Probleme $\mathcal{NP}$-complete

- Algoritmi nedeterministi (reamintire)
- Clasele $\mathcal{P}$ si $\mathcal{NP}$
- Probleme $\mathcal{NP}$-dificile si $\mathcal{NP}$-complete
- Exemple de probleme $\mathcal{NP}$-complete

Algoritmi nedeterministi (reamintire)

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 $\{0, 1, \ldots, n-1\}$.

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

Această definiție a algoritmilor nedeterminiști este strâns legată de rezolvarea problemelor de decizie. Reamintim că, în general, orice problemă poate fi redusă la rezolvarea unei probleme de decizie.

Problema rezolvata de un algoritm nedeterminist

Spunem ca un algoritm nedeterminst A rezolva o problema P daca:

- ⇨ pentru orice instanta p a lui P, exista o conguratie $<A; \sigma_p>$ astfel incat σ_p include structuri date ce descrie p;
- ⇨ **exista** o executia lui A din conguratia initiala $<A; \sigma_p>$ care se termina intr-o conguratie $< . ; \sigma'>$; si σ' include structuri de date ce descriu P(p).

Probleme de decizie (reamintire)

- formalizare
  - instanta: A, $B \subseteq A$, $x \in A$
  - intrebare: $x \in B$?
- exemplu: problema rucsacului
  - instanta
    - o multime de obiecte O,
    - fiecare obiect are o marime $w(o) \in Z_+$ si o valoare $p(o) \in \mathbf{Z}_+$
    - restrictie: $M \in \mathbf{Z}_+$
    - scop $K \in \mathbf{Z}_+$
  - intrebare
    - exista $O' \subseteq O$ a.i. $\sum_{o \in O'} w(o) \leq M$ si $\sum_{o \in O'} p(o) \geq K$?

Algoritm nedeterminist pentru rucsac 0/1

```
procedure rucsacND(O, s, v, M, K, x)
begin
   /* ghiceste */
   for each o ∈ O do
      x[o] ← random(2)
   /* verifica */
   wGhicit ← vGhicit ← 0
   for each o ∈ O do
      wGhicit ← wGhicit + x[o]*w[o]
      pGhicit ← pGhicit + x[o]*p[o]
   if (wGhicit ≤ M and pGhicit ≥ K)
   then success
   else failure
end
```

Clasele $\mathcal{P}$ si $\mathcal{NP}$

- $\mathcal{P}$ = clasa problemelor care pot fi rezolvate de algoritmi deterministi in timp polinomial
- $\mathcal{NP}$ = clasa problemelor care pot fi rezolvate de algoritmi NEdeterministi in timp polinomial
- E usor de vazut ca $\mathcal{P} \subseteq \mathcal{NP}$
- Intrebarea e se pune daca $\mathcal{P} \subset \mathcal{NP}$ sau $\mathcal{P} = \mathcal{NP}$?
- Este una dintre cele mai celebre probleme nerezolvate (rezolvarea ei este premiata cu 1 milion de dolari)
- Deocamdata putem demonstra doar

Teorema

Daca P este in $\mathcal{NP}$, atunci exista polinoamele p(n) si q(n) si un algoritm determinist care rezolva P in timpul $O(p(n)2^{q(n)})$.

- In ipoteza ca $\mathcal{P} \subset \mathcal{NP}$, care sunt problemele ce candideaza a fi in $\mathcal{P}$ si nu in $\mathcal{NP}$?

Probleme $\mathcal{NP}$-complete

- P este problema $\mathcal{NP}$**-dificila** daca pentru orice problema Q din $\mathcal{NP}$ are loc $Q \propto P$.

- P este problema $\mathcal{NP}$**-completa** daca:
  - P este in $\mathcal{NP}$ si
  - P este $\mathcal{NP}$-dificila

SAT - enunt

- Problema satisfiabilitatii (SAT)
  - ⇨ instanta: o formula F din calculul propozitional in forma normala conjuctiva si in care apar variabile din $\{x_0, ..., x_{n-1}\}$
  - ⇨ intrebare: exista o atribuire a variabilelor pentru care F este satisfacuta?

Teorema (Steven Cook, 1971)

SAT este $\mathcal{NP}$-completa.

SAT - demonstratie

- algoritm nedeterminist care rezolva SAT:
  1. ghiceste o atribuire pentru variabile
  2. calculeaza valoarea formulei
  3. daca formula este satisfacuta intoarce **`success`**; altfel intoarce **`failure`**.

SAT - demonstratie (cont.)

