Demostraciones de NP-Completitud en Problemas

Daniel Machado Pérez
Diseño y Análisis de Algoritmos

Índice

Base Teórica .Problemas NP-Completos

- SAT
- 3-SAT
- k-Clique
- Conjunto Independiente
- Vertex Cover
- Subset Sum
- Mochila
- Hamilton (Dirigido)
- Viajante

1. Exact Cover

El problema **Exact Cover** puede describirse formalmente como sigue:
Entrada: Un conjunto finito X y una colección S de subconjuntos de X.
Pregunta: ¿Existe un subcolector $S' \subseteq S$ tal que cada elemento de X aparezca **exactamente una vez** en los subconjuntos de S'?

1.1. Pertenencia a NP

Para demostrar que **Exact Cover** pertenece a la clase NP, debemos mostrar que, dada una solución candidata S', es posible verificar en tiempo polinomial si S' constituye una solución válida.

Dado que $|X| = n$ y $|S| = m$, la verificación puede realizarse de la siguiente manera:

1. Crear un registro inicializado en cero para cada elemento $x \in X$.
2. Para cada subconjunto $S_i \in S'$, incrementar el contador asociado a cada elemento $x \in S_i$.
3. Comprobar que el contador de cada elemento en X es exactamente igual a uno. Si esta condición se cumple, entonces S' es una solución válida.

Conclusión: Como la verificación puede completarse en tiempo polinomial respecto al tamaño de la entrada, **Exact Cover** pertenece a NP.

1.2. NP-Hardness

Demostraremos que **Exact Cover** es NP-Hard mediante una reducción desde **3-SAT** en tiempo polinomial

1.2.1. Demostración. Reducción desde 3-SAT

Dada una fórmula booleana F en forma normal conjuntiva con n variables $x_1, x_2, \ldots, x_n$ y m cláusulas $C_1, C_2, \ldots, C_m$, donde cada cláusula C_i está formada por l_i literales L_{ik}, construiremos una instancia del problema **Exact Cover** como sigue:

1. Definimos el universo U como:

$$U = \{x_1, x_2, \ldots, x_n\} \cup \{C_1, C_2, \ldots, C_m\} \cup \{p_{ik} \mid C_i \text{ contiene el literal } L_{ik}\}.$$

2. Construimos los subconjuntos S de U de la siguiente manera:
   - Para cada literal p_{ik}, agregamos un subconjunto $S_{p_{ik}} = \{p_{ik}\}$.
   - Para cada variable x_j:
     - Si x_j es *True (T)* agregamos $S_{x_j,T}$, que contiene x_j y todas sus instancias negativas.
     - Si x_j es *False F* agregamos $S_{x_j,F}$, que contiene x_j y todas sus instancias positivas.
   - Para cada cláusula C_i, agregamos subconjuntos $S_{C_{ik}} = \{C_i, p_{ik}\}$ para cada literal en C_i.

Probemos que: Si F es satisfacible, entonces la instancia de Exact Cover es *True*.
Sea t una asignación de verdad que satisface F:

- Para cada variable x_j, seleccionamos $S_{x_j,T}$ si $t(x_j) = \mathit{True}$, o $S_{x_j,F}$ si $t(x_j) = \mathit{False}$.
- Para cada cláusula C_i, seleccionamos un subconjunto $S_{C_{ik}}$ asociado a un literal L_{ik} que sea verdadero en la asignación satisfactoria.
- Para terminar, para cada literal p_{ik} que no haya sido cubierto, seleccionamos $S_{p_{ik}}$.

Esto cubre exactamente todos los elementos en $U \Rightarrow$ cumple **Exact Cover**.

Probemos que: Si la instancia de Exact Cover tiene solución, entonces F es satisfacible. Sea C una solución de **Exact Cover**:

- Para cada variable x_j, el subconjunto seleccionado en C determina si $x_j = \mathit{True}$ o $x_j = \mathit{False}$.
- Para cada cláusula C_i, existe un conjunto $S_{C_{jk}}$ en C que cubre C_i. Por lo tanto, al menos uno de sus literales debe ser verdadero. Entonces, al menos un literal de cada cláusula debe ser verdadero.

Por lo tanto, la asignación correspondiente satisface F.

Conclusión: Dado que F es satisfacible si y solo si la instancia de **Exact Cover** tiene solución, y la reducción se realiza en tiempo polinomial, **Exact Cover** es NP-Hard.

1.3. NP-Completitud

Dado que **Exact Cover** pertenece a NP y es NP-Hard, concluimos que es NP-Completo.

