


Adrian GROZA

INTELIGENŢĂ ARTIFICIALĂ

EXERCIŢII

U.T. PRESS
CLUJ-NAPOCA, 2019
ISBN 978-606-737-374-5


Adrian GROZA

INTELIGENȚĂ ARTIFICIALĂ

EXERCIȚII

Editura UTPRESS
Cluj-Napoca, 2019
ISBN 978-606-737-374-5

Editura U.T.PRESS
Str.Observatorului nr. 34
400775 Cluj-Napoca
Tel.:0264-401.999
e-mail: utpress@biblio.utcluj.ro
http://biblioteca.utcluj.ro/editura

Director: Ing. Călin D. Câmpean

Recenzia: Conf.dr.ing. R. Slăvescu
Conf.dr.ing. A. Mărginean

ISBN 978-606-737-374-5

Cuprins

Cuvânt înainte

Despre inteligenţa artificială se vorbeşte din ce în ce mai mult. Ca urmare, cei care încep să studieze inteligenţa artificială sunt mai eterogeni şi au metode şi profile de învăţare diferite. Exerciţiile din această carte vin sub forma unor jocuri din două motive. Pe de o parte, limbajul universal al jocului permite mai multor tipuri de studenţi să se aproprie de domeniul vast şi deloc simplu al inteligenţei artificiale. Pe de altă parte, jocul este un mecanism încă eficient în creşterea motivaţiei studenţilor care rezonează tot mai puţin cu învăţarea doar pe baza unui curs. Jocurile se potrivesc profilului studentului contemporan care nu respinge competiţia, care doreşte un câştig imediat chiar dacă este mic sau care este adeseori ghidat de imperativul distracţiei.

Această carte este un instrument ajutător celor care studiază inteligenţa artificială utilizând manualul *Artificial Intelligence - A Modern Approach* (AIMA) a lui Russell şi Norvig [1]. AIMA este văzută de către studenţi ca extrem de densă. Pentru a sparge această nucă tare numită AIMA, am dorit oferirea către studenţi a unui instrument complementar care să-i ajute în procesul de construire a hărţii cognitive proprii pe domeniul inteligenţei artificiale. Pentru crearea de conexiuni între numeroasele metode prezentate în AIMA mă bazez pe două tipuri de jocuri: 1) jocuri de tip Kahoot (www.kahoot.com), respectiv 2) rebusuri în inteligenţă artificială.

Jocurile de tip Kahoot reprezintă un test rapid, în limita a 5 minute, la care studenţii participă la finalul fiecărui curs de inteligenţă artificială. Sunt relevante corectitudinea răspunsului şi timpul de gândire. Ipoteza este că un student atent la discuţiile din timpul cursului ar putea răspunde la întrebări. Exerciţiile propuse în această carte sunt un antrenament pentru testele din timpul cursurilor. Formatul tipărit poate reproduce unele proprietăti ale jocului precum timpul alocat fiecărei întrebări, dar pierde elemente specifice interacţiunii directe: utilizarea dispozitivelor mobile, interacţiunea şi suportul colegilor vecini, efectele grafice şi muzica specifice platformei Kahoot.

Jocuri de tip rebus au fost dezvoltate pentru fiecare capitol din AIMA. Studenţii trebuie să vâneze cuvintele cerute în textul sursă. Se încurajează astfel identificarea unor concepte din AIMA, uneori complementare celor prezentate la curs. Ipoteza este că un student curios ar putea citi mai mult despre metodele din inteligenţa artificială care sunt prezentate doar într-o singură propoziţie în AIMA. Aceste metode ar putea trece neobservate de către studenţii care citeşc din AIMA doar părţile relevante pentru întelegerea prezentărilor din timpul cursului.

Prin această abordare oarecum inedită pentru un manual de inteligenţă artificială îl invit pe cititor în largirea prin joc a universului său de cunoaştere.

