CoNP y problemas de funciones

Gabriela Vargas y Dalia Camacho

Acordeón

Satisfacibilidad

Una fórmula Φ en FNC es satisfacible si existe una asignación de valores verdaderos y falsos para las variables $x_1, x_2, \ldots, x_n$ tal que Φ se evalúa a verdadero.

Testigos

En teoría de complejidad, un testigo (también llamado certificado) es un string que certifica que $x \in L$ (ejemplo, problema HP en donde M se Detiene en a lo más t pasos, t es el testigo)

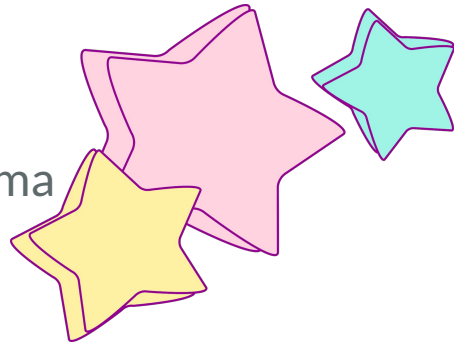
NP y coNP

CoNP

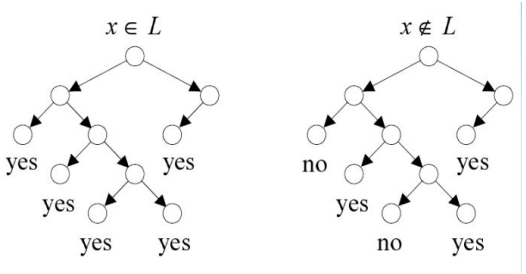
Clase de complejidad que contiene al complemento de todos los problemas clasificados en NP.

La clase NP está compuesta por los problemas que tienen un certificado sucinto (también llamado testigo polinómico) para todas las instancias cuya respuesta es un SÍ.

coNP entonces está compuesta por aquellos problemas que tienen descalificaciones sucintas, es decir, una instancia "no" de un problema en coNP posee una breve prueba de que es una instancia "no"



Ejemplos de problemas coNP



VALIDITY

Dada una expresión Booleana Φ , determinar si es válida (satisfacible para todas las asignaciones de valores de verdad).

coHAMILTONPATH

Conjunto de todos los grafos que NO tienen un circuito Hamiltoniano.

Estos problemas no están en NP, lo que tienen es una Máquina polinomial no determinista en la que la respuesta es afirmativa al problema si TODAS las opciones responden SI

$\{L \mid \exists \text{ polyn.time NTM } M: L=\{w \mid \text{all computations paths of } M(w) \text{ accept}\}\}$

P ⊆ coNP

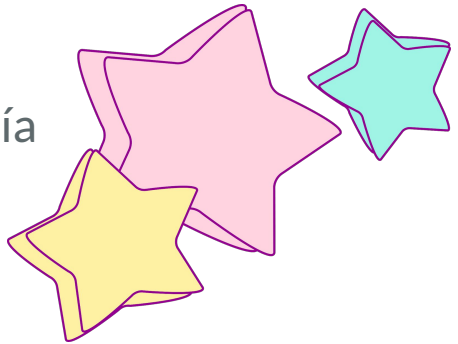
Demostración:

P ⊆ NP es una clase de complejidad determinística y, por lo tanto, es cerrada bajo complemento. Por esto concluimos que P ⊆ coNP□

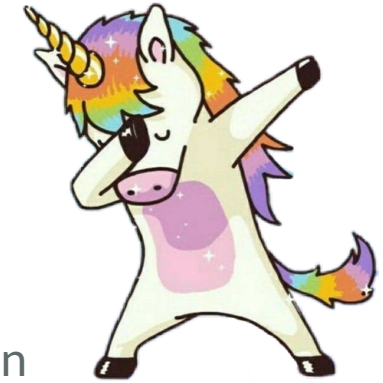
Cualquier lenguaje L ∈ coNP es reducible a VALIDITY

Idea de demostración:

Si L ∈ coNP, entonces Lcomp ∈ NP y por lo tanto, existe una reducción R de Lcomp a SAT. La reduccion de L a VALIDITY sería R'(x) = ¬ R(x) para una cadena x ∈ Lcomp



NP=coNP??



Es una pregunta fundamental que aún no se resuelve.
Si se llegara a demostrar que P=NP, entonces también tendríamos NP=coNP, ya que P es cerrada bajo complemento. Aunque también podríamos tener P≠NP y aún así tener NP=coNP, pero se cree que son diferentes.

