UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Modelos Avanzados de Computación
Curso Académico 2025/2026

NP-Completitud: Problema del cubrimiento por cliques

Nombre: Manuel González Santana

Granada, a 2 de julio de 2026

Índice

1. Consideraciones Generales

1.1. Enunciado del problema

Dado un grafo no dirigido $G = (V, E)$ y un entero K, existe una partición de V en K conjuntos $V_1, \ldots, V_K$ de tal manera que cada V_i es completo: para cada $a, b \in V_i$, $\{a, b\} \in E$.

En lo que procede se va a emplear la notación K-Cubrimiento.

1.2. Proceso de la demostración

La idea de esta demostración es seguir el proceso que realizó Karp en su paper original, pero añadiendo un paso intermedio para simplificar y facilitar la comprensión de la misma.

En la demostración original se reduce SAT a K-coloring [1] y el problema K-Cubrimiento consiste en resolver el problema K-coloring en el grafo complementario.

En este trabajo vamos a emplear otra aproximación que consiste en las siguientes reducciones:

1. 3-SAT exacto a 3-coloring
2. 3-coloring a K-coloring con $K > 3$
3. K-coloring a K-cubrimiento

Para la validez de esta reducción se va a demostrar que todos estos problemas pertenecen a la clase NP (recalcando que el 3-coloring es un caso particular de K-coloring) y las sucesivas reducciones probarán que son NP-completos.

1.3. Proceso para las reducciones

Para las reducciones se empleará el proceso explicado en la teoría, el cual es el siguiente: Si tenemos dos problemas $P_1(x)$ y $P_2(y)$, para demostrar que $P_1(x) \propto P_2(y)$ hay que:

1. Dar un algoritmo REDU que transforme cada entrada, x, de P_1 en una entrada, $y = \text{REDU}(x)$, del problema P_2.
2. Comprobar que dicho algoritmo es logarítmico en espacio.
3. Comprobar que si $P_1(x)$ tiene solución positiva, entonces $P_2(y)$ también la tiene.
4. Comprobar que si $P_2(y)$ tiene solución positiva, entonces $P_1(x)$ también la tiene, o comprobar que si $P_1(x)$ tiene solución negativa, entonces $P_2(y)$ también la tiene negativa.

1.4. Comprobación de que los algoritmos son espacio logarítmicos

En teoría también se ha visto que, para contabilizar el espacio que emplea un algoritmo, se siguen las siguientes reglas:

[1] Este problema nos pregunta si existe una partición de los nodos del grafo en K conjuntos tal que $\forall a, b \in V_i$, $\{a, b\} \notin E$

- Se cuentan las casillas que se usan (aquellas en las que se escribe o sobre las que se pasa).
- Si nunca se escribe sobre la cinta de entrada, entonces las casillas de esta cinta no se cuentan.
- Si las casillas de la cinta de salida se escriben de izquierda a derecha, sin volver nunca hacia atrás, tampoco se cuentan.

2. Demostración de que los problemas empleados son NP

Para demostrar que un problema es NP se ha empleado la caracterización de que existe una máquina de Turing no determinista que lo resuelve en tiempo polinómico.

Por lo visto en teoría, la representación que empleemos del grafo no es relevante para la complejidad computacional, para las siguientes máquinas de Turing se ha empleado la representación lista de vértices mas lista de pares de aristas.

2.1. K-coloring es NP

Partimos de un grafo $G = (V, E)$ y un entero K. Se denota:

- Sea n el número de vértices ($|V|$).
- Sea m el número de aristas ($|E|$).

Es claro que la entrada que recibe la máquina de Turing M depende de forma polinómica de los enteros n, m y $\log_2 K$, por lo que podemos medir la complejidad del algoritmo en base a estos valores.

Es importante tener en cuenta que si se presenta una dependencia en función de K esta dependencia ha de ser logarítmica, ya que para reprensentar este valor se necesitan $\lceil \log_2 K \rceil$ celdas en la máquina de Turing.

Denotando por $T(n, m, \log_2 K)$ al número de operaciones que realiza la máquina de Turing en función de los tamaños de entrada, se busca acotar esta función por una polinómica en función de los valores de entrada.

