

Monte Carlo Tree Search

1. Tree Traversal

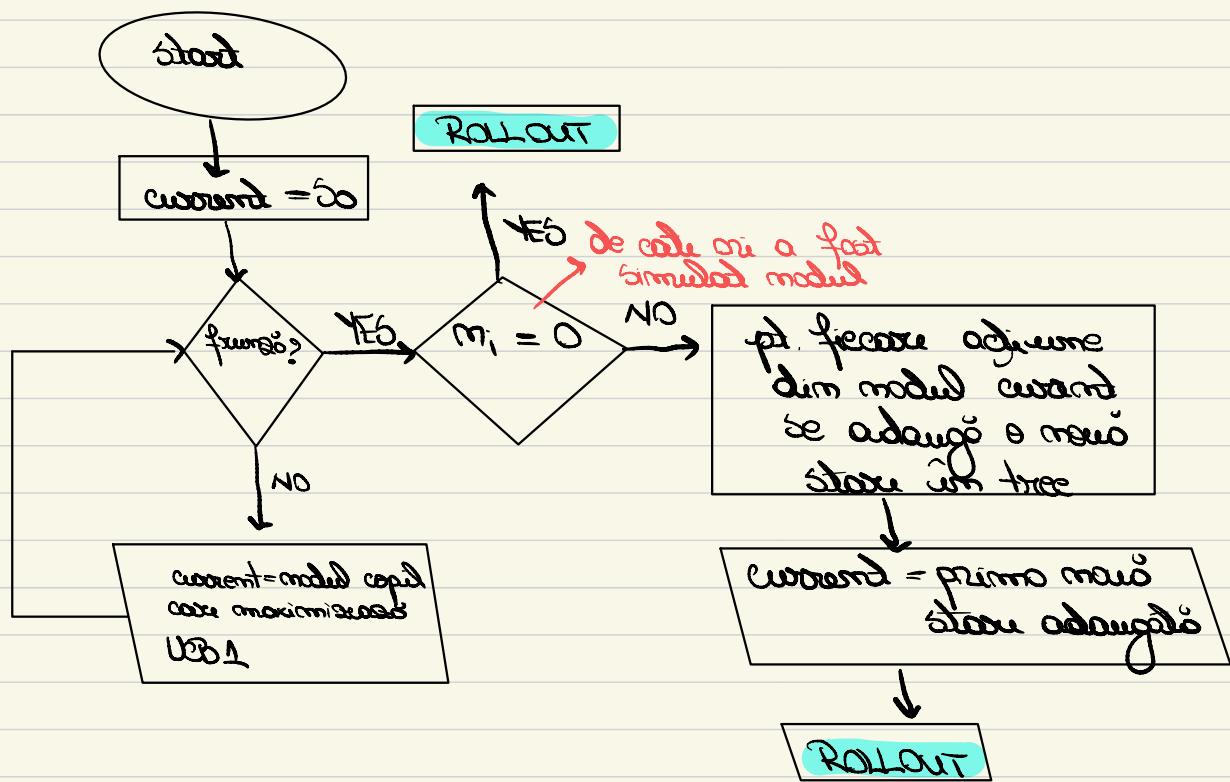
2. Node expansion

3. Rollout (Random Simulation)

4. Backpropagation

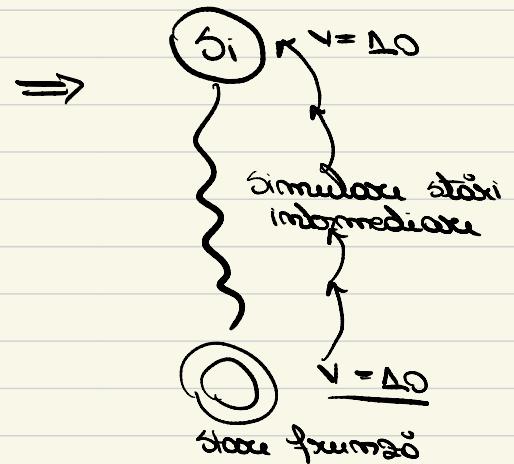
→ valoarea de la nodul curent este expandată în afară către nodurile parcurse în simulare