Partea I

Exerciţii

Capitolul 1

Întrebări cu răspuns multiplu

1. **Introducere în inteligenţă artificială**

(a) Când s-a născut inteligenţa artificială? (30s)

- ◯ 1956 - întâlnirea de la Dartmouth
- ◯ 1943 - modelarea creierului cu circuite booleene
- ◯ 1950 - articolul lui Turing *Computing Machinery and Intelligence*
- ◯ 1956 - algoritmul complet al lui Robinson pentru raţionare logică

(b) Testul lui Turing tratează IA din perspectiva (30s)

- ◯ gândirii umane
- ◯ gândirii raţionale
- ◯ comportamentului uman
- ◯ comportamentului raţional

(c) Perspectiva AIMA asupra IA impune agenţilor să (30s)

- ◯ gândească uman
- ◯ gândească raţional
- ◯ acţioneze uman
- ◯ acţioneze raţional

(d) Numele primului calculator care a învins campionul mondial la şah? (30s)

- ◯ Deep Purple
- ◯ Windows Azure
- ◯ Deep Blue
- ◯ Black Friday

(e) Cine nu apare pe coperta AIMA? (30s)

- ◯ Alan Turing (1912–1954)
- ◯ Aristotle (384–322 I.C.),
- ◯ Thomas Bayes (1702–1761)
- ◯ Marvin Minski (1927–2016)

(f) Testul Turing total include (30s)

- ◯ învăţare automată şi viziune computerizată
- ◯ viziune computerizată şi robotică
- ◯ procesarea limbajului natural
- ◯ înţelegerea limbajului şi robotică

(g) Predicţii legate de testul Turing (30s)

- ◯ până în 2000, 30% şanse ca cineva să fie indus în eroare pentru 5 minute
- ◯ până în 2000, 30% şanse ca un expert să fie indus în eroare
- ◯ până în 2000, să fie păcălită orice persoană
- ◯ 20% şanse să fie păcălită orice persoană

(h) Sistemele multi-agent s-au dezvoltat începând cu (30s)

◯ 1989
◯ 2001
◯ 2003
◯ 1995

(i) Ultimul joc la care agentul uman a avut supremaţia a fost (30s)

◯ Şah
◯ Go
◯ Spinner
◯ Game of Thrones

2. **Agenţi**

(a) Raţionalitatea implică (30s)

- ◯ omniscienţă, învăţare, autonomie
- ◯ explorare, învăţare, autonomie
- ◯ culegere de informaţii, învăţare, omniscienţă
- ◯ succes, învăţare, autonomie

(b) PEAS provine de la (30s)

- ◯ Planning, Environment, Actuators, Sensors
- ◯ Performance measure, Environment, Actuators, Search
- ◯ Performance measure, Environment, Agents, Sensors
- ◯ Performance measure, Environment, Actuators, Sensors

(c) Mediul este semi-static dacă (30s)

- ◯ mediul şi funcţia de performanţă se schimbă
- ◯ mediul se schimbă, funcţia de performanţă nu
- ◯ mediul nu se schimbă, funcţia de performantă se schimbă
- ◯ mediul şi funcţia de performanţă nu se schimbă

(d) În medii nondeterminste stărilor succesoare le sunt ataşate probabilităţi (30s)

- ◯ Adevărat
- ◯ Fals

(e) Mediul în jocurile de tip *rebus* este (30s)

- ◯ deterministic şi continuu
- ◯ static şi parţial observabil
- ◯ multi-agent şi secvenţial
- ◯ nicio variantă din cele anterioare

(f) Mediul pentru şah fără ceas este (30s)

- ◯ semi-static şi complet observabil
- ◯ static şi complet observabil
- ◯ continuu şi secvenţial
- ◯ episodic şi multi-agent

(g) Mediul pentru şoferul de taxi este (30s)

- ◯ continuu şi complet observabil
- ◯ dinamic şi complet observabil
- ◯ continuu şi secvenţial
- ◯ episodic şi multi-agent

(h) Un agent reflex bazat pe model menţine o stare interioară (30s)

