Petits Apunts sobre P i NP

(versió encara preliminar)

Albert Atserias

Novembre 2004 (revisat Febrer 2006)

1 Definicions i Teorema de Cook

Problemes computacionals. Un *problema computacional* s'especifica indicant

- un conjunt de *possibles entrades* E,
- un conjunt de *possibles sortides* S,
- una *relació d'entrada/sortida* $R \subseteq E \times S$, i
- una *funció de talla* $|\cdot| : E \to \mathbf{N}$.

La relació d'entrada/sortida R indica quins parells (x, y) formats per una entrada $x \in E$ i una sortida $y \in S$ són solucions vàlides del problema. Informalment, el problema es formularà així: "Donada una entrada $x \in E$, trobar una sortida $y \in S$ tal que $(x, y) \in R$, si n'hi ha". La *talla* d'una entrada $x \in E$ es denota $|x|$.

Problemes decisionals. El cas particular de problema computacional en què S és el domini booleà $\{0, 1\}$, i per cada $x \in E$ existeix un únic $y \in S$ tal que $(x, y) \in R$, s'anomena problema decisional. Així doncs, un *problema decisional* s'especifica indicant

- un conjunt de *possibles entrades* E,
- un subconjunt $L \subseteq E$ d'*instàncies positives*,
- una *funció de talla* $|\cdot| : E \to \mathbf{N}$.

Sovint farem servir l'expressió "Sigui $L \subseteq E$ un problema decisional..." i entendrem que E és el conjunt d'entrades i que L és el subconjunt d'instàncies positives. La funció de talla també serà clara pel context. Informalment, el problema es formularà així: "Donada una entrada $x \in E$, determinar si $x \in L$ o no".

Cost i algorismes polinòmics. Sigui $A : E \to S$ un algorisme que agafa entrades de E i retorna sortides de S. Per cada entrada $x \in E$, sigui $T_A(x)$ el nombre de passos que triga l'algorisme A amb entrada x. Per cada $n \in \mathbf{N}$, sigui

$$t_A(n) = \max\{T_A(x) : x \in E,\ |x| = n\}.$$

Es diu que A es un algorisme polinòmic si existeix una constant $c \geq 0$ tal que $t_A(n) = \mathrm{O}(n^c)$.

Les classes P i NP. Sigui $L \subseteq E$ un problema decisional. Es diu que L és *decidible en temps polinòmic* si existeix un algorisme polinòmic $A : E \to \{0, 1\}$ tal que, per a tot $x \in E$, tenim

$$x \in L \Leftrightarrow A(x) = 1.$$

En aquest cas diem que *L pertany a* $\mathbf{P}$, o *L és a la classe* $\mathbf{P}$, o simplement $L \in \mathbf{P}$. Es diu que L és *decidible en temps polinòmic indeterminista* si existeix un polinomi $p(n)$ i un algorisme polinòmic $B : E \times E' \to \{0, 1\}$ tal que, per tot $x \in E$, tenim

$$x \in L \Leftrightarrow \text{ existeix algun } y \in E' \text{ tal que } |y| \leq p(|x|) \text{ i } B(x, y) = 1.$$

En aquest cas diem que L *pertany a* $\mathbf{NP}$, o L *és a la classe* $\mathbf{NP}$, o simplement $L \in \mathbf{NP}$. L'y de la definició de $\mathbf{NP}$ sovint s'anomena *certificat*, o *testimoni*, o *prova*, i l'algorisme B de la definició de $\mathbf{NP}$ sovint s'anomena *verificador*. Per tant, per veure que un problema decisional pertany a $\mathbf{NP}$ cal veure que les instàncies positives tenen certificats de mida polinòmica que es poden verificar en temps polinòmic.

