13.04.2023

IA
curs 8

KR (Partea a doua)

flarisha @ fmi.unibuc.ro

- PROBLEMĂ: stări + acțiuni
- DRUM în arbore de căut → { nodurile = stări; muchii = (mutări) de la o st. la alta }

↓

toate comb. de mutări posibile

[caută] în mulț. drumurilor, stările care conduc la câștig

prin stări succesive

1. repr. tabla
2. căutăm printre succ. de config. ale tablei una câștigătoare

- Căutare oarbă: în adâncime DF, în lățime BF
  ↳ agentul care caută nu deține info. asupra sp. în care caută; el doar face test-scop.
- Căutare informată: agentul deține și info. despre config. spațiului de căutare.

euristica

ex: ARAD → BUCUREȘTI;

ARAD → SIBIU → Buc.
ARAD → O... → Buc.

- căut. oarbă: au șanse egale
- căut. informată: va ști dif. între cele 2 rute

ALGORITMUL A*: cea fol. la câștig. partidei de șah

= PROBLEMĂ =

STARE INIȚIALĂ → mult. acțiunilor (operator) → SP. DE STĂRI (la care putem ajunge)

drum

și face TESTUL SCOP

costul = suma costurilor muchiilor

REZOLVAREA PB:

→ st. inițială

1. apl. testul - scop
2. apl. operatorii acestei stări ↳ extinderea stării

→ st. fin

Eval. unei strat de căutare { - completitudine → se găs. sol. dacă e adm.
- cplx. timp
- cplx. spațiu
- optimalit.

Căutare best-first:

$\hat{f}$ fct. de evaluare / estimare pt. val. unui nod { VREM să fie
- cât mai apr. de val. reală
- să fie realistă, nu prea optimistă

- NOTAȚII

$\hat{f}(n)$ minim ⇒ n cel mai valoros (nod)

extind un nod ⇒ alg. top. fucc. sun.

best first ⇒ extind nodul cu val. cea mai mică

ex: formă liniară

$\hat{f} = w_1 p_1 + w_2 p_2 + \dots + w_n p_n$ { · p - caracteristici
· w - ponderile acestor caracteristici

le stab. programatorul

le determină cu învățare sau din literatură

$\hat{f} = \hat{g} + \hat{h}$ ↳ evaluarea euristică a nodului

↳ adâncimea în graf

Algoritmul Graph Search: T_r - arbore n_0 - nod rădăc.

2 liste: { OPEN - nodurile de unde alegem pt. extindere
CLOSE - nodurile extinse deja, nu le mai extindem

o sortăm crescător după $\hat{f}$

04.05.2025

Curs IA

cu al ...-lea profesor

1. jocuri adversariale
2. alg. minimax
3. α-β retezare

slide 70 pt. examen

1. Jocuri adversariale

- șah – determinist → știu piesele adversarului
  cărți – stohastic → nu știu cărțile adv.
- complet sau parțial observabile

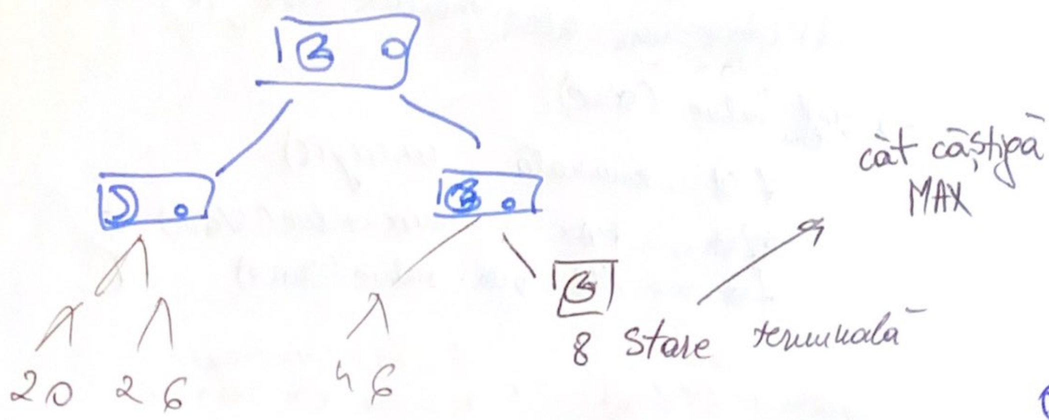
șah dame : det, compl. obs
poker : stoh, parț. obs
pacman : stoh, compl. obs.

