Capítol 6 : Complexitat computacional

Matemàtica discreta
Grau d'Enginyeria Informàtica
Escola d'Enginyeria
Universitat Autònoma de Barcelona

Contingut

Un **problema de decisió** és aquell que sempre té resposta $\mathrm{Sí}$ o bé No, per a totes les dades possibles d'entrada.

Exemple 1.

- *Sigui $n \in \mathbb{N}$, és n un primer?*
- *Sigui $G(V, A)$ i $x, y \in V$. És y accessible des de x?*
- *És G un graf hamiltonià?*

Un **problema de càlcul** demana calcular unes dades de sortida a partir de les dades d'entrada. Així, un problema de decisió podria veure's com un cas particular de problema de càlcul, on les dades de sortida sempre són del conjunt $\{S, N\}$.

Exemple 2.

- *Sigui $n \in \mathbb{N}$, calcular els divisors de n.*
- *Sigui $G(V, A)$ i $x, y \in V$. Trobar un camí de x a y.*
- *Trobar un circuit hamiltonià en un graf G.*

Un **problema d'optimització** demana trobar una solució òptima (en algun sentit) d'entre el conjunt de solucions d'un problema de càlcul.

Exemple 3.

- *Trobar el camí de cost mínim entre dos vèrtexs x i y.*
- *Trobar un arbre generador òptim en un graf ponderat.*
- *Resoldre el TSP.*

Queda clar que per a les tres versions d'un problema, l'ordre de dificultat, de menys a més, seria *decisió, càlcul, optimització*. Per exemple, si resolem el TSP en un graf G, automàticament estem trobant un circuit hamiltonià a G, cosa que ens indica que G és hamiltonià.

Problemes resolubles i problemes irresolubles

Un problema és **irresoluble** si no existeix –ni pot existir– cap algorisme (convencional) que el resolgui.

A l'hora de parlar de problemes resolubles o irresolubles, habitualment pensem sempre en problemes de decisió. L'explicació és que qualsevol problema pot descompondre's en una seqüència de problemes de decisió, de manera que solucionant aquests problemes de decisió queda resolt el problema inicial.

En aquest cas, se sol parlar de problemes **decidibles** (resolubles) o **indecidibles** (irresolubles).

Problemes resolubles i problemes irresolubles

Està demostrat que existeixen molts problemes indecidibles (en realitat, el conjunt de problemes indecidibles és infinit no numerable). L'argument per a veure-ho es pot resumir així:

Tot algorisme A admet una codificació binària $< A >$, de forma que $A = A_i$ on i és el nombre decimal corresponent al nombre binari $< A >$. D'aquesta manera, el conjunt d'algorismes és $\{A_0, A_1, A_2, \ldots\}$ que és infinit numerable.

Problemes resolubles i problemes irresolubles

Si codifiquem les dades d'entrada de qualsevol problema de decisió com un nombre natural, llavors podem identificar tot problema de decisió amb una funció $f : \mathbb{N} \longrightarrow \{0, 1\}$. Donada una funció d'aquest estil també li podem fer correspondre un problema de decisió ("és $f(n) = 1$?"). El conjunt de funcions $\{f : \mathbb{N} \longrightarrow \{0, 1\}\}$, també el podem identificar amb el conjunt de tots els vectors binaris d'infinites coordenades (serien les imatges de 1, 2, ...). Ja sabem que aquest conjunt és no numerable (vegeu el primer capítol).

Per tant no es pot definir cap aplicació injectiva del conjunt de problemes de decisió cap al conjunt d'algorismes. O sigui, que hi ha problemes als quals no els podem assignar un algorisme. Intuïtivament –i molt informalment– es podria dir que "hi ha més problemes que algorismes".

Problemes resolubles i problemes irresolubles

Observem però, que si un problema de decisió té un un nombre finit d'arguments possible, llavors és *trivialment decidible*.

Per exemple, si P és un problema amb 2 arguments possibles $\{w_1, w_2\}$, llavors podríem pensar en 4 algorismes $A1$, $A2$, $A3$ i $A4$, de forma que $A1$ sempre respon que sí; $A2$ sempre respon que no; $A3(w_1)$ respon que sí; $A3(w_2)$ respon que no; $A4(w_1)$ respon que no i $A4(w_2)$ respon que sí. Altra vegada resulta que un dels quatre algorismes ens està solucionant correctament el problema P. Per tant, P és decidible.

