# Isomorfismo de subgrafos

Universidad de la Habana
Facultad de matematica y computacion
Alejandro Alvarez Lamazares
Frank Pérez Fleita
Angel Daniel Alonso Guevara

20 de febrero de 2025

**Resumen**

Este trabajo investiga el problema del isomorfismo de subgrafos en el contexto de la planificación urbana. Se centra en el desarrollo de métodos para identificar y replicar patrones estructurales entre infraestructuras urbanas de diferentes escalas, con el objetivo de optimizar los procesos de planificación y diseño.

## Palabras clave

planificación urbana, isomorfismo de grafos, subgrafos, optimización de infraestructura.

## 1. Introducción

En el contexto de la planificación urbana contemporánea, surge un desafío metodológico fundamental que requiere atención académica rigurosa. La observación sistemática revela que durante el proceso de diseño de infraestructuras urbanas de menor magnitud, frecuentemente se identifican patrones estructurales que presentan analogías significativas con diseños previos de mayor escala. Esta circunstancia plantea una oportunidad metodológica sustancial para optimizar los procesos de planificación urbana.

La presente investigación aborda el problema de la replicación de infraestructura urbana, entendido como la búsqueda sistemática de correspondencias estructurales entre diseños de infraestructura urbana existentes y nuevos proyectos de mayor envergadura. Este problema se formaliza matemáticamente mediante la búsqueda de isomorfismos entre subgrafos, donde un grafo $G_A$ representa la infraestructura crítica de mayor escala y un grafo $G_B$ representa la nueva infraestructura propuesta.

La relevancia académica de este estudio radica en su capacidad para contribuir significativamente a la optimización de procesos de planificación urbana. La identificación sistemática de patrones estructurales equivalentes permite la transferencia controlada de soluciones probadas, minimizando la redundancia en el proceso de diseño y optimizando la asignación de recursos. Este enfoque metodológico se fundamenta en el principio de que la experiencia acumulada en proyectos de mayor escala puede ser sistemáticamente aprovechada para mejorar la eficiencia y efectividad en el desarrollo de infraestructuras urbanas menos complejas.

# 2. Marco Teórico

## 2.1. Definiciones Preliminares

### 2.1.1. Infraestructura Urbana como Grafo

Una infraestructura urbana se modela como un grafo no dirigido $G = (V, E)$, donde:

- $V$ es el conjunto de nodos que representan elementos clave de la infraestructura (e.g., hospitales, escuelas, estaciones de transporte)
- $E \subseteq V \times V$ es el conjunto de aristas que representan relaciones espaciales entre los nodos. Cada arista $e = (u, v) \in E$ está asociada a una distancia o tiempo máximo $d(e)$, que puede ser fijo o variable. Para simplificar, asumimos que $d(e)$ es constante y representa una distancia fija entre los nodos conectados

### 2.1.2. Diseño Previo

Un diseño previo es un grafo $G' = (V', E')$, donde $|V'| \geq |V|$. Este diseño ha sido validado como funcional y cumple con todas las pruebas requeridas para ser considerado una infraestructura bien diseñada.

### 2.1.3. Isomorfismo de Grafos

Dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$ son isomorfos si existe una función biyectiva $f : V_1 \to V_2$ tal que: Para todo par de nodos $u, v \in V_1$, se tiene que $(u, v) \in E_1 \iff (f(u), f(v)) \in E_2$. En este contexto, también consideramos que las aristas deben preservar sus atributos asociados (e.g., distancias)

### 2.1.4. Isomorfismo de Subgrafos

Un grafo $G = (V, E)$ es subisomorfo a un grafo $G' = (V', E')$ si existe un subgrafo de $G'$, denotado como $G'' = (V'', E'')$, tal que $G'' \cong G$.

