Problem Set de Diseño y Análisis de Algortimos

Ana Paula González Muñoz
Dennis Daniel González Durán
C412

1. Set cover

Definición del problema

Dado un conjunto X y una colección S de subconjuntos de X, el problema consiste en determinar si existe un subcolector $S' \subseteq$ S tal que cada elemento de X aparezca exactamente una vez en los subconjuntos de S'.

Solución

Para demostrar que el problema es NP-completo primero desmostremos que es NP. El problema pertenece al conjunto de problemas NP porque, dado una solución se puede comprobar en tiempo polinomial que cada elemento del universo pertenece solo a un subconjunto de los seleccionados en la cobertura exacta

Para probar que **set cover** es NP-completo, reducimos el Problema de Satisfacibilidad (SAT) a él. Dado un conjunto $F = \{C_1, ..., C_l\}$ de l cláusulas construidas a partir de n variables proposicionales $x_1, ..., x_n$, debemos construir en tiempo polinómico una instancia $\tau(F) = (U, \gamma)$ de set cover tal que F es satisfacible si y solo si $\tau(F)$ tiene una solución.

Para construir (U, γ) a partir de $F = \{C_1, ..., C_l\}$ procedemos de la siguiente manera:

> Digamos que $C_j = (L_{j1} \vee \vee L_{jmj})$ es la j-ésima cláusula en F, donde L_{jk} denota el k-ésimo literal en C_j y $m_j \geq 1$. El universo de $\tau(F)$ es el conjunto $U = \{x_i \mid 1 \leq i \leq n\} \cup \{C_j \mid 1 \leq j \leq l\} \cup \{p_{jk} \mid 1 \leq j \leq l, 1 \leq k \leq m_j\}$ donde en el tercer conjunto p_{jk} corresponde al k-ésimo literal en C_j
>
> En γ se incluyen los siguientes subconjuntos:
>
> - Hay un conjunto $\{p_{jk}\}$ para cada p_{jk}
> - Para cada variable booleana x_i:
>   1. $T_{i,T} = \{x_i\} \cup \{p_{jk} \mid L_{jk} = \overline{x_i}\}$ que contiene x_i y todas las ocurrencias negativas de x_i
>   2. $T_{i,F} = \{x_i\} \cup \{p_{jk} \mid L_{jk} = x_i\}$ que contiene x_i y todas las ocurrencias positivas de x_i
> - Para cada cláusula C_j, los conjuntos m_j $\{C_j, p_{jk}\}$ están en γ

Queda por demostrar que F es satisfacible si $\tau(F)$ tiene una solución. Afirmamos que si v es una asignación de verdad que satisface F, entonces podemos hacer una cobertura exacta C de la siguiente manera: para cada x_i, ponemos el subconjunto $T_{i,T}$ en C si $v(x_i) = T$, de lo contrario, ponemos el subconjunto $T_{i,F}$ en C si $v(x_i) = F$. Además, para cada cláusula C_j, ponemos algún subconjunto $\{C_j, p_{jk}\}$ en C para un L_{jk} literal que se hace verdadero por v. Por la construcción de $T_{i,T}$ y $T_{i,F}$, este p_{jk} no está en ningún conjunto en C seleccionado hasta ahora. Puesto que por hipótesis F es satisfacible, tal literal existe para cada cláusula. Habiendo cubierto todos x_i y C_j, ponemos un conjunto $\{p_jk\}$ en C para cada p_{jk} restante que aún no ha sido cubierto por los conjuntos que ya están en C

Si C es una cobertura exacta de $\tau(F)$, definimos una asignación de verdad de la siguiente manera: para cada x_i, si $T_{i,T}$ está en C, entonces establecemos$v(x_i) = T$, de lo contrario, si $T_{i,F}$ está en C, entonces establecemos $v(x_i) = F$.

Demostremos que esta asignación es válida. Para ello necesitamos demostrar que:

1. la asignación es consistente en el valor de cada variable.
2. todas las cláusulas en el problema de satisfacibilidad se activan.

