormente verificando que este sea independiente de los niveles superiores.

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
3			1	0	0	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
3			1	0	0	1	0

Tabla 5. Verificación de los CI en niveles superiores.

El proceso continuará de esta forma agregando y quitando niveles de la tabla y verificando cuando exista un posible conjunto independiente para algún nivel.

A continuación se muestra parte del resto del procesamiento de la búsqueda:

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
4				1	0	0	1

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
4				1	0	0	1

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0
5					1	1	0

	2	3	4	5	6	7	8
1	1	0	0	0	1	1	0

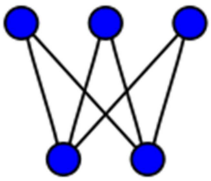
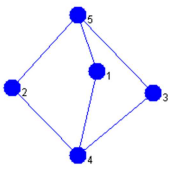
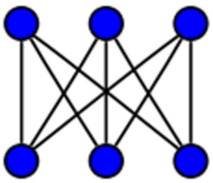
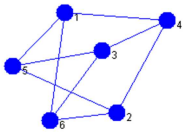
Tabla 6. Conclusión del algoritmo.

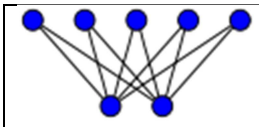
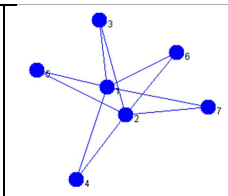
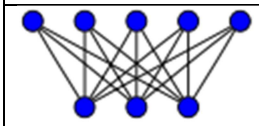
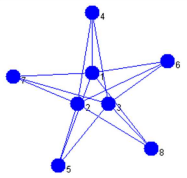
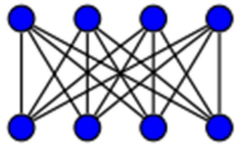
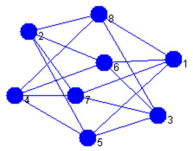
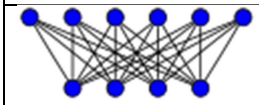
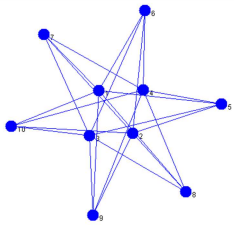
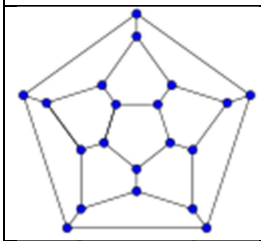
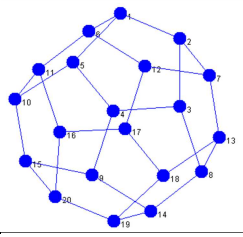
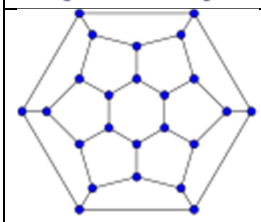
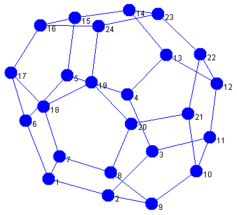
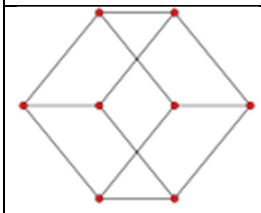
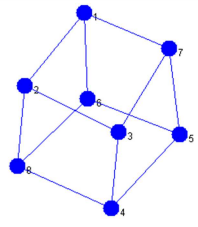
Lo que se presenta hasta aquí da como resultado todos los conjuntos independientes que contengan al nodo 1, posteriormente se continuará con la siguiente fila y se realizará el mismo procedimiento de búsqueda hasta que se llegue a la última fila.

