Teórica 10: Teoría de complejidad computacional - 2da parte

1. Clase NP-completo

Este concepto es el centro de la teoría de la complejidad computacional. Intuitivamente, agrupa a los problemas de decisión que son los más difíciles de la clase **NP**.

Los problemas de la clase **NP-Completo** tienen dos propiedades sumamente importantes:

- No se conocen algoritmos polinomiales para resolverlos.
- Si existe un algoritmo polinomial para uno de ellos, existen algoritmos polinomiales para todos ellos.

Una creencia es que los problemas de esta clase son intratables desde el punto de vista computacional, por lo que no existirían algoritmos eficientes para resolverlos. Pero ésto no está demostrado y podría no ser verdadero.

Formalmente:

Definición 1. Un problema de decisión Π es **NP-completo** si:

1. $\Pi \in$ **NP**
2. $\forall \bar{\Pi} \in$ **NP**, $\bar{\Pi} \leq_p \Pi$

Se define la clase ***NP-difícil*** como los problemas que cumplen la condición 2 (al menos tan "difícil" como todos los problemas de **NP**).

SAT fue el primer problema que se demostró que pertenece a la clase **NP-completo**, dando origen a esta clase.

Teorema 1 (Teorema de Cook(1971) - Levin(1973))**.** *SAT es* **NP-completo**.

La demostración de Cook es directa: considera un problema genérico $\Pi \in$ NP y una instancia genérica $d \in D_\Pi$. A partir de la hipotética MTND que resuelve Π, genera en tiempo polinomial una fórmula lógica $\varphi_{\Pi,d}$ en forma normal (conjunción de disyunciones) tal que $d \in Y_\Pi$ si, y sólo si, $\varphi_{\Pi,d}$ es satisfactible.

Usando la transitividad de las reducciones polinomiales, a partir de este primer resultado podemos probar que otros problemas son **NP-completos**.

Si Π es un problema de decisión, podemos probar que $\Pi \in$ **NP-completo** encontrando otro problema Π_1 que ya sabemos que es **NP-completo** y demostrando que:

1. $\Pi \in$ **NP**
2. $\Pi_1 \leq_p \Pi$

La segunda condición en la definición de problema NP-completo se deriva de la transitividad:

Sea Π' un problema cualquiera de NP. Como Π_1 es NP-completo, sabemos que $\Pi' \leq_p \Pi_1$. Como ahora probamos que $\Pi_1 \leq_p \Pi$, resulta, por transitividad de las reducciones polinomiales, $\Pi' \leq_p \Pi$.

Desde 1971, se ha probado la NP-completitud de muchos problemas usando el método anterior. A partir del Teorema de Cook-Levin, Richard Karp demostró en 1972 que otros 21 problemas son NP-completos. Actualmente se conocen más de 3.000 problemas pertenecientes a esta clase.

Ejemplo 1. *CLIQUE es* **NP-completo**.

Para demostrar que CLIQUE es **NP-completo**, *alcanza con probar que:*

1. *CLIQUE* $\in$ **NP**.
2. *Para algún problema* $\Pi \in$ **NP-completo**, $\Pi \leq_p$ *CLIQUE.*

En el ejemplo 12 ya probamos **??**.

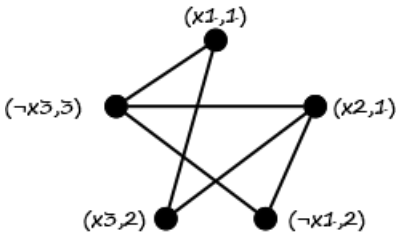
Para probar la segunda condición utilizaremos SAT y demostraremos que SAT $\leq_p$ *CLIQUE:*

Dada una intancia de SAT, I_S, *tenemos que transformarla polinomialmente en una instancia de CLIQUE,* I_C, *tal que CLIQUE responda SI para* I_C *si, y sólo si, SAT responde SI para* I_S.

Para ésto, tomemos una instancia genérica de SAT, I_S, *con* k_S *cláusulas.*

Definimos $k_C = k_S$ *y* $G = (V, X)$ *de la siguiente manera. Para cada literal* x *en la cláusula* i, *ponemos un vértice* $(x, i) \in V$ *y* $((x, i), (y, j)) \in X$ *si, y sólo si,* $x \neq \bar{y}$ *y* $i \neq j$.

$$\varphi = C_1 \wedge C_2 \wedge C_3 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$$



Tenemos que ver que I_S *es satisfcible si, y sólos si,* G *tiene una clique de tamaño mayor o igual a* k.