## 2.2. Definicion del Problema

Dado un nuevo diseño propuesto representado por el grafo $G = (V, E)$ y un único diseño previo validado representado por el grafo $G' = (V', E')$, queremos determinar si el nuevo diseño es funcional basándonos en la estructura del diseño previo. Esto se traduce en verificar si el grafo propuesto es subisomorfo al grafo validado. En términos formales, queremos verificar si:

$$\exists f : V \to V', \quad f \text{ es una función biyectiva entre } V \text{ y un subconjunto } V'' \subseteq V',$$

tal que:

$$(u, v) \in E \iff (f(u), f(v)) \in E'.$$

Si tal relación de subisomorfismo existe, podemos concluir que el nuevo diseño propuesto comparte las propiedades estructurales y espaciales del diseño previo validado y, por lo tanto, es funcional.

### 2.2.1. Formalización del Problema

Sea:

- Un diseño previo validado representado por el grafo $G' = (V', E')$
- Un nuevo diseño propuesto representado por el grafo $G = (V, E)$

El problema consiste en determinar si existe un subgrafo de $G'$, denotado como $G'' = (V'', E'')$, tal que:

$$G \cong G''$$

Esto implica encontrar una función biyectiva entre los nodos de $G$ y un subconjunto de nodos de $G'$, preservando las conexiones entre los nodos.

## 2.3. Demostración de NP-completitud

### 2.3.1. Verificación en NP

Para demostrar que el problema está en NP, se puede proporcionar un certificado polinómico. Un posible certificado es la biyección $f$ mencionada anteriormente. Dado $G$ y $H$, un verificador puede comprobar en tiempo polinómico si esta biyección preserva la adyacencia, lo que confirma que $H$ es isomorfo a un subgrafo de $G$.

La idea de la verificación pasa por dos pasos:

1. Primero, verificar si es una biyección correcta, en cuyo caso resolvemos con pasar por todos los vértices de $V$ y verificar que existe y es único el $v \in V'$ tal que $f(u) = v$. Lo cual se puede hacer en $O(n)$, siendo $n$ la cantidad de vértices.

2. En segundo lugar, tenemos que verificar que efectivamente sea un isomorfismo. La verificación se hace de la siguiente manera: por cada arista $\langle u, v \rangle$ del grafo $V$, vamos a verificar si la arista $\langle f(u), f(v) \rangle$ pertenece al grafo $V'$.

Un código que implementa esta verificación puede ser el siguiente:

```
def verifica_isomorfismo(G, H, f):
    # G y H son grafos representados como listas de adyacencia
    for u in G:
        for v in G[u]:  # Para cada vecino v de u en G
            if (f[u], f[v]) not in H:  # Comprobamos si la arista está en H
                return False  # No preserva la adyacencia
    return True  # Preserva la adyacencia
```

La complejidad del algoritmo está dada por la cantidad de aristas ($m$), siendo $O(m)$ o $O(n^2)$ para el caso peor si se mira desde la perspectiva de la cantidad de nodos.

### 2.3.2. Reducción

**Reducción al problema del Clique**

Para probar que el problema es NP-duro, realizamos una reducción desde el problema del clique, que es conocido por ser NP-completo. El problema del clique se define como sigue: dado un grafo $G$ y un entero $k$, ¿existe un subgrafo completo (clique) de tamaño $k$? La reducción se realiza de la siguiente manera:

Dado un par $(G, k)$ del problema del clique, construimos dos grafos:

$$G_1 = K_k, \text{ el grafo completo con } k \text{ vértices.}$$
$$G_2 = G.$$

Queremos demostrar que existe una clique de tamaño $k$ en $G$ si y solo si existe un subgrafo isomorfo a $K_k$ en $G_2$.

Si existe una clique de tamaño $k$ en $G$, entonces los vértices de esta clique forman un subgrafo completo, que es isomorfo a $K_k$. Si existe un subgrafo isomorfo a $K_k$ en $G_2 = G$, esto implica que hay un conjunto de $k$ vértices en $G$ donde cada par está conectado por una arista, lo cual significa que estos vértices forman una clique.

Dado que hemos mostrado que la existencia de una clique en $G$ se traduce directamente a la existencia de un subgrafo isomorfo a $K_k$ en el grafo original, concluimos que:

$$\text{CLIQUE} \leq_p \text{SUBGRAPH ISOMORPHISM}$$

#### 2.3.3. Luego

Dado que hemos establecido que el problema de isomorfismo de subgrafos está en NP y hemos realizado una reducción polinómica desde el problema del clique, podemos concluir formalmente que el problema de isomorfismo de subgrafos es NP-completo. Esta demostración se basa en los resultados conocidos sobre la complejidad computacional y las propiedades fundamentales del problema.