2. 3-Coloreo

El problema **3-Coloreo** puede describirse formalmente de la siguiente manera:
Entrada: Un grafo no dirigido $G = (V, E)$.
Pregunta: ¿Es posible asignar un color a cada vértice de G utilizando a lo sumo 3 colores, de forma que dos vértices adyacentes no compartan el mismo color?

2.1. Pertenencia a NP

Para demostrar que **3-Coloreo** pertenece a NP, debemos mostrar que, dada una solución candidata (es decir, una asignación de colores a los vértices de G), es posible verificar en tiempo polinomial si esta solución es válida.

1. Verificar que cada vértice tiene asignado uno de los 3 colores. Esto toma tiempo $O(|V|)$.
2. Para cada arista $(u, v) \in E$, verificar que los colores asignados a u y v son distintos. Esto toma tiempo $O(|E|)$.

Dado que el tiempo total para realizar esta verificación es $O(|V| + |E|)$, el problema **3-Coloreo** pertenece a NP.

2.2. NP-Hardness

Demostraremos que **3-Coloreo** es NP-Hard mediante una reducción polinomial desde el problema **3-SAT**, el cual es conocido como NP-Completo.

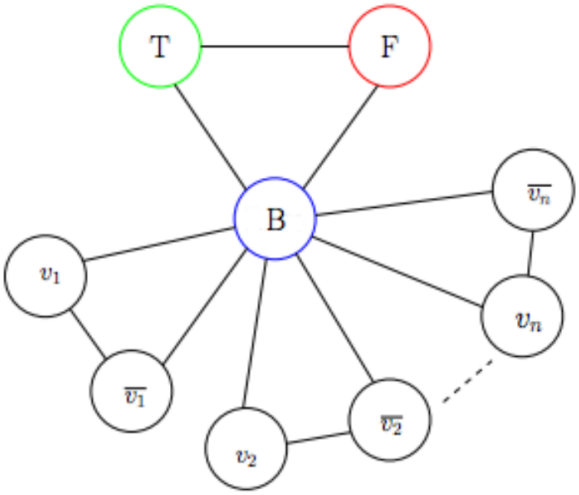
2.2.1. Demostración. Reducción desde 3-SAT

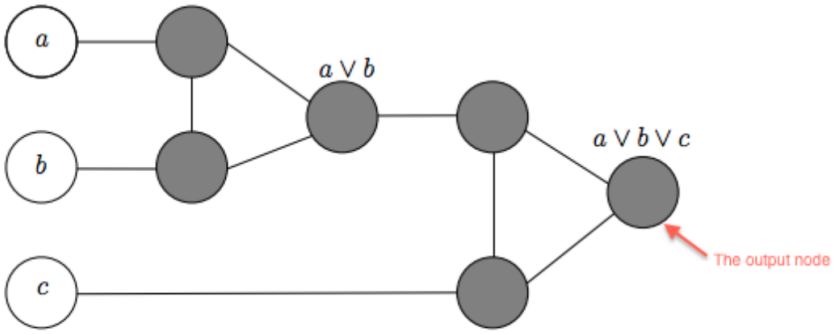
Dada una fórmula booleana F en forma normal conjuntiva con n variables $x_1, x_2, \ldots, x_n$ y m cláusulas $C_1, C_2, \ldots, C_m$, construiremos un grafo $G = (V, E)$ tal que:

- Cada solución de F se corresponde con una coloración válida de G utilizando exactamente 3 colores.
- Si G tiene una coloración válida con 3 colores, entonces F es satisfacible.

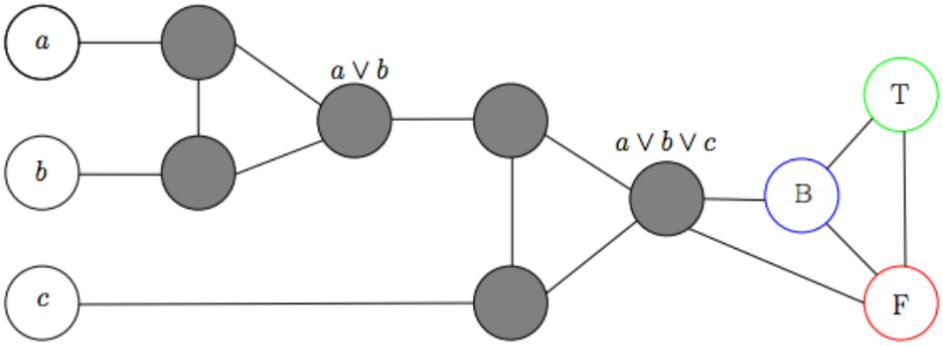
Construcción del grafo G:

1. Crear un triángulo en G con tres vértices etiquetados como T, F, y B (que representan *True*, *False* y *Base*), conectados entre sí. Esto asegura que T, F y B reciban colores diferentes.
2. Para cada variable x_i, agregar dos vértices etiquetados como v_i y $\overline{v_i}$, y conectarlos tanto al vértice B como entre ellos. Esto garantiza que uno de ellos reciba el color T y el otro el color F, representando una asignación de verdad.
3. Para cada cláusula $C_j = (a \vee b \vee c)$, introducir un *OR-gadget* (**Clause Satisfiability Gadget**) con vértices y conexiones que capturen el valor de $a \vee b \vee c$. El gadget se conecta a los vértices correspondientes de los literales a, b, y c, y a los vértices B y F. Esto permite que si al menos uno de los literales tiene el color T, el nodo salida también se colorea como T, y solo se coloreará como F cuando todos los literales se coloreen como F.





Finalmente, el vértice de salida de cada gadget se conecta a los vértices B y F del triángulo inicial, lo que asegura que solo pueda ser coloreado como T .



Probemos que: Si F es satisfacible, entonces G es 3-colorable.
Sea t una asignación de verdad que satisface F. Asignamos colores a los vértices de G como sigue:

- Si una variable x_i es verdadera en t, coloreamos el vértice x_i con el color T y el vértice $\neg x_i$ con el color F.
- Para cada cláusula C_j, al menos uno de los literales en C_j es verdadero. Usamos el *OR-gadget* para colorear los vértices de manera que el nodo de salida reciba el color T, reflejando la satisfacibilidad de la cláusula.

Esto asegura que no haya dos vértices adyacentes con el mismo color.

Probemos que: Si G es 3-colorable, entonces F es satisfacible.
Dada una coloración válida de G, asignamos valores de verdad a las variables de F como sigue:

- Si el vértice x_i tiene el color T, asignamos *True* a x_i. Si el vértice $\neg x_i$ tiene el color T, asignamos *False* a x_i.
- Para cada cláusula C_j, al menos uno de los vértices correspondientes a sus literales debe tener el color T, garantizando que la cláusula esté satisfecha.

Por lo tanto, t satisface F.

Conclusión: Como la construcción de G se realiza en tiempo polinomial y la satisfacibilidad de F es equivalente a la colorabilidad de G, **3-Coloreo** es NP-Hard.

2.3. NP-Completitud

Dado que **3-Coloreo** pertenece a NP y es NP-Hard, concluimos que es NP-Completo.

3. Número Cromático

El problema del **Número Cromático** se define como sigue:

Definición: El número cromático de un grafo $G = (V, E)$ es el número mínimo de colores necesarios para colorear los vértices del grafo de manera que dos vértices adyacentes no compartan el mismo color.

Problema: Determinar el número cromático de un grafo G.

3.1. Pertenencia a NP

A diferencia de los problemas de decisión como **3-Coloreo**, no se conoce un algoritmo polinomial para determinar el número cromático de un grafo, por lo que no podemos asegurar su pertenencia o no a la clase NP.

3.2. NP-Hardness

Sin embargo, podemos demostrar que determinar el número cromático es **NP-Hard** al reducir polinomialmente desde un problema NP-Completo como **3-Coloreo**.

3.2.1. Demostración. Reducción desde 3-Coloreo

El problema **3-Coloreo** es un caso especial del problema del número cromático, donde se pregunta si el número cromático de un grafo es a lo sumo 3. Aprovecharemos esta relación para construir la reducción.

Para ello debemos convertir una entrada de **3-Coloreo** en una entrada de **Número Cromático** en tiempo polinomial y resolver **3-Coloreo** a partir de la salida de **Número Cromático**, también en tiempo polinomial. La entrada de **3-Coloreo** es un grafo $G = (V, E)$, que será la misma entrada de **Número Cromático**. Luego, al resolver **Número Cromático** obtenemos la cantidad mínima de colores necesarios para colorear el grafo sin que dos nodos adyacentes tengan el mismo color. Convertiremos esa salida en una solución para **3-Coloreo** de la siguiente forma:

- Si la cantida de colores es menor o igual a 3 ⇒ **3-Coloreo** es *True*.
- Si la cantidad de colores es mayor que 3 ⇒ **3-Coloreo** es *False*.

Por lo tanto, resolver el problema del **Número cromático** en G nos permite decidir si G es 3-colorable, completando la reducción.

3.2.2. Conclusión

Como el problema **3-Coloreo** es NP-Completo, y hemos demostrado que puede reducirse polinomialmente al problema del **Número cromático**, concluimos que determinar el número cromático es **NP-Hard**.

