Petits Apunts sobre P i NP

Albert Atserias

Novembre 2004 (revisat Febrer 2006, tornat a revisar Novembre 2010)

1 Dominis de dades, problemes decisionals i computacionals

Dominis de dades Un *domini de dades* és un conjunt que porta associat una *funció de talla* que assigna un nombre natural a cada element del conjunt. Alguns exemples de dominis de dades són el conjunt dels nombres enters, el conjunt dels nombres racionals, el conjunt dels grafs, el conjunt dels grafs dirigits acíclics amb pesos enters associats a les arestes, el conjunt dels dos valors booleans $\{0, 1\}$, etcètera. La funció de talla associada a cadascun d'aquests dominis es sobreentén pel context i típicament correspon al nombre d'unitats de memòria que es requereix per representar una codificació estàndar de l'objecte. Si x és un objecte d'un domini de dades, la talla o *mida* de x es denota $|x|$.

Problemes decisionals Són els problemes on es demana determinar si un objecte donat satisfà una propietat. Per exemple: determinar si un graf donat conté un cicle hamiltonià, és a dir, un cicle que passa per cada vèrtex exactament una vegada. En abstracte, un *problema decisional* s'especifica indicant:

- un conjunt de *possibles entrades* E,
- un subconjunt $T \subseteq E$ d'*instàncies positives* (la *propietat*).

La resta d'instàncies, les de la diferència $E - T$, s'anomenen *instàncies negatives*. Informalment, el problema es plantejarà així: "Donada una entrada $x \in E$, determinar si x pertany a T o a $E - T$".

De vegades el problema que realment ens interessa no és determinar si un objecte satisfà o no una propietat, sinó calcular-ne un resultat més complex que una simple resposta SÍ o NO. Per exemple, seguint amb el problema de determinar si un graf donat conté un cicle hamiltonià, el més probable és que ens interessi trobar aquest cicle en cas que existeixi. Definim aquest tipus de problemes.

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* $R \subseteq E \times S$.

Informalment, el problema es planteja així: "Donada una entrada $x \in E$, trobar una sortida $y \in S$ tal que $(x, y) \in R$, si n'hi ha".

Recordem el problema decisional on, donat un graf, es demana determinar si conté un cicle hamiltonià o no, i la versió computacional on, donat un graf, es demana trobar un cicle hamiltonià, si n'hi ha. És evident que resoldre la segona versió ens permet resoldre la primera. Però, dit això, no és gens difícil entendre que aquestes dues versions del problema són, de fet, essencialment equivalents. En efecte, si som capaços de determinar si un graf qualsevol conté un cicle hamiltonià, també podrem trobar quines són les arestes d'un d'aquests cicles: escollim un vèrtex v qualsevol, esborrem totes les arestes que toquen v menys dues, determinem si el subgraf resultant conté un cicle hamiltonià, i repetim fins que en trobem un. Si iterem aquest procés fins a esgotar tots els vèrtexs acabarem per construir el cicle buscat.

Vist això, que de fet és molt general i s'aplica a la majoria dels problemes que ens trobem a la pràctica, ens satisfarem de desenvolupar la teoria estrictament per problemes decisionals.

2 Classes P i NP

Classes de problemes En aquesta secció introduïrem classes de problemes decisionals; és a dir, conjunts de problemes que típicament compartiran alguna característica comuna. En aquest punt convé fer notar els tres nivells d'abstracció que hi intervenen:

instància	problema decisional	classe de problemes decisionals
x	Π	$\mathbf{C}$

Per al que ve a continuació és important sentir-se còmode amb aquests tres nivells d'abstracció creixent (com indica la llargada del seu nom), i no confondre'ls.

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 elementals (en el model de càlcul sota consideració) que triga l'algorisme A amb entrada x. Per cada $n \in \mathbb{N}$, sigui

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

Si existeix un polinomi $p(n)$ tal que $t_A(n) \leq p(n)$ per tot $n \in \mathbb{N}$, es diu que A és un algorisme polinòmic.