Demostremos que la asignación es consistente con el valor de cada variable:

> Dado que, para cada i, al menos uno de los conjuntos $T_{i,T}$ o $T_{i,F}$ debe pertenecer a la cobertura exacta (ya que son los únicos que contienen al elemento x_i del universo), encontramos que, para cada variable, exactamente uno de los conjuntos $T_{i,T}$ o $T_{i,F}$ estará en la cobertura exacta. Además, este conjunto será único para cada x_i, puesto que $x_i \subseteq T_{i,T}$ y $x_i \subseteq T_{i,F}$, lo que implica que ambos conjuntos no pueden formar parte simultáneamente de la cobertura exacta.
>
> Por lo tanto, en la asignación de variables, cada una de ellas tendrá un valor único asignado.

Demostremos que todas las cláusulas en el problema de satisfacibilidad se activan:

> Realizaremos la demostración basándonos en la construcción de la entrada del problema de set cover y su relación con el problema de SAT.
>
> Recordemos que cada p_{jk} representa la k-ésima variable en la j-ésima cláusula de la fórmula en SAT. Asimismo, por la definición de los conjuntos $T_{i,T}$ y $T_{i,F}$ en la construcción, tenemos lo siguiente:
>
> 1. Propiedad de los conjuntos $T_{i,T}$ y $T_{i,F}$:
>   - El conjunto $T_{i,T}$ contiene todos los elementos p_{jk} que no pueden activarse si la variable x_i toma el valor verdadero ($x_i =$ True).
>   - El conjunto $T_{i,F}$ contiene los elementos p_{jk} que no pueden activarse si x_i toma el valor falso ($x_i =$ False).
>
>   Dado esto, al seleccionar los conjuntos $T_{i,T}$ o $T_{i,F}$ correspondientes a cada variable x_i, los p_{jk} que permanecen disponibles representan, para cada cláusula, las posibles variables que pueden activarla, en función de sus valores.
> 2. Por construcción, los conjuntos $\{C_j, p_{jk}\}$ están diseñados para garantizar que, al ser seleccionados en la solución de la cobertura exacta, se elige un único p_{jk} para cada cláusula C_j. Este p_{jk} corresponde a una variable x_i que puede satisfacer dicha cláusula según su valor asignado.
> 3. La cobertura exacta selecciona todos los conjuntos necesarios para cubrir los elementos del universo (que incluye todos los C_j), cada cláusula tiene asociado un p_{jk} válido. Esto implica que, para cada cláusula, existe al menos una variable que la satisface. Por lo tanto, todas las cláusulas de la fórmula en SAT quedan activadas.
>
> De esta forma, hemos demostrado que, si existe una solución para la cobertura exacta construida según nuestra representación, todas las cláusulas de la fórmula en SAT serán satisfechas.

2. Clique

Definición del problema

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. Hallar el clique de mayor tamaño en un grafo.

Solución

Dado un grafo G con n vértices, un parámetro k, y un conjunto W de k vértices, verificar que cada par de vértices en W está conectado en G puede verificarse en tiempo polinomial, lo que implica que Clique pertenece a NP.

Para demostrar que clique pertenece a NP-Hard se realiza mediante una reducción de 3-SAT a Clique. Consideremos una fórmula F de 3-SAT definida sobre n variables y m cláusulas. Construimos el grafo G de la fórmula F de la siguiente manera:

1. Para cada literal en la fórmula, generamos un vértice y etiquetamos dicho vértice con el literal correspondiente. Cada cláusula contiene tres de estos vértices.
2. Conectamos dos vértices en el grafo si:
   (a) Pertenecen a cláusulas diferentes, y
   (b) No son literales negados uno del otro.

Probaremos que F es satisfacible si y solo si existe un clique de tamaño m en G.

Parte 1: Si F es satisfacible, entonces existe un clique de tamaño m en G. Sean $x_1, \ldots, x_n$ las variables en F, y sea $v_1, \ldots, v_n \in \{0, 1\}$ una asignación que satisface F. Dado que F es verdadera, al menos un literal en cada cláusula se evalúa como verdadero. Seleccionemos un vértice de G correspondiente a un literal verdadero de cada cláusula. Sea W el conjunto resultante de vértices seleccionados.