3.3. NP-Completitud

Dado que no se conoce si el problema del **Número cromático** pertenece a NP, tampoco se puede afirmar que sea NP-Completo.

4. Clique Máximo

Definición: Un clique es un subgrafo completo dentro de un grafo. Formalmente, un clique en un grafo $G = (V, E)$ es un subconjunto de vértices $C \subseteq V$, tal que todos los pares de vértices en C están conectados directamente por una arista. En otras palabras, todos los vértices del clique están mutuamente conectados.

Problema: Determinar el clique de mayor tamaño en un grafo G.

4.1. Pertenencia a NP

A diferencia de los problemas de decisión como **k-Clique**, no se conoce un algoritmo polinomial para determinar el clique de mayor tamaño en un grafo, por lo que no podemos asegurar su pertenencia o no a la clase NP.

4.2. NP-Hardness

Sin embargo, podemos demostrar que determinar el clique máximo es **NP-Hard** al reducir polinomialmente desde un problema NP-Completo como **k-Clique**.

4.2.1. Demostración. Reducción desde k-Clique

El problema **k-Clique** es un caso especial del problema del clique máximo, donde se pregunta si existe un clique de tamaño exactamente k. Aprovecharemos esta relación para construir la reducción.

Para ello debemos convertir una entrada de **k-Clique** en una entrada de **Clique Máximo** en tiempo polinomial y resolver **k-Clique** a partir de la salida de **Clique Máximo**, también en tiempo polinomial. La entrada de **k-Clique** es un grafo $G = (V, E)$ y un entero k. La entrada para **Clique Máximo** será el mismo grafo $G = (V, E)$. Luego, al resolver **Clique Máximo** obtenemos el clique máximo y su tamaño en G. Convertiremos esa salida en una solución para **k-Clique** de la siguiente forma:

- Si el tamaño del clique máximo es mayor o igual a k ⇒ **k-Clique** es *True*.

- Si el tamaño del clique máximo es menor que $k \Rightarrow$ **k-Clique** es *False*.

Por lo tanto, resolver el problema del **Clique Máximo** en G nos permite decidir si G tiene un clique de tamaño k, completando la reducción.

4.2.2. Conclusión

Como el problema **k-Clique** es NP-Completo, y hemos demostrado que puede reducirse polinomialmente al problema del **Clique Máximo**, concluimos que determinar el clique máximo es **NP-Hard**.

4.3. NP-Completitud

Dado que no se conoce si el problema del **Clique Máximo** pertenece a NP, tampoco se puede afirmar que sea NP-Completo.

5. Cobertura de Cliques

Definición: Dado un grafo $G = (V, E)$, una cobertura de cliques es un conjunto de cliques $C_1, C_2, \ldots, C_k$ tal que cada arista $(u, v) \in E$ pertenece a al menos uno de estos cliques.

Problema: Determinar el número mínimo de cliques necesarios para cubrir todas las aristas del grafo.

5.1. Pertenencia a NP

El problema de **Cobertura de Cliques** pertenece a NP, ya que, dado un conjunto de cliques, es posible verificar en tiempo polinomial si cada arista en E está cubierta por al menos uno de los cliques.

5.2. NP-Hardness

Podemos demostrar que **Cobertura de Cliques** es NP-Hard al reducir polinomialmente desde un problema NP-Completo como **Número Cromático**.

5.2.1. Demostración. Reducción desde Número Cromático

El problema **Número Cromático** pregunta por el número mínimo de colores necesarios para colorear los vértices de un grafo de modo que no haya dos vértices adyacentes con el mismo color. Se conoce que un conjunto de vértices en un grafo G es un clique si y solo si es un conjunto independiente en el grafo complemento G^c. Una partición de los vértices de G en cliques (cobertura de cliques de G) corresponde a una partición de G^c en conjuntos independientes. Una coloración de un grafo es un partición de sus vértices en conjuntos independientes. Por lo tanto, una cobertura de cliques de G será una coloración de G^c, por lo que la cobertura mínima de G será el número cromático de G^c.

Aprovecharemos esta relación para construir la reducción. La entrada de **Número Cromático** es un grafo $G = (V, E)$.

Construcción: Dado un grafo $G = (V, E)$, construimos un grafo complementario $G^c = (V, E^c)$, donde E^c contiene todas las aristas que no están en E. La entrada para **Cobertura de Cliques** será G^c. Luego de resolver **Cobertura de Clique** en G^c obtendremos un k que es la mínima cantidad de cliques en que podemos particionar el grafo tal que cada arista esté contenida en al menos uno de esos cliques, que es equivalente a la cantidad de conjuntos independientes de G, que a su vez es equivalente a su número cromático.