La classe P Sigui Π el problema decisional següent: "Donat $x \in E$, determinar si x pertany a T o no". Es diu que Π és *decidible en temps polinòmic* si existeix un algorisme polinòmic $A : E \to \{0, 1\}$ tal que, per tot $x \in E$, tenim

$$x \in T \Rightarrow A(x) = 1$$
$$x \notin T \Rightarrow A(x) = 0.$$

$\mathbf{P}$ és la classe dels problemes decisionals que són decidibles en temps polinòmic. Si Π pertany a $\mathbf{P}$ també diem que Π *és* $\mathbf{P}$.

La classe NP Sigui Π el problema decisional següent: "Donat $x \in E$, determinar si x pertany a T o no". Es diu que Π és *decidible en temps polinòmic indeterminista* si existeix un domini de dades E', un algorisme polinòmic $B : E \times E' \to \{0, 1\}$, i un polinomi $p(n)$, tals que, per tot $x \in E$, tenim

$$x \in T \Rightarrow B(x, y) = 1 \text{ per algun } y \in E' \text{ tal que } |y| = p(|x|)$$
$$x \notin T \Rightarrow B(x, y) = 0 \text{ per tot } y \in E' \text{ tal que } |y| = p(|x|).$$

$\mathbf{NP}$ és la classe dels problemes decisionals que són decidibles en temps polinòmic indeterminista. Si Π pertany a $\mathbf{NP}$ també diem que Π *és* $\mathbf{NP}$. El nom $\mathbf{NP}$ prové de l'anglès "non-deterministic polynomial time". L'y garantit pel cas $x \in T$ en la definició s'anomena *certificat*, o *testimoni*, o *prova*. L'algorisme B s'anomena *verificador*. Per tant, per veure que un problema decisional Π pertany a $\mathbf{NP}$ cal veure que les instàncies positives de Π són les que tenen certificats de talla polinòmica que es poden verificar per algun algorisme de temps polinòmic.

3 Reduccions, NP-completesa, i Teorema de Cook

Reduccions Siguin Π i Π' dos problemes decisionals amb conjunts d'entrades E i E', i amb conjunts d'instàncies positives T i T', respectivament. Diem que Π *es redueix a* Π' *en temps polinòmic* si existeix un algorisme polinòmic $A : E \to E'$ tal que, per tot $x \in E$, tenim

$$x \in T \Rightarrow A(x) \in T'$$
$$x \notin T \Rightarrow A(x) \notin T'.$$

Diem que A és una *reducció polinòmica* de Π a Π'. Noti's que si $A' : E' \to \{0,1\}$ és un algorisme que decideix Π' i $A : E \to E'$ és una reducció polinòmica de Π a Π', aleshores la composició $A' \circ A$ (o $A \mid A'$ en notació *pipe* de UNIX) és un algorisme que decideix Π. Es diu que *s'ha reduït* Π *a* Π' i es denota $\Pi \leq^p \Pi'$. La relació de reducció polinòmica entre problemes decisionals estableix un (pre-)ordre (parcial) entre aquests. En cas que $\Pi \leq^p \Pi'$ i $\Pi' \leq^p \Pi$ diem que Π i Π' són *polinòmicament equivalents*, i ho denotem per $\Pi \equiv^p \Pi'$.

Problemes NP-difícils, i NP-complets Sigui $\dot{\Pi}$ un problema decisional. Es diu que $\dot{\Pi}$ és $\mathbf{NP}$*-difícil* si $\Pi \leq^p \dot{\Pi}$ per tot Π de $\mathbf{NP}$; és a dir, si $\dot{\Pi}$ és una fita superior, en l'ordre de les reduccions polinòmiques, a tots els problemes de $\mathbf{NP}$. Es diu que $\dot{\Pi}$ és $\mathbf{NP}$*-complet* si és $\mathbf{NP}$-difícil i a més pertany a $\mathbf{NP}$; és a dir, si $\dot{\Pi}$ és un màxim, en l'ordre de les reduccions polinòmiques, del conjunt dels problemes de $\mathbf{NP}$.