Si I_S *es satisfacible, existe un conjunto de literales que hacen verdaderas las* k_S *cláusulas simultáneamente. Tomando para cada cláusula un literal de este conjunto, representarán distintos vértices en* G *todos adyacentes entre sí (por ser de distintas cláusulas), formando una clique de tamaño* $k_S = k_C$ *en* G.

Por otro lado, si H *es una clique de* G *de tamaño mayor o igual a* k_C, *cada vértice de* H *representará un literal de una cláusula distinta que pueden ser simultáneamente verdaderos por no ser complementarios. Asignándole valor verdadero a estos literales,* I_S *toma valor verdadero. Por lo tanto,* I_S *es satisfacible.*

Ejemplo 2. *3-SAT es* **NP-completo**.

El problema 3-SAT es un caso particular del problema SAT, en el cual se pide que cada cláusula tenga exáctamente tres literales.

1. *Como 3-SAT es un caso particular de SAT y ya sabemos que SAT* $\in$ **NP**, *podemos decir que 3-SAT está en* **NP**.

2. *Para probar que 3-SAT es* **NP-completo**, *vamos a reducir polinomialmente SAT a 3-SAT.*

   Tomemos una instancia genérica de SAT, $\varphi = C_1 \wedge \cdots \wedge C_m$ *sobre las variables booleanas* $x_1, \ldots, x_n$. *Vamos a reemplazar cada* C_i *por una conjunción de disyunciones* φ'_i, *donde cada disyunción tenga tres literales, de manera que* φ *sea satisfactible si, y sólo si,* $\varphi' = \varphi'_1 \wedge \cdots \wedge \varphi'_m$ *lo es.*

   - *Si* C_i *tiene tres literales:*
   $$\varphi'_i = C_i.$$
   - C_i *tiene dos literales,* a_1 *y* a_2, *agregamos una variable nueva y y definimos:*
   $$C_i = (a_1 \vee a_2) \rightarrow \varphi'_i = (a_1 \vee a_2 \vee y^i) \wedge (a_1 \vee a_2 \vee \neg y^i).$$
   - *Si* C_i *tiene* $k \geq 4$ *literales, agregamos* $k - 3$ *variables nuevas:*
   $$C_i = (a_1 \vee a_2 \vee \ldots \vee a_r \vee \ldots \vee a_{k-1} \vee a_k) \rightarrow$$
   $$\varphi'_i = (a_1 \vee a_2 \vee y^i_1) \wedge \ldots \wedge (\neg y^i_{r-2} \vee a_r \vee y^i_{r-1}) \wedge \ldots \wedge (\neg y^i_{k-3} \vee a_{k-1} \vee a_k)$$
   Por ejemplo: $C_i = (a_1 \vee a_2 \vee a_3 \vee a_4 \vee a_5) \rightarrow \varphi'_i = (a_1 \vee a_2 \vee y^i_1) \wedge (\neg y^i_1 \vee a_3 \vee y^i_2) \wedge (\neg y^i_2 \vee a_4 \vee a_5)$

   Falta ver que φ *es satisfactible si, y sólo si,* $\varphi' = \varphi'_1 \wedge \cdots \wedge \varphi'_m$ *lo es.*

   Para ver esto, supongamos primero que φ *es satisfacible. Sea* $x^*_1, \ldots, x^*_n$ *valores de las variables* $x_1, \ldots, x_n$ *que hacen verdadera* φ.

   Entonces por lo menos un literal a^i_j *de cada cláusula* C_i *tiene que ser verdadero (si hay más de uno elegimos el primero por ejemplo). Para ver que* φ' *es satisfacible, mantenemos el valor de las variables originales* $x_j = x^*_j$ *para* $j = 1, \ldots, n$. *Si el literal* a^i_j *es verdadero, fijamos* $y^i_1, \ldots, y^i_{j-2}$ *en verdadero y* $y^i_{j-1}, \ldots, y^i_{k-3}$ *en falso. Esto hará a la cláusula* φ'_i *verdadera, haciendo a* φ *verdadera.*

   Ahora supongamos que $\varphi' = \varphi'_1 \wedge \cdots \wedge \varphi'_m$ *es verdadera. Al menos uno de los literales* a^i_j *de la conjunción de cláusulas* φ'_i *debe ser verdadero, de lo contrario* y^i_1 *debe ser verdadera, pero eso impone que* y^i_2 *sea verdadera, y así siguiendo. Pero ésto hará que la última cláusula* φ'_m *sea falsa, contradiciendo que* φ' *es verdadera. Ésto asegura que tomando esos valores de las variables* x_j, *la cláusula* C_i *es verdadera.*

Ejemplo 3. *COLOREO es* **NP-completo**.

