Examen Inteligență Artificială 2021-2022

Notă:

- **Acest fișier conține rezolvarea și explicațiile subiectului dat la examenul de Inteligență Artificială din Ianuarie 2022;**
- **În total sunt 18 întrebări, fiecare valorând 5 puncte;**
- **Răspunsurile corecte sunt marcate cu verde;**

1. **Care este tipul de Inteligență Artificială mai potrivit pentru un sistem conceput să treacă testul Turing?**

   - ☐ **Comportament uman.**
   - ☐ Gândire umană.
   - ☐ Gândire rațională.
   - ☐ Comportament rațional.

   Explicație:

   Testul Turing este un experiment din domeniul IA și se referă la posibilitatea mașinilor de calcul de a se **comporta** mai mult sau mai puțin asemanator oamenilor.

   Dacă o persoană nu este capabilă să facă diferența dintre om și calculator, atunci putem spune că mașina de calcul a trecut testul Turing, din această cauză este mai potrivit să se comporte uman, decât să gândească asemenea unui om.

2. **Considerăm datele de mai jos. Aplicați regula de antrenare a perceptronului pentru a actualiza ponderile. Considerați funcția de activare treaptă, pragul egal cu 0.2 și rata de învățare de 0.1. Ponderile inițiale sunt egale cu 0.3 și -0.1. Care din următoarele afirmații sunt adevărate?**

Inputs		Desired output
x_1	x_2	Y_d
0	0	0
0	1	0
1	0	0
1	1	1

- ☐ După al treilea exemplu, ponderile sunt 0.2 și -0.2
- ☐ **După primul exemplu, perceptronul returnează valoarea 0**
- ☐ După al treilea exemplu, perceptronul returnează valoarea 0
- ☐ **După primul și al doilea exemplu, vectorul de ponderi nu se modifică**
- ☐ **După al treilea exemplu, ponderile sunt 0.2 și -0.1**

Explicație:

Aplic regula de antrenare a perceptronului: $w_i \leftarrow w_i + \Delta w_i$

$\Delta w_i = \eta(t-o)x_i$

learning rate = 0.1

Primul exemplu: $0 \cdot 0.1 - 0 \cdot 0.1 = 0 \Rightarrow 0$, deci al doilea răspuns este adevărat

Al doilea exemplu: $0 \cdot 0.3 - 1 \cdot 0.1 = -0.1 \Rightarrow 0$, deci al patrulea răspuns este adevărat

Al treilea exemplu: $0.3 \cdot 1 - 0.1 \cdot 0 = 0.3 > 0.2 \Rightarrow 1$, deci al 3-lea răspuns este greșit

Calculez ponderile:

Err: $0 - 1 = -1$

$-1 \cdot 0,1 = -0,1$

$w_1 = 0,3 + (0-1) \cdot 0,1 \cdot 1 = 0,3 - 0,1 = 0,2$

$w_2 = -0,1 + (0-1) \cdot 0,1 \cdot 0 = -0,1 + 0 = -0,1$

} => ultima variantă este corectă, și prima este greșită.

3. **Considerăm problema de planificare reprezentată mai jos. Considerăm planul inițial la care am adăugat acțiunile Buy(Drill, HW), Buy(Milk, SM), Buy(Bananas, SM). Care din afirmațiile de mai jos sunt adevărate?**

```
Actions:
Buy(x, store)
PRECOND: At(store), Sells(store, x)
EFFECT: Have(x)
Go(x, y)
PRECOND: At(x)
EFFECT: At(y), ¬At(x)
Init: At(Home) and Sells(SM, Milk) and Sells(SM, Banana) and Sells(HW, Drill)
Goal: Have(Milk) and Have(Banana) and Have(Drill)
```