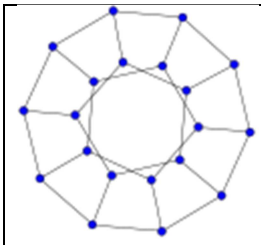
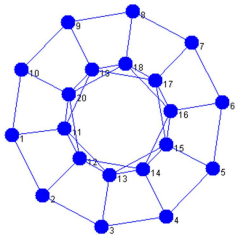
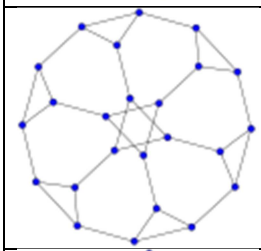
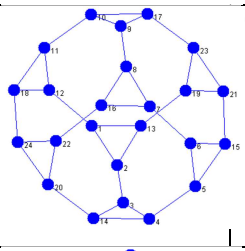
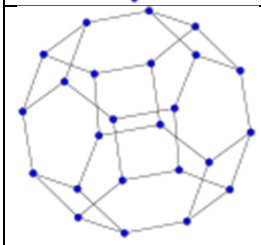
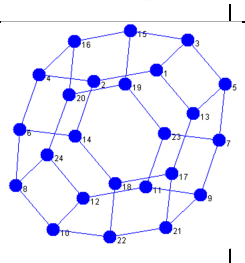
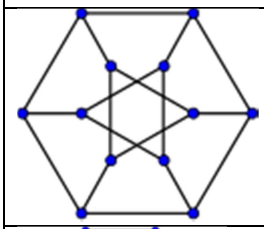
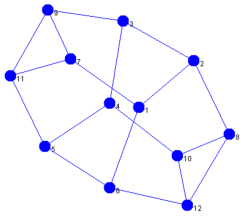
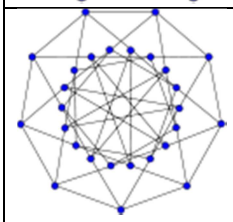
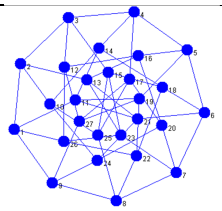
Al terminar toda la tabla con este ejemplo deberían dar 35 conjuntos independientes distribuidos de la siguiente forma: Tamaño 0=1, Tamaño 1=8,Tamaño 2=16,Tamaño 3=8,Tamaño 4=2.

3. Resultados

Como resultado de las pruebas de los CI tenemos las siguientes instancias de prueba:

Grafo	Generado por programa	Nombre	Cantidad de CI	Tiempo de ejecución
		K2,3	11 Tamaño 0=1,Tamaño 1=5,Tamaño 2=4,Tamaño 3=1	<1 milisegundo
		K3,3	15 Tamaño 0=1,Tamaño 1=6,Tamaño 2=6,Tamaño 3=2	<1 milisegundo

		K2,5	35 Tamaño 0=1,Tamaño 1=7,Tamaño 2=11,Tamaño 3=10, Tamaño 4=5, Tamaño 5=1	<1 milisegundo
		K3,5	39 Tamaño 0=1,Tamaño 1=8,Tamaño 2=13,Tamaño 3=11,Tamaño 4=5, Tamaño 5=1	<1 milisegundo
		K4,4	31 Tamaño 0=1, Tamaño 1=8, Tamaño 2=12, Tamaño 3=8, Tamaño 4=2	<1 milisegundo
		K4,6	79 Tamaño 0=1,Tamaño1=10,Tamaño 2=21,Tamaño 3=24,Tamaño 4=16,Tamaño 5=6,Tamaño 6=1	<1 milisegundo
		20-fullerene	5828 Tamaño 0=1, Tamaño 1=20, Tamaño 2=160, Tamaño 3=660,Tamaño 4=1510, Tamaño 5=1912, Tamaño 6=1240, Tamaño 7=320, Tamaño 8=5	447 ms
		24-fullerene	33327 Tamaño 0=1, Tamaño 1=24, Tamaño 2=240, Tamaño 3=1304, Tamaño 4=4212, Tamaño 5=8316, Tamaño 6=9918, Tamaño 7=6756, Tamaño 8=2292, Tamaño 9=264	18.31 segundos
		Cubo	35 Tamaño 0=1, Tamaño 1=8, Tamaño 2=16, Tamaño 3=8, Tamaño 4=2	<1 milisegundo

		Dodecaedro	3063 Tamaño 0=1, Tamaño 1=20,Tamaño 2=150, Tamaño 3=540, Tamaño 4=995, Tamaño 5=922, Tamaño 6=385, Tamaño 7=50	516 ms
		Cubo truncado	26441 Tamaño 0=1, Tamaño 1=24, Tamaño 2=240, Tamaño 3=1296, Tamaño 4=4092, Tamaño 5=7632, Tamaño 6=8056, Tamaño 7=4272, Tamaño 8=828	10.80 segundos
		Octaedro truncado	37076 Tamaño 0=1, Tamaño 1=24,Tamaño 2=240, Tamaño 3=1304, Tamaño 4=4218, Tamaño 5=8400, Tamaño 6=10356, Tamaño 7=7824, Tamaño 8=3555, Tamaño 9=960, Tamaño 10=168, Tamaño 11=24, Tamaño 12=2	21.28 segundos
		Grafo de Dürer	171 Tamaño 0=1, Tamaño 1=12, Tamaño 2=48, Tamaño 3=74, Tamaño 4=36	<1 milisegundo
		Grafo de Holt	47860 Tamaño 0=1,Tamaño 1=27,Tamaño 2=297,Tamaño 3=1737, Tamaño 4=5913, Tamaño 5=12042, Tamaño 6=14472, Tamaño 7=9693, Tamaño 8=3213, Tamaño 9=438,Tamaño 10=27	47.08 segundos