¿Por qué?

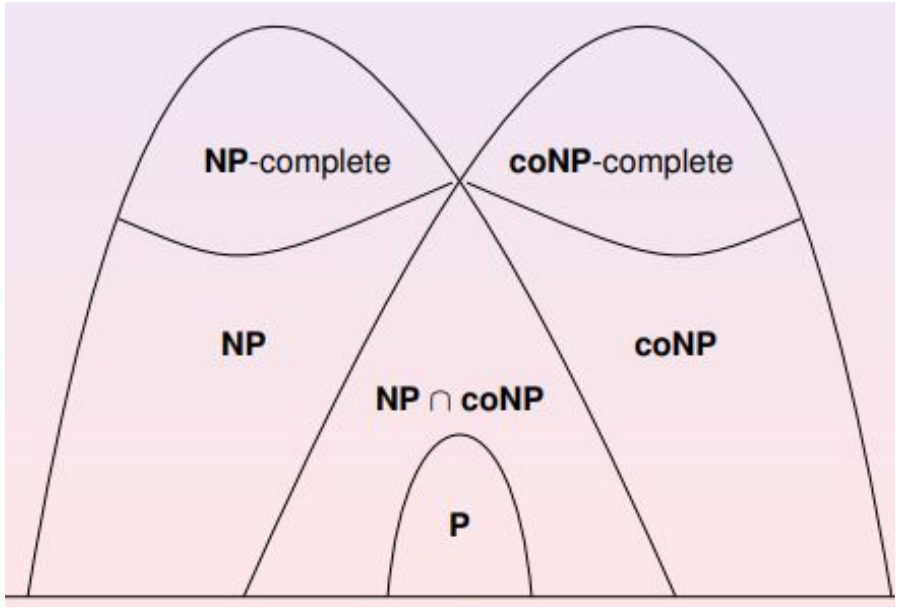
1. Todos los esfuerzos por encontrar demostraciones cortas para VALIDITY han fallado.
2. No se conocen caracterizaciones de grafos no-Hamiltonianos que nos permitan demostrar la ausencia de un camino Hamiltoniano de manera sucinta.

La clase NP∩coNP

Si los problemas NP son los que tienen certificados sucintos y los problemas coNP, en cambio, tienen descalificaciones sucintas NP ∩ coNP será la clase que tenga ambas! (Pero no al mismo tiempo :P)

Esto quiere decir que una instancia de NP ∩ coNP tendrá la siguiente propiedad: O posee un certificado sucinto, en este caso sería para una instancia “SÍ”; o posee una descalificación sucinta, en cuyo caso es una instancia “NO”.

Entonces, vemos que $P \subseteq NP \cap coNP$, pero existen problemas en NP ∩ coNP que NO se sabe si están en P.



Prop: Si L es un lenguaje NP-completo, entonces su complemento Lcomp= Σ^*-L será coNP completo.

Prop: Si un problema coNP $\subseteq$ NP, entonces NP=coNP

Primalidad

El problema de determinar si un número es primo está en NP y en coNP.

¿coNP?

- Un número primo *p* es tal que **para toda** n tal que $1 \leq n < p$ el máximo común divisor entre *n* y *p*. Dicho de otra forma *n* divide a *p* si y solo si $n = 1$.
  - Dado un entero *x* si existe $n \neq 1$ que divide a *n*. Entonces se tiene un testigo para indicar que *x* no es primo.
  - El cuantificador **para todo** nos indica que es un problema coNP

Primalidad

El problema de determinar si un número es primo está en NP y en coNP.
¿NP?

- Un número $p>1$ es primo si y sólo si **existe** r tal que $1<r<p$, $(r^{p-1} \bmod p)= 1$ y para todo divisor primo q de $(p-1)$ se cumple que $(r^{(p-1)/q} \bmod p)\neq 1$.
  - Para que sea NP se debe mostrar la existencia de un testigo que cumpla con las características mencionadas y que sea comprobable en tiempo polinomial.
  - El testigo no sólo es r, se tienen que dar r, todos los divisores primos de (p-1) y el testigo que cada uno de estos es primo.
  - El tamaño del testigo es $O(4\log^2 p)$
  - Verificar $(r^{p-1} \bmod p)= 1$ y son $(r^{(p-1)/q} \bmod p)\neq 1$ $O(n^3)$, como esto se debe repetir para cada q el tiempo total es $O(n^4)$.
  - Dado un testigo éste se escribe en espacio logarítmico y se verifica en tiempo polinomial por lo tanto es NP.