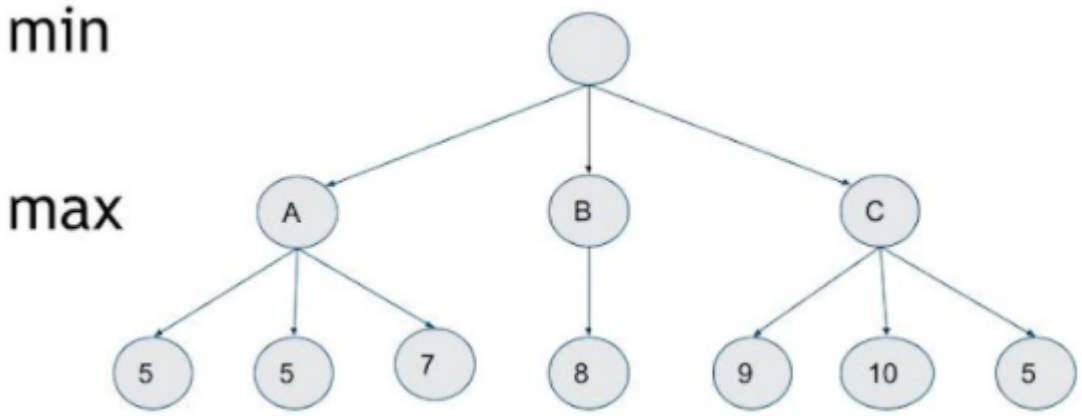
- ☐ Precondiția At(SM) este satisfăcută din starea inițială
- ☐ **Precondiția deschisă Sells(SM, Milk) este satisfăcută din starea inițială**
- ☐ **Pentru a satisface precondiția deschisă At(SM) adăugăm acțiunea Go(Home, SM)**
- ☐ Planul este o soluție
- ☐ **Am adăugat acțiunea Buy(Milk, SM) pentru a produce Have(Milk) necesară satisfacerii obiectivului**

Explicație:

Le luăm pe rând:

- Prima variantă nu este corecta deoarece precondiția At(Sm) nu se găsește în Init.
- A doua variantă este corectă deoarece precondiția deschisă Sells(SM, Milk) se găsește în Init.
- A treia variantă este adevărată deoarece Go(x, y) spune că trebuie să fii neapărat în x(Home în cazul nostru, și avem precondiția satisfăcută în Init), și are ca efect să fii în y(SM în cazul nostru) și să nu mai fii în x(Home). Fără acțiunea Go(Home, SM) nu poți ajunge la SM.
- A patra variantă nu este corectă deoarece fără Go(Home, SM), care nu exista în planul inițial, nu se poate ajunge la soluție.
- Ultima variantă este corectă deoarece Have(Milk) este imposibil de satisfăcut fără acțiunea Buy(Milk, SM). Același lucru se aplică și pentru Have(Banana) cu Buy(Banana, SM) și Have(Drill) cu Buy(Drill, SM).

4. **Pentru explorarea MINIMAX cu retezarea ALFA BETA din arborele de mai jos, câte stări nu vor mai fi evaluate?**



- [x] **2**
- [ ] 3
- [ ] 4
- [ ] 0
- [ ] 1

Explicație:

Cele două frunze care nu vor mai fi evaluate sunt 10 și 5 din subarborele cu rădăcina C.

Pentru arborele dat, MINIMAX cu retezarea ALPHA BETA va rula astfel:

1. Pornește din rădăcina, și inițializează pe alfa, beta și scorul rădăcinii cu -1000(să zicem).
2. Ajunge în A cu valorile din rădăcină.
3. Ia pe rând frunzele din nodul A și verifică dacă sunt mai mari decât scorul din nod, iar în caz afirmativ, beta și scorul vor fi actualizate corespunzător. De asemenea, verifica și dacă scorul este mai mare decat alpha, caz în care stările rămase nu vor mai fi evaluate.
4. Verifică dacă scorul din A este mai mic decât cel aflat în rădăcină, și actualizează scorul și pe alpha iar. Beta va fi din nou inițializat cu -1000.
5. Merge în B cu valorile din rădăcină.
6. Ia frunza din nodul B și verifică dacă e mai mare decât scorul din nod, iar în caz afirmativ, beta și scorul vor fi actualizate corespunzător. De asemenea, verifica și dacă scorul este mai mare decat alpha, caz în care stările rămase(aici nu e cazul) nu vor mai fi evaluate.
7. Verifică dacă scorul din B este mai mic decât cel aflat în rădăcină, și actualizează scorul și pe alpha iar. Beta va fi din nou inițializat cu -1000.

8. Ajunge la nodul C cu alfa având valoarea 7, iar beta și scorul egale cu -1000. După ce ia valoarea 9, acesta este mai mare decât scorul actual(-1000), și este trecută ca noul scor al lui C. Se face verificarea dacă scorul(9) este mai mic decat alfa(7), din moment ce 9 nu este mai mic decat 7, putem spune că nu exista vreo valoare în C care sa fie mai mica decat 7, deci restul stărilor(adică 10 și 5) nu mai trebuie evaluate și alfa va rămâne 7.

