



Instituto Tecnológico de Costa Rica

Unidad de Computación

Proyecto #1

Análisis de Algoritmos

Profesora:

Ana Lorena Valerio Solís

Estudiantes:

Roosevelt Alejandro Pérez González

José Andrés Lorenzo Segura

Arnold Jafeth Alvares Rojas

Campus San Carlos

I Semestre 2024

Índice de contenidos

Índice de figuras	i
Introducción	1
Análisis del problema.....	1
Solución del problema	8
Análisis de resultados.....	13
Medición Empírica.....	13
Notación Bron - Kerbosch	16
Notación Búsqueda Local.....	17
Medición analítica	20
Bron - Kersboch.....	20
Búsqueda Local.....	21
Conclusiones	23
Bibliografía.....	24
Minutas	25
Bitácora de Actividades.....	30
Coevaluación	35
Autoevaluación	36

Índice de figuras

Figura 1: Primer grafo introducido en el sistema	4
Figura 2: Segundo grafo introducido en el sistema	4
Figura 3: Séptimo grafo introducido en el sistema	5
Figura 4: Décimo grafo introducido en el sistema	6
Figura 5: Código para generar diagramas de grafos	7

Figura 6: Inicialización de los conjuntos vacíos	8
Figura 7: Algoritmo de la Exploración recursiva del espacio de soluciones	9
Figura 8: Finalización de la llamada recursiva	9
Figura 9: Algoritmo para calcular la intersección entre dos algoritmos	10
Figura 10: Optimización del bucle de optimización	10
Figura 11: Método retainAll	11
Figura 12: Inicialización del algoritmo de búsqueda Local	11
Figura 13: Iteración de búsqueda local	12
Tabla 1: Medición empírica del algoritmo Bron - Kersboch.....	13
Tabla 2: Medición empírica del algoritmo de Búsqueda Local.....	13
Tabla 3: Factor de crecimiento del algoritmo Bron – Kerbosch.....	15
Tabla 4: Factor de crecimiento del algoritmo Bron - Kerbosch usando solo los grafos de 20 vértices	15
Tabla 5: Factor de crecimiento del algoritmo Búsqueda Local.....	16
Tabla 6: Factor de crecimiento del algoritmo Búsqueda Local usando solo los grafos de 20 vértices	16
Tabla 7: Comportamiento de los Algoritmos	18
Tabla 8: Comportamiento de los Algoritmos con 20 vértices y aristas variantes.	18

Introducción

Uno de los problemas más trascendentales en la teoría de grafos es el de encontrar el clique máximo de un grafo. Este problema, que, a primera vista podría parecer simple posee una gran complejidad computacional. Este problema ha sido objeto de estudio debido a su aplicación en una amplia gama de disciplinas, que van desde la informática hasta la biología molecular.

El clique máximo de un grafo se refiere a un subconjunto de vértices (nodos) de un grafo que están todos conectados entre sí, la conexión entre todos estos grafos se realiza mediante una arista.

Este proyecto se enfocará en implementar dos posibles algoritmos para afrontar los desafíos que presenta este problema. Los algoritmos por implementar son el de Bron - Kerbosch y el de Búsqueda Local. Estos algoritmos representan dos estrategias distintas para encontrar la solución de este problema. El algoritmo de Bron – Kerbosch se basa en la técnica de backtracking, explorando exhaustivamente todas las posibles combinaciones de vértices para determinar el clique máximo.

Por otro lado, el de búsqueda local se centra en mejorar iterativamente soluciones existentes mediante el movimiento a través del espacio de soluciones vecinas.

A lo largo de este proyecto se explorará en detalle la implementación y el rendimiento de estos dos enfoques algorítmicos en la resolución del problema del clique máximo.

Se analizará sus complejidades computacionales, se pondrán a prueba con una variedad de grafos, y se compararán entre estos mismos.

Al final de la investigación se espera obtener información valiosa sobre sus aplicaciones prácticas y su eficacia en diferentes escenarios.

Análisis del problema

Según [1] el objetivo del problema del clique máximo es encontrar un clique máximo en un grafo no dirigido arbitrariamente. Se menciona que este problema es computacionalmente equivalente a otros problemas importantes en teoría de grafos, como el problema del conjunto independiente máximo (o conjunto estable) y el problema de la cobertura mínima de vértices. Dado que estos son problemas NP-duros, no se esperan algoritmos polinomiales para resolverlos. Sin embargo, debido a que estos problemas tienen varias

aplicaciones prácticas relevantes, existe un gran interés en desarrollar algoritmos exactos rápidos para instancias pequeñas.

Según [2] y [3] este problema se clasifica dentro de los problemas de NP-duros, los cuales son muy complicados de resolver. Debido a esta complejidad, es necesario desarrollar algoritmos heurísticos y/o metaheurísticos para lograr una solución cercana al óptimo a un tiempo razonable. Según [2] este problema tiene aplicaciones reales como lo son: La teoría de códigos, diagnósticos de errores, visión computacional, análisis de agrupamiento, recuperación de información, aprendizaje automático, entre otros.

Durante la fase inicial del proyecto, es decir, la investigación de los posibles algoritmos a implementar, se han encontrado diferentes opciones, las cuales se van a enumerar a continuación.

- Algoritmo: Búsqueda local
- Algoritmos Genéticos
- Algoritmo: Bron - Kerbosch
- Algoritmo: Búsqueda tabú
- Algoritmo: Optimización por colonias de hormigas
- Algoritmo: Ramificación y Acota

Luego de la valoración de todos los posibles algoritmos a implementar, se seleccionaron los algoritmos de Bron - Kerbosch y Búsqueda Local.

A través de la revisión de los datos obtenidos en [4] y en [5] se puede hacer una comparativa entre el algoritmo de búsqueda local, el cual fue seleccionado para el proyecto, con respecto al algoritmo genético, esto debido a que en ambos documentos los algoritmos realizan una serie de pruebas brindadas por Center for Discrete Mathematics and Theoretical Computer Science (DINAMICS). En el cual se toma el tamaño del clique a través de instancias las cuales brinda el DINAMICS con su respectivo valor a aproximarse, viéndose reflejado en el cuadro del algoritmo de búsqueda local el cómo llega a la solución idónea en la mayoría de los casos, siendo la prueba de brock en la única que se logró destacar un poco mejor el algoritmo genético con respecto al de búsqueda local.

A su vez en [6] se aprecia un análisis del algoritmo de colonia de hormigas con el cual se realizan las mismas pruebas y con el cual se denota que llega a ser muy semejante a la

eficiencia del algoritmo genético, sin embargo, este mismo llega a superarlo en la mayoría de pruebas acercándose más al valor exacto del clique máximo que se espera, así mismo en [7] se menciona que el comportamiento y la naturaleza de la búsqueda tabú es muy semejante al algoritmo genético y al de colonia, sin embargo, no se hace mención de cuan eficiente es exactamente siguiendo lo estipulado por las pruebas hechas a los algoritmos ya mencionados.

Con respecto a la estructura de los códigos utilizados, tanto el código de Bron como el de búsqueda local utilizan una estructura de grafo, sin embargo, estas se diferencian en cómo se compone su estructura de datos siendo que en el grafo de Bron se utiliza una lista de adyacencia, que es una colección de conjuntos donde cada conjunto representa los vértices adyacentes a un vértice en particular y en el grafo de búsqueda local se utiliza una matriz de adyacencia, que es una matriz booleana donde cada elemento indica si hay una conexión entre dos vértices.

En el caso del acceso a la información de conexión, el grafo de Bron, para saber si hay una conexión entre dos vértices, consulta el conjunto de adyacencia del vértice correspondiente y se verifica si el otro vértice está presente en ese conjunto, por otro lado el grafo de búsqueda local, para saber si hay una conexión entre dos vértices, consulta la matriz de adyacencia y se verifica el valor booleano correspondiente a la posición de los vértices en la matriz.

Por último, la lista de adyacencia del grafo de Bron es más eficiente en términos de espacio cuando el grafo es disperso, es decir, cuando la mayoría de los vértices tienen pocos vecinos y la matriz de adyacencia del grafo de búsqueda local es más eficiente en términos de espacio cuando el grafo es denso, es decir, cuando la mayoría de los vértices están conectados entre sí.