1 + 2:



3: Rollout(5):

loop forever:
if s_i stare terminată
return $v(s_i)$
 A_i = alegere-random(s_i)
 s_i = simulate(A_i, α)



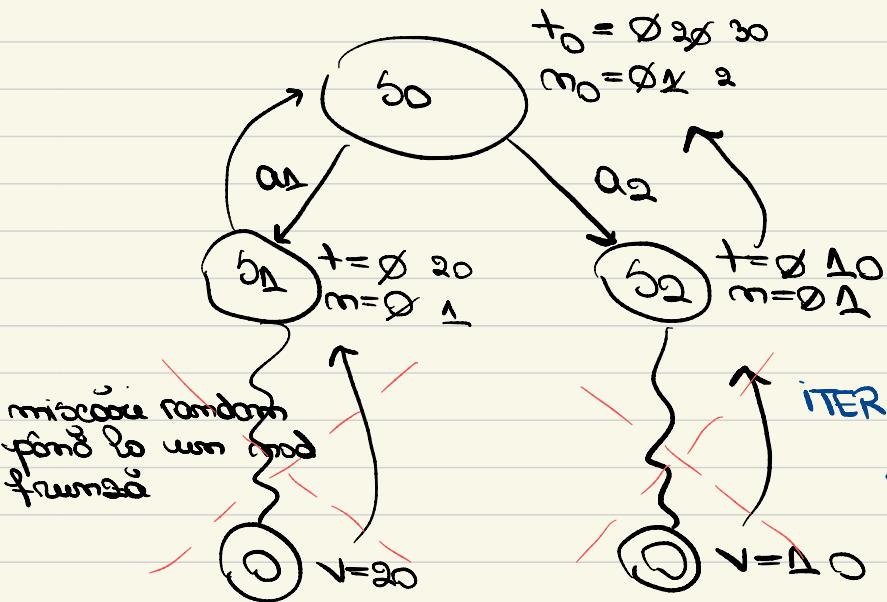
ex: UCB1 (Upper Confidence Bound)

$$= \bar{v}_i + 2 \sqrt{\frac{\ln N}{m_i}} \rightarrow$$

explorare

$\ln N \rightarrow$ nr. de vizitări ale perioadei atunci
 $m_i \rightarrow$ nr. simulării din mod

exposure



ITER. 1: • alegere dintre s_1 și s_2 în funcție de valoarea lor pt. UCB1

• ambele valori = $\infty \rightarrow$ se alege unul dintre noduri după un acordat considerat

• se alege din s_1

ITER. 2: • $UCB1(s_1) = 20 + 2 \cdot \sqrt{\frac{\ln 1}{1}} = 20$

• $UCB1(s_2) = \infty$

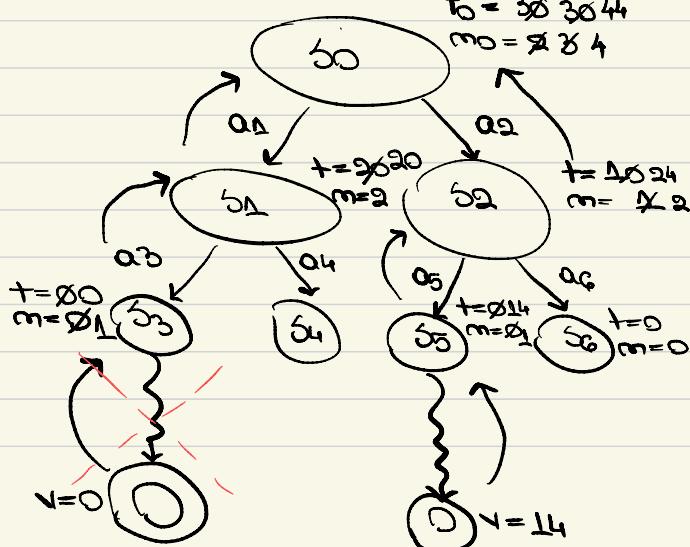
base alege s_2

(adăugăm aici \Rightarrow pasul backpropagation pînă la s_1 , $V=20$)