1. *Es fácil probar que COLOREO es* **NP**.

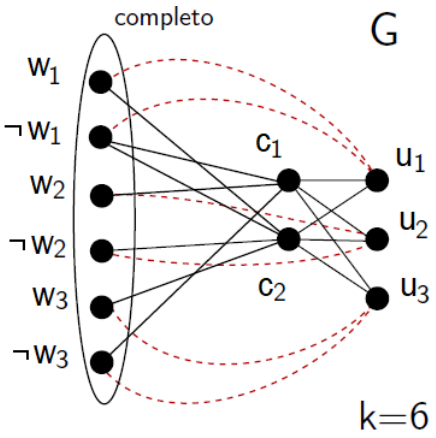
2. *Para probar que COLOREO es* **NP-completo**, *vamos a reducir polinomialmente SAT a COLOREO.*

   Tomemos una instancia genérica de SAT, $\varphi = C_1 \wedge \cdots \wedge C_m$ *sobre las variables* $x_1, \ldots, x_n$*. Vamos a construir un grafo* G *y determinar un número* k *de manera que* φ *sea satisfactible si, y sólo si,* G *se puede colorear con* k*-colores.*

   Definimos k *= dos veces la cantidad de variables de* φ *y* $G = (V_1 \cup V_2 \cup V_3, X)$ *como:*

   - V_1*: dos vértice* w_i *y* $\neg w_i$ *por cada variable* x_i *(representando a todos los posibles literales), todos adyacentes entre si, formando un* K_k*.*
   - V_2*: un vértice* c_j *por cada cláusula* C_j*, adyacente a los literales de* V_1 *que no aparecen en la cláusula.*
   - V_3*: un vértice* u_i *por cada variable* x_i*, adyacente a todos los vértices de* V_2 *y a los literales de* V_1 *correspondientes a otras variables (*u_i *no adyacente a* w_i *ni a* $\neg w_i$*).*

   $$\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$$

   Falta ver que φ *es satisfactible si, y sólo si,* G *es* k*-coloreable.*

   Supongamos que φ *es satisfactible. Sea* $x_1^*, \ldots, x_n^*$ *valores de las variables* $x_1, \ldots, x_n$ *que hacen verdadera* φ*. Pintamos a* G *con* k *colores de la siguiente forma:*

   - *Para cada vértice de* V_1 *obviamente utilizamos un color distinto.*
   - *Al vértice* $c_j \in V_2$ *lo pintamos del color del vértice de* V_1 *correspondiente a un literal que la hace verdadera.*
   - *Si* $x_i^* = V$*, pintamos a* $u_i \in V_3$ *con el mismo color que* $\neg w_i$*, y si* $x_i^* = F$*, con el mismo color que* w_i*.*

   Obtenemos así un k*-coloreo de* G*.*

   Consideremos ahora que G *es* k*-coloreable. En un* k*-coloreo de* G*, cada uno de los* k *colores se utiliza exáctamente una vez en los vértices de* V_1*. Un vértice* $u_i \in V_3$ *tendrá el mismo color que* w_i *o de* $\neg w_i$*, ya que es adyacente al resto de los vértices de* V_1*. El color del vértice* c_j *será el mismo que tiene algún vértice de* V_1 *representando a un literal de la cláusula* C_j *(porque es adyacente al resto de los vértices de* V_1*) y distinto a*

todos los colores de los vértices de V_3.

Si u_i tiene el mismo color que w_i, le asignamos valor F a la variable x_i, mientras que si tiene el mismo color que $\neg w_i$, le asignamos valor V. Entonces, el vértice c_j comparte color con un literal que es verdadero en esta asignación, haciendo este literal verdadera a la cláusula C_j. Por lo tanto esta asignación hace verdadera a φ, demostrando que φ es satisfacible.

Ejemplo 4. *CONJUNTO INDEPENDIENTE es* **NP-completo**.

1. *Es fácil probar que CONJUNTO INDEPENDIENTE es* **NP**.
2. *Para probar que es* **NP-difícil**, *vamos a reducir polinomialmente CLIQUE a CONJUNTO INDEPENDIENTE.*

   Tomemos una instancia genérica de CLIQUE, I_C, esto es un grafo G y $k \in \mathbb{N}$ y definamos la instancia I_{CI} de CONJUNTO INDEPENDIENTE, $\bar{G}$ y k.
   Como una clique S en un grafo G es un conjunto independiente en $\bar{G}$, se cumple que G tiene una clique de tamaño mayor o igual a k si, y sólo si, $\bar{G}$ tiene un conjunto independiente de tamaño mayor o igual a k.

