TEORÍA DE ALGORITMOS 1
Clases de complejidad

Por: ING. VÍCTOR DANIEL PODBEREZSKI
vpodberezski@fi.uba.ar

1. Introducción

Llevamos ya un tiempo presentando diferentes problemas y buscando algoritmos para resolverlos. En algunos casos encontramos algunos que definimos como “buenos”. Estos resuelven cada instancia posible en tiempo polinomial en función al tamaño de la misma. Para un subconjunto de problemas presentamos diferentes soluciones algorítmicas que los resolvían. Entre estos pudimos observar diferentes cotas de complejidad temporal o especial. Para otro subconjunto de problemas - sin embargo - no pudimos proponer soluciones algorítmicas polinomiales en tiempo. Algunos los resolvimos en tiempo pseudo polinomial, exponencial o factorial.

Llegando a este momento es natural preguntarse a qué se debe esta diferencia. ¿Será que aún no investigamos lo suficiente acerca de la naturaleza de estos problemas? ¿Será que en un futuro podremos resolverlos, tal vez mediante una reducción polinomial y transformarlos en problemas manejables (o tratables)? Encontraremos algún algoritmo aún no descubierto? Tal vez ocurra lo contrario, que no podamos hallar una solución con las características buscadas porque esta no existe. Tal vez logremos demostrar esta ausencia.

Todas estas preguntas no son originales. La comunidad científica hace décadas formuló y comenzó a generar abundante estructura teórica para intentar responder las mismas. Presentaremos a continuación apenas la superficie de toda la temática.

Trabajaremos con un tipo concreto de problemas: los **problemas de decisión**. En estos se desea verificar si para una instancia se cumple cierta condición. La respuesta es acotada a dos valores posibles sí o no. Otros problemas que retornan un conjunto de respuestas más

complejas se los engloba dentro de los **problemas funcionales** (functional problem). Estos incluyen los problemas de optimización, de búsqueda y otros.

En muchos casos el enunciado de un problema funcional se puede vincular directamente con un problema de decisión y viceversa. Aunque en algunos casos esta relación no es posible o no es tan clara. Daremos algunos ejemplos en la siguiente tabla:

Nombre	**Tipo de instancia**	**Problema funcional**	**Problema de decisión**
Camino Eureleano	Dado un grafo G=(V,E) conexo y no dirigido	Construir un camino que recorra todos los ejes exactamente una vez.	¿Es posible construir un camino que recorra todos los ejes exactamente una vez?
Flujo máximo / Corte mínimo	Dado la red de flujo G=(V,E). Un par de nodos “s” y “t”	Deseamos obtener el flujo máximo entre los nodos “s” y “t”	¿Es posible establecer un flujo de al menos k entre “s” y “t”?
Camino mínimo	Dado un grafo G=(V,E) pesado y dirigido. Un nodo “o” origen y un nodo destino “d”.	Obtener el camino mínimo desde “o” a “d”	¿Es posible llegar de “o” a “d”
Cobertura de vértices	Dad un grafo G=(V,E)	Seleccionar la cantidad mínima de vértices tal que todo eje sea cubierto en su extremo por alguno de estos	¿Es posible seleccionar “k” o menos vértices para cubrir todos los ejes?
Problema de la mochila	Dado un conjunto de elementos con un peso y un valor Una mochila con capacidad determinada	Obtener el máximo valor ingresando un subconjunto de elementos en la mochila sin superar la capacidad de la misma.	¿Es posible ubicar un subconjunto de elementos en la mochila que no supere su capacidad logrando una ganancia mayor o igual a k?
Problema del viajante de comercio	Un grafo completo y pesado G=(V,E)	Obtener el ciclo hamiltoniano de menor costo	¿Es posible obtener el ciclo hamiltoniano de costo menor a “k”?

Algunos de estos problemas ya los estudiamos. Más adelante volveremos sobre estos y otros problemas clásicos en la bibliografía de las últimas décadas. Seleccionaremos problemas representativos dado que es imposible trabajar sobre todos.

La existencia de esta gran cantidad de problemas y su necesidad de analizarlos y compararlos fue abordado agrupando conjuntos de problemas de complejidad computacional similar. A estos conjuntos los denominaremos **clases de complejidad**. Una clase de complejidad particular incluye a todos los problemas que pueden resolverse con cierta medida de complejidad en función de una dimensión de un problema. Las dimensiones generalmente corresponden al tiempo o espacio requerido para resolverlos. Comenzaremos trabajaremos con la primera dimensión.

En las siguientes secciones definiremos diferentes clases de complejidad y la relación entre ellos. Ejemplificaremos utilizando diferentes problemas que se encuentran en ellas.

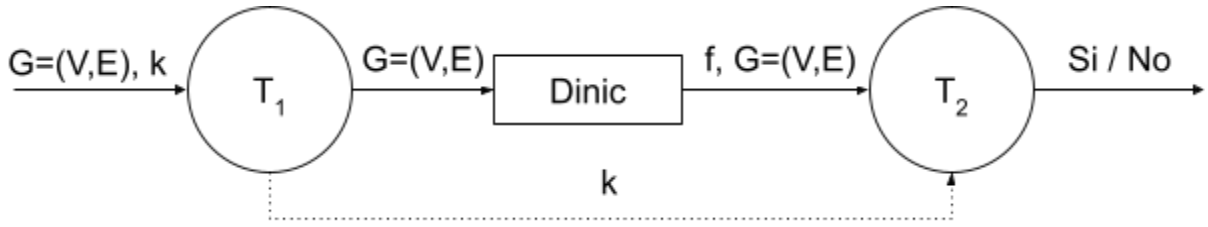
2. Problemas clase “P”

Se conoce a la **clase de complejidad “P”** al conjunto de todos los problemas de decisión para los que existe un algoritmo “A” que resuelve cualquier instancia I del mismo en tiempo polinomial. Utilizando las definiciones previas (Tesis Cobham–Edmonds) podemos afirmar entonces que un problema de decisión manejable pertenece a la clase “P”.

Tomemos el problema de decisión acerca de la existencia de un camino euleriano en un grafo G=(V,E) conexo y no dirigido. Lo llamaremos de forma resumida “CE”. Responderemos de forma afirmativa la evaluación de una instancia del problema simplemente verificando el teorema de Euler: Existe un camino euleriano en un grafo conexo si solo dos o cero de sus vértices tienen grado impar. El grado de un vértice se calcula sumando la cantidad de ejes que inciden en él. Podemos responder entonces este problema simplemente contando el grado de cada vértice. En caso de no cumplirse la condición, la respuesta es negativa. Dependiendo la forma de representar el grafo será la complejidad del procedimiento. Si se utiliza una matriz de adyacencia el proceso se puede realizar en $O(V^2)$, en cambio con una lista de adyacencias la complejidad corresponde a O(E+V) o O(E) simplificando. En definitiva, como existe un algoritmo bueno que resuelve el

problema, podemos afirmar que “CE” ∈ P. Lease “CE” pertenece a la clase de complejidad P.

Muchos de los problemas vistos hasta el momento en sus versiones de decisión pertenecen a la clase “P”. Por ejemplo, llamamos “FM” al problema de decisión de Flujo máximo en el cual queremos responder a la pregunta “¿Es posible establecer un flujo de al menos k?. En este caso conocemos varios algoritmos buenos que resuelven el problema de flujo máximo de optimización: Dinic, Edmonds-Karp, entre otros. Podemos realizar una reducción polinomial de un problema al otro. Partimos de la instancia del problema de decisión y realizamos la primera transformación de forma trivial. La instancia del problema de maximización contiene la red de flujo sin modificaciones. Esta será la entrada de la caja negra que resuelve el problema de optimización del problema de flujo máximo. La caja negra corresponde al algoritmo que resuelve en tiempo polinomial este último problema. Tomamos la salida de la caja negra que será una asignación de flujo. Obtenemos de esta asignación el valor del flujo de la red y lo comparamos con el valor k. Respondemos sí o no de acuerdo a si se cumple o no la condición consultada. Como ambas transformaciones son polinomiales y la caja negra se ejecuta en tiempo polinomial, todo el proceso es polinomial y nos permite resolver el problema de decisión de forma manejable. Por lo tanto podemos afirmar que “FM” ∈ P.



