Ejemplo clásico de problema NP: Viajante de comercio

Dado un conjunto de ciudades y carreteras que las unen, y una longitu máxima d, hay que decidir si existe alguna trayectoria que pase una sola vez por cada ciudad recorriendo una distancia no mayor que d.

Este problema puede resolverse con una MTN de la siguiente manera:

1)El input w contiene el nro. n de ciudades, las distancias que las separan y la longitud máxima d permitida

2)De manera no determinística se escribe una permutación cualquiera de los nros entre 1 y n. Puede hacerse así: Se escriben 2 listas, la primera con los nros ordenados de 1 a n y la segunda vacía. Se elige de manera no determinista uno de la primera lista que se tacha y pasa a la segunda. Cada ciclo tiene un costo de O(n), que se repite n veces hasta completar la permutación ($O(n^2)$)

3)Se calcula la distancia del recorrido elegido. Suponiendo un costo O(n) para localizar en la entrada la distancia entre 2 ciudades cualquiera, el costo de calcular la distancia del recorrido es $O(n^2)$

4)Si la distancia del recorrido es menor que d para en q_A sino para en q_R.

Def. Co-NP={L / $L^C \in$ NP} // Usamos L^C para referirme a L complemento

Teorema. Si $L \in P \rightarrow L \in (NP \cap Co\text{-}NP)$

Si $L \in P \rightarrow L \in NP$ (por def. de P y NP)

Si $L \in P \rightarrow L^C \in P$ (simplemente intercambiando qa y qr en la MTD

$\rightarrow L^C \in NP$ (por def de P y NP)

$\rightarrow L \in Co\text{-}NP$ (por def Co-NP)

$\rightarrow L \in (NP \cap Co\text{-}NP)$

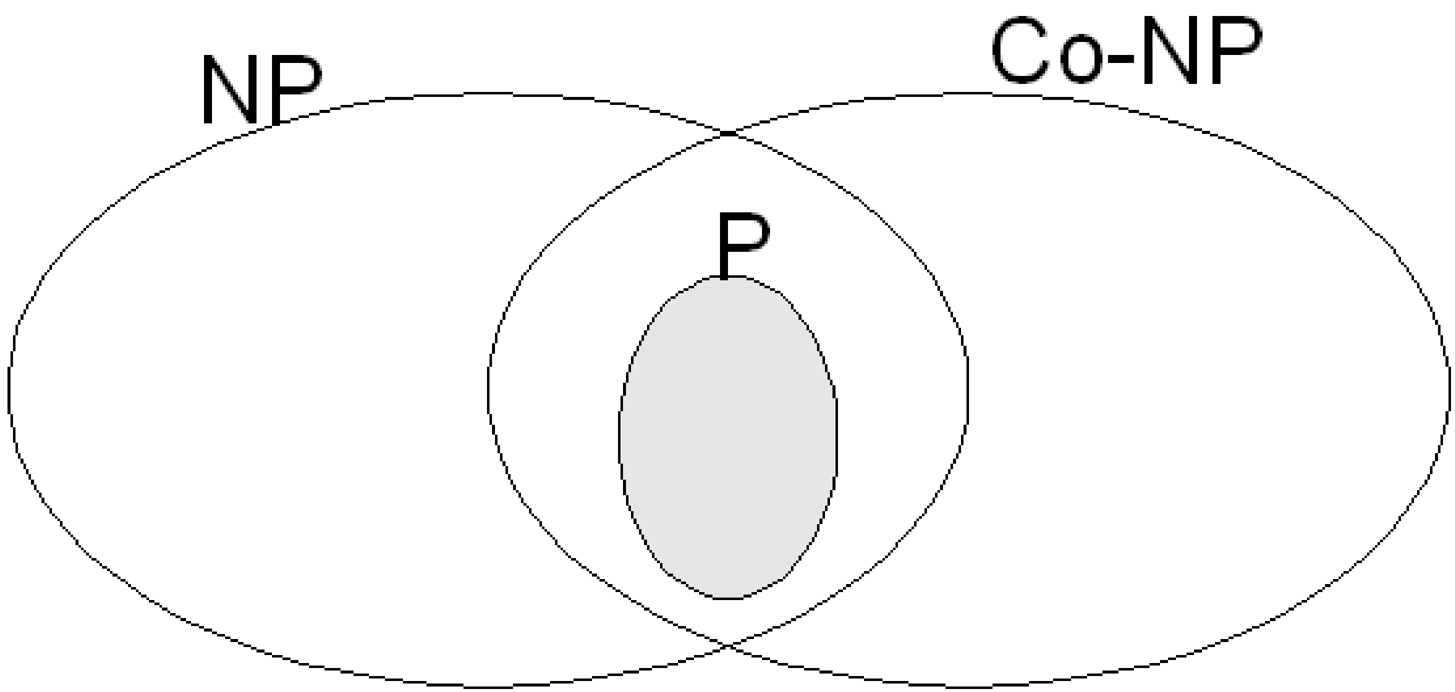
Nota: No se sabe cómo está relacionados NP y Co-NP. No se puede asumir que son iguales, reconocer el complemento de un lenguaje, intentando intercambiar estados qa y qr de su MTN no funciona, una MTN puede aceptar teniendo computaciones que terminan en qr pues alcanza que una sola termina en qa.