Així doncs, un problema $\mathbf{NP}$-difícil és un problema almenys 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 és almenys tan difícil com qualsevol problema de $\mathbf{NP}$, però no estrictament més difícil que tots ells perquè ell mateix pertany a $\mathbf{NP}$.

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

> Donat un circuit C amb portes lògiques 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 vèrtexs tenen grau d'entrada 0, 1 o 2, i en què hi ha exactament un vèrtex amb grau de sortida 0. Els vèrtexs amb grau d'entrada 0 representen les entrades del circuit i estan etiquetats amb una variable x_i, els de grau d'entrada 1 representen una porta lògica NOT, i els de grau d'entrada 2 representen una porta lògica AND o OR i estan etiquetats amb A o O per indicar-ho. La sortida del circuit és l'únic vèrtex amb grau de sortida 0.

És evident que CIRCUIT-SAT pertany a $\mathbf{NP}$: el certificat d'una instància positiva C és precisament la seqüència de bits $a_1, \ldots, a_n$ tal que $C[x_1 := a_1, \ldots, x_n := a_n] = 1$, condició que es pot verificar en temps polinòmic sense cap dificultat. Demostrar que és $\mathbf{NP}$-difícil, 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 de manera eficient amb un circuit electrònic amb portes AND, OR i NOT si l'entrada es codifica en binari (aquesta és la part que no demostrarem aquí). Sigui doncs Π un problema de $\mathbf{NP}$ qualsevol amb verificador $B : E \times E' \to \{0,1\}$ i certificats de talla $p(n)$ per entrades de talla n, i sigui $C_n(\overline{x}, \overline{y})$ el circuit de mida polinòmica que implementa $B(x,y)$ amb entrada $x \in E$ de talla n i certificat $y \in E'$ de talla $p(n)$, prèvia verificació que $\overline{x}$ i $\overline{y}$ en efecte codifiquen un x de E i un y de E'.

Aleshores per tot $x \in E$ de talla n tenim que

$$\begin{aligned} x \in T &\Rightarrow B(x,y) = 1 \text{ per algun } y \in E' \text{ tal que } |y| = p(|x|) \\ &\Rightarrow C_n(\overline{x},\overline{y}) = 1 \text{ per algun } y \in E' \text{ tal que } |y| = p(|x|) \\ &\Rightarrow C_n(\overline{x},\overline{y}) = 1 \text{ per algun } \overline{y} \\ &\Rightarrow C_n(\overline{x},\cdot) \text{ és satisfactible} \end{aligned}$$

i

$$\begin{aligned} x \notin T &\Rightarrow B(x,y) = 0 \text{ per tot } y \in E' \text{ tal que } |y| = p(|x|) \\ &\Rightarrow C_n(\overline{x},\overline{y}) = 0 \text{ per tot } y \in E' \text{ tal que } |y| = p(|x|) \\ &\Rightarrow C_n(\overline{x},\overline{y}) = 0 \text{ per tot } \overline{y} \\ &\Rightarrow C_n(\overline{x},\cdot) \text{ és insatisfactible}, \end{aligned}$$

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

4 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 fórmules en forma normal conjuntiva (restringida) 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 $\mathbf{NP}$-difícil, 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 portes lògiques 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 a F les següents clàusules, de com a molt tres literals cadascuna, que forcen les variables y_u a prendre el valor que prendrien les portes que representen si s'avalués el circuit:

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 vénen 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 vénen 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 vénen 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 vénen de les portes v_1 i v_2, també afegim $\neg y_u \vee y_{v_1} \vee y_{v_2}$ a F.

Fixeu-vos que no hem fet gran cosa: només hem forçat les variables a prendre el valor que s'espera d'elles. Finalment cal afegir una clàusula que forci l'avaluació de la sortida de C a 1:

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 vèrtexs. És a dir, determinar si existeix un subconjunt $A \subseteq V$ de k vèrtexs tal que $\{u, v\} \notin E$ per tot $u, v \in A$.