Reducción de problema de decisión de flujo máximo a a problema de optimización de flujo máximo

Para otro conjunto de problemas que analizamos conseguimos algoritmos no polinomiales. No logramos algoritmos buenos ni tampoco reducirlos polinomialmente a problemas que demostramos que pertenecen a “P”. Ejemplos de estos problemas son el problema de la mochila o del viajante de comercio. Por lo tanto no podemos afirmar que pertenecen a esta clase. Pero tampoco podemos afirmar que no pertenece.

Antes de proseguir vale la pena detenerse en unos puntos que suelen generar confusiones. Es incorrecto hablar de un algoritmo bueno como perteneciente a “P”. Los algoritmos no pertenecen a clases de complejidad. Por otra parte, es tentador tomar un problema funcional que conocemos tratable y afirmar que pertenece a la clase “P”. Pero estaremos violando la definición: Solo los problemas de decisión pueden pertenecer a este conjunto. Para los problemas funcionales existe una definición de clase de complejidad particular. Se conoce como **clase FP** al conjunto de problemas funcionales que pueden ser resueltos en tiempo polinomial. Su estudio y caracterización excede el alcance de este escrito.

3. Problemas clase “NP”

Llamaremos **certificador** a un algoritmo “B” que recibe por parámetro una instancia I de un determinado problema de decisión y una evidencia “e” de que la solución es “Si”. La evidencia o “certificado” (como es también llamado) corresponde a las indicaciones que se deben seguir para constatar que la instancia cumple con la condición solicitada por el problema. El certificador verifica que realmente la evidencia es suficiente para responder como afirmativo a la pregunta del problema. Retorna “Si” o “No” como respuesta.

Se conoce a la **clase de complejidad “NP”** al conjunto de todos los problemas de decisión para los que existe un certificador “B” que en tiempo polinomial puede verificar cualquier par de instancia y evidencia. Al evaluar el tiempo lo realizamos en función del tamaño tanto de la instancia como de la evidencia.

Tomemos el problema de decisión acerca de la existencia de un camino euleriano en un grafo G=(V,E) conexo y no dirigido. Para una instancia I determinada podemos usar como evidencia “e” una secuencia de ejes que conforman el camino euleriano en el grafo. Podemos expresar e como e={e_1, e_2, … e_x}. El algoritmo certificador debe verificar:

- Que el número de ejes en la lista sea igual a |E|
- Que cada uno de los ejes en “e” exista en el grafo G
- Que cada uno de los ejes en “e” aparezca solo una vez
- Que cada par de ejes contiguos en e, e_i=(a,b) y e_{i+1}=(c,d) cumplan con que el nodo b corresponda al nodo c (b=c).

Basta que algunas de las condiciones no se cumplan para afirmar que el certificado no es válido. En consonancia, si todos los puntos se verifican podemos afirmar que la respuesta a la existencia de un camino euleriano para la instancia I es afirmativa. El certificador retorna en el primer caso “No” y en el segundo “Si”. Se puede observar que cada uno de los procesos se puede realizar en forma polinomial. La complejidad total depende de cómo se almacena el grafo. Pero realizado de forma adecuada todo el proceso se puede realizar en O(E).

Se conoce como problema del viajante de comercio “VC” a aquel donde partiendo de un grafo completo y pesado buscamos un ciclo hamiltoniano. En su versión de optimización buscamos el menor costo posible. En cambio en su versión de decisión pretendemos saber si existe un ciclo hamiltoniano con un costo menor a un valor k. El costo de ciclo se calcula sumando el peso de cada uno de los ejes que utiliza. Un ciclo en un grafo G=(V,E) es hamiltoniano si recorre todos sus nodos una y solo una vez excepto por el nodo de inicio que también corresponde al del final.

Este problema “VC” pertenece a NP. Para demostrarlo, probaremos que existe un algoritmo bueno que lo certifica eficientemente. Como evidencia “e” tomaremos una posible secuencia de nodos por los que se conforma el ciclo. Lo podemos representar como e={v_1, v_2, … , v_x}. El certificador recibe la instancia I, “e” y en tiempo polinomial controla que:

- Que el número de vértices en la lista sea igual a |V|+1
- Que cada uno de los vértices en “e” exista en el grafo G
- Que cada uno de los vértices en “e” (excluyendo al primero y último) aparezcan solo una vez
- Que el primer y último de los vértices sea el mismo.
- Que la suma de los pesos de los ejes que une a cada par de vértices v_i y v_{i+1} sea menor al valor k.

Todos estos procesos se pueden realizar utilizando las estructuras adecuadas en O(V). Por lo tanto se ejecuta en tiempo polinomial en función de “e” y la instancia I. Basta que alguna de las condiciones no se cumpla para que el certificador retorne “No”. Si el conjunto se cumple la respuesta será “Si”.

Nuevamente se hace hincapié en que los problemas pertenecientes a esta clase corresponden a problemas de decisión. Para los problemas funcionales existe otra definición similar conocida como FNP. Una problema funcional pertenece a la **clase FNP** si dado una instancia y una solución posible se puede verificar en tiempo polinomial que esta última corresponde a la esperada por resolver la instancia. No profundizaremos más en este concepto.

4. Comparación de complejidad entre problemas

Hasta el momento tenemos herramientas para comparar dos algoritmos diferentes y determinar cual es de mayor complejidad que el otro. Corresponde a un análisis que podemos hacer utilizando la dimensión temporal o espacial. Ahora comparar la complejidad entre dos problemas es algo más complejo. No podemos simplemente tomar el mejor algoritmo conocido hasta el momento de cada uno y compararlos. No sería una medida fiable ni completa. Continuamente se conocen nuevos algoritmos. De muchos algoritmos aún no se conoce sus límites inferiores de complejidad. De todas formas tampoco es necesario una granularidad tan grande al analizar los problemas.

Nos interesará conocer que problemas de decisión son “polinomialmente” más complejos que otros. Para eso nos servirá como herramienta un procedimiento que estudiamos anteriormente: la reducción polinomial.

Cuando realizamos una reducción polinomial de un problema A a un problema B estamos afirmando que el problema A es a lo sumo polinomialmente tan complejo que el problema B. Es decir que A puede ser menos o igual de complejo que B, pero nunca más complejo. Habitualmente expresamos la reducción polinomial de A a B como: $\mathbf{A \leq_P B}$. Y la misma expresión nos está mostrando la desigualdad.

Supongamos que no conocemos cómo resolver un determinado problema de decisión X. Aunque tenemos la capacidad de reducirlo polinomialmente a otro problema Y. Además demostramos que el problema Y pertenece a la clase P. Todo el proceso de resolución del problema X se encuentra realizado en orden polinomial. Por lo tanto puedo afirmar que el problema X al poder resolverse en tiempo polinomial, pertenece también a la clase de complejidad P.

Analicemos una situación similar. Queremos resolver el problema de decisión X y desconocemos cómo hacerlo algorítmicamente. Logramos reducirlo polinomialmente a un problema Y. Hasta el momento ninguno de los algoritmos conocidos para resolver Y es polinomial. Por lo tanto, para resolver todo el problema X mediante la reducción tenemos en la caja negra un proceso no polinomial. ¿Qué podemos decir acerca del problema X? Sin posibilidad de equivocarnos podemos afirmar que podría no pertenecer a P, pero también podría formar parte. No es muy útil esa afirmación. Aunque es también esperable: parece razonable suponer que siempre es posible complejizar innecesariamente la forma de resolver un problema. Regresando a la situación, algo que sí podemos afirmar es que X será a lo sumo tan complicado de resolver polinomialmente que Y. Esta información será útil si en algún momento se logra demostrar la pertenencia (o no) de X o Y a cierta clase de complejidad.