- ◯ Adevărat
- ◯ Fals

(i) Agenţii bazaţi pe utilitate (30s)

- ◯ utilizează reguli de tip *conditie* → *actiune*
- ◯ aleg acţiunea care îi conduce la cea mai bună utilitate expectată
- ◯ folosesc planificarea pentru a atinge starea ţintă

(j) Un agent care învaţă îşi poate îmbunătăţi (30s)

- ◯ doar cunoştinţele iniţiale
- ◯ doar funcţia de evaluare a performanţei
- ◯ doar generatorul de probleme utilizat în învăţare
- ◯ toate elementele sale

3. **Căutare neinformată**

(a) Dacă agentul nu ştie care este starea lui iniţială, problema de rezolvat este de tipul (30s)

- ◯ stare unică (single state)
- ◯ conformant
- ◯ contingent

(b) Complexitatea în timp a unui algoritm reprezintă (30s)

- ◯ timpul în secunde pentru calcularea soluţiei
- ◯ numărul mediu de noduri în memorie
- ◯ numărul total de noduri generate/expandate
- ◯ numărul maxim de noduri în memorie în timpul căutării

(c) Un algoritm care găseşte întotdeauna soluţia dacă aceasta există este (30s)

- ◯ complet
- ◯ optimal

(d) Care este cea mai bună strategie de căutare pe un calculator cu memorie puţină? (30s)

- ◯ în adâncime
- ◯ în lăţime
- ◯ cost uniform

(e) Căutarea în adâncime găseşte cea mai bună soluţie (30s)

- ◯ adevărat
- ◯ fals

(f) A* este un algoritm de tip greedy ghidat de functia de estimare a costului $f(n) =$ (30s)

- ◯ costul de la nodul n la ţintă
- ◯ costul de la starea iniţială la nodul n
- ◯ maximul dintre costul de la starea iniţială la n şi costul de la n la ţintă
- ◯ nicio variantă din cele enumerate

(g) Care nu este un algoritm de căutare informată? (20s)

- ◯ cost uniform
- ◯ greedy-best search
- ◯ A*

(h) În problema jocului culisant, care nu sunt euristici admisibile? (30s)

- ◯ $h_1 =$ numărul de piese care nu se află în poziţia finală
- ◯ $h_2 =$ suma distanţelor de la fiecare piesă la poziţia ei finală
- ◯ $h_3 = max(h_1, h_2)$
- ◯ $h_4 = sum(h_1, h_2)$

4. **Dincolo de căutarea clasică**

(a) Care dintre următorii algoritmi este complet? (30s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- ◯ first choice hill climbing
- ◯ random-restart hill climbing

(b) Când o stare are mii de stări succesoare, vei utiliza în căutare (60s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- ◯ first choice hill climbing

(c) Algoritmul de *răcire controlată* păstrează în memorie (20s)

- ◯ 1 stare
- ◯ k stări, $k > 1$

(d) Algoritmul de *răcire controlată* porneşte cu (20s)

- ◯ temperatură ridicată
- ◯ temperatură joasă

(e) Algoritmul de *răcire controlată* (30s)

- ◯ nu permite selectarea stărilor cu utilitate mai mică decât cea curentă
- ◯ permite selectarea stărilor mai proaste cu probabilitatea $P = \frac{1}{e^{\frac{\Delta E}{T}}}$
- ◯ permite selectarea stărilor mai proaste cu probabilitatea $P = \frac{1}{e^{T}}$
- ◯ nicio variantă dintre acestea

(f) *Local beam search* cu k stări este similar cu k algoritmi hill climbing rulaţi în paralel. (30s)

- ◯ Adevărat
- ◯ Fals

(g) Care nu este un operator genetic în algoritmii genetici? (30s)

- ◯ ADN
- ◯ mutaţie
- ◯ încrucişare
- ◯ clonare

(h) Într-un nod de tip AND, ramificaţia este introdusă de (20s)

- ◯ alegerile mediului
- ◯ alegerea facută de agent

(i) *Belief state* reprezintă: (30s)

