# Rezolvarea problemelor prin cautare

Curs 5

**2010**

# Cautare

- Gasirea unei solutii pentru o problema specificata declarativ in spatiul tuturor solutiilor
- Cautarea = enumerarea sistematica a solutiilor potentiale si/sau partiale ale unei probleme astfel incat fiecare este verificata daca este sau nu o solutie finala. Cautarea presupune:
  - O definitie a solutiilor potentiale
  - O metoda “isteata” de generare sistematica a solutiilor potentiale
  - O metoda de verificare daca o solutie potentiala este o solutie finala.

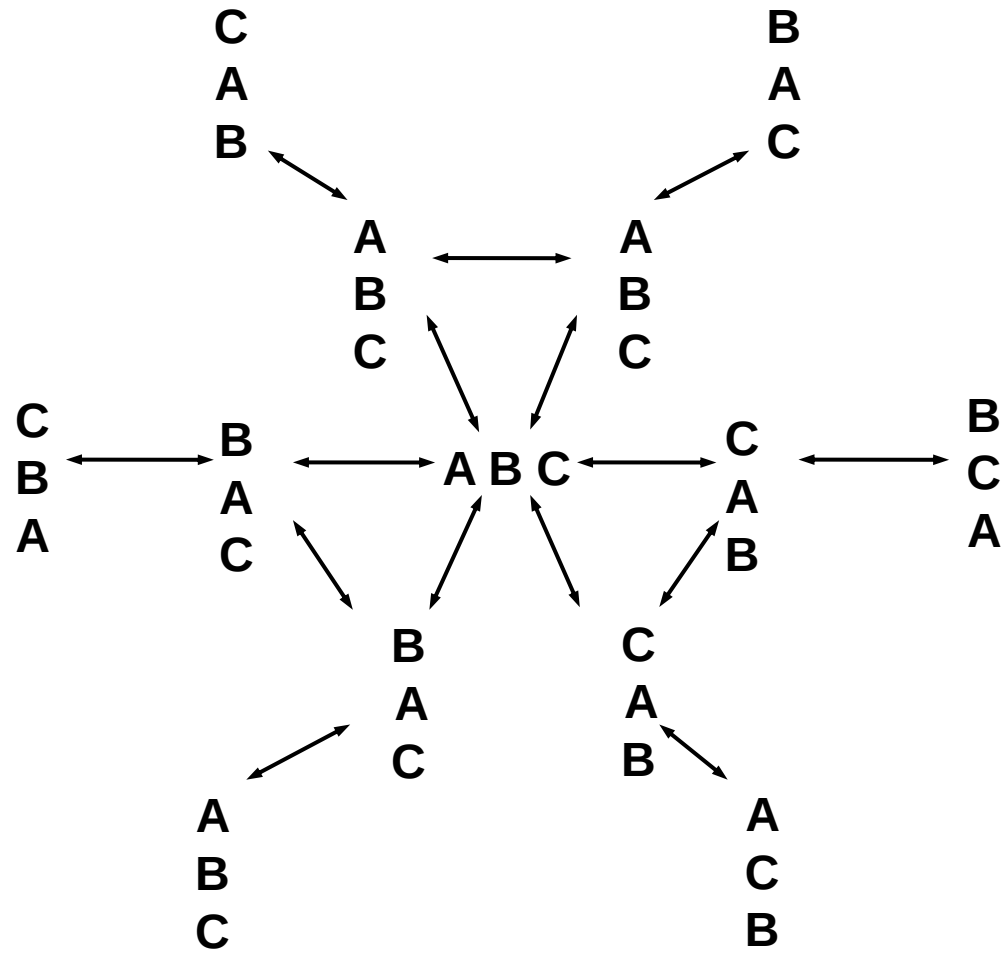
# De ce e utila cautarea?

- Cautarea este necesara in general pentru rezolvarea problemelor NP-complete – acele probleme pentru care testarea conditiilor pe care trebuie sa le indeplineasca o solutie se poate face eficient, dar nu exista metode eficiente pentru gasirea directa a acestor solutii.
- Cautarea este utila pentru implementarea nedeterminismului nu-se-stie din algoritmii din IA.

# Cautare in grafuri

- Rezolvarea problemelor prin cautare poate fi abstractizata ca procesul de determinare a cailor intr-un graf orientat.
- Fiind date o multime de noduri de start si o multime de noduri obiectiv (engl.goal nodes) se numeste solutie o cale de la un nod start la unul dintre nodurile obiectiv.
- Uneori fiecarui arc i se asociaza un cost, numar real pozitiv. Costul unei cai va fi suma costurilor arcelor sale.