A continuación, se presentarán cuatro diagramas de grafos introducidos para probar los algoritmos. Los nodos en color verde son los que forman el clique máximo del grafo.

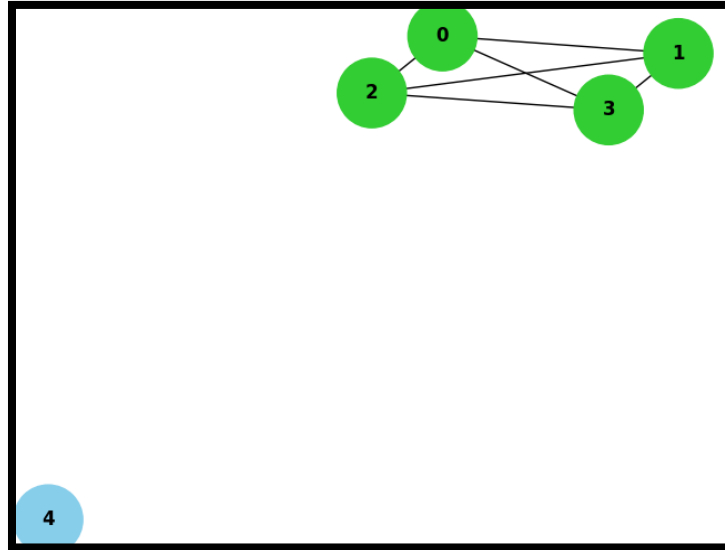


Figura 1: Primer grafo introducido en el sistema

En esta primera prueba de los algoritmos implementados, se ha introducido un grafo de cinco vértices con seis aristas. Las seis aristas se han utilizado para formar un clique de 4 vértices. De hecho, si nos basamos en la siguiente formula

$$\frac{x * (x - 1)}{2} = \text{aristas del clique de } x \text{ vertices}$$

podemos apreciar que en ese clique se utilizaron todas las posibles aristas. El sistema deberá de imprimir que el clique máximo de esta prueba está compuesto por los vértices [0,1,2,3].

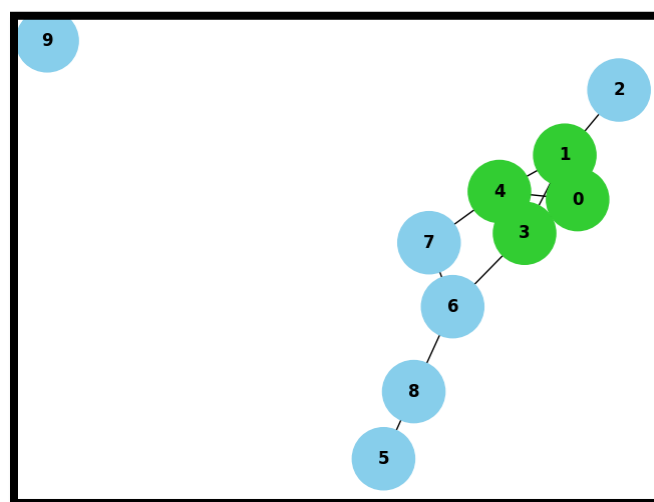


Figura 2: Segundo grafo introducido en el sistema

La segunda prueba introducida en el sistema se trata de un grafo con 10 vértices con 12 aristas. Nuevamente se ha elegido un clique de 4 vértices, el cual, ya sabemos que tiene seis aristas. Las otras seis fueron elegidas están en las conexiones [(2,1), (7,4), (6,3), (6,8), (5,8)]. El clique esperado de esta prueba está formado por los vértices [0,1,3,4].

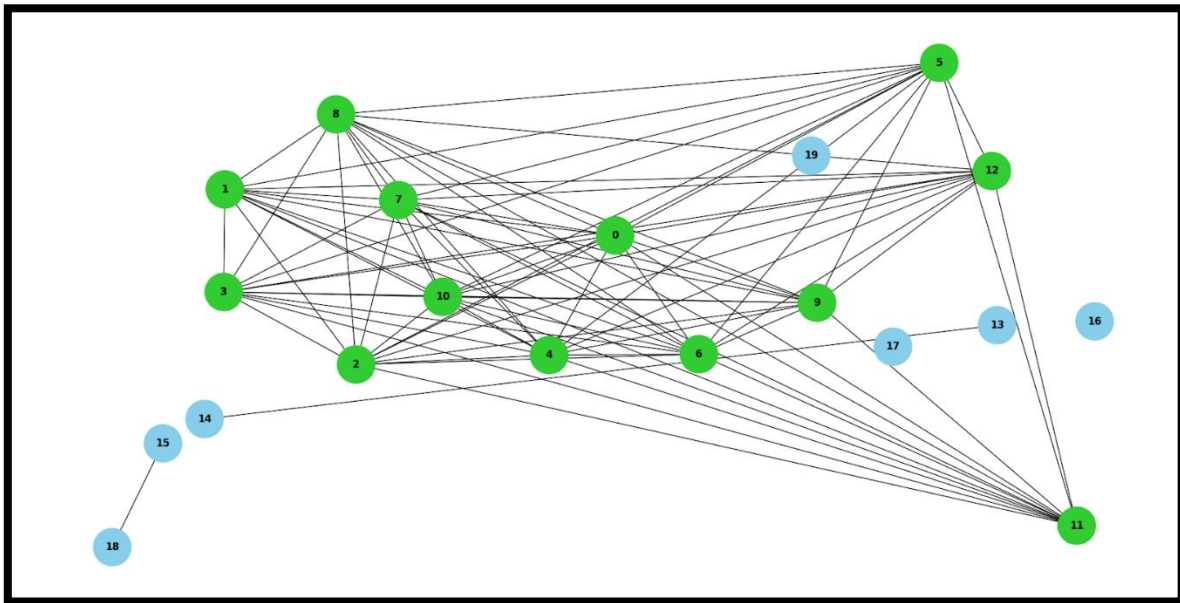


Figura 3: Séptimo grafo introducido en el sistema

En la séptima prueba podemos apreciar un grafo más grande que los otros dos vistos con anterioridad. En esta prueba se ha introducido un grafo de 20 vértices con 80 aristas. En esta prueba se buscó realizar un clique con el máximo número de aristas posibles. Basándonos en la formula presentada con anterioridad, podemos notar que el clique máximo a realizar en este grafo cuenta con 78 aristas, que es el resultado de hacer un clique de 13 vértices.

$$\frac{13 * (12)}{2} = 78$$

Por ende, se realizó la formación de ese clique que va desde el vértice 0 hasta el 12. El clique esperado es [0,1,2,3,4,5,6,7,8,9,10,11,12] y las dos conexiones restantes son [(13, 14), (18, 15)].

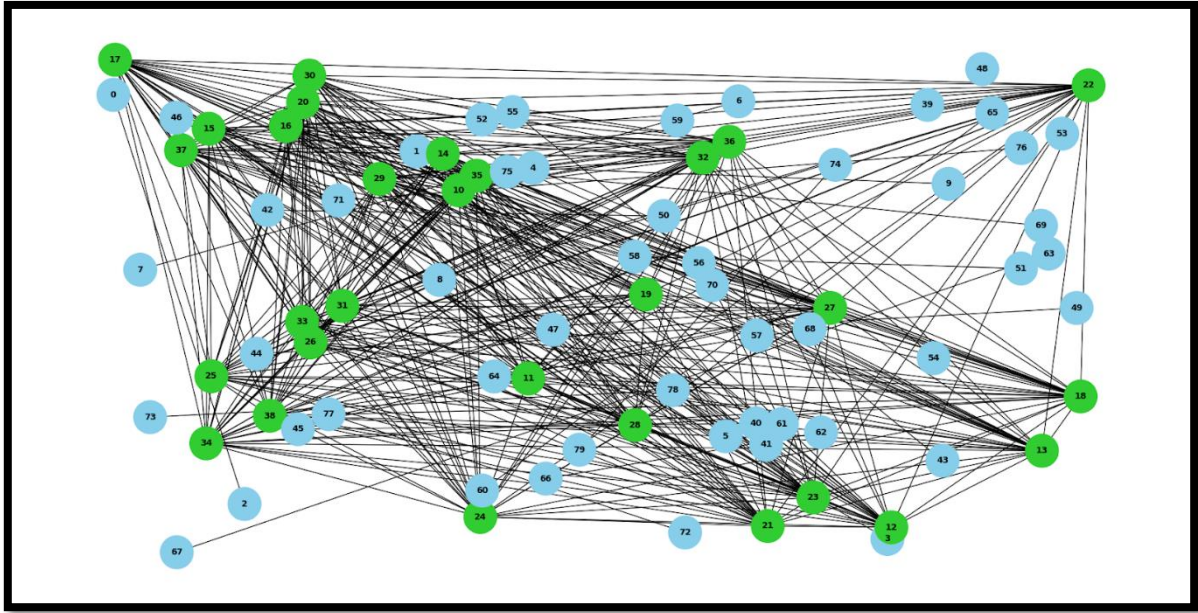


Figura 4: Décimo grafo introducido en el sistema

La ultima prueba realizada en los algoritmos desarrollados es un grafo que cuenta con 80 vértices y 400 aristas. En este grafo se realizó un clique de 28 vértices, es decir, se usaron para definir este clique 378 aristas.

$$\frac{28 * 27}{2} = 378$$

El clique que se espera de grafo es [10,11,12...37,38].