Si se sabe que si $NP \neq Co\text{-}NP \rightarrow P \neq NP$

Dem. Si $P=NP \rightarrow NP=Co\text{-}NP$ (porque $P=Co\text{-}P$)

entonces $NP \neq Co\text{-}NP \rightarrow P \neq NP$ (por contrarecíproca)

Esta podría ser la situación:

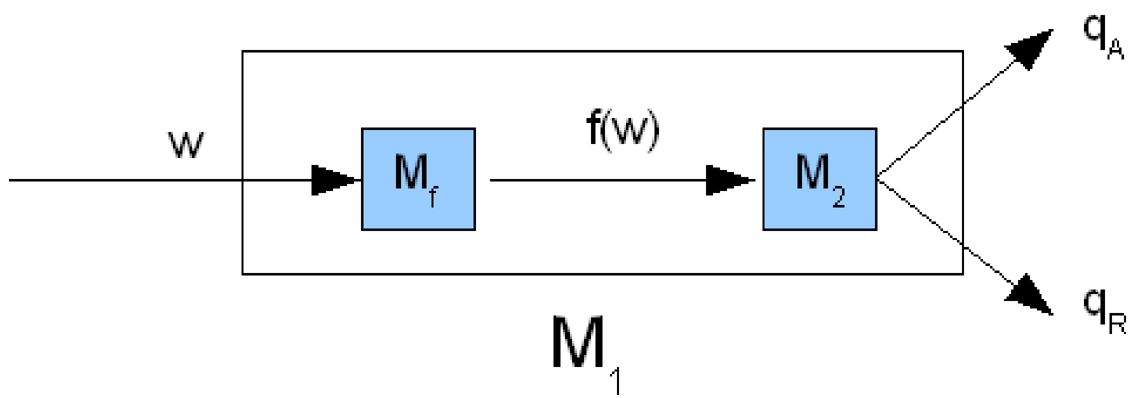


Reducción polinómica.

Sean L_1 y L_2 dos lenguajes incluidos en Σ^*, decimos que L_1 se reduce polinómicamente a L_2 (se denota $L_1 \alpha_p L_2$) sii $L_1 \alpha L_2$ y además la función de reducibilidad f es computada por una MTD que trabaja en tiempo polinomial ($f \in P$)

Teorema Si $L_1 \alpha_p L_2$ y $L_2 \in P \rightarrow L_1 \in P$

Dem. Sea M_f la MTD que computa f en tiempo polinomial. Construimos M_1 una MTD tal que $L_1=L(M_1)$ de la siguiente manera:



$L_1=L(M_1)$?