# Spatiul configuratiior in CubeWorld



# Gasirea unei derivari SLD

$a \leftarrow b \wedge c.$ $a \leftarrow g.$

$a \leftarrow h.$ $b \leftarrow j.$

$b \leftarrow k.$ $d \leftarrow m.$

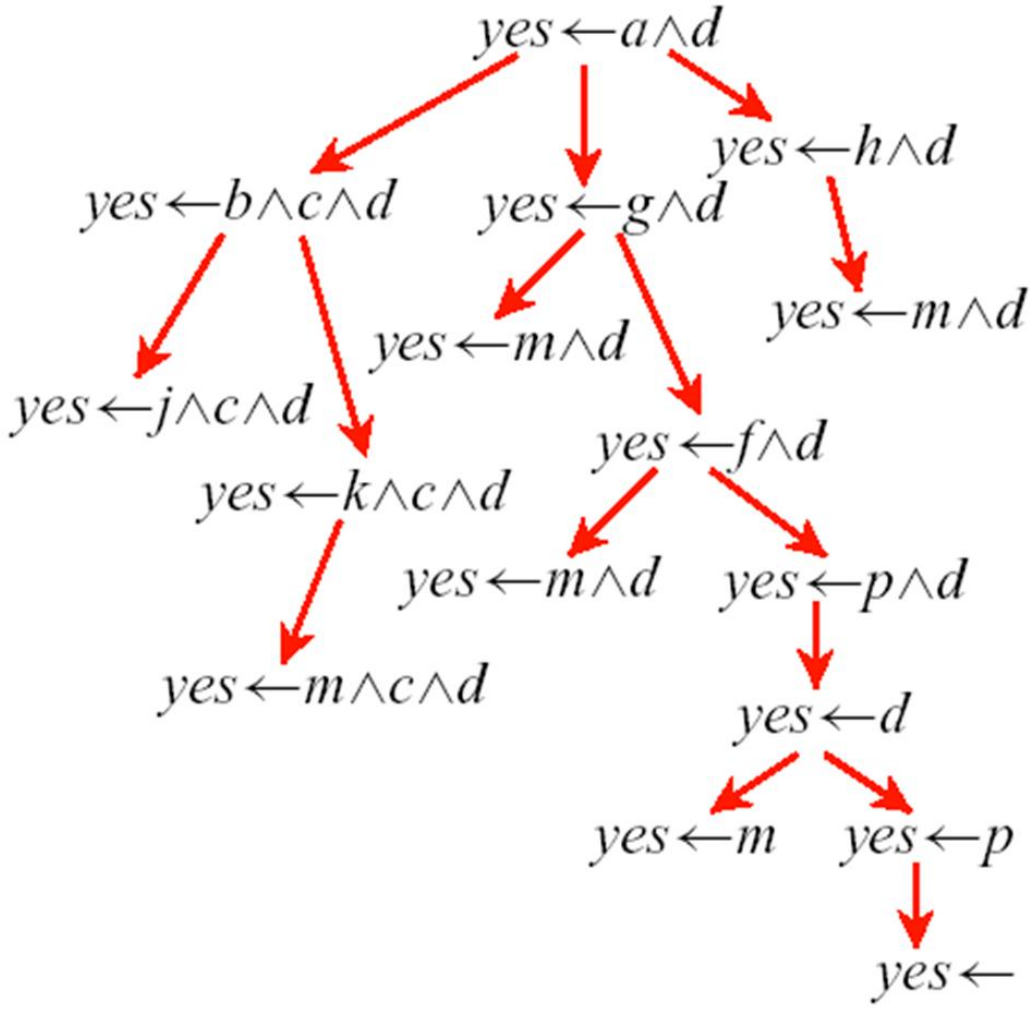
$d \leftarrow p.$ $f \leftarrow m.$

$f \leftarrow p.$ $g \leftarrow m.$

$g \leftarrow f.$ $k \leftarrow m.$

$h \leftarrow m.$ $p.$

$?a \wedge d$



# Graf de cautare

- Poate reprezenta un graf de stari. In acest caz un nod reprezinta o stare a lumii si un arc reprezinta trecerea de la o stare la o alta stare. Un exemplu este graful configuratiilor multimii celor 3 cuburi.
- Poate reprezenta un graf de subprobleme. In acest caz un nod reprezinta o subproblema care trebuie rezolvata iar arcele reprezinta diverse modalitati de descompunere a unei probleme. Un exemplu este gasirea unei derivari prin rezolutie SLD.