Tool online pentru vizualizarea lui MINIMAX: http://homepage.ufp.pt/jtorres/ensino/ia/alfabeta.html

5. **Fie problema aranjării în ordine crescătoare a 8 piese numerotate (http://www.8puzzle.com/8_puzzle_problem.html). Care din următoarele euristici sunt admisibile?**

- ☐ f(s) = distanța în linie dreaptă între poziția curentă a primei piese pe poziție greșită și poziția corectă a acelei piese
- ☐ f(s) = suma pieselor adiacente căsuței libere
- ☐ **f(s) = suma distanțelor în linie dreaptă între pozițiile curente ale pieselor și poziția corectă a acelor piese**
- ☐ **f(s) = numărul de piese care nu se găsesc pe poziția corectă (în ordine crescătoare)**

Explicație:

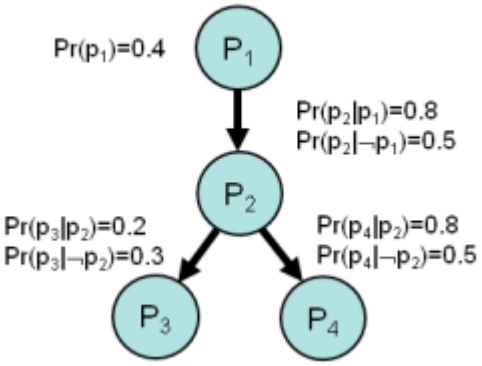
După cum știm, o euristică bună trebuie în primul rând să ajute la găsirea stării finale, să nu supraestimeze niciodată distanța dintre o stare oarecare și goal și să fie relativ ușor de calculat.

Toate euristicile din variante sunt relativ ușor de calculat, dar nu toate ajută la găsirea stării finale.

Primele două euristici nu ajută deoarece să știm o singură distanță între poziția curentă a primei piese pe poziție greșită și poziția corectă a acelei piese nu este îndeajuns, deoarece nu am avea un ansamblu complet al întregii matrici, ci doar pentru două elemente, iar suma pieselor adiacente căsuței libere este irelevantă pentru orice strategie am alege.

Ultimele două euristici pot fi de folos alături de strategia potrivită, suma distanțelor în linie dreaptă între pozițiile curente ale pieselor și poziția corectă a acelor piese ne spune că mai trebui făcute mișcări(dacă e mai mare decât 2) sau nu, iar numărul de piese care nu se găsesc pe poziția corectă ne spune dacă am ajuns sau nu în starea finală(dacă întoarce 0) sau nu.

6. **Fie rețeaua bayesiană de mai jos. Valoarea probabilității P(p1 | p2, not p3) este:**



- ☐ 1
- ☐ 0.256
- ☐ **0.516**
- ☐ 0
- ☐ 2.016

Explicație:

Folosim algoritmul de inferență prin enumerare:

P_4 este singura variabilă ocazională

Acum vai calcula independent $P(P_1 | P_2, \neg P_3)$ și $P(\neg P_1 | P_2, \neg P_3)$

$$P(P_1 | P_2, \neg P_3) = \alpha \cdot \sum_{P_4 \in \{T, F\}} P(P_1, P_2, \neg P_3, P_4) = \alpha \cdot \sum_{P_4 \in \{T, F\}} P(P_1) \cdot P(P_2 | P_1) \cdot P(\neg P_3 | P_2) \cdot P(P_4 | P_2)$$

$$= \alpha \cdot P(P_1) \cdot P(P_2 | P_1) \cdot P(\neg P_3 | P_2) \cdot \sum_{P_4 \in \{T, F\}} P(P_4 | P_2) = \alpha \cdot 0,4 \cdot 0,8 \cdot 0,8 \cdot (0,8 + 0,2)$$

$$= \boxed{0,256\,\alpha}$$

$$P(\neg P_1 | P_2, \neg P_3) = P(\neg P_1) \cdot P(P_2 | \neg P_1) \cdot P(\neg P_3 | P_2) \cdot \sum_{P_4 \in \{T, F\}} P(P_4 | P_2) \cdot \alpha$$

$$= 0,6 \cdot 0,5 \cdot 0,8 \cdot (0,8 + 0,2) \cdot \alpha = \boxed{0,24\,\alpha}$$