Aquí va la reducció de 3-SAT a CONJUNT INDEPENDENT. Donades clàsules $C_1, \ldots, C_m$, definim:

1. $V = \{(\ell, C_i) : \ell$ és un literal que apareix a $C_i\}$,
2. $E = \{\{(\ell_1, C_i), (\ell_2, C_j)\} : i = j$ o $\ell_1 \equiv \neg\ell_2\}$,
3. $k = m$.

No és gens difícil veure que $C_1 \wedge \cdots \wedge C_m$ és satisfactible si, i només si, el graf $G = (V, E)$ conté un conjunt independent de k vèrtexs. 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 $\mathbf{NP}$-difícil i també $\mathbf{NP}$-complet perquè pertany a $\mathbf{NP}$.

RECOBRIMENT:

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

Proposem una reducció de CONJUNT INDEPENDENT a RECOBRIMENT. 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 vèrtexs si, i només si, G' té un recobriment de les arestes amb k' vèrtexs. 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. Això demostra que RECOBRIMENT és $\mathbf{NP}$-difícil i també $\mathbf{NP}$-complet perquè pertany a $\mathbf{NP}$.

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 de 3-SAT a INEQUACIONS LINEALS 0-1. 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 $\mathbf{NP}$-difícil i $\mathbf{NP}$-complet perquè pertany a $\mathbf{NP}$.

SUMA DE SUBCONJUNT:

> Donats n enters positius $p_1, \ldots, p_n$ escrits amb d dígits decimals cadascun, i donat un objectiu c també escrit amb d dígits decimals, determinar si existeix un subconjunt $S \subseteq \{1, \ldots, n\}$ dels enters que sumi c; és a dir, tal que $\sum_{i \in S} p_i = c$.

Reduïm 3-SAT a SUMA DE SUBCONJUNT. Donat un conjunt de clàusules $C_1, \ldots, C_m$ sobre les variables $x_1, \ldots, x_n$, introduïm dos enters positius a_i i b_i per cada variable x_i, i dos enters positius c_j i d_j per cada clàusula C_j. Tots els enters tenen $d = n + m$ dígits decimals i es defineixen així. Per cada $j \in \{1, \ldots, m\}$:

1. si C_j té un sol literal, definim $c_j = d_j = 0$,
2. si C_j té exactament dos literals, definim $c_j = 10^{j-1}$ i $d_j = 0$,
3. si C_j té exactament tres literals, definim $c_j = d_j = 10^{j-1}$.

Per cada $i \in \{1, \ldots, m\}$:

1. si x_i apareix exactament a $C_{j_1}, \ldots, C_{j_{r_i}}$ sense negació, definim $a_i = 10^{m+i-1} + \sum_{\ell=1}^{r_i} 10^{j_\ell - 1}$,
2. si x_i apareix exactament a $C_{k_1}, \ldots, C_{k_{s_i}}$ amb negació, definim $b_i = 10^{m+i-1} + \sum_{\ell=1}^{s_i} 10^{k_\ell - 1}$.

Finalment definim $c = \sum_{i=1}^{n} 10^{m+i-1} + \sum_{j=1}^{m} |C_j| \cdot 10^{j-1}$. Cal demostrar que la conjunció $C_1 \wedge \cdots \wedge C_m$ és satisfactible si, i només si, existeix un subconjunt dels enters $a_1, b_1, \ldots, a_n, b_n, c_1, d_1, \ldots, c_m, d_m$ que sumen c. Un cop fet això haurem demostrat que el problema és $\mathbf{NP}$-difícil i per tant $\mathbf{NP}$-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 3-SAT a NÚMERO CROMÀTIC. 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 $\mathbf{NP}$-difícil i també $\mathbf{NP}$-complet perquè pertany a $\mathbf{NP}$.