Complejidad
Problemas NP-Completos

- **Objetivo:** *queremos algoritmos eficientes para resolver distinto tipo de problemas.*

Algoritmo eficiente = Algoritmo de complejidad polinomial

- **Problema bien resuelto:** *Un problema para el cual conocemos algoritmos polinomiales para resolverlo.*

Queremos ahora clasificar los problemas según su complejidad.

La clasificación y el estudio de la teoría de complejidad se hace para **problemas de decisión**:

problemas cuya respuesta es si o no.

Varias versiones de un mismo problema de optimización Π:

- Versión de optimización**:** *Encontrar una solución del problema* Π *de valor máximo o mínimo.*
- Versión de evaluación: *Determinar el valor de una solución óptima de* Π.
- Versión de localización: *Determinar una estructura que sea una solución factible de* ***Π*** .
- Versión de decisión**:** Dado un entero k, una instancia I de un problema Π , y dada una estructura S: ***es cierto que S es una solución de*** Π ***tal que c(S) < k ? (si el problema es de minimización) o tal que c(S) > k ?(si el problema es de maximización)***

<u>Ejemplo:</u> Problema del viajante de comercio (TSP):

- Versión de optimización: *dado un grafo G con longitudes asignadas a los ejes determinar un circuito hamiltoniano de longitud mínima.*

- Versión de evaluación: *dada una solución óptima o sea un circuito hamiltoniano de longitud mínima determinar el valor de la misma, o sea su longitud.*

- Versión de localización: : *dado un grafo G con longitudes asignadas a los ejes, determinar un circuito hamiltoniano en G.*

- Versión de decisión: *dado un número k, existe un circuito hamiltoniano de longitud* $\leq k$*?.*

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

Problemas intratables

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: *no existe un algoritmo para decidir si dado un algoritmo y un input para el mismo, el algoritmo para o no (Turing 1936)*).

- el problema es decidible pero no se conocen algoritmos polinomiales que lo resuelvan.

Las clases P y NP

Def: Un problema de decisión está en la clase **P** si existe un algoritmo polinomial para resolverlo.

Def: Un problema de decisión está en la clase **NP** si dada una instancia de SI, y una evidencia (certificado o solución) para la respuesta SI, esta puede verificarse en tiempo polinomial.

P : polinomial

NP: nondeterministic polinomial

Es fácil de ver que:

$$\mathbf{P \subset NP}$$

PROBLEMA ABIERTO:

es $\mathbf{P \neq NP}$?

Problema abierto más importante de teoría de la computación.

Ejemplos de problemas en NP:

Están en NP las versiones de decisión de los siguientes problemas:

- Problema de determinar un árbol generador mínimo ($\in$ P!)
- Problema de clique máxima
- TSP
- Determinar un conjunto independiente de cardinal máximo
- Problema de satisfabilidad **(SAT)**: *dado un conjunto de claúsulas C_1,……C_m formadas por variables Booleanas x_1,…x_m determinar si hay una asignación de valores a los x_i tal que la expresión $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ sea verdadera.*

Lema:

Si Π es un problema de decisión que está en la clase NP, entonces una respuesta SI o NO para el problema puede ser obtenida en tiempo exponencial respecto del tamaño del problema.

Otra forma de caracterizar a los problema de la clase NP

La clase NP se puede definir como el conjunto de problemas de decisión que se pueden resolver por un algoritmo polinomial nodeterminístico.

Ejemplo de algoritmo nodeterminístico

Problema: Decidir si un grafo tiene un conjunto independiente de tamaño $\geq k$.