Es claro que W forma un clique en G, ya que:

- Cada vértice en W proviene de una cláusula diferente, por lo que los vértices están conectados entre sí.
- Los vértices seleccionados no son literales negados uno del otro, por la construcción del grafo.

Por tanto, W forma un clique de tamaño m en G.

Parte 2: Si existe un clique de tamaño m en G, entonces F es satisfacible. Sea U un conjunto de m vértices que forman un clique en G. Construimos una asignación para F como sigue:

1. Asignamos $x_i \leftarrow$ TRUE si existe un vértice en U etiquetado con x_i.
2. Asignamos $x_i \leftarrow$ FALSE si existe un vértice en U etiquetado con $\neg x_i$.

Supongamos, por reducción al absurdo, que existe una variable x_i tal que tanto x_i como $\neg x_i$ están en U. Por construcción, los vértices correspondientes a x_i y $\neg x_i$ no están conectados en G, lo que contradice que U es un clique. Por tanto, la asignación es válida.

Además, dado que U contiene un vértice de cada cláusula, al menos un literal en cada cláusula es verdadero bajo esta asignación. Por lo tanto, F es satisfacible.

Hemos demostrado que, dado un algoritmo polinomial para resolver Clique, podríamos resolver 3-SAT en tiempo polinomial. Esto implica que Clique es NP-Hard.

Dado que Clique es NP-Hard y pertenece a NP, concluimos que Clique es NP-Completo.

3. Número crómatico

Definición del problema

El número cromático de un grafo es el número mínimo de colores necesarios para colorear los vértices del grafo de manera que dos vértices adyacentes no compartan el mismo color. Hallar el número cromático en un grafo.

Problema de decisión asociado

Dado un grafo $G = (V, E)$ y un entero k, saber si el grafo es k-coloreable

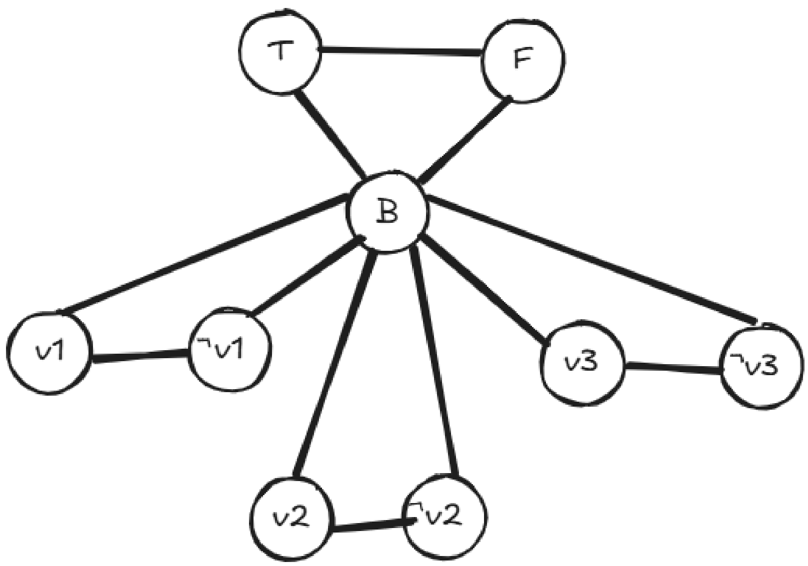
Solución

El problema pertenece a NP ya que dado coloración se puede verificar en tiempo polinómico que se utilizan como máximo k colores, y que ningún par de nodos unidos por una arista recibe el mismo color.

Para demostrar que el problema pertenece a NP-Hard haremos una reducción a 3-SAT. Dada una instancia arbitraria de 3-SAT, con variables $x_1, \ldots, x_n$ y cláusulas $C_1, \ldots, C_k$, vamos a construir un grafo G de tal manera que este es 3-coloreable si y solo si la instancia de 3-SAT es satisfacible.

Para la construcción definimos nodos v_i y $\overline{v_i}$, correspondientes a cada variable x_i y su negación $\overline{x_i}$. También definimos tres "nodos especiales": T, F y B, a los que llamamos Verdadero, Falso y Base.