Las conexiones restantes que son 22, son las siguientes.

[(0, 2), (3, 8), (4, 9), (6, 7), (1, 5), (40, 45), (41, 46), (42, 50), (43, 55), (44, 49), (51, 58), (52, 59), (60, 65), (61, 70), (62, 66), (63, 67), (64, 68), (69, 75), (71, 76), (72, 77), (73, 78), (74, 79)].

Este grafo, siendo el de mayor tamaño de toda la investigación, se espera que presente mayores desafíos para los algoritmos implementados.

Estos diagramas fueron realizados con Python, con las librerías networkx y matplotlib.

Un ejemplo de cómo realizar estos diagramas es el siguiente

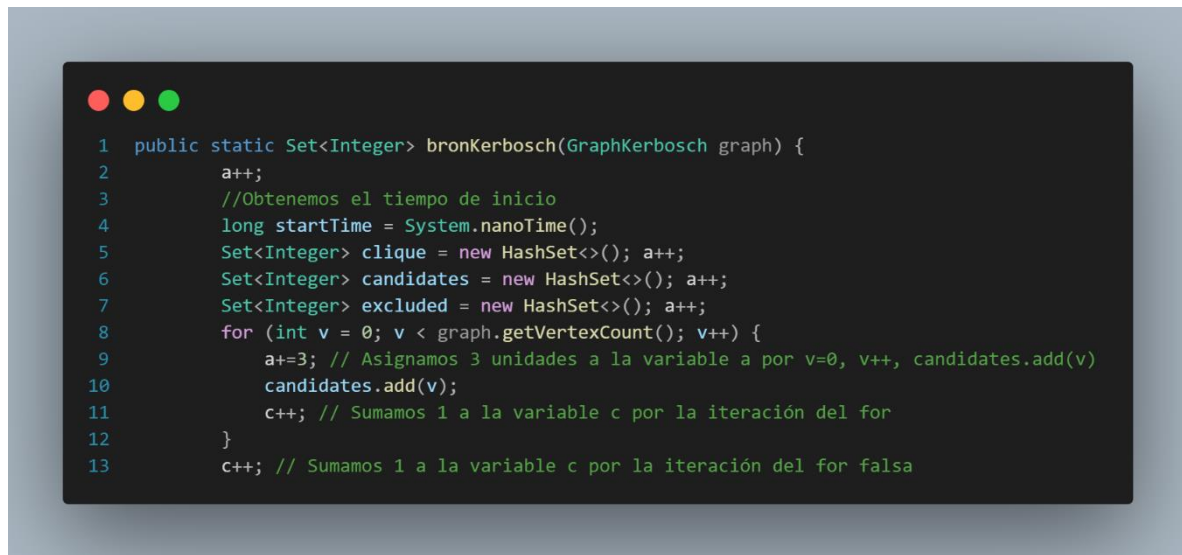
```
1 import networkx as nx #Citar estas librerias
2 import matplotlib.pyplot as plt #Citar estas librerias
3 # Crear un grafo
4 G = nx.Graph()
5 for i in range(20):
6     G.add_node(i)
7 # Generamos el clique de nuestro ejercicio
8 for i in range(13):
9     for j in range(i + 1, 13):
10         G.add_edge(i, j)
11 G.add_edge(13,14);
12 G.add_edge(18,15);
13 # Encontrar todos los cliques en el grafo
14 cliques = list(nx.find_cliques(G))
15 # Supongamos que queremos colorear el clique máximo
16 max_clique = max(cliques, key=len)
17 # Configurar colores para los nodos
18 node_color = ['skyblue' if not node in max_clique else 'limegreen' for node in G.nodes()]
19 # Dibujar el grafo
20 pos = nx.random_layout(G) # Posicionamiento de los nodos
21 nx.draw(G, pos, with_labels=True, node_color=node_color, node_size=2000, font_size=12, font_weight='bold')
22 plt.show()
```

Figura 5: Código para generar diagramas de grafos

Solución del problema

En cuanto a la lógica detrás del código de Bron - Kerbosch se observan los siguientes puntos claves:

Inicialización: Se inicializan tres conjuntos vacíos: clique, candidates y excluded. El conjunto candidates contiene todos los vértices del grafo.

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and shows the initialization of the Bron-Kerbosch algorithm. It includes comments in Spanish and uses standard Java syntax for sets and loops. The code is as follows:

```
1 public static Set<Integer> bronKerbosch(GraphKerbosch graph) {  
2     a++;  
3     //Obtenemos el tiempo de inicio  
4     long startTime = System.nanoTime();  
5     Set<Integer> clique = new HashSet<>(); a++;  
6     Set<Integer> candidates = new HashSet<>(); a++;  
7     Set<Integer> excluded = new HashSet<>(); a++;  
8     for (int v = 0; v < graph.getVertexCount(); v++) {  
9         a+=3; // Asignamos 3 unidades a la variable a por v=0, v++, candidates.add(v)  
10        candidates.add(v);  
11        c++; // Sumamos 1 a la variable c por la iteración del for  
12    }  
13    c++; // Sumamos 1 a la variable c por la iteración del for falsa
```

Figura 6: Inicialización de los conjuntos vacíos

Exploración recursiva: Se explora recursivamente el espacio de soluciones. En cada llamada recursiva, se elige un vértice de los candidatos, se agrega al conjunto clique y se actualiza el conjunto candidates para contener sólo los vértices adyacentes al vértice elegido que no están en el conjunto excluded. Luego, se llama recursivamente al algoritmo con estos conjuntos actualizados.

```

1 private static void bronKerbosch(GraphKerbosch graph, Set<Integer> clique, Set<Integer> candidates, Set<Integer> excluded) {
2     a+=4; // Asignamos 4 unidades a la variable a por los 4 parámetros de la función
3     c++; // Sumamos 1 a la variable c por la condición del if
4     if (candidates.isEmpty() && excluded.isEmpty()) {
5         c++; // Sumamos 1 a la variable c por la condición del if anidado
6         if (clique.size() > maxCliqueKersboch.size()) {
7             maxCliqueKersboch = new HashSet<>(clique);
8             a++; // Sumamos 1 a la variable a por la creación de maxCliqueKersboch
9         }
10        return;
11    }
12
13    List<Integer> candidatesList = new ArrayList<>(candidates);
14    a++; // Sumamos 1 a la variable a por la creación de candidatesList
15    for (Integer v : candidatesList) {
16        Set<Integer> neighbors = graph.getNeighbors(v);
17        a++; // Sumamos 1 a la variable a por la creación de neighbors
18        Set<Integer> newClique = new HashSet<>(clique);
19        a++; // Sumamos 1 a la variable a por la creación de newClique
20        newClique.add(v);
21        a++; // Sumamos 1 a la variable a por la adición de v a newClique
22        bronKerbosch(graph, newClique, intersection(candidates, neighbors), intersection(excluded, neighbors));
23        candidates.remove(v);
24        excluded.add(v);
25        a+=2; // Sumamos 2 a la variable a por la eliminación de v de candidates y la adición de v a excluded
26    }
27 }

```

Figura 7: Algoritmo de la Exploración recursiva del espacio de soluciones

Caso base: La recursión termina cuando no quedan vértices en el conjunto candidates y excluded. En este punto, se verifica si el conjunto clique es un clique válido. Si lo es y es más grande que el clique máximo encontrado hasta el momento, se actualiza el clique máximo.