Algoritmo:

Primitiva o módulo adivinador:

Definir $m(S) = \{ 0$ si $V = S$ o $\forall j \in V , \Gamma(j) \cap S \neq \emptyset$

$\{ j$ para $j \in V \setminus S$ y $\Gamma(j) \cap S = \emptyset$

Algoritmo Indep (G,n,k,S)

Inicio

Poner $S = \emptyset$

Mientras $m(S) \neq 0$ hacer

$S = S \cup \{ j\}$

Si $|S| \geq k$ retorna ÉXITO

Si no, FRACASO

Máquinas de Turing no determinísticas

Una máquina de Turing no determinística tiene los mismos componentes que vimos para una máquina de Turing determinística, más un módulo "adivinador" similar a la cabeza de lectura, pero que sólo puede escribir. La máquina actua en dos etapas:

i) En primer lugar el módulo adivinador escribe un string de datos como input al problema.

ii) Después este módulo queda inactivo y la máquina funciona como la máquina de Turing determinística.

Se puede pensar que una máquina de Turing no determinística ejecuta muchas copias en paralelo del mismo algoritmo, una para cada input diferente.

Otra definición de la clase NP

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

Transformaciones polinomiales

Dados dos problemas de decisión Π_1 y Π_2 se dice que Π_1 se reduce o transforma polinomialmente a Π_2 ($\Pi_1 \propto \Pi_2$) si existe un algoritmo polinomial que transforma cada instancia de Π_1 en una instancia de Π_2 y es capaz de traducir la solución de la instancia transformada de Π_2 en una solución de la instancia original de Π_1.

Problemas NP-Completos

Def: Un problema de decisión Π se dice que es **NP-completo** si:

i) $\Pi \in$ NP

ii) todo otro problema Π' perteneciente a NP se puede transformar polinomialmente a Π.

Para que esta definición tenga sentido necesitamos saber que existe algún problema NP-completo….

la respuesta nos la da el siguiente teorema….

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

Importancia de este resultado.

Cómo podemos probar a partir de tener este primer resultado que otros problemas son NP – Completos?

Si Π es un problema de decisión podemos probar que Π es NP-completo encontrando otro problema Π_1 que ya sabemos que es NP-completo y demostrando que :

i) Π está en NP

ii) Π_1 es polinomialmente transformable en Π

Usando este método se ha probado desde 1971 que hay miles de problemas NP-completos…..

- *Qué importancia tiene la clase de problemas NP-completos desde el punto de vista de intentar resolver el problema abierto*

  “es P ≠ NP?” ?.

Prop: Clique es NP-Completo
Dem: *reducir SAT a CLIQUE.*

Prop: Conjunto independiente es NP-Completo
Dem: *usar que CLIQUE es NP-Completo.*

Prop: Recubrimiento de los ejes es NP-Completo
Dem: *usar que Conjunto independiente es NP-Completo.*

Prop: 3-SAT es NP completo.

Dem: El problema 3-SAT es una variante del problema SAT, en el cual cada cláusula tiene exactamente tres literales. Como es una restricción del dominio de SAT, está en NP, y en principio es no más difícil que SAT.

Para probar que 3-SAT es NP-completo, vamos entonces a reducir SAT a 3-SAT. Tomemos una instancia genérica de SAT

$$\varphi = C_1 \wedge \quad \wedge C_m$$

Vamos a reemplazar cada Ci por una conjunción de disyunciones φ_i' donde cada disyunción tenga tres literales, y de manera que φ sea satisfacible si y sólo si

$\varphi_1' \wedge \ldots \quad \wedge \varphi_m'$ lo es.

- Si Ci tiene tres literales:

$$\varphi_i = Ci$$

- Si Ci tiene dos literales, x_1 y x_2, o sea $Ci = (x_1 \vee x_2)$

agregamos una variable nueva y, y definimos:

$$\varphi_i' = (x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \neg y)$$

- Si Ci tiene k≥ 4 literales, agregamos k - 3 variables nuevas con un criterio similar. Por ejemplo si

$$Ci = (x_1 \vee x_2 \vee x_1 \vee x_2 \vee x_5)$$

ponemos

$$\varphi_i' = (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge (\neg y_2 \vee x_4 \vee x_5)$$

Es fácil ver que $\varphi = C_1 \wedge \quad \wedge C_m$ es satisfactible si y sólo si $\varphi_1' \wedge \ldots \quad \wedge \varphi_m'$

Prop: Circuito Hamiltoniano en un grafo orientado es NP-completo.