Problemas de funciones

Hemos visto problemas de decisión...

¿Pero si quiero conocer un resultado que no sea sí o no?

Ejemplo:

- No sólo quiero saber si una fórmula φ en FNC es satisfacible, quiero conocer la asignación de verdad que la hace satisfacible.
- Entonces definimos el problema de funciones FSAT

El problema FSAT

Sea $\phi(x_1, x_2, \ldots, x_n)$ una expresión Booleana:

- Si Φ es satisfacible, entonces regresa una asignación satisfacible para Φ.
- Regresar “no” en otro caso.

Se llama a una función de ordenamiento (function of sorts). Para cada input Φ, esta función puede tener distintos valores (cualquier asignación satisfacible), o ninguna, en caso de que regresemos “no”.

Entonces, si SAT puede resolverse en tiempo polinomial, también FSAT podrá resolverse en tiempo polinomial y lo mismo aplica de regreso.

Demostración

Si existiera un algoritmo polinomial para SAT, entonces:

Dada una expresión Booleana $\phi(x_1, x_2, \ldots, x_n)$, nuestro algoritmo primero pregunta si Φ es satisfacible. Si la respuesta es "no", entonces el algoritmo se detiene y regresa "no".

Esto se pone interesante cuando la respuesta es "Sí" y nuestro algoritmo tiene que devolver una asignación verdadera.

El algoritmo entonces hace los siguiente: Evalúa la dos asignaciones $\Phi[x_1=\text{True}]$ y $\Phi[x_2=\text{False}]$ y elige la que es satisfacible. Entonces el valor de x_1 será sustituido en Φ y posteriormente se evaluará con x_2.

Demostración

Entonces es claro que con a lo más 2n llamadas al algoritmo polinomial de satisfacibilidad, todas con expresiones más simples que Φ, el algoritmo encontrará una asignación verdadera satisfacible.□

Este algoritmo implementa la propiedad de auto-reducibilidad inherente a SAT y a otros problemas NP-completos.

Se dice que un problema es auto-reducible si el problema de búsqueda se reduce (por Reducción de Cook) a un problema de decisión en tiempo polinomial. En otras palabras, hay un algoritmo para resolver el problema de búsqueda el cual tiene polinomialmente muchos pasos, donde cada paso es,
(A) Un paso computacional convencional, o
(B) Una llamada a la subrutina para resolver el problema de decisión.

Fuente: https://courses.engr.illinois.edu/cs473/fa2011/lec/24_notes.pdf

¿Y los problemas de optimización?

Ejemplo: Agente viajero

- Dada una lista de ciudades (nodos) encontrar el ciclo con costo mínimo que recorra todas las ciudades.
- Se define el problema de decisión si existe un ciclo que recorra todas las ciudades a un costo menor o igual a C.
- Disminuir C hasta que la respuesta sea ‘no’, entonces el C óptimo es el C del paso anterior.
- Dado C se puede conocer la ruta de la siguiente forma:
  - Para cada distancia entre dos ciudades modificar el costo de la arista que las une a C+1, si la respuesta al problema de decisión con un costo fijo C es no, entonces esa arista se encuentra en la ruta, si no esa arista no es importante para la ruta.
  - Esto se puede hacer en $O(n^2)$

Relación entre problemas de decisión y de funciones

- Sea L ∈ NP entonces existe una relación $R_L(x,y)$ decidible en tiempo polinomial tal que *x* está en L si y sólo si existe una cadena *y* tal que $R_L(x,y)$ es verdadero.
- En particular el problema de función FL asociado al lenguaje L tiene como resultado de una entrada:
  - No si *x* no está en L
  - La cadena *y* si x está en L.
- Los problemas de funciones asociados a lenguajes en NP corresponden a la clase FNP

Reducciones

Un problema de funciones se puede reducir el problema A al problema B...

- Si existen funciones R y S que transforman cadenas en espacio logarítmico y cumplen con lo siguiente:
  - A partir de *x*, instancia de A, se obtiene R(x), instancia de B. Dada R(x) se obtiene una solución z al problema B y S(z) corresponde al resultado del problema A dado x.

A es completo para una clase de problemas de funciones FC...