1. **Inicialización:** Se lee la cinta de entrada para determinar el número total de nodos n.
2. **Fase de Generación (No determinista):** La máquina escribe de forma no determinista en una cinta una coloración para el grafo, asignando a cada uno de los n nodos un valor del conjunto $\{1, 2, \ldots, K\}$.
3. **Fase de Verificación (Determinista):** Se recorre secuencialmente la lista de las m aristas del grafo. Para cada arista $(u, v) \in E$:
   - La máquina busca en la cinta de colores el valor asignado a u y a v.
   - Si $\text{color}(u) = \text{color}(v)$, la máquina **rechaza**.
4. **Aceptación:** Si se procesan las m aristas sin detectar ningún conflicto de color, la máquina **acepta**.

Evaluamos el tiempo de ejecución de la Máquina de Turing:

- **Contar nodos:** Requiere leer la entrada, tomando un tiempo de $O(n)$.
- **Escribir la coloración:** Cada color requiere $\log_2 K$ celdas en la cinta. Escribir la asignación para los n nodos toma un tiempo de $O(n \cdot \log_2 K)$.
- **Verificar aristas:** Para cada una de las m aristas, buscar secuencialmente los colores de sus dos extremos en la cinta toma $O(n \cdot \log_2 K)$ pasos. Al repetir esto para las m aristas, el coste de esta fase es de $O(n \cdot m \cdot \log_2 K)$.

El tiempo total de ejecución está dominado por la fase de verificación:

$$T(n, m, \log_2 K) = O(n \cdot m \cdot \log_2 K)$$

Al ser $T(n, m, \log_2 K)$ una función polinómica respecto al tamaño de la entrada, queda demostrado que el problema pertenece a la clase **NP**.

2.2. K-cubrimiento es NP

Usando la misma notación que en el algoritmo previo, el diseño de la máquina de Turing es análogo.

1. **Inicialización:** Se lee la cinta de entrada para determinar el número total de nodos n.
2. **Fase de Generación (No determinista):** La máquina escribe de forma no determinista en la cinta una asignación de partición. A cada uno de los n nodos se le asigna un identificador de conjunto de la lista $\{1, 2, \ldots, K\}$, indicando a qué subgrafo V_i pertenece.
3. **Fase de Verificación (Determinista):** La máquina debe comprobar que cada conjunto V_i sea completo. Para ello, analiza todos los pares posibles de nodos (a, b) con $a \neq b$:
   - Busca en la cinta de asignaciones los conjuntos de a y b.
   - Si pertenecen al mismo conjunto ($a, b \in V_i$), la máquina escanea la lista de aristas E para verificar si la arista $\{a, b\}$ existe.
   - Si la arista **no** existe en E, la máquina **rechaza** inmediatamente.
4. **Aceptación:** Si se evalúan todos los pares de nodos y no se detecta ninguna arista faltante dentro de los mismos conjuntos, la máquina **acepta**.

Evaluamos el tiempo de ejecución de la Máquina de Turing:

- **Contar nodos:** Requiere una lectura inicial de la entrada, tomando un tiempo de $O(n)$.
- **Escribir la partición:** Asignar a cada uno de los n nodos un identificador de tamaño $\log_2 K$ requiere un tiempo de $O(n \cdot \log_2 K)$.
- **Verificar subgrafos completos:**
  - Existen $\frac{n(n-1)}{2} = O(n^2)$ pares posibles de nodos en el grafo.
  - Para cada par que comparta el mismo conjunto, buscar si la arista existe en la lista de entrada E (que contiene m aristas) requiere recorrer secuencialmente dicha lista, lo que toma $O(m \cdot \log_2 n)$ pasos.
  - Por lo tanto, el coste de revisar todas las combinaciones es de $O(n^2 \cdot m \cdot \log_2 n)$.

El tiempo total de ejecución está dominado por la fase de verificación:

$$T(n, m, \log_2 K) = O(n^2 \cdot m \cdot \log_2 n) + O(n \cdot \log_2 K)$$

Al ser $T(n, m, \log_2 K)$ una función polinómica respecto al tamaño del grafo, queda demostrado que este problema también pertenece a la clase **NP**.

3. K-cubrimiento es NP-Completo

3.1. Reducción de 3-SAT exacto a 3-coloring

3.1.1. Algoritmo de reducción

Partimos de una instancia del problema 3-SAT exacto compuesta por un conjunto de variables $X = \{x_1, \ldots, x_n\}$ y un conjunto de cláusulas $C = \{C_1, \ldots, C_k\}$, donde cada cláusula contiene exactamente 3 literales.

Para realizar esta reducción, interpretaremos los tres colores disponibles en el problema de 3-coloring como los valores de una lógica ternaria. Por ello, definimos el conjunto de colores como $\{T, F, B\}$, que representan los estados **Verdadero** (*True*), **Falso** (*False*) y **Base**, respectivamente.