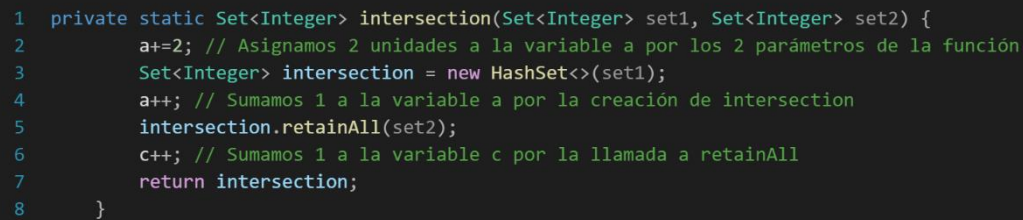
```

1 private static void bronKerbosch(GraphKerbosch graph, Set<Integer> clique, Set<Integer> candidates, Set<Integer> excluded) {
2     a+=4; // Asignamos 4 unidades a la variable a por los 4 parámetros de la función
3     c++; // Sumamos 1 a la variable c por la condición del if
4     if (candidates.isEmpty() && excluded.isEmpty()) {
5         c++; // Sumamos 1 a la variable c por la condición del if anidado
6         if (clique.size() > maxCliqueKersboch.size()) {
7             maxCliqueKersboch = new HashSet<>(clique);
8             a++; // Sumamos 1 a la variable a por la creación de maxCliqueKersboch
9         }
10        return;
11    }

```

Figura 8: Finalización de la llamada recursiva

Intersección de conjuntos: La función intersection se utiliza para calcular la intersección entre dos conjuntos.



```

1 private static Set<Integer> intersection(Set<Integer> set1, Set<Integer> set2) {
2     a+=2; // Asignamos 2 unidades a la variable a por los 2 parámetros de la función
3     Set<Integer> intersection = new HashSet<>(set1);
4     a++; // Sumamos 1 a la variable a por la creación de intersection
5     intersection.retainAll(set2);
6     c++; // Sumamos 1 a la variable c por la llamada a retainAll
7     return intersection;
8 }

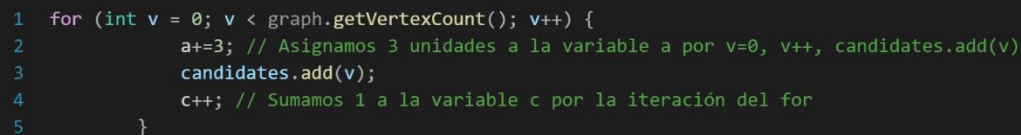
```

Figura 9: Algoritmo para calcular la intersección entre dos algoritmos

Las mejoras que se implementan conforme se ajustaba el funcionamiento del código fueron:

Optimización del bucle de inicialización: En lugar de iterar sobre todos los vértices del grafo para agregarlos al conjunto candidates, se utiliza un bucle for simple que agrega los vértices al conjunto candidates durante la inicialización del algoritmo.

Esto es, en lugar de agregar uno a uno los



```

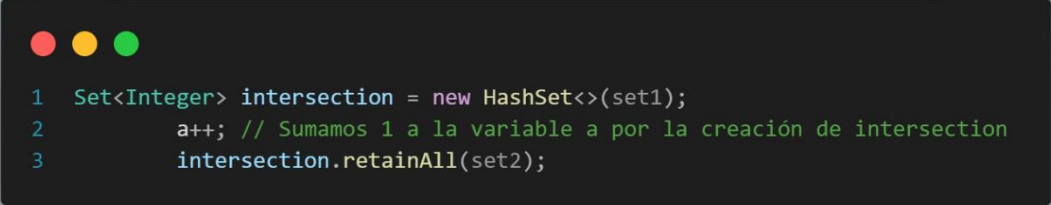
1 for (int v = 0; v < graph.getVertexCount(); v++) {
2     a+=3; // Asignamos 3 unidades a la variable a por v=0, v++, candidates.add(v)
3     candidates.add(v);
4     c++; // Sumamos 1 a la variable c por la iteración del for
5 }

```

Figura 10: Optimización del bucle de optimización

Optimización del manejo de conjuntos: Se utilizan conjuntos (HashSet) para almacenar los vértices y sus adyacencias, lo que permite una búsqueda eficiente y la eliminación rápida de vértices durante el proceso de backtracking.

Mejora en el cálculo de intersección: La función intersection se implementa de manera eficiente utilizando el método retainAll de la clase HashSet, lo que reduce el tiempo de cálculo de la intersección entre conjuntos.



```

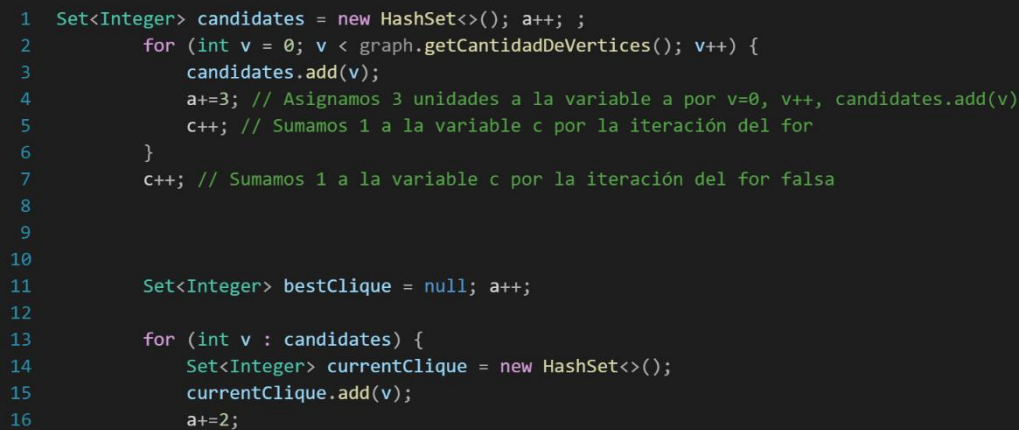
1 Set<Integer> intersection = new HashSet<>(set1);
2     a++; // Sumamos 1 a la variable a por la creación de intersection
3     intersection.retainAll(set2);

```

Figura 11: Método retainAll

Para el caso de la lógica utilizada en búsqueda local se tratan los siguientes puntos claves:

Inicialización: Se inicializa un conjunto currentClique con el primer vértice como punto de partida. También se inicializa un conjunto bestClique para mantener el mejor clique encontrado hasta el momento.



```

1 Set<Integer> candidates = new HashSet<>(); a++; ;
2     for (int v = 0; v < graph.getCantidadDeVertices(); v++) {
3         candidates.add(v);
4         a+=3; // Asignamos 3 unidades a la variable a por v=0, v++, candidates.add(v)
5         c++; // Sumamos 1 a la variable c por la iteración del for
6     }
7     c++; // Sumamos 1 a la variable c por la iteración del for falsa
8
9
10
11     Set<Integer> bestClique = null; a++;
12
13     for (int v : candidates) {
14         Set<Integer> currentClique = new HashSet<>();
15         currentClique.add(v);
16         a+=2;

```

Figura 12: Inicialización del algoritmo de búsqueda Local

Búsqueda Local: Se realiza una búsqueda local iterativa, donde en cada iteración se intenta mejorar la clique actual agregando o eliminando un vértice. Se repite este proceso hasta que no se puedan realizar más mejoras.

