Proiectarea algoritmilor: Probleme $\mathbb{NP}$-complete

Dorel Lucanu

Faculty of Computer Science
Alexandru Ioan Cuza University, Iași, Romania
dlucanu@info.uaic.ro

PA 2014/2015

Plan

Reamintire: probleme de decizie

O problemă de decizie este o problemă P la care răspunsul este de forma "DA" sau "NU" (echivalent, "true" sau "false"). Mai precis, pentru orice instanță $p \in P$, $P(p) \in \{"DA", "NU"\}$ ($P(p) \in \{"true", "false"\}$).

O problemă de decizie este reprezentată în general printr-o pereche (instance, question) (sau (instanță, întrebare)).

Exemplu de problemă de decizie: SAT

SAT

Instance O formulă propoziţională F cu variabile din $X = \{x_1, \ldots, x_n\}$.

Question Există o atribuire $a : X \to \{0, 1\}$ (sau $a : X \to \{false, true\}$) care satisface F?

Exemplu de instanță:

$X = \{x, y, z\}$

$F = ((x \wedge \neg y) \vee (\neg x \wedge z)) \wedge ((\neg y \vee z) \wedge (x \vee \neg y \vee \neg z))$

Exemplu de problemă de decizie: CSAT

CSAT

Instance O formulă propoziţională F cu variabile din $X = \{x_1, \ldots, x_n\}$ și în formă normală conjunctivă.

Question Există o atribuire $a : X \to \{0, 1\}$ (sau $a : X \to \{false, true\}$) care satisface F?

Exemplu de instanță:

$X = \{x, y, z\}$

$F = (x \vee \neg y \vee z) \wedge (\neg x \vee z) \wedge (\neg x \vee \neg y \vee z \vee x)$

Exemplu de problemă de decizie: DSAT

DSAT

Instance O formulă propoziţională F cu variabile din $X = \{x_1, \ldots, x_n\}$ și în formă normală disjunctivă.

Question Există o atribuire $a : X \to \{0, 1\}$ (sau $a : X \to \{false, true\}$) care satisface F?

Exemplu de instanță:

$X = \{x, y, z\}$

$F = (x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge z) \vee (\neg x \wedge \neg y \wedge y \wedge z)$

Exemplu de problemă de decizie: k-CSAT

k-SAT

Instance O formulă propoziționalăă F cu variabile din $X = \{x_1, \ldots, x_n\}$, în formă normală conjunctivă și fiecare clauză (disjuncție) are exact k literali.

Question Există o atribuire $a : X \to \{0, 1\}$ (sau $a : X \to \{false, true\}$) care satisface F?

Cazuri de interes: $k = 3$ (3-CSAT), $k = 2$ (2-CSAT).

Exemplu de instanță 3-CSAT:

$X = \{x, y, z\}$
$F = (x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg y \vee z \vee x)$

Exemplu de instanță 2-CSAT:

$X = \{x, y, z\}$
$F = (x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (\neg x \vee z) \wedge (\neg y \vee z)$

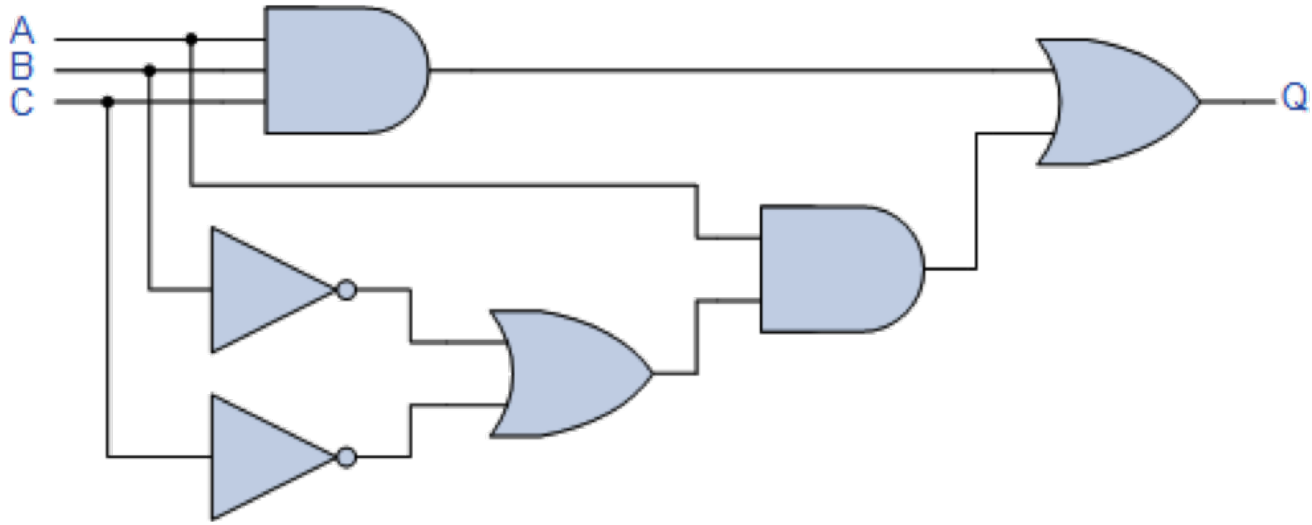
Exemplu de problemă de decizie: CIRCUIT-SAT

CIRCUIT-SAT

Instance Un circuit boolean construit cu porțile logice AND, OR, NOT, și care are o singură ieșire.

