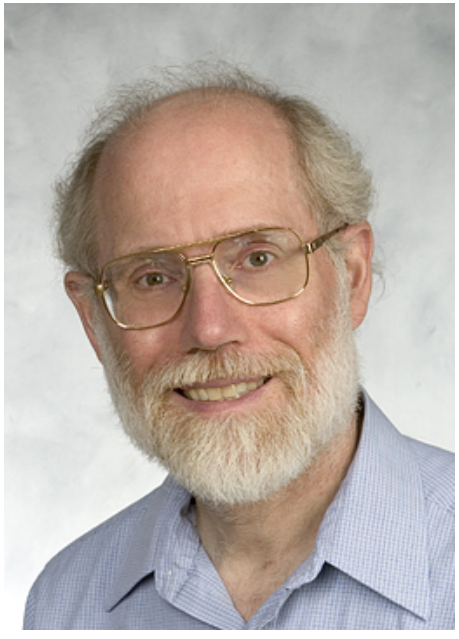
Algoritmos y Estructuras de Datos III

Primer cuatrimestre 2022

Teoría de NP-completitud

Teoría de NP-completitud

Michael Garey



David Johnson

- M. Garey y D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, 1979.

Distintas versiones de un problema de optimización

Dada una instancia I del problema Π con función objetivo f:

- Versión de optimización: Encontrar una solución óptima del problema Π para I (de valor mínimo o máximo)
- Versión de evaluación: Determinar el valor de una solución óptima de Π para I.
- Versión de localización: Dado un número k, determinar una solución factible S de Π para I tal que $f(S) \leq k$ si el problema es de minimización (o $f(S) \geq k$ si el problema es de maximización).
- Versión de decisión: Dado un número k, ¿existe una solución factible S de Π para I tal que $f(S) \leq k$ si el problema es de minimización (o $f(S) \geq k$ si el problema es de maximización)?

Ejemplo: Problema del viajante de comercio

Dado un grafo G con longitudes asignadas a sus aristas:

- Versión de optimización: Determinar un circuito hamiltoniano de G de longitud mínima.
- Versión de evaluación: Determinar el valor de una solución óptima, o sea la longitud de un circuito hamiltoniano de G de longitud mínima.
- Versión de localización: Dado un número k, determinar un circuito hamiltoniano de G de longitud menor o igual a k.
- Versión de decisión: Dado un número k, ¿existe un circuito hamiltoniano de G de longitud menor o igual a k?

Teoría de complejidad computacional

- Para muchos problemas de optimización combinatoria las cuatro versiones son equivalentes: si existe un algoritmo eficiente para una de ellas, entonces existe para todas.
- La clasificación y el estudio se realiza sobre problemas de decisión.
- Un problema de decisión tiene respuesta SI o NO.
- Esto permite uniformizar el estudio, ya que hay problemas que no tienen versión de optimización.

Instancias de un problema

- Definimos un problema dando su entrada y su salida.

  SATISFACTIBILIDAD (SAT)
  1. **Entrada:** Una fórmula proposicional f en forma normal conjuntiva.
  2. **Salida:** ¿Existe una asignación de valores de verdad a las proposiciones de f que hace que f sea verdadera?

- Una instancia de un problema es una especificación de sus parámetros.

- Un problema de decisión Π tiene asociado un conjunto D_Π de instancias, y un subconjunto $Y_\Pi \subseteq D_\Pi$ de instancias cuya respuesta es SI.

Máquina de Turing determinística (MTD)

Alan Turing
(1912–1954)

- La máquina de Turing es un modelo de cómputo teórico propuesto por Alan Turing (1937).

Máquina de Turing determinística (MTD)

- Una cabeza lecto-escritora.
- Una cinta infinita con celdas indexadas en $\mathbb{Z}$.
- La cabeza lectora se puede mover en ambas direcciones, $M = \{+1, -1\}$, derecha $(+1)$ o izquierda (-1).
- La celda 0 es la celda inicial.

Máquina de Turing determinística (MTD)

- Un alfabeto finito Σ y un símbolo especial $*$ llamado "blanco". Definimos $\Gamma = \Sigma \cup \{*\}$.
- Sobre la cinta está escrita la entrada, que es un string de símbolos de Σ y el resto de las celdas tiene $*$ (blancos).
- Un conjunto Q finito de estados.
- Un estado inicial $q_0 \in Q$.
- Un conjunto de estados finales $Q_f \subseteq Q$ ($Q_f = \{q_{si}, q_{no}\}$ para problemas de decisión).

Máquina de Turing determinística (MTD)

- Una instrucción en una MTD es una quíntupla $S \in Q \times \Gamma \times Q \times \Gamma \times M$.
- La quíntupla (q, s, q', s', m) se interpreta como:

  Si la máquina está en el estado q y la cabeza lee s, entonces escribe s' en esa celda, realiza el movimiento m (izquierda o derecha) y pasa al estado q'.

- Un programa es un conjunto finito de instrucciones.
- **Determinismo.** Suponemos que para todo par $(q, s) \in Q \times \Gamma$ existe en el programa a lo sumo una quíntupla que comienza con ese par.