# 3. Algoritmo de Ullman para Isomorfismo de Subgrafos

El algoritmo de Ullman, propuesto en 1976, es un método de búsqueda en árbol para encontrar todos los isomorfismos de subgrafos entre dos grafos dados[1]. A continuación, se presenta una formalización del algoritmo.

## 3.1. Definición del Problema

Sean $G_a = (V_a, E_a)$ y $G_b = (V_b, E_b)$ dos grafos, donde:

- $|V_a| = p_a$ y $|E_a| = q_a$
- $|V_b| = p_b$ y $|E_b| = q_b$

Las matrices de adyacencia de $G_a$ y $G_b$ se denotan como $A = [a_{ij}]$ y $B = [b_{ij}]$, respectivamente.

## 3.2. Matriz de Correspondencia

Se define una matriz $M'$ de dimensiones $p_a \times p_b$, cuyos elementos son 0's y 1's, tal que:

- Cada fila contiene exactamente un 1
- Cada columna contiene a lo sumo un 1

La matriz $M' = [m'_{ij}]$ se utiliza para permutar las filas y columnas de $B$, produciendo una nueva matriz $C$:

$$C = [c_{ij}] = M'(M'B)^t$$

donde $t$ denota transposición.

## 3.3. Condición de Isomorfismo

Un isomorfismo entre $G_a$ y un subgrafo de $G_b$ existe si y solo si:

$$\forall i \in [1, p_a], \forall j \in [1, p_a] : a_{ij} = 1 \Rightarrow c_{ij} = 1$$

En este caso, si $m'_{ij} = 1$, entonces el j-ésimo vértice en $G_b$ corresponde al i-ésimo vértice en $G_a$ en este isomorfismo.

## 3.4. Inicialización

Al inicio del algoritmo, se construye una matriz $M^0 = [m^0_{ij}]$ de dimensiones $p_a \times p_b$ de acuerdo con:

$$m^0_{ij} = \begin{cases} 1 & \text{si grado}(v^b_j) \geq \text{grado}(v^a_i) \\ 0 & \text{en otro caso} \end{cases}$$

donde $v^b_j$ y $v^a_i$ son el j-ésimo vértice de $G_b$ y el i-ésimo vértice de $G_a$, respectivamente.
De la fomra en que se construye la matriz asegura que si $[M^0_{i,j}] = 0$ nunca el nodo i va a poder corresponder con el nodo j en un isomorfismo, esto quiere decir que en el espacio de busqueda resultante que son los 1 en la matriz solo hay posibles soluciones. Luego, basta con generar todas las posibles combinaciones de 1 en esas matrices

## 3.5. Proceso de Búsqueda

El algoritmo genera sistemáticamente todas las matrices $M'$ tales que:

$$\forall i, j : m'_{ij} = 1 \Rightarrow m^0_{ij} = 1$$

Para cada matriz $M'$ generada, el algoritmo verifica si representa un isomorfismo aplicando la condición de isomorfismo mencionada anteriormente.

Las matrices $M'$ se generan cambiando a 0 todos menos uno de los 1's en cada una de las filas de $M^0$, sujeto a la condición de que ninguna columna de $M'$ puede contener más de un 1.

## 3.6. Estructura del Árbol de Búsqueda

En el árbol de búsqueda:

- Los nodos terminales están a una profundidad $d = p_a$ y corresponden a distintas matrices $M'$.
- Cada nodo no terminal a una profundidad $d < p_a$ corresponde a una matriz distinta $M$ que difiere de $M^0$ en que $d$ de las filas tienen todos menos uno de los 1's cambiados a 0.

Este proceso de búsqueda exhaustiva garantiza encontrar todos los isomorfismos de subgrafos entre $G_a$ y $G_b$, si existen.