El problema de la correspondència de Post (PCP)

Sigui Σ un alfabet (per exemple, el binari). Tenim dues llistes $A = \{w_1, \ldots, w_k\}$ i $B = \{x_1, \ldots, x_k\}$ de paraules sobre Σ. El problema de la correspondència de post (*Postman Correspondence Problem*) o PCP demana si existeix alguna seqüència finita d'índexs $i_1, \ldots, i_m$ $(i_j \in \{1, \ldots, n\}, \ \forall j = 1, \ldots, m)$, tal que $w_{i_1} \cdots w_{i_m} = x_{i_1} \cdots x_{i_m}$.

Per exemple si $A = \{w_1 = 1, w_2 = 10111, w_3 = 10\}$ i $B = \{x_1 = 111, x_2 = 10, x_3 = 0\}$, llavors podem veure que $2, 1, 1, 3$ és una solució ja que

$$w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3 = 101111110.$$

En canvi, si $A = \{w_1 = 10, w_2 = 011, w_3 = 101\}$ i $B = \{x_1 = 101, x_2 = 11, x_3 = 011\}$, no és difícil veure que no hi ha solució.

Problemes resolubles i problemes irresolubles

Si pensem en un algorisme exhaustiu (que va provant totes les combinacions de longitud 1, després les de longitud 2, etc.) resulta que, si hi ha solució, acabarà trobant-la, però si no hi ha solució, l'algorisme no para mai i, per tant, mai acabem sabent si hi ha o no solució. Es pot demostrar que PCP és indecidible.

El problema de la parada (HP)

Donat un algorisme A qualsevol amb unes dades d'entrada w qualssevulla, el problema de la parada (*Halting Problem*) o HP demana si l'algorisme A amb entrada w, $A(w)$, s'aturarà o no.

Aquest problema és necessàriament indecidible. Cas contrari, podríem donar-li l'anterior algorisme A_{PCP} per al problema PCP. Si A_{HP} fos un algorisme per al HP que s'aturés sempre, llavors podríem saber si $A_{PCP}(w)$ s'atura o no. Si s'atura és que PCP$(w) =$Sí, si no s'atura és que PCP$(w) =$No. Però com que està demostrat que el PCP és indecidible, això és impossible.

Definició 1.

Denotem per $CT(P)$ i $CE(P)$, la complexitat temporal i espacial, respectivament, d'un problema P. Aleshores, per a cada funció de complexitat f, definim les classes de problemes:

(i) $DTIME(f(n)) = \{P \mid CT(P) \leq O(f(n))\}$.

(ii) $DSPACE(f(n)) = \{P \mid CE(P) \leq O(f(n))\}$.

Si pensem que per a visitar qualsevol posició de memòria, cal fer una pas o transició, ens resulta que $DTIME(f(n)) \subseteq DSPACE(f(n))$. De tota manera, ens limitarem a estudiar complexitats temporals.

Definició 2.

Un problema P està a la **classe** $\mathcal{P}$ *($P \in \mathcal{P}$ o P és un problema $\mathcal{P}$) si $CT(P) = O(n^r)$, per a algun $r \in \mathbb{N}$. En altres paraules:*

$$\mathcal{P} = \bigcup_{r \geq 1} DTIME(n^r).$$

Aquests són els problemes tractables, els de complexitat temporal polinòmica. Els que estan fora de $\mathcal{P}$ són els intractables.

Exemple 4.

- *Problemes d'ordenació de vectors.*
- *Trobar camins òptims en un graf ponderat.*
- *Resoldre el problema del carter xinès en un graf simètric ponderat.*

Considerem el problema de si un graf G és hamiltonià i suposem que hem trobat una solució positiva del problema, o sigui que G és hamiltonià.

Donar un *certificat* de la solució voldria dir donar un circuit hamiltonià a G. Resulta clar que verificar aquest certificat ho podem fer en temps polinòmic.