Máquina de Turing determinística (MTD)

- Una máquina M resuelve el problema Π si para toda instancia alcanza un estado final y responde de forma correcta (o sea, termina en un estado final correcto).
- La complejidad de una MTD está dada por la cantidad de movimientos de la cabeza, desde el estado inicial hasta alcanzar un estado final, en función del tamaño de la entrada.

$$T_M(n) = \text{máx}\{m : M \text{ realiza } m \text{ movimientos con la entrada } x, x \in D_\Pi,\ |x| = n\}$$

- Una MTD M es polinomial para Π cuando $T_M(n)$ es una función polinomial.

La clase P

- **Definición.** Un problema de decisión Π pertenece a la clase **P** (polinomial) si existe una MTD de complejidad polinomial que lo resuelve.
- Esto es equivalente a afirmar que existe un algoritmo polinomial (en la Máquina RAM) que resuelve Π.

Máquina de Turing no determinística (MTND)

- No se pide unicidad de la quíntupla que comienza con cualquier par (q, s).
- Cuando la máquina está en el estado q y lee el símbolo s, aplica cualquier quíntupla que comience con (q, s).
    1. No está especificado de antemano qué tupla selecciona para continuar la ejecución.
- Un programa puede interpretarse como una tabla que mapea un par (q, t) a un conjunto de ternas (q', s', m).

Máquina de Turing no determinística (MTND)

- Una MTND resuelve el problema de decisión Π si existe una secuencia de alternativas que lleva a un estado de aceptación si y sólo si la respuesta (teórica) es SI.
- Podemos interpretar la ejecución de una MTND como un árbol de alternativas.
- En este caso, la definición es equivalente a afirmar que para toda instancia de Y_Π existe una rama de ejecución que llega a un estado final q_{si} y para toda instancia en $D_\Pi \setminus Y_\Pi$ ninguna rama llega a un estado final q_{si}.

Máquinas de Turing No Determinísticas (MTND)

- La complejidad temporal de una MTND M se define como el máximo número de pasos que toma como mínimo reconocer una instancia de Y_Π en función de su tamaño:

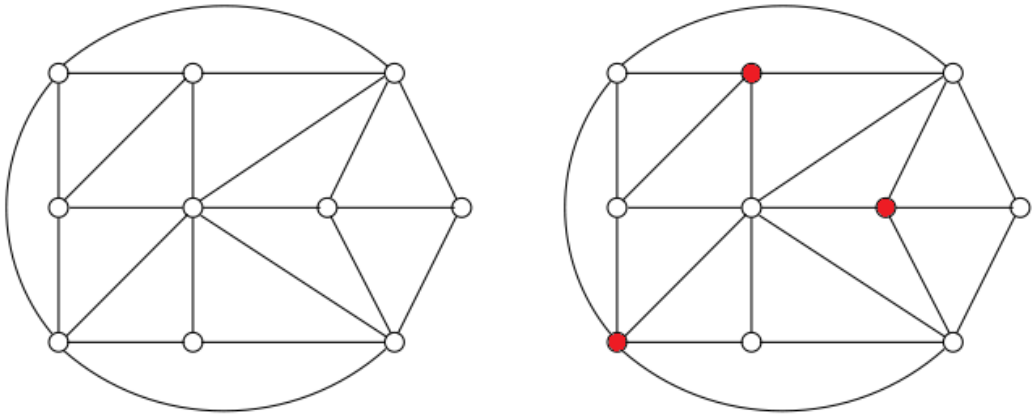
$$T_M(n) = \text{máx}\{m : \text{la rama más corta de } M \text{ termina en estado } q_{si} \text{ en } m \text{ movimientos cuando la entrada es } x,\ x \in Y_\Pi, |x| = n\}$$

- Una MTND M es polinomial para Π cuando $T_M(n)$ es una función polinomial.

La clase NP

- Un problema de decisión Π pertenece a la clase **NP** (polinomial no-determinístico) si existe una MTND polinomial que reconoce todas las instancias de Y_Π.
- Equivalentemente, dada una instancia de Π con respuesta SI se puede dar un certificado de longitud polinomial que garantiza que la respuesta es SI, y esta garantía puede ser verificada en tiempo polinomial.

La clase NP - Conjunto independiente máximo



- Un conjunto independiente en un grafo es un conjunto de vértices que no son vecinos entre sí.

CONJUNTO INDEPENDIENTE MÁXIMO (MIS)

1. **Entrada:** Un grafo G y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Existe un conjunto independiente en G de tamaño k o mayor?

La clase NP - Conjunto independiente máximo

- Dados un grafo $G = (V, X)$ y $k \in N$, ¿G tiene un conjunto independiente de tamaño mayor o igual a k?
- Para una instancia con respuesta SI, podemos exponer $S \subseteq V$ conjunto independiente de G tal que $|S| \geq k$.
- Es posible chequear polinomialmente que S cumple estas dos propiedades: ser conjunto independiente de G y tener cardinal mayor o igual a k.
- Esto demuestra que conjunto independiente pertenece a la clase NP.