Agregar vértices: En cada iteración, se verifica si agregar un vértice no presente en la clique actual resultaría en un clique válido. Si es así y la nueva clique es más grande que la mejor encontrada hasta el momento, se actualiza la mejor clique.

```

1  for (int u : candidates) {
2      c++; // Sumamos 1 a la variable c por la condición del if
3      if (currentClique.contains(u)) continue; // Evita agregar vértices ya presentes en el clique
4      boolean isConnectedToAll = true;
5      a++; // Asignamos 1 unidad a la variable a por la creación de la variable isConnectedToAll
6      for (int w : currentClique) {
7          c++; // Sumamos 1 a la variable c por la condición del if
8          if (!graph.tieneConexion(u, w)) {
9              isConnectedToAll = false;
10             a++; // Sumamos 1 a la variable a por la asignación de false a isConnectedToAll
11             break;
12         }
13     }
14     if (isConnectedToAll) {
15         c++; // Sumamos 1 a la variable c por la condición del if
16         currentClique.add(u);
17         a++; // Sumamos 1 a la variable a por la adición de u a currentClique
18     } c++; // Sumamos 1 a la variable c por la condición del if
19 }
20
21 if (bestClique == null || bestClique.size() < currentClique.size()) {
22     c+=2; // Sumamos 2 a la variable c por la condición del if
23     bestClique = currentClique;
24     a++; // Sumamos 1 a la variable a por la asignación de currentClique a bestClique
25 }

```

Figura 13: Iteración de búsqueda local

Criterio de parada: El algoritmo termina cuando no se pueden realizar más mejoras en la clique actual.

Las mejoras que se implementan conforme se ajustaba el funcionamiento del código fueron:

Optimización de la verificación de cliques: Se utiliza un método eficiente para verificar si un conjunto de vértices forma un clique, lo que reduce el tiempo de cálculo y mejora la eficiencia del algoritmo.

Exploración local eficiente: El algoritmo realiza exploración local eficiente, intentando agregar y eliminar vértices de la clique actual de manera inteligente para mejorar la solución.

Análisis de resultados

Al realizar la toma de datos no se presentó ningún inconveniente, se pudieron obtener 2 algoritmos para realizar la comparación entre los mismos y esto a su vez se desarrolló con pequeños inconvenientes iniciales al ejecutar el código que fueron corregidos logrando la toma de los datos que se solicitaban.

A continuación, se mostrarán las tablas correspondientes a la medición empírica, donde se encuentran los datos de los tiempos de ejecución, cantidad de comparaciones, cantidad de asignaciones, cantidad de líneas de código y el total de líneas ejecutadas en cada prueba de ambos algoritmos.

Nótese que cuando los vértices son iguales, la variante son la cantidad de aristas que tiene cada grafo.

Medición Empírica

Vertices	Bron-K				
	Asignaciones	Comparaciones	Cantidad de líneas ejecutadas	Tiempo de ejecucion	Cantidad de líneas de codigo
5	279	57	336	0 ms	25
10	464	101	565	0 ms	
20	1206	254	1460	0 ms	
20	2439	484	2923	1 ms	
20	1731	357	2088	1 ms	
20	4232	819	5051	1 ms	
20	131264	24628	155892	37 ms	
40	525276	98533	623809	78 ms	
60	67109762	12853142	79962904	3329 ms	
80	8589935965	1610613064	10200549029	442645 ms	

Tabla 1: Medición empírica del algoritmo Bron - Kersboch

Vertices	Busqueda local				
	Asignaciones	Comparaciones	Cantidad de líneas ejecutadas	Tiempo de ejecucion	Cantidad de líneas de codigo
5	69	97	166	36 ms	20
10	234	325	559	0 ms	
20	864	1263	2127	0 ms	
20	864	1357	2221	1 ms	
20	865	1321	2186	1 ms	
20	864	1858	2722	0 ms	
20	864	2201	3065	0 ms	
40	3324	6413	9737	1 ms	
60	7386	15847	23233	4 ms	
80	13045	30943	43988	7 ms	

Tabla 2: Medición empírica del algoritmo de Búsqueda Local

Como se observa en las Tablas 1 y 2, el algoritmo de búsqueda local (BL) muestra un rendimiento claramente superior en términos de asignaciones comparado con el algoritmo de Bron-Kerbosch (BK). Específicamente, en grafos pequeños de apenas cinco vértices, el algoritmo BK realiza casi cuatro veces más asignaciones que el algoritmo BL, lo que sugiere una eficiencia considerablemente menor en el manejo de grafos densos incluso a pequeña escala. Además, en cuanto al número de líneas ejecutadas, el algoritmo de BK se muestra 2.02 veces más intensivo que el algoritmo BL, evidenciando un uso más pesado de recursos computacionales para tareas equivalentes.

Estas diferencias son particularmente notables en el análisis de las comparaciones efectuadas desde grafos de 5 vértices hasta grafos de 20 vértices con 80 aristas (el último grafo de 20 vértices analizado), periodo durante el cual el algoritmo de Bron-Kerbosch (BK) aún demuestra cierta superioridad en algunos aspectos, posiblemente debido a su capacidad para identificar cliques en entornos más controlados o menos extensos. No obstante, a partir de este punto, específicamente en grafos de más de 20 vértices con 80 aristas, se observa un deterioro notable en el rendimiento del algoritmo BK, lo que refleja su menor escalabilidad en comparación con el algoritmo BL, especialmente en entornos con una mayor densidad y complejidad de conexiones.

Es importante destacar la superioridad general del algoritmo de búsqueda local (BL) sobre el de Bron-Kerbosch (BK), particularmente evidente en el grafo de mayor tamaño evaluado, que cuenta con 80 vértices y 400 aristas. En este caso, el algoritmo BK realiza aproximadamente 658,484.93 veces más asignaciones que el algoritmo BL. Además, la diferencia en el tiempo de ejecución es significativa, alcanzando hasta siete minutos a favor del algoritmo de búsqueda local. Esta eficiencia notoria del algoritmo BL no solo implica un menor consumo de recursos, sino que también se traduce en una capacidad superior para manejar grandes volúmenes de datos, una ventaja crucial en aplicaciones de análisis de redes grandes como las encontradas en la bioinformática y las redes sociales.

Estos hallazgos destacan la importancia de elegir el algoritmo adecuado basándose en el tamaño y las características específicas del grafo a analizar, así como sugieren áreas potenciales de mejora para el algoritmo de Bron-Kerbosch, particularmente en su aplicación a grandes estructuras de datos.

Tamaño del grafo en vértices	Bron-K				
	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor Tiempo de ejecución
10/5	2	1,663082437	1,771929825	1,681547619	0
20/10 (40 Arcos)	2	3,730603448	3,534653465	3,695575221	0
40/20	2	303,4523397	1,905511811	2,002054795	78
60/20	3	38769,35991	36003,19888	38296,40996	42,67948718
80/40	2	16353,18569	16345,92537	16352,03889	5674,935897
80/20	4	4962412,458	4511521,188	4885320,416	442645
80/10	8	18512793,03	15946664	18054069,08	0
80/5	16	30788300,95	28256369,54	30358776,87	0

Tabla 3: Factor de crecimiento del algoritmo Bron – Kerbosch

Se observa en la tabla del factor de crecimiento del algoritmo de Bron-Kerbosch (BK) que este exhibe una alta complejidad algorítmica, sugiriendo inicialmente un comportamiento exponencial o factorial, ambos conocidos por sus demandantes tiempos de ejecución. Recordemos que, según los datos recabados, este algoritmo tardó más de siete minutos en completar la tarea en el grafo más grande analizado.

La literatura especializada clasifica la complejidad de este algoritmo en $O(3^{n/3})$, indicativo de su naturaleza intensiva en recursos. Sin embargo, los resultados mostrados en nuestra tabla sugieren que la implementación realizada por los estudiantes podría estar operando con una complejidad que ronda entre lo exponencial y lo factorial, lo cual es considerablemente más demandante de lo teóricamente esperado.

Para efecto práctico de este proyecto, se dirá que este algoritmo tiene una complejidad factorial, esto para tomar el peor caso.

