Práctica 1
Representación del Conocimiento

David Ruiz Rodríguez, Mohamed Rodrigo El Badry,
Nicolas Recinella Vidán, Denilson Palomino Adán, Javier Lamas

3 de octubre de 2024

Índice

1. Enunciado

En esta práctica debemos familiarizarnos con el manejo de grafos en python, implementar el algoritmo enseñado en clase para decidir $X \perp_G Y|Z$ y calcular el coste de dicho algoritmo.

Reparto de trabajo:

- Pseudocódigo: Nicolas y Mohamed.
- Codificación en python: David.
- Ejemplos: Denilson.
- Calculo del coste: Javier.

2. Codificación del Algoritmo Descrito

2.1. Pseudo-Código

Algorithm 1: Algoritmo_Separacion(G[V[1..n],A[1..m]], x_y, z)

```
xy_union_z ← x ∩ y ∩ z
// Paso 1: Eliminar nodos hoja que no esten en xy_union_z
explorados ← []
for nodo ∈ V do
    if nodo ∉ explorados then
        elimina_hojas_recursivo(G, nodo, x_y, z, explorados)
    end
end
// Paso 2: Unir padres con hijos en común e ignorar la dirección de las
   aristas
nuevas_aristas ← []
// Buscamos padres con hijos en común
for nodo ∈ V do
    hijos ← G.sucesores(nodo)
    for hijo ∈ hijos do
        padres ← G.predecesores(hijo)
        if padres.length() > 1 then
            for i = 0; i < padres.length(); i++ do
                for j = i + 1; j < padres.length(); j++ do
                    nuevas_aristas ← (padres[i], padres[j])
                end
            end
        end
    end
end
// Añadimos las aristas entre padres con el mismo hijo
 A ← nuevas_aristas
// Convertimos el grafo en no dirigido
G2 ← []
G2 ← V, A
// Paso 3: Ver si hay cominos entre X e Y en el grafo que no pasen por Z
G2 ← G2 − {z}
nodos_visitados ← []
return not busqueda_camino(G2, x, y, nodos_visitados)
```

Algorithm 2: elimina_hojas_recursivo(G,nodo,x_y,z,explorados)

$explorados \leftarrow nodo$
for $i \in nodo.hijos()$ **do**
    if $i \notin explorados$ **then**
        $elimina\_hojas\_recursivo(G, i, x\_y, z, explorados)$
    end
end
if $nodo \notin x\_y$ ***and*** $nodo \notin z$ ***and*** $nodo.hijos() == 0$ **then**
    $G.remove(nodo)$
end

Algorithm 3: busqueda_camino(G,x,y,nodos_visitados)

$nodos\_visitados \leftarrow x$
for $nodo \in x.vecinos()$ **do**
    if $nodo \notin nodos\_visitados$ **then**
        if $nodo == y$ **then**
            return True
        else
            return $busqueda\_camino(G, nodo, y, nodos\_visitados)$
        end
    end
end
return False

El código en python se encuentra anexado a este documento.

3. Eficacia del Código Propuesto

Los gráficos probados son los siguientes:

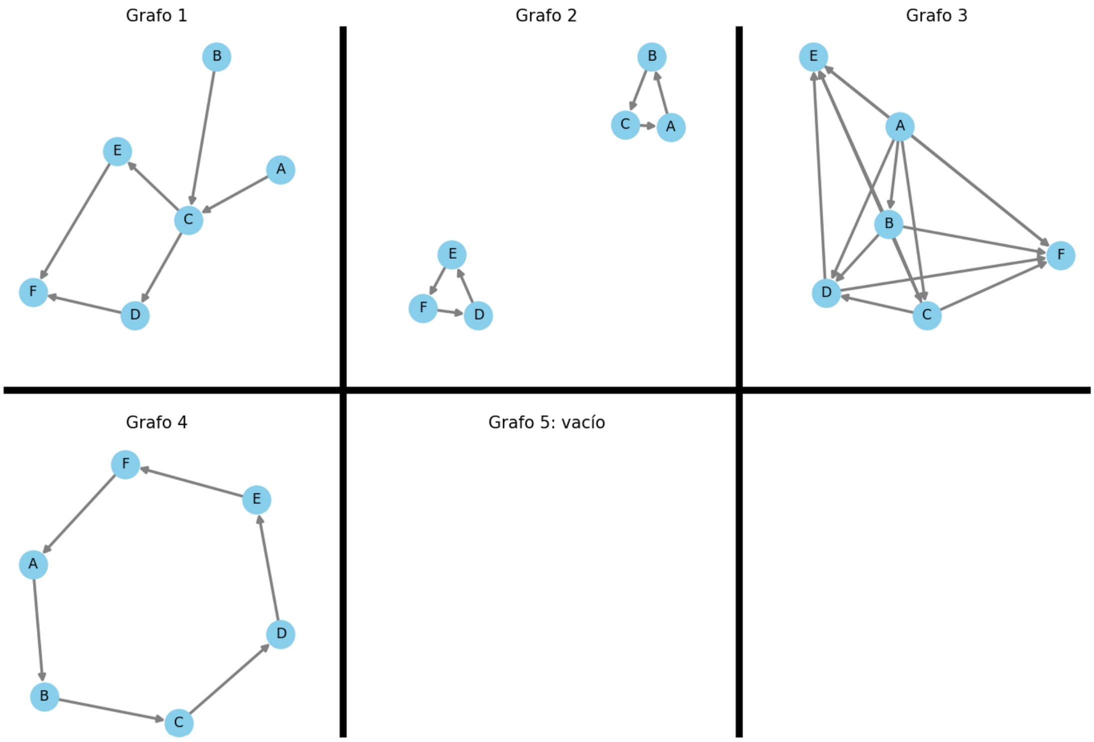

Figura 1: Conjunto de prueba

Sobre estos gráficos se realizan varias pruebas.

- Grafo 1:
  - Ejemplo 1:
    - $X = A$, $Y = C$ y $Z = E$
    - Resultado: False, no son separables.
  - Ejemplo 2:
    - $X = D$, $Y = E$ y $Z = C$
    - Resultado: True, sí son separables.
- Grafo 2:
  - Ejemplo 1:
    - $X = A$, $Y = C$ y $Z = B$
    - Resultado: False, no son separables.
  - Ejemplo 2:
    - $X = B$, $Y = D$ y $Z = E$
    - Resultado: True, sí son separables.
- Grafo 3:

- Ejemplo 1:
  - $X = A$, $Y = D$ y $Z = E$
  - Resultado: False, no son separables.
- Ejemplo 2:
  - $X = A$, $Y = C$ y $Z = D, E, F$
  - Resultado: False, no son separables.

- Grafo 4:
  - Ejemplo 1:
    - $X = B$, $Y = D$ y $Z = A, F$
    - Resultado: False, no son separables.
  - Ejemplo 2:
    - $X = B$, $Y = D$ y $Z = A, C, E, F$
    - Resultado: True, sí son separables.

4. Calculo del Coste Temporal

El algoritmo implementado para decidir si $X \perp_G Y \mid Z$ tiene varias etapas, y cada una de ellas tiene un impacto en el coste temporal global. A continuación, desglosamos el coste de cada paso del algoritmo:

4.1. Paso 1: Limpieza del grafo

El primer paso del algoritmo consiste en eliminar los nodos hoja que no forman parte de los conjuntos X, Y o Z. Para ello, se recorre todo el grafo, explorando cada nodo y sus vecinos. Esta operación se realiza mediante la función `elimina_hojas_recursivo`, que explora el grafo de manera recursiva, eliminando los nodos que cumplen con las condiciones.

Dado que en este paso recorremos todos los nodos y aristas del grafo al menos una vez, el coste temporal de esta fase es $O(n+m)$, donde n es el número de nodos y m el número de aristas del grafo.

4.2. Paso 2: Unión de padres y eliminación de la direccionalidad

En este paso, nos centramos en unir los padres que comparten un mismo hijo. Para hacerlo, recorremos todas las aristas del grafo dos veces. Primero, identificamos los padres de cada hijo y luego añadimos una arista entre cada par de padres que comparten un hijo.

Este proceso implica recorrer las aristas en bucles anidados, lo que genera un coste cuadrático. Por tanto, el coste de este paso es $O(m^2)$. Después de esto, eliminamos la direccionalidad del grafo, lo que también contribuye al coste de recorrer las aristas.

4.3. Paso 3: Búsqueda de caminos

Una vez modificado el grafo, el siguiente paso es verificar si existe un camino entre los nodos X e Y sin pasar por los nodos en Z. Para esto, se utiliza una búsqueda de caminos, , cuya complejidad es $O(n+m)$.

4.4. Coste total

Sumando los costes de cada paso, obtenemos que el coste total del algoritmo es:

$$O(n+m+m^2)$$

El término dominante es $O(m^2)$, lo que significa que, a medida que el número de aristas aumenta, el tiempo de ejecución del algoritmo crecerá de manera cuadrática.

.