Question Există o atribuire de valori $\{0, 1\}$ pentru intrările circuitului astfel încât valoarea de la ieșire să fie 1?

Exemplu de circuit boolean



Reprezentarea problemelor de decizie ca limbaje

Un limbaj L este o mulțime de șiruri (cuvinte) peste un alfabet dat Σ ($L \subseteq \Sigma^*$).

Putem presupune că o instanță (intrare) a unei probleme este codificată de un șir w peste un alfabet dat A. Putem presupune că $\Sigma = \{0, 1\}$ (w este un șir de biți).

Limbajul definit de o problemă de decizie P este L_P este mulțimea codificărilor w ale instanțelor pentru care răspunsul este "DA" (true).

O problemă de decizie P poate fi reprezentată acum ca o problemă de apartenență la un limbaj:

Instance Σ, $L \subseteq \Sigma^*$, $w \in \Sigma^*$.
Question $w \in L$?

Dacă A este un algoritm care rezolvă P, atunci spunem că L este acceptat de A (sau că A acceptă L).

Reamintire: algoritmi nedeterminiști

Activitatea unui algoritm nedeterminist se desfășoară în două etape: într-o primă etapă "se ghicește" o anumită structură S și în etapa a doua se verifică dacă S satisface o condiția de rezolvare a problemei. Putem adăuga "puteri magice de ghicire" unui limbaj de programare adăugându-i o funcție de forma:

> `random`(N) – care întoarce un număr aleatoriu din mulțimea $\{0, 1, \ldots, n-1\}$.

Observație. Alternativ se poate considera un operator de alegere nedeterministă:
$S_1 \mid S_2 \mid \cdots \mid S_n$ cu semantica
$S_1 \mid S_2 \mid \cdots \mid S_n \Rightarrow S_i,\ i = 1, \ldots, n$

Pentru a ști dacă verificarea s-a terminat cu succes sau nu adăugă și două instrucțiuni de terminare:

> `success` – care semnalează terminarea verificării (și a algoritmului) cu succes, și
>
> `failure` – care semnalează terminarea verificării (și a algoritmului) fără succes.

Probleme de decizie rezolvate de algoritmi nedeterminiști

Intuitiv, un algoritm nedeterminist A rezolvă o problemă P dacă pentru orice instanță $p \in P$ există un calcul al lui A care pleacă dintr-o configurație inițială ce codifică p, se termină cu succes și oferă răspunsul $P(p)$ codificat în configurația finală.

Un caz aparte îl constituie problemele de decizie:

Definiție

Un algoritm nedeterminist rezolvă o problemă de decizie dacă se termină cu succes atunci și numai atunci când răspunsul este "DA" (true).

Clasele $\mathbb{P}$ și $\mathbb{NP}$

$\mathbb{P}$ = clasa problemelor P pentru care există un algoritm determinist care rezolvă P în timp polinomial

$\mathbb{NP}$ = clasa problemelor P pentru care există un algoritm nedeterminist care rezolvă P în timp polinomial.

Deoarece orice algoritm determinist este un caz particular de algoritm nedeterminist, are loc incluziunea:

$$\mathbb{P} \subseteq \mathbb{NP}$$

O definiție alternativă pentru $\mathbb{NP}$

Definiția alternativă este bazată pe o verificare deterministă și din acest motiv este poate mai intuitivă.

Considerăm problemele reprezentate ca limbaje.

Un limbaj L este verificat de un algorithm A dacă pentru orice șir $x \in L$, există un șir y astfel încât A răspunde "DA" pentru intrarea $x + y$ ($+$ desemnează concatenarea de șiruri). Șirul y este un certificat pentru apartenența lui x la L. De notat că nu se verifică dacă $x \in L$.

Clasa $\mathbb{NP}$ este definită ca fiind clasa limbajelor, definind probleme de decizie, ce pot fi verificate în timp polinomial. Adică există un algoritm A, care pentru orice x din L verifică dacă intr-adevar $x \in L$ utilizând un certificat y. Dimensiunea unei instanțe este lungimea n a lui x. Dacă A face verificarea în timpul $p(n)$ (p un polinom), atunci lungimea certificatului y va fi $\leq p(n)$.