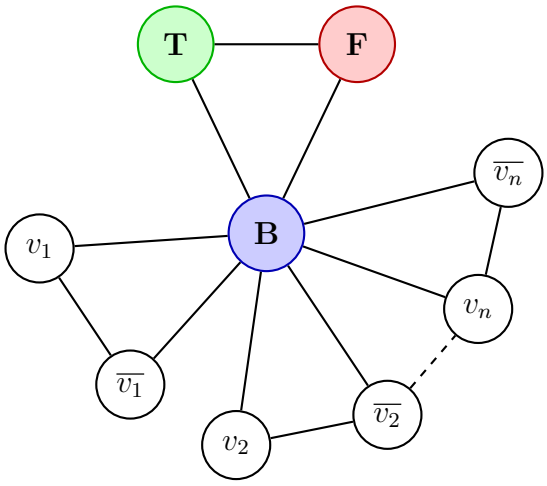
El proceso de construcción del grafo intermediario se realiza mediante los siguientes pasos:

1. **Estructura de control inicial:** Se crea un triángulo central con tres nodos etiquetados como T, F y B, conectados todos entre sí mediante las aristas (T, F), (T, B) y (F, B). Esto fuerza a que estos tres nodos de control tengan obligatoriamente colores distintos.

2. **Gadget de variables:** Por cada variable $x_i \in X$, añadimos un par de nodos literales v_i y $\overline{v_i}$ a la lista de vértices. Asimismo, añadimos las aristas (B, v_i), $(B, \overline{v_i})$ y $(v_i, \overline{v_i})$. Debido a la conexión con el nodo Base (B), este diseño fuerza a que uno de los literales adquiera el color T y el otro el color F, emulando una asignación booleana.

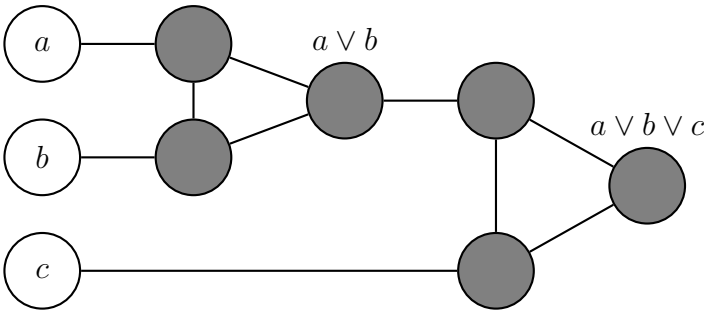
Cabe destacar que en la asignación de la estructura de color inicial, se emplea un abuso de notación. No podemos garantizar que la coloración que vamos a obtener respete el triángulo que hemos definido. No obstante, existirá una aplicación biyectiva que nos permitirá llevar la coloración del grafo obtenida a la que hemos definido, por lo que sin pérdida de generalidad podemos suponer que si existe una coloración válida cumplirá esta restricción.

Tras este proceso inicial, se obtiene la siguiente estructura:

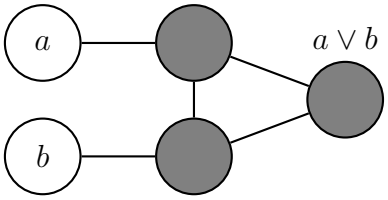


Posteriormente, procedemos a la verificación de las cláusulas. Para ello, empleamos un dispositivo de modelado denominado *Clause Satisfiability Gadget* (o compuerta OR mediante grafos). La finalidad de este componente es añadir las restricciones necesarias para que el problema de 3-coloring sea resoluble si y solo si la fórmula 3-SAT original es satisfacible.

Por cada cláusula $C_j = \{a \vee b \vee c\}$, construimos el siguiente subgrafo:

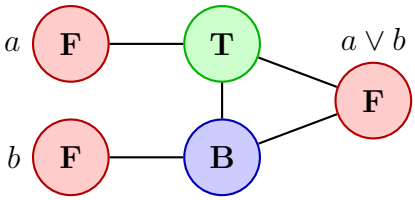


Para demostrar que esta construcción se comporta, en efecto, como una compuerta lógica OR bajo las reglas de coloración de grafos, podemos simplificar el análisis aislando el siguiente subgrafo:

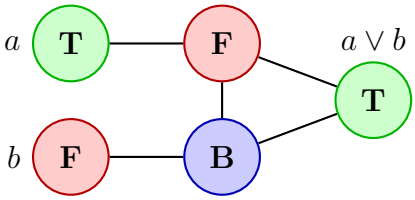


Esta subestructura representa el or binario $a \vee b$. Asumiendo que $\text{color}(a), \text{color}(b) \in \{T, F\}$, evaluamos su comportamiento:

- **Caso en que ambas entradas son falsas ($\textbf{color}(a) = \textbf{color}(b) = F$):** Los dos nodos internos adyacentes a las entradas están conectados entre sí y, por ende, deben tener colores distintos extraídos del conjunto $\{T, B\}$. Al estar el nodo de salida conectado directamente a ambos nodos internos, se ve obligado a rechazar los colores T y B. Por tanto, el nodo de salida $a \vee b$ está forzado a colorearse como F.



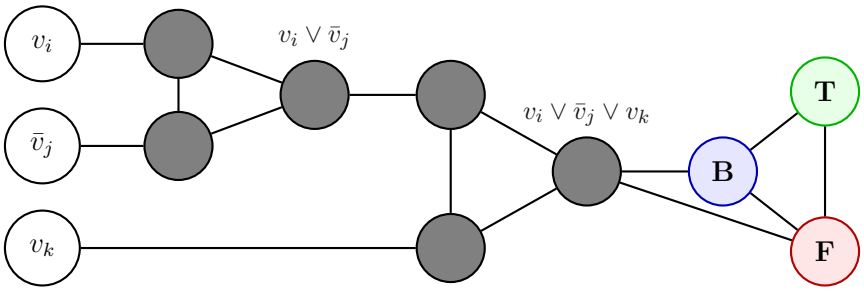
- **Caso en que al menos una entrada es verdadera:** Si una de las entradas (o ambas) se colorea como T, existe al menos una coloración válida para el subgrafo donde el nodo de salida $a \vee b$ puede ser coloreado con el color T.



Para integrar el gadget completo por cada cláusula del problema 3-SAT exacto, conectamos los tres nodos que representan a sus respectivos literales a las entradas correspondientes del gadget de la cláusula. Específicamente, si un literal aparece de forma afirmativa (x_i), se vincula al nodo v_i; si aparece negado ($\neg x_i$), se vincula a su nodo conjugado $\overline{v_i}$.

Por ejemplo, para la cláusula $C_j = x_i \vee \neg x_j \vee x_k$, las tres entradas del gadget se conectarán a los nodos v_i, $\overline{v_j}$ y v_k. Finalmente, para garantizar que la cláusula sea obligatoriamente satisfecha en la solución global, el nodo de salida final del gadget se conecta

directamente a los nodos de control Base (B) y Falso (F). Esta doble conexión impide que dicha salida adopte los colores B o F, forzándola a ser coloreada exclusivamente como Verdadero (T).



3.1.2. El algoritmo es espacio logarítmico

El algoritmo genera el grafo resultante $G' = (V', E')$ de manera directa y secuencial empleando un esquema de indexación, para poder trabajar con índices. Definimos formalmente los identificadores numéricos únicos para cada nodo en V' en función del número total de variables n y de cláusulas m:

- **Nodos de control iniciales:**

$$\mathrm{ID}(T) = 1, \quad \mathrm{ID}(F) = 2, \quad \mathrm{ID}(B) = 3$$

- **Nodos de literales (Variables):** Para cada variable x_i (donde $1 \leq i \leq n$):

$$\mathrm{ID}(v_i) = 3 + 2i - 1 = 2 + 2i$$

$$\mathrm{ID}(\overline{v_i}) = 3 + 2i$$

- **Nodos internos de las cláusulas:** Para cada cláusula C_j (donde $1 \leq j \leq m$) que cuenta con 6 nodos internos para su *gadget*:

$$\mathrm{ID}(G_{j,k}) = 3 + 2n + 6(j-1) + k \quad \text{para } k \in \{1, 2, 3, 4, 5, 6\}$$

El número total de nodos es fijo: $|V'| = 3 + 2n + 6m$. Al conocer la fórmula de indexación, el algoritmo ejecuta la reducción en tres fases secuenciales utilizando únicamente la cinta de trabajo para almacenar punteros y contadores:

1. **Impresión de Vértices:** La máquina inicializa un contador i desde 1 hasta n para escribir los nodos de las variables, y luego un contador j desde 1 hasta m para los nodos de los gadgets. El valor máximo que alcanzan estos contadores está acotado por $\max(n, m)$. Guardar estos índices en binario requiere únicamente $O(\log n + \log m)$ celdas de espacio.