La clase NP - SAT

- Satisfactibilidad (SAT)
  1. **Entrada:** Una fórmula proposicional f en forma normal conjuntiva.
  2. **Salida:** ¿Existe una asignación de valores de verdad a las proposiciones de f que hace que f sea verdadera?

- El certificado que podemos mostrar es una asignación de valores de verdad a las variables que haga verdadera a la expresión f.
- Como es posible verificar en tiempo polinomial que esta expresión es verdad con esos valores de las variables, entonces SAT $\in$ NP.

La clase NP - Circuito hamiltoniano

- CIRCUITO HAMILTONIANO
  1. **Entrada:** Un grafo $G = (V, E)$.
  2. **Salida:** ¿Existe una secuencia $i_1, \ldots, i_n$ de vértices tal que $i_j i_{j+1} \in E$ para $j = 1, \ldots, n-1$, y además $i_n i_1 \in E$?
- La evidencia que soporta una respuesta positiva es un ciclo hamiltoniano de G.
- Dada un lista de vértices, se puede chequear polinomialmente si define un ciclo hamiltoniano.

La clase NP - TSP

- Problema del viajante de comercio (TSP)
  1. **Entrada:** Un grafo $G = (V, E)$, una función $d : E \to \mathbb{R}$ y $k \in \mathbb{R}$.
  2. **Salida:** ¿Existe un camino hamiltoniano en G con distancia menor o igual a k?

- La evidencia que soporta una respuesta positiva es un ciclo hamiltoniano de G con distancia menor o igual a k.

- Dada una lista de vértices, se puede chequear polinomialmente si define un ciclo hamiltoniano y que la suma de las distancias de las aristas respectivas es menor o igual a k.

P vs NP

- **Observación.** $P \subseteq NP$.
- **Teorema.** Si Π es un problema de decisión que pertenece a la clase NP, entonces Π puede ser resuelto por un algoritmo determinístico en tiempo exponencial respecto del tamaño de la entrada.
- **Problema abierto.** ¿$P = NP$?
- Mientras tanto, se estudian clases de complejidad relativa, comparando la dificultad entre problemas.

Transformaciones polinomiales

- Una transformación o reducción polinomial de un problema de decisión Π' a uno Π es una función polinomial que transforma una instancia I' de Π' en una instancia I de Π tal que I' tiene respuesta SI para Π' si, y sólo si, I tiene respuesta SI para Π:

$$I' \in Y_{\Pi'} \Longleftrightarrow f(I') \in Y_{\Pi}$$

- El problema de decisión Π' se reduce polinomialmente a otro problema de decisión Π, $\Pi' \leq_p \Pi$, si existe una transformación polinomial de Π' a Π.

- **Proposición.** Las reducciones polinomiales son transitivas:

$$\text{si } \Pi_1 \leq_p \Pi_2 \text{ y } \Pi_2 \leq_p \Pi_3 \text{ entonces } \Pi_1 \leq_p \Pi_3.$$

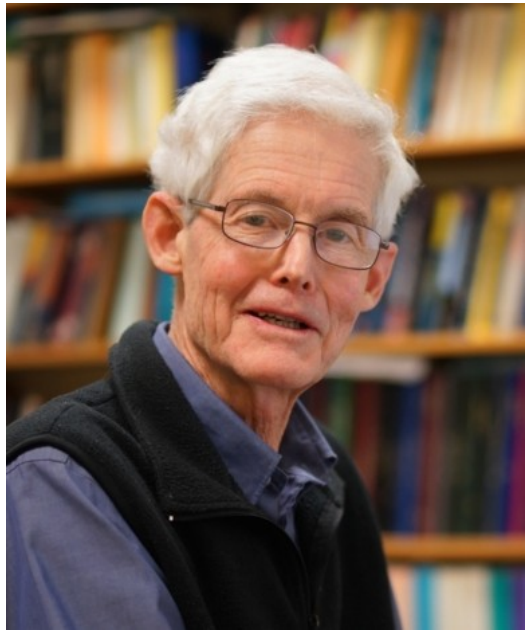
La clase NP-completo

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

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

Si un problema Π verifica la condición 2., Π es **NP-difícil** (es al menos tan difícil como todos los problemas de NP).

Problemas NP-completos



Stephen Cook
(1939–)

Leonid Levin
(1948–)

- **Teorema (Cook, 1971 – Levin, 1973).**
  SAT es NP-completo.

SAT

Teorema (Cook, 1971 – Levin, 1973).
SAT es NP-completo.

Demostración. Sabemos que SAT $\in$ NP, con lo cual solamente queda ver que $\Pi \leq_p$ SAT para todo $\Pi \in$ NP. Sea $\Pi \in$ NP, con lo cual existe una MTND que lo resuelve. Llamamos:

- Γ al alfabeto de esta MTND,
- Q a su conjunto de estados,
- $s \in Q$ al estado inicial,
- $F \subseteq Q$ al conjunto de estados finales, y
- $\Delta \subseteq Q \times \Gamma \times Q \times \Gamma \times M$ al conjunto de instrucciones

(con $M = \{-1, 0, +1\}$).

