Teoría de NP-completitud

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella

UNIVERSIDAD
TORCUATO DI TELLA

Complejidad computacional: Las reglas del juego

Problema

En el contexto de la teoría de complejidad computacional, llamamos problema a la descripción de los datos de entrada y la respuesta a proporcionar para cada dato de entrada.

Instancia

Una instancia de un problema es un juego válido de datos de entrada.

- Ejemplo:

  1. **Entrada:** Un número n entero no negativo.
  2. **Salida:** ¿El número n es primo?

- En este ejemplo, una instancia está dada por un número entero no negativo.

Definición

Un problema de optimización consiste en encontrar la mejor solución dentro de un conjunto:

$$z^* = \max_{x \in S} f(x) \qquad \text{o bien} \qquad z^* = \min_{x \in S} f(x)$$

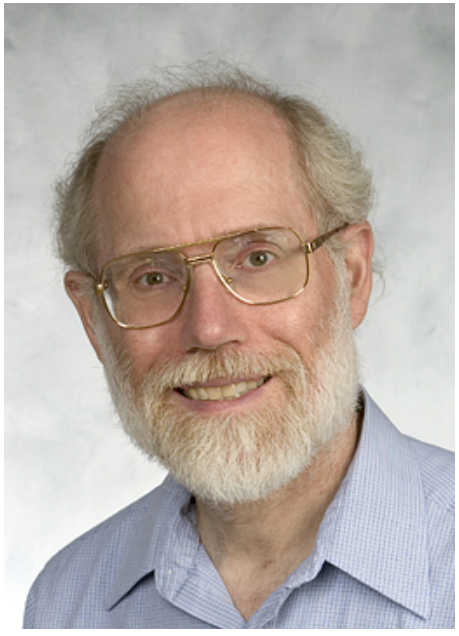
donde

- La función $f : S \to \mathbb{R}$ se denomina función objetivo del problema.
- El conjunto S es la región factible y los elementos $x \in S$ se llaman soluciones factibles.
- El valor $z^* \in \mathbb{R}$ es el valor óptimo del problema, y cualquier solución factible $x^* \in S$ tal que $f(x^*) = z^*$ se llama un óptimo del problema.

Complejidad computacional: Las reglas del juego

- Suponemos una Máquina RAM (*random access memory*).
  1. La memoria está dada por una sucesión de celdas numeradas. Cada celda puede almacenar un valor de b bits.
  2. Supondremos habitualmente que el tamaño b en bits de cada celda está fijo, y suponemos que todos los datos individuales que maneja el algoritmo se pueden almacenar con b bits.
  3. Se tiene un programa imperativo no almacenado en memoria, compuesto por asignaciones y las estructuras de control habituales.
  4. Las asignaciones pueden acceder a celdas de memoria y realizar las operaciones estándar sobre los tipos de datos primitivos habituales.

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: Camino mínimo

Dado un digrafo pesado $G = (N, A)$ con longitudes asociadas a sus arcos, y dados dos nodos $s, t \in N$:

- Versión de optimización: Encontrar un camino en G entre s y t de longitud mínima.
- Versión de evaluación: Determinar la longitud de un camino mínimo en G entre s y t (es decir, determinar el valor de la solución óptima).
- Versión de localización: Dado un número k, encontrar (si existe) un camino en G entre s y t de longitud menor o igual a k.
- Versión de decisión: Dado un número k, ¿existe un camino en G entre s y t con longitud menor o igual a k?

- 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 Sí 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.
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 Sí.