Por lo tanto, resolver el problema de **Cobertura de Cliques** en G^c nos permite decidir el número cromático de G, completando la reducción.

5.2.2. Conclusión

Como el problema **Número Cromático** es NP-Completo, y hemos demostrado que puede reducirse polinomialmente al problema de **Cobertura de Cliques**, concluimos que **Cobertura de Cliques** es **NP-Hard**.

5.3. NP-Completitud

Como **Cobertura de Cliques** es **NP** y **NP-Hard**, podemos concluir que es **NP-Completo**.

6. Conjunto Dominante

Definición: En un grafo $G = (V, E)$, un conjunto de vértices $D \subseteq V$ es un conjunto dominante si cada vértice de V que no está en D es adyacente a al menos un vértice en D.

Problema: Determinar el conjunto dominante de menor cardinalidad en un grafo G.

6.1. Pertenencia a NP

El problema **Conjunto Dominante** pertenece a NP porque, dado un conjunto candidato D, podemos verificar en tiempo polinomial si D domina todos los vértices de G.

6.2. NP-Hardness

Podemos demostrar que **Conjunto Dominante** es NP-Hard mediante una reducción desde el problema **Vertex Cover**, que se conoce que es NP-Completo.

6.2.1. Demostración. Reducción desde Vertex Cover

Dado un grafo $G = (V, E)$ como entrada para el problema **Vertex Cover**, construiremos un grafo $G' = (V', E')$ tal que:

- V' contiene todos los vértices de V y, por cada arista $(u, v) \in E$, un nuevo vértice w.
- E' incluye todas las aristas de E más dos nuevas aristas (u, w) y (w, v) para cada arista (u, v) de E.

La construcción de G' se realiza en tiempo polinomial, ya que solo duplicamos las aristas y agregamos un nuevo vértice por cada arista original.

Probemos que: Si S es un vertex cover en G, entonces S también es un conjunto dominante en G'.
Como S es un vertex cover, cada arista en G tiene al menos un extremo en S. Consideremos $v \in G'$. Si v es un nodo original de G, entonces o $v \in S$ o debe existir alguna arista que conecte a v a otro nodo u. Como S es un vertex cover, si $v \notin S$, entonces u tiene que estar en S, por lo que habría un nodo adyacente a v en S. Entonces v está cubierto por algún elemento de S. Por otra parte, si w es un nodo adicional en G' entonces tiene dos nodos adyacentes u y $v \in G$ y utilizando el argumento anterior al menos uno de ellos está en S. Entonces los nodos adicionales también están cubiertos por S. Entonces si G tiene un vertex cover, G' tendrá un conjunto dominante de al menos el mismo tamaño.

Probemos que: Si D es un conjunto dominante de tamaño k de G, entonces en G' existe eun vertex cover de tamaño a lo sumo k.
Si D es un conjunto dominante de tamaño k en G', consideremos todos los vértices adicionales $w \in D$. Observemos que w debe estar conectado exactamente a dos vértices $u, v \in G$. Ahora, podemos reemplazar w de manera segura por u o v. El vértice w en D nos ayudará a dominar solo a $u, v, w \in G'$. Sin embargo, estas tres aristas forman un ciclo de 3 (*3-cycle*), por lo que podemos elegir u o v y seguir dominando todos los vértices que w dominaba anteriormente. De esta manera, podemos eliminar todos los vértices adicionales de la forma descrita. Dado que todos los vértices adicionales corresponden a una de las aristas en G, y dado que todos los vértices adicionales están cubiertos por el conjunto D modificado, esto significa que todas las aristas en G están cubiertas por el conjunto. Por lo tanto, si G' tiene un conjunto dominante de tamaño k, entonces G tiene un vertex cover de tamaño a lo sumo k.

Así, hemos probado ambos lados de la equivalencia. Existe un conjunto dominante de tamaño k en G' si y solo si existe un vertex cover de tamaño k en G.

6.3. Conclusión

Dado que sabemos que el problema **Vertex Cover** es NP-Completo, y demostramos que es posible hacer una reducción en tiempo polinomial de **Vertex Cover** a **Conjunto Dominante**, el problema **Conjunto Dominante** es NP-Hard.

6.4. NP-Completitud

Como **Conjunto Dominante** es NP-Hard y pertenece a NP, concluimos que es NP-Completo.