Para comenzar, unimos cada par de nodos $v_i, \overline{v_i}$ entre sí con una arista, y unimos ambos nodos a Base. (Esto forma un triángulo en $v_i, \overline{v_i}$ y Base para cada i). También unimos T, F y B en un triángulo. El grafo simple G que hemos definido hasta ahora tiene algunas propiedades útiles:

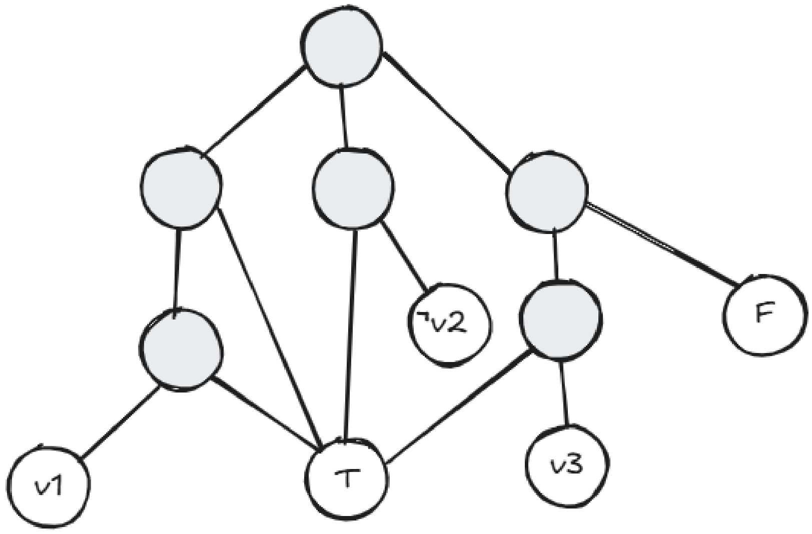


- En cualquier 3-coloración de G, los nodos v_i y $\overline{v_i}$ deben obtener colores diferentes, y ambos deben ser diferentes de Base.
- En cualquier 3-coloración de G, los nodos T, F y B deben recibir los tres colores en alguna permutación. Por lo tanto, podemos referirnos a los tres colores como el color Verdadero, el color Falso y el color Base, según cuál de estos tres nodos recibe cada color. En particular, esto significa que para cada i, uno de v_i o $\overline{v_i}$ recibe el color Verdadero, y el otro recibe el color Falso. Para el resto de la construcción, consideraremos que la variable x_i está asignada a 1 en la instancia dada de 3-SAT si y solo si el nodo v_i recibe el color Verdadero.

En resumen, ahora tenemos un grafo G en el que cualquier 3-coloración determina implícitamente una asignación de verdad para las variables en la instancia de 3-SAT. Ahora necesitamos extender G de tal manera que solo las asignaciones satisfactorias puedan extenderse a 3-coloraciones del grafo completo.

Como en otras reducciones de 3-SAT, consideremos una cláusula como $x_1 \vee \overline{x_2} \vee x_3$. En el lenguaje de las 3-coloraciones de G, esto dice: "Al menos uno de los nodos v_1, v_2 o v_3 debe recibir el color Verdadero". Entonces, lo que necesitamos es un pequeño subgrafo que podamos çonectar.ª G, de modo que cualquier 3-coloración que se extienda a este subgrafo deba tener la propiedad de asignar el color Verdadero a al menos uno de v_1, v_2 o v_3. El subgrafo que se usa es el siguiente:

Este subgrafo de seis nodos se "adhiere" al resto de G en cinco nodos existentes: T, F, y aquellos correspondientes a los tres términos en la cláusula que estamos tratando de representar (en este caso, v_1, v_2 y v_3). Supongamos ahora que en alguna 3-coloración de G, los tres v_1, v_2 y v_3 son asignados al color Falso. Entonces, los dos nodos sombreados inferiores en el subgrafo deben recibir el color Base, los tres nodos sombreados superiores deben recibir, respectivamente, los colores Falso, Base y Verdadero,



y, por lo tanto, no hay un color que pueda asignarse al nodo sombreado superior. En otras palabras, una 3-coloración en la que ninguno de v_1, v_2 o v_3 esté asignado al color Verdadero no puede extenderse a una 3-coloración de este subgrafo.