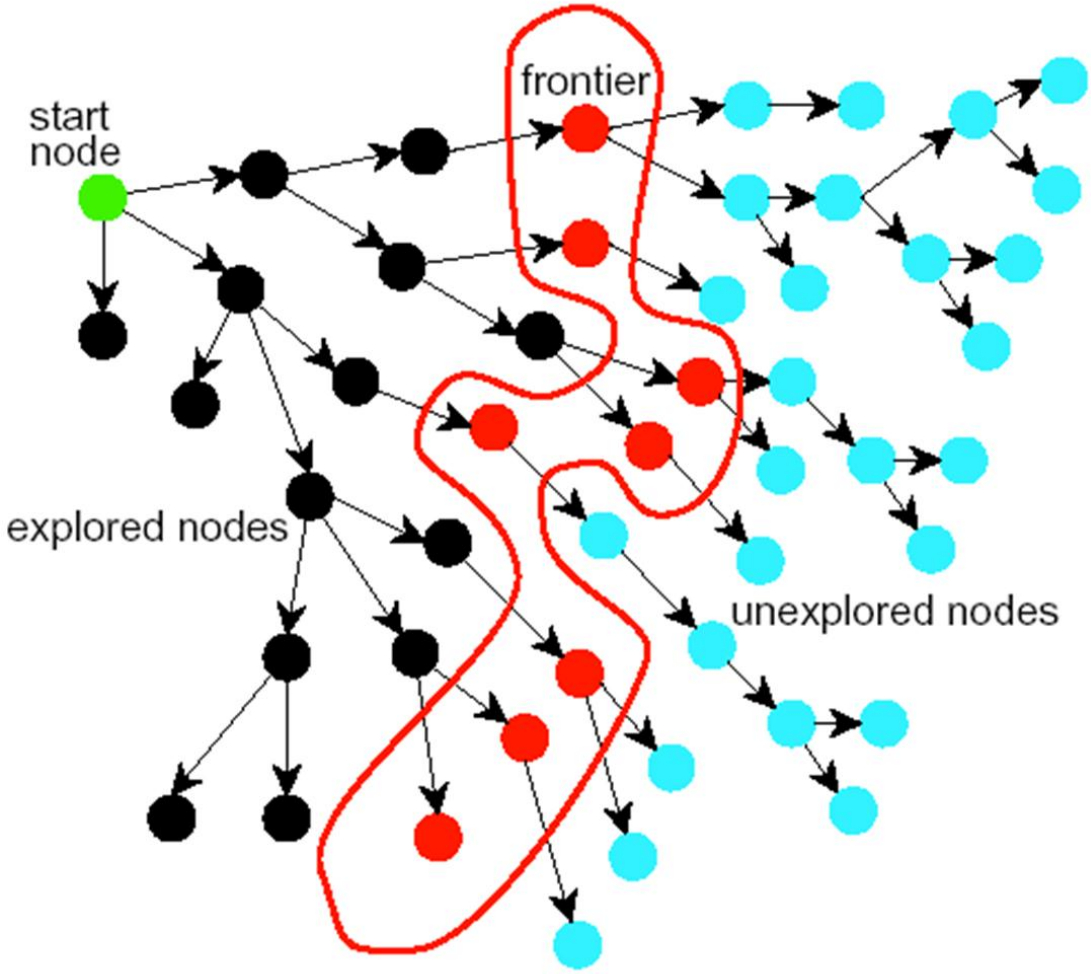
# Procesul de cautare intr-un graf

- Se pleaca de la un graf, un nod de start si o multime de noduri obiectiv. Se exploreaza incremental caile ce pleaca de la nodurile de start pana la gasirea nodurilor obiectiv.
- Cautarea gestioneaza o multime frontiera (engl.fringe sau frontier) a tuturor cailor explorate ce pleaca de la nodul de start.
- De-alungul cautarii, frontiera se extinde cu noi noduri neexplorate inca, pana se ajunge la un nod obiectiv.

# Procesul de cautare intr-un graf (cont'd)

- Modul in care are loc expandarea frontierei defineste strategia de cautare. Acest lucru inseamna modul in care un nod din frontiera este ales pentru a fi expandat.
- O strategie de cautare va fi cu atat mai buna cu cat va expanda frontiera astfel incat sa se ajunga cat mai repede la un nod obiectiv.