SAT

Reducimos Π a SAT del siguiente modo. Dado un input I de Π, construimos una fórmula proposicional f tal que f es satisfactible si y sólo si $I \in Y_\Pi$. Sea $p(n)$ la función (polinomial) de complejidad de la MTND. La fórmula f contiene las siguientes proposiciones:

- T_{ijk}: la celda i contiene el símbolo j en el paso k de la ejecución de la MTND,
- H_{ik}: el cabezal de la MTND está ubicado sobre la celda i en el paso k,
- Q_{qk}: la MTND está en estado q en el paso k,

para $i = -p(n), \ldots, p(n)$, $k = 0, \ldots, p(n)$, $j \in \Gamma$ y $q \in Q$.

Llamamos $I = (j_0, \ldots, j_{n-1})$, y suponemos que el input comienza en la celda 0.

SAT

La fórmula se construye como la conjunción de los siguientes términos.

- T_{ij_i0}: la celda i contiene el valor j_i en tiempo 0, para $i = 0, \ldots, n-1$.
- T_{i*0}: la celda i contiene el valor blanco en tiempo 0, para $i \in \{-p(n), \ldots, p(n)\} \backslash \{0, \ldots, n-1\}$.
- H_{00}: el cabezal comienza en la celda 0.
- Q_{s0}: la máquina comienza en el estado s.
- $\neg(T_{ijk} \wedge T_{ij'k})$: la celda i contiene a lo sumo un símbolo en el paso k $(j \neq j')$
- $\vee_{j \in \Gamma} T_{ijk}$: le calda i contiene al menos un símbolo en el paso k.

SAT

- $T_{ijk} \wedge T_{ij',k+1} \rightarrow H_{ik}$: las celdas no apuntadas por el cabezal no cambian ($j \neq j'$, $k < p(n)$).
- $\neg(Q_{qk} \wedge Q_{q'k})$: la máquina está en a lo sumo un estado en el paso k ($q \neq q'$).
- $\neg(H_{ik} \wedge H_{i'k})$: el cabezal apunta a lo sumo a una celda en el paso k ($i \neq i'$).
- $(H_{ik} \wedge Q_{qk} \wedge T_{i\sigma k}) \rightarrow \vee_{(q,\sigma,q',\sigma',m)\in\Delta}(H_{i+m,k+1} \wedge Q_{q',k+1} \wedge T_{i,\sigma',k+1})$: transiciones posibles en el paso $k < p(n)$.
- $\vee_{k=0}^{p(n)} \vee_{f\in F} Q_{fk}$: la máquina termina en un estado final.

SAT

Si hay un cómputo de la MTND con el input I que termina en un estado de F, entonces f es satisfactible asignando a las proposiciones su interpretación. Recíprocamente, si f es satisfactible entonces existe un cómputo de la MTND a partir de I, siguiendo los pasos especificados por las proposiciones verdaderas. Finalmente, la fórmula tiene $O(p(n)^2)$ proposiciones y $O(p(n)^3)$ cláusulas, con lo cual la transformación es polinomial.

Como Π es un problema arbitrario en NP, concluimos que cualquier problema de NP reduce a SAT que, por lo tanto, es NP-completo. $\square$

¿Cómo se prueba que un problema es NP-completo?