2. **Impresión de Aristas Base:** Se escriben las aristas fijas del triángulo de control $(1, 2), (1, 3), (2, 3)$ en tiempo y espacio constante $O(1)$. Acto seguido, mediante un bucle que incrementa el contador i de 1 a n, se generan las aristas los literales: $(\mathrm{ID}(B), \mathrm{ID}(v_i))$, $(\mathrm{ID}(B), \mathrm{ID}(\overline{v_i}))$ y $(\mathrm{ID}(v_i), \mathrm{ID}(\overline{v_i}))$. Al requerir solo el almacenamiento del índice actual i, el espacio ocupado es $O(\log n)$.

3. **Impresión de Aristas de Cláusulas:** La máquina recorre de izquierda a derecha la lista de cláusulas de la cinta de entrada. Para la cláusula actual j, la máquina lee los tres literales. Almacena temporalmente en la cinta de trabajo tres variables con los índices de los nodos correspondientes a dichos literales (buscando si son afirmativos o negados para aplicar la fórmula de variables). Luego, usando el índice j, calcula los IDs de sus 6 nodos internos e imprime en la salida las aristas internas del gadget y sus conexiones hacia los literales y los nodos de control B y F.

Puesto que la máquina solo mantiene en su cinta de trabajo los contadores de los bucles (i y j),los IDs calculados de los literales involucrados en la cláusula bajo análisis, el valor numérico máximo almacenado en cualquier instante de la computación es proporcional a la longitud de la entrada $|x|$.

Como almacenar un número entero N en representación binaria requiere estrictamente $\lceil \log_2(N) \rceil$ espacios, el espacio máximo ocupado en la cinta de trabajo es logarítmico respecto a la entrada. Por lo tanto, el algoritmo de reducción funciona en espacio logarítmico.

3.1.3. Demostración de la correctitud de la reducción

Para validar la reducción 3-SAT exacto $\propto$ 3-coloring, debemos comprobar la equivalencia de las soluciones:

($\Rightarrow$): Si la instancia de 3-SAT exacto tiene una solución positiva, entonces existe una asignación para las variables $\{x_1^*, \ldots, x_n^*\}$ que satisface simultáneamente todas las cláusulas. A partir de esta asignación, construimos la coloración del grafo de la siguiente manera:

- Si x_i^* = Verdadero, asignamos el color T al nodo v_i y el color F al nodo $\overline{v_i}$.
- Si x_i^* = Falso, asignamos el color F al nodo v_i y el color T al nodo $\overline{v_i}$.

Dado que cada cláusula C_j posee al menos un literal verdadero, la propiedad de la compuerta OR analizada previamente garantiza que existe una asignación válida con el color T en el nodo de salida de cada gadget. Como este nodo de salida está conectado a B y F, la asignación del color T no genera ningún conflicto. Por lo tanto, el grafo completo admite una 3-coloración válida (Solución positiva para 3-coloring).

($\Leftarrow$): Si el problema de 3-coloring responde afirmativamente, significa que existe una coloración válida para el grafo construido utilizando los colores $\{T, F, B\}$. Extraemos la asignación de variables booleanas observando los nodos literales: si el nodo v_i está coloreado como T, asignamos el valor verdadero a la variable x_i; si está coloreado como F, le asignamos el valor falso.

Por construcción, el nodo de salida de cada gadget de cláusula está conectado a los nodos de control B y F, lo que exige que en cualquier coloración válida dicho nodo tenga asignado el color T. Si existiera una cláusula donde sus tres literales de entrada fuesen coloreados como F (equivalente a una cláusula no satisfecha), la propiedad de propagación forzaría a que el nodo final del gadget fuese coloreado como F. Esto entraría en contradicción directa con la arista conectada al nodo de control F, haciendo imposible la 3-coloración del grafo. Por reducción al absurdo, cada cláusula contiene al menos una entrada coloreada como T, lo que asegura que todas las cláusulas se satisfacen bajo la asignación booleana propuesta (Solución positiva para 3-SAT exacto).