Știm că $P(P_1 | P_2, \neg P_3) + P(\neg P_1 | P_2, \neg P_3) = 1 \Rightarrow 0,256\,\alpha + 0,24\,\alpha = 1$

$\Rightarrow 0,496\,\alpha = 1 \Rightarrow \boxed{\alpha = 2,0161}$

Deci $P(P_1 | P_2, \neg P_3) = 0,256 \cdot 2,0161 = 0,5161$ $\Rightarrow$ varianta 3

7. **Care din următoarele strategii, aplicate la o problemă modelată cu stări, permit totdeauna recuperarea unei soluții, dacă ea există?**

- ☐ **Depth First Search**
- ☐ **A*, indiferent de euristica aleasă**
- ☐ **Backtracking**
- ☐ Hillclimbing, indiferent de euristica aleasă

Explicație:

La această întrebare părerile au fost împărțite cu privire la faptul ca DFS trebuia bifat sau… astfel s-a decis ca ambele variante(cu sau fără DFS bifat) să fie considerate corecte.

Începem cu cea mai clară dintre toate: Hillclimbing, indiferent de euristica aleasă, selectează doar stările cu scorul mai mare sau egal cu cel al stării curente, dar pot exista și soluții în care să fie necesară alegerea unei stări cu scor mai mic, și astfel Hillclimbing riscă sa se blocheze într-un punct de optim local, deci nu va găsi soluție.

A* este una dintre cele mai bune strategii, tot timpul alege stările neexplorate care sunt cele mai apropiate atât de starea inițială, cat și de starea finală, și nu se oprește când ajunge la starea finală. Deci permite totdeauna recuperarea unei soluții, dacă ea există.

Despre BKT știm că setează o anumită ordine în care explorează vecinii, nu memorează stările și în general nu are bucle, deci întoarce soluție.

DFS vizitează cel mai apropiat vecin până când ajunge la starea finală, trebuie să memoreze fiecare stare generată, poate revizita aceleași stări, și uneori se poate bloca din cauza buclelor(în grafuri infinite), și să fie în incapabilitate de a recupera o soluție.

8. **Considerăm următoarele categorii ale unui meniu: aperitiv (A), fel principal (F), băutură (B), desert (D). Domeniile acestor variabile sunt: A: legume, brânză, B: apă, suc, lapte, F: pește, vită, paste, D: plăcintă, înghețată, fructe. Avem următoarele restricții: 1) alegem la aperitiv legume sau felul principal trebuie să fie pește sau paste (sau ambele) 2) dacă servim brânză, ne permitem doar apă 3) servim cel puțin una din: lapte, înghețată, fructe. Care din afirmațiile de mai jos sunt adevărate?**

- ☐ **Dacă A=brânză, prin aplicarea unui prim pas al algoritmului Forward Checking, vor fi eliminate valorile suc, lapte, vită din domeniile lui B și F**
- ☐ Graful de restricții are 4 noduri și 4 muchii

- ☐ Dacă A=legume, prin aplicarea unui prim pas al algoritmului Forward Checking, va fi eliminată valoarea plăcintă din D
- ☐ Problema nu admite soluții
- ☐ **Instanțierea A=brânză, F=pește, B=apă, D=înghețata este o soluție**

Explicație:

Prima variantă este corectă deoarece:

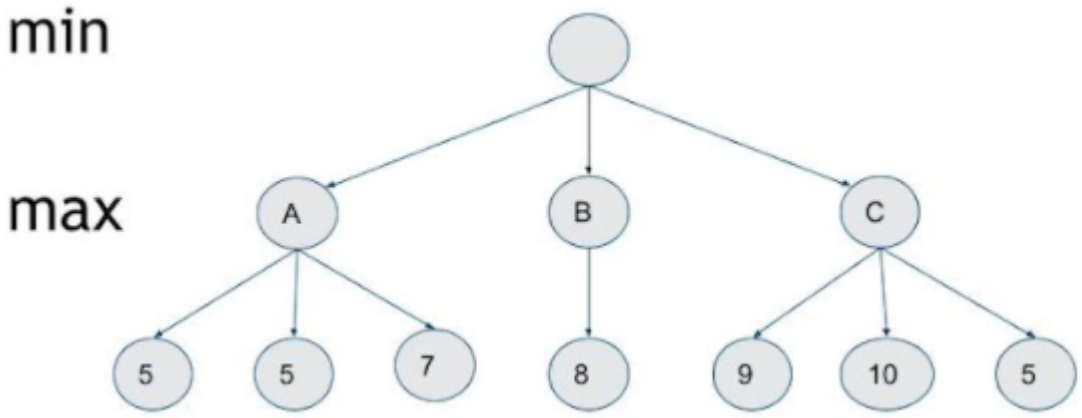
- Forward checking actualizează domeniul variabilelor neasignate, atunci când selectăm o valoare pentru o variabilă, elimină valoarea din domeniu variabilelor neasignate, conectate cu aceasta.
- Prima restricție spune că dacă la aperitiv nu avem legume, la felul principal trebuie neapărat pește sau pase sau ambele. Deci vită este eliminat din domeniul F..
- A două restricție spune că dacă luăm brânză, ne permitem doar apă, deci suc și lapte vor fi eliminate din domeniul lui B.