(după ce se alege este complet se face stocare / măsoară total
 \Rightarrow se actualizează complet)

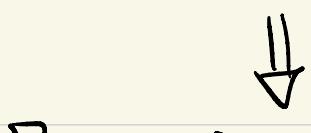
! Scop METO: reducere multiple pînă când alg converge la valoarea t_i corectă ale modurilor

! ITER. 3: \rightarrow se alege stocarea s_1 care maximizașă UCB1
 \rightarrow a fost corisat că pînă în 0 data \Rightarrow explorare stocare



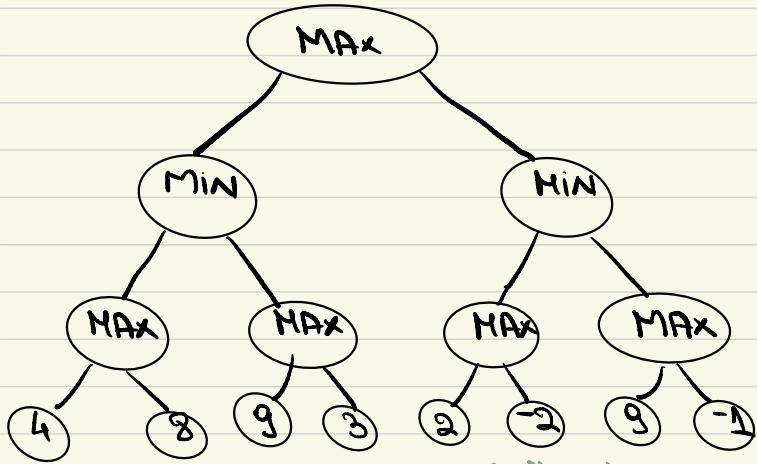
ITER. 4: $UCB1(s_1) = \frac{20}{2} + 2 \cdot \sqrt{\frac{\ln 3}{2}} = 11.48$

$UCB1(s_2) = \frac{10}{2} + 2 \cdot \sqrt{\frac{\ln 3}{1}} = 12.10$



! Daos alg. s-ori opre dupo 4 iter. \Rightarrow S-ori alege a_2 dim \bar{w} , desearce \bar{z}_2 cu o val. + mai mare decât \bar{z}_1

Mimax with α - β pruning



- MAX: maximizare costig
- MIN: minimizare costig
- parcurgere SRD (inonder)

Max-Value (s, α, β):

ces mai bună alternativă
a unei MAX următoare
din stocul \bar{w}

Min-Value (s, α, β):

ces mai bună
alternativă
pt. MIN

if s mod terminal:

return $V(s)$

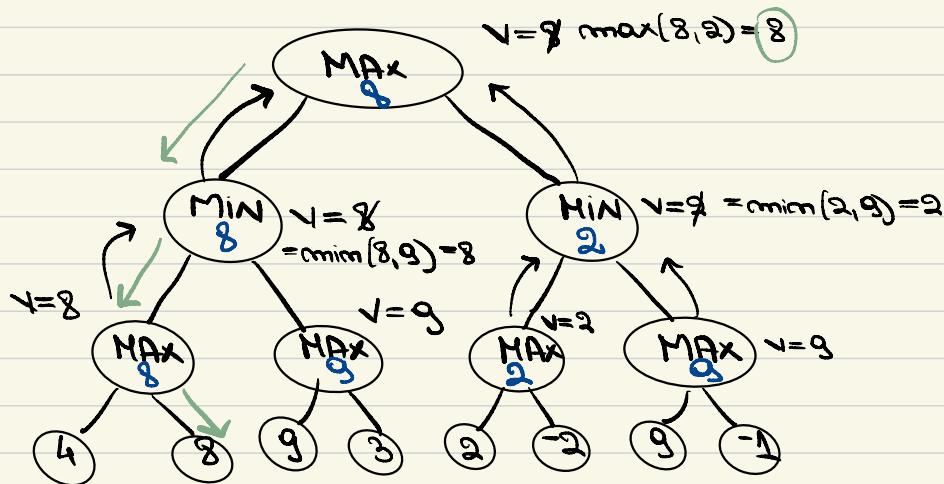
$v = -\infty$

for c in next(s):

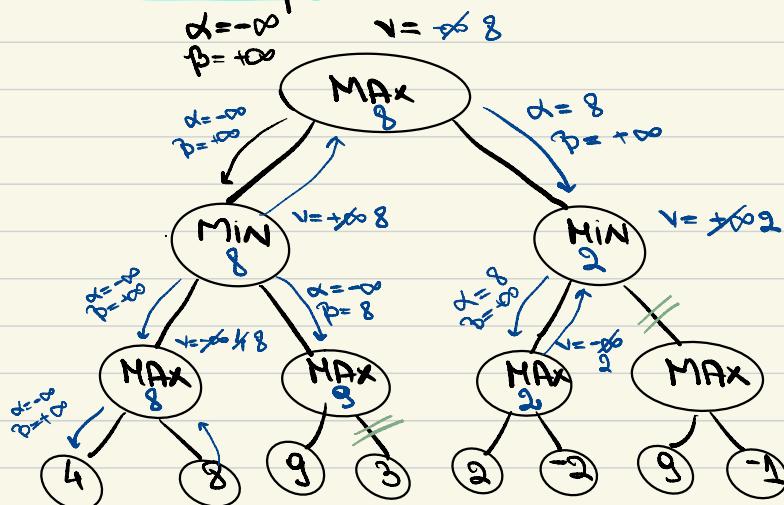
$v' = \text{Min-Value}(c, \alpha, \beta)$

if $v' > v$, $v = v'$

Ex. fără tăiere $\alpha - \beta$



Ex. cu tăiere $\alpha - \beta$:



α : cea mai bună alternativă pt. MAX până la momentul actual (de la starea inițială → starea curentă)

de updatat cînd se găsește o valoare mai bună pt. v.

β : cea mai bună alternativă pt. MIN pe path-ul de la starea curentă până la finalizare

* am găsit o valoare mai bună pt. α ?
(* alternativă mai bună pt. MAX)

! prima MAX \rightarrow 3: valoarea curentă a lui \Rightarrow pt. MAX este
mai mare decât $\beta \rightarrow$ MIN nu va alege acea
valoare

\Rightarrow nu mai are sens să continuă
expansarea pt. acel nod

! prima pe nivel MAX

: dacă valoarea curentă $v > \beta$ (căci și MIN până în acel punct)

MIN nu va alege ca MAX să aibă o valoare
mai mare, așa că nu va alege decic
aceea acela \Rightarrow MAX nu va rea să scrie
fiecare moduluri copii,
indiferent dacă ar găsi
valoarea mai bună

! prima pe nivel MIN : va alege MAX pasul următor? Va putea să-i maximizeze dacă $\alpha < \beta \Rightarrow$ MAX nu va alege pasul următor
valoarea curentă pt. MIN

Algoritmi de căutare

- **complet**: garantează găsirea unei soluții dacă I, dacă nu reușește pe cale optimă
- **admisibil**: garantează găsirea soluției optimă

- alg. infor.
- **best-first**: generalizare alg. de căutare neinformată
 - **A*** → **admisibilitate**:
 - * $h(s)$ admisibilă
 - * cost-arc(s, s') = c , $c > 0$ constantă finită

Functie heuristică h

- **compatibilă** (regula triunghiului)

$$h(s) \leq h(s') + \text{cost-arc}(s, s')$$
- **monotonă**: $h(x) > h(y)$, $x > y$
- **admisibilă**: $\forall s \neq s_f, h(s) \leq h^*(s)$

$$h(s_f) = 0$$

- h compatibilă $\Leftrightarrow h$ monotonă
- h compatibilă \Rightarrow admisibilă

- **IDA***:
 - pe principiul ID, dacă limită este doar de cănd drumul se reia
 - algoritm cu o nouă limită de cănd până când se găsește sol.

