

ISOMORFISMO DE SUBGRAFOS - PROYECTO ANÁLISIS DE ALGORITMOS

ENTREGA 1

 **Santiago Martínez Loaiza**
s.martinezl234@uniandes.edu.co
202510729

 **Estefania Laverde**
e.laverdeb@uniandes.edu.co
201922512

 **Pablo Ortega**
p.ortegac@uniandes.edu.co
202021700

March 14, 2025

1 Modelaje del problema

En este proyecto se explorará dentro del contexto de teoría de grafos el problema de *isomorfismo de grafos*, entendiendo el problema e investigando la teoría e implementación de los algoritmos

- Ullmann: algoritmo de backtracking propuesto en 1976 [8] [5].
- VF2: igualmente un algoritmo de backtracking con búsqueda en profundidad eficiente y reducción del espacio de búsqueda. Publicado en el año 2004 [3][1].
- VF2++: mejora propuesta a VF2 basada en ordenamiento de nodos y nuevas reglas que reducen el espacio de búsqueda publicada en 2018 [3].

Para comprender el problema se tiene la siguiente definición.

Definición: Sean G y P dos grafos no dirigidos. Se dice que $G' \subset G$ es un subgrafo isomorfo a P si existe una biyección entre sus conjuntos de vértices que preserven la adyacencia de sus ejes [4].

Un ejemplo gráfico se encuentra en la figura 1, donde se tienen P y G dos grafos no dirigidos y se encuentra una correspondencia entre P y un subconjunto de vértices de G .

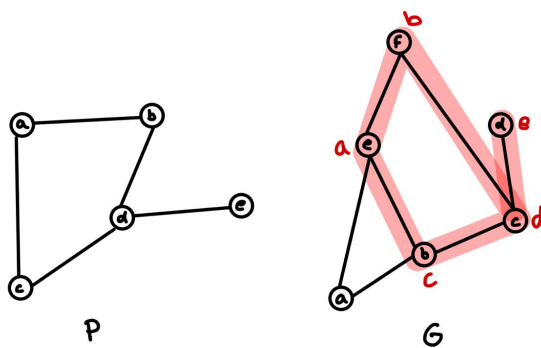


Figure 1: Ejemplo isomorfismo de subgrafos

El problema de isomorfismo de subgrafos es un problema *NP-COMplete*, como se muestra a continuación, y tiene aplicación en diversas áreas de conocimiento que van más allá de la computación. Es por esto de gran utilidad explorar diversos algoritmos de solución y encontrar de ser posible mejoras a estos mismos.

1.1 Problema NP-COMPLETE

Como se mencionó anteriormente, demostremos que el problema de isomorfismo de subgrafos es un problema NP-COMPLETE. Para ello veamos que teniendo acceso a un algoritmo que soluciona al problema de isomorfismo de subgrafos podemos dar solución al problema del ciclo hamiltoniano, el cual es un problema NP-COMPLETE bien conocido. También veremos que existe un verificador polinomial para el problema de isomorfismo de subgrafos.

Sea A un algoritmo para el problema de isomorfismo de subgrafos. Dados grafos G y S , $A(G, S)$ es true si y solo si G tiene un subgrafo isomorfo a S . En cuanto al problema del ciclo hamiltoniano, sea G un grafo para el cual queremos saber si tiene o no un ciclo hamiltoniano, suponga G tiene n nodos y sea C_n el grafo ciclo de n nodos, así G tiene ciclo hamiltoniano si y solo si $A(G, C_n)$ es true, con esto obtenemos la reducción requerida.

La función de isomorfismo se puede tomar como certificado y puede verificarse en tiempo polinomial, con esto concluimos que el problema de isomorfismo de subgrafo es NP-COMPLETE, este es un ejercicio clásico en teoría de la computación[7].

2 Contexto real en el que se requiera solucionar el problema

El uso de grafos para modelamiento de problemas se extiende a gran cantidad de disciplinas. En particular, el problema de isomorfismo de subgrafos se encuentra muy presente en aplicaciones de **bioinformática** y **química** [3], donde se representan estructuras moleculares como grafos, entendiendo sus nodos como átomos y ejes como enlaces químicos. En este contexto, una de las técnicas más utilizadas consiste en encontrar un subgrafo común máximo, MCS [6] entre dos estructuras moleculares, lo cual permite identificar patrones estructurales que establezcan alguna relación entre compuestos con características parecidas. Por otro lado, en el campo de **visión por computadora**, la capacidad de modelar objetos, como símbolos o rostros, en forma de grafos que capturan características clave, dan lugar al uso de algoritmos de isomorfismo de subgrafos para reconocer estos mismos a partir de rasgos comunes en diferentes imágenes[3]. Este tipo de aplicaciones han ido mejorando con el paso del tiempo a medida que se investigan y desarrollan algoritmos de menor complejidad computacional.

3 Descripción del algoritmo de solución de los artículos encontrados

Dividiremos la sección con el enfoque en los tres algoritmos seleccionados: Ullmann, VF2 y VF2++.

3.1 Algoritmo de Ullmann

- Inputs: Dos grafos etiquetados G y P .
- Outputs: M matriz de correspondencia de vértices.