Tamaño del grafo en arcos usando solo las pruebas de 20 vértices	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor Tiempo de ejecución
30/20	1,5	2,02238806	1,905511811	2,002054795	0
40/20	2	1,435323383	1,405511811	1,430136986	0
50/20	2,5	3,509121061	3,224409449	19,88582677	0
80/20	4	108,8424544	96,96062992	106,7753425	0
80/50	1,6	31,01701323	30,07081807	30,86359137	37

Tabla 4: Factor de crecimiento del algoritmo Bron - Kerbosch usando solo los grafos de 20 vértices

Nuevamente las tablas demuestran que el algoritmo presenta una alta complejidad que ronda entre

Tamaño del grafo en vértices	Busqueda Local				
	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor Tiempo de ejecución
10/5	2	3,391304348	3,350515464	3,36746988	0
20/10 (40 Arcos)	2	3,692307692	4,064615385	3,910554562	0
40/20	2	3,842774566	4,854655564	4,454254346	1
60/20	3	8,538728324	11,99621499	10,62808783	4
80/40	2	3,924488568	4,825042882	4,517613228	7
80/20	4	15,08092486	23,42392127	20,12259835	7
80/10	8	55,74786325	95,20923077	78,69051878	0
80/5	16	189,057971	319	264,9879518	0,1944

Tabla 5: Factor de crecimiento del algoritmo Búsqueda Local

La tabla de factores de crecimiento del algoritmo de Búsqueda Local (BL) destaca uno de los hallazgos más significativos de esta investigación: la complejidad algorítmica cuadrática del algoritmo, es decir, $O(n^2)$. Este resultado es fundamental, ya que indica que estamos frente a un algoritmo con un tiempo polinomial aceptable. En comparación con otros algoritmos de complejidades exponenciales o factoriales, el algoritmo BL ofrece una solución en un tiempo considerablemente menor.

Es importante tener en cuenta que el enfoque de este algoritmo es heurístico, lo que significa que su principal objetivo es proporcionar una respuesta en un tiempo aceptable, como lo está demostrando. Sin embargo, es esencial recalcar que, debido a su naturaleza heurística, el algoritmo BL podría no garantizar la respuesta correcta en todo momento. Existe la posibilidad de que se estanque en un máximo local, que no necesariamente coincide con el máximo global del problema.

Tamaño del grafo en arcos usando solo las pruebas de 20 vértices	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor Tiempo de ejecución
30/20	1,5	1	1,07442597	1,0441937	0
40/20	2	1,001157407	1,045922407	1,027738599	0
50/20	2,5	1	1,471100554	1,279736718	0
80/20	4	1	1,742676168	1,440996709	0
80/50	1,6	1	1,184607104	1,126010287	0

Tabla 6: Factor de crecimiento del algoritmo Búsqueda Local usando solo los grafos de 20 vértices

En la tabla presentada, se puede observar que el factor de crecimiento del algoritmo de búsqueda local en términos de asignaciones, comparaciones, líneas ejecutadas y tiempo de ejecución muestra variaciones mínimas, limitándose a cambios de apenas unos cuantos decimales. Esta consistencia sugiere que, al mantener constante la cantidad de vértices, el comportamiento del algoritmo de búsqueda local puede considerarse efectivamente constante. Esta observación es crucial, ya que indica una estabilidad en el rendimiento del algoritmo bajo condiciones específicas, lo cual es un aspecto positivo para su aplicación en escenarios donde la escala del problema no varía significativamente.

A continuación, se mostrarán las notaciones para cada comportamiento de los algoritmos.

Notación Bron - Kerbosch

Clasificación en notación O Grande según sus comparaciones, asignaciones, líneas ejecutadas y tiempo de ejecución. Contemplando el tamaño de los vértices del grafo:

Usar la notación O	
--------------------	--

Clasificación del comportamiento de las asignaciones	$O(n!)$
Clasificación del comportamiento de las comparaciones	$O(n!)$
Clasificación del comportamiento de las líneas ejecutadas	$O(n!)$
Clasificación del comportamiento en el tiempo de ejecución	$O(n!)$

Clasificación en notación O Grande según sus comparaciones, asignaciones, líneas ejecutadas y tiempo de ejecución. Contemplando el tamaño fijo de vértices del grafo cambiando la cantidad de arcos:

Usar la notación O	
Clasificación del comportamiento de las asignaciones	$O(n!)$
Clasificación del comportamiento de las comparaciones	$O(n!)$
Clasificación del comportamiento de las líneas ejecutadas	$O(n!)$
Clasificación del comportamiento en el tiempo de ejecución	$O(n!)$

Notación Búsqueda Local

Clasificación en notación O Grande según sus comparaciones, asignaciones, líneas ejecutadas y tiempo de ejecución. Contemplando el tamaño de los vértices del grafo:

Usar la notación O	$O(n^2)$
Clasificación del comportamiento de las asignaciones	$O(n^2)$
Clasificación del comportamiento de las comparaciones	$O(n^2)$
Clasificación del comportamiento de las líneas ejecutadas	$O(n^2)$
Clasificación del comportamiento en el tiempo de ejecución	$O(1)$

Clasificación en notación O Grande según sus comparaciones, asignaciones, líneas ejecutadas y tiempo de ejecución. Contemplando el tamaño fijo de vértices del grafo cambiando la cantidad de arcos:

Usar la notación O	$O(n)$
Clasificación del comportamiento de las asignaciones	$O(n)$
Clasificación del comportamiento de las comparaciones	$O(n)$

Clasificación del comportamiento de las líneas ejecutadas	$O(n)$
Clasificación del comportamiento en el tiempo de ejecución	$O(n)$

A continuación, se mostrarán las gráficas correspondientes al proyecto.

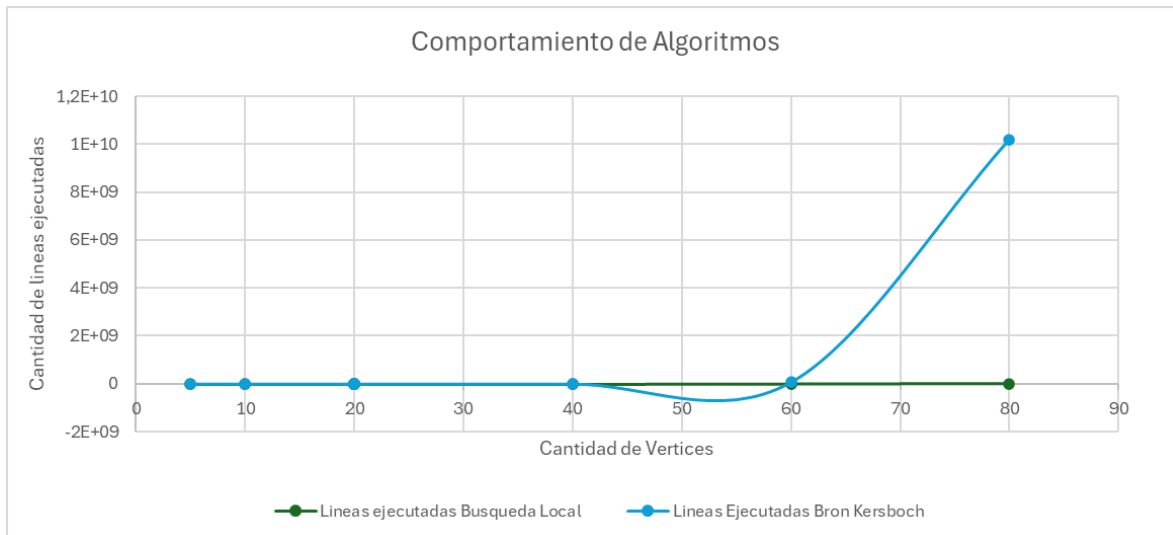


Tabla 7: Comportamiento de los Algoritmos

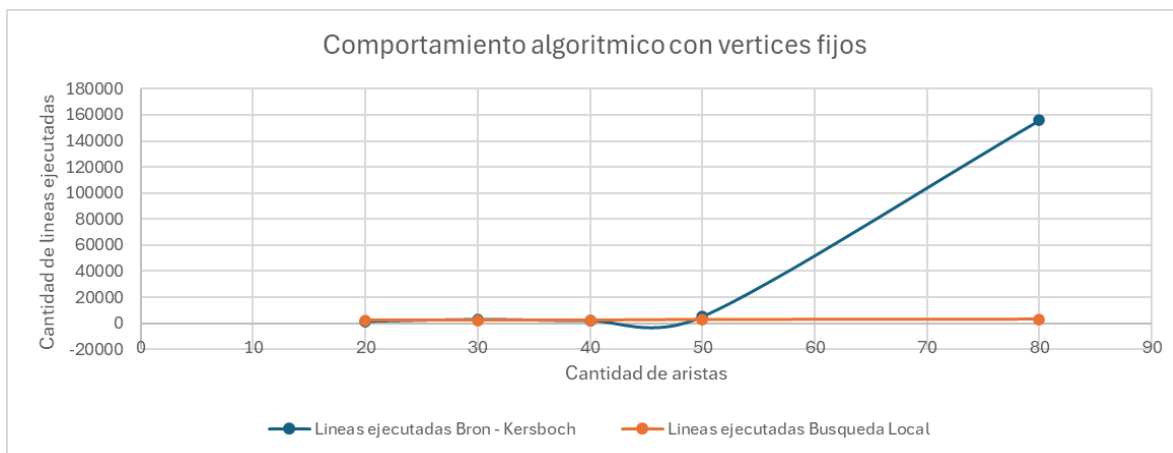


Tabla 8: Comportamiento de los Algoritmos con 20 vértices y aristas variantes.