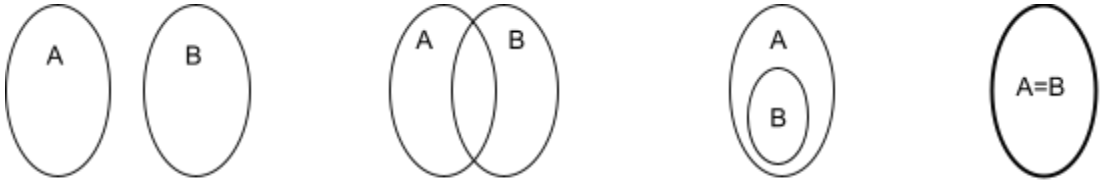
Existe una propiedad que se deriva de la aplicación de la reducción polinomial: la **transitividad**. Dado un problema A que se puede reducir polinomialmente a un problema B. Es decir $\mathbf{A} \leq_P \mathbf{B}$. Y dado que el problema B se puede reducir polinomialmente a un problema C: $\mathbf{B} \leq_P \mathbf{C}$. Entonces se puede afirmar que a A se puede reducir polinomialmente a C: $\mathbf{A} \leq_P \mathbf{C}$. Esta propiedad se utiliza para encadenar problemas y poder acotar o establecer complejidades basándonos en el conocimiento de la complejidad de alguna de las complejidades de los problemas analizados. Por ejemplo, si la complejidad del problema al que luego de una serie de reducciones polinomiales se termina llegando pertenece a la clase P, entonces el resto de los problemas anteriores también pertenece a esta clase de complejidad o menor. No definiremos clases de menor complejidad que P, pero se han definido y pueden ser estudiadas con más profundidad en diversa bibliografía. Un libro clásico sobre el tema es el “Computational Complexity” de Papadimitriou[1]

Corresponde a un error común del novicio considerar que si un problema no pertenece a P, entonces pertenece a NP. Surge por una asociación casi involuntaria: “si P implica que el problema se puede resolver en tiempo **P**olinomial, entonces NP significa que se puede resolver en tiempo **n**o **p**olinomial”. Siguiendo esa línea de pensamiento “si un problema no pertenece a “P”, tiene que pertenecer a “NP”. Sin embargo, basta prestar atención a las

[1] Papadimitriou, Christos H. Computational Complexity, 1994, Addison Wesley

definiciones de estos conceptos para ver que no es lo que significan. Pero, entonces ¿existe una relación entre las clases P y NP?

Para analizar dos conjuntos y su relación nos interesa conocer cómo estos se intersectan. Es decir si existen algunos problemas que pertenecen a ambos conjuntos o no. Si la respuesta es afirmativa podría ocurrir que un conjunto contenga completamente al otro, parte o que ambos conjuntos sean equivalentes. En la imagen siguiente se muestran visualmente todos estos casos.



Opción 1 Opción 2 Opción 3 Opción 4

Comencemos por la primera opción. Podemos descartarla rápidamente si encontramos un ejemplo de algún problema que pertenece a P y NP. Nos centraremos en un problema que analizamos previamente: el problema de decisión de flujo máximo. Al presentar la clase P se utilizó como ejemplo este mismo problema. Por lo tanto podemos afirmar que pertenece a esta clase. Por otro lado, la pertenencia a la clase NP de este problema también la podemos asegurar. Supongamos que recibimos para una instancia de este problema como evidencia una determinada asignación de flujo de un determinado conjunto de ejes del grafo. Se Puede verificar que:

- Los ejes presentados en la evidencia existan en el grafo y no se repitan.
- Para cada nodo v del grafo E-{s,t} la suma del flujo entrante al nodo es igual al saliente (conservación de flujo)
- Para cada eje de la evidencia que su asignación de flujo no supere su capacidad (restricción de capacidad)
- La suma del flujo de los ejes que salen de la fuente en la evidencia tengan por valor un número mayor o igual al k indicado en el problema.

Como todos estos pasos se pueden hacer en tiempo polinomial en función a la instancia y la evidencia, podemos afirmar que el problema pertenece a NP. Podemos con este análisis

desestimar la opción 1. Pero aún podemos hacer más. Demostraremos que la opción 2 también es inválida.

Procederemos a demostrar que todo problema de decisión que se puede resolver en tiempo polinomial, también se puede verificar en esa misma escala de complejidad temporal. Tomemos un problema que se puede resolver en tiempo polinomial. Es decir que pertenece a la clase P. Al momento de verificar una instancia de este problema solicitamos como parámetro del certificador una cierta evidencia. No se establece la naturaleza de esta evidencia. Muchas veces utilizamos una configuración determinada de la instancia o una serie de decisiones a tomar que resulte útil para lograr la verificación. Tampoco se establece cómo se obtiene esa evidencia. No es relevante para la tarea. Lo importante es que le sirva al verificador para emitir el certificado de forma adecuada. Para que el problema pertenezca a NP, el certificador puede realizar cualquier proceso siempre que este se mantenga en la complejidad polinomial. Teniendo en cuenta eso, entonces nuestro certificado será vacío. No necesitamos nada más. Dentro del certificador llamaremos al algoritmo para solucionar el problema. Sabemos que se ejecuta en tiempo polinomial y el resultado corresponde a toda la evidencia de que la instancia puede o no responder de forma positiva la pregunta del problema. Más concretamente sabremos la misma respuesta. Todo el proceso es polinomial. Por lo tanto si un problema pertenece a P también pertenece a NP.

Queda, para dilucidar la pregunta que realizamos, determinar si la opción 3 o la opción 4 es la correcta. En el primer caso podremos afirmar que P<>NP. En el siguiente caso P=NP. Sin embargo no tenemos aún suficiente conocimiento para responder. Continuaremos ahondando sobre el tema y regresaremos a la cuestión durante la próxima sección.

5. Problemas clase “NP-Hard” y “NP-Complete”

Con motivo de dilucidar si P y NP son equivalentes o no, se comenzó a trabajar en intentar encontrar problemas particulares que den respuesta a la cuestión. Un avance importante ocurrió de la mano de dos científicos en la década del 70. El primero de origen

estadounidense, Stephen Cook[2] y el segundo sovietico, Leonid Anatólievich Levin[3]. Ellos presentaron dos nuevas clases de complejidad: NP-Hard y NP-Complete.

Un problema pertenece a **NP-Hard** (también escrito como NP-H) si todo problema que pertenece a NP se puede reducir polinomialmente a este. Expresado matemáticamente $X \in NP\text{-}H \Leftrightarrow \forall\ Y \in NP,\ Y \leq_P X$. Lo que postula esta definición es que un problema para pertenecer a este conjunto tiene que ser igual o más difícil que todos los problemas que pertenecen a la clase de complejidad NP. Por eso el nombre de este conjunto "Hard" en su traducción al español como "difícil".

Se debe observar que la definición en este caso no restringe el tipo de problema a uno de decisión. Se define que un problema funcional pertenece a NP-Hard si su problema de decisión correspondiente es NP-Hard.

Por otro lado se define la clase de complejidad **NP-Complete** (o NP-C o NP-Completo) a aquellos problemas que pertenecen a NP y también forman parte de NP-Hard. Es decir a aquellos problemas de decisión que perteneciendo al conjunto de NP corresponden a los más difíciles entre ellos. Matemáticamente lo podemos expresar como $X \in NP\text{-}C \Leftrightarrow X \in NP$ y $X \in NP\text{-}H$.

La existencia de un problema más difícil que otros en NP puede servir como herramienta para demostrar que P es igual o diferente a NP. Si al menos un problema de los más difíciles (en NP-C) se puede resolver en tiempo polinomial, entonces todos los problemas en NP se podrían resolver en tiempo polinomial. Esto se explica por la definición de transitividad de la reducción polinomial y la definición misma de NP-H. Podemos en un encadenamiento de razonamientos demostrarlo: $X \in NP\text{-}C, Y \in P$, si $X \leq_P Y \Rightarrow \forall\ Z \in NP, Z \leq_P X \leq_P Y \Rightarrow \forall\ Z \in NP\ Z \leq_P Y \Rightarrow \forall\ Z \in NP\ Z, Z \in P \Rightarrow P=NP$.

Por otro lado, alcanza con demostrar que al menos un problema en NP no se puede resolver en tiempo polinomial para definir que P<>NP. En ambos casos, los problemas más difíciles en NP son los candidatos para trabajar e intentar llegar a estas demostraciones.

[2] Stephen A. Cook, 1971, The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing (STOC '71). Association for Computing Machinery, New York, NY, USA, 151–158.
[3] L. Levin. Universal Search Problems. 1973, Problemy Peredachi Informatsii, 9:3, pp. 265–266.

Esta definición de clases de complejidad serían incompletas a no ser que se demuestra que existen problemas que pertenecen a estas. Cook y Levin lo realizaron de forma independiente. Hoy día se conoce como **Teorema de Cook-Levin** a aquel que afirma que el problema de satisfacibilidad booleana (SAT) es NP-Complete. De esa forma este problema fue el primer problema en pertenecer a NP-C. No nos detendremos en este momento a demostrar el teorema. Nos hacen falta varios conocimientos importantes antes de poder realizarlo.