Dem: *usamos que recubrimiento de los ejes (Vertex cover) es NP-Completo*.

Sea G un grafo en el cual queremos ver si hay un recubrimiento de cardinal ≤ k.

Definimos un grafo G´=(V´,X´) de la siguiente forma:

a) Ponemos tres nodos en G´ por cada eje (i,j) en G: (i,j), {i,j} y (j,i). Y ponemos también k nuevos nodos {1,2,3,…k}

b) El conjunto de ejes estará compuesto por 3 tipos de ejes, o sea

$X' = A \cup B \cup C$, donde :

- Los nodos de la forma (i,j), {i,j} y (j,i) están relacionados entre si por los siguientes 4 ejes:

  $A = \{((i,j), \{i,j\}), (\{i,j\}, (i,j)), ((j,i), (i,j)), (\{i,j\}, (j,i))\}$

ii) Para $h \neq j$ se define el siguiente conjunto de ejes:

$B = \{ ((h,i), (i,j))$ para $(h,i), (i,j) \in X, h \neq j \}$

iii) Se agregan además dos ejes entre cada uno de los nodos 1,2…k y los nodos de tipo (i,j), o sea se agrega:

$C = \{ ((i,j), h), (h, (i,j)), ((j,i), h), (h, (j,i))$ para $(i,j) \in X$, $h = 1,2\ldots k \}$

Ejemplo: (*no están dibujados acá todos los ejes del grafo G´ para no complicar el dibujo)*

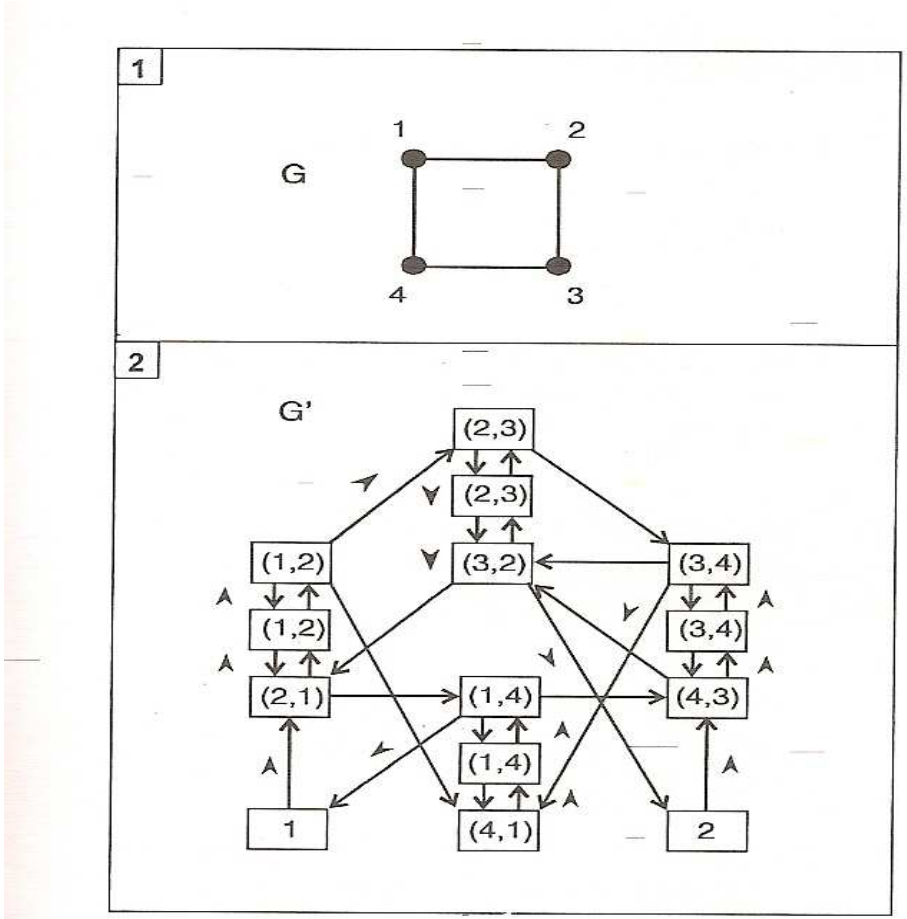


Fig. 2.11 Vertex Cover ∝ Ciclo Hamiltoniano

Se puede ver que si G tiene un recubrimiento de los ejes formado por k nodos (*)