Algoritmi de căutare locală

- **Hill Climbing**
- **stochastic**: se alege aleator o singură stocare s_j pt. care $E(s_j) > E(s)$
 - **first-choice**: generează aleator succesor până când $E(s_j) > E(s)$
 - **random restart**: generează stocare inițiale aleator + HC din fiecare

finet

- Local Beam Search → generație de stări și alegeră cele mai bune la stări succesoare dintr-o setă
- Stochastic Local Beam Search → alegeră zonă random pt. cea k succo-

Algoritmi de căutare online

- Online DFS → moduri expandate pe bază localității
 - ↳ măștă (stări, acțiuni) la stări generate
- Programare dinamică asimorantă (ADP) → pt. fiecare stare i: $f(i) = c(i, j) + f(j)$
j vecin al lui i

- Learning Real Time A* (LRTA*) → explorare repetată a sp. pătră cănd modurile au valori min.
 - ↳ $f_i(j) = \min_j (f_i(j))$
je vecin(i)
 - ↳ f_i este calculat în raport decât cu starea curentă, nu cu starea inițială

CSP (Constraint Satisfaction Problem)

Algoritmi de îmbunătățire a consistenței reprezentării
(încărcături de încreștere căutării)

- Propagare locală a restricțiilor : se ia fiecare restricție și se modifică domeniile variabilelor
 - AC-3 (Arc Consistency) : $O(e \cdot a^3)$ → cardinalitate max.
↳ (x, y) arc consistent $\Leftrightarrow \forall x \in D_x, \exists y \in D_y \text{ a.s. } R(x, y)$
 - AC-4 : $O(e \cdot a^2)$
- Verif. consistență
deorice orice

- PC-4 (m-cale consistență) : $O(N^3 \cdot a^3)$

• 2 cale consistentă: $R_{ij} \leftarrow R_{ij} \wedge (R_{ik} * R_{kj} * R_{ij})$

• trebuie să se poată verifica consistența pe doar o singură cale

Backtracking: Verificarea de coherență și backjumping

- **MAC (Maintaining Arc Consistency)**: păstrarea coherenței

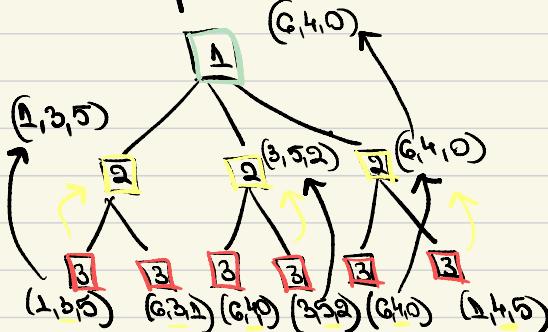
Euristică

- **variabile** → fail-first: cele mai puține valori posibile
- fail-first → fail-last: cele mai multe valori
- **valori** → fail-first: valoare care elimină cele mai multe valori din dom. valoarelor variabilei menținute cu care este legată variabila
- fail-first → fail-last

Algoritmi de jocuri

Jocuri cu mai mulți jucători

- **MAX^m**:
 - generalizare Minimax pt. m jucători
 - m-tuple cu valori corespondente pt. fiecare jucător



- **Paromioic**
 - un jucător MAX, restul MIN \Rightarrow se poate aplica MINIMAX
 - se poate folosi $\alpha-\beta$

Monte Carlo Tree Search

Machine Learning

Clasificare

- **Naive Bayes**:
 - nu ia în considerare ordinea de apariție a evenimentelor
 - consideră independentele atributelor unei variabile