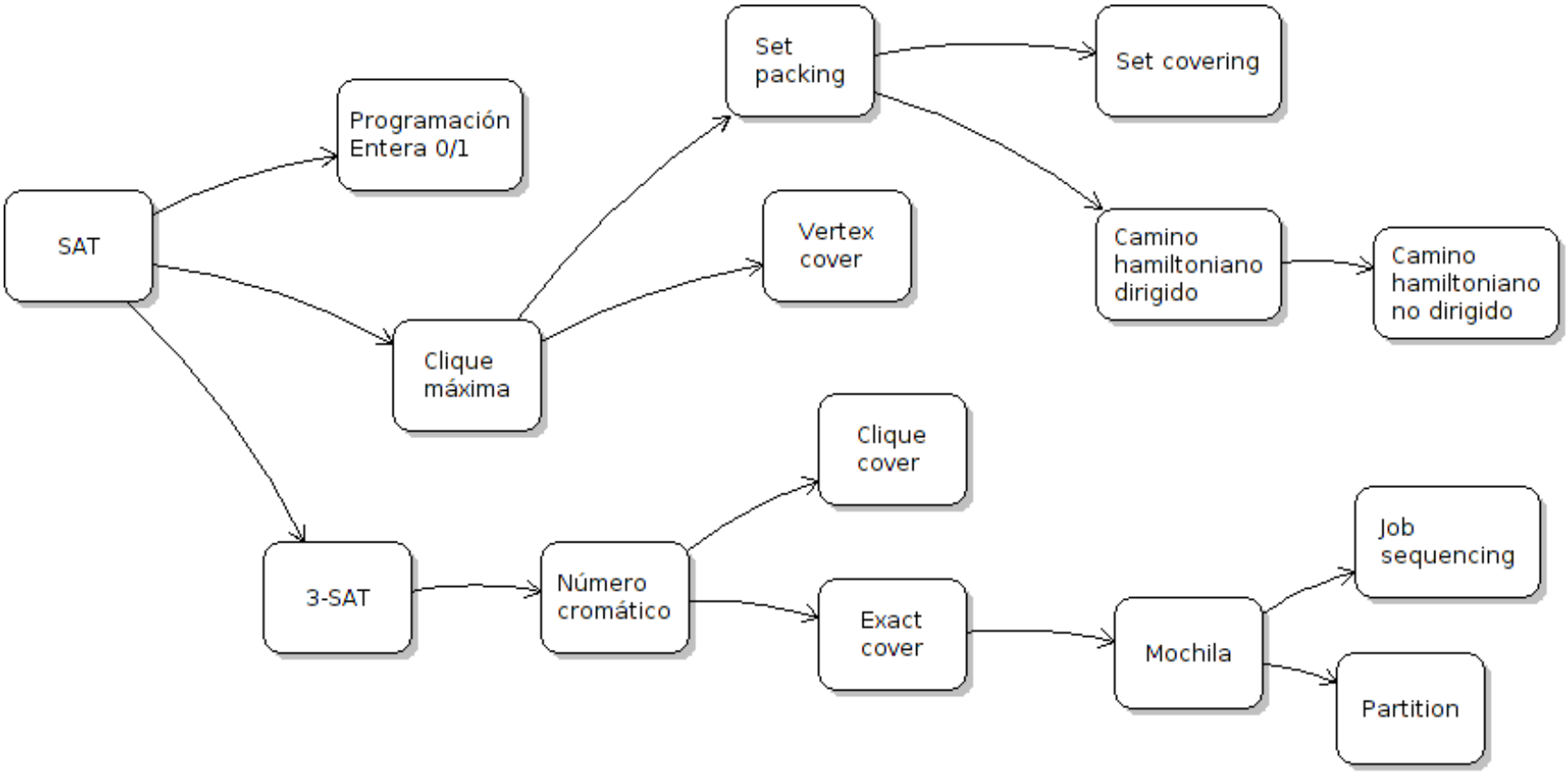
- Usando la transitividad de las reducciones polinomiales, a partir de este primer resultado podemos probar que otros problemas son NP-completos.
- Si Π es un problema de decisión, podemos probar que $\Pi \in$ NP-completo encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que:
  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.

Problemas NP-completos

Richard Karp

- A partir del Teorema de Cook-Levin, Karp demostró en 1972 que otros 21 problemas son NP-completos.
- Actualmente se conocen más de 3.000 problemas NP-completos!

Problemas NP-completos



3-SAT

SATISFACTIBILIDAD (SAT)

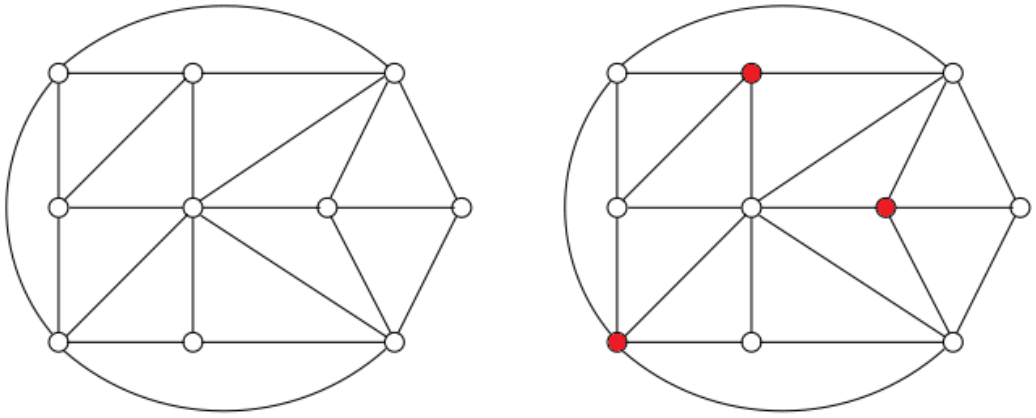
1. **Entrada:** Una fórmula proposicional f en forma normal conjuntiva.
2. **Salida:** ¿Existe una asignación de valores de verdad a las proposiciones de f que hace que f sea verdadera?

SAT CON TRES LITERALES POR CLÁUSULA (3-SAT)

1. **Entrada:** Una fórmula proposicional f en forma normal conjuntiva en la que cada cláusula tiene exactamente tres literales.
2. **Salida:** ¿Existe una asignación de valores de verdad a las proposiciones de f que hace que f sea verdadera?

- **Teorema.** 3-SAT es NP-completo.

Conjunto independiente máximo



Conjunto independiente máximo (MIS)

1. **Entrada:** Un grafo G y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Existe un conjunto independiente en G de tamaño k o mayor?

- **Teorema.** MIS es NP-completo.