Una vez demostrado que el conjunto NP-C no estaba vacío comenzó un camino de evidenciar que no estaba solo. Quien sentó precedente de cómo realizarlo fue Richard Karp en su histórico paper "Reducibility Among Combinatorial Problems"[4]. En este presenta 21 problemas pertenecientes a NP-C. Para hacerlo utiliza la herramienta que ya presentamos: la reducción polinomial. Veremos algunos de estos problemas más adelante.

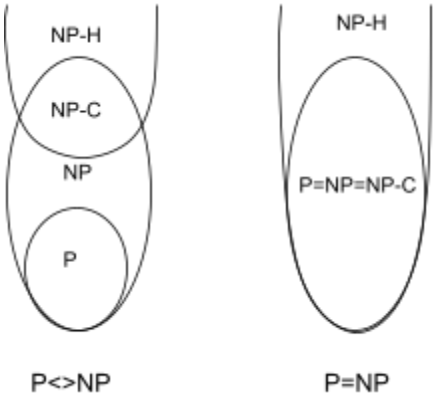
En las siguientes décadas se han demostrado que cientos de problemas pertenecen a NP-Complete. Sin embargo, no han existido avances significativos que hayan logrado demostrar cual es la relación entre las clases P y NP. Corresponde a un problema abierto y de gran interés de la comunidad. Esta atención ha dado lugar a competencias. Por dar un ejemplo existe un premio de un millón de dólares propuesto por el "Clay Mathematics Institute" para quien pueda resolver este problema. Junto con otros 6 problemas conforman los "The Millennium Prize Problems"[5]. Además hay debates continuos e incluso encuestas[6] entre especialistas donde se intenta conocer entre otras cosas: si consideran que son o no iguales. Incluso si sería posible demostrarlo en un lapso de tiempo determinado.

¿Por qué tanto interés en resolver esta cuestión? Más allá del reto matemático, muchos problemas que se han demostrado en NP-Complete resultan de aplicación en la vida cotidiana. Por lo tanto, encontrar una manera eficiente de resolverlos es de interés. Demostrar que P=NP presenta una excelente noticia para quienes los enfrentan. Al mismo

[4] Karp, R.M. Reducibility among Combinatorial Problems. 1972, In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series. Springer, Boston, MA.
[5] https://www.claymath.org/millennium-problems/millennium-prize-problems
[6] Lane A. Hemaspaandra. SIGACT news complexity theory column 36. 2002. SIGACT News 33, 2 (Junio 2002), 34–47.

tiempo, demostrar que son diferentes, permitirá redireccionar esfuerzos. Pero no solo eso, muchas aplicaciones prácticas se construyen bajo el supuesto que P y NP son diferentes. El ejemplo más claro de esto son los sistemas criptográficos. Estos problemas esperan que la complejidad de desencriptar un texto con la clave (o llave) sea "fácil" y que hacerlo sin ella sea "difícil". Si P es igual a NP entonces muchos de los métodos más utilizados se volverían inseguros.

Antes de finalizar presentaremos el ecosistema de clases de complejidad presentadas hasta el momento y su relación. Se muestran las dos posibilidades. En el primer caso si P y NP son diferentes y luego si P y NP son iguales. En el segundo caso la mayoría de los conjuntos se unifican.



A continuación nos tomaremos unos instantes para explicar en qué consiste el problema SAT y lo utilizaremos como puntapié para mostrar la existencia de otros problemas que pertenecen a NP-C.

6. SAT El primer problema en NP-C y una variante

Se conoce como el problema de **satisfacibilidad booleana** o **SAT** al problema de determinar si dada una fórmula de lógica proposicional, expresada en forma normal conjuntiva (CNF), la misma puede ser satisfecha.

La fórmula, también conocida como expresión booleana, está conformada de variables booleanas que permiten como valor verdadero (TRUE) o falso (FALSE). Estas operan entre sí utilizando los conectores lógicos AND (conjunción), OR (disyunción) y NOT (negación).

Una fórmula en forma normal conjuntiva está construida en base a cláusulas. Una cláusula es una unidad de la fórmula encerrada por paréntesis. Dentro de una cláusula hay una disyunción de literales. Un literal es una variable o la negación de esta (la variable con un NOT). El conjunto de cláusulas se une entre sí mediante una conjunción.

Se entiende como satisfacción de una fórmula a la asignación de valores a las variables y que la evaluación de la misma de cómo resultado el valor verdadero (TRUE). Esto implica que al menos uno de los literales en cada cláusula tiene que tener el valor verdadero

(TRUE). Pueden existir diferentes formas de satisfacer una expresión. Pero para el problema de decisión únicamente nos interesa saber si es posible o no hacerlo.

En resumidas cuentas el problema se puede enunciar como:

Problema SAT
Dada una expresión booleana ϕ compuesta de n variables y k cláusulas, queremos saber si la misma se puede satisfacer.

El siguiente ejemplo corresponde una fórmula booleana expresada en CNF: $\phi=(\neg v_1) \wedge (v_2 \vee v_3) \wedge (v_1 \vee \neg v_3) \wedge (v_1 \vee v_2 \vee v_3)$*. Se compone de 3 variables y 4 cláusulas. El símbolo "¬" corresponde a la negación y significa negar el valor de la variable. El símbolo "∨" corresponde a la disyunción y se aplica a dos variables. El símbolo "∨" corresponde a la disyunción y se aplica a dos cláusulas. Podemos asignar valor a cada variable y evaluar la expresión. Por ejemplo* v_1*=true,* v_2*=true,* v_3*=true. La cláusula primera se evalúa* $(\neg v_1)=(\neg true)=false$*. La segunda cláusula* $(v_2 \vee v_3)=(true \vee true)=true$*. La tercera cláusula* $(v_1 \vee \neg v_3)=(true \vee \neg true)=(true \vee false)=true$*. Finalmente la última cláusula* $(v_1 \vee v_2 \vee v_3)=(true \vee true \vee true)=true$*. unificando las cláusulas nos queda finalmente en false la función evaluada. Se ve que todas excepto la primera se evalúan en true. pero basta para que una sea false para que la conjunción de por resultado false.*

Esta instancia se puede satisfacer por ejemplo aplicando las variables v_1*=false,* v_2*=true,* v_3*=false. De esta forma la cláusula primera la activa* v_1*, la segunda la activa* v_2*, la tercera* v_3 *y la cuarta* v_2*. Esta asignación es la evidencia necesaria para poder responder que esta instancia se puede satisfacer.*

El problema SAT es fácilmente demostrable que pertenece a NP. La evidencia corresponderá a una determinada asignación de valor de las variables. El algoritmo certificador lo utilizará como entrada junto a la instancia. Cada cláusula puede puede tener n literales (no tiene sentido que la misma variable este negada y sin negar en la misma cláusula). En un caso extremo tiene a todas sus variables y sus negadas. Cada evaluación y operación se puede hacer en O(1). Al tener k cláusulas el proceso general será O(nk). Corresponde a una complejidad polinomial y por lo tanto el certificador es eficiente y el problema pertenece a NP.

No se conoce un algoritmo eficiente que resuelve el problema en tiempo polinomial. En el peor de los casos, al resolverlo por fuerza bruta, se deben probar todas las combinaciones de las n variables. Se cuenta con 2^n posibles configuraciones a evaluar en O(nk) lo que nos deja una complejidad final de $O(2^n nk)$.

Este problema fue probado como perteneciente a NP-Complete por Cook y Levin. En la demostración probaron que todo problema perteneciente a NP puede reducirse polinomialmente a este. Si queremos probar que otro problema pertenece a NP-Complete podemos hacer el mismo procedimiento demostrativo. Lo que resulta muchas veces complicado. Por suerte existe una alternativa utilizando la propiedad de transitividad de la reducción polinomial para hacerlo.