Intuïtivament, sempre sembla més fàcil verificar una solució positiva que no pas trobar-la (tot i que això no està demostrat). En el cas dels problemes $\mathcal{P}$, és evident que sempre es pot verificar un certificat de solució positiva en temps polinòmic, ja que, de fet, trobar una tal solució ja es pot fer en temps polinòmic.

Definició 3.

Un problema P està a la **classe** $\mathcal{NP}$ *($P \in \mathcal{NP}$ o P és un problema $\mathcal{NP}$) si un certificat de solució positiva sempre és verificable en temps polinòmic.*

Clarament $\mathcal{P} \subseteq \mathcal{NP}$ però avui en dia no tenim una demostració acceptada de que $\mathcal{P} \neq \mathcal{NP}$.

És un dels sis problemes no resolts (dels set que hi havia) anomenats *Millennium prize problems* pels quals el *Clay Mathematics Institute* paga un milió de dòlars a qui en resolgui un.

Sigui P un problema de decisió i Σ^* el conjunt de possibles arguments de P, de forma que $P(w) \in \{S, N\}$, per a tot $w \in \Sigma^*$.

Definició 4.

El problema **complementari** *de P, denotat per $\overline{P}$, és aquell tal que $\overline{P}(w) \neq P(w), \ \ \forall w \in \Sigma^*$.*

Exemple 5.

(i) P: *És G hamiltonià?* $\overline{P}$: *És G no hamiltonià?*

(ii) P: *És G connex?* $\overline{P}$: *És G disconnex?*

Donada una classe de complexitat $\mathcal{C}$ qualsevulla, es defineix la co-classe $co - \mathcal{C}$ com el conjunt de problemes complementaris dels que estan a la classe $\mathcal{C}$. A nosaltres ens interessa el següent cas concret.

Definició 5.

$$co - \mathcal{NP} = \{P \mid \overline{P} \in \mathcal{NP}\}.$$

Clarament $co - \mathcal{P} = \mathcal{P}$ i no se sol parlar mai de $co - \mathcal{P}$. En canvi, no se sap si $\mathcal{NP}$ és igual o no a $co - \mathcal{NP}$, encara que majoritàriament es pensa que $\mathcal{NP} \neq co - \mathcal{NP}$. Per exemple, en el cas del problema "és G hamiltonià?", verificar una solució positiva pot fer-se en temps polinòmic, però no es coneix cap mètode per verificar una solució negativa en temps polinòmic.

Proposició 1.

Tot problema de la classe $\mathcal{P}$ és també un problema $\mathcal{NP}$ i $co-\mathcal{NP}$:

$$\mathcal{P} \subseteq \mathcal{NP} \cap co-\mathcal{NP}.$$

Proposició 2.

Si fos cert que $\mathcal{P} = \mathcal{NP}$, llavors $\mathcal{NP}$ també seria igual a $co-NP$.

Corol·lari 1.

Si $\mathcal{NP} \neq co-\mathcal{NP}$, llavors segur que $\mathcal{P} \neq \mathcal{NP}$.

Observem però, que podria passar $\mathcal{P} \neq \mathcal{NP} = co-\mathcal{NP}$. Per tant, tenim tres situacions possibles.

Definició 6.

Un problema P_1 es **redueix (polinòmicament)** *a un altre P_2 si existeix una transformació T, $w \longrightarrow T(w)$, calculable en temps polinòmic, tal que $P_1(w) = P_2(T(w))$, per a totes les dades d'entrada possibles w.*

La transformació s'ha de poder fer en temps polinòmic, d'aquesta manera, si P_2 té una complexitat almenys polinòmica, resulta que P_1 *no és més difícil que* P_2. Efectivament, per resoldre P_1, sempre podríem fer la transformació –que no modifica essencialment la complexitat– i llavors resoldre P_2. Ara bé, P_1 podria ser més fàcil que P_2, ja que res impedeix l'existència d'un algorisme de menor complexitat per resoldre P_1.