- Si A está en FC y todos los problemas en FC se reducen a A
- FSAT es FNP completo

FP y FNP son cerrados bajo reducciones

Funciones totales

Hay problemas de funciones que si los transformamos a problemas de decisión el resultado es siempre de aceptación

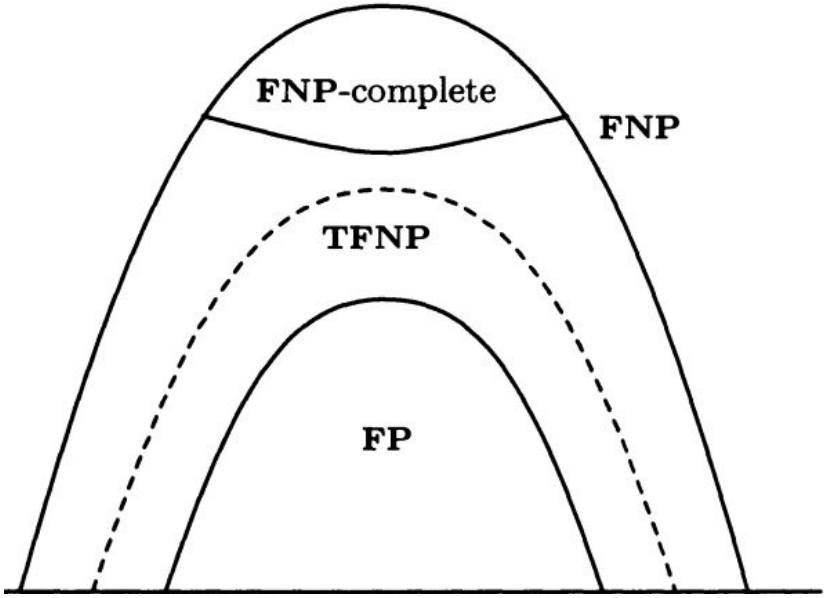
Ejemplo:

- Problema de factorización en primos
  - Dado un testigo de la factorización en primos de un número N, es posible verificarlo en tiempo polinomial, por lo que está en NP.
  - Hasta ahora **no** existen algoritmos polinomiales que lo resuelvan.
  - **Todo** entero tiene una factorización en primos, por lo que el problema de decisión de si existe la factorización en primos de un entero N es siempre afirmativo.
  - A problemas de este tipo se les llama funciones **totales.**

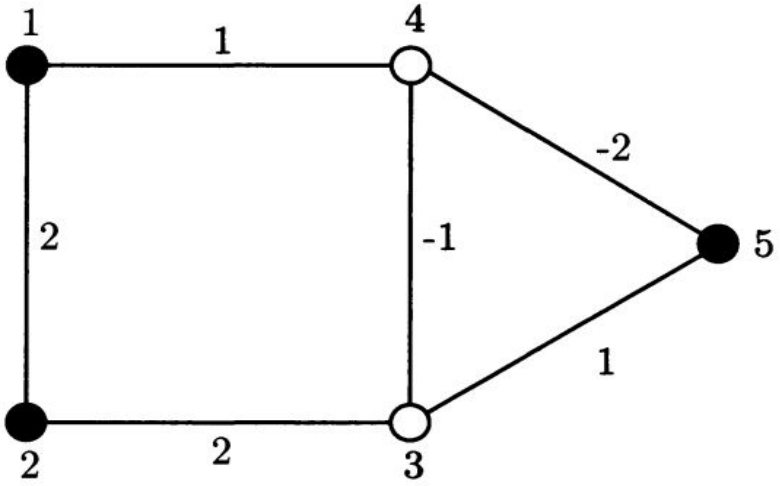
Funciones totales

Un problema R en FNP es total si para cada entrada x existe y tal que R(x,y) es verdadero.

A estos problemas se les denota como TFNP



Funciones totales: HappyNet



$$S(i) \cdot \sum_{[i,j] \in E} S(j) w[i,j] \geq 0.$$

H1=1(1*2+(-1)*1)=1

H2=1(1*2+(-1*2))=0

H3=-1(1*2+(-1)*(-1))=-3

H4=-1(1*1+(-1)*(-1)+1*(-2))=0

H5=1((-1)*(-2)+(-1)*1)=1

Funciones totales: Otro ciclo Hamiltoniano

- Si existe un ciclo Hamiltoniano conocido (un ciclo que pasa por todos los vértices), ¿es posible encontrar otro?
- Para gráficas cuyos nodos son de grado 3 (tienen 3 vecinos) dado un ciclo Hamiltoniano siempre se puede encontrar otro.