Sea un problema de decisión X que queremos demostrar que pertenece a NP-Complete. Primero debemos demostrar que pertenece a NP mediante un certificador polinomial. Luego para poder demostrar que pertenece a NP-Hard tomaremos un problema Y que sabemos que pertenece a NP-C. Reduciremos polinomialmente el problema Y al problema X: $Y \leq_P X$. Cómo Y pertenece a NP-C entonces $\forall\ Z \in NP,\ Z \leq_P Y$. Aplicando la transitividad de la reducción polinomial $\forall\ Z \in NP,\ Z \leq_P Y \leq_P X \Rightarrow \forall\ Z \in NP\ Z \leq_P X \Rightarrow X \in$ NP-H. y como $X \in$ NP-H y $X \in NP \Rightarrow X \in$ NP-C. En pocas palabras: Como el problema X es igual o más difícil que el problema que es igual o más difícil que el resto de los problemas de NP, entonces el problema X tiene que ser igual o más difícil que el resto de los problemas de NP.

Este procedimiento lo utilizaremos a continuación para mostrar otros problemas como parte de la clase NP-Complete. Comenzaremos con algunas variantes de SAT

Un problema que nos resultará de suma utilidad es la variante **3SAT**. Corresponde a un conjunto de todos los problemas de SAT: aquellos que por cláusula tienen 3 literales:

Problema 3SAT
Dada una expresión booleana ϕ compuesta de n variables y k cláusulas, donde cada cláusula tiene 3 literales. queremos saber si la misma se puede satisfacer.

Uno podría suponer que al ser un subconjunto de un problema NP-C este debería también serlo. Aunque esto sería un error. Suponer que corresponde al mismo problema es

implícitamente realizar una reducción polinomial del problema a mayor. Aqui el razonamiento erróneo completo “Como 3SAT es un subconjunto de SAT tiene la misma complejidad que SAT”. Proceso de pensamiento que puede continuar como “Por lo tanto para resolver 3SAT puedo verlo simplemente como una instancia de SAT”. Este pensamiento esconde una reducción polinomial de 3SAT a SAT donde las transformaciones son O(1) al mantener la misma instancia entre problemas. Esta reducción se expresa como $3SAT \leq_P SAT$ y claramente no estamos logrando demostrar que 3SAT es tan difícil a lo sumo que SAT sino totalmente lo contrario. El proceso correcto es la reducción opuesta: demostrar que para cualquier instancia de SAT esta se puede reducir polinomialmente a una instancia de 3SAT. Veamos como esto es posible.

Partamos de una instancia genérica de SAT. Cada cláusula puede tener entre 1 y “n” literales. Si tiene 3 la dejamos como esta. Si tiene 1 o 2 realizaremos un proceso para transformarlas en una cláusula con 3. Para eso tendremos que agregarle variables ficticias. Debemos asegurarnos que esas variables no sean relevantes a la hora de satisfacer la cláusula final (pero sí para mantener las restricciones de las variables existentes en las cláusulas de la instancia del problema original). Veamos cómo realizar este procedimiento.

Para cada cláusula de ϕ de la instancia de SAT

- Si tiene dos literales ($l_1 \vee l_2$) que contienen respectivamente variables con o sin negar. Agregaremos una nueva variable α y su negada $\neg\alpha$ de la siguiente forma: (($l_1 \vee l_2$) $\wedge$ ($\alpha \vee \neg\alpha$)) = ($l_1 \vee l_2 \vee \alpha$)$\wedge$ ($l_1 \vee l_2 \vee \neg\alpha$). Pasamos a dos cláusulas de 3 instancias.
- Si tiene un literal (l_1). Agregamos una nueva variable β y su negada $\neg\beta$ y nos quedarán dos cláusulas de la siguiente forma ($l_1 \vee \beta$)$\wedge$ ($l_1 \vee \neg\beta$). Para cada una de las nuevas cláusulas realizaremos el procedimiento de pasaje de 2 a 3 literales por cláusula. Quedan 4 cláusulas totales de 3 literales.
- Si tiene 4 a n literales ($l_1 \vee l_2 \vee l_3 \vee \ldots \vee l_n$). Agregamos una nueva variable γ y su negada $\neg\gamma$ y podremos separar la cláusula en dos partes. La primer parte mantiene los primeros dos literales y γ y la segunda el resto con $\neg\gamma$. Quedará de la siguiente manera ($l_1 \vee l_2 \vee \gamma$) $\wedge$ ($\neg\gamma \vee l_3 \vee \ldots \vee l_n$). La primera cláusula al tener 3 literales no se modifica. A la segunda le podemos aplicar recursivamente las mismas 3 reglas presentadas hasta quedar únicamente con cláusulas de 3 literales. Esta separación

donde hay más de 3 literales en la cláusula se puede realizar un máximo de n-2 veces por cláusula.

Todo el proceso tiene una complejidad de O(nk) y por lo tanto polinomial. El resultado es una nueva instancia de 3sat que tiene más variables que el original. Esta nueva instancia de 3SAT se puede satisfacer únicamente si la instancia original SAT se puede satisfacer. Además si encontramos una evidencia correspondiente a la asignación de variables de la instancia de 3SAT esa misma nos sirve como evidencia de la instancia de SAT original quitando las variables adicionadas.

El siguiente ejemplo corresponde una fórmula booleana expresada en CNF: $\phi=(\neg v_1) \wedge (v_1 \vee v_2 \vee \neg v_4 \vee \neg v_5) \wedge (v_1 \vee \neg v_2 \vee \neg v_3 \vee v_4 \vee v_5)$. Para reducirlo a una instancia del problema 3SAT realizaremos la transformación de cada una de las cláusulas. Comenzamos con $(\neg v_1)$. En este caso debemos llevarlo a cláusulas de 3 elementos. Aplicamos la segunda y luego la primera regla. Quedará de la siguiente manera $(\neg v_1 \vee \alpha \vee \beta) \wedge (\neg v_1 \vee \alpha \vee \neg\beta) \wedge (\neg v_1 \vee \neg\alpha \vee \beta) \wedge (\neg v_1 \vee \neg\alpha \vee \neg\beta)$. La segunda cláusula $(v_1 \vee v_2 \vee \neg v_4 \vee \neg v_5)$ requiere separar en cláusulas más pequeñas. En este caso tomamos la los primeros dos literales por un lado y el segundo por el otro quedando de la siguiente manera $(v_1 \vee v_2 \vee \gamma) \wedge (\neg\gamma \vee \neg v_4 \vee \neg v_5)$. Logramos tener todas cláusulas de 3 literales. Finalmente transformaremos $(v_1 \vee \neg v_2 \vee \neg v_3 \vee v_4 \vee v_5)$. Primero aplicamos la tercera transformación y nos quedará una cláusula de 3 literales por un lado y otra de 4. En esta última volvemos a aplicar la separación y nos quedarán 2 cláusulas de 3. Quedará expresado como $(v_1 \vee \neg v_2 \vee \ ) \wedge (\neg \ \vee \neg v_3 \vee \varepsilon) \wedge (\neg\varepsilon \vee v_4 \vee v_5)$. Unificando todas las cláusulas la expresión de 3SAT construida será $(\neg v_1 \vee \alpha \vee \beta) \wedge (\neg v_1 \vee \alpha \vee \neg\beta) \wedge (\neg v_1 \vee \neg\alpha \vee \beta) \wedge (\neg v_1 \vee \neg\alpha \vee \neg\beta) \wedge (v_1 \vee v_2 \vee \gamma) \wedge (\neg\gamma \vee \neg v_4 \vee \neg v_5) \wedge (v_1 \vee \neg v_2 \vee \ ) \wedge (\neg \ \vee \neg v_3 \vee \varepsilon) \wedge (\neg\varepsilon \vee v_4 \vee v_5)$. Esta expresión de 9 cláusulas y 11 variables (5 previas y 6 agregadas) si puede ser satisfecha en el problema de 3SAT, entonces la expresión original en SAT también lo puede.

La siguiente asignación de variables satisface la expresión de 3SAT: v_1=false, v_2=false, v_3=false, v_4=false, v_5=false, α=false, β=false, γ=true, =false, ε=false. Seleccionando únicamente las variables originales podemos observar que estas satisfacen la expresión del problema SAT original.

Aplicando esta reducción polinomial para cualquier instancia del problema SAT se construye una instancia de 3SAT en la que existe una relación entre ambas y sus posibles

configuraciones de variables para satisfacerlas. Construyendo la tabla de verdad de cada una de las expresiones podemos hacer un análisis completo. Observando aquellas asignaciones que satisfacen a la expresión en 3SAT y seleccionando únicamente las variables correspondientes a la expresión original se puede ver que estas satisfacen la expresión en SAT. Esta relación permite asegurarnos que la reducción polinomial sea realmente aplicable para cualquier instancia de SAT.

