Clases de complejidad computacional: P y NP

Optimización :: DM, FCEyN, UBA

1er cuatrimestre 2006

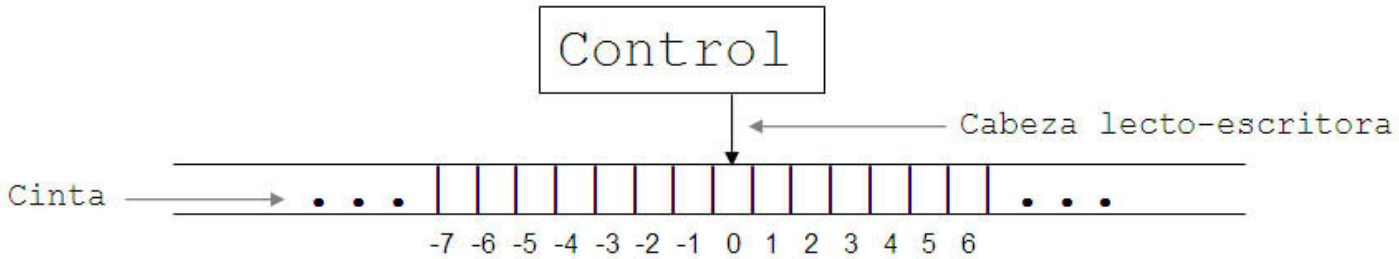
La teoría de NP-completitud

- Se aplica a problemas de decisión, o sea problemas que tienen como respuesta SI o NO (aunque es sencillo ver que sus implicancias pueden extenderse a problemas de optimización).
- En el caso del problema de TSP, la variante de decisión se podría formular como: "¿existe un circuito Hamiltoniano de longitud menor o igual a k?"
- Un problema de decisión π consiste entonces de un conjunto D_π de instancias y un subconjunto $Y_\pi \subseteq D_\pi$ de instancias cuya respuesta es SI.

Modelos de Computadoras: DTM

Recordemos la noción de Máquina de Turing Determinística (DTM)

- Consiste de un control, una cabeza lecto-escritora y una cinta con el siguiente esquema.



- Σ finito, el alfabeto; $\Gamma = \Sigma \cup \{*\}$;
- Q finito, el conjunto de estados;
- $q_0 \in Q$, estado inicial; $Q_f \subseteq Q$, estados finales (q_{si} y q_{no} para problemas de decisión)

Modelos de Computadoras: DTM

- Sobre la cinta tengo escrito el input que es un string de símbolos de Σ a partir de la celda 1, y el resto de las celdas tiene $*$ (blancos).
- Definimos un programa S como un conjunto de quíntuplas $S \subseteq Q \times \Gamma \times Q \times \Gamma \times M$, donde $M = \{+1, -1\}$ son los movimientos de la cabeza a derecha o izquierda.
- Para todo par (q_i, s_j), existe exactamente una quíntupla que comienza con ese par (máquina determinística).

Modelos de Computadoras: DTM

- ¿Qué significa la quíntupla $(q_i, s_h, q_j, s_k, +1)$? Significa que si estando en el estado q_i la cabeza lee s_h, entonces escribe s_k, se mueve a la derecha y pasa al estado q_j.
- La complejidad de una DTM está dada por la cantidad de movimientos de la cabeza en función del tamaño de la entrada.
- **Un problema está en P si existe una DTM de complejidad polinomial que lo resuelve.**
- Existen otros modelos de computadoras determinísticas (máquina de Turing con varias cintas, Random Access Machines, etc.) pero puede probarse que son equivalentes en términos de la polinomialidad de los problemas a la DTM.

Modelos de Computadoras: NDTM

Máquinas de Turing No Determinísticas (NDTM)

- No se pide unicidad de la quíntupla que comienza con cualquier par (q_i, s_j).
- En caso de que hubiera más de una quíntupla, la máquina se replica continuando cada una por una rama distinta.
- Decimos que una NDTM resuelve el problema π si para toda instancia de Y_π existe una rama que llega a un estado final q_{si} y para toda instancia en $D_\pi \setminus Y_\pi$ ninguna rama llega a un estado final q_{si}.

