Repaso: Complejidad - Problemas NP-Completos

Problemas de Grafos y Tratabilidad Computacional

Teoría de Complejidad

- Un **algoritmo eficiente** es un algoritmo de complejidad polinomial.
- Un problema está **bien resuelto** si se conocen algoritmos eficientes para resolverlo.
- El objetivo es clasificar los problemas según su complejidad.
- Un **problema de decisión** es un problema cuya respuesta es sí o no.
- La clasificación y el estudio de teoría de complejidad se hace para problemas de decisión.

Distintas versiones de un problema de optimización Π

Dada una instancia I del problema Π:

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

Ejemplo: Problema del viajante de comercio

Dado un grafo G con longitudes asignadas a sus aristas:

- 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 **optimización**: Determinar un circuito hamiltoniano de G de longitud mínima.
- Versión de **decisión**: Dado un número k, ¿existe un circuito hamiltoniano de G de longitud menor o igual a k?
- Versión de **localización**: Dado un número k, determinar un circuito hamiltoniano de G de longitud menor o igual a k.

Distintas versiones de un problema de optimización Π

¿Qué relación hay en la dificultad de resolver las distintas versiones de un mismo problema?

Si resolvemos el problema de decisión, podemos en general:

- Resolver el problema de evaluación usando búsqueda binaria sobre el parámetro k.
- Resolver el problema de localización resolviendo el problema de decisión para el parámetro k para una versión reducida de la instancia.
- Resolver el problema de optimización resolviendo el problema de decisión para el valor óptimo para una versión reducida de la instancia.

Problemas intratables

Definición: Un problema es **intratable** si no puede ser resuelto por algún algoritmo eficiente.

Un problema puede ser intratable por distintos motivos:

- El problema requiere una repuesta de longitud exponencial (ejemplo: pedir todos los circuitos hamiltonianos de longitud a lo sumo k).
- El problema es indecidible (ejemplo: problema de la parada).
- El problema es decidible pero no se conocen algoritmos polinomiales que lo resuelvan.

Las clases P y NP

Definiciones:

- Un problema de decisión pertenece a la clase **P (polinomial)** si existe un algoritmo polinomial para resolverlo.
- Un problema de decisión pertenece a la clase **NP (polinomial no-determinísticamente)** si dada una instancia de **SI** y evidencia de la misma, puede ser verificada en tiempo polinomial.

Relaciones entre las clases:

- $P \subseteq NP$
- **Problema abierto:** ¿Es $P = NP$? (problema abierto más importante de teoría de la computación)

Ejemplos de problemas en NP

- Suma de enteros.
- Multipliación de enteros.
- Árbol generador mínimo.
- Clique máxima.
- Camino mínimo entre un par de nodos.
- Problema del viajante de comercio.
- Conjunto independiente de cardinal máximo.
- Problema de satisfabilidad (SAT): Dado un conjunto de cláusulas $C_1, \ldots, C_m$ formadas por literales basados en las variables booleanas $X = \{x_1, \ldots, x_n\}$, determinar si hay una asignación de valores de verdad a las variables de X tal que la expresión $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ sea verdadera.

Máquinas de Turing no-determinísticas (MTND)

Los componentes de una MTND son:

- Cinta infinita dividida en celdas iguales que pueden contener un único símbolo de un alfabeto finito Σ. Es la entrada.
- Cabeza de lectura escritura que apunta a una de las celdas.
- Un conjunto estados posibles $\mathcal{Q}$. En $\mathcal{Q}$ hay un estado inicial q_0 y al menos un estado final. En todo momento, la máquina debe estar en algun estado, llamamos al estado actual q_i.
- Una tabla de quintúplas $T \subset \mathcal{Q} \times \Sigma \times \mathcal{Q} \times \Sigma \times \{-1, +1, 0\}$ que representa al programa.

Máquinas de Turing no-determinísticas (MTND)

La operatoria de una MTND:

1. lectura del símbolo t_i inscripto en la celda señalada por la cabeza.
2. Buscar en T las entradas cuyas primeras dos coordenadas coinciden con (q_i, t_i). En el caso de no hallar ninguna entrada, rechazar la entrada. Si hay $s > 0$ entradas, se debe clonar para tener s hilos/copias de ejecución paralelas, una por cada entrada $(q_i, t_i, q_f, t_f, n_f)$ encontradas. Cada hilo/copia debe continuar a partir de la siguiente instrucción.
3. guarda en esta celda el símbolo t_f.
4. transición al estado q_f.
5. movimiento de la cabeza de lectura/escritura a izquierda (-1), derecha $(+1)$ o inmóvil (0), de acuerdo al valor de n_f.
6. si q_f es un estado final entonces parar todos los hilos/copias de ejecución y aceptar la entrada. Caso contrario, empezar de nuevo el ciclo de la operatoria.