Cuando realizamos una reducción polinomial para demostrar que cierto problema es NP-Hard este es el análisis que debemos realizar. Establecer la relación entre cada una de las instancias y cada una de las posibles soluciones y que siempre concuerden. Muchas veces al apresurarse se realizan reducciones que funcionan con un subconjunto de instancias y estas no prueban realmente lo que estamos buscando demostrar.

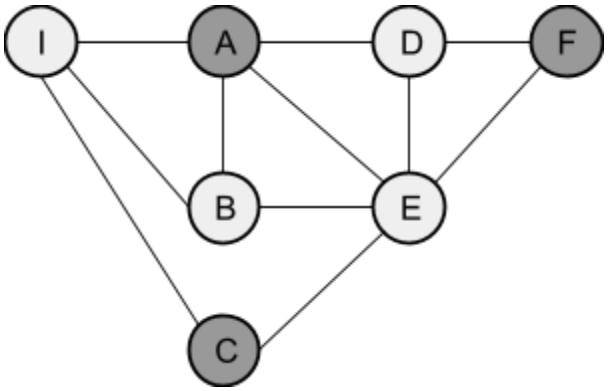
Ya demostramos que 3SAT pertenece a NP-Hard. Sumando la demostración de que es certificable (utilizando el mismo mecanismo utilizado para demostrar que SAT lo es) podemos afirmar que pertenece a NP y en conclusión a NP-Complete.

7. Conjunto independiente

En un grafo G=(V,E) no dirigido podemos observar que ciertos nodos tienen ejes que los conectan. Para un par de nodos dentro de un grafo diremos que son independientes si no hay un eje entre ellos. Formalmente $u,v \in V$ si $\nexists e=(u,v) \in E \Rightarrow$ u v son independientes.

Se conoce como el problema conjunto independiente a encontrar un subconjunto de nodos en el grafo que sean independientes entre sí. El tamaño del conjunto independiente corresponde a la cantidad de nodos en el subconjunto. En su problema de maximización corresponde a encontrar el mayor tamaño posible del conjunto independiente. En su versión de decisión se desea conocer si es posible conformar un conjunto independiente mayor o igual a un valor k. Formalizaremos el problema de decisión como:

Conjunto independiente (CI)
Dado un grafo G=(V,E) no dirigido, queremos saber si existe un conjunto independiente de nodos de tamaño mayor o igual a k.



En el grafo de la imagen se muestra el conjunto independiente {A,C,F}. Cada uno de estos nodos no están conectados entre ejes del grafo. Se puede observar también que es imposible agregar alguno de los otros nodos del grafo dado que el conjunto dejaría de ser independiente. Al estar compuesto por 3 nodos podemos afirmar que existe un posible conjunto independiente de tamaño 3 en el grafo. No es el único, otras posibilidades corresponden a los conjuntos {B,C,F} y {B,C,D}. Una vez observado la existencia de un conjunto de cierto tamaño, se puede construir conjuntos de tamaño menor simplemente quitando los nodos requeridos del conjunto. Se puede observar que en este ejemplo no existe un conjunto independiente de tamaño 4 o mayor.

No se conoce un algoritmo eficiente que pueda determinar la existencia o no de un conjunto independiente de tamaño dado para un grafo genérico. Probaremos que este problema corresponde a un problema NP-Complete.

Comenzamos definiendo un certificado y certificador polinomial que para cualquier instancia del problema nos permita afirmar o no que existe para una instancia determinada un conjunto independiente de tamaño k. Pediremos que la evidencia sea el conjunto “C” de nodos que forma un posible conjunto independiente de ese tamaño. El algoritmo certificador entonces:

- Verifica que la cantidad de nodos en C sea igual a k
- Controla que cada nodo de C exista realmente en el grafo y no aparezca más de una vez.
- Que cada par de nodos v,u en C no tenga un eje e=(v,u) en el grafo G

Si todo esto se cumple, existe un conjunto independiente de tamaño k. Todo este proceso se puede realizar en tiempo polinomial. En el conjunto C no puede existir más que k nodos y este valor de k no puede ser mayor al número de nodos del grafo. El proceso que lleva más tiempo es el tercer control. Tiene una complejidad de $O(k^2)$ o $O(V^2)$. Podría ser algo mayor si verificar la existencia o no de un eje en el grafo no es O(1). Sin embargo, en el

peor de los casos esto seguirá siendo polinomial en función de los parámetros del grafo. Por lo tanto, podemos entonces afirmar que nuestro certificador es polinomial y que el problema pertenece a NP.

Para demostrar que es NP-Hard utilizaremos un problema que previamente probamos que pertenece a NP-Complete (y por lo tanto pertenece a NP-Hard) y lo reduciremos polinomialmente al problema de conjunto independiente. Al momento solo probamos la pertenencia de SAT y 3SAT. Utilizaremos este último.

Para cualquier instancia de 3SAT compuesta por n variables y c cláusulas queremos transformarlo en una instancia de conjunto independiente: un grafo G=(V,E) y un valor k del tamaño del conjunto a lograr. Estas dos instancias de problemas diferentes queremos que estén íntimamente relacionados. Tal que la demostración de posibilidad de cumplimiento del segundo nos garantice la posibilidad de satisfacción del primero.

Partimos de una expresión booleana 3SAT y la transformamos de la siguiente manera:

Por cada cláusula (en total c):

- crearemos un nodo en el grafo G por cada literal (3 nodos).
- Creamos un eje por cada par de nodos creados (3 ejes)

Los nuevos nodos “representan” una variable negada o sin negar de la expresión original. Una misma variable puede aparecer sin negar en varios nodos. Igualmente puede aparecer negada en varios nodos.

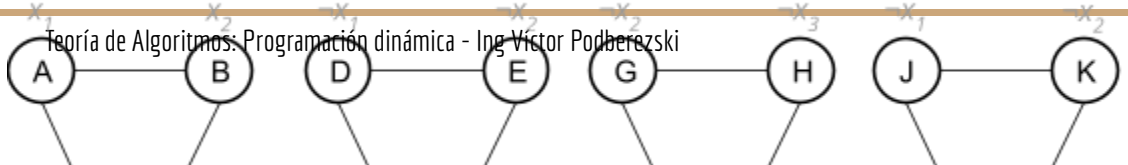
Realizaremos el siguiente proceso:

- Por cada par de nodos que representa una variable negada y la misma sin negar las unimos mediante un eje.

Finalmente:

- Definimos k=c (número de cláusulas c de la expresión a satisfacer).

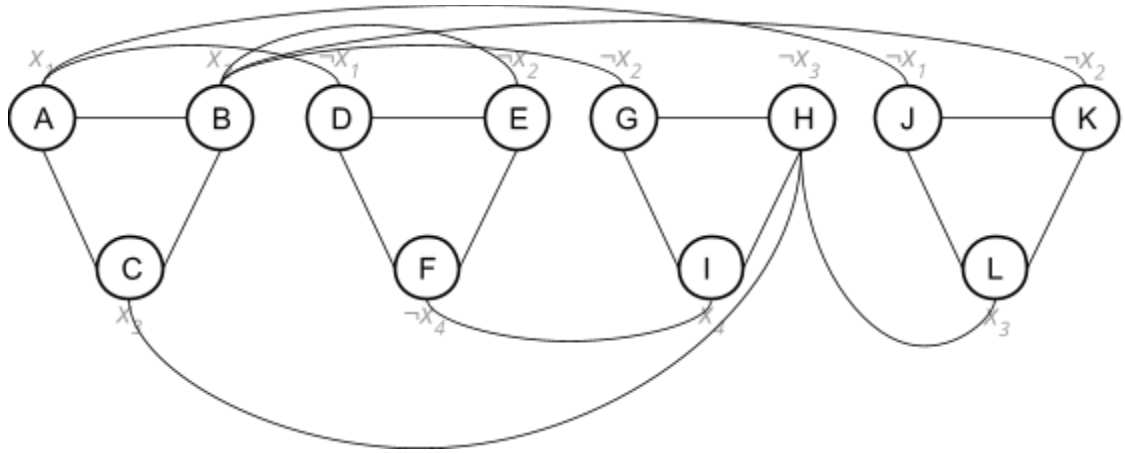
Veamos un caso práctico utilizando la instancia de 3SAT $\phi=(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$. Contiene 4 cláusulas y un total de 4 variables diferentes. Realizaremos la reducción polinomial paso a paso. Comenzamos creando los nodos y ejes por



cada cláusula. Se observa en la siguiente imagen que cada cláusula forma un subgrafo completo de 3 nodos. A cada nodo se le indica a modo aclaratorio a cuál literal de la cláusula representa.