Finalmente, y de manera inversa, al revisar los casos a mano se verifica que siempre que uno de v_1, v_2 o v_3 esté asignado al color Verdadero, el subgrafo completo puede ser 3-coloreado.

A partir de esto, podemos completar la construcción: comenzamos con el grafo G definido anteriormente, y para cada cláusula en la instancia de 3-SAT, conectamos un subgrafo de seis nodos como se muestra en la figura. Llamemos G' al grafo resultante.

Ahora afirmamos que la instancia dada de 3-SAT es satisfacible si y solo si G' tiene una 3-coloración. Primero, supongamos que hay una asignación satisfactoria para la instancia de 3-SAT. Definimos una coloración de G' coloreando inicialmente Base, True y False arbitrariamente con los tres colores, luego, para cada i, asignamos v_i el color Verdadero si $x_i = 1$ y el color Falso si $x_i = 0$. Luego asignamos a $\overline{v_i}$ el único color disponible. Finalmente, como se argumentó anteriormente, ahora es posible extender esta 3-coloración a cada subgrafo de seis nodos de la cláusula, resultando en una 3-coloración de todo G'.

Por el contrario, supongamos que G' tiene una 3-coloración. En esta coloración, cada nodo v_i está asignado ya sea al color Verdadero o al color Falso; asignamos la variable x_i en consecuencia. Ahora afirmamos que en cada cláusula de la instancia de 3-SAT, al menos uno de los términos en la cláusula tiene el valor de verdad 1. Porque si no, entonces los tres nodos correspondientes tienen el color Falso en la 3-coloración de G' y, como hemos visto anteriormente, no hay una 3-coloración del subgrafo correspondiente consistente con esto, lo cual es una contradicción.

Cuando $k > 3$, es muy fácil reducir el problema de **3-Coloración** al de k-Coloración. Esencialmente, todo lo que hacemos es tomar una instancia de **3-Coloración**, representada por un grafo G, agregar $k - 3$ nuevos nodos y unir estos nuevos nodos entre sí y con cada nodo en G. El grafo resultante es k-colorable si y solo si el grafo original G es **3-colorable**. Por lo tanto, el problema k-Coloración para cualquier $k > 3$ también es NP-completo.

Como el problema de decisión asociado a **Número crómatico** pertenece a NP-Completo, el mismo (que es de optimización) pertenece a NP-Hard

4. Retroalimentación de vértices

Definición del problema

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 un grafo acíclico o un bosque, si es no dirigido). El objetivo del problema es encontrar el conjunto de retroalimentación de vértices de tamaño mínimo.

Solución

Vamos a demostrar que el problema es NP-Hard. Para ello, primero demostraremos que el problema de retroalimentación de vértices de tamaño k es NP-Completo:

El problema pertenece a NP porque dada una instancia de retroalimentación de vértices y un conjunto solución D, basta verificar que el tamaño de D es menor o igual que k y que al eliminar esos vértices del grafo, este es acíclico. Esto se puede comprobar realizando un recorrido por el grafo.

Para demostrar que es NP-Hard, intentaremos reducir el problema de **Vertex Cover** (el cual sabemos que es NP-Completo) a Retroalimentación de Vértices. La reducción se realizará de la siguiente forma: dada una instancia de Vertex Cover con $G = (V, E)$, crearemos un grafo dirigido $G' = (V', E')$ con $V' = V$ y, por cada arista $(a, b) \in E$, crearemos las aristas (a, b) y (b, a) en E', generando un ciclo en G'.

Para Vertex Cover, necesitamos un conjunto mínimo tal que cada arista esté cubierta. Para Retroalimentación de Vértices, necesitamos un conjunto de vértices mínimo tal que, si lo eliminamos del grafo, este resulta acíclico.

Dado que cada arista en G de Vertex Cover se transforma en dos aristas dirigidas en G' formando un ciclo, el conjunto seleccionado para satisfacer Retroalimentación de Vértices, digamos F, debe incluir al menos uno de los dos vértices para romper ese ciclo. Si tenemos entonces F en G', F debe cubrir cada arista en G. Esto significa que, para cada ciclo dirigido correspondiente en G', F debe contener al menos uno de los vértices de ese ciclo para romperlo.