Els problemes SAT i $3-SAT$

Tenim un conjunt de variables lògiques $\{v_1, \ldots, v_n\}$, ($v_i \in \{0, 1\}$, per a tot $i = 1, \ldots, n$). Diem que una variable v_i és *certa* si, i només si, $v_i = 1$. Un literal és una variable lògica complementada o no: v_i o $\bar{v}_i$. Una clàusula és una suma (*or*) de literals. El problema SAT (*satisfiability*) demana si donat un producte lògic (*and*) de clàusules existeix una assignació de veritat (una assignació de valors a les variables) de forma que el producte es satisfaci, és a dir, que tingui resultat 1 o cert.

Per exemple, si tenim el producte de clàusules:

$$(v_1 + v_2)\bar{v}_3(\bar{v}_2 + v_3 + v_4 + \bar{v}_5)(v_4 + v_5),$$

assignant els valors $v_1 = v_3 = v_5 = 0$ i $v_2 = v_4 = 1$, tenim que el producte val 1.

Si afegim la restricció de que totes les clàusules tinguin exactament tres literals, llavors es diu el problema $3-SAT$. Sembla que el $3-SAT$ hauria de ser més fàcil que el SAT, però la realitat és que són igual de difícils, ja que podem reduir el SAT al $3-SAT$:

Teorema 1.

El problema SAT pot reduir-se al $3-SAT$.

Parlarem sempre de reduccions polinòmiques. Si un problema P_1 es pot reduir a un altre P_2, ho escriurem $P_1 \hookrightarrow P_2$.

Proposició 3.

Suposem que $P_1 \hookrightarrow P_2$. *Aleshores*

(i) $P_2 \in \mathcal{P} \implies P_1 \in \mathcal{P}$.

(ii) $P_2 \in \mathcal{NP} \implies P_1 \in \mathcal{NP}$.

Definició 7.

Un problema P és $\mathcal{NP}-hard$ ($\mathcal{NP}-$difícil o $\mathcal{NP}-dur$) si tot problema de la classe $\mathcal{NP}$ pot reduir-se a P.

Definició 8.

Un problema P és $\mathcal{NP}-complet$ si és $\mathcal{NP}-hard$ i és $\mathcal{NP}$.

Clarament les reduccions són transitives:

$$\left.\begin{array}{l} P_1 \hookrightarrow P_2 \\ P_2 \hookrightarrow P_3 \end{array}\right\} \Longrightarrow P_1 \hookrightarrow P_3.$$

Observem que si un problema P és $\mathcal{NP}-$hard i es demostrés que $P \in \mathcal{P}$, llavors, per transitivitat, tindríem que $\mathcal{P} = \mathcal{NP}$.
De fet, per demostrar que un determinat problema P és $\mathcal{NP}-$complet, cal fer dos passos:

(i) Demostrar que $P' \hookrightarrow P$, per a algun problema $P' \in \mathcal{NP}-$hard conegut (aleshores, per transitivitat, tot problema $\mathcal{NP}$ pot reduir-se a P).

(ii) Demostrar que $P \in \mathcal{NP}$. Això no acostuma a ser difícil, es tracta de provar que un certificat positiu pot validar-se en temps polinòmic.

Naturalment, això exigeix que es conegui almenys un primer problema $\mathcal{NP}-$hard.

Un algorisme evident per solucionar el problema SAT seria provar totes les assignacions possibles de valors. Si la resposta fos que el producte no es pot satisfer, és obvi que no ho sabríem fins haver provat les 2^n possibles assignacions. O sigui que seria una complexitat exponencial. De fet, no es coneix cap algorisme que sigui essencialment millor.

Teorema 2 (Cook).

El problema SAT és $\mathcal{NP}-$complet.

El teorema de Cook és molt important ja que va servir per trobar un primer problema $\mathcal{NP}-$complet.

Com que SAT pot reduir-se a $3-SAT$, amb el teorema de Cook obtenim immediatament el següent.

Corol·lari 2.

El problema $3-SAT$ és $\mathcal{NP}-$complet.

En canvi, el $2-SAT$ resulta ser un problema $\mathcal{P}$. El $3-SAT$ ha estat el problema més utilitzat per a demostrar la $\mathcal{NP}-$completitud d'altres problemes.