   Ésto muestra que CLIQUE responde que SI para I_C si, y sólo si, CONJUNTO INDEPENDIENTE responde SI para I_{CI}.

Ejemplo 5. *Dado un grafo $G = (V, X)$, un* ***recubrimiento de las aristas*** *de G, es un conjunto $R_a \subseteq V$ de vértices tal que para todo $e \in X$, e es incidente al menos a un vértice $v \in R_n$. El problema de RECUBRIMIENTO DE ARISTAS en su versión de decisión es: dados un grafo $G = (V, X)$ y $k \in N$, ¿G tiene un recubrimiento de aristas de tamaño menor o igual a k?*

NP-completo.

1. *Es fácil probar que RECUBRIMIENTO DE ARISTAS es* **NP**.

2. *Para probar que es* **NP-completo**, *vamos a reducir polinomialmente CONJUNTO INDEPENDIENTE a RECUBRIMIENTO DE ARISTAS.*

   Tomamos una instancia genérica de CONJUNTO INDEPENDIENTE, $G = (V, X)$ de n vértices y $k \in \mathbb{N}$, $k \leq n$ y la transformamos a la instancia de RECUBRIMIENTO DE ARISTAS, G y $k' = n - k$.

   Dado $G = (V, X)$, S es conjunto independiente si, y sólo si, $V \setminus S$ es recubrimiento de aristas. Esto asegura que G tendrá un conjunto independiente de tamaño mayor o igual a k si, y sólo si, G tiene un recubrimiento de aristas de tamaño menor o igual a k'.

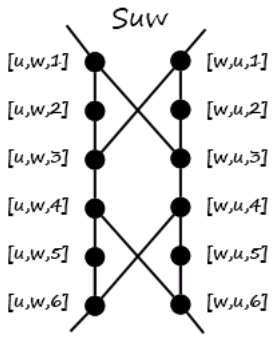
Ejemplo 6. *CIRCUITO HAMILTONIANO es* **NP-completo** *(optativo).*

1. *En el Ejemplo 13 mostramos que este problema es* **NP**.

2. *Para probar que es* **NP-completo**, *vamos a reducir polinomialmente RECUBRIMIENTO DE ARISTAS a CIRCUITO HAMILTONIANO.*

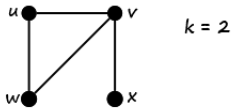
   Tomamos una instancia genérica de RECUBRIMIENTO DE ARISTAS, $G = (V, X)$ *de* n *vértices y* $k \in \mathbb{N}$ *y la transformamos a una instancia de CIRCUITO HAMILTONIANO,* $G' = (V', X')$*, de forma tal que* G *tenga un recubrimiento de aristas de* k *o menos vértices si, y sólo si,* G' *es hamiltoniano.*

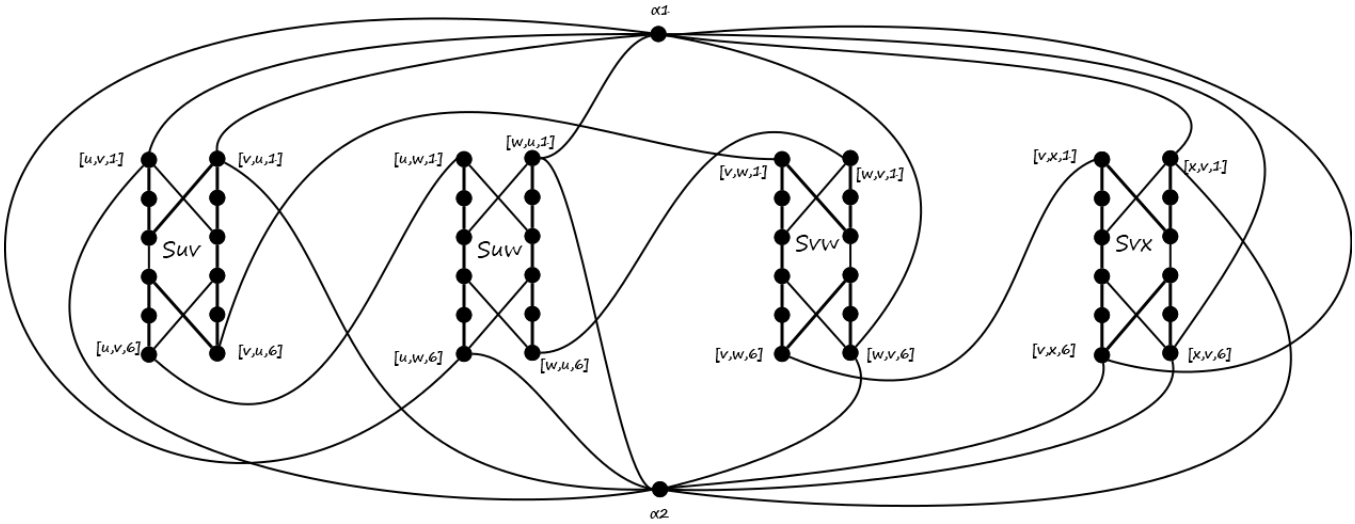
   Por cada arista $(u, w) \in X$*, colocamos 12 vértices en* G' *con la siguiente estructura* S_{uw}*:*