Echivalența celor două definiții

Exercițiu. Să se arate că cele două definiții sunt echivalente.

$\mathbb{P} \subset \mathbb{NP}$ sau $\mathbb{P} = \mathbb{NP}$?

Algoritmii nedeterminiști constituie un instrument mult mai puternic decât cei determiniști și până acum nu s-a putut găsi o metodă prin care algoritmi nedeterminiști să poată fi convertiți în algoritmi determiniști în timp polinomial.

Aceste argumente conduc la ideea că răspunsul este $\mathbb{P} \subset \mathbb{NP}$, dar demonstrația pare a ține de fenomene pe care nu le stăpânim deocamdata.

Plan

Reamintire: Reducerea Turing/Cook)

Problema P se reduce polinomial la problema (rezolvabilă) Q, notăm $P \propto Q$, dacă se poate construi un algoritm care rezolvă P după următoarea schemă:

1. se consideră la intrare o instanță p a lui P;
2. preprocesează în timp polinomial intrarea p
3. se apelează algoritmul pentru Q, posibil de mai multe ori (un număr polinomial)
4. se postprocesează rezultatul dat de Q în timp polinomial

Dacă pașii de preprocesare și postprocesare necesită $O(g(n))$ timp, atunci scriem $P \propto_{g(n)} Q$. Dacă $g(n)$ este un polinom dar nu interesează, scriem doar $P \propto Q$.

Reducerea Turing/Cook se mai numește și reducerea "many-many".

Reducerea relativ la un oracol

Atunci când este definită în modelul de calcul al mașinilor Turing, algoritmul pentru Q este abstractizat prntr-un oracol (pentru a prinde și cazul când nu se știe dcă există un algoritm pentru Q; de exemplu când Q este problema opririi).

Rolul oracolului este de a furniza un model de calcul care să poată răpunde la întrebări de forma "Dacă știu să rezolv Q, atunci ce altceva mai pot rezolva?" (complexitate relativă)

Reamintire: Reducerea Karp

Se consideră P și Q probleme de decizie.
Problema P se reduce polinomial la problema (rezolvabilă) Q, notăm $P \propto Q$, dacă se poate construi un algoritm care rezolvă P după următoarea schemă

1. se consideră la intrare o instanță p a lui P;
2. preprocesează în timp polinomial intrarea p
3. se apelează (o singură dată) algoritmul pentru Q
4. răspunsul pentru Q este același cu cel al lui P (fără postprocesare)

Dacă pasul de preprocesare necesită $O(g(n))$ timp, atunci scriem $P \propto_{g(n)} Q$. Dacă $g(n)$ este un polinom dar nu interesează, scriem doar $P \propto Q$.

Reducerea Karp se mai numește și reducerea "many-one".

Relația dintre cele două reduceri

Reducerea Karp implică reducere Turing/Cook.

Există perechi de probleme pentru care se cunoaște reducere Turing/Cook dar nu se știe dacă există reducere Karp.

Reducerea Karp permite analize mai fine. De exemplu dacă vom considera problema $\overline{\text{3-SAT}}$, care decide dacă o formulă în forma 3-CNF este nesatisfiabilă, atunci avem 3-SAT$\propto \overline{\text{3-SAT}}$ cu reducerea Turing/Cook (este suficient să negăm răspunsul dat de oracol). Vom vedea mai târziu că aceasta influențează direct relația dintre clasele $\mathbb{NP}$ și co-$\mathbb{NP}$.

Echivalența

Problemele P și Q sunt echivalente (Karp sau Turing/Cook) dacă $P \propto Q$ și $Q \propto P$.

Este surprinzător de observat că numărul claselor de echivalență este extrem de mic în raport cu întreaga clasă de probleme. Complexitatea unei probleme se poate determina arătând că problema este echivalentă (computațional) cu una dintr-o listă relativ scurtă.

Proprietăți ale reducerii

Definiție

O clasă $\mathcal{C}$ de probleme este închisă la o reducere (Karp sau Turing/Cook) dacă ori de câte ori $P \propto Q$ și $Q \in \mathcal{C}$, rezultă $P \in \mathcal{C}$.

Teoremă

$\mathbb{P}$ este închisă la ambele reduceri, Karp ș Turing/Cook.

Exercițiu. Să se demonstreze teorema.

Teoremă

Ambele reduceri, Karp și Turing/Cook, sunt tranzitive.

Exercițiu. Să se demonstreze teorema.