Reduccions, problemes NP-hard, i problemes NP-complets. Siguin $L \subseteq E$ i $L' \subseteq E'$ dos problemes decisionals. Diem que L *es redueix a* L' *en temps polinòmic* si existeix un algorisme polinòmic $A : E \to E'$ tal que, per a tot $x \in E$, tenim

$$x \in L \Leftrightarrow A(x) \in L'.$$

Diem que A és una *reducció polinòmica*. Així doncs, si $A' : E' \to \{0,1\}$ és un algorisme per L' i $A : E \to E'$ és una reducció polinòmica de L a L', aleshores la composició $A' \circ A$ (o $A \mid A'$ en notació *pipe* de UNIX) és un algorisme per L. Es diu que *s'ha reduït un problema a l'altre*.

Es diu que un problema decisional $L \subseteq E$ és $\mathbf{NP}$*-hard* si tot problema que pertany a $\mathbf{NP}$ s'hi redueix en temps polinòmic. Es diu que és $\mathbf{NP}$*-complet* si, a més de ser $\mathbf{NP}$-hard, el problema pertany a $\mathbf{NP}$. Així doncs, un problema $\mathbf{NP}$-hard és un problema al menys tan difícil com qualsevol problema que pertanyi a $\mathbf{NP}$. En efecte, si el poguéssim resoldre amb un algorisme polinòmic, aleshores tots els problemes de $\mathbf{NP}$ també es podrien resoldre amb un algorisme polinòmic. Un problema $\mathbf{NP}$-complet, a més de ser al menys tan difícil com qualsevol problema de $\mathbf{NP}$, ell mateix pertany a $\mathbf{NP}$.

Problemes NP-complets, Teorema de Cook. Existeixen problemes $\mathbf{NP}$-complets? Aquí en tenim un anomenat CIRCUIT-SAT:

> Donat un circuit C amb connectives AND, OR, NOT, variables $x_1, \ldots, x_n$, i una única sortida, determinar si C és satisfactible. És a dir, determinar si existeixen $a_1, \ldots, a_n \in \{0,1\}$ tals que $C[x_1 := a_1, \ldots, x_n := a_n] = 1$.

Recordem que un circuit booleà és un graf dirigit acíclic en què tots els nodes tenen grau d'entrada 0, 1 o 2, i en què hi ha exactament un node amb grau de sortida 0. Els nodes amb grau d'entrada 0 són entrades i estàn etiquetats amb una variable x_i, els de grau d'entrada 1 porten l'etiqueta NOT, i els de grau d'entrada 2 porten l'etiqueta AND o OR segons el tipus de porta lògica que representin. La sortida del circuit és l'únic node amb grau de sortida 0.

És evident que CIRCUIT-SAT pertany a $\mathbf{NP}$. Demostrar que CIRCUIT-SAT és $\mathbf{NP}$-hard, i per tant $\mathbf{NP}$-complet, és bastant més laboriós. El resultat es coneix com el Teorema de Cook. La idea principal de la demostració és que qualsevol algorisme pot acabar essent implementat amb un circuit electrònic amb portes AND, OR i NOT si l'entrada es codifica amb valors booleans. Per tant, si L és un problema de $\mathbf{NP}$ qualsevol amb verificador B i certificats afitats pel polinomi $p(n)$, i si C_n és el circuit que implementa B amb entrades de talla n, aleshores per a tot $x \in E$ de talla n tenim que

$$\begin{aligned} x \in L &\Leftrightarrow \exists y(|y| \leq p(|x|) \wedge B(x,y) = 1) \\ &\Leftrightarrow \exists y(|y| \leq p(|x|) \wedge C_n(x,y) = 1) \\ &\Leftrightarrow C_n(x,\cdot) \text{ satisfactible.} \end{aligned}$$

Per tant, la funció $x \to C_n(x,\cdot)$ és una reducció polinòmica de L a CIRCUIT-SAT.

2 Algunes Reduccions

Un cop s'ha demostrat que CIRCUIT-SAT és $\mathbf{NP}$-complet ja podem demostrar que molts altres problemes són $\mathbf{NP}$-complets. Com a pas previ veurem que una variant del problema CIRCUIT-SAT per circuits molt senzillets ja és $\mathbf{NP}$-complet.