   Dado un vértice $u \in V$*, sean* $w_1, \ldots, w_{d(u)}$ *sus adyacentes. Entonces hacemos adyacentes en* G' *los vértices* $[u, w_i, 6]$ *y* $[u, w_{i+1}, 1]$ *para* $i = 1, \ldots, d(u) - 1$*.*

   Además, colocamos en V' k *nuevos vértices* $\alpha_1, \ldots, \alpha_k$ *y los hacemos adyacentes a* $[u, w_1, 1]$ *y* $[u, w_{d(u)}, 6]$*.*

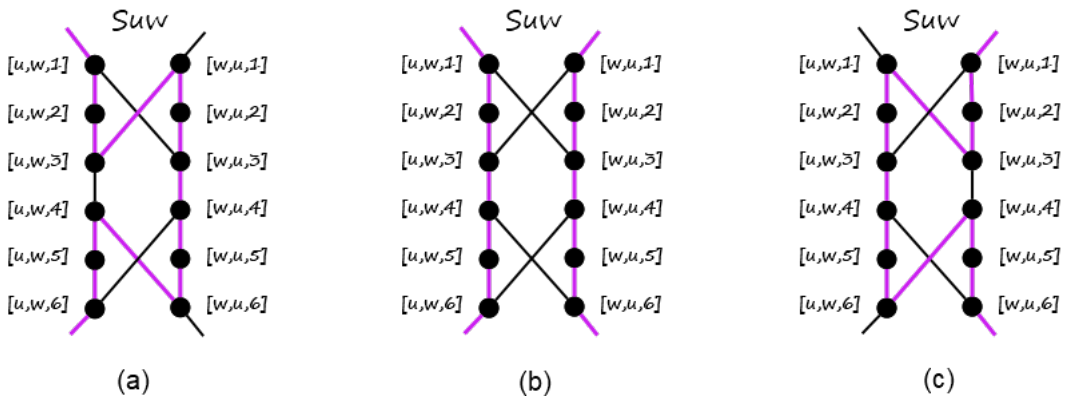




Esta transformación es polinomial. Ahora veamos que G tiene un recubrimiento de aristas de tamaño menor o igual a k si, y sólo si, G' tiene un circuito hamiltoniano.

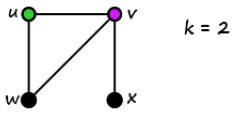
Supongamos que G tiene un recubrimiento de aristas, R, de tamaño k (si fuese de tamaño menor agregamos vértices para que sea de tamaño k), $R = \{r_1, \ldots, r_k\}$. Queremos ver que G' tiene un circuito hamiltoniano.