7. Retroalimentación de Vértices

Definición: Dado un grafo $G = (V, E)$, un conjunto de retroalimentación de vértices es un subconjunto de vértices $F \subseteq V$ tal que al eliminar todos los vértices en F (y sus aristas incidentes), el grafo resultante no contiene ciclos (es decir, el grafo resultante es acíclico o un bosque, si es no dirigido).

Problema: Encontrar el conjunto de retroalimentación de vértices de tamaño mínimo.

7.1. Pertenencia a NP

El problema **Retroalimentación de Vértices** pertenece a NP porque, dado un conjunto candidato F, podemos verificar en tiempo polinomial si eliminar F hace que el grafo sea acíclico.

7.2. NP-Hardness

Podemos demostrar que **Retroalimentación de Vértices** es NP-Hard mediante una reducción desde el problema **Vertex Cover**, que es conocido como NP-Completo.

7.2.1. Demostración. Reducción desde Vertex Cover

Dado un grafo no dirigido $G = (V, E)$ y un entero k como entrada para el problema **Vertex Cover**, construiremos un grafo $G' = (V', E')$ como sigue:

- Por cada vértice $v \in V$, incluimos un vértice correspondiente $v \in V'$.
- Por cada arista $e = (u, v) \in E$, agregamos un nuevo vértice w en V'.
- En E', incluimos las aristas (u, v), (u, w), y (w, v) para cada arista $e = (u, v)$ en E.

Esta construcción de G' se realiza en tiempo polinomial, ya que simplemente agregamos un nuevo vértice y dos nuevas aristas por cada arista en G.

- **Probemos que: Si S es un vertex cover en G de tamaño k, entonces S también es un conjunto de retroalimentación de vértices en G' de tamaño k.**
  Como S es un vertex cover, cada arista (u, v) en G tiene al menos un extremo en S. En G', cualquier ciclo debe incluir una de las aristas (u, v), (u, w) o (w, v). Dado que S contiene al menos uno de los extremos de (u, v), cualquier ciclo en G' intersecta con S. Por lo tanto, eliminar S asegura que G' sea acíclico.

- **Probemos que: Si F es un conjunto de retroalimentación de vértices en G' de tamaño k, entonces F contiene un conjunto equivalente en G que es un vertex cover.**
  Consideremos cualquier ciclo en G'. Dicho ciclo debe incluir uno de los vértices adicionales w correspondientes a una arista (u, v) en G. En este caso, podemos reemplazar w por uno de sus extremos u o v en F, ya que eliminar u o v es suficiente para romper todos los ciclos que incluyen w. Finalmente, dado que todos los ciclos en G' se corresponden con las aristas de G, F debe contener un vértice de cada arista en G, lo que implica que F es un vertex cover en G.

Dado que S es un vertex cover de tamaño k en G si y solo si F es un conjunto de retroalimentación de vértices de tamaño k en G', y como sabemos que **Vertex Cover** es NP-Completo, hemos probado que **Retroalimentación de Vértices** es NP-Hard.

7.3. NP-Completitud

Como **Retroalimentación de Vértices** es NP-Hard y pertenece a NP, concluimos que es NP-Completo.

8. Retroalimentación de Arcos

Definición: Dado un grafo dirigido $G = (V, E)$, un conjunto de retroalimentación de arcos es un subconjunto de arcos $F \subseteq E$ tal que, al eliminar todos los arcos en F, el grafo resultante no contiene ciclos (es decir, el grafo resultante es acíclico).

Problema: Encontrar el conjunto de retroalimentación de arcos de tamaño mínimo.

8.1. Pertenencia a NP

El problema **Retroalimentación de Arcos** pertenece a NP porque, dado un conjunto candidato F, podemos verificar en tiempo polinomial si eliminar F hace que el grafo sea acíclico.

8.2. NP-Hardness

Podemos demostrar que **Retroalimentación de Arcos** es NP-Hard mediante una reducción desde el problema **Retroalimentación de Vértices**, que demostramos que es NP-Completo.

8.2.1. Demostración. Reducción desde Retroalimentación de Vértices

Dado un grafo $G = (V, E)$ y un entero k como instancia del problema **Retroalimentación de Vértices**, construiremos una instancia (G', k') del problema **Retroalimentación de Arcos** como sigue:

- Por cada vértice $v \in V$, creamos dos nuevos vértices v_1 y v_2 en G' y añadimos el arco interno (v_1, v_2).
- Por cada arco $(u, v) \in E$ en el grafo original, añadimos un arco externo (u_2, v_1) en G'.

Gracias a esto, cada vértice en G se duplica en G'. Los arcos del grafo original se dividen en:

- **Arcos internos:** Entre las dos copias de un mismo vértice, (v_1, v_2).
- **Arcos externos:** Entre las copias de vértices diferentes, (u_2, v_1).

