

TEORÍA DE NP-COMPLETITUD

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella



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.

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

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)

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 .

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).

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)?

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.

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).

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 .

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.

- 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.

- 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**.

- 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.

- Definimos un **problema** dando su entrada y su salida.

- 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?

- 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.

- 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,

- 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**.

Clase P

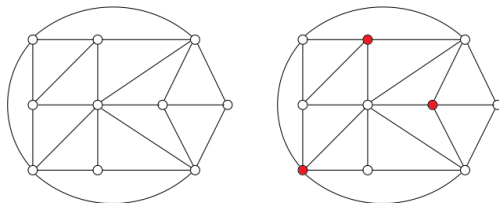
Un problema de decisión Π pertenece a la clase **P** (polinomial determinístico) si existe un algoritmo polinomial que lo resuelve. Es decir, dada una instancia de Π con respuesta **Sí** se puede dar un **cómputo** de longitud polinomial que garantiza que la respuesta es **Sí**.

Clase P

Un problema de decisión Π pertenece a la clase **P** (polinomial determinístico) si existe un algoritmo polinomial que lo resuelve. Es decir, 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.



- 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?

- Dados un grafo $G = (V, X)$ y $k \in \mathbb{N}$, ¿ 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 $\text{MIS} \in \text{NP}$.

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?

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 $\text{SAT} \in \text{NP}$.

Circuito hamiltoniano

1. **Entrada:** Un grafo $G = (V, E)$.
2. **Salida:** ¿Existe una secuencia i_1, \dots, i_n de vértices tal que $i_j i_{j+1} \in E$ para $j = 1, \dots, n - 1$, y además $i_n i_1 \in E$?

Circuito hamiltoniano

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

Problema del viajante de comercio (TSP)

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

Problema del viajante de comercio (TSP)

1. **Entrada:** Un grafo $G = (V, E)$, una función $d : E \rightarrow \mathbb{R}$ 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 .

Observación

$P \subseteq NP$.

Observación

$P \subseteq NP$.

Problema abierto.

¿ $P = NP$?

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.

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.

- 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 .

- 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:

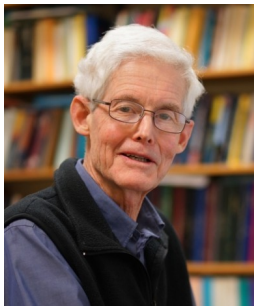
1. $\Pi \in \text{NP}$
2. $\forall \bar{\Pi} \in \text{NP}, \bar{\Pi} \leq_p \Pi$

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 tan difícil como todos los problemas de NP).



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.

¿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 $\Pi \in \text{NP-completo}$ encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que:

¿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 $\Pi \in \text{NP-completo}$ encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que:
 1. $\Pi \in \text{NP}$

¿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 $\Pi \in \text{NP-completo}$ encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que:
 1. $\Pi \in \text{NP}$
 2. $\Pi_1 \leq_p \Pi$

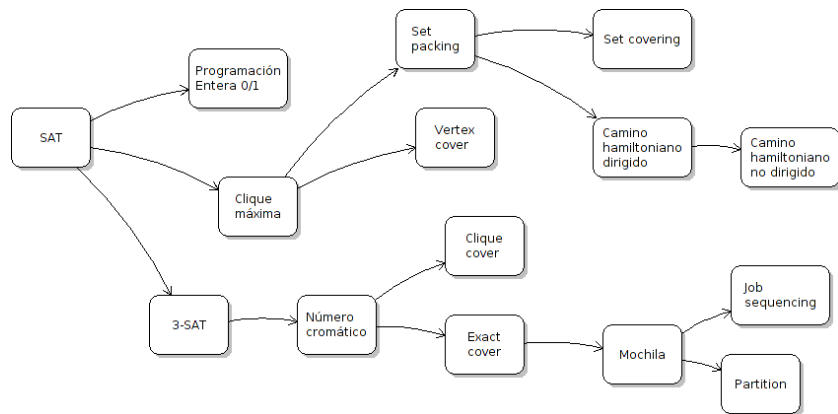
¿Cómo se prueba que un problema es NP-completo?

- 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 $\Pi \in \text{NP-completo}$ encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que:
 1. $\Pi \in \text{NP}$
 2. $\Pi_1 \leq_p \Pi$
- La segunda condición en la definición de problema NP-completo se deriva de la transitividad.



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!



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?

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?

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?

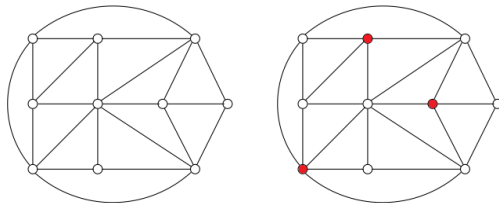
SAT con tres literales por cláusula (3-SAT)

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.

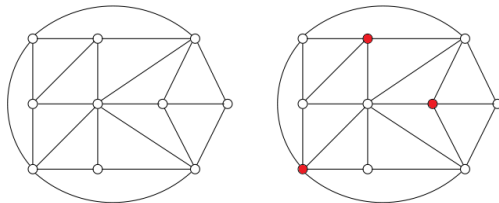
Conjunto independiente máximo



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?