Conjunto independiente de peso máximo

Conjunto independiente de peso máximo (MWIS)

1. **Entrada:** Un grafo G, pesos $w : V \to \mathbb{R}$ y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Existe un conjunto independiente I en G de peso $\sum_{i \in I} w_i$ mayor o igual a k?

- **Teorema.** MWIS es NP-completo.

Clique máxima (CLIQUE)

1. **Entrada:** Un grafo G y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Existe una clique en G de tamaño k o mayor?

- **Teorema.** CLIQUE es NP-completo.

Otros problemas NP-completos

Número cromático

1. **Entrada:** Un grafo G y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Se puede colorear G con k colores?

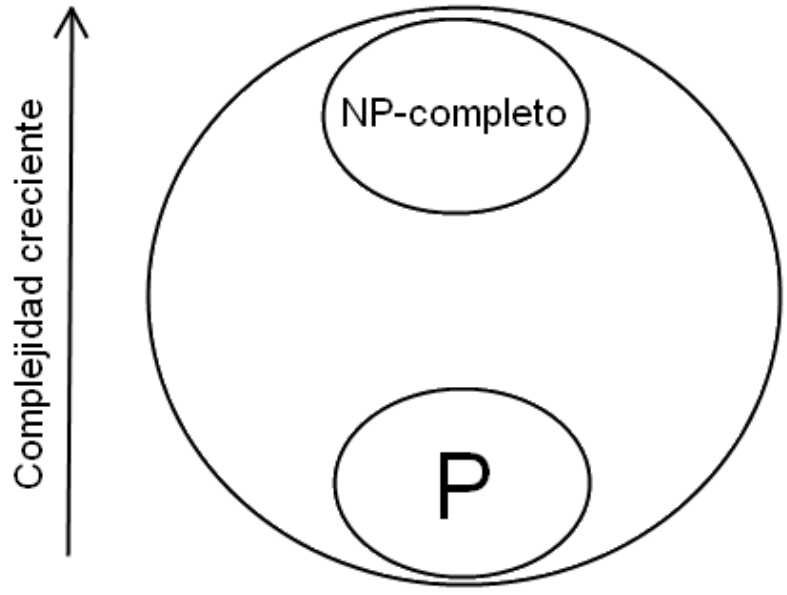
Set-partitioning

1. **Entrada:** Un conjunto finito A y una función $s : A \to \mathbb{Z}_+$.
2. **Salida:** ¿Existe un subconjunto $A' \subseteq A$ tal que $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$?

Matching 3D

1. **Entrada:** Un conjunto $M \subseteq A \times B \times C$, donde $|A| = |B| = |C| = q \in \mathbb{Z}$.
2. **Salida:** ¿Existe un subconjunto $M' \subseteq M$ tal que $|M'| = q$ y no hay dos elementos de M' que coincidan en alguna coordenada?

P vs NP



- Hasta el momento no se conoce ningún problema en NP-completo $\cap$ P.
- Tampoco se ha demostrado que exista algún problema en NP\P. En ese caso se probaría que P $\neq$ NP.

P vs NP

- Determinar si P=NP o P≠NP es uno de los problemas del milenio.
- Se trata de uno de los problemas abiertos más importantes de la computación.

Extensión de un problema

- El problema Π es una restricción de un problema $\bar{\Pi}$ si el dominio de Π está incluido en el dominio 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.
- Es habitual que un caso particular (restricción) de un problema NP-completo esté en P, pero no se puede dar la situación recíproca, salvo que P=NP.

Extensión de un problema

- 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 que 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 (no puede tener a K_5 como subgrafo), mientras que Clique es NP-completo.

Extensión de un problema

Si Π es una restricción de $\bar{\Pi}$, 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.

La clase Co-NP

- Vimos que circuito hamiltoninano está en la clase NP. Pero consideremos ahora su versión inversa:

  CIRCUITO HAMILTONIANO COMPLEMENTO
  1. **Entrada:** Un grafo $G = (V, E)$.
  2. **Salida:** ¿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 circuito.

- Este certificado obviamente no es polinomial, por lo tanto no permite mostrar que el problema está en NP.

La clase Co-NP

- 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:
  1. $D_{\Pi^c} = D_\Pi$ y
  2. $Y_{\Pi^c} = D_\Pi \setminus Y_\Pi$
- ¿Qué otros pares de problemas complementarios se pueden mencionar?