Modelos de Computadoras: NDTM

- Una NDTM es **polinomial** para π cuando existe una función polinomial $T(n)$ de manera que para toda instancia de Y_π de tamaño n, alguna de las ramas termina en estado q_{si} en a lo sumo $T(n)$ pasos.
- **Un problema $\pi \in$ NP si existe una NDTM polinomial que resuelve π.**
- Equivalentemente, un problema $\pi \in$ NP si para toda instancia en Y_π, existe un "certificado" que puede ser verificado en tiempo polinomial. En el caso de TSP, el certificado consiste en un circuito Hamiltoniano de longitud $\leq k$. Es fácil ver que puede verificarse en tiempo polinomial si es un circuito Hamiltoniano y si su longitud es $\leq k$.
- Claramente, P $\subseteq$ NP. **Conjetura: P $\neq$ NP.**

Modelos de Computadoras: NDTM

- Dado un problema π, se define su problema complemento $\overline{\pi}$ como aquel tal que $D_{\overline{\pi}} = D_\pi$ e $Y_{\overline{\pi}} = D_\pi \setminus Y_\pi$.

  Ejemplo: $\pi =$ ¿Es p primo?; $\overline{\pi} =$ ¿Es p compuesto?.
- **Un problema $\pi \in$ co-NP si $\overline{\pi} \in$ NP.**
- Claramente, P $\subseteq$ co-NP.

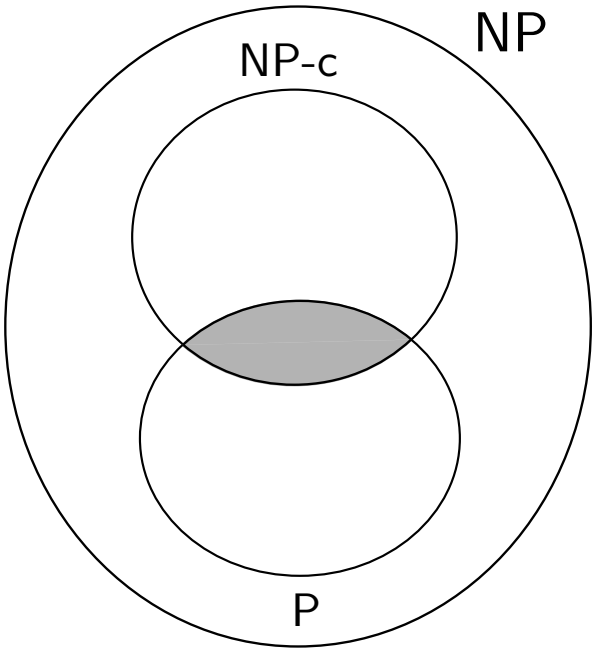
NP-completitud

- **Reducción polinomial:** Sean π y π' dos problemas de decisión. Decimos que $f : D_{\pi'} \to D_{\pi}$ es una reducción polinomial de π' en π si f se computa en tiempo polinomial y para todo $d \in D_{\pi'}$, $d \in Y_{\pi'} \Leftrightarrow f(d) \in Y_{\pi}$. Notación: $\pi' \preccurlyeq \pi$.
- Notemos que si $\pi'' \preccurlyeq \pi'$ y $\pi' \preccurlyeq \pi$ entonces $\pi'' \preccurlyeq \pi$, ya que la composición de dos reducciones polinomiales es una reducción polinomial.
- Un problema π es **NP-completo** si:
  1. $\pi \in$ NP.
  2. Para todo $\pi' \in$ NP, $\pi' \preccurlyeq \pi$.
- Si un problema π verifica la condición 2., π es NP-Hard (es al menos tan "difícil" como todos los problemas de NP).