Ne referim la jocuri {deterministe, doi jucători, complet obs., de sumă 0 (zero)} → ce câștigă x pierde y

MAX → maximiz. câșt.
MIN → — ~~

De obicei : MAX face prima mutare

Exemplu : PAC-MAN
1 mutare = -1p
mâncăm 1 pct = +10p



val. unei stări = cea mai bună utilitate ce poate
fi atinsă din ac. stare

- stare term: după regulile jocului
- → ~~netern~~ neterm: $\max_{s' \in children(s)} V(s')$

stări controlate de MIN (el vrea să îl facă pe MAX să piardă)

: $\min_{s' \in child.} V(s')$

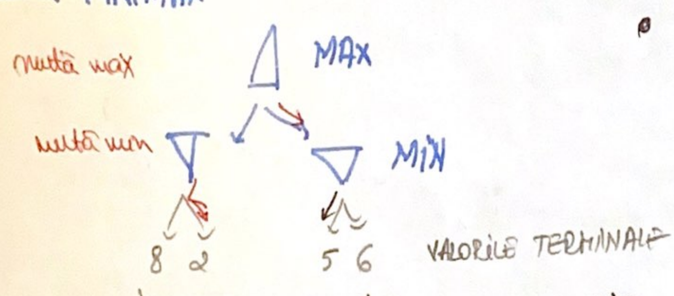
X și O:

nr. de ~~jocuri~~ stări: mare !

Δ MAX

∇ MIN

2. Algoritmul MINIMAX



- cel mai bun câștig GARANTAT este 5

MAX alege dr. pt că în stg, MIN l-ar duce ușor la 2

Pași: a) generăm arborele

b) obț. val. stării

c) mergem de jos în sus în arbore și propagăm val.

părinte: max → maximul fiilor

min → minimul fiilor

d) alege max acea mutare care îl duce la câștig

→ def value (stare):

if st == terminală: utility(s)

~~if~~ st == MAX : max-value(stare)

if st == MIN : min-value(stare)

cplx { timp : $O(b^m)$; b = nr. ramificații

spațiu : $O(b \cdot m)$; m = nr. de mutări

Limitări ale alg. MINIMAX:

→ greu de generat tot arborele

→ sol: caută până la o anumită limită în adâncime (aducem în joc o fct. euristică)

Ex: 100 sec.
100000 noduri/sec.

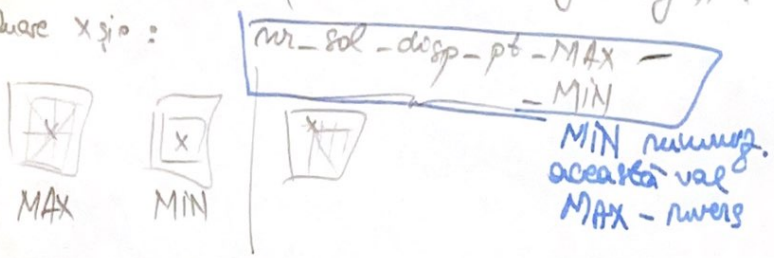
⇒ 100 × 1000 = 1 M

→ alg α-β reușește ajunge la adâncime 8 = performanță bună în șah

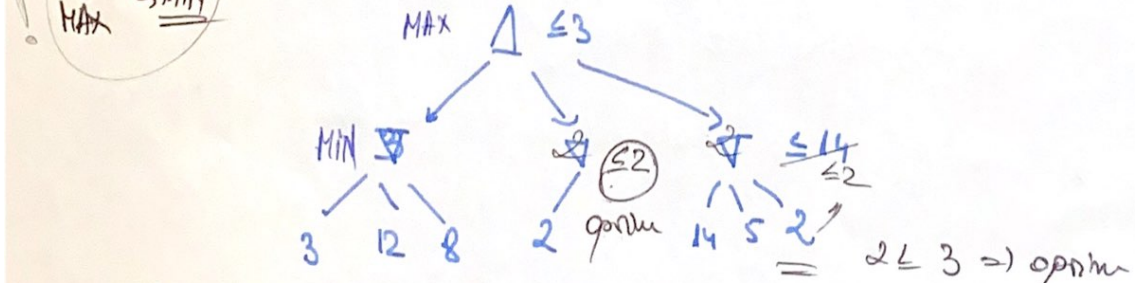
funcție de evaluare șah: $Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$

$f_1(s) = (\#\text{regine\_albe} - \#\text{regine\_negre})$, $w_1 = 100$

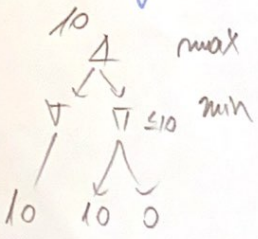
funcție de evaluare X și 0:



3. α-β retezare: tăiem subarborii care nu aduc inf. utilă



α-β doar ne ajută să accelerăm algoritmul.



→ NE PUTEM PĂCĂLI!

a) 1 2 3 4 — se aseaz

0			
x			
x	0		

=> x are nr. 4 mutări

$b = 4$ — — $b = 1$ spre sf. jocului

b) după 3 mutări

x	0		
0	x		0
x	x	0	x

0 0

x	0		
0	x	0	0
x	x	0	x

x	0	0	
0	x	0	
x	x	0	x

x 0 x

x → 0 0 0
0 x
0

c) ATENȚIE CINE MUTĂ PRIMUL!

d) subarborii se pot reordona a.î. să facem α-β tăieri