3-SAT:

> Donat un conjunt de clàusules $C_1, \ldots, C_m$ de com a molt tres literals cadascuna, determinar si la conjunció $C_1 \wedge \cdots \wedge C_m$ és satisfactible.

És clar que 3-SAT pertany a $\mathbf{NP}$. Per veure que 3-SAT és $\mathbf{NP}$-hard, i per tant $\mathbf{NP}$-complet, n'hi ha prou a demostrar que CIRCUIT-SAT es redueix a 3-SAT. Aquí va la reducció. Sigui C un circuit booleà amb connectives AND, OR i NOT i entrades $x_1, \ldots, x_n$. Construïrem un conjunt de clàusules $F = \{C_1, \ldots, C_m\}$ de manera que C sigui satisfactible si, i només si, la conjunció $C_1 \wedge \cdots \wedge C_m$ ho és. Per cada porta u de C, incloses les entrades, definim una nova variable y_u i afegim les següents clàusules de com a molt tres literals a F:

1. Si u és una entrada de C etiquetada x_i, afegim $\neg y_u \vee x_i$ i $y_u \vee \neg x_i$ a F.
2. Si u és una NOT i el cable ve de la porta v, afegim $\neg y_v \vee \neg y_u$ i $y_v \vee y_u$ a F.
3. Si u és una AND i els cables venen de v_1 i v_2, afegim $\neg y_{v_1} \vee \neg y_{v_2} \vee y_u$ a F.
4. Si u és una AND i els cables venen de v_1 i v_2, també afegim $\neg y_u \vee y_{v_1}$ i $\neg y_u \vee y_{v_2}$ a F.
5. Si u és una OR i els cables venen de les portes v_1 i v_2, afegim $\neg y_{v_1} \vee y_u$ i $\neg y_{v_2} \vee y_u$ a F.
6. Si u és una OR i els cables venen de les portes v_1 i v_2, també afegim $\neg y_u \vee y_{v_1} \vee y_{v_2}$ a F.
7. Si u és la sortida de C, també afegim y_u a F.

És clar que el conjunt de clàusules F es pot construïr a partir de C en temps polinòmic. A més, no és gens difícil veure que C és satisfactible si, i només si, la conjunció de les clàusules de F és satisfactible. Per tant, l'algorisme que, donat un circuit C, retorna el corresponent conjunt de clàusules F és una reducció polinòmica de CIRCUIT-SAT a 3-SAT. Per tant, 3-SAT és $\mathbf{NP}$-complet.

CONJUNT INDEPENDENT:

> Donat un graf $G = (V, E)$ i un nombre natural k, determinar si G té un conjunt independent de k nodes. És a dir, determinar si existeix un subconjunt $A \subseteq V$ de k nodes tal que $\{u, v\} \notin E$ per a tot $u, v \in A$.

Aquí va la reducció des de 3-SAT. Sigui $F = \{C_1, \ldots, C_m\}$ un conjunt de clàusules. Definim k i construïm un graf $G = (V, E)$ així:

1. $V = \{(l, C_i) : l$ apareix a $C_i\}$,
2. $E = \{\{(l_1, C_i), (l_2, C_j)\} : i = j$ o $l_1 \equiv \neg l_2\}$,
3. $k = m$.

No és gens difícil veure que $C_1 \wedge \cdots \wedge C_m$ és satisfactible si, i només si, G conté un conjunt independent de mida k. Per tant, l'algorisme que, donat un conjunt de clàusules $C_1, \ldots, C_m$, retorna el corresponent graf G i el número k és una reducció polinòmica de 3-SAT a CONJUNT INDEPENDENT. Per tant, CONJUNT INDEPENDENT és $\mathbf{NP}$-hard, i donat que pertany a $\mathbf{NP}$, és $\mathbf{NP}$-complet.

RECOBRIMENT:

> Donat un graf $G = (V, E)$ i un nombre natural k, determinar si G té un recobriment de les arestes amb k nodes. És a dir, determinar si existeix un subconjunt $A \subseteq V$ de k nodes tal que per cada $\{u, v\} \in E$ tenim que al menys un dels extrems u o v pertany A.