Al revisar los datos obtenidos se observaron los siguientes aspectos entre ambos algoritmos:

El algoritmo de Bron-Kerbosch, en contraste con la Búsqueda Local, tiene una complejidad temporal factorial en el peor de los casos. Esto significa que para grafos densos o con muchos cliques, el rendimiento puede deteriorarse significativamente. Bron-Kerbosch realiza una exploración exhaustiva de las posibles combinaciones de vértices

para encontrar cliques máximos siendo en el peor de los casos un comportamiento de $O(3^{N/3})$, esto según la literatura, lo que puede resultar en un alto costo computacional incluso para grafos relativamente pequeños. Aunque Bron-Kerbosch puede encontrar el clique máximo de un grafo, puede volverse ineficiente para grafos grandes debido a su naturaleza exhaustiva y su enfoque poco selectivo.

Por otro lado, la Búsqueda Local tiene una complejidad temporal que depende del número de iteraciones y la cantidad de operaciones realizadas en cada iteración. En comparación con Bron-Kerbosch, la complejidad puede ser menor en muchos casos. La Búsqueda Local se enfoca en buscar soluciones locales óptimas, explorando solo un subconjunto de todas las posibles combinaciones de vértices. Este enfoque más selectivo y la evitación de explorar todas las posibles combinaciones de vértices hacen que la Búsqueda Local sea más adecuada para grafos grandes, donde puede ser más eficiente en términos de tiempo de ejecución y recursos computacionales.

Esto se puede apreciar mejor en [5] donde se muestra que el comportamiento de la búsqueda local al aplicar las pruebas de DINAMICS, logra obtener los resultados esperados, logrando ser más eficiente que el Bron el cual según [8], hace mención del comportamiento inevitable que llega a poseer el Bron en entradas grandes, siendo su comportamiento exponencial.

El aumentar la cantidad de arcos sin aumentar la cantidad de vértices no provoca un cambio muy grande, generando solo que el cambio de donde se produce la exponencial del bron pase de ser en la cantidad de 60 aristas a la cantidad de 50, para el caso de la búsqueda local esta sigue un crecimiento muy semejante, siendo un poco más elevado a comparación que para la prueba sin limitar la cantidad de vértices.

Medición analítica

Bron - Kersboch

[illegible]

	$3n^3$ $=$ $3(n^3+n^2+n)+4$
Clasificación en notación O Grande	$O(N^2)$

Conclusiones

En conclusión, el análisis comparativo entre el algoritmo de Bron-Kerbosch y la Búsqueda Local para el problema del máximo clique revela diferencias significativas en términos de complejidad temporal y eficiencia computacional.

Por un lado, el algoritmo de Bron-Kerbosch presenta una complejidad temporal exponencial en el peor de los casos, lo que lo hace menos eficiente para grafos densos o con muchos cliques. Su enfoque exhaustivo de explorar las posibles combinaciones de vértices puede resultar en un alto costo computacional, especialmente para grafos relativamente grandes. Aunque puede encontrar el clique máximo, su rendimiento puede deteriorarse para grafos grandes debido a su naturaleza poco selectiva.

Por otro lado, la Búsqueda Local ofrece una alternativa con una complejidad temporal que depende del número de iteraciones y la cantidad de operaciones realizadas en cada iteración. Su enfoque más selectivo de explorar solamente un subconjunto de combinaciones de vértices la hace más adecuada para grafos grandes. Esto se traduce en un mejor rendimiento en términos de tiempo de ejecución y recursos computacionales en comparación con Bron-Kerbosch, especialmente para grafos densos.

En nuestra evaluación gráfica y empírica, la Búsqueda Local demostró ser más eficiente en la mayoría de los casos, especialmente para grafos grandes con densidad variable. Sin embargo, es importante tener en cuenta que la eficiencia de cada algoritmo puede variar según la entrada de los datos, incluyendo la cantidad de vértices y arcos en el grafo.

Bibliografía

- [1] P. R. Östergård, "A fast algorithm for the maximum clique problem," *Discrete Applied Mathematics*, vol. 120, no. 1, pp. 197–207, 2002. Special Issue devoted to the 6th Twenty Workshop on Graphs and Combinatorial Optimization.
- [2] J. C. Ponce, E. E. P. de León, A. Padilla, F. Padilla, and A. O. O. Zezzatti, "Algoritmo de colonia de hormigas para el problema del clique máximo con un optimizador local k-opt," *Hífen, Uruguiana*, vol. 30, no. 58, pp. 191–196, 2006.
- [3] G. V. GÓMEZ, *El problema del clique máximo: Análisis, resolución e implementación*. PhD thesis, Universidad Nacional Autónoma de México, 2019.
- [4] S. Cagnoni, "Applications of Evolutionary Computing," Springer, pp. 112-120, 2002.
- [5] Wayne Pullan, "Dynamic Local Search for the Maximum Clique Problem," 2006, pp. 159-185.
- [6] J. Ponce, E. Ponce de Leon Senti, A. Padilla, F. Padilla, y A. Ochoa-Zezzatti, "Algoritmo de Colonia de Hormigas para el Problema del Clique Máximo con un Optimizador Local K-opt," págs. 190-196, 2008.
- [7] D. Smith, R. Montemanni, y S. Perkins, "The Use of an Exact Algorithm within a Tabu Search Maximum Clique Algorithm," *Algorithms*, vol. 13, pág. 253, octubre 2020. doi: 10.3390/a13100253.
- [8] G. Villeda Gómez, "El problema del clique máximo: Análisis, resolución e implementación," *Tesis de Licenciatura, Universidad Nacional Autónoma de México, Facultad de Ciencias, Ciudad Universitaria, Ciudad de México*, 2019, pp. 35-36.

Minutas

Minuta de Reunión #1

Lugar: Llamada por discord	Fecha y hora: 26/03/2024, 1:00 pm
Objetivo de la reunión: Iniciar con el proyecto de análisis de algoritmos.	

Participantes	Cargo	Firma
Roosevelt Alejandro Pérez González		Presente
José Andrés Lorenzo Segura		Presente

Ausentes	Cargo	Justificación
No Aplica		

Temas por tratar

Tema 1

Investigación de los posibles algoritmos a trabajar.

Acuerdos

- Se acordó investigar sobre los siguientes algoritmos:
 1. Algoritmo: Búsqueda local
 2. Algoritmos Genéticos
 3. Algoritmo: Bron - Kerbosch
 4. Algoritmo: Búsqueda tabú
 5. Algoritmo: Optimización por colonias de hormigas
 6. Algoritmo: Ramificación y Acota

Tema 2

Como realizar la documentación del proyecto

Acuerdos

- Usar la herramienta Overleaf para la documentación del proyecto, debido a que esta ofrece facilidades al momento de realizar citas.

Minuta de Reunión #2

Lugar: Cubículos de la biblioteca	Fecha y hora: 04/04/2024, 5:00 pm
Objetivo de la reunión: Iniciar con la implementación de los algoritmos.	

Participantes	Cargo	Firma
Roosevelt Alejandro Pérez González		Presente
José Andrés Lorenzo Segura		Presente
Arnold Jafeth Alvares Rojas		Presente

Ausentes	Cargo	Justificación
No Aplica		

Temas por tratar

Tema 1

Elegir de los algoritmos elegidos los que vamos a implementar

Acuerdos <ul style="list-style-type: none"> Se acordó aplicar implementar los siguientes algoritmos: <ol style="list-style-type: none"> Bron-Kerbosch Búsqueda Local
Tema 2 Se distribuyeron tareas. Acuerdos <ul style="list-style-type: none"> Roosevelt y Lorenzo trabajaran en el apartado de la algoritmia y Arnold se encargará del apartado de buscar más información sobre los algoritmos.