## 3.7. Mejora del algoritmo

Se busca modificar de manera segura algunos de los unos en la matriz para reducir el tiempo de cómputo. Para ello, se utiliza una observación simple. Si algún vértice $p \in VP$ tiene vecinos $p_1, \ldots, p_l \in P$ y se mapea $p$ a algún vértice $g \in VG$, entonces también deben mapearse los vecinos $p_1, \ldots, p_l$ a vecinos de $g$.

Se entiende que un uno en la posición $(i, j)$ de la matriz $M$ indica que todavía se considera posible que el vértice $v_i \in P$ corresponda al vértice $v_j \in G$. Sin embargo, cuando se descubre que algún vecino del vértice $v_i \in P$ no puede mapearse a ningún vecino del vértice $v_j \in G$, queda demostrado que el uno en la posición $(i, j)$ es incorrecto y puede modificarse de manera segura a cero. Esta modificación puede hacer que otros mapeos sean imposibles, por lo que se itera esta verificación hasta que no sea posible realizar más cambios. Si durante este refinamiento se eliminan todos los unos de alguna fila, se puede dar por finalizado el proceso completo, ya que la matriz $M$ no podrá completarse a un isomorfismo.

## 3.8. Pseudocodigo

```
recurse(used_columns, cur_row, G, P, M')
     if cur_row = num_rows(M)
        if  M' is an isomorphism:
            output yes and end the algorithm
    M = M'
    prune(M)
    for all unused columns c
        set column c in M to 1 and other columns to 0
        mark c as used
        recurse(used_column, cur_row+1, G, P, M)
        mark c as unused

    output no

prune(M)
    do
    for all (i,j) where M is 1
        for all neighbors x of vi in P
            if there is no neighbor y of vj s.t. M(x,y)=1
                M(i,j)=0
    while M was changed
```

## 3.9. Complejidad temporal

Dado que el algoritmo realiza una búsqueda en el espacio de las permutaciones de nodos de un grafo, la búsqueda se hace en $\mathcal{O}(n!)$. Por cada una se hace una comparación entre las matrices de adyacencia en $\mathcal{O}(n^2)$, por lo que la complejidad del algoritmo para el caso peor es de $\mathcal{O}(n! \cdot n^2)$.

La complejidad espacial es de $\mathcal{O}(n^2)$, debido al trabajo con las matrices de adyacencia.

# 4. Algoritmo aproximado

## 4.1. Demostración de condiciones necesarias en grafos isomorfos

Si dos grafos $G_1$ y $G_2$ son isomorfos entonces cumplen que:

1. Ambos grafos deben tener el mismo número de vértices
2. Para cada vértice, el número de conexiones (grado) debe ser el mismo en ambos grafos. Esto significa que la secuencia de grados de los vértices debe coincidir cuando se ordena.

1. Como $G_1$ y $G_2$ son isomorfos existe una función biyectiva $f$ que mapea cada vértice de $G_1$ a $G_2$. Como $f$ es inyectiva entonces a cada nodo de $G_1$ le corresponde un único nodo en $G_2$ y como es sobreyectiva todos los nodos de $G_2$ son cubiertos, por tanto, $G_1$ y $G_2$ poseen la misma cantidad de nodos.

2. Supongamos que el grado de un par de nodos $u \in G_1$ y $v \in G_2$ es distinto ($\text{gr}(u) > \text{gr}(v)$). Dado que $G_1$ y $G_2$ son isomorfos entonces existe $f$ una función sobreyectiva que mapea cada vértice de $G_1$ a $G_2$, y sea $v = f(u)$, luego como $\text{gr}(u) > \text{gr}(f(u))$, va a existir una arista $(u, w) \in G_1$, $(f(u), f(w)) \notin G_2$, lo cual es una contradicción con la definición de isomorfismo.

## 4.2. Algoritmo de verificacion por aristas

- Propósito: Verificar si un grafo es sub-isomorfo a otro representados por matrices de adyacencia $G_1$ y $G_2$, utilizando una estrategia basada en los grados de los nodos y las aristas.
- Entradas: Dos grafos $G_1$ y $G_2$ representados por matrices de adyacencia de $n \times n$, donde $n$ es la cantidad de nodos de cada grafo.
- Salida: Un valor booleano, donde true significa que existe un isomorfismo de subgrafos y false que no.