Al eliminar F en G', todos los ciclos dirigidos se eliminan, lo que hace que G' sea acíclico. Por tanto, si en Retroalimentación de Vértices hay una solución de tamaño k, entonces también la hay en Vertex Cover.

Habiendo demostrado que este problema es NP-Completo, podemos reducirlo al problema de hallar el conjunto mínimo que satisfaga Retroalimentación de Vértices. Si el mínimo es menor que k, entonces hay una solución de tamaño k, y si no lo es, no la hay. Por tanto, el problema de hallar el conjunto mínimo que satisfaga Retroalimentación de Vértices es NP-Hard.

5. Retroalimentación de arcos

Definición del problema

Dado un grafo$G = (V, E)$, 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 un grafo acíclico o un bosque, si es no dirigido). El objetivo del problema es encontrar el conjunto de retroalimentación de arcos de tamaño mínimo.

Solución

Al igual que en el problema anterior, demostraremos primero que el problema de encontrar un conjunto de arcos de longitud k tal que al quitarlos no queden ciclos en G es NP-Completo. Para ello, intentaremos reducir el problema de **Retroalimentación de Vértices** a este en tiempo polinomial.

Dada una instancia (G, k) de Retroalimentación de Vértices con $G = (V, E)$, crearemos una instancia (G', k') de Retroalimentación de Arcos de la siguiente forma:

- Por cada nodo $v \in V$ creamos dos nodos v_1 y v_2 en G' y el arco (v_1, v_2). A estos arcos los llamaremos *arcos internos.*
- Por cada arco $(u, v) \in E$ crearemos en E' el arco (u_2, v_1). Estos serán los *arcos externos.*

Cada vértice en G se duplica en G', creando dos copias de vértices. Las aristas del grafo original se dividen en *aristas internas* (entre las copias de un mismo vértice) y *aristas externas* (entre copias de vértices diferentes).

Al resolver Retroalimentación de Arcos en G', buscamos un conjunto de aristas que rompa todos los ciclos en G'. La clave está en que todos los ciclos que involucran aristas externas también pasan por aristas internas, por lo que eliminar solo aristas internas (que corresponden a vértices en G) es suficiente para romper los ciclos.

Por lo tanto, cualquier conjunto de aristas internas que forme una solución para Retroalimentación de arcos en G' corresponde a un conjunto de vértices que forma una solución para Retroalimentación de Vértices en G. De esta forma, resolver Retraolimentación de arcos en G' es equivalente a resolver Retraolimentación de vértices en G, ya que los ciclos que se rompen en G' equivalen a los ciclos que se rompen en G.

Análogamente al problema anterior, demostramos que hallar el conjunto mínimo es NP-Hard. En este caso, probar que encontrar el conjunto mínimo de aristas es NP-Hard también demuestra que resolver el problema de *Retroalimentación de Arcos* es NP-Hard.

6. 3D matching

Definición del problema

El problema se basa en encontrar un emparejamiento dentro de un conjunto tridimensional. Supongamos que tienes tres conjuntos disjuntos: X, Y, y Z, cada uno de tamaño n. También tienes un conjunto T de ternas de la forma (x, y, z), donde $x \in X, y \in Y$, y $z \in Z$. El objetivo es determinar si existe un subconjunto de T de tamaño n (es decir, n ternas) tal que cada elemento deX, Y, y Z aparezca exactamente una vez en las ternas seleccionadas.

Solución

Paso 1: Mostrar que 3D Matching pertenece a NP
Dado una instancia del problema 3D Matching con:

- X, Y, Z: conjuntos disjuntos, cada uno con n elementos,
- T: un conjunto de triplas (x, y, z), con $x \in X$, $y \in Y$, $z \in Z$,
- $C \subseteq T$: subconjunto de T.

Para verificar que C es una solución válida:

- Comprobamos que $|C| = n$.
- Verificamos que cada elemento de X, Y y Z aparece exactamente una vez en las triplas de C.

Ambas verificaciones pueden realizarse en tiempo polinomial respecto al tamaño de la entrada. Por lo tanto, 3D Matching pertenece a NP.