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 (MWIS)

1. **Entrada:** Un grafo G , pesos $w : V \rightarrow \mathbb{R}$ 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 ?

Conjunto independiente de peso máximo (MWIS)

1. **Entrada:** Un grafo G , pesos $w : V \rightarrow \mathbb{R}$ 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.

Conjunto independiente de peso máximo (MWIS)

1. **Entrada:** Un grafo G , pesos $w : V \rightarrow \mathbb{R}$ 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?

Conjunto independiente de peso máximo (MWIS)

1. **Entrada:** Un grafo G , pesos $w : V \rightarrow \mathbb{R}$ 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.

Número cromático

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

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 $s : A \rightarrow \mathbb{Z}_+$.
2. **Salida:** ¿Existe un subconjunto $A' \subseteq A$ tal que
$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Número cromático

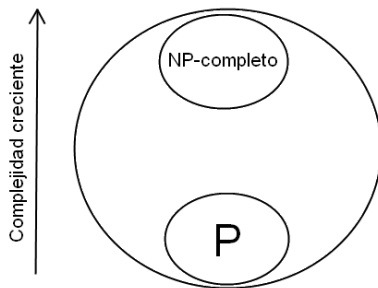
1. **Entrada:** Un grafo G y un número $k \in \mathbb{Z}_+$.
2. **Salida:** ¿Se puede colorear G con k colores?

Set-partitioning

1. **Entrada:** Un conjunto finito A y una función $s : A \rightarrow \mathbb{Z}_+$.
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?



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



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

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

- 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 .

- 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.

- 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 $P=NP$.

- 3-SAT es una restricción de SAT. Ambos son problemas NP-completos.

- 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.

- 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.

- 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.

Si Π es una **restricción** de $\bar{\Pi}$, podemos deducir que:

Si Π es una **restricción** de $\bar{\Pi}$, podemos deducir que:

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

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

- Vimos que circuito hamiltoniano 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?

- Vimos que circuito hamiltoniano 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.

- Vimos que circuito hamiltoniano 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.

- Vimos que circuito hamiltoniano 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.

- 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 **Sí** para las instancias que Π responde **No** y viceversa.

- 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 **Sí** 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. $Y_{\Pi^c} = D_{\Pi} \setminus Y_{\Pi}$

- 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 **Sí** 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?

Proposición

Si $\Pi \in P$, entonces $\Pi^c \in P$.

Proposición

Si $\Pi \in P$, entonces $\Pi^c \in P$.

- Este argumento no aplica para la clase 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.

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 $\Pi \in P$, entonces $\Pi^c \in P$.

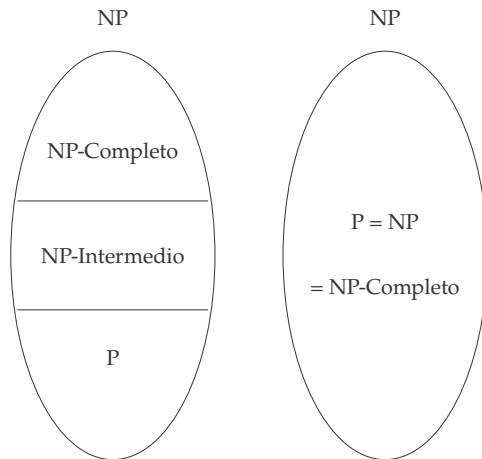
- Este argumento no aplica para la clase NP:
 1. Si un problema Π está en NP no sirve este argumento para demostrar que Π^c está en NP.
 2. Más aún, no se sabe si esto es cierto en general.
- Un problema de decisión pertenece a la clase Co-NP si dada una instancia de **No** y evidencia polinomial de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

Proposición

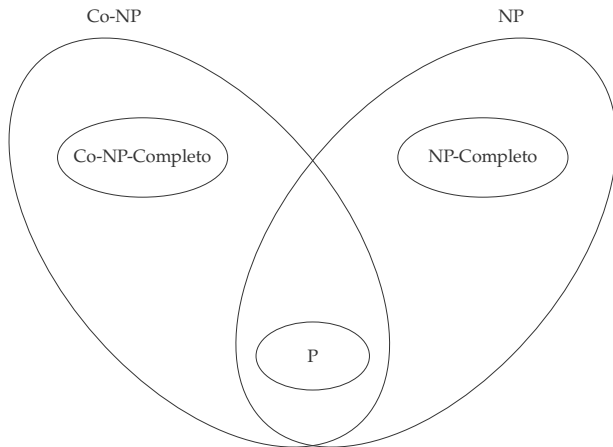
Si $\Pi \in NP$, entonces $\Pi^c \in \text{Co-NP}$.

Con estas nuevas definiciones tenemos los siguientes problemas abiertos:

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



Dos mapas posibles para las clases de complejidad



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

Dado un problema de decisión en NP, tenemos tres posibilidades:

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).

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).

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**!

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 ...

- 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 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.

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]
TRIVIALY 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?

¿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.

¿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**.

¿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).

¿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**.

¿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.

¿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.