Las clases P y NP

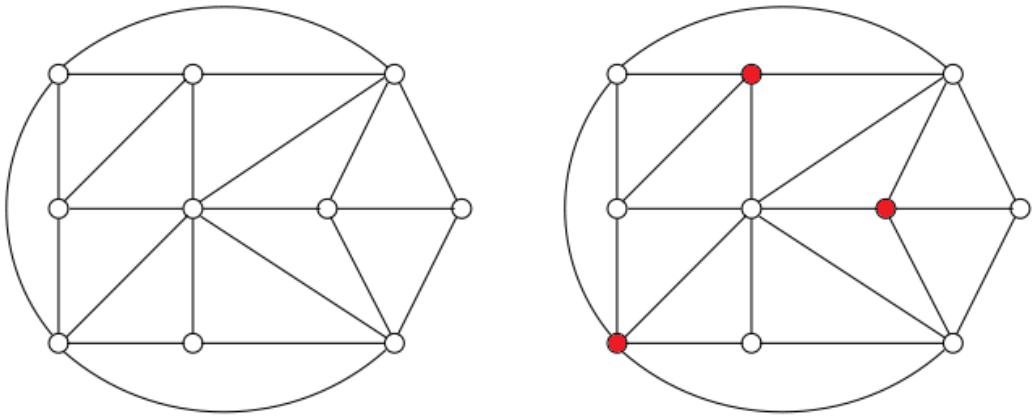
Clase P

Un problema de decisión Π pertenece a la clase **P** (polinomial determinístico) si existe un algoritmo polinomial que lo resuelve. Esdecir, dada una instancia de Π con respuesta Sí se puede dar un cómputo de longitud polinomial que garantiza que la respuesta es Sí.

Clase NP

Un problema de decisión Π pertenece a la clase **NP** (polinomial no-determinístico) si dada una instancia de Π con respuesta Sí se puede dar un certificado de longitud polinomial que garantiza que la respuesta es Sí, 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 Sí, 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 MIS $\in$ NP.

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?

- **Certificado:** Una asignación concreta de valores de verdad a todas las variables que satisface la fórmula f.
- **Verificación:** Sustituir los valores en la fórmula y comprobar que se evalúa a verdadero. Esto se hace en tiempo polinomial. (Cómo?)
- Entonces SAT $\in$ NP.

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 $i_n i_1 \in E$?

- **Certificado:** La lista ordenada de vértices que forman el ciclo.
- **Verificación:** (Cómo?)
  - Verificar que cada vértice aparece exactamente una vez,
  - que cada par consecutivo de vértices está conectado por una arista,
  - y que el último vértice se conecta con el primero.

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?

- **Certificado:** Una lista de ciudades que representa el recorrido.
- **Verificación:**
  - Verificar que cada vértice aparece exactamente una vez,
  - que cada par consecutivo de vértices está conectado por una arista,
  - y que el último vértice se conecta con el primero,
  - sumar las distancias del recorrido y verificar que el total sea menor o igual a k.

P vs NP

Observación

$P \subseteq NP$.

Problema abierto.

¿$P = NP$?

- La pregunta por $P = NP$ apunta a distinguir si computar una solución a un problema es polinomialmente equivalente a verificar la solución de un problema.
- No se sabe si $P = NP$ o si $P \neq NP$. 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 Sí para Π' si, y sólo si, I tiene respuesta Sí para Π:

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

- 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:

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

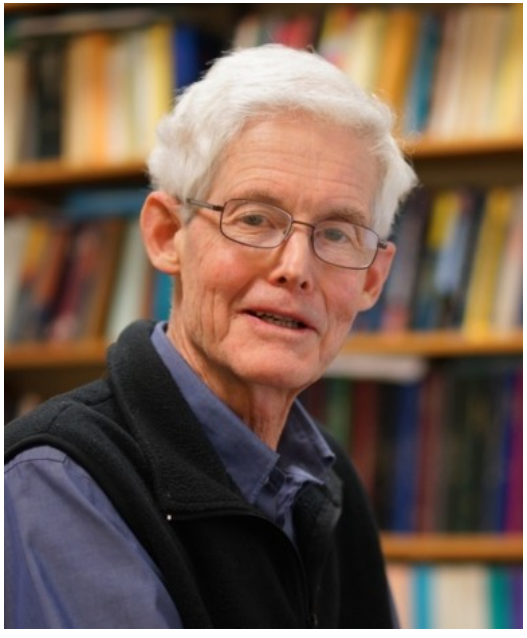
Clase NP-completo

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, decimos que Π es NP-difícil (es al menos tandifícil como todos los problemas de NP).