4. Conclusiones

Si bien este repositorio tiene por objetivo ser el contenedor de grafos ponderados también permite importar archivos con topologías complejas de grafos para el conteo de conjuntos independientes cuyo campo de aplicación se encuentra en las redes y en el análisis químico entre otros.

El proyecto realizado desde el punto de vista didáctico apoya a los docentes que imparten la materia de análisis y diseño de algoritmos, estructuras de datos y matemáticas discretas puesto que no existen herramientas donde los alumnos puedan diseñar sus grafos y posteriormente editarlos o modificarlos por lo que se considera que será muy útil para materias posteriores donde el número de nodos de un grafo sea muy muy grande como en aplicaciones de redes y cadenas poligonales en química.

La contribución que se brinda con el presente trabajo en el área de grafos permitirá abrir un espacio para la participación activa tanto de expertos, como del público en general que se dé la oportunidad de experimentar el diseño de distintas topologías de grafos, permitiendo que esta crezca y se difunda.

Por otro lado se propone generar ejercicios con html5 y css3 y así permitir que el sistema se enriquezca con mayor diversidad de tipos de ejercicios y lenguajes soportados.

Finalmente se establece como un trabajo futuro de gran importancia la implementación del proyecto para dispositivos móviles, ya que como se sabe es la tendencia en tecnología a nivel mundial

5. Referencias

[1] S. Miranda, Modelo para la generación automática de resúmenes abstractos basado en grafos conceptuales. 2013. Instituto Politécnico Nacional. México.121 p.

[2] Rafael López Bracho, María Paula Ortuño Sánchez, Un algoritmo paralelo para el problema del conjunto independiente, Revista de Matemática: Teoría y Aplicaciones. Vol. 7. No. 1-2. 2000.: 125-134 pp.

[3] K. XU. Benchmarks with Hidden Optimum Solutions for Graph Problems Disponible en: http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm. Consulta 23 de enero de 2015.

[4] Graph Theory, MathWorld. Disponible en: http://mathworld.wolfram.com/topics/GraphTheory.html. Consulta: 16 de enero de 2015.

[5] R. Guerequeta, A. Vallecillo, Técnicas de Diseño de Algoritmos. 2da edición. 1998. Servicio de Publicaciones de la Universidad de Málaga. España.

[6] R. Espinioza, Á. Lluch, Introducción a la teoría de grafos. 1997. Ediciones de la Universdad de Simón Bolívar. Colombia.

[7] R. Diestel, Graph Theory. 4ª edición. 2010. Springer. Estados Unidos.

[8] I. Kuckir, Graph drawer, Imágenes de grafos tomadas de http://g.ivank.net/. Consulta: 2011.

[9] R. L. Shackelford, Computing and algorithms. 1998. Addison-Wesley.

[10] S. Baase, V. Gelder, Algoritmos computacionales. 3ra edición. 2002. Addison-Wesley.

[11] C. Greenhill, “The Complexity of Counting Colourings and Independent Sets in Sparse Graphs and Hypergraphs”. Computational Complexity. Vol. 9. No. 1. 2000.

[12] S. P. Vadhan, The Complexity of Counting. Undergraduate thesis. Hardvard University. 1995.

[13] M. Xiao, H. Nagamochi, An exact algorithm for maximum independent set in degree-5 graphs. Discrete Applied Mathematics. 2014.

[14] N. Nash, S. Lelait, D. Gregg, Efficiently Implementing Maximum Independent Set Algorithms on Circle Graphs. ACM Journal of Experimental Algorithmics. Vol. 13. December 2008.

6. Autores

I.C.C. Juan Antares Perdomo Flandez obtuvo su título de Ingeniería en Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación

M. en C. Pedro Bello López obtuvo su título de Maestría en Ciencias de la Computaciónen la Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación.

M. en C. Meliza Contreras González obtuvo su título de Maestría en Ciencias de la Computación en la Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación.

Brayan Chavez Benavides es estudiante de la Ingeniería en Ciencias de la Computación por Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación, y participa en el proyecto VIEP 2015, *"Aplicaciones de combinatoria en modelos químicos y financieros"*.