La clase Co-NP

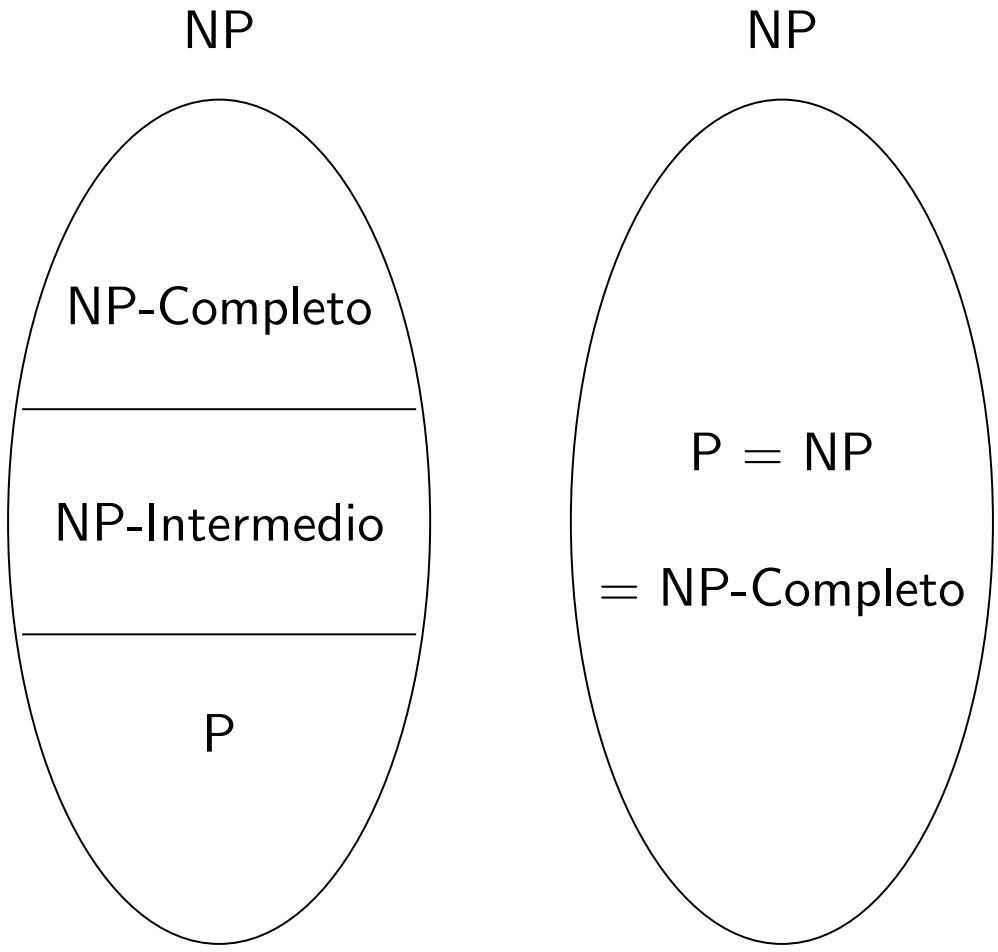
- **Proposición.** Si $\Pi \in P$, entonces $\Pi^c \in P$.
- Este argumento no aplica para la clase NP:
  1. Si un problema Π está en NP no sirve este argumento para demostrar que Π^c está en NP.
  2. Más aún, no se sabe si esto es cierto en general.
- Un problema de decisión pertenece a la clase Co-NP si dada una instancia de NO y evidencia polinomial de la misma, puede ser verificada en tiempo polinomial.
- **Proposición,** Si $\Pi \in$ NP, entonces $\Pi^c \in$ Co-NP.

Problemas abiertos

Con estas nuevas definiciones tenemos los siguientes problemas abiertos:

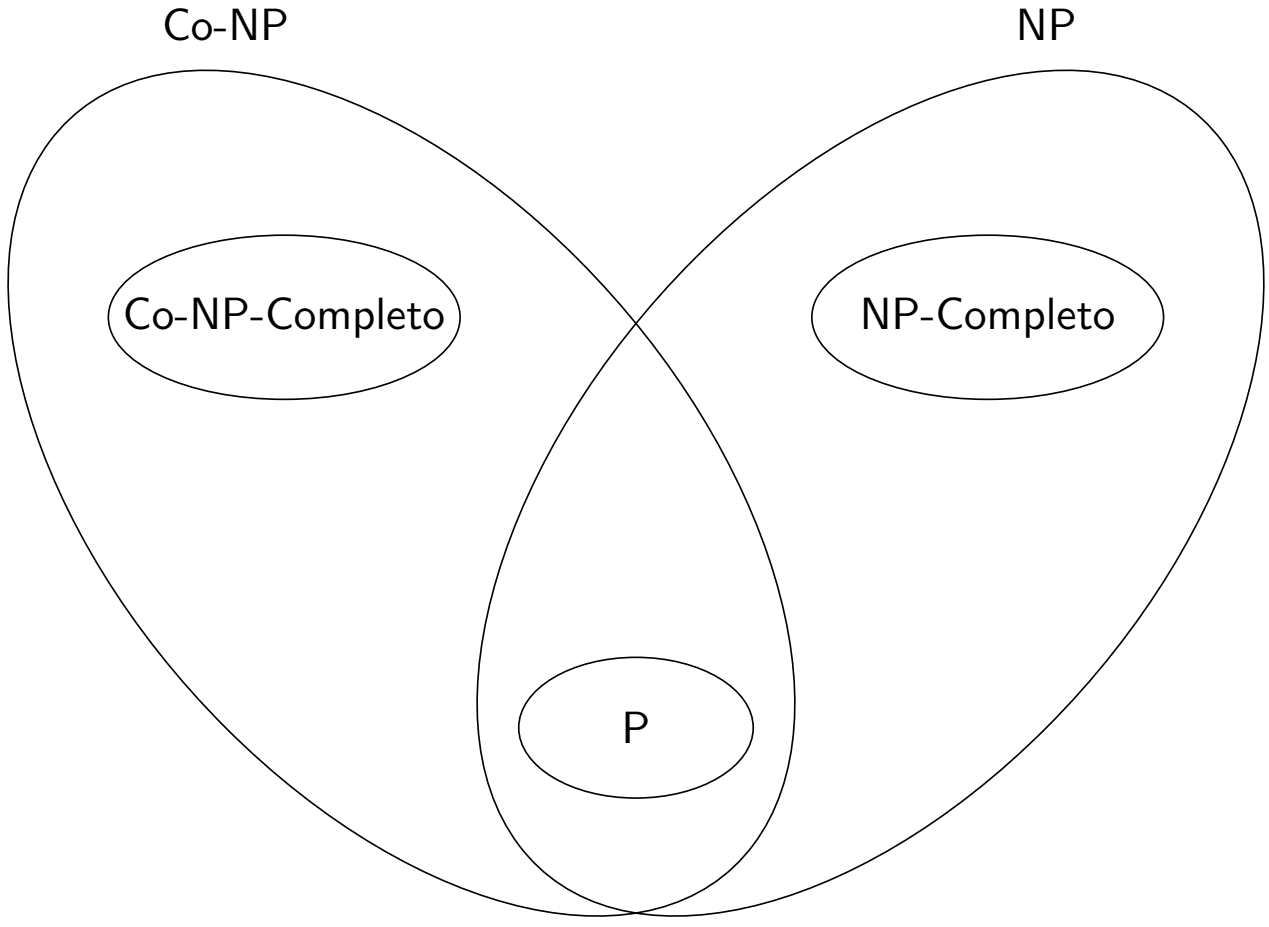
- ¿Es P=NP?
- ¿Es Co-NP=NP?
- ¿Es P=Co-NP $\cap$ NP?