Ultima variantă este corectă deoarece satisface toate constrângerile, și anulează a patra variantă.

A doua variantă nu este corectă deoarece graful nu are 4 muchii.

A patra variantă nu este corectă deoarece nu există constrângere care să interzică placinta la desert dacă s-au servit legume la aperitiv.

9. **Pentru explorarea MINIMAX din arborele de mai jos, care va fi starea selectată ca optimă pentru mutarea următoare și ce scor are asociat acea stare?**



- ☐ C cu scor 9
- ☐ C cu scor 10
- ☐ A cu scor 5

- ☐ B cu scor 8
- ☐ **A cu scor 7**

Explicație:

După explorarea explicată pe larg la exercițiul 4., valoarea lui alfa este 7, deci starea selectată ca optimă pentru mutarea următoare este A și are scorul 7.

10. Care este echilibrul Nash pentru jocul din imagine?

A \ B	s	d
s	2, 1	0, 0
d	0, 0	1, 2

- ☐ nu exista
- ☐ (1/2,1) pentru A, (1,1/2) pentru B
- ☐ **(1/3,2/3) pentru A, (2/3,1/3) pentru B**
- ☐ (1/2,1/2) pentru A, (1/2,1/2) pentru B
- ☐ (1/2,1/2) pentru A, (1/3,2/3) pentru B

Explicație:

Prima variantă din start iese din calcul deoarece teorema spune că orice joc are un măcar un echilibru Nash, fie el pur sau mixt. La a doua variantă, suma probabilităților din paranteze dă peste 1, ceea ce nu e posibil.

Pentru a determina echilibrul Nash al acestui joc, este de ajuns să vedem ce strategii ar alege jucătorii în orice situație, și anume:

- Dacă A ar alege strategia s, cel mai convenabil pentru B ar fi să aleagă tot strategia s(1 > 0).
- Dacă A ar alege strategia d, cel mai convenabil pentru B ar fi să aleagă tot strategia d(2 > 0).
- Dacă B ar alege strategia s, cel mai convenabil pentru A ar fi să aleagă tot strategia s(2 > 0).
- Dacă B ar alege strategia d, cel mai convenabil pentru A ar fi să aleagă tot strategia d(1 > 0).

Observăm că avem echilibru Nash atunci când ambii jucători aleg aceeași strategie, mai rămâne să deducem probabilitățile din tabel. A are scor mai mare dacă alege tot timpul s, deci va alege s cu probabilitatea de 2/3 și pe d cu

probabilitatea de 1/3. La B este invers, el va alege pe d cu probabilitatea de 2/3 și pe s cu probabilitatea de 1/3. Rezultă că varianta a treia este corecta.

11. **Care din afirmațiile de mai jos sunt adevărate?**
    - ☐ Spațiul necesar pentru a reprezenta distribuția comună de probabilitate este O(n), n numărul de variabile
    - ☐ **Rețelele bayesiene permit specificarea compactă a distributiei comune de probabilitate**
    - ☐ Următorul produs de factori corespunde unei rețele bayesiene valide peste variabilele A, B, C, D: P(A | B) P(B | C) P(C | D) P(D | A)

Explicație:

A doua variantă este corectă deoarece după cum vedem spre exemplu la exercițiul 6, distribuția comună de probabilitate este specificată compact. Asta ajută să realizăm raționamente cât mai eficient.

Prima variantă nu este corectă deoarece spațiul necesar pentru a reprezenta distribuția comună de probabilitate este mai mare decât O(n), cu siguranță pot exista probabilități condiționate pe lângă cele simple.

Iar ultima variantă nu poate fi validă, deoarece în acel produs cu siguranță nu toate variabilele depind doar de părinți.