Máquinas de Turing no-determinísticas (MTND)

- Una Máquina de Turing determinística (MTD) es un caso particular de una MTND, donde s siempre vale 1.
- Una MTND resuelve un problema de decisión si alguna de las copias para en un estado de aceptación cuando se ejecuta sobre una instancia de SI y ninguna copia lo hace para instancias de NO.
- La complejidad temporal de una MTND se define como el máximo número de pasos que toma reconocer una entrada aceptable en función de su tamaño.

Clase NP - Otra caracterización

Un problema de decisión está en la clase NP si las instancias de **SI** son reconocidas por una máquina de Turing no-determinística polinomial.

La clase NP se puede definir como el conjunto de problemas de decisión que se pueden resolver por un algoritmo polinomial no-determinístico.

Lema: Si Π es un problema de decisión que pertence a la clase NP, entonces Π puede ser resuelto por un algoritmo determinístico en tiempo exponencial respecto del tamaño de la entrada.

Algoritmo no-determinístico - Conj. Independiente Máximo

Dado un grafo $G = (V, X)$, ¿tiene G un conjunto independiente de tamaño mayor o igual a k?

Selecc(S): función multivaluada que retorna un elemento de S y crea $|S|$ copias del algoritmo, una para cada elemento de S.

Falla: hace que la copia que se está ejecutando pare.

Exito: retorna VERDADERO y hace que todas las copias paren.

$I := \emptyset$
mientras $S \neq \emptyset$ **hacer**
    $v := Selecc(S)$
    $S := S \setminus \{v\}$
    si $\Gamma(v) \cap I = \emptyset$ **entonces** $I := I \cup \{v\}$
    si $|I| \geq k$ **entonces** *Exito*
fin mientras
Falla

Transformaciones polinomiales

Deficiones:

- Una **transformación o reducción polinomial** de un problema de decisión Π_1 a uno Π_2 es una función polinomial que transforma una instancia I_1 de Π_1 en una instancia I_2 de Π_2 tal que I_1 tiene respuesta SI para Π_1 si y sólo si I_2 tiene respuesta SI para Π_2.

- El problema de decisión Π_1 se **reduce polinomialmente** a otro problema de decisión Π_2, $\Pi_1 \leq_p \Pi_2$, si existe una transformación polinomial de Π_1 a Π_2.

Las reducciones polinomiales son transitivas, es decir, si $\Pi_1 \leq_p \Pi_2$ y $\Pi_2 \leq_p \Pi_3$, entonces $\Pi_1 \leq_p \Pi_3$.

Problemas NP-Completos

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

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

Teorema de Cook (1971): SAT es NP-completo.

Problemas NP-Completos

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. $\Pi_1 \leq_p \Pi$

Desde 1971, se ha probado la NP-completitud de muchos problemas usando este método.

Problemas NP-Completos

- CLIQUE (dado un grafo $G = (V, X)$ y un entero positivo k, ¿G tiene una clique de tamaño mayor o igual a k?) es NP-Completo.
  Para demostrar que CLIQUE es NP-Completo, alcanza con probar que:
  1. CLIQUE $\in$ NP.
  2. Para algún problema $\Pi \in$ NP-Completo, $\Pi \leq_p$ CLIQUE.

- Conjunto independiente (dado un grafo G y un entero positivo k, ¿G tiene un conjunto independiente de tamaño mayor o igual a k?) es NP-Completo.

- Recubrimiento de aristas (dado un grafo G y un entero positivo k, ¿G tiene un recubrimiento de aristas de tamaño menor o igual a k?) es NP-Completo.

La clase NP-Difícil

Definición: Un problema de decisión Π es **NP-difícil** si todo otro problema de NP se puede transformar polinomialmente a Π.

(En la práctica esta definición a veces se usa por un abuso de lenguaje también para problemas que no son de decisión y cuya versión de decisión es NP-completa.)

La clase Co-NP

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

- El **problema complemento** de un problema de decisión Π, Π^c, es el problema de decisión que responde al complemento de la decisión de Π.
  Ejemplo: problema de primalidad y problema de número compuesto.

- El problema complemento tiene respuesta NO si y sólo si Π tiene respuesta SI.

- La clase Co-NP es la clase de los problemas complemento de los problemas de la clase NP.