El siguiente paso corresponde a unir con un eje a aquellos que contienen una determinada variable con aquellos que contienen la misma pero negada. En el ejemplo se generarán los siguientes ejes por las variables X_1: (A,D), (A,J) , X_2: (B,E), (B,G), (B,K) , X_3: (C,H), (H,L) , X_4: (F,I). Finalmente solicitamos para la instancia de Conjunto independiente que tenga tamaño k=4 o mayor.

Se puede observar que en un conjunto independiente solo se puede seleccionar 1 nodo por cada subgrafo de 3 nodos creados. Seleccionar un nodo en un subgrafo equivale a determinar el valor de la variable dentro de ese nodo para activar la cláusula de la fórmula 3SAT asociada. Además no se pueden seleccionar dos nodos que representan a una variable negada y sin negar al mismo tiempo. Por lo que no se puede usar la misma variable con diferente valor para activar cláusulas. La evidencia de conseguir un conjunto independiente de 4 nodos se puede transformar en la evidencia de satisfacer la fórmula 3SAT. Si alguna variable no queda



representada se le puede asignar cualquier valor y la fórmula seguirá siendo evaluada en forma afirmativa.

Con la transformación realizada, si existe un conjunto independiente en G de tamaño k, entonces la expresión 3SAT se puede satisfacer. Esta reducción se realiza en tiempo polinomial y funciona para cualquier instancia 3SAT. Por lo tanto 3SAT $\leq_P$ CI. y CI $\in$ NP-Hard. Como previamente probamos que CI $\in$ NP, entonces CI $\in$ NP-Complete.

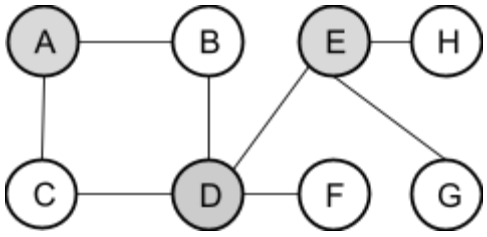
8. Cobertura de vértices

Continuando dentro de los problemas de decisión sobre grafos presentaremos la cobertura de vértices. Diremos que un vértice cubre un eje si corresponde a uno de los extremos de este último. Un subconjunto de vértices de un grafo lo cubre si para todo eje del grafo alguno de sus vértices se encuentra en el subconjunto.

El problema de optimización de cobertura de vértices intenta determinar la menor cantidad de vértices posibles para lograr la cobertura de un determinado grafo. El problema de decisión intenta responder la pregunta “Es posible utilizar “k” o menos vértices del grafo para cubrir el grafo?”. Formalmente:

Cobertura de vértices (CV)
Dado un grafo G=(V,E) no dirigido, queremos saber si existe un conjunto de nodos de tamaño menor o igual a “k” que cubra a todos sus ejes.

En el grafo de la imagen podemos seleccionar una cobertura de vértices incluyendo todos los vértices (tamaño 8). También se podría seleccionar los vértices {A,D,G,H} con tamaño 4 o una cobertura de tamaño 3 con {A,D,E}. No es posible conseguir una cobertura de tamaño 2.

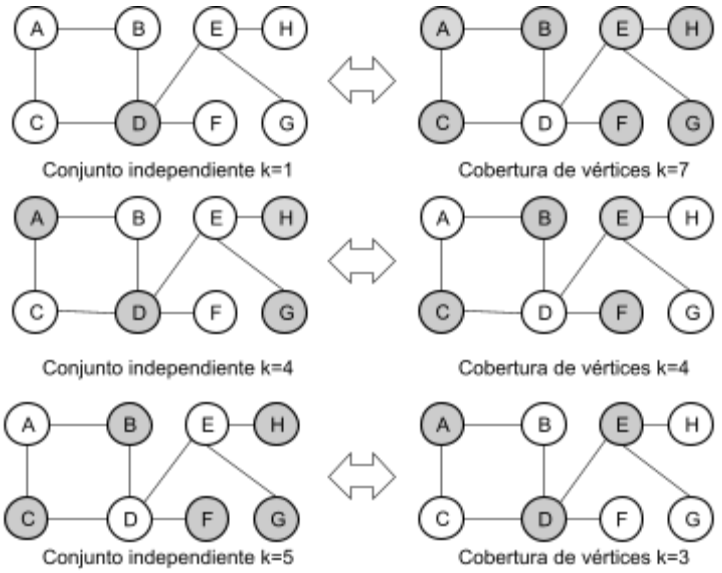


Es fácil ver que el problema de decisión pertenece a NP. El certificado corresponde a un subconjunto de vértices. Podemos verificar en tiempo polinomial que la cantidad de sea menor o igual al valor k. Que los vértices de la evidencia existan en el grafo y luego que realmente cubran a todos los ejes del grafo. Para demostrar que pertenece a NP.

Continuaremos demostrando su pertenencia a NP-Hard utilizando el problema de decisión de conjunto independiente. Veremos que entre un conjunto independiente y una cobertura de vértices existe una relación.

Supongamos que un grafo G=(V,E) contiene un conjunto independiente I ⊆ V. Llamaremos m al tamaño del conjunto independiente. Si analizamos los ejes del grafo en relación al

conjunto independiente, podremos ver que como mucho uno de sus extremos pertenecen a I. No hay ejes con sus dos vértices extremos dentro de I, de lo contrario no sería un conjunto independiente. Sí puede ocurrir que un eje no esté cubierto por ninguno de sus vértices en I. Llamamos O al complemento de I. Es decir al conjunto de vértices de E que no está en I. Al analizar los ejes del grafo en relación a O observamos que aquellos ejes que no estaban cubiertos pasarán a estar cubiertos por sus ambos extremos. De igual manera aquellos que estaban cubiertos por un vértice, quedarán cubiertos por el otro. Es imposible que un eje quede sin cobertura, puesto que como ya observamos en I no existen ejes con dos vértices cubiertos. Por lo tanto, O corresponde a una cobertura de vértices.



Esta relación es en ambos sentidos. Si tenemos un conjunto independiente de m vértices y tomamos el complemento tendremos una cobertura de vértices de |V|-m vértices. Similarmente si tenemos un conjunto independiente de x vértices y tomamos el complemento tendremos una cobertura de vértices de |V|-x vértices. En la imagen se muestran 3 ejemplos para un mismo grafo.

Habiendo probado la relación, podemos plantear la siguiente reducción polinomial para demostrar CI $\leq_P$ CV y por lo tanto que cobertura de vértices pertenece a NP-Hard.

Partimos de un grafo G=(V,E) y con un valor k de un problema CI y la transformamos de la siguiente manera:

- Mantenemos el mismo grafo G que usaremos para buscar una cobertura de vértices.
- Construiremos un nuevo parámetro c que será el tamaño de la cobertura de vértices buscado como |V|-k

Con la transformación realizada, si existe un cobertura de vértices en G de tamaño c, entonces existe un conjunto independiente de tamaño k (el complemento de los vértices). Esta reducción se realiza en tiempo constante (y por lo tanto polinomial) y funciona para cualquier instancia de conjunto independiente.

Ya demostramos que Cobertura de vértices pertenece a NP-Hard. Sumando la demostración de que pertenece a NP, podemos afirmar que pertenece a NP-Complete.

9. Clique

Se conoce como clique a un subconjunto de vértices C en un grafo G=(V,E) tal que para todo par de vértices en C existe un eje que los une. En estudio de redes sociales (tanto virtuales como reales) se suelen buscar cliques para estudiar comunidades. El tamaño del clique corresponde a la cantidad de vértices que lo conforma.

El problema de decisión de clique intenta responder la existencia o no de un clique de un determinado tamaño (o superior) en un determinado grafo. Su variante de optimización intenta encontrar el clique de mayor tamaño posible. Mostraremos que en su versión de decisión pertenece a NP-Complete. El enunciado del problema es el siguiente:

Clique
Dado un grafo G=(V,E) no dirigido, queremos saber si existe un clique de tamaño mayor o igual a “k”.