$$V = \{v_1, \ldots\ldots v_k\}$$

y escribimos los ejes de G como

$$X = \{ \{(v_h, w_{h1}), \ldots\ldots\ldots\ldots (v_h, w_{hsh}) \} \text{ para } h=1,2\ldots k \}$$

entonces G´ tiene un circuito hamiltoniano:

$$H = \{1, (v_1, w_{11}), \{ v_1, w_{11} \}, (w_{11}, v_1) \ldots\ldots (v_1, w_{1s1}), \{ v_1, w_{1s1} \}, (w_{1s1}, v_1) \ldots, k, (v_k, w_{k1}), \{ v_k, w_{k1} \}, (w_{k1}, v_k) \ldots, (v_k, w_{ksk}), \{ v_k, w_{ksk} \}, (w_{ksk}, v_k), 1\}$$

(*) si tiene un recubrimiento de cardinal menor a k, se puede extender a uno de k nodos.

Recíprocamente, si G´tiene un circuito hamiltoniano, eliminando en G´ todos los ejes incidentes a los nodos

$$\{1,\ldots\ldots k\}$$

el circuito queda descompuesto en k caminos. Si uno de estos caminos que empieza en (i,j) correspondiente a un eje {i,j} de G , tiene que visitar los nodos {i,j} y (j,i), y después los nodos (i,j´) , {i,j´} y (j´,i) correspondientes a algún eje {i,j´} de G y así sucesivamente.

Este camino corresponde entonces a un nodo I de G que cubre a los ejes {i,j} , {i,j´} , etc.

Como el circuito pasa por cada {i,j} una sóla vez, cada eje {i,j} queda cubierto por uno de los k nodos de $\{1,\ldots\ldots k\}$ *, que por lo tanto es un recubrimiento de tamaño k.*

Prop: Coloreo es NP-completo

Dem:

- El problema de coloreo es NP.
- Para probar que coloreo es NP-completo, vamos entonces a reducir SAT a coloreo.

Tomemos una instancia genérica de SAT

$$\varphi = C_1 \wedge \ldots \wedge C_m$$

Vamos a construir un grafo G y determinar un número k de manera que φ sea satisfactible si y solo si G se puede colorear con k-colores.

Construimos el grafo G de la siguiente forma. El conjunto de nodos es $V = V_1 \cup V_2 \cup V_3$

- V_1: tiene un nodo por cada variable negada y afirrmada, todos adyacentes entre si.