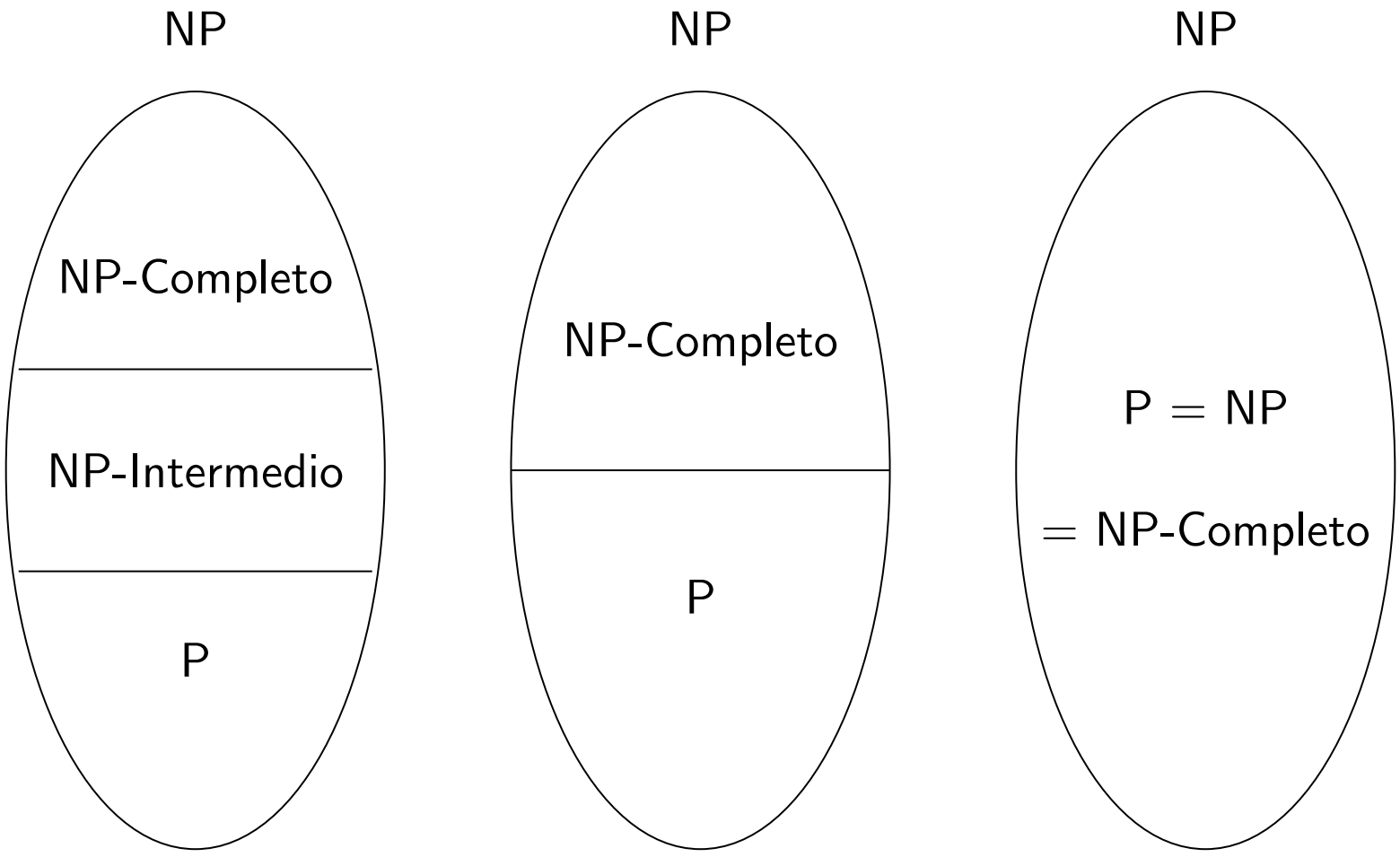
- La clase de los problemas polinomiales (P), está contenida también en Co-NP.

Problemas abiertos de Teoría de Complejidad

Con estas nuevas definiciones tenemos los siguientes problemas abiertos:

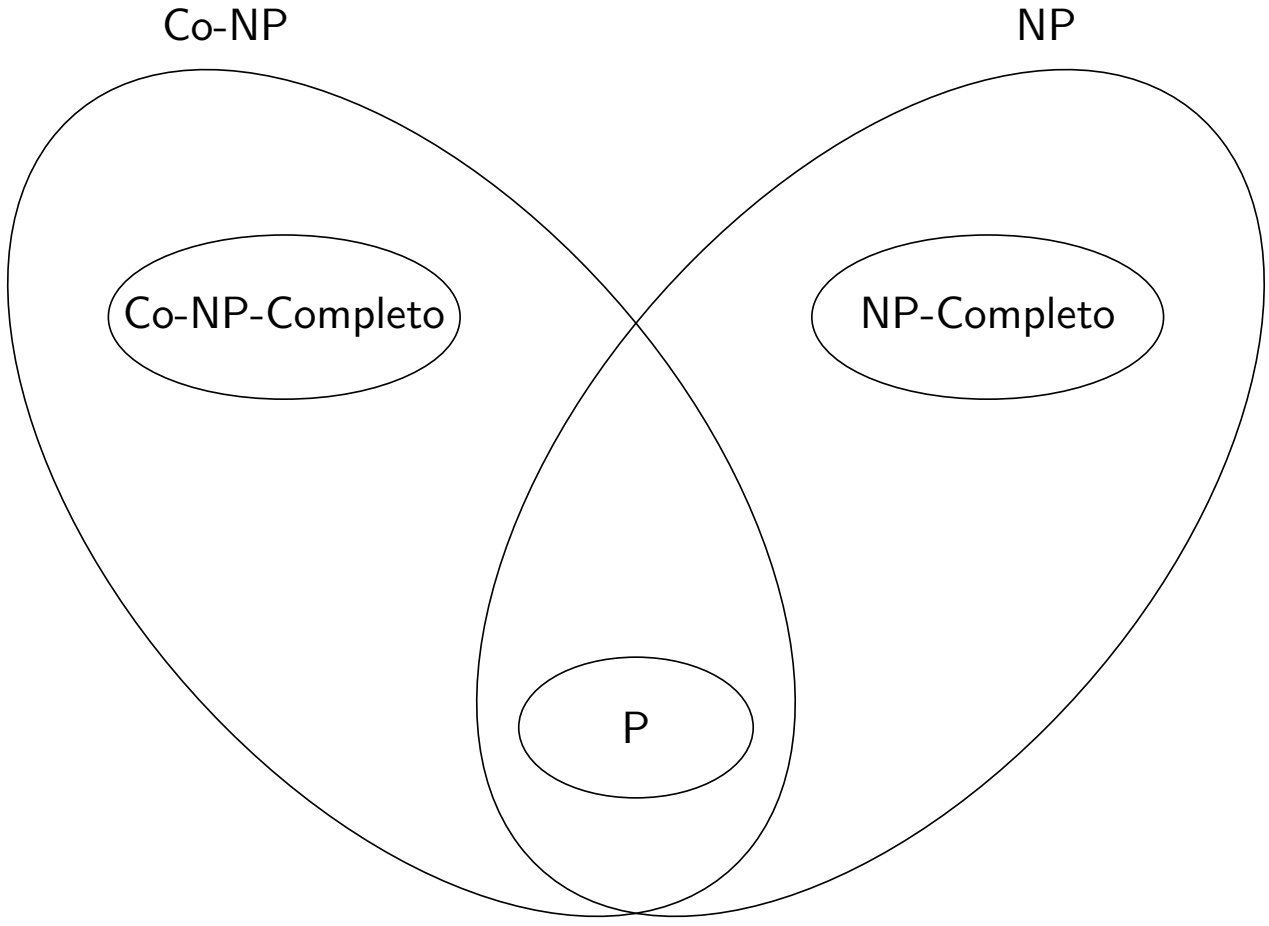
- ¿Es P=NP?
- ¿Es Co-NP=NP?
- ¿Es P=Co-NP $\cap$ NP?

Las incógnitas...



Tres mapas posibles para las clases de complejidad

Las incógnitas...



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

Extensión de un problema

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

- Se dice que $\bar{\Pi}$ es una extensión de Π.
- Si $\Pi \in$ NP-Completo, entonces $\bar{\Pi} \in$ NP-Difícil.

Ejemplos:

- Isomorfismo de subgrafos es una extensión de CLIQUE.
- Viajante de comercio es una extensión de Circuito Hamiltoniano.
- 3-SAT es una restricción de SAT. Sabiendo que SAT es NP-completo, ¿podemos sacar de esto una conclusión sobre la complejidad de 3-SAT?

Algoritmos Pseudopolinomiales

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

Ejemplos:

- Primalidad.
- El problema de la mochila es NP-Completo, sin embargo, existe un algoritmo de complejidad $\mathcal{O}(nB)$ que lo resuelve, donde n es la cantidad de objetos y B el peso máximo que se puede cargar en la mochila (asumiendo que los pesos son números enteros no negativos).

Teoría de Complejidad

- ¿Qué hacer ante un problema del que no sabemos en que clase está?
- ¿Qué importancia tiene saber si un problema está en P o no, desde el punto de vista teórico?.
- ¿Qué importancia tiene la misma pregunta desde el punto de vista práctico, o sea ante una aplicación real que se quiere resolver?
- ¿Qué hacemos si el problema que tenemos en la práctica sabemos que es NP-completo?