El algoritmo de Ullmann hace uso de una técnica de backtracking para resolver el problema de isomorfismo de subgrafos. En este, se define una representación de isomorfismos mediante una matriz de correspondencias M de tamaño $|V_P| \times |V_G|$, entre los vértices de P (filas) y los vértices de G (columnas), y cuyas entradas son

$$M[i, j] = \begin{cases} 1, & \text{si el vértice } v_i \in P \text{ puede ser mapeado a } v_j \in G \\ 0, & \text{d.l.c.} \end{cases}$$

La matriz se inicializa según un criterio de grados del vértice, teniendo en cuenta que para que exista un mapeo de un vértice en $v_i \in P$ a un vértice en $v_j \in G$

$$\text{grado}(v_i) \leq \text{grado}(v_j).$$

Después de inicializar M , el algoritmo de manera recursiva prueba con todas las matrices M que pueden ser obtenidas de la inicial, eliminando todos excepto un 1 de cada fila (para que cada vértice de P quede con un correspondiente vértice en G), y teniendo como máximo 1 en cada columna. Como se puede intuir, el algoritmo implementado de esta manera no es eficiente, pues recorre muchas posibles matrices de correspondencias para llegar a la solución. Por esta razón se incorpora una fase de *pruning*, donde se eliminan de la matriz M los 1's no válidos teniendo en cuenta que si a un vértice en $v_i \in P$ se le asigna un vértice en $v_j \in G$, entonces los vecinos de v_i deben tener correspondencia con los vecinos de v_j . Si satisface las condiciones se obtiene el mapeo, y de lo contrario se retrocede y se busca otra opción [8]. [5].

3.2 VF2

- Inputs: Dos grafos etiquetados G y P . Cada vértice de ambos grafos debe tener una etiqueta.
- Outputs: M Una correspondencia (*mapping*) de los vértices de P en los vértices de G que satisfaga un isomorfismo de subgrafo.

El algoritmo VF2 se basa en explorar asignaciones parciales entre los vértices de P y G de forma recursiva, utilizando para ello una **búsqueda en profundidad con backtracking**. En la inicialización, se analiza cada vértice de P a fin de determinar candidatos en G que tengan etiquetas y grados compatibles. Una vez identificado un par posible, se añade temporalmente al mapeo parcial y se comprueba la consistencia de la asignación, asegurando que las adyacencias respeten la relación de subgrafo. Si la extensión resultante se mantiene coherente, el proceso prosigue hacia el siguiente vértice de P ; de lo contrario, se deshace el último emparejamiento y se busca otra alternativa. Si todos los vértices de P se logran asignar satisfactoriamente, se confirma la existencia de un isomorfismo de subgrafo entre P y G . En el algoritmo se hace un mejor uso de memoria a comparación con el algoritmo de Ullman pues no trabaja con grandes matrices de correspondencia y hace uso de mejores estructuras de datos para que hacen al algoritmo más eficiente [3].

3.3 VF2++

- Inputs: Dos grafos etiquetados G y P . Cada vértice de ambos grafos debe tener una etiqueta.
- Outputs: M Una correspondencia (*mapping*) de los vértices de P en los vértices de G que satisfaga un isomorfismo de subgrafo.

El algoritmo VF2++ se basa en el algoritmo VF2 pero propone dos mejoras que incrementan su eficiencia. La primera se centra en determinar un orden de selección de emparejamiento para los nodos de P , de modo que se podan rápidamente aquellas extensiones que resultan infactibles. En segundo lugar, VF2++ introduce reglas de poda más eficaces y fáciles de verificar, de manera que el espacio de búsqueda se reduce con mayor rapidez. Como los métodos previos, VF2++ construye su solución mediante un esquema de backtracking que explora las asignaciones posibles mediante un DFS. Para iniciar, se calcula el orden en que se mapearán los vértices de P atendiendo a criterios como rareza de etiquetas o cantidad de vecinos, con el fin de ubicar primero a los nodos “difíciles”, y así filtrar tempranamente las extensiones inviables. Una vez fijado el orden, se recorre cada vértice de P y se busca un posible vértice candidato en G , comprobando la etiqueta y las relaciones de vecindad para descartar inconsistencias. En caso de superar las verificaciones, se agrega el par al mapeo parcial y se continúa de manera recursiva; si no, se hace backtracking y se busca otra opción. Cuando todos los vértices de P han sido asignados correctamente, se obtiene un mapeo que satisface el isomorfismo de subgrafo [3].

En la investigación realizada se encuentran gráficas de comparación para los algoritmos mencionados [2], mostrando como han ido mejorando en tiempo de ejecución las soluciones del problema descrito a medida que se reducen los espacios de búsqueda y se utilizan estructuras de representación más adecuadas.

4 Enlace a una herramienta de software libre que implemente al menos uno de los algoritmos encontrados

El algoritmo VF2++ se encuentra implementado en la librería Networkx, específicamente en la función `vf2pp_isomorphism`.

References

- [1] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, 2004.
- [2] P. Foggia, C. Sansone, and M. Vento. A performance comparison of five algorithms for graph isomorphism. *Proceedings of the 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 188–199, 2001.
- [3] Alpár Jüttner and Péter Madarasi. Vf2++—an improved subgraph isomorphism algorithm. *Discrete Applied Mathematics*, 242:69–81, 2018. Computational Advances in Combinatorial Optimization.
- [4] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.
- [5] Adrian Neumann. Ullman’s subgraph isomorphism algorithm, 2013. Accessed: 2025-03-12.

- [6] John W. Raymond and Peter Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(1):68–79, 2011.
- [7] Ronald L. Rivest Clifford Stein Thomas H. Cormen, Charles E. Leiserson. *Introduction to algorithms*. The MIT Press, 2001.
- [8] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.