- $(\forall P \text{ in } \mathcal{NP})\ P \propto SAT$
  - ⇨ fie A care rezolva P
  - ⇨ Se considera variabilele:
    - B_{ijt} = valoarea bitului j din locatia i la momentul t
    - $S_{kt} \Leftrightarrow$ instructiunea de eticheta k se executa la momentul t
  - ⇨ asociem lui A pentru intrarea x formula

    $F(A, x) = F1 \wedge F2 \wedge F3 \wedge F4 \wedge F5 \wedge F6$ unde
    - F1 - starea initiala
    - F2 - prima instructiune care se executa
    - F3 - dupa t pasi se executa exact o instructiune
    - F4 - calculul instructiunii urmatoare
    - F5 - schimbarea memoriei
    - F6 - terminarea cu succes

SAT - demonstratie (cont.)

```
1:  if (x > 2)
2:  then y ← x*x
3:  else y ← x*x*x
4:success
```

- locatia 0 memoreaza 2, locatia 1 memoreaza x, locatia 2 memoreaza y
- starea initiala pentru x = 3

  $F_1 = B_{0,0,0} \wedge \neg B_{0,1,0} \wedge B_{1,0,0} \wedge B_{1,1,0}$
- prima instructiune care se executa

  $F_2 = S_{1,0} \wedge \neg S_{2,0} \wedge \neg S_{3,0} \wedge \neg S_{4,0}$
- dupa t pasi se executa exact o instructiune

  $F_3 = G_0 \wedge G_1 \wedge G_2$

  $G_t = G_{1,t} \oplus G_{2,t} \oplus G_{3,t} \oplus G_{4,t}$

  ...

  $G_{2,t} = \neg S_{1,t} \wedge S_{2,t} \wedge \neg S_{3,t} \wedge \neg S_{4,t}$

  ....
- etc

Problema $\mathcal{U}$

- Formulare
  - Instanta
    - un program A, o intrare x, un intreg $k > 0$
  - Intrebare
    - programul A cu intrarea x se termina cu raspunsul DA in $\leq k$ pasi?
- $\mathcal{U}$ este in $\mathcal{NP}$
  - construim un algoritm U care simuleaza k pasi ai lui A si se termina cu DA daca si numai daca A se termina cu DA in cei k pasi
- $\mathcal{U}$ este in $\mathcal{NP}$-dificila
  - daca Q este in $\mathcal{NP}$, atunci exista un alg. nedet. A care rezolva Q
  - transformam o instanta $q \in Q$ de dimensiune n intr-o instanta $(A, q, T_A(n))$

Exemple de probleme $\mathcal{NP}$-complete

- SAT
- 3SAT
- Rucsac 0/1
- Submultime de suma data

- V-acoperire (VA)
  - instanta: un graf $G = (V, E)$, $K \in Z_+$
  - intrebare: exista o V-acoperire V' a.i. $\#V' \leq K$?

- Circuit Hamiltonian intr-un digraf (CHD)
  - instanta: un digraf $D = (V, A)$
  - intrebare: exista un circuit Hamiltonian?

- Circuit Hamiltonian intr-un graf (CHG)

Exemple de probleme $\mathcal{NP}$-complete (continuare I)

- Comis voiajor (CV)
  - instanta: un graf ponderat $G = (V, E, c)$, $c(\{i,j\}) \in Z_+$, $K \in Z_+$
  - intrebare: exista un circuit Hamiltonian de cost $\leq K$?

- Planificare procesoare (PP)
  - instanta: o multime P de programe, m procesoare, un timp de executie t(p) pentru fiecare program p, un termen D
  - intrebare: exista o planificare a procesoarelor pentru P a.i. orice program sa fie executat in termenul D?

Exemple de probleme $\mathcal{NP}$-complete (continuare II)

- Congruente patratice (CP)
  - ⇨ instanta: $a, b, c \in Z_+$
  - ⇨ intrebare: exista $0 < x < c$ a.i. $x^2 \ \mathbf{mod}\ b = a$?

- Ecuatii diofantice patratice
  - ⇨ instanta: $a, b, c \in Z_+$
  - ⇨ intrebare: exista $x, y \in Z_+$ a.i. $ax^2 + by = c$?