Minuta de Reunión #3

Lugar: Cubículos de la biblioteca	Fecha y hora: 07/04/2024, 5:00 pm
Objetivo de la reunión: Corregir los errores de los algoritmos.	

Participantes	Cargo	Firma
Roosevelt Alejandro Pérez González		Presente
José Andrés Lorenzo Segura		Presente
Arnold Jafeth Alvares Rojas		Presente

Ausentes	Cargo	Justificación
No Aplica		

Temas por tratar

<p>Tema 1</p> <p>Corregir los errores de los algoritmos.</p> <p>Acuerdos</p> <ul style="list-style-type: none"> Se acordó mejorar la eficiencia de los algoritmos, debido a que presentaban una serie de errores.
<p>Tema 2</p> <p>Agregar los primeros datos de prueba.</p> <p>Acuerdos</p> <ul style="list-style-type: none"> Se acordó probar los algoritmos con los primeros casos de prueba para verificar la fiabilidad de los algoritmos.
<p>Tema 3</p> <p>Agregar las pruebas finales del proyecto.</p> <p>Acuerdos</p> <ul style="list-style-type: none"> Debido a la prueba exitosa de los primeros casos de prueba, se decidió agregar los datos finales del proyecto.

Minuta de Reunión #4

Lugar: Llamada por discord	Fecha y hora: 12/04/2024, 5:00 pm
Objetivo de la reunión: Iniciar con la documentación del proyecto.	

Participantes	Cargo	Firma
Roosevelt Alejandro Pérez González		Presente
José Andrés Lorenzo Segura		Presente
Arnold Jafeth Alvares Rojas		Presente

Ausentes	Cargo	Justificación
No Aplica		

Temas por tratar

<p>Tema 1</p> <p>Empezar a realizar las tablas, graficas, y documento necesario para el proyecto.</p> <p><u>Acuerdos</u></p> <ul style="list-style-type: none"> Se acordó las tareas para cada estudiante. Roosevelt y Arnold trabajaran en el documento, y Roosevelt y Lorenzo trabajaran en el Excel que permitirá las tablas y graficas requeridas.
<p>Tema 2</p> <p>Realizar los diagramas los grafos.</p> <p>Acuerdos</p> <ul style="list-style-type: none"> Roosevelt va realizar los diagramas de los grafos con Python con ayuda de librerías.

Bitácora de Actividades

Nombre del Estudiante: Roosevelt Alejandro Pérez González

Fecha: 26/03/24 – 03/04/24

Actividad: Búsqueda de información de algoritmos

Duración: 10 horas

Descripción de la actividad:

Luego de la reunión con José Lorenzo, empecé a investigar sobre los posibles algoritmos a implementar, como el de Bron – Kersboch, colonia de hormigas, tomita entre otros.

Nombre del Estudiante: Roosevelt Alejandro Pérez González

Fecha: 04/04/24 – 07/04/24

Actividad: Iniciar con la implementación de los algoritmos

Duración: 10 horas

Descripción de la actividad:

Luego de haber elegido los algoritmos a implementar, se empezó en la implementación de los algoritmos, entender como trabajan los algoritmos, que técnicas utilizan, en este caso, backtracking y voraz.

A medida que se va desarrollando los algoritmos se van haciendo mejoras al mismo.

Nombre del Estudiante: Roosevelt Alejandro Pérez González

Fecha: 07/04/24 – 12/04/24

Actividad: Corrección de más errores

Duración: 10 horas

Descripción de la actividad:

La implementación de los algoritmos sigue dando errores, por ende, la corrección de la misma es crucial.

Una vez las correcciones se han hecho, se han introducido los primeros datos de prueba para seguidamente insertar los datos finales del proyecto.

Nombre del Estudiante: Roosevelt Alejandro Pérez González

Fecha: 13/04/24 – 16/04/24

Actividad: Realización de documento

Duración: 10 horas

Descripción de la actividad:

Durante la última etapa del proyecto, se desarrollo los documentos requeridos para la documentación externa.

Entre lo trabajado en estas horas se encuentra análisis del problema, realización de los diagramas de los grafos, tablas en excel, entre otras cosas.

Nombre del Estudiante: Arnold Jafeth Álvarez Rojas

Fecha: 04/04/2024

Actividad: Realizar la implementación de los algoritmos seleccionados

Duración: 6 horas

Descripción de la actividad:

Tras buscar varios algoritmos, se llegó a la conclusión de utilizar el algoritmo de Bron y el de búsqueda local, con los cuales se iba a realizar el proyecto.

Con lo que se buscó información sobre estos 2 códigos seleccionados, a través de libros y demás artículos para hacer uso de referencia al hablar de sus cualidades.

Nombre del Estudiante: Arnold Jafeth Álvarez Rojas

Fecha: 04/04/2024

Actividad: Realizar la implementación de los algoritmos seleccionados

Duración: 6 horas

Descripción de la actividad:

Tras buscar varios algoritmos, se llegó a la conclusión de utilizar el algoritmo de Bron y el de búsqueda local, con los cuales se iba a realizar el proyecto.

Con lo que se buscó información sobre estos 2 códigos seleccionados, a través de libros y demás artículos para hacer uso de referencia al hablar de sus cualidades.

Nombre del Estudiante: Arnold Jafeth Álvarez Rojas

Fecha: 07/04/2024

Actividad: Corrección de errores en los algoritmos

Duración: 4 horas

Descripción de la actividad:

Tras la primera implementación de los algoritmos, se observaron procesos que podían modificarse para que el código no diera problemas al devolver los resultados y procesos más eficientes aplicados de otras formas, además, se incorporaron casos iniciales para verificar que el proceso funcionara correctamente y que los mismos devolvieran la información de lo que se esperaba.

Tras comprobarse la fiabilidad de los códigos, se implementaron los casos solicitados en el proyecto para ambos códigos.

Nombre del Estudiante: Arnold Jafeth Álvarez Rojas

Fecha: 12/04/2024

Actividad: Documentación del documento

Duración: 8 horas

Descripción de la actividad:

Tras una charla entre los miembros del proyecto, se acordaron roles para trabajar con lo faltante del proyecto.

Se realizó una búsqueda de documentos que hablaran de la gran mayoría de códigos seleccionados, tanto los usados para el proyecto, como los no utilizados. Esto con el fin de realizar la parte de análisis del problema, solución del problema, análisis de resultados y las conclusiones.

Nombre del Estudiante: José Andrés Lorenzo Segura

Fecha: 26/03/24

Duración: 3 horas

Descripción de la actividad: Se empezó con la búsqueda de los algoritmos necesarios para el proyecto

Nombre del Estudiante: José Andrés Lorenzo Segura

Fecha: 04/04/24 Duración: 7 horas

Descripción de la actividad: Se redujeron los algoritmos que se iban a estudiar y se comenzó con la documentación

Nombre del Estudiante: José Andrés Lorenzo Segura

Fecha: 07/04/24

Duración: 4 horas

Descripción de la actividad: Se buscaron los códigos de los algoritmos y se ajustaron

Nombre del Estudiante: José Andrés Lorenzo Segura

Fecha: 12/04/24

Duración: 5 horas

Descripción de la actividad: Continuamos con la documentación del proyecto y la realización de las tablas del Excel

Coevaluación

CRITERIO	1 Deficiente 0 pts.	2 Requiere mejorar 1 pt.	3 Bueno 2 pts.	4 Excelente 3pts.	Observaciones o comentarios
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.	
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo, incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupación en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.	
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Assumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Assumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Assumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.	
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, muestran una limitada escucha con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.	
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumple con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.	
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecidas, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.	

Nombres y firma:	Nota ponderada por el subgrupo
Roosevelt Alejandro Pérez González	100%
José Andrés Lorenzo Segura	100%
Arnold Jafeth Alvares Rojas	100%

Autoevaluación

CRITERIO	1 Deficiente 0 pts.	2 Requiere mejorar 1 pt.	3 Bueno 2 pts.	4 Excelente 3pts.
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo, incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupación en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Assumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Assumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Assumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, escuchan con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumplen con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecidas, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.

Nombre del estudiante	Nota Obtenida
Roosevelt Alejandro Pérez González	100%
José Andrés Lorenzo Segura	100%
Arnold Jafeth Alvares Rojas	100%