Resolver **Retroalimentación de Arcos** en G' equivale a encontrar un conjunto de arcos cuyo borrado elimine todos los ciclos en G'. Nótese que cualquier ciclo que contenga arcos externos también debe pasar por al menos un arco interno $\Rightarrow$ eliminar sólo arcos internos (que corresponden a eliminar vértices en G) es suficiente para romper los ciclos.

Correctitud:

- Cualquier conjunto de arcos internos que forme una solución para **Retroalimentación de Arcos** en G' corresponde a un conjunto de vértices que forman una solución para **Retroalimentación de Vértices** en G.
  Esto se debe a que romper los ciclos en G' eliminando arcos internos es equivalente a romper los ciclos en G eliminando los vértices correspondientes.

- Por otro lado, cualquier conjunto de vértices en G que rompa todos los ciclos se puede traducir directamente a un conjunto de arcos internos en G' que rompa los ciclos en G'.

Por lo tanto, resolver **Retroalimentación de Arcos** en G' es equivalente a resolver **Retroalimentación de Vértices** en G.

8.2.2. Conclusión

Dado que **Retroalimentación de Vértices** es NP-Completo, y hemos demostrado que se puede reducir polinomialmente al problema **Retroalimentación de Arcos**, concluimos que **Retroalimentación de Arcos** es NP-Hard.

8.3. NP-Completitud

Como **Retroalimentación de Arcos** es NP-Hard y pertenece a NP, concluimos que es NP-Completo.

9. NAE 4-SAT (Not-All-Equal 4-SAT)

Definición: Dada una fórmula booleana en CNF donde cada cláusula tiene exactamente 4 literales, determinar si existe una asignación de valores de verdad a las variables de modo que en cada cláusula no todos los literales sean verdaderos ni todos sean falsos.

9.1. Pertenencia a NP

El problema **NAE 4-SAT** pertenece a NP porque, dada una asignación de valores de verdad a las variables, podemos verificar en tiempo polinomial si no todos los literales en cada cláusula son iguales y la fórmula es satisfacible.

9.2. NP-Hardness

Demostraremos que **NAE 4-SAT** es NP-Hard mediante una reducción desde **3-SAT**.

9.2.1. Demostración. Reducción desde 3-SAT

Dada una instancia de **3-SAT** ϕ, construiremos una instancia equivalente de **NAE 4-SAT** ϕ' en tiempo polinomial. Para ello agregamos una nueva variable z a cada cláusula de ϕ. Demostremos que ϕ es satisfacible si y solo si ϕ' es satisfacible.

($\Rightarrow$) Si $x_1, \ldots, x_n$ es una asignación satisfacible para ϕ, entonces la misma asignación satisface ϕ' cuando elegimos $z = 0$.

($\Leftarrow$) Supongamos que $x_1, \ldots, x_n, z$ es una asignación satisfacible de ϕ'. Nótese que $\neg x_1, \ldots, \neg x_n, \neg z$ también es una asignación satisfacible de ϕ' (porque $NAE(a, b, c, d) = NAE(\neg a, \neg b, \neg c, \neg d)$). En una de estas dos asignaciones, el valor asignado a z es 0. Esa asignación corresponde a una asignación satisfacible para ϕ.

9.2.2. Conclusión

Como **3-SAT** es NP-Completo, y hemos demostrado una reducción polinomial hacia **NAE 4-SAT**, concluimos que **NAE 4-SAT** es NP-Hard.

9.3. NP-Completitud

Como **NAE 4-SAT** pertenece a NP y es NP-Hard, es NP-Completo.

10. NAE 3-SAT (Not-All-Equal 3-SAT)

Definición: Dada una fórmula booleana en CNF donde cada cláusula tiene exactamente 3 literales, determinar si existe una asignación de valores de verdad a las variables de modo que en cada cláusula no todos los literales sean verdaderos ni todos sean falsos.

10.1. Pertenencia a NP

El problema **NAE 3-SAT** pertenece a NP porque, dada una asignación de valores de verdad a las variables, podemos verificar en tiempo polinomial si no todos los literales en cada cláusula son iguales y la fórmula sea satisfacible.

10.2. NP-Hardness

Demostraremos que **NAE 3-SAT** es NP-Hard mediante una reducción desde **NAE 4-SAT**.

10.2.1. Demostración. Reducción desde NAE 4-SAT

Dada una instancia de **NAE 4-SAT** ϕ, construiremos una instancia equivalente de **NAE 3-SAT** ϕ' en tiempo polinomial. Para cada cláusula $\text{NAE}(a, b, c, d)$ en ϕ:

- Introducimos una nueva variable w.
- Reemplazamos $\text{NAE}(a, b, c, d)$ por las cláusulas $\text{NAE}(a, b, w)$ y $\text{NAE}(\neg w, c, d)$ en ϕ'.