Dado que $w \in L_1$ sii $f(w) \in L_2$ se tiene que $L_1= L(M_1)$. Falta ver que M_1 trabaja en tiempo polinomial

Sea $n = |w|$

M_f computa f(w) en a lo sumo cn^k pasos,

por lo tanto $| f(w) | <= cn^k$

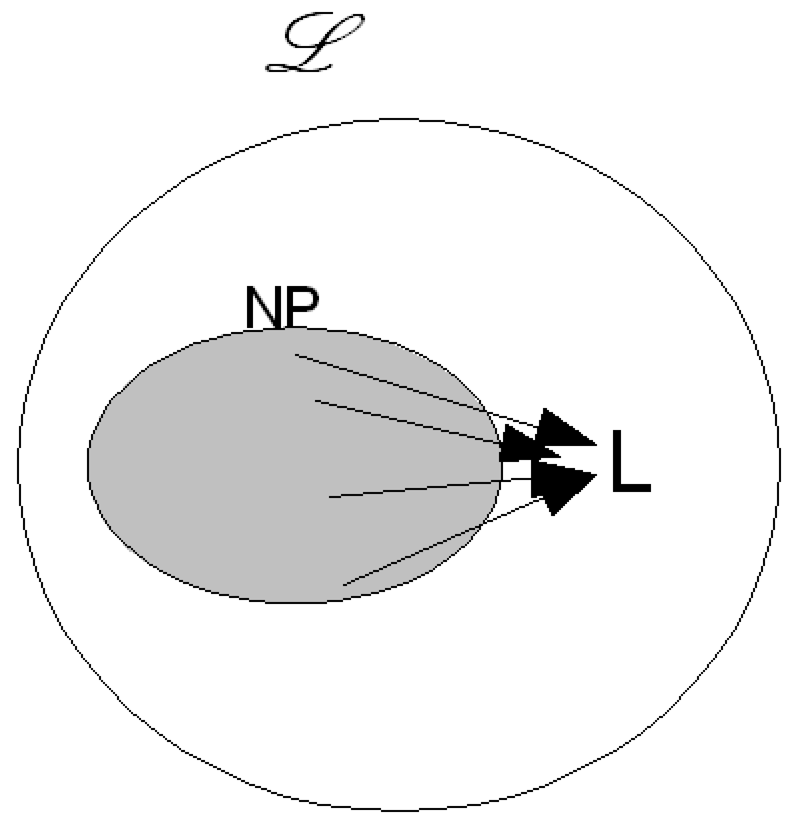
por lo tanto M_1 hará a lo sumo $cn^k + c_2(cn^k)^{k2}$

por lo tanto M_1 trabaja en tiempo $O(n^{k1})$

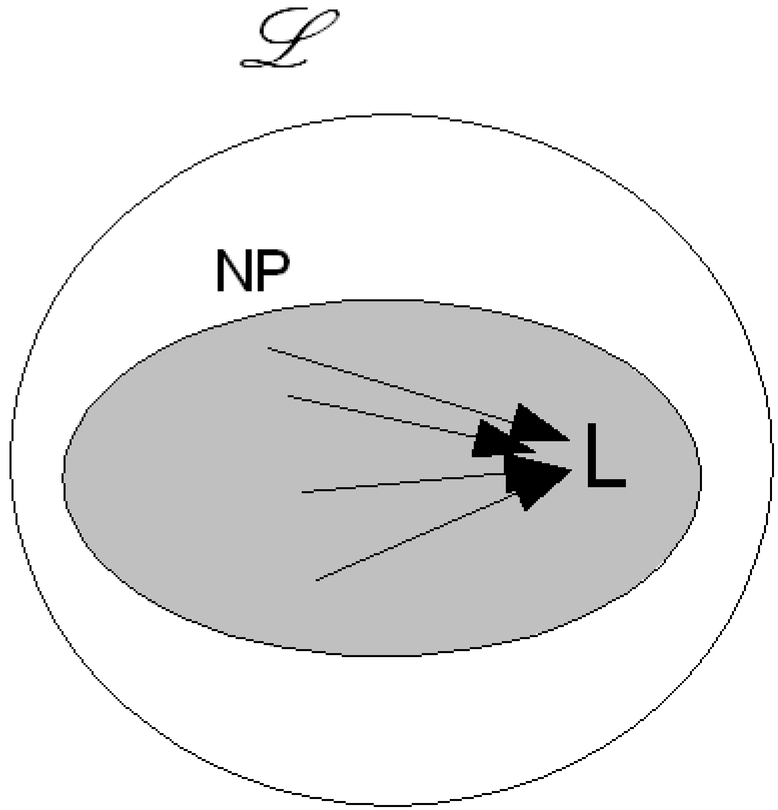
Teorema. Sea L un lenguaje tal que $\emptyset \subset L \subset \Sigma^*$ entonces para cualquier lenguaje L' perteneciente a P, vale que $L' \alpha_p L$