Cum se arata $\mathcal{NP}$-completitudinea

- reducere

  daca P este in $\mathcal{NP}$, Q este $\mathcal{NP}$-completa si $Q \propto P$

  atunci P este $\mathcal{NP}$-completa

  ⇨ exemplu: SAT $\propto$ 3SAT

  - $c = u_1$

    $c' = (u_1 \vee y_1 \vee y_2) \wedge (u_1 \vee y_1' \vee y_2) \wedge (u_1 \vee y_1 \vee y_2') \wedge (u_1 \vee y_1' \vee y_2')$
  - $c = u_1 \vee u_2$

    $c' = (u_1 \vee u_2 \vee y_1) \wedge (u_1 \vee u_2 \vee y_1')$
  - $c = u_1 \vee u_2 \vee u_3 \vee u_4$

    $c' = (u_1 \vee u_2 \vee y_1) \wedge (u_3 \vee u_4 \vee y_1')$

  ⇨ 3SAT $\propto$ VA

  ⇨ VA $\propto$ CHG

Cum se arata $\mathcal{NP}$-completitudinea (cont.)

- restrictia

  daca Q este $\mathcal{NP}$-completa si Q este caz special al lui P

  atunci P este $\mathcal{NP}$-completa

  - exemplu: CHG caz special al lui CHD

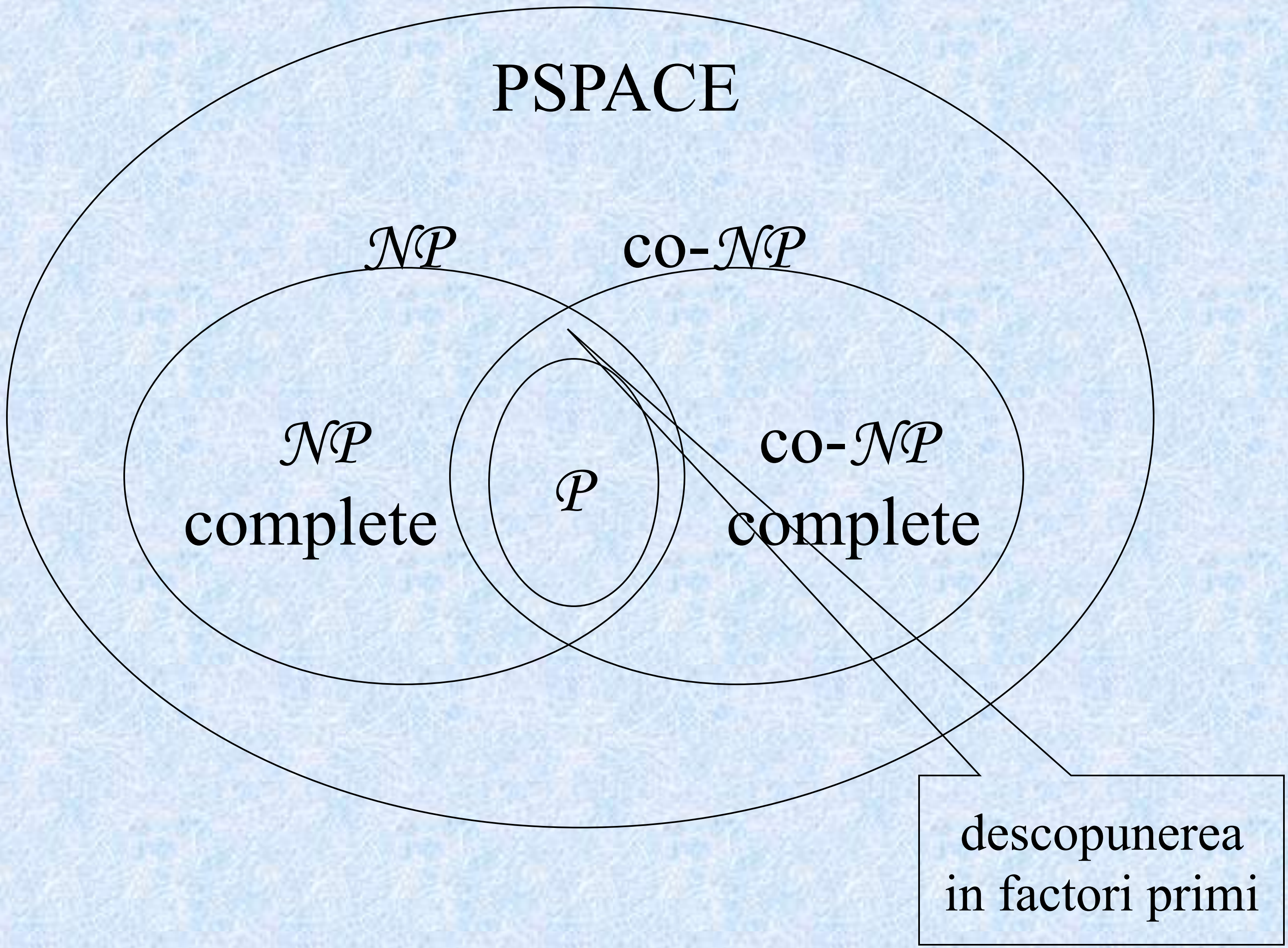
Alte clase

- PSPACE = clasa problemelor care sunt rezolvate de algoritmi deterministi in timp nelimitat si utilizand spatiu polinomial
- NPSPACE = clasa problemelor care sunt rezolvate de algoritmi NEdeterministi in timp nelimitat si utilizand spatiu polinomial