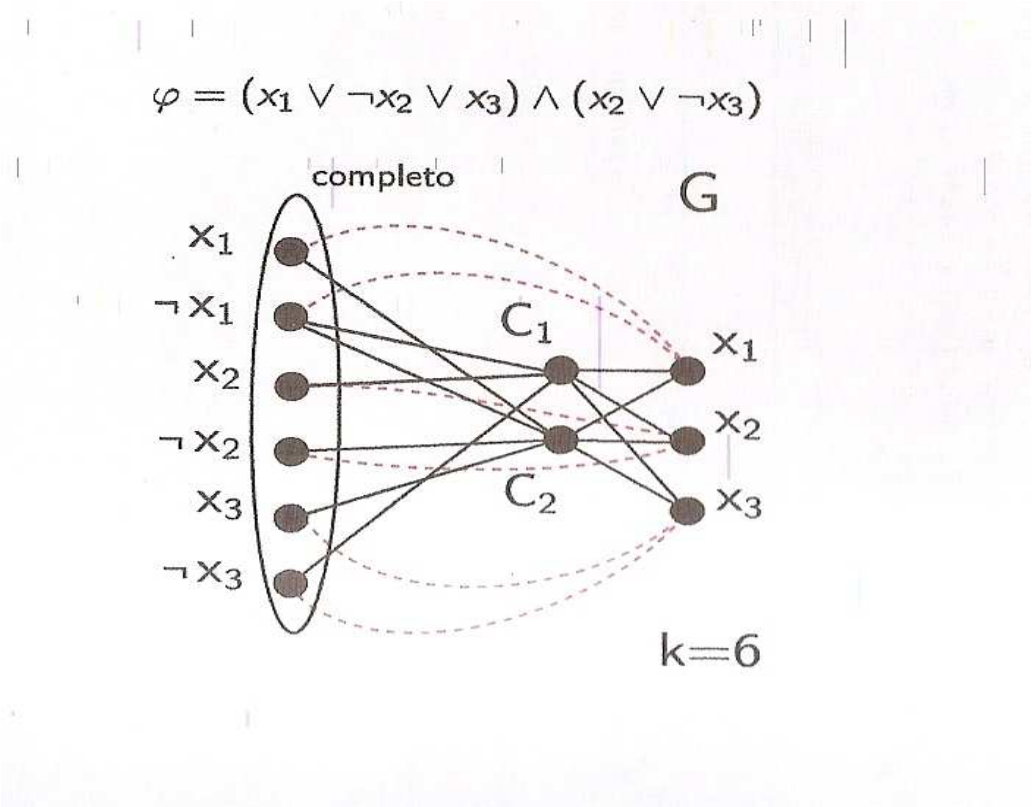
- V_2: tiene un nodo por cada cláusula, adyacente a los literales de V_1 que no aparecen en la cláusula.

- V_3: tiene un nodo por cada variable, adyacente a todo V_2 y a los literales de V_1 correspondientes a las otras variables.

Tomamos k = dos veces la cantidad de variables.

Se puede probar que si G es coloreable con k colores entonces podemos tener una asignación de los valores a las variables que hace que la expresión φ sea verdadera. (EJERCICIO)

Ejemplo (*no están dibujados acá todos los ejes entre los elementos de V_3 y los de V_1, las líneas punteadas corresponden a los ejes que NO ESTAN en el grafo)*



La clase NP-Hard

Def:

Se dice que un problema de decisión es **NP-hard** si todo otro problema de NP se puede transformar polinomialmente a Π.

(o sea si cumple la parte ii) de la definición de NP-completo)

(En la práctica esta definición se usa a veces por un abuso de lenguaje también para problemas que no son de decisión y cuya "versión de decisión" es NP-completo o para indicar que el problema es al menos tan difícil como un problema Np-completo….)

Los problemas Co-NP y los problemas P

Def: Un problema de decisión es Co-NP si para todas las instancias que tienen una respuesta NO esta puede ser verificada en tiempo polinomial.

Problema complemento de un problema de decisión Π : Es el problema de decisión relacionado con Π que responde al complemento de la decisión de Π.

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

O sea la clase CO-NP es la clase de los problemas complemento de los problemas de la clase NP.

Ejemplo: problema de primalidad y problema de número compuesto.

La clase de los problemas polinomiales (P), está contenida también en Co-NP, porque se puede usar el algoritmo polinomial para verificar dada una instancia, cuál es la respuesta correcta.