12. **Două ontologii de domeniu, create de două persoane diferite, ce includ aceleași concepte și relații semantice vor fi întotdeauna identice?**
    - ☐ Da, doar dacă sunt create corect
    - ☐ **Nu**
    - ☐ Da, dacă autorii au niveluri similare de competență

Explicație:

Răspunsul este clar Nu, nivelul de competență al autorilor nu este relevant în crearea unei ontologii și corectitudinea unei ontologii nu poate fi determinată în mod obiectiv.

13. **Care din următoarele afirmații sunt adevărate?**
    - ☐ **În cadrul unui proces de decizie Markov, recompensele sunt cunoscute**
    - ☐ Dacă factorul de discount este aproximativ egal cu 0, recompensele din viitorul îndepărtat sunt cele mai semnificative

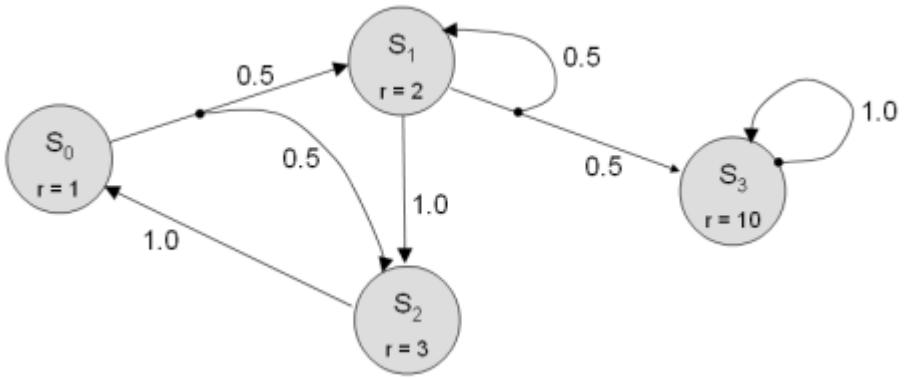
- ☐ **În cadrul metodei ε-greedy agentul are mereu o șansă de explorare deoarece ε nu scade sub un prag**
- ☐ În cadrul algoritmilor de învățare cu întărire, modelul de tranziții este cunoscut

Explicație:

Le vom lua pe rând:

- Recompensele sunt cunoscute, astfel agentul este ghidat către starea finală favorabilă.
- Aici este invers, dacă factorul de discount este aproximativ egal cu 0, recompensele din viitorul apropiat sunt cele mai semnificative. Adică agentul gândește pe termen scurt, nu pe termen lung.
- Adevărat deoarece în cadrul metodei ε-greedy, ε nu scade niciodată sub un prag setat, și astfel agentul evită optimele locale.
- Modelul de tranziții nu este cunoscut, acesta este format în timp real cu fiecare decizie a agentului.

14. **Considerăm următorul proces de decizie Markov. Aplicați algoritmul Value Iteration. Factorul de discount este 1. Există o singură acțiune disponibilă pentru fiecare stare, cu excepția stării S1 care are două acțiuni. Valorile inițiale ale utilităților sunt 0. Care din afirmațiile de mai jos sunt adevărate?**



- ☐ **Utilitatea stării S1 după prima iterație este 2**
- ☐ **Utilitatea stării S1 după a doua iterație este 8**
- ☐ Utilitatea stării S0 după prima iterație este 0
- ☐ **Utilitatea stării S2 după prima iterație este 3**

Explicație:

Aplicăm Value iteration

Deoarece $\gamma = 1$

Iterația 1:

$U_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a) U_i(s')$

P	S_0	S_1	S_2	S_3
S_0	-	0.5	0.5	-
S_1	-	0.5	1	0.5
S_2	1	-	-	-
S_3	-	-	-	1

$U_2(S_0) = R(S_0) + \gamma \max(0,5 \cdot 0 \ , 0,5 \cdot 0)$

$= 1 + 1 \cdot 0 = 1 \Rightarrow$ a 3-a variantă este greșită

$U_2(S_1) = R(S_1) + \gamma \max(0,5 \cdot 0 \ ; 1 \cdot 0 ; 0,5 \cdot 0)$

$= 2 + 1 \cdot 0 = 2 \Rightarrow$ prima variantă este corectă

$U_2(S_2) = 3 + \gamma \max(1 \cdot 0)$

$= 3 + 1 \cdot 0 = 3 \Rightarrow$ ultima variantă este corectă

$U_2(S_3) = 10 + \gamma \max(1 \cdot 0) = 10$

Iterația 2:

$U_3(S_1) = R(S_1) + \gamma \max(0,5 \cdot 2 \ ; \ 1 \cdot 3 \ ; \ 0,5 \cdot 10)$

$= 3 + 1 \cdot \max(1, 3, 5) = 3 + 5 = 8 \Rightarrow$ a 2-a variantă este corectă.