Las incógnitas...



Dos mapas posibles para las clases de complejidad

Las incógnitas...



Situación si se probara que $P \neq NP$, $NP \neq Co - NP$, $P \neq Co - NP \cap NP$

Teoría de complejidad computacional

- Dado un problema de decisión en NP, tenemos tres posibilidades:
  1. Existe un algoritmo polinomial para el problema (y se demuestra encontrando un algoritmo polinomial que lo resuelve).
  2. El problema es NP-completo (y se demuestra a través de una transformación polinomial desde otro problema NP-completo).
  3. Es un problema abierto!
- ¿Qué importancia tiene saber si un problema está en NP-completo desde el punto de vista teórico?

Teoría de complejidad computacional

- COLOREO es NP-completo para grafos generales, y también para ...
  1. grafos arco-circulares,
  2. grafos que no contienen P_5 como subgrafo inducido,
  3. grafos planares (incluso 4-regulares),
  4. grafos sin triángulos (incluso para $k = 3$),
  5. etc.
- COLOREO es polinomial para ...
  1. grafos arco-circulares propios,
  2. cografos (grafos sin P_4 inducidos),
  3. grafos de intervalos,
  4. grafos cordales,
  5. grafos perfectos,
  6. grafos sin $K_{1,3}$ inducidos,
  7. etc.

Teoría de complejidad computacional

Class	coloring	PrExt	μ-col.	(γ, μ)-col.	list-col.
Complete bipartite	P	P	**P**	**P**	NP-c [20]
Bipartite	P	NP-c [17]	NP-c [4]	NP-c	NP-c [22]
Cographs	P [13]	P [18]	P [4]	?	NP-c [20]
Distance-hereditary	P [13]	**NP-c**	**NP-c**	**NP-c**	NP-c [20]
Interval	P [13]	NP-c [3]	**NP-c**	NP-c	NP-c
Unit interval	P	NP-c [23]	?	NP-c	NP-c
Split	P	P [18]	**NP-c**	**NP-c**	NP-c
Complete split	P	P	**P**	**P**	NP-c [20]
Trivially perfect	P	P	P	?	NP-c
Threshold	P	P	P	?	NP-c
Line of $K_{n,n}$	P [21]	NP-c [8]	**NP-c**	NP-c	NP-c
Complement of bipartite	P [13]	P [18]	?	?	NP-c [19]
Line of K_n	P [21]	**NP-c**	**NP-c**	**NP-c**	NP-c [22]

Teoría de complejidad computacional

- ¿Qué hacemos si tenemos que resolver un problema NP-completo?
  1. Estudiar si no estamos ante una restricción del problema que se pueda resolver en forma eficiente.
  2. Analizar si el problema admite un algoritmo pseudopolinomial.
  3. Analizar si se puede reducir a un problema NP-completo que tenga solvers eficientes en la práctica (como SAT o programación lineal entera).
  4. Analizar si el tamaño de las instancias a resolver permite un enfoque basado en fuerza bruta o backtracking.
  5. Diseñar heurísticas para el problema, tratando de aprovechar su estructura particular.
  6. Analizar si existen algoritmos aproximados para el problema.