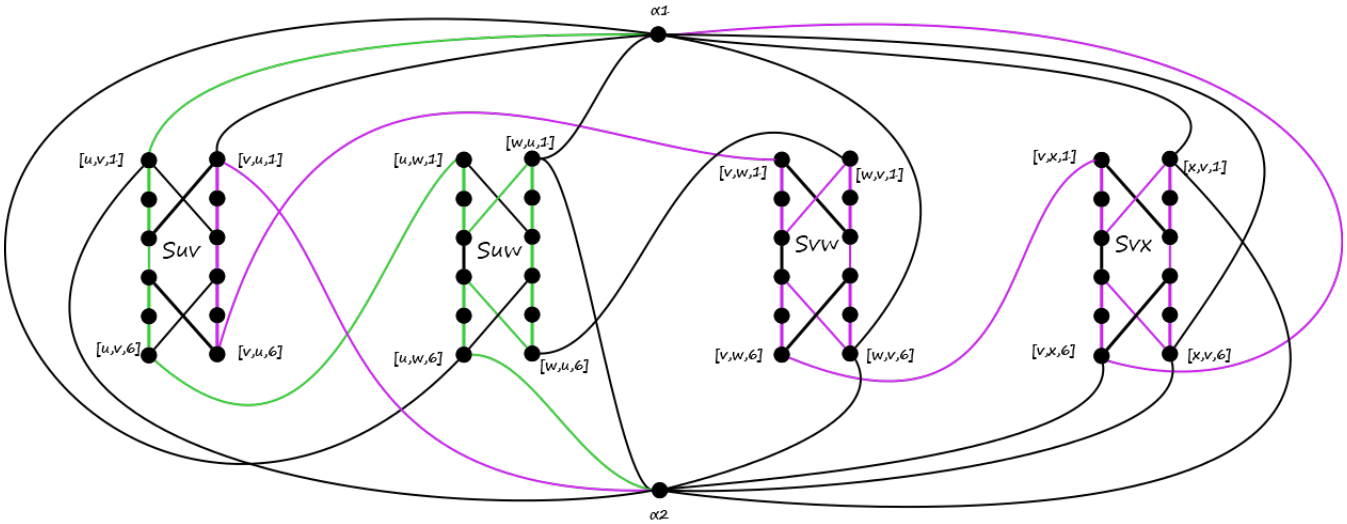
Como solamente los vértices con tercera coordenada 1 ó 6 tienen adyacentes fuera de la estructura y un circuito hamiltoniano debe recorrer todos los vértices, en cada estructura un circuito hamiltoniano debe seguir alguno de estos tres patrones:



Si el vértice $u \in R$ y $w \notin R$, vamos a elegir el patrón (a). En cambio, si $u \notin R$ y $w \in R$, nos quedaremos con el patrón (c). Si ambos vértices están en R, seguiremos el patrón (c).

Entonces armaremos el circuito hamiltoniano: $\alpha_1[r_1, w_{r_1}, 1] \dots [r_1, w_{r_1}, 6][r_1, w_{r_2}, 1] \dots [r_1, w_{d(r_1)}, 6]\alpha_2 \dots \alpha_1$.





Ahora, si G' *tiene un circuito hamiltoniano* H*, debemos probar que* G *tiene un recubrimiento de aristas* R *de cardinal menor o igual a* k*. Como* H *recorre todos los vértices, pasará por todas las estructuras* S *siguiendo alguno de los tres patrones posibles. Si quitamos los vértices* α_i *del circuito hamiltoniano* H*, éste quedará partido en* k *subcaminos. Cada uno de estos subcaminos comenzará en un vértice de tercera coordenada 1,* $[u, v, 1]$*, atravesará la estructura* S_{uv} *siguiendo alguno de los tres patrones posibles, y saldrá de la estructura por el vértice* $[u, v, 6]$*, para luego finalizar o entrar en otra estructura por un vértice* $[u, w, 1]$ *y así continuar. Por el diseño de* G'*, todas las estructuras que recorre un subcamino tienen un vértice en común,* u*, entrando y saliendo de ellas por vértices de primera coordenada* u*. Ese vértice en común* u *formará parte del recubrimiento de aristas* R*. Como* H *recorre todos las estructuras (por recorrer todos los vértices), todas las aristas quedarán recubiertas por al menos un vértice de* R*, y como hay* k *subcaminos, el cardinal de* R *será* k*.*

Ejemplo 7. *TSP es* **NP-completo**.

1. *En el Ejemplo 14 vimos que TSP es* **NP**.

2. *Para probar que es* **NP-completo**, *vamos a reducir polinomialmente CIRCUITO HAMILTONIANO a TSP.*

   Dado un grafo G *vamos a construir un grafo completo* G' *con peso en las aristas y definir* $k \in \mathbb{Z}_{\geq 0}$*, de forma tal que* G *sea hamiltoniano si, y sólo si,* G' *tiene un circuito hamiltoniano de peso menor o igual a* k*.*

   Si G *tiene* n *vértices,* G' *será el grafo completo de* n *vértices. Fijaremos peso 0 para las aristas de* G' *que están en* G *y 1 para las que no están y definiremos* $k = 0$*. Entonces, claramente,* G' *tiene un circuito hamiltoniano*

de peso 0 si, y sólo si, G es hamiltoniano.

Muchos investigadores han intentado encontrar un algoritmo polinomial para resolver algún problema $\Pi \in$ **NP-completo**. Veamos por qué esto es tan importante.

Lema 1. *Si existe un problema* Π *en* **NP-completo** $\cap$ **P**, *entonces* **P**=**NP**.

Demostración. Si $\Pi \in$ **NP-completo** $\cap$ **P**, existe un algoritmo polinomial que resuelve Π, por estar en **P**. Por otro lado, como Π es **NP-completo**, para todo $\Pi' \in$ **NP**, $\Pi' \leq_p \Pi$.