La demostración se basa en que cuatro valores booleanos a, b, c, d no son todos iguales si y solo si existe un valor w tal que $\text{NAE}(a, b, w)$ y $\text{NAE}(\neg w, c, d)$ son verdaderos.

10.2.2. Conclusión

Como **NAE 4-SAT** es NP-Completo, y hemos demostrado una reducción polinomial hacia **NAE 3-SAT**, concluimos que **NAE 3-SAT** es NP-Hard.

10.3. NP-Completitud

Como **NAE 3-SAT** pertenece a NP y es NP-Hard, es NP-Completo.

11. Máximo Corte

Definición: Sea $G = (V, E)$ un grafo con aristas ponderadas. Un corte es una división de los vértices en dos conjuntos T y $V - T$. El costo de un corte es la suma de los pesos de las aristas que van de T a $V - T$. El problema **Máximo Corte** consiste en determinar si existe un corte con costo al menos c en un grafo dado.

11.1. Pertenencia a NP

El problema **Máximo Corte** pertenece a NP porque, dado un corte T, podemos verificar en tiempo polinomial si su costo es al menos c sumando los pesos de las aristas de cruce del corte.

11.2. NP-Hardness

Demostraremos que **Máximo Corte** es NP-Hard mediante una reducción desde **NAE 3-SAT**.

11.2.1. Demostración

Dada una instancia de **NAE 3-SAT** ϕ, construiremos una instancia equivalente de **Máximo Corte** (G, c) como sigue:

- Por cada variable x_i de ϕ, agregamos dos vértices en G etiquetados como x_i y $\neg x_i$, conectados por una arista con capacidad $M = 10 \cdot m$, donde m es el número de cláusulas en ϕ.
- Por cada cláusula C en ϕ, agregamos un triángulo entre los vértices correspondientes a los términos en C, con aristas de capacidad 1.

Demostremos que G contiene un corte con capacidad al menos $nM + 2m$ (donde n es el número de variables) si y solo si ϕ es satisfacible.

- ($\Leftarrow$) Si ϕ es satisfactible, entonces existe un corte en G con capacidad al menos $nM + 2m$: Supongamos que ϕ es satisfacible y consideremos cualquier asignación satisfacible. Esta asignación corresponde a un corte en G (Un lado del corte consiste en todos los vértices etiquetados por términos que evalúan 1 en la asignación. El otro lado consiste en todos los vértices etiquetados por términos que evalúan 0). Este corte tiene capacidad el menos $nM + 2m$ porque:
  - Como exactamente uno de los términos x_i y $\neg x_i$ evalúa 1 en la asignación, todos los arcos entre x_i y $\neg x_i$ cruzan el corte (contribuyendo $n \cdot M$).
  - Como la asignación satisface ϕ, exactamente dos aristas en cada triángulo cruzan el corte (contribuyendo $2 \cdot m$).
- ($\Rightarrow$) Si existe un corte con capacidad al menos $n \cdot M + 2 \cdot m$, entonces corresponde a una asignación satisfactoria para ϕ: Supongamos que G contiene un corte con capacidad el menos $nM + 2m$. Probemos que:

- Todas las aristas entre x_i y $\neg x_i$ cruzan el corte, garantizando una asignación consistente. Esto ocurre porque cualquier corte que no contenga una de estas aristas tendrá como mucho capacidad $(n-1)M + 3m = nM + 3m - 10m$, que es estrictamente menor que $nM + 2m$.
- Exactamente dos aristas en cada triángulo cruzan el corte, satisfaciendo cada cláusula de ϕ. Esto ocurre porque ningún corte puede separar las tres aristas de un triángulo y por lo tanto si un corte separa menos de dos aristas en uno de los triángulos de cláusula, entonces su capacidad es estrictamente menor que $nM + 2m$.

Como todas las aristas entre x_i y $\neg x_i$ cruzan el corte, este corresponde a una asignación para ϕ. Además, como exactamente dos aristas en cada triángulo cruzan el corte, la asignación correspondiente es satisfactible.

11.2.2. Conclusión

Como **NAE 3-SAT** es NP-Completo, y hemos demostrado una reducción polinomial hacia **Máximo Corte**, concluimos que **Máximo Corte** es NP-Hard.

11.3. NP-Completitud

Como **Máximo Corte** pertenece a NP y es NP-Completo, es NP-Completo.