Fem una reducció des de CONJUNT INDEPENDENT. Sigui $G = (V, E)$ un graf i k un nombre natural. Definim k' i un graf $G' = (V', E')$ així:

1. $V' = V$,
2. $E' = E$,
3. $k' = |V| - k$.

És clar que G té un conjunt independent de k nodes si, i només si, G' té un recobriment de les arestes amb k' nodes. Per tant, l'algorisme que, donat un graf G i un nombre k, retorna el graf G' i el nombre k' és una reducció polinòmica de CONJUNT INDEPENDENT a RECOBRIMENT.

INEQUACIONS LINEALS 0-1:

> Donada una matriu A de $m \times n$ enters i donat un vector b de m enters, determinar si existeix un vector $x \in \{0, 1\}^n$ tal que $Ax \geq b$.

Considerem la següent reducció des de 3-SAT. Donat un conjunt de clàusules $C_1, \ldots, C_m$ sobre les variables $x_1, \ldots, x_n$, definim una matriu A de dimensions $m \times n$ i un vector b de dimensió m així:

1. $A[i, j] = 1$ si x_j apareix a C_i sense negació,
2. $A[i, j] = -1$ si x_j apareix a C_i amb negació (i no sense),
3. $A[i, j] = 0$ si x_j no apareix a C_i,
4. $b[i] = 1 - n_i$ on n_i és el nombre de literals amb negació de C_i.

No és gens difícil veure que $C_1 \wedge \cdots \wedge C_m$ és satisfactible si, i només si, existeix un $x \in \{0, 1\}^n$ tal que $Ax \geq b$. A més, la matriu A i el vector b es poden construïr en temps polinòmic a partir de $C_1, \ldots, C_m$. Per tant, la reducció és polinòmica, i el problema és doncs $\mathbf{NP}$-hard i $\mathbf{NP}$-complet perquè pertany a $\mathbf{NP}$.

NÚMERO CROMÀTIC:

> Donat un graf $G = (V, E)$ i nombre natural k, determinar si es pot pintar G amb k colors. És a dir, determinar si existeix una functió $f : V \to \{1, \ldots, k\}$ tal que si $\{u, v\} \in E$, aleshores $f(u) \neq f(v)$.

És clar que pertany a $\mathbf{NP}$. Reduïm a partir de 3-SAT. Donades clàusules $C_1, \ldots, C_m$ sobre les variables $x_1, \ldots, x_n$, definim k i un graf $G = (V, E)$ així:

1. $k = 3$,
2. $V = \{T, F, R\} \cup \{x_j, \neg x_j : j = 1, \ldots, n\} \cup \{(l, C_i, 1), (l, C_i, 2) : i = 1, \ldots, m \text{ i } l \in C_i\}$,
3. E conté: un triangle entre T, F i R; un triangle entre x_j, $\neg x_j$ i R; un triangle entre $(l_1, C_i, 1)$, $(l_2, C_i, 1)$ i $(l_3, C_i, 1)$ per cada $C_i = \{l_1, l_2, l_3\}$; un triangle entre $(l_1, C_i, 1)$, $(l_2, C_i, 1)$ i T per cada $C_i = \{l_1, l_2\}$; un triangle entre $(l, C_i, 1)$, T i F per cada $C_i = \{l\}$; una aresta entre $(l, C_i, 1)$ i $(l, C_i, 2)$; una aresta entre $(l, C_i, 2)$ i T; i una aresta entre $(l, C_i, 2)$ i l.

És un xic més complicat que en els altres casos, però no massa tampoc, veure que això és una reducció. És a dir, que $C_1 \wedge \cdots \wedge C_m$ és satisfactible si, i només si, G es pot pintar amb k colors. Clarament, la reducció és polinòmica. Per tant, NÚMERO CROMÀTIC és $\mathbf{NP}$-complet.