Reducerea problemelor de optim la problemă de decizie

RUCSAC II OPTIM

Input $w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, n$

$M \in \mathbb{Z}$

Output $x_i \in \{0, 1\}, i = 1, \ldots, n$ cu proprietatea maximizează funcția obiectiv și obiectele alese încap în rucsac:

$$\max \sum_{i=1}^{n} x_i \cdot p_i$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq M$$

Reducerea problemelor de optim la problemă de decizie

RUCSAC II DECIZIE

Instance $w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, n$
$M \in \mathbb{Z}_+$
$K \in \mathbb{Z}_+$ (scop)

Question Există $x_i \in \{0, 1\}, i = 1, \ldots, n$ cu proprietatea că scopul este atins și obiectele alese încap în rucsac:

$$\sum_{i=1}^{n} x_i \cdot p_i \geq K$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq M?$$

Theorem

RUCSAC II OPTIM $\propto$ *RUCSAC II DECIZIE (reducere Turing/Cook).*

Ipoteză de lucru

De acum considerăm numai probleme de decizie și reducere Karp.

Completitudine

Fie $\mathcal{C}$ o clasă de probleme.

Definiție

O problemă P este $\mathcal{C}$-dificilă (hard) dacă $Q \propto P$ pentru orice $Q \in \mathcal{C}$.

De remarcat că nu trebuie să avem $P \in \mathcal{C}$.

Definiție

O problemă P este $\mathcal{C}$-completă dacă este $\mathcal{C}$-dificilă ș $P \in \mathcal{C}$.

$\mathcal{C} = \mathbb{P}$: probleme $\mathbb{P}$-complete

$\mathcal{C} = \mathbb{NP}$: probleme $\mathbb{NP}$-complete
S-a sperat că se poate arăta că problemele $\mathbb{NP}$-complete sunt în $\mathbb{NP} \setminus \mathbb{P}$.

Teorema lui Cook

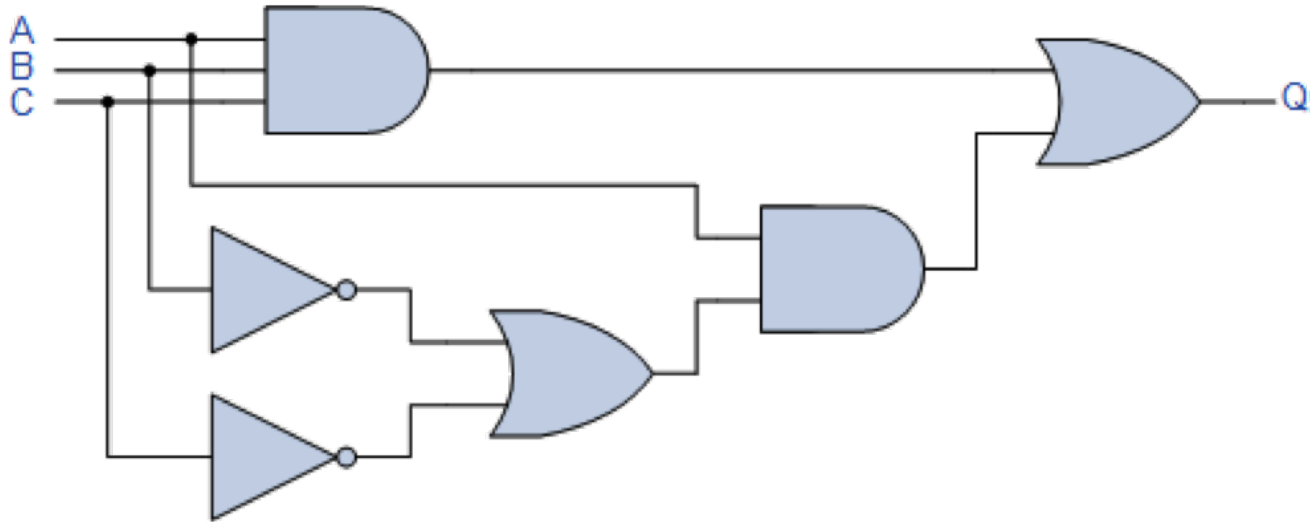
Teoremă

CSAT este $\mathbb{NP}$-completă.

CSAT a fost prima problemă pentru care s-a demonstrat $\mathbb{NP}$-completitudinea (1971), utilizându-se altă demonstrație.

Vom demonstra teorema lui Cook după ce vom arăta $\mathbb{NP}$-completitudinea unei probleme mai simple, dar foarte apropiată de SAT.

Exemplu de circuit boolean



CIRCUIT-SAT este $\mathbb{NP}$-completă

Teoremă

CIRCUIT-SAT este $\mathbb{NP}$-completă.