Problemas NP-completos



Stephen Cook
(1939–)

Leonid Levin
(1948–)

Teorema (Cook, 1971 – Levin, 1973)

SAT es NP-completo.

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

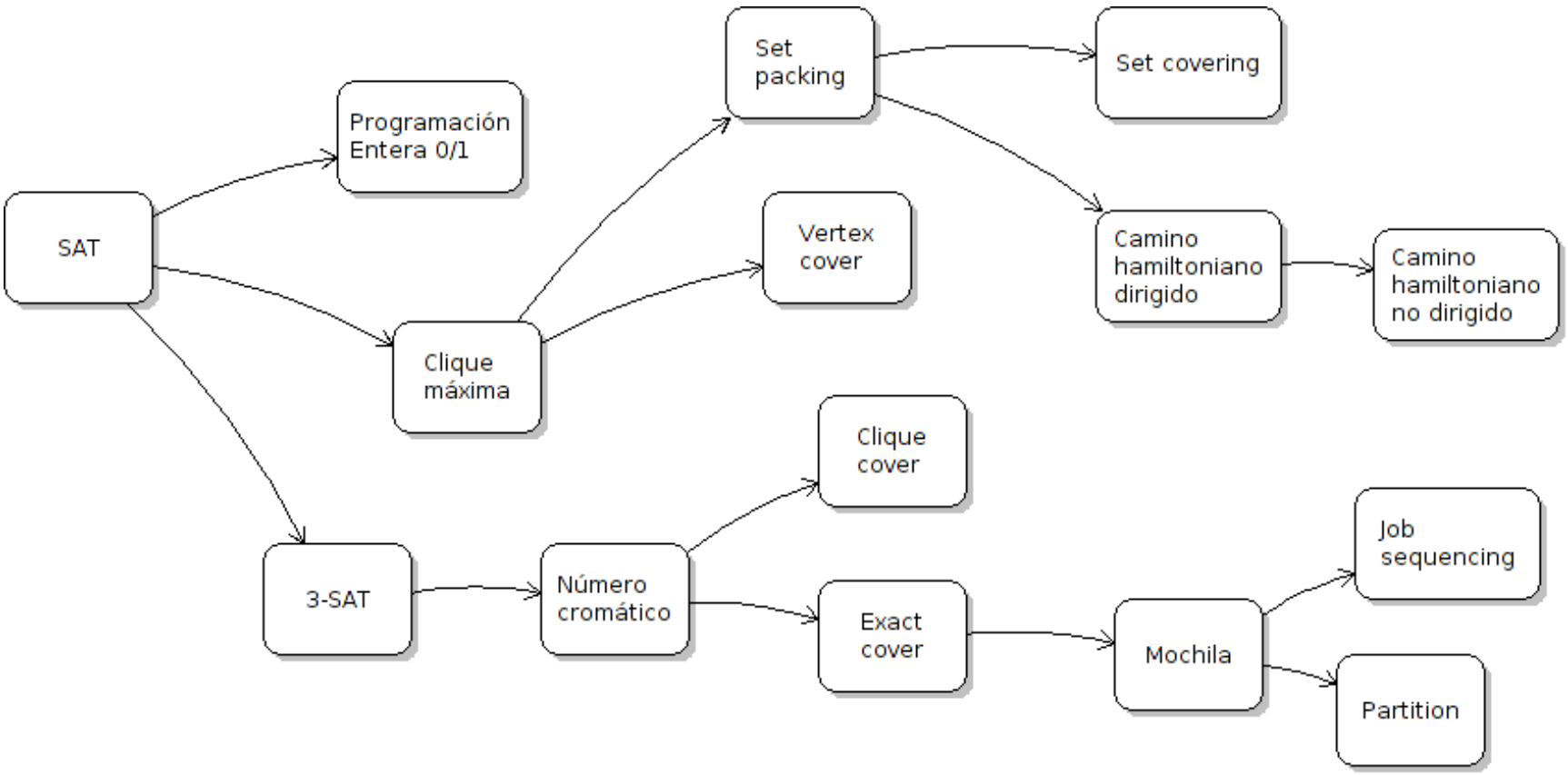
- Usando la transitividad de las reducciones polinomiales, a partir de este primer resultado podemos probar que otros problemas son NP-completos.
- Si Π es un problema de decisión, podemos probar que $\Pi \in$ NP-completo encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que: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