Pasos del algoritmo

1. **Encontrar subgrafos de tamaño igual a $G_1$ en $G_2$:**
   - Primero, buscamos todos los subgrafos dentro de $G_2$ que tengan el mismo tamaño que $G_1$. Para hacerlo, tomamos todos los posibles subconjuntos de $k$ nodos en $G_2$ y verificamos si forman un subgrafo. Este proceso tiene una complejidad de $\mathcal{O}\left(\binom{n}{k} \cdot n^2\right)$.
2. **Calcular los grados de cada nodo:**
   - Para cada subgrafo encontrado, calculamos el grado de cada nodo, es decir, cuántas aristas están conectadas a cada nodo tanto en el grafo original $G_1$ como en los subgrafos $G_i'$ de $G_2$.
3. **Buscar las aristas entre los nodos:**
   - Recorremos los subgrafos y verificamos si existe una arista entre cada par de nodos $i$ y $j$. Si existe una arista, la guardamos en la forma $(i, j)$, asegurándonos de que el primer nodo (el nodo $i$) tenga el menor grado.
4. **Ordenar y comparar las aristas:**

- Una vez que tenemos todas las aristas, las ordenamos según los grados de los nodos que conectan. Si, después de ordenar las aristas, no coinciden entre los subgrafos y $G_1$, entonces los grafos no son isomorfos.

**Complejidad**

El análisis de la complejidad del algoritmo se puede dividir en dos pasos principales. En el primer paso, buscamos todos los subgrafos dentro del grafo $G_2$ que tengan el mismo tamaño que $G_1$. Para lograr esto, tomamos todos los posibles subconjuntos de $k$ nodos en $G_2$ y verificamos si forman un subgrafo válido. Este proceso tiene una complejidad de $\mathcal{O}\left(\binom{n}{k} \cdot n^2\right)$, donde $\binom{n}{k}$ representa el número de combinaciones posibles de $k$ nodos seleccionados entre $n$ nodos, y $n^2$ corresponde al tiempo necesario para verificar las conexiones entre los nodos seleccionados.

En el segundo paso, procesamos cada uno de los subgrafos encontrados. Para cada nodo dentro del subgrafo, calculamos su grado, lo cual se realiza en $\mathcal{O}(k^2)$, ya que los grafos en cuestión tienen tamaño $k$, y calcular el grado implica recorrer todas las posibles conexiones entre nodos del subgrafo. Adicionalmente, revisamos las aristas existentes entre cada par de nodos y realizamos un ordenamiento de estas aristas. El ordenamiento tiene una complejidad de $\mathcal{O}(m \log m)$, donde $m$ es el número de aristas del subgrafo. En el caso peor, cuando $m = k^2$, esta complejidad se convierte en $\mathcal{O}(k^2 \log k)$. Por lo tanto, la complejidad parcial asociada al procesamiento de cada subgrafo es $\mathcal{O}(k^2 \log k)$.

Finalmente, combinando ambos pasos, la complejidad total del algoritmo es $\mathcal{O}\left(\binom{n}{k} \cdot n^2 \cdot k^2 \cdot \log k\right)$. Este algoritmo es mas lento para el caso peor pero para determinados casos, especificamente para los casos en que:

$$\frac{n^2 \cdot \log n}{(n-k)! \cdot k!} \leq 1$$

es mejor que el primero que se analiza en este trabajo siendo su complejidad $O(n! * n^2)$

En cuanto a la complejidad espacial, esta es $\mathcal{O}\left(\binom{n}{k} \cdot n^2\right)$, debido al uso de matrices de adyacencia.

# Referencias

Ullmann, Julian R. (1976), Ün algoritmo para el isomorfismo de subgrafos", Journal of the ACM , 23 (1): 31– 42, doi : 10.1145/321921.321925 , S2CID 17268751.
https://en-m-wikipedia-org.translate.goog/wiki/Subgraph_isomorphism_problem?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=wa
https://adriann.github.io/Ullman%20subgraph%20isomorphism.html#fn1