Dem. Como $\emptyset \subset L \subset \Sigma^*$ entonces existe algún $x \in L$ y algún $z \notin L$. La reducción consiste para cada w, resolver L' con su MTD polinomial, y asignar x o z según la respuesta sea sí o no.

Def. L $\in$ NPH (NP-Hard) sii para todo L' $\in$ NP se cumple L' α_p L



Def. $L \in NPC$ (NP-Completo) sii $L \in NPH$ y $L \in NP$.



Teorema. Si $L \in NPC$, se cumple que si $\mathbf{L} \in \mathbf{P} \rightarrow \mathbf{P} = \mathbf{NP}$

Dem. Si $L \in NPC$ entonces para todo $L' \in NP$ se cumple

$L' \alpha_p L$ (por def. de NPC)

Si $L \in P$ entonces para todo $L' \in NP$ se cumple

$L' \in P$ (por teorema anterior)

Por lo tanto $NP \subseteq P$

Por lo tanto $P=NP$

Teorema. Sean $L_1, L_2 \in NP$. Si $L_1 \in NPC$ y $L_1 \alpha_p L_2$ entonces $L_2 \in NPC$

Dem. Como $L_1 \in NPC$, vale que para todo $L \in NP$ existe la reducción $L \alpha_p L_1$, por propiedad transitiva α_p de se cumple que para todo $L \in NP$ existe la reducción $L \alpha_p L_2$ por lo tanto $L_2 \in NPC$

Para practicar: Demostrar que α_p es transitiva

Ejemplo de lenguaje NPC: **SAT**

Literal: es una variable proposicional o la negación de una variable proposicional.

FNC: fórmula normal conjuntiva, es una conjunción de una disjunción de literales.

Enunciado del problema SAT: Dada una fórmula proposicional en FNC, determinar si es satisfactible, es decir si existe alguna asignación de verdad para las variables que haga verdadera la fórmula.

Teorema (Cook/Levin) SAT es NPC.

Primero se prueba que SAT pertenece a NP y luego se toma un lenguaje arbitrario L perteneciente a NP y se prueba que L α_p SAT. Como L pertenece a NP sabemos que existe una MTN M tal que M(x) para (aceptando o rechazando) en tiempo polinomial p(|x|). A continuacion se define una fbf F(M; x) con el objetivo que F(M; x) sea satisfacible si y solo si M(x) termina en el estado aceptador.

Más de 1000 problemas de dominios muy diferentes, y con variadas aplicaciones se han probado que pertenecen a NPC haciendo reducciones desde un lenguaje que ya se conozca pertenece a NPC. Para ninguno de ellos se ha podido encontrar una solución polinomial.

Ejemplos:

SAT α_p 3-SAT

3-SAT α_p VC={(G,k)/G es un grafo que tiene un cubrimiento de vértices de tamaño k}

VC α_p Lclique={(G,k)/ G tiene un clique de tamaño k}, clique es un subgrafo totalmente conectado

Sea G=(V,E) un grafo y sea C$\subset$V un subconjunto de vértices. Decimos que C es un *cubrimiento de vértices* si cualquier e$\in$E tiene un extremo que pertenece a C.

P = NP ?

Por varias décadas de estudio los investigadores no han podido responder a esta pregunta. Por ello se cree que P ≠ NP es mas plausible que P = NP.