Satisfactibilidad (SAT)

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

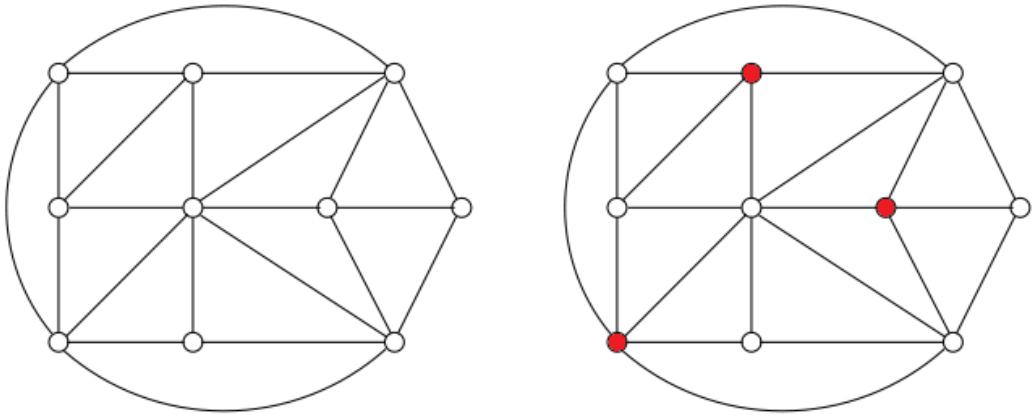
SAT con tres literales por cláusula (3-SAT)

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?

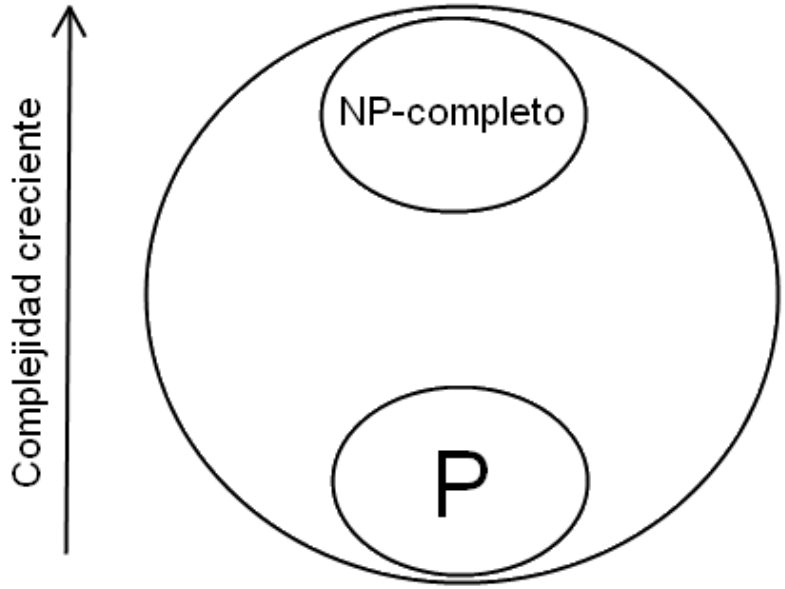
Set-partitioning

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 ∩ P.
- Tampoco se ha demostrado que exista algún problema en NP\P. En ese caso se probaría que P ≠ 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:

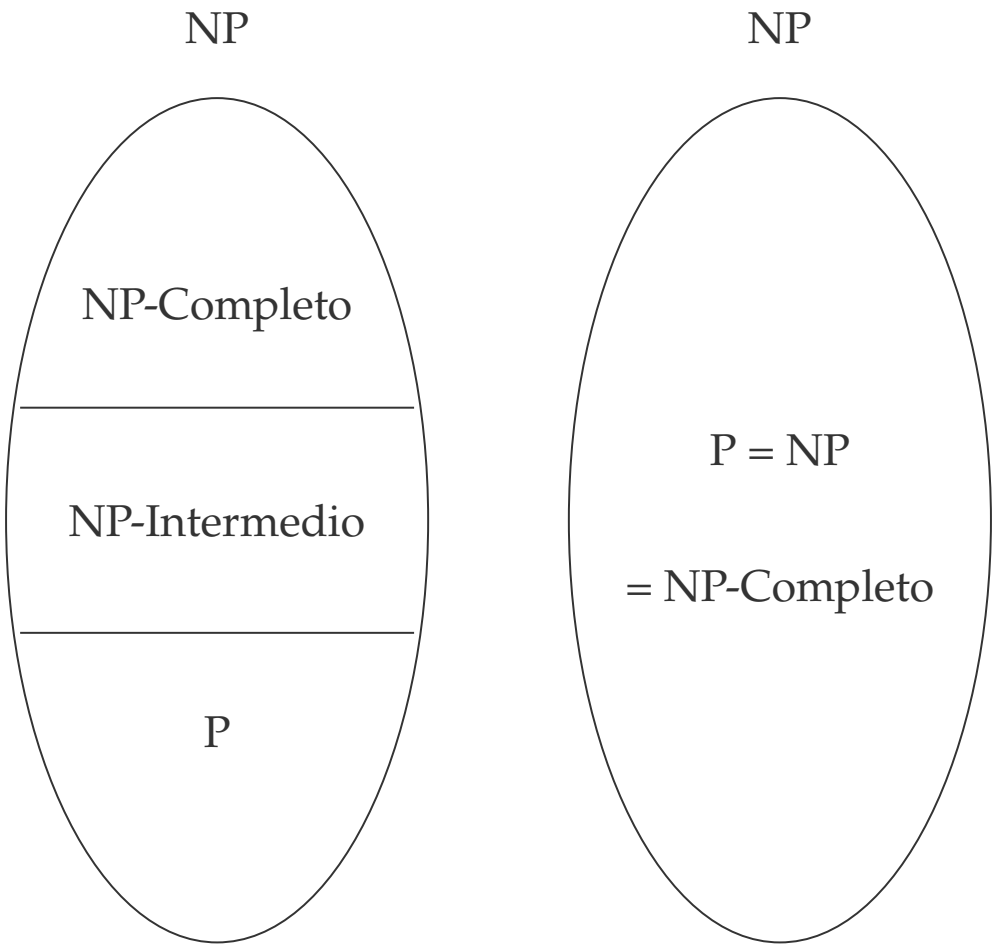
- Si $\bar{\Pi} \in \mathbf{P}$, entonces $\Pi \in \mathbf{P}$.
- Si $\bar{\Pi} \in \mathbf{NP}$, entonces $\Pi \in \mathbf{NP}$.
- Si $\Pi \in$ NP-completo, entonces $\bar{\Pi} \in$ NP-difícil.

Problemas abiertos

Con estas nuevas definiciones tenemos los siguientes problemas abiertos:

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

Las incógnitas...



Dos mapas posibles para las clases de complejidad

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**?

- 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]

¿Qué hacemos si tenemos que resolver **en la práctica** 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, aunque con peor caso exponencial (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.