3.2. Reducción de 3-coloring a K-coloring ($K > 3$)

3.2.1. Algoritmo de reducción

Dado un grafo no dirigido $G = (V, E)$, donde $V = \{x_1, \ldots, x_n\}$, construimos una nueva instancia para el problema de K-coloring consistente en un grafo no dirigido $H = (V', E')$ mediante el siguiente procedimiento:

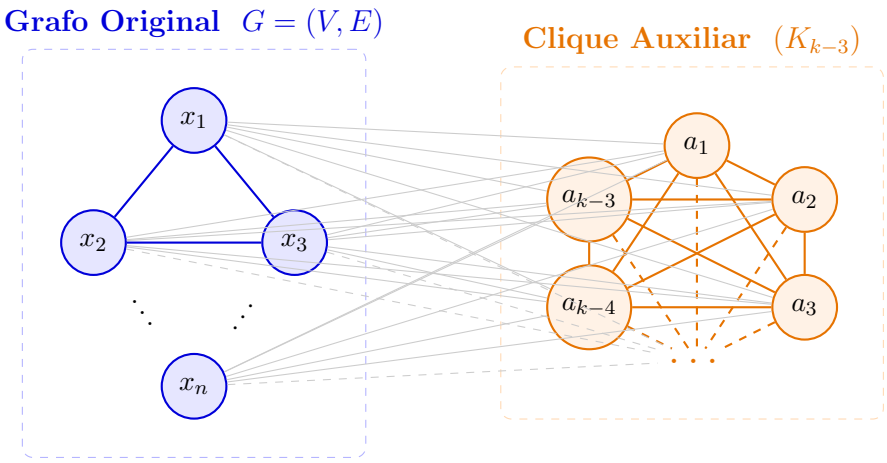
1. **Definición de Vértices:** El conjunto de nodos de H se compone de los nodos originales de G más un conjunto de $K - 3$ nuevos nodos auxiliares:

$$V' = V \cup \{a_1, \ldots, a_{K-3}\}$$

2. **Definición de Aristas:** El conjunto de aristas de H incluye las aristas originales de G, las aristas necesarias para conectar todos los nuevos nodos entre sí (formando un subgrafo completo o *clique* de tamaño $K - 3$), y aristas que conectan cada uno de los nodos auxiliares con absolutamente todos los nodos del grafo original:

$$E' = E \cup \{(a_i, a_j) : 1 \leq i < j \leq K - 3\} \cup \{(x_i, a_j) : x_i \in V,\ a_j \in \{a_1, \ldots, a_{K-3}\}\}$$

Gráficamente, la estructura de la reducción se representa de la siguiente manera:



Bajo esta construcción, cualquier K-coloración válida para el nuevo grafo H debe cumplir obligatoriamente dos restricciones:

- Los nodos auxiliares $\{a_1, \ldots, a_{K-3}\}$ deben recibir colores distintos entre sí, ya que forman un *clique* ($K - 3$ nodos completamente conectados).
- El color de cualquier nodo auxiliar debe ser estrictamente diferente al de cualquier nodo del grafo original V, debido a que existen aristas que conectan de forma exhaustiva el conjunto V con el conjunto de nodos auxiliares.

3.2.2. El algoritmo es espacio logarítmico

La máquina de Turing genera el nuevo grafo H de forma secuencial a través de las siguientes etapas:

1. **Generación del conjunto de vértices V':**

- **Copiado de nodos originales:** La máquina recorre la lista de vértices V en la cinta de entrada y los transcribe directamente a la salida. Para controlar este proceso, utiliza un contador binario i que va desde 1 hasta n.
- **Inserción de nodos auxiliares:** Inmediatamente después, la máquina inicializa un segundo contador j desde 1 hasta $K-3$ para imprimir en la salida los nuevos identificadores correspondientes a los nodos del *clique* de control $\{a_1, \ldots, a_{K-3}\}$.

2. **Generación del conjunto de aristas E':**

   - **Copiado de aristas originales:** La máquina se posiciona en la sección de aristas E de la entrada y las copia textualmente en la cinta de salida.
   - **Construcción del clique de control:** Para añadir las aristas internas de la estructura auxiliar, la máquina ejecuta dos bucles anidados utilizando dos contadores en la cinta de trabajo, i y j, que varían de 1 a $K-3$. De este modo, imprime de forma secuencial todas las aristas de la forma (a_i, a_j) para todo $i < j$.
   - **Conexión exhaustiva:** Finalmente, para añadir las aristas que enlazan los nodos auxiliares con los originales, se implementa otro doble bucle. El primer contador recorre los nodos originales de 1 a n, mientras que el segundo contador recorre los nodos auxiliares de 1 a $K-3$. El algoritmo escribe en la salida la arista (x_i, a_j) para cada combinación.

Durante todo el transcurso de la reducción, la máquina de Turing jamás almacena el grafo resultante ni el original en su memoria intermedia. Las únicas variables que persisten en la cinta de trabajo son los límites numéricos n y K, junto con los contadores de los bucles (i y j).