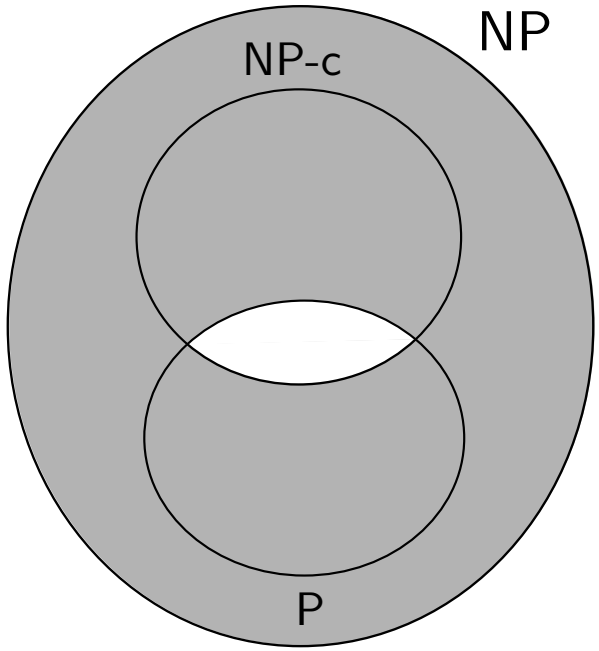
¿P ≠ NP? La pregunta del millón...

- **Si existe un problema en NP-c ∩ P, entonces P=NP.**
  - Si $\pi \in$ NP-c $\cap$ P, existe un algoritmo polinomial que resuelve π, por estar π en P. Por otro lado, como π es NP-completo, para todo $\pi' \in$ NP, $\pi' \preccurlyeq \pi$.
  - Sea $\pi' \in$ NP. Apliquemos la reducción polinomial que transforma instancias de π' en instancias de π y luego el algoritmo polinomial que resuelve π. Por definición de reducción polinomial, es fácil ver que lo que se obtiene es un algoritmo polinomial que resuelve π'.
- Hasta el momento no se conoce ningún problema en NP-c $\cap$ P, así como tampoco se ha demostrado que un problema esté en NP $\setminus$ P. En ese caso, obviamente, se probaría que P $\neq$ NP.

Esquema de clases



si P=NP...



si P≠NP...

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

El problema SAT consiste en decidir si, dada una fórmula lógica φ expresada como conjunción de disyunciones (ej: $\varphi = x_1 \wedge (x_2 \vee \neg x_1) \wedge (x_3 \vee \neg x_4 \vee x_1)$), existe una valuación de sus variables que haga verdadera φ.

Es fácil ver que SAT $\in$ NP. El certificado en este caso sería una valuación que satisfaga φ. Evaluar una fórmula es polinomial.

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

La demostración de Cook es directa: considera un problema genérico $\pi \in$ NP y una instancia genérica $d \in D_\pi$. A partir de la hipotética NDTM que resuelve π, genera en tiempo polinomial una fórmula lógica $\varphi_{\pi,d}$ en forma normal (conjunción de disyunciones) tal que $d \in Y_\pi$ si y sólo si $\varphi_{\pi,d}$ es satisfactible.

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

A partir del Teorema de Cook, la técnica standard para probar que un problema π es NP-completo aprovecha la transitividad de $\preccurlyeq$, y consiste en lo siguiente:

1. Mostrar que π está en NP.
2. Elegir un problema π' apropiado que se sepa que es NP-completo.
3. Construir una reducción polinomial f de π' en π.

La segunda condición en la definición de problema NP-completo sale usando la transitividad: sea π'' un problema cualquiera de NP. Como π' es NP-completo, $\pi'' \preccurlyeq \pi'$. Como probamos que $\pi' \preccurlyeq \pi$, resulta $\pi'' \preccurlyeq \pi$.

Reducción de SAT a 3-SAT

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 $\varphi = C_1 \wedge \cdots \wedge C_m$. Vamos a reemplazar cada C_i por una conjunción de disyunciones φ'_i, donde cada disyunción tenga tres literales, y de manera que φ sea satisfactible si y sólo si $\varphi_1 \wedge \cdots \wedge \varphi_m$ lo es.

Reducción de SAT a 3-SAT

- Si C_i tiene tres literales, queda como está.
- C_i tiene menos de tres literales, agregamos nuevas variables como en el ejemplo:

$$(x_1 \vee \neg x_2) \longrightarrow (x_1 \vee \neg x_2 \vee y) \wedge (x_1 \vee \neg x_2 \vee \neg y)$$

- Si C_i tiene cuatro o más literales, agregamos nuevas variables como en el ejemplo:

$$(x_1 \vee \neg x_2 \vee x_3 \vee x_4 \vee \neg x_5) \longrightarrow$$

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

Queda como ejercicio escribir formalmente la reducción y demostrar que es una reducción polinomial de SAT a 3-SAT.