15. **Pentru problema "Turnurile din Hanoi" generalizată, formulată aici: https://ggle.io/4ZCI , schimbăm regula de mutare în sensul că permitem mutarea a cel mult două piese de pe aceeași tijă simultan. Ce trebuie schimbat în modelarea discutată la curs pentru a fi adaptată la această modificare?**

- ☐ Nimic
- ☐ Strategia
- ☐ Verificarea stării finale
- ☐ **Tranziția și validarea ei**
- ☐ Inițializarea

Explicație:

Tranziția reprezintă modul în care putem ajunge de la o stare la alta(cum mutăm piesele în cazul nostru), deci trebuie schimbată. Desigur ca și funcția de validare se schimbă odată cu tranziția. Starea finală este aceeași, inițializarea de asemenea și strategia poate funcționa la fel.

16. **Care este specificul unui "weak AI" - inteligență artificială slabă?**

- ☐ **Simulează comportamente inteligente**
- ☐ Nu poate interacționa cu mediul extern
- ☐ Posedă mecanisme de raționament inteligent
- ☐ Produce soluții ineficiente pentru probleme

Explicație:

Spre deosebire de un ”strong AI”, care se comportă inteligent și într-adevăr este așa(are mecanisme inteligente), ”weak AI” este mai degrabă implementat pe baza unui template(nu posedă mecanisme inteligente) și astfel el simulează inteligența, dar poate interacționa cu mediul extern(ChatBot) și produce soluții eficiente.

17. **Care din afirmațiile de mai jos referitoare la planificare cu ordine parțială sunt adevărate?**

- ☐ **Acțiunile din Start au ca efecte literalii din starea inițială**
- ☐ **Se adaugă restricții de ordonare pentru a rezolva conflicte între legăturile cauzale**
- ☐ Acțiunile din Finish nu au precondiții care trebuie satisfăcute
- ☐ **Planul vid conține Start și Finish**
- ☐ Într-o soluție pot exista precondiții deschise

Explicație:

Începem cu variantele greșite:

- Știm că precondițiile deschise nu sunt îndeplinite de nici o acțiune, deci nu au ce căuta într-o soluție.
- Precondițiile din Finish sunt deschise, deci există.

Variantele corecte:

- Prima nu are nevoie de explicații.
- Este adevărat, se adaugă astfel de condiții de ordonare deoarece, spre exemplu, un ciclu $A < B$ și $B < A$ reprezintă o contradicție.

- Un plan vid conține doar acțiunile Start și Finish, deci e corectă și a treia variantă.

18. **Care din următoarele afirmații referitoare la probleme de satisfacere a restricțiilor sunt adevărate?**

- ☐ Nu este necesară utilizarea algoritmului Backtracking după aplicarea algoritmului Arc consistency pentru a identifica soluția problemei
- ☐ Euristica Least Constraining Value este utilizată pentru a identifica următoarea variabilă de asignat
- ☐ **Algoritmii de propagare a constrângerilor pot reduce domeniile variabilelor**
- ☐ O asignare consistentă este o asignare în care toate variabilele au atribuite valori
- ☐ Utilizarea euristicilor de ordonare a valorilor asigură rezolvarea problemelor CSP în timp liniar

Explicație:

Le luăm pe rând:

- Arc consistency se ocupă de reducerea domeniilor variabilelor, și este nevoie de BKT pentru identificarea soluției, deci e Fals.
- Euristica Least Constraining Value este utilizată pentru a alege valoarea cea mai puțin constrânsă (cea care exclude cele mai puține valori din variabilele vecine), deci e Fals.
- Pentru asta sunt concepuți, și dacă e nevoie, aceștia pot reduce domeniile variabilelor, deci e Adevărat.
- O asignare este consistenta dacă nu sunt încălcate constrângeri, deci e Fals.
- Doar probleme CSP cu structură arborescentă pot fi rezolvate în timp liniar, deci e Fals.