Paso 2: Mostrar que 3D Matching es NP-Hard
Reduciremos el problema 3SAT a 3D Matching en tiempo polinomial. Consideremos una instancia de 3SAT con:

- n variables: $\{x_1, x_2, \ldots, x_n\}$,
- k cláusulas: $\{c_1, c_2, \ldots, c_k\}$, cada una con exactamente tres literales.

Construcción de la instancia de 3D Matching:
Para cada variable x_i creamos los siguientes conjuntos:

- $A_i = \{a_{i1}, a_{i2}, \ldots, a_{i2k}\}$, con $2k$ elementos.
- $B_i = \{b_{i1}, b_{i2}, \ldots, b_{i2k}\}$, con $2k$ elementos.
- $T_i = \{(a_{ij}, a_{ij+1}, b_{ij}) \mid 1 \leq j \leq 2k\}$.

Cada tripla en T_i conecta elementos de A_i y B_i. Notemos que para cubrir todos los elementos de A_i y B_i, debemos seleccionar todas las triplas correspondientes a índices j pares o todas las de índices j impares, pero no una combinación de ambos pues se repetiría algún elemento de A_i.

Para cada cláusula c_j, creamos:

- Dos nuevos elementos: p_j y p_{j1}.

- Una tripla para cada literal de la cláusula c_j:
  - Si el literal es x_i, agregamos la tripla $(p_j, p_{j1}, b_{i,2j-1})$.
  - Si el literal es $\neg x_i$, agregamos la tripla $(p_j, p_{j1}, b_{i,2j})$.

  En total, para cada cláusula se crean tres triplas.

Interpretación: Si en la solución de 3D MATCHING se seleccionan las triplas de T_i correspondientes a índices j pares, entonces la variable x_i está asignada a `true` en 3SAT. Si se seleccionan las triplas con índices j impares, entonces x_i está asignada a `false`.
Las b_{ij} que no sean seleccionadas en las triplas $(a_{ij}, a_{ij+1}, b_{ij})$ pueden ser seleccionadas en las triplas (p_j, p_{j1}, b_{ij}) o en otras que se definirán más adelante.

Cada cláusula tiene asociadas tres triplas, pero solo podemos seleccionar una de ellas para activar la cláusula o para indicar que esta se satisface. Notemos que si una cláusula c_l contiene una variable x_p no negada, y esta variable está asignada a `true`, esto implica que se seleccionaron todas las triplas de la forma $(a_{pj}, a_{pj+1}, b_{pj})$ con j par en los b_{pj}. En este caso, es posible seleccionar la tripla $(p_l, p_{l1}, b_{p,2l-1})$ para activar la cláusula c_l.

En total, hay $2nk$ elementos b_{ij}, ya que hay n variables y para cada una se crean $2k$ elementos b_{ij}. Si existe una solución para el problema, notemos que las triplas asociadas a las cláusulas cubren k de estos b_{ij} (dado que cada cláusula tiene tres triplas asociadas, pero solo podemos seleccionar una). Además, las triplas asociadas a las variables cubren nk de los b_{ij}. Por lo tanto, faltaría por cubrir $(n-1)k$ elementos b_{ij}.

Para garantizar que todos los elementos se cubren, creamos:

- Nuevos elementos q_m y q_{m1} para $1 \leq m \leq (n-1)k$.
- Triplas adicionales (q_m, q_{m1}, b) que conectan estos elementos con todos los b_{ij}.

Finalmente, construimos los conjuntos disjuntos X, Y, Z y el conjunto T de triplas:

- $X = \{a_{ij} \mid j \text{ par}\} \cup \{p_j\} \cup \{q_m\}$.
- $Y = \{a_{ij} \mid j \text{ impar}\} \cup \{p_{j1}\} \cup \{q_{m1}\}$.
- $Z = \{b_{ij}\}$.
- T contiene todas las triplas definidas anteriormente.

Si existe una solución para 3D MATCHING, podemos construir una asignación para 3SAT seleccionando las triplas según se explicó anteriormente. Como la reducción es polinomial y 3SAT es NP-Hard, 3D MATCHING también lo es.

3D MATCHING pertenece a NP y es NP-Hard, por lo que es NP-completo.