• ID3 (Arbore de decizie) : $O(|C| \cdot |A| \cdot |N|)$

→ se termină în întindere

$$\rightarrow G(A) = I(AD_{p,m}) - E(A)$$

$$\rightarrow I(AD_{p,m}) = -\frac{p}{m+p} \log \frac{p}{p+m} - \frac{m}{m+p} \log \frac{m}{m+p}$$

$$\rightarrow E(A) = \sum_{i \in m} \frac{p_i \cdot m_i}{p \cdot m} \cdot I(AD_{p_i, m_i})$$

! • Cazuri speciale

I → $\exists a \in C$ pt. care $A = A_j$

II → argument um si

III → valori necunoscute de atribută

(Algoritm Ch.5) → corăge AD pt. necunoscute valori
lipsă (val. lini A sunt doar de tip)

atribut val.
cu cel mai mare
freqv.

spun determinarea
dp a valoarelor lini
A în C

$$P(A=A_i | Q_{00} = C)$$

IV → atribut cu mai multe valori

produciere valori
în intervale

- imposibile valori
 $(-\infty, A_1]$ și $(A_i, +\infty)$, $i = \overline{1, m}$
- selecție pozitie cu cotație max.

$$\bullet \text{inf. de separare: } I(\text{sep}(A)) = - \sum_{i=1}^m \frac{p_i \cdot m_i}{p \cdot m} \log \frac{p_i \cdot m_i}{p \cdot m}$$

$$GR(A) = \frac{G(A)}{I(\text{sep}(A))} \text{ max.}$$

Rețele Neurale

• f. de activare → softmax: trimit un vector de nr. reale într-o distribuție de probabilități

Rețele Backpropagation

→ Alg. Forward

Alg. Backward: chain rule

Rețele Neurale Convoluționale

Schimbări convoluționale →

- nucleu de convol. + op. de convoluție
- rez. hărți de activare
- nucleu / filtre aplicate în paralel
- outputul i se aplică f. de activare.

Schimbări de pooling

Procesarea Limbajului Natural

Grammatici regulate

$$G = \{N, \Sigma, R, S\}, R = \{x \rightarrow a \text{ sau } x \rightarrow a'y\}$$

Grammatica imdep. de context

$R = \{ \text{ comb. de terminali } + \text{metaterminali} \}$

Grammatica imdep. de context - probabilistica

Algoritm CKY : $O(N^3 \cdot M)$

- lungimea curorii
- nr. de metaterminali

- aplicare pe Gicp în Forma Normală Chomsky

Grammatica cu lărgire definită

(DCG)

Ex: Prolog

Model N-Gram : distanță de probabilitate a unui cuv. / cuv. | coracor, având în considerare $N-1$ cuvinte / cuvinte anterioare

Word Embedding : vector de cod. pt. fiecare cuvânt
↳ să exprimeasem conținutul dimite cuvântu

PMI (Pointwise Mutual Information)

$$\text{PMI} = \log_{10} \left(\frac{P(a,b)}{P(a)P(b)} \right) \quad (=0 \Rightarrow a, b \text{ sunt corelate})$$

Agensi

- cognitivi : atât mediul, cât și starea agentului se modifică în f. de acțiuni
- reactivi : mediul se modifică în f. de acțiunile agentului

Negocieri → bazați pe teoria jocurilor \rightarrow Joc cu coșting comun

\rightarrow Joc cu suveran mediu / comunitate

Eficiență Pareto : \times sănătate este ef. Pareto doar :

mai \exists altă sol. x' a. i. \rightarrow căl puțin un agent are o utilitate mai mare decât în x'
utilizatul meu are o utilitate mai mică în x' față de x

Relationalitate Individuală (IR)

\rightarrow rolul agentului în urmă participarea la negocieri nu este mai unică decât rolul doar nu ar participa la negocieri

Bunăstare socială :

- \sum utilităților
- bimbi general