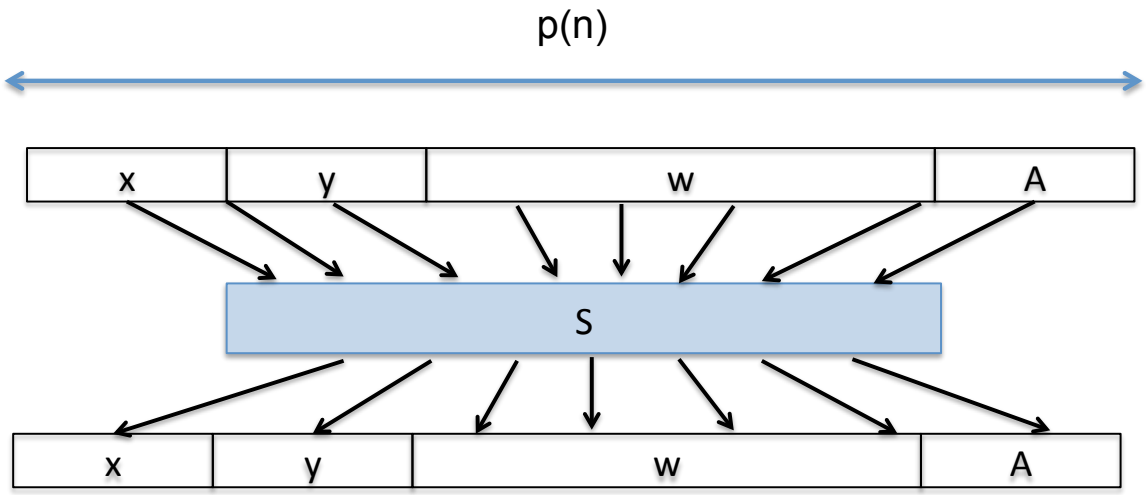
Demonstrație (schiță).

1. CIRCUIT-SAT este în $\mathbb{NP}$. Exercițiu.
2. Vom arăta că CIRCUIT-SAT este $\mathbb{NP}$-dificilă.

Fie P o problemă din $\mathbb{NP}$. Există un algoritm determinist A astfel încât P este verificat de A. Rezultă că pentru orice instanță $x \in P$ cu $P(x) =$ "DA" există un certificat y cu proprietatea că se termină cu succes pentru intrarea $x + y$ în timp polinomial. Fie $n = |x|$ și $p(n)$ timpul pentru cazul cel mai nefavorabil al lui A.

Presupunem că memoria peste care lucrează A este o secvență de biți și că toate operațiile care se numără sunt la nivel de bit. Deoarece numărul de biți vizitați de A este cel mult $p(n)$, rezultă că activitatea unui pas al lui A poate fi descrisă de un CPU S, format din porțile logice AND, OR, NOT. Mai mult, numărul porților este cel mult $O(p(n)^2)$.

CIRCUIT-SAT este $\mathbb{NP}$-completă



x este instanța lui P,
y este certificatul,
W este memoria de lucru (e.g., valorile variabilelor din program și contorul de program),
A este algoritmul A reprezentat ca șir de biți.