Problemas abiertos de Teoría de Complejidad

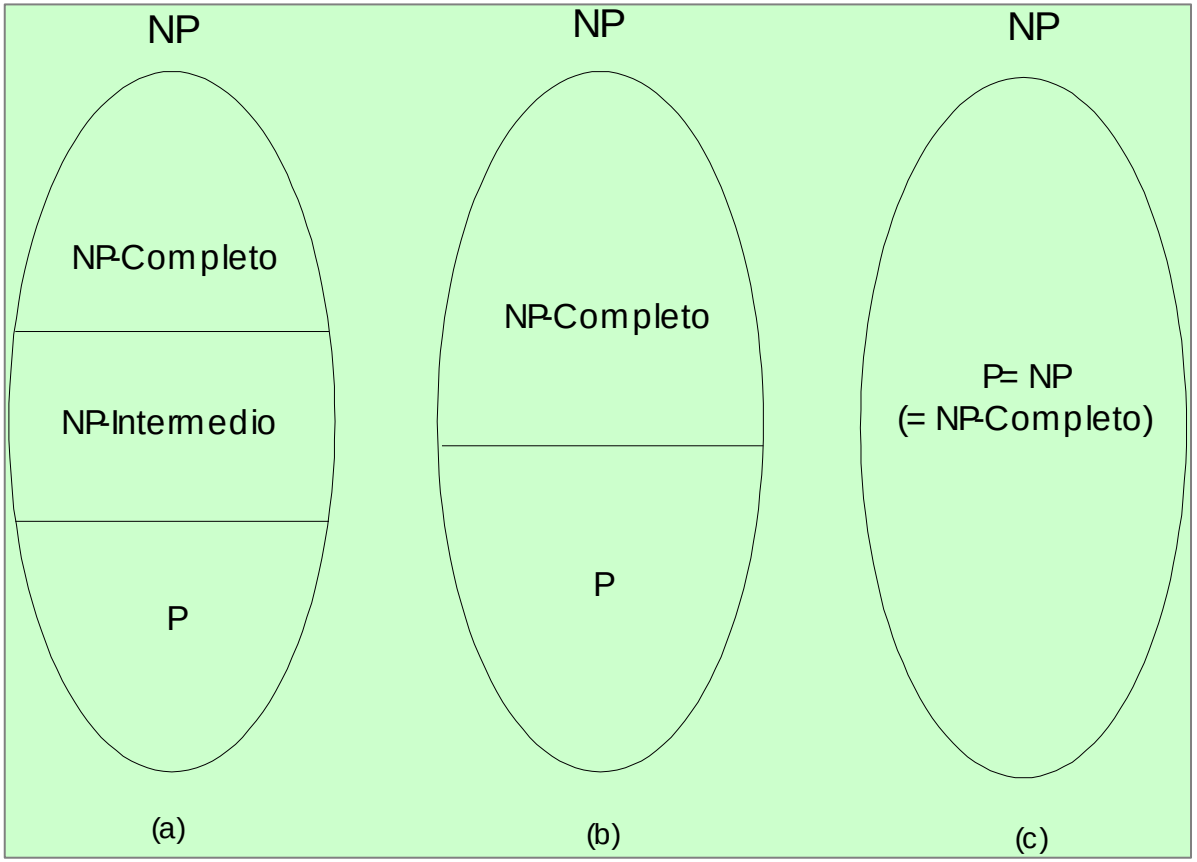
Con estas nuevas definiciones vemos que tenemos los siguientes problemas abiertos:

¿Es P=NP?

¿Es Co-NP=NP?