Sea $\Pi' \in$ NP. Aplicando la reducción polinomial que transforma instancias de Π' en instancias de Π y luego el algoritmo polinomial que resuelve Π, por definición de reducción polinomial, se obtiene un algoritmo polinomial que resuelve Π'.

■

Hasta el momento no se conoce ningún problema en **NP-completo** $\cap$ **P**. Sin embargo, tampoco se ha demostrado que exista algún problema en **NP** $\setminus$ **P**. En ese caso se probaría que **P** $\neq$ **NP**.

2. La clase co-NP

Vimos que circuito hamiltoninano está en la clase **NP**. Pero consideremos ahora su versión *inversa*, ***circuito hamiltoniano complemento***:

Dado un grafo G, ¿es G no hamiltoniano?

¿Estará este problema también en **NP**? No sabemos la respuesta. Hasta el momento, la forma de verificar que un grafo general no tiene un circuito hamiltoniano es listar todas las permutaciones de sus vértices y verificar que ninguna define un cirtuito. Este certificado obviamente no es polinomial, por lo tanto no nos sirve para responder la pregunta.

Definición 2. El ***problema complemento*** de un problema de decisión Π, Π^c, es el problema de decisión cuyo conjunto de instancias es igual al de Π y responde ***SI*** para las instancias que Π responde ***NO*** y viceversa. Es decir Π^c es el problema de decisión tal que:

$$D_{\Pi^c} = D_\Pi \text{ y } Y_{\Pi^c} = D_\Pi \setminus Y_\Pi$$

Ejemplo 8. *El problema de primalidad y el problema de número compuesto son problemas complementarios.*

Ejemplo 9. *El problema complemento de TSP es:*

Dado un grafo $G = (V, X)$ con peso en sus aristas y $k \in \mathbb{N}$, ¿es verdad que todos los circuitos hamiltonianos de G tienen peso mayor que k?

Ejemplo 10. *Los problemas:*

- *Dado un grafo $G = (V, X)$, ¿es conexo?*
- *Dado un grafo $G = (V, X)$, ¿es disconexo?*

son problemas complemenatarios.

Es fácil ver el siguiente resultado:

Proposición 1. *Si un problema* Π *pertenece a* **P**, *entonces* Π^c *también pertenece a* **P**.

Demostración.

Como $\Pi \in$ **P**, existe un algoritmo polinomial para resolver Π. Este mismo algoritmo sirve para resolver Π^c invirtiendo la respuesta. ■

Este argumento no aplica para la clase **NP**. Es decir, si un problema Π está en **NP** no sirve este argumento para demostrar que Π^c está en **NP** (es más, no se sabe si ésto es cierto o no). Este concepto da origen a la siguiente definición:

Definición 3. Un problema de decisión pertenece a la clase **Co-NP** si dada una instancia de ***NO*** y evidencia polinomial de la misma, puede ser verificada en tiempo polinomial.

Ejemplo 11. *Circuito hamiltoniano complemento pertenece a la clase* **Co-NP**, *porque una instancia de* ***NO*** *es un grafo que tiene un ciruito hamiltoniano, y por lo tanto, podemos dar evidencia chequeable polinomialmente.*

La relación que podemos plantear entre estas clases es:

Proposición 2. *Si un problema* Π *pertenece a* **NP**, *entonces* Π^c *pertenece a* **Co-NP**.

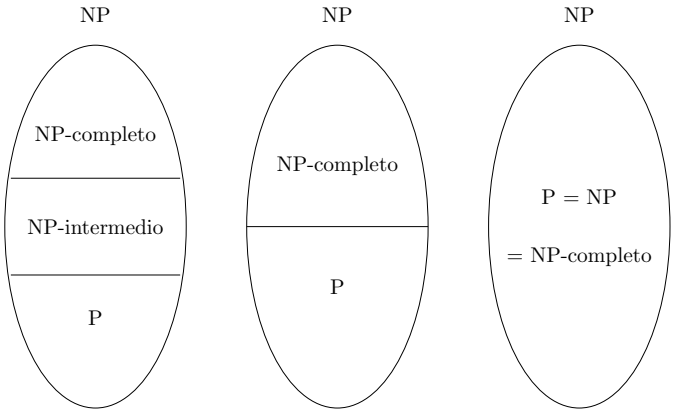
Demostración. Como $\Pi \in$ **NP**, para toda instancia de ***SI*** se puede dar evidencia de esto chequeable polinomialmente. Estas instancias son las instancias de ***NO*** de Π^c, y entonces esta misma evidencia sirve para chequear que son instancias de ***NO***. ■