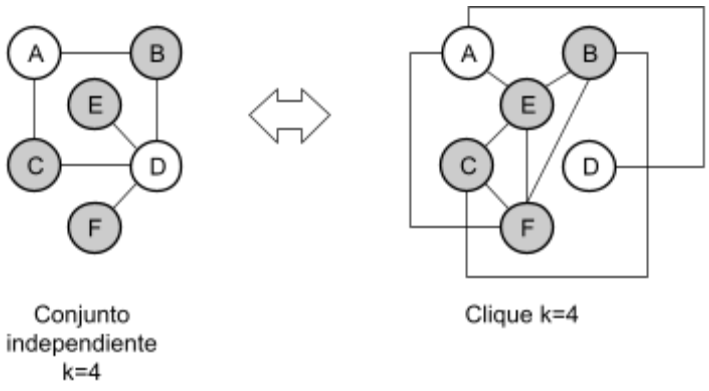
Demostrar que el problema pertenece a NP no es complejo. Utilizaremos como evidencia un subconjunto de vértices del grafo. Primero analizaremos si el certificado contiene k o más vértices que pertenecen al grafo. Proceso que se puede hacer en tiempo lineal en función de la cantidad de vértices. Luego verificamos si existe un eje entre cada par de vértices. El pareo entre vértices se puede hacer en $O(V^2)$ dado que en el peor de los casos tendremos V vértices. La existencia de los ejes entre cada par de vértices depende de cómo se almacena el grafo. En caso de ser una matriz corresponderá a un proceso O(1). Unificando todo el proceso vemos que se realiza en tiempo polinomial.

Para la demostración de la pertenencia de Clique a NP-Hard utilizaremos Conjunto independiente. Realizaremos CI $\leq_P$ Clique. Para hacerlo veremos la relación entre estos dos problemas y sus soluciones. Aunque previo necesitamos una definición

Sea un grafo G=(V,E), llamamos a su grafo complementario G^C=(V',E'). Los vértices de ambos grafos son los mismos: V=V'. Los ejes E' del complementario corresponden a aquellos ejes que unen vértices que no se encuentran unidos en el grafo G. Para construir el complementario podemos recorrer cada par de vértices en V y en caso de no existir el eje que lo une en E crearlo en E'. Este proceso se puede realizar en $O(V^2)$ partiendo de una matriz de adyacencia.

Supongamos que un grafo G=(V,E) contiene un conjunto independiente I $\subseteq$ V. Llamaremos m al tamaño del conjunto independiente. Sabemos que no existen ejes entre los vértices de I. De lo contrario no sería un conjunto independiente. Si construimos el grafo complementario G^C podemos observar que cada par de ejes en I tendrá un eje en G^C. Ocurre por la definición de grafo complementario: crea ejes donde no existían en el grafo original y quita los ejes existentes. Esto conforma un clique con los mismo vértices de I y por lo tanto con el mismo tamaño.

En la imagen se muestra la relación entre los dos problemas para un ejemplo particular. En el grafo existe un conjunto independiente de tamaño 4. En el complemento del grafo se puede observar que esos mismos vértices conforman un clique de tamaño 4.



Conjunto independiente k=4

Clique k=4

Para realizar la reducción polinomial partimos de un grafo G=(V,E) y con un valor k de un problema CI y la transformamos de la siguiente manera:

- Construimos el grafo complementario de G con los mismos vértices y con los ejes que no se encuentran en el grafo original.
- Tomamos el parámetro k del tamaño del conjunto independiente y lo establecemos como el parámetro que representa el tamaño del clique a hallar.

Con la transformación realizada, si existe un clique en G^c de tamaño k, entonces existe un conjunto independiente de tamaño k (utilizando los mismos vértices). Esta reducción se realiza en tiempo polinomial y funciona para cualquier instancia de conjunto independiente.

Ya demostramos que Clique pertenece a NP-Hard. Sumando la demostración de que pertenece a NP, podemos afirmar que pertenece a NP-Complete.

10. Cobertura de conjuntos

Contamos con un conjunto U de "n" elementos al que llamamos universo. Además una colección de "m" subconjuntos de U. Cada subconjunto S_i puede tener entre 1 y n elementos. Un elemento de U puede estar en varios subconjuntos. Diremos que una colección de los subconjuntos cubre U si contiene todos los elementos del mismo. El tamaño de la colección corresponde a la cantidad de subconjuntos que contiene. La unión de todos los subconjuntos cubre U. Por lo tanto se puede cubrir U con "m" subconjuntos.

El problema de optimización consiste en determinar la menor cantidad de subconjuntos a seleccionar para cubrir U. Mientras que la versión de decisión consiste en responder si es posible cubrir el universo de elementos con "k" o menos subconjuntos. Este problema denominado en inglés como "set cover" lo definiremos formalmente como:

Set cover (SC)
Dado un conjunto U de "n" elementos y una colección de "m" subconjuntos de U. Queremos saber si existe una colección que no supere los "k" subconjuntos que cubra U.

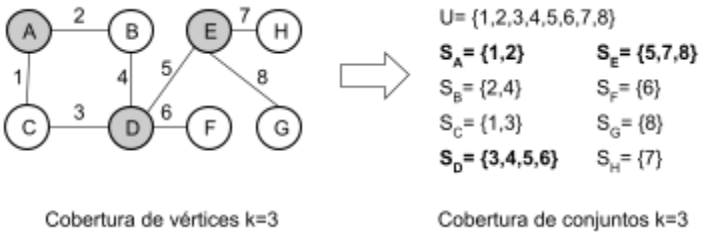
En el ejemplo siguiente nuestro conjunto U contiene los siguientes 9 elementos {a, b, c, d, e, f, g, h, i}. Contamos con los siguientes 8 subconjuntos S_1={a,b,d,e}, S_2={b,e,h}, S_3={c,f}, S_4={a,d,g}, S_5={a,e,i}, S_6={e,f}, S_7={b,d}, S_8={i}. La unión de los subconjuntos S_2, S_3, S_4, S_8 corresponde a una cobertura de de conjuntos de tamaño 4.

Para demostrar que el problema pertenece a la clase NP debemos plantear un certificador polinomial. La evidencia será en forma de una enumeración de subconjuntos. El certificador deberá verificar que esos subconjuntos sean parte de la instancia, que no sean

más de “k” y que incluyan a todos los elementos de U. Cada uno de estos puntos se puede realizar en forma polinomial.

Para la demostración de la pertenencia de Cobertura de conjuntos (SC) a NP-Hard utilizaremos Cobertura de vértices. Realizaremos CV $\leq_P$ SC. Para hacerlo veremos la relación entre estos dos problemas y sus soluciones.

En la cobertura de vértices buscamos que todos los ejes queden cubiertos. Estos mismos se pueden pensar como los elementos del universo. Por otro lado, quienes cubren los ejes corresponden a los vértices. Un eje puede ser cubierto únicamente por sus vértices adyacentes. Cada vértice se puede pensar como un subconjunto que tiene como elementos a aquellos que representan a cada uno de los ejes que inciden en él. Con eso definido si se resuelve el problema de cobertura de conjuntos y es posible seleccionar “k” o menos subconjuntos, estos representan a la misma cantidad de vértices que se puede seleccionar para cubrir todos los ejes del grafo del problema de cobertura de vértices.



Cobertura de vértices k=3

Cobertura de conjuntos k=3

Vemos en la imagen un ejemplo donde una instancia del problema de cobertura de vértices donde existe una solución posible para k=3 se puede transformar en un problema de cobertura de conjuntos. Este último problema es posible encontrar una cobertura con igual cantidad de subconjuntos.

Formalizaremos a continuación la reducción polinomial. Partimos de una instancia G=(V,E) de cobertura de vértices con valor k buscado.

- Construimos el universo con cada uno de los E ejes del grafo
- Construimos |V| subconjuntos. Por cada v vértices incluimos a sus ejes incidentes en el subconjunto S_v
- Solicitamos que la cobertura de conjuntos tenga “k” o menos subconjuntos.

Con la transformación realizada, si existe una cobertura de conjuntos de tamaño k, entonces existe una cobertura de vértices de tamaño k. Esta reducción se realiza en tiempo polinomial y funciona para cualquier instancia de cobertura de vértices.

Demostramos que la cobertura de conjuntos pertenece a NP y a NP-Hard. En conclusión, podemos afirmar que pertenece a NP-Complete.