Dado que el valor máximo que puede almacenar cualquiera de estos contadores está acotado superiormente por $\max(n, K)$, y considerando que un número entero N codificado en base binaria ocupa exactamente $\lceil \log_2(N) \rceil$ celdas de memoria, el espacio total consumido en la cinta de trabajo es logarítmico.

3.2.3. Demostración de la correctitud de la reducción

Para validar que 3-coloring $\propto K$-coloring, demostramos la equivalencia de las soluciones en ambas direcciones:

($\Rightarrow$): Si el problema 3-coloring sobre el grafo G responde afirmativamente, significa que existe una función de coloración válida para sus vértices utilizando un conjunto de tres colores, por ejemplo, $\{1, 2, 3\}$.

Para colorear el grafo H, mantenemos esta misma asignación para los nodos $\{x_1, \ldots, x_n\}$. Acto seguido, para los $K-3$ nodos auxiliares $\{a_1, \ldots, a_{K-3}\}$, empleamos una paleta de $K-3$ colores completamente nuevos y distintos a los tres primeros (por ejemplo, $\{4, 5, \ldots, K\}$), asignando un color único a cada nodo a_j.

Es obvio que la nueva coloración es válida, las aristas originales no suponen problema, mientras que las aristas añadidas siempre son conexiones con un nodo auxiliar y estos tienen colores distintos a los de cualquier otro nodo.

Por lo tanto, el grafo H es K-coloreable (Solución positiva para K-coloring).

($\Leftarrow$): Supongamos ahora que el grafo H admite una K-coloración válida. Como se justificó en el análisis de la construcción, los $K-3$ nodos del conjunto $\{a_1, \ldots, a_{K-3}\}$ están completamente conectados entre sí, lo que obliga a utilizar exactamente $K-3$ colores distintos para ellos.

Además, dado que cada uno de estos nodos auxiliares comparte una arista con cada uno de los vértices del conjunto original $V = \{x_1, \ldots, x_n\}$, ninguno de los nodos originales puede reutilizar ninguno de los $K-3$ colores consumidos por el *clique* de control. En consecuencia, los nodos originales $\{x_1, \ldots, x_n\}$ disponen únicamente de 3 colores para ser coloreados.

Dado que la coloración en H es válida, la restricción al grafo G también presenta una coloración válida y por lo indicado anteriormente, usando a lo sumo 3 colores. (Solución positiva para 3-coloring).

3.3. Reducción de K-coloring a K-cubrimiento

3.3.1. Algoritmo de reducción

El problema de K-cubrimiento es complementario al de K-coloring.

Dada una instancia del problema de K-coloring constituida por un grafo no dirigido $G = (V, E)$, donde $V = \{x_1, \ldots, x_n\}$, y un entero K, construimos una instancia para el problema de K-cubrimiento representada por el grafo complementario $G' = (V, E')$. En esta construcción, el conjunto de vértices se mantiene idéntico y el conjunto de aristas E' se define formalmente como:

$$E' = \{\{x_i, x_j\} : x_i, x_j \in V,\, x_i \neq x_j\} \setminus E$$

Conceptualmente, el algoritmo de reducción invierte la estructura de adyacencia del grafo original: si existía una arista entre dos nodos en G, esta se elimina en G'; por el contrario, si dos nodos no estaban conectados en G, se añade una arista entre ellos en G'.

3.3.2. El algoritmo es espacio logarítmico

La máquina de Turing genera el nuevo grafo G' de forma secuencial a través de las siguientes etapas:

1. **Copiado de Vértices:** El conjunto de nodos de G' es idéntico al de G. La máquina inicializa un contador i en su cinta de trabajo para recorrer secuencialmente los n nodos de la entrada y transcribirlos directamente a la salida.

2. **Generación de Aristas Complementarias:** Para construir E', la máquina ejecuta un doble bucle mediante dos variables de control, i y j, que representan todos los pares posibles de nodos distintos (x_i, x_j) con $1 \leq i < j \leq n$. Para cada par:

   - La máquina posiciona un puntero en el inicio de la lista de aristas E de la cinta de entrada.
   - Escanea secuencialmente toda la lista E buscando si la arista $\{x_i, x_j\}$ está presente.
   - Si el escaneo finaliza y la arista no se encuentra en la entrada, la máquina escribe de forma inmediata la arista $\{x_i, x_j\}$ en la cinta de salida. Si la arista ya existía, simplemente se ignora y se continúa con el siguiente par.