Alguns problemes intractables

En teoria de grafs hi ha molts problemes que se sap que són $\mathcal{NP}-$complets. La situació habitual és que el problema de decisió és $\mathcal{NP}-$complet i la versió d'optimització és $\mathcal{NP}-$hard i es creu que no és $\mathcal{NP}$. Anem a veure alguns d'aquests problemes $\mathcal{NP}-$complets.

Sigui $G(V, A)$ un graf simètric d'ordre $|V| = n$.

Definició 9.

Un **recobriment de vèrtexs (Vertex Cover, VC)** *és un subconjunt $S \subseteq V$ tal que tota aresta és incident a algun vèrtex de S.*

Els VC interessants seran els minimals i/o mínims (recordem que mínim implica minimal, però no a l'inrevés).

Alguns problemes intractables

Definició 10.

Una **colla (Clique, CL)** *és un subgraf complet de* G*.*

Naturalment seran interessants les CL maximals i/o màximes.

Definició 11.

Un **conjunt independent (Independent Set, IS)** *de vèrtexs és un subconjunt* $S \subseteq V$ *tal que a* S *no hi ha adjacències.*

En aquest cas, seran interessants els IS maximals i/o màxims.

Les següents propietats són òbvies.

Proposició 4.

Sigui $G(V, A)$ un graf simètric.

(i) *$S \subseteq V$ és un VC (minimal) si, i només si, $V \setminus S$ és un IS (maximal).*

(ii) *$S \subseteq V$ és una CL (maximal) si, i només si, S és un IS (maximal) a $\overline{G}$.*

Alguns problemes intractables

Atès que construir el graf complementari pot fer-se en temps polinòmic respecte del nombre de vèrtexs (o d'arestes), tenim que els següents problemes són equivalents:

- Trobar un VC minimal.
- Trobar un IS maximal.
- Trobar una CL maximal.

Les versions com a problemes de decisió serien: donat un valor k, $1 \leq k \leq n$,

- Existeix un VC amb k o menys vèrtexs?
- Existeix un IS amb k o més vèrtexs?
- Existeix una CL amb k o més vèrtexs?

És evident que donada una solució a qualsevol dels problemes, podem verificar-la amb, com a molt, $O(|A|) \leq O(n^2)$ passos. Per tant, els tres problemes estan a $\mathcal{NP}$. A més tenim que VC pot reduir-se a IS i a l'inrevés. També tenim que IS pot reduir-se a CL i a l'inrevés.

Teorema 3.

VC, IS i CL són problemes $\mathcal{NP}-$complets.

Alguns problemes intractables

Anem a veure alguns problemes intractables més, sense demostrar-ho.

- Donat k, $1 \leq k \leq |V| = n$, existeix una γ-coloració, on $\gamma \leq k$? Per a $k = 1$ o $k = 2$ és molt fàcil però per a $k = 3$ el problema ja és $\mathcal{NP}-$complet.
- És G hamiltonià? També és un problema $\mathcal{NP}-$complet.
- Té G un recorregut que passi per tots els vèrtexs amb cost menor o igual que k? El problema també és $\mathcal{NP}-$complet. La versió d'optimització, resoldre el TSP, és $\mathcal{NP}-$hard. Observem que per verificar una solució, sembla que hauríem de calcular tots els circuits hamiltonians i veure que la solució proposada és realment la de menor cost. Per tant, sembla que el TSP no és un problema $\mathcal{NP}$.

Alguns problemes intractables

- El problema de la motxilla (*Knapsack*) és el següent: tenim una capacitat C (per exemple, el volum d'un portamaletes) i diferents objectes (maletes) de pesos (volums) w_i i valors v_i. Volem carregar objectes maximitzant el valor $\sum_i v_i$ sense sobrepassar la capacitat, $\sum_i w_i \leq C$. Aquest problema és $\mathcal{NP}-$hard.
  La versió de decisió seria: donats els valors $\{w_1, \ldots, w_n\}$, $\{v_1, \ldots, v_n\}$, C i k, determinar si existeix $I \subseteq \{1, \ldots, n\}$, tal que
  $$\sum_{i \in I} v_i \geq k \quad \text{i} \quad \sum_{i \in I} w_i \leq C.$$
  El problema és clarament $\mathcal{NP}$ i, de fet, és $\mathcal{NP}-$complet.