- PSPACE = NPSPACE (Savitch, 1970)

- problema co-P:
  - aceleasi instante ca P dar raspunde DA daca P raspunde NU si raspunde NU daca P raspunde DA
- co-$\mathcal{NP}$ = clasa problemelor co-P cu P in $\mathcal{NP}$

Alte clase


PSPACE
NP
co-NP
NP complete
P
co-NP complete
descopunerea in factori primi

Problema TQBF

- P(x) o formula booleana care depinde de variabila booleana x
- Cuantificator universal :∀ $\forall x P(x)$ inseamna "pentru orice $x \in \{0,1\}$, P(x) este adevarat"
- Cuantificator existential :∀ $\exists x P(x)$ inseamna "exista $x \in \{0,1\}$, P(x) este adevarat"
- *Formula booleana quantificata complet (fully quantified Boolean formula)* = o formula prefixata cu cuantificator un $(\forall x)$ sau $(\exists x)$ pentru fiecare variabila x
- Exemple:

  $\forall x (x \vee \underline{x})$

  $\forall x \forall y\ (x \vee y)$

  $\forall x \exists y\ ((x \wedge y) \vee (\underline{x} \wedge \underline{y}))$

  $\exists z \forall x \exists y\ ((x \wedge y \wedge z) \vee (\underline{x} \wedge \underline{y} \wedge z))$
- o formula booleana quantificata complet este adevarata sau falsa

Problema TQBF

- TQBF(**True** **Quantified** **Boolean** **Formulas**)
  - instanta: o formula booleana quantificata complet F
  - intrebare: este F adevarata?

- TQBF este PSPACE completa
  - TQBF este in PSPACE

    $F \equiv Q_1 Q_2 \ \dots \ Q_n \, \Phi(x_1, x_2, \dots, x_n)$

    construim un algoritm recursiv val(F) astfel:

    daca F nu are cuantificatori (si deci nici variabile), intoarce valoarea lui F

    altfel

    $A = val(Q_2 \ \dots \ Q_n \, \Phi(0, x_2, \dots, x_n))$

    $B = val(Q_2 \ \dots \ Q_n \, \Phi(1, x_2, \dots, x_n))$

    daca $Q_1 \equiv (\exists x_1)$, atunci intoarce $A \vee B$

    daca $Q_1 \equiv (\forall x_1)$, atunci intoarce $A \wedge B$

Problema TQBF

- spatiul ocupat de val(): O(n + log n)
- timpul: $O(2^n)$

⇨ $(\forall P$ in PSPACE $)\ P \propto$ TQBF

$\Phi_{c1,\ c2,\ t}$ = *true* daca si numai daca din configuratia c1 se poate ajunge in configuratia c2 in cel mult t pasi

pentru un P in PSPACE data:

c1 = starea initiala

c2 = starea finala

t = timpul dat de algritmul care rezolva P (exponential, marginit de un T)

Problema TQBF

- cum poate fi calculata $\Phi_{c1,\, c2,\, t}$ in timp polinomial?

  t = 1, trivial

  $t > 1 : \Phi_{c1,\, c2,\, t} = (\exists m)\ \Phi_{c1,\, m,\, \lceil t/2 \rceil} \wedge \Phi_{m,\, c2,\, \lceil t/2 \rceil}$

  din pacate timpul este exponential pentru formula de mai sus

  $\Phi_{c1,\, c2,\, t} = (\exists m)(\forall (c3,c4) \in \{(c1,m),(m,c2)\})\Phi_{c3,\, c4,\, \lceil t/2 \rceil}$

  $= (\exists m)\ p(m, c1, c2, c3, c4) \Rightarrow \Phi_{c3,\, c4,\, \lceil t/2 \rceil}$

  $= (\exists m)\ \underline{p(m, c1, c2, c3, c4)} \vee \Phi_{c3,\, c4,\, \lceil t/2 \rceil}$

Bibliografie suplimentara

- An Annotated List of Selected NP-complete Problems

http://www.csc.liv.ac.uk/~ped/teachadmin/COMP202/annotated_np.html