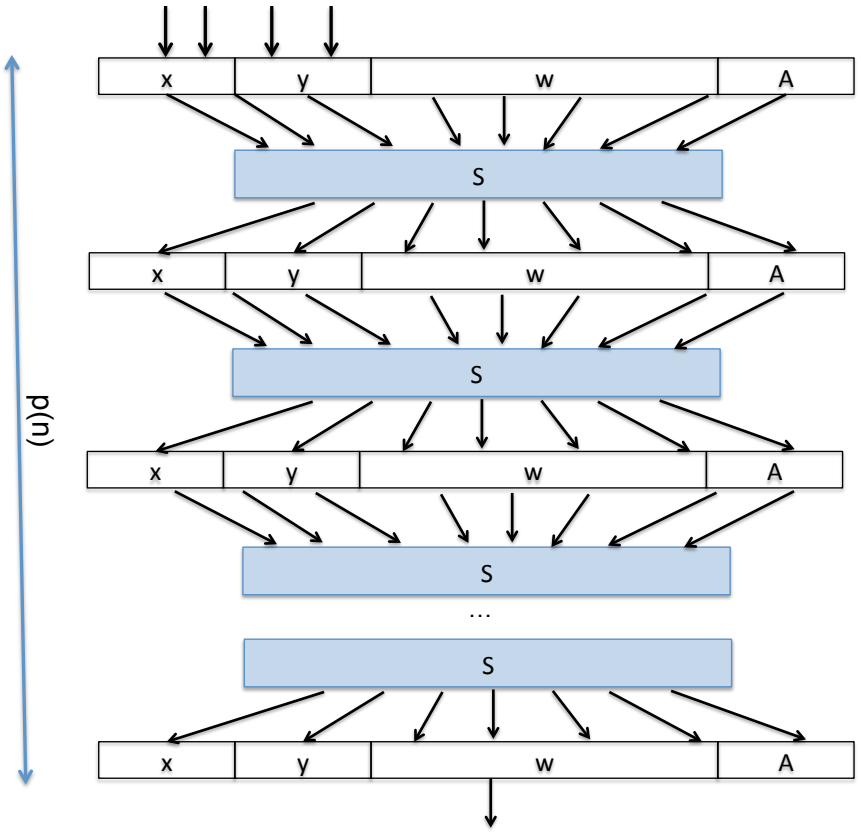
CIRCUIT-SAT este $\mathbb{NP}$-completă

Execuția algorimului A pentru o intrare x de dimensiune n poate fi simultaă de un circuit boolean C compus din $p(n)$ copii ale lui S (câte una pentru fiecare pas) așezate astfelîncât ieșirea de la unul este intrare pentru următorul.

Se daugă la ultima copie o ieșire care va avea valoarea 1 dacă și numai dacă A se termină cu succes (certifică pe x).

Pentrun x dat, intrarile circuitului C depind doar de y.

CIRCUIT-SAT este $\mathbb{NP}$-completă



CIRCUIT-SAT este $\mathbb{NP}$-completă

Trebuie să arătăm că există o atribuire de valori la intrare astfel încât circuitul C are la ieșire 1 dacă și numai dacă A se termină cu succes pentru un certicat y.

Dacă există o atribuire de valori la intrare astfel încât ieșirea lui C este 1, atunci certificatul y este dat de valorilor intrărilor corespunzătoare lui y. Algoritmul A se termină cu succes pentru că C simulează activitatea sa și are ieșirea numai în cazul când A certifică apartenența lui x la P.

Reciproc, dacă a lgoritmul A se termină cu succes pentru x cu certificatul y, atunci atribuirea intrărilor va fi cea dată de acest certificat.

Demonstrarea $\mathbb{NP}$-completitudinii prin reducere

Theorem

Dacă Q este $\mathbb{NP}$-completă, P este în $\mathbb{NP}$ și $Q \propto P$, atunci P este $\mathbb{NP}$-completă.

Se utilizează faptul că $\propto$ este tranzitivă.

Teorema lui Cook (continuare)

Teoremă

CSAT este $\mathbb{NP}$-completă.

Demonstrație (schiță).
1. CSAT este în $\mathbb{NP}$. Exercițiu.
2. Vom arăta că CIRCUIT-SAT $\propto$ CSAT.

Fiecare poartă logică g poate fi descrisă cu o formulă F_g astfel încât F_g este satisfiabilă dacă și numai dacă valoarea de la ieșire corespunde operației efectuate peste intrări. Exemple

NOT

Presupunem că intrarea este modelată de variabila u și ieșirea de variabila v. F_{NOT} este $(u \vee v) \wedge (\neg u \vee \neg v)$.)

AND

Presupunem că intrarea este modelată de variabilele u, v și ieșirea de variabila w. F_{AND} este $(u \vee \neg w) \wedge (v \vee \neg w) \wedge (w \vee \neg u \vee \neg v)$.

3-CSAT este $\mathbb{NP}$-completă

Teoremă

3-CSAT este $\mathbb{NP}$-completă.

Clauzele obținute din porțile booleene au lungimea 2 sau 3. Orice clauză de lungime 2 $u \vee v$ este eqchivalentă cu conjuncția a două clauze de lungime 3: $(u \vee v \vee w) \wedge (u \vee v \vee \neg w)$.

Demonstrarea $\mathbb{NP}$-completitudinii prin caz particular

Theorem

Dacă Q este $\mathbb{NP}$-completă, P este în $\mathbb{NP}$ și Q este un caz particular al lui P, atunci P este $\mathbb{NP}$-completă.

Corolar

SAT este $\mathbb{NP}$-completă.

Atenție la reducrea SAT $\propto_{g(n)}$ CSAT

Orice formulă booleană poate fi transformată în una echivalentă în formă normală conjunctivă:

- negația este împinsă în interior cu legile lui De Morgan:
  $\neg(u \vee v) \equiv \neg u \wedge \neg v$
  $\neg(u \wedge v) \equiv \neg u \vee \neg v$
- disjuncția de la ră dăcină este eliminată prin distributivitate:
  $u \vee (v \wedge w) \equiv (u \vee v) \wedge (u \vee w)$
- dubla negație: $\neg \neg u \equiv u$

Rezultă SAT $\propto_{g(n)}$ CSAT.
Reducerea nu este polinomială ($g(n)$ nu este polinom): lungimea formulei transformate poate fi exponențială comparativ cu lungimea formulei inițiale:
$(x \wedge y) \vee (u \wedge v) \equiv (x \vee u) \wedge (y \vee u) \wedge (x \vee v) \wedge (y \vee v)$

Din acest motiv această reducere nu poate fi utilizată pentru a arăta $\mathbb{NP}$-completitudinea lui SAT.