# Procesul de cautare intr-un graf (cont'd)



# Un algoritm generic de cautare

***cauta***(*F0*) ←
    *selecteaza*(*Nod*,*F0*,*F1*) ∧
    *este_obiectiv*(*Nod*)
***cauta***(*F0*) ←
    *selecteaza*(*Nod*,*F0*,*F1*) ∧
    *vecini*(*Nod*,*NN*) ∧
    *ad_la_frontiera*(*NN*,*F1*,*F2*) ∧
    *cauta*(*F2*).

***cauta***(*F*) ← exista o cale de la un nod din frontiera *F* la un nod obiectiv

***este_obiectiv***(*N*) ← *N* este un nod obiectiv

***vecini***(*N*,*NN*) ← *NN* este lista vecinilor nodului *N*

***selecteaza***(*N*,*F0*,*F1*) ← *N* ∈ *F0* ∧ *F1* = *F0* \ { *N* }

***ad_la_frontiera***(*NN*,*F1*,*F2*) ← *F2* = *F1* ∪ *NN*

# Costuri si cai

- Costuri: uneori intereseaza si costul unei solutii. Atunci se indica costul fiecarui arc: `cost(Nod1, Nod2, Cost)`
- Cai:
  - In algoritmul generic de cautare caile sunt gasite doar implicit, nu si explicit.
  - Pentru determinarea explicita a cailor, fiecare nod din frontiera va fi un termen cu structura: `nod(Nod,Cale,CostCale)` unde `Cale` este lista nodurilor de la nodul de start la `Nod` (fara `Nod`) si `CostCale` este costul acestei cai. Extinderea unei cai cu un nod presupune adaugarea unui element in capul acestei liste.

# Determinarea cailor

```
cauta(F0,[N | P]) ←
    selecteaza(nod(N,P,PC),F0,F1) ∧
    este_obiectiv(N)
cauta(F0,Cale) ←
    selecteaza(nod(N,P,PC),F0,F1) ∧
    vecini(N,NN) ∧
    ad_cai(NN, nod(N,P,PC),NN1) ∧
    ad_la_frontiera(NN1,F1,F2) ∧
    cauta(F2,Cale).
```

```
ad_cai(NN, nod(N,P,PC),NN1) ←
    NN1 este lista noilor
    elemente de pe frontiera ce
    vor inlocui elementul
    nod(N,P,PC)
ad_cai([],N,[]) ←
ad_cai([M|R],nod(N,P,PC),
    [nod(M,[N|P],NPC)|FR]) ←
    cost(N,M,C) ∧
    NPC is PC+C ∧
    ad_cai(R,nod(N,P,PC),FR)
```

# Strategii de cautare

- O strategie de cautare specifica modul in care se selecteaza si se adauga noduri la multimea frontiera (`selecteaza` si `ad_la_frontiera`).
- Strategiile de cautare se clasifica in:
  - Strategii de cautare *neinformate*, care nu iau in considerare cat de "departat" este un nod obiectiv de nodul curent.
  - Strategii de cautare *informate* sau euristice care selecteaza pentru expandare nodurile cele mai promitatoare din multimea frontiera.

# Cautarea in adancime

- Cautarea in adancime se implementeaza tratand multimea frontiera ca o stiva: intotdeuna va fi selectat pentru expandare ultimul element adaugat la frontiera.
- Daca frontiera este `[e1, e2, …]` se va selecta pentru expandare `e1`. El va fi inlocuit in multimea frontiera cu vecinii sai. `e2` va fi selectat numai dupa ce toate caile din `e1` vor fi explorate.

# Cautarea pe nivel

- Cautarea pe nivel se implementeaza tratand multimea frontiera ca o coada: intotdeuna va fi selectat pentru expandare elementul adaugat cel mai tarziu la frontiera.
- Daca frontiera este `[e1, e2, …]` se va selecta pentru expandare `e1`. El va fi inlocuit in multimea frontiera cu vecinii sai, acestia fiind adaugati la sfarsitul aceteia. `e2` va fi imediat selectat pentru explorare dupa `e1`.

# Cautarea de cost minim

- In cautarea cu cost minim, caile de la nodul de start sunt generate in ordinea crescatoare a costului lor. Intotdeauna se va selecta pentru expandare calea de cost minim.
- Costul unei cai de la nodul de start la nodul curent n este egal cu suma costurilor arcelor sale si se nodeaza cu $g(n)$.
- Cautarea cu cost minim se implementeaza tratand multimea frontiera ca o coada cu prioritati: intotdeuna va fi selectat pentru expandare elementul de cost minim din frontiera.