Durante toda la ejecución, la memoria de trabajo de la máquina solo almacena los dos contadores de los nodos (i y j)

Dado que el valor numérico máximo de cualquier índice o posición de la cinta está acotado directamente por el tamaño total de la entrada $|x|$, su representación binaria ocupa espacio logarítmico. En consecuencia, la reducción es espacio logarítmica.

3.3.3. Demostración de la correctitud de la reducción

Para demostrar la validez de la reducción K-coloring $\propto$ K-cubrimiento, comprobamos la equivalencia de las soluciones en ambas direcciones:

($\Rightarrow$): Supongamos que la instancia de K-coloring para el grafo G admite una solución positiva. Esto implica la existencia de una función de coloración válida que particiona el conjunto de vértices V en K subconjuntos disjuntos $V_1, \ldots, V_K$ (donde cada subconjunto agrupa a los nodos que comparten el mismo color), cumpliendo estrictamente que:

$$\forall V_i, \quad \forall a, b \in V_i \implies \{a, b\} \notin E$$

Dado que el conjunto de aristas de G' es el complementario exacto de E, cualquier par de nodos distintos a, b que pertenezcan al mismo conjunto V_i y que no compartan una arista en G, obligatoriamente deben compartir una arista en G'. Por lo tanto, en el grafo modificado se cumple que:

$$\forall V_i, \quad \forall a, b \in V_i \implies \{a, b\} \in E'$$

Esto demuestra que cada subconjunto V_i induce un subgrafo completo (un *clique*) en G'. Como existen exactamente K conjuntos, la misma partición es una solución válida para el problema de K-cubrimiento (Solución positiva).

($\Leftarrow$): Supongamos ahora que el problema de K-cubrimiento sobre el grafo G' responde de forma afirmativa. Esto significa que existe una partición del conjunto de vértices V en K subconjuntos $V_1, \ldots, V_K$ de tal manera que cada subconjunto es completo en G', es decir:

$$\forall V_i, \quad \forall a, b \in V_i \implies \{a, b\} \in E'$$

Aplicando la definición del grafo complementario, si la arista $\{a, b\}$ existe en E', entonces dicha arista no puede existir en el conjunto de aristas original E. Por consiguiente, se deduce que:

$$\forall V_i, \quad \forall a, b \in V_i \implies \{a, b\} \notin E$$

Esta propiedad define exactamente la restricción de una coloración válida de grafos: no existen dos nodos adyacentes en G que pertenezcan al mismo conjunto V_i. Al asignar un color único y diferenciado a cada uno de los K subconjuntos, obtenemos una asignación cromática válida para G utilizando a lo sumo K colores. Por lo tanto, la instancia original de K-coloring es resoluble (Solución positiva).

4. Conclusión

Tras esta última reducción tenemos la cadena de reducciones.

$$\text{3-SAT exacto} \propto \text{3-coloring} \propto K\text{-coloring} \propto K\text{-cubrimiento}$$

Sabemos por teoría que 3-SAT exacto es NP-Completo y que cada uno de los problemas presentes en la cadena de reducciones son NP. Lo que nos permite concluir que todos son NP-Completos.

Por lo que queda demostrado que K-cubrimiento es NP-Completo.

5. Declaración de uso de Inteligencia Artificial

De conformidad con las directrices de integridad científica y transparencia, se declara explícitamente el uso de herramientas de Inteligencia Artificial en la elaboración de este trabajo y de la presente memoria:

- **Herramientas y modelos utilizados:** GitHub Copilot propulsado por el modelo de lenguaje grande Gemini 3.1 Pro.
- **Funciones empleadas:** La Inteligencia Artificial se ha utilizado fundamentalmente: Como asistencia en redacción de la memoria. Sus funciones se han limitado a la revisión ortotipográfica, el ajuste del tono académico del texto y la generación de la estructura y código LaTeX.
- **Secciones aplicadas:** La asistencia de redacción se ha aplicado de forma transversal en todo el documento. No se ha empleado IA para la elaboración de las demostraciones.

Se certifica que todo el contenido documental asistido por IA ha sido rigurosamente supervisado, revisado y validado de forma manual.

Referencias

[1] Richard M. Karp. *Reducibility among combinatorial problems.* In R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972. https://doi.org/10.1007/978-1-4684-2001-2_9

[2] Lalla Mouatadid. *Course Notes on 3-Coloring (CSC 373 - Algorithm Design, Analysis, and Complexity).* University of Toronto, Summer 2016. https://www.cs.toronto.edu/~lalla/373s16/notes/3col.pdf