DSAT este în $\mathbb{P}$

Analog obținem SAT $\propto_{g(n)}$ DSAT, dar, din nou, nu putem deduce că DSAT este $\mathbb{NP}$-completă.

Mai mult, DSAT este în $\mathbb{P}$:
Exemplu: $F = (x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge z) \vee (\neg x \wedge \neg y \wedge y \wedge z)$

- o formulă DNF F este satisfiabilă $\iff$ există o clauză satisfiabiă
- o clauză este satisfiabilă $\iff$ nu include x și $\neg x$ pentru o variabilă x oarecare

Observație. În [LC08] există o afirmație greșită despre DSAT.

2-CSAT este în $\mathbb{P}$

- se alege o variabilă x și i se atribuie o valoare care satisface cel puțin o clauză
- apoi se calculează (prin propagare) valorilor variabilelor care apar în aceeași clauză cu x
- dacă procesul se oprește și au mai rămas clauze, atunci acestea sunt independente de atribuirea de valori pentru cele eliminate; procedeul continuă pentru clauzele rămase.

Exemplu:
$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (x3 \vee x4) \wedge (\neg x_3 \vee x_5) \wedge (\neg x_4 \vee \neg x_5) \wedge (\neg x_3 \vee x_4)$

INDEPENDENT-SET

Domeniul problemei:
Fie $G = (V, E)$ un graf. O submulțime $U \subseteq V$ se numește independentă dacă oricare două vârfuri din U nu sunt adiacente.

INDEPENDENT-SET
Instance Un graf $G = (V, E)$, un număr întreg $k \geq 0$.
Question Există o submulțime $U \subseteq V$ independentă cu cel puțin k elemente? ($|U| \geq k$).

INDEPENDENT-SET

Lemă

3-SAT $\propto$ INDEPENDENT-SET.

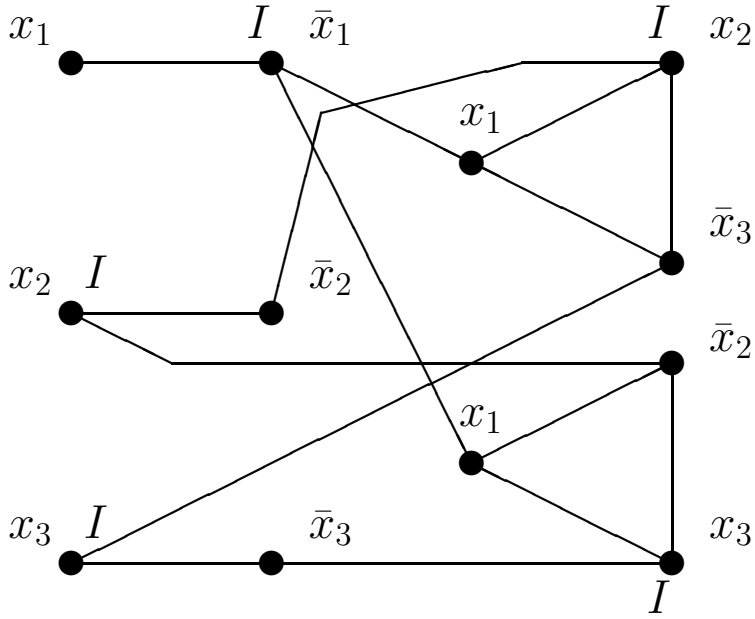
Fie F o formulă 3CNF cu variabilele $\{x_1, \ldots, x_n\}$ și m clauze $C_1, \ldots, C_m$. Construim G_F astfel:

- există $2n$ vârfuri-treaptă $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_n$ ($\overline{x}_i$ corespunde lui $\neg x_i$) și câte o muchie $[x_i \overline{x}_i]$;
- pentru fiecare clauză C_j se construiește o componentă distinctă care are ca vârfuri sunt etichetate cu literalii din clauze și există muchie între oricare două vârfuri ale componentei;
- pentru fiecare nod etichetat cu un x_i există o muchie ce leagă componenta cu opusul $\overline{x}_i$;
- pentru fiecare nod etichetat cu un $\overline{x}_i$ există o muchie ce leagă componenta cu opusul x_i.

Instanța F este transformată în $(G_F, k = n + m)$.

INDEPENDENT-SET: Exemplu

$$F = (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_3).$$



INDEPENDENT-SET: corectitudine

Trebuie să arătăm că F este satisfiabilă dacă și numai dacă există o mulțime independentă cu $n + m$ elemente.
U se construiește astfel: se alege câte un vârf din fiecare componentă C_j și câte unul dintre vârfurile-treaptă x_i sau $\overline{x}_i$ astfel încât:

- dacă din C_j se alege un vârf etichetat cu x_i, atunci se va alege și vârful-treaptă x_i;
- dacă din C_j se alege un vârf etichetat cu $\overline{x}_i$, atunci se va alege și vârful-treaptă $\overline{x}_i$;

Atribuirea a care satisface F va fi dată de $a(x_i) = 1$ dacă $x_i \in U$, $a(x_i) = 0$ dacă $\overline{x}_i \in U$.