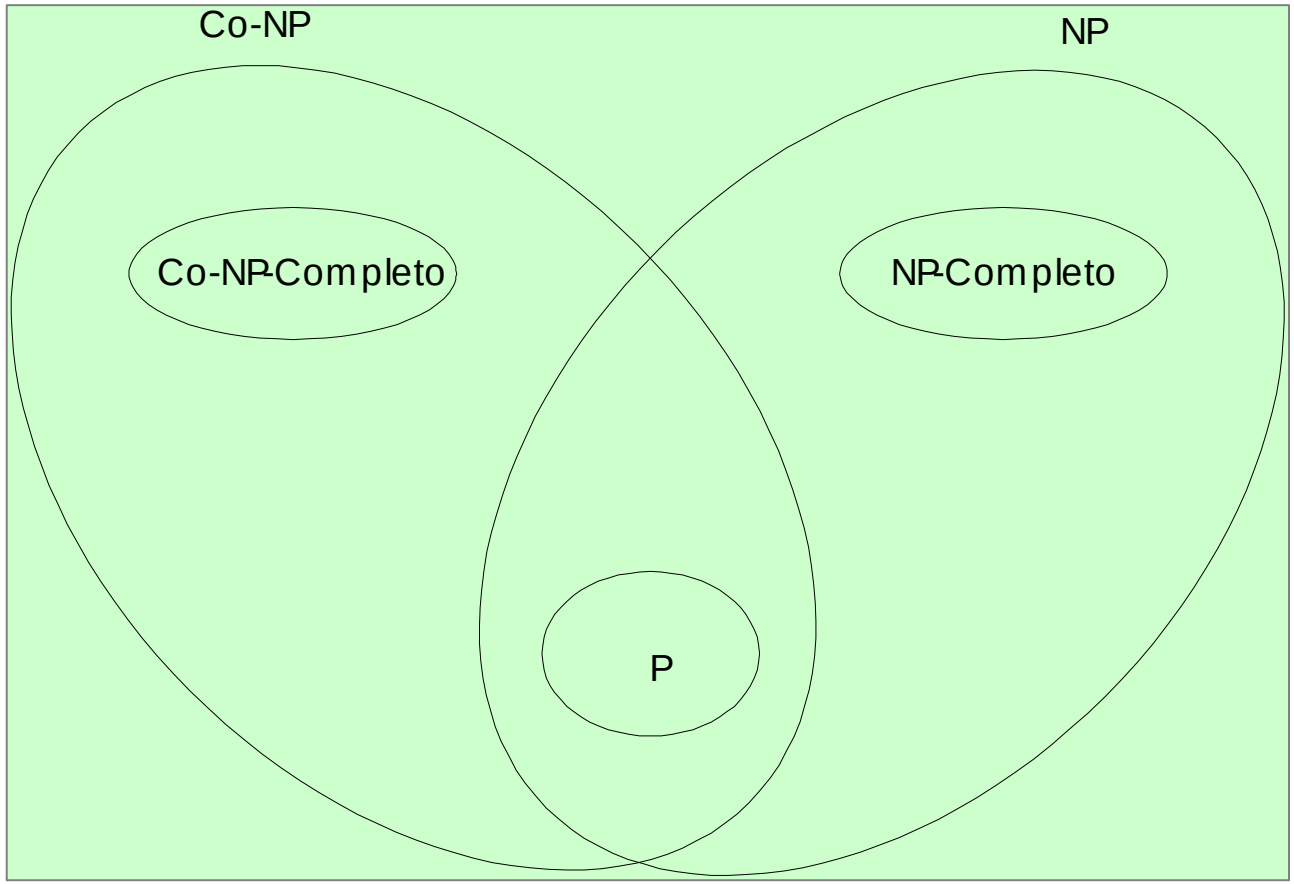
¿Es P=Co-NP$\cap$NP?

Las incógnitas.....



Tres mapas posibles para las clases de complejidad

Las incógnitas.....



Esta sería la situación si se probara que $P \neq NP$, $NP \neq Co\text{-}NP$, $P \neq Co\text{-}NP \cap NP$

Def: El problema Π´ es una **restricción** de un problema Π si el dominio de Π´ está incluído en el de Π. Y en este caso se dice que Π es una **extensión** de Π´.

Qué implican estas definiciones en función de decidir cual es la complejidad de un problema?

Ejemplos: Los siguientes problemas son NP-Completos

- **Isomorfismo de subgrafos**: CLIQUE es una restricción de Isomorfismo de Subgrafos.
- **Viajante de comercio**: Circuito Hamiltoniano es una restricción del problema del Viajante de Comercio.

Ejemplo: 3SAT es una restricción de SAT, y SAT es NP-completo.
Podemos sacar de esto una conclusión sobre la complejidad de 3SAT?

Algoritmos Pseudopolinomiales

Un algoritmo para resolver un problema Π es pseudopolinomial cuando la complejidad del mismo es polinomial en función del tamaño de la entrada de Π codificada en unario.

Ejemplo: se puede probar que el problema de la mochila es NP completo, sin embargo existe un algoritmo de programación dinámica de complejidad O(nB) que lo resuelve. (n es la cantidad de objetos y B el peso máximo que se puede cargar).

Entonces:

- *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 requiere resolver?.*

- *Qué hacemos si el problema que tenemos en la práctica sabemos que es NP-completo?.*

FIN