3. Problemas abiertos de Teoría de Complejidad

Con estas nuevas definiciones tenemos los siguientes problemas abiertos:

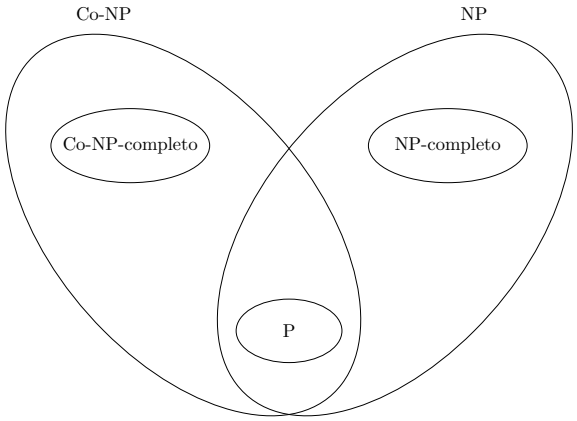
- ¿Es P = NP?
- ¿Es Co-NP = NP?
- ¿Es P = Co-NP ∩ NP?

La pregunta más relevante en el área es la primera, **¿Es P=NP?**. Éstos son tres mapas posibles para las clases de complejidad:



En la clase **NP-intermedio** están los problemas de **NP** que no son **P** ni **NP-completo**. Por supuesto no se sabe si hay algún problema en esta clase, ya que la exsistencia de este problema implicaría que **P** $\neq$ **NP**.

Ésta sería la situación si se probara que $P \neq NP$, $NP \neq Co-NP$, $P \neq Co-NP \cap NP$:



4. Extensión de un problema

Definición 4. El problema Π es una ***restricción*** de un problema $\bar{\Pi}$ si el dominio de Π está incluído en el de $\bar{\Pi}$.

Si Π es una restricción de $\bar{\Pi}$, se dice que $\bar{\Pi}$ es una ***extensión*** o ***generalización*** de Π.

Es intuitivo pensar que cuanto más general es el problema, más difícil es de resolver. Algunas veces, casos particulares (restricciones) de un problema **NP-completo** puede ser **P**. Pero no se puede dar la situación recíproca (ser el caso general más fácil que el caso particular).

Ejemplo 12.

- *3-SAT es una restricción de SAT. Ambos son problemas NP-completos.*
- *2-SAT es una restricción de SAT. 2-SAT es polinomial, mientras que SAT es NP-completo.*
- *COLOREO de grafos bipartitos es una restricción de COLOREO. Colorear un grafo bipartito es un problema polinomial, mientras COLOREO es NP-completo.*
- *CLIQUE de grafos planares es una restricción de CLIQUE. Encontrar una clique máxima de un grafo planar es un problema polinomial (porque sabemos que no puede tener a K_5 como subgrafo), mientras CLIQUE es NP-completo.*
- *CARTERO CHINO en grafos planares es una restricción de CARTERO CHINO. Ambos son problemas NP-completos [1].*

Formalmente podemos deducir que:

- Si $\bar{\Pi} \in \mathbf{P}$, entonces $\Pi \in \mathbf{P}$.
- Si $\bar{\Pi} \in \mathbf{NP}$, entonces $\Pi \in \mathbf{NP}$.
- Si $\Pi \in$ NP-Completo, entonces $\bar{\Pi} \in$ NP-Difícil.

5. Algoritmos Pseudopolinomiales

Definición 5. Un algoritmo para resolver un problema Π es ***pseudopolinomial*** si la complejidad del mismo es polinomial en función del valor de la entrada.

Ejemplo 13. *El problema de la mochila es NP-Completo, sin embargo, existe un algoritmo de complejidad $\mathcal{O}(nB)$ que lo resuelve, donde n es la cantidad de objetos y B el peso máximo que se puede cargar en la mochila.*

Una instancia de este problema es $c_1, \ldots, c_n, B$, por lo tanto su tamaño es $\mathcal{O}(nlogB)$. Por lo tanto $\mathcal{O}(nB)$ no es polinomial en el tamaño de la instancia, pero sí en el valor de la instancia.

Desde un punto de vista práctico, muchas veces los algoritmos pseudopolinomiales, a pesar de ser exponenciales, son aplicables.

6. Bibliografía recomendada

- M. Garey and D. Johnson, *Computers and intractability: a guide to the theory of NP- Completeness*, W. Freeman and Co., 1979.
- Capítulos 15 y 16 de C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications INC, 1998.

Referencias