Teoremă

INDEPENDENT-SET este $\mathbb{NP}$-completă.

VERTEX-COVER

Domeniul problemei:
Fie $G = (V, E)$ un graf. O submulțime $W \subseteq V$ se numește V-acoperire dacă oricare muchie din E este incidentă într-un vârf din W.

VERTEX-COVER
Instance Un graf $G = (V, E)$, un număr întreg $k \geq 0$.
Question Există o submulțime $W \subseteq V$ care este V-acoperire și are cel mult k elemente? ($|W| \leq k$).

VERTEX-COVER

Lemă

INDEPENDENT-SET $\propto$ *VERTEX-COVER.*

O instanță (G, k) este transformată într-o instanță $(G, n - k)$.

Teoremă

VERTEX-COVER este $\mathbb{NP}$*-completă.*

Plan

Mulțimi

SUBSET-SUM

Instance O mulțime A, o mărime $s(a) \in \mathbb{Z}_+$, pentru orice $a \in A$ și $K \in \mathbb{Z}_+$.

Question Există o submulțime $A' \subseteq A$ pentru care suma mărimilor elementelor din A' să fie exact K?

SET-COVER

Instance O coelcție de m mulțimi $A_1, \ldots, A_m$ și $k \in \mathbb{Z}_+$.

Question Există o subcolecție de k mulțimi $A_{i_1}, \ldots, A_{i_k}$ astfel încât

$$A_1 \cup \cdots \cup A_m = A_{i_1} \cup \cdots \cup A_{i_k}?$$

Planificare

MULTIPROCESSOR SCHEDULING

Instance O mulțime P de programe, un număr m de procesoare, un timp de execuție $t(p)$, pentru fiecare $p \in P$, și un termen D.

Question Există o planificare a procesoarelor pentru P astfel încât orice program să fie executat înainte de termenul D?

Algebră

QUADRATIC RESIDUU

Instance Întregii pozitivi a, b, c.

Question Există un întreg pozitiv $x < c$ astfel încât $x^2 \equiv a \pmod{b}$?

QUADRATIC DIOPHANTIC EQUATIONS

Instance Întregii pozitivi a, b, c.

Question Există $x, y \in \mathbb{Z}_+$ astfel încât $ax^2 + by = c$?

Programare matematică

INTEGER PROGRAMMING

Instance O mulțime X de perechi (x, b), unde $x = (x_1, \ldots, x_m), x_i, b \in \mathbb{Z}$, o secvență $c = (c_1, \ldots, c_m)$ și un întreg K.

Question Există $y = (y_1, \ldots, y_m)$ astfel încât:

1. $\sum_i x_i y_i \leq b$ pentru orice $(x, b) \in X$;
2. $\sum_i c_i y_i \geq K$?

KNAPSACK 0/1

Instance $w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, n$
$M \in \mathbb{Z}_+$
$K \in \mathbb{Z}_+$ (scop)

Question Există $x_i \in \{0, 1\}, i = 1, \ldots, n$ cu proprietatea că scopul este atins și obiectele alese încap în rucsac:

$$\sum_{i=1}^{n} x_i \cdot p_i \geq K$$

$$\sum_{i=1}^{n} x_i \cdot w_i \leq M?$$

Grafuri

K-COLORING

Instance Un graf $G = (V, E)$ și $k \in \mathbb{Z}_+$.

Question Există o colorare cu k culori a grafului G?

HAMILTONIAN CIRCUIT DIGRAPH

Instance Un digraf $D = (V, A)$.

Question Conține D un circuit hamiltonian (circuit care trece prin toate vârfurile din V)?

Grafuri

HAMILTONIAN CIRCUIT GRAPH

Instance Un graf $G = (V, E)$.

Question Conține G un circuit hamiltonian?

TRAVELING SALESMAN PROBLEM

Instance Un graf ponderat $((V, E), c)$ cu $c(i, j) \in \mathbb{Z}_+$ pentru orice muchie $\{i, j\} \in E$ și $K \in \mathbb{Z}_+$.

Question Există un circuit hamiltonian de cost $\leq K$?

Grafuri

LONGEST PATH

Instance Un graf ponderat $G = (V, E)$ cu ponderele $\ell : E \to \mathbb{Z}_+$, un întreg pozitiv K și două vârfuri distincte i și j.

Question Există în G un drum de la i la j de lungime $\geq K$?