- ◯ setul stărilor posibile în care un agent poate fi
- ◯ spaţiul de căutare al unui agent religios
- ◯ cea mai probabilă stare în care agentul poate fi
- ◯ spaţiul de căutare al unui agent ateist

5. **Căutare adversarială**

(a) Minimax execută în arborele jocului o căutare de tipul (20s)

- ◯ în adâncime
- ◯ în lăţime
- ◯ cost uniform
- ◯ A*

(b) Ordinea de analiză a mutărilor nu afectează eficienţa eliminării nodurilor (pruning) (30s)

- ◯ fals
- ◯ adevărat

(c) Tăierea cu alpha-beta nu afectează rezultatul final (10s)

- ◯ adevărat
- ◯ fals

(d) În practică, pentru evaluarea poziţiei se utilizează o combinaţie de atribute (20s)

- ◯ liniară
- ◯ nonliniară

(e) Căutarea de tip *quiescent* analizează poziţiile interesante la o adâncime mai mare (10s)

- ◯ fals
- ◯ adevărat

(f) Ce se poate utiliza pentru a implementa *tăierea înainte (forward pruning)*? (30s)

- ◯ hill climbing
- ◯ beam search
- ◯ hill climbing stocastic
- ◯ răcire controlată (simulating annealing)

(g) Valoarea expectată a unei poziţii în jocuri stocastice este (30s)

- ◯ media tutoror rezultatelor posibile ale nodurilor de tip *min* şi *max*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *frunză*
- ◯ media tutoror rezultatelor posibile ale nodurilor de tip *şansă*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *şansă*

(h) În jocuri stocastice, comportamenul se prezervă pentru orice transformare a funcţiei de evaluare care este (30s)

- ◯ monotonă
- ◯ nonmonotonă
- ◯ liniar pozitivă
- ◯ liniar negativă

(i) În jocuri stocastice, căutarea la adâncimi mari este esenţială (30s)

- ◯ adevărat
- ◯ fals

6. **Probleme de satisfacere a constrângerilor**

(a) Algoritmul de backtracking poate fi îmbunătăţit prin: (30s)

- ◯ nu poate fi imbunătăţit!
- ◯ sortarea variabilelor în ordine lexicografică
- ◯ detectarea timpurie a eşecului inevitabil
- ◯ exploatând structura problemei

(b) Algoritmul de backtracking poate beneficia de euristici în alegerea (30s)

- ◯ domeniului unei variabile
- ◯ constrângerilor care să fie eliminate
- ◯ variabilei care se instanţiază la pasul următor
- ◯ valorii din domeniu care se asignează variabilei curente

(c) Euristica *valorile minime rămase* selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilor învecinate
- ◯ valorile care introduc cele mai multe constrângeri în variabile învecinate

(d) Euristica grad selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilelor învecinate
- ◯ valorile care introduc cele mai multe constrângeri în variabilele învecinate

(e) Euristica *cele mai puţine constrângeri* selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilelor învecinate
- ◯ valorile cure introduc cele mai multe constrângeri în variabilele învecinate

(f) Pentru a detecta eşecul cât mai repede posibil se utilizează euristica (30s)

- ◯ valorile minime rămase
- ◯ grad
- ◯ cele mai puţine constrângeri

(g) Pentru a reduce factorul de ramificare se utilizează (30s)

- ◯ valorile minime rămase
- ◯ grad
- ◯ cele mai puţine constrângeri

(h) Verificarea înainte (forward checking) evită detectarea târzie a conflictului (20s)

- ◯ adevărat
- ◯ fals

(i) $X, Y, Z \in \{0, 1, 2, 3\}, X < Y < Z$ (30s)

- ◯ $X \in \{0, 1, 2\}$
- ◯ $X \in \{2, 3\}$
- ◯ $X \in \{1, 2\}$
- ◯ $X \in \{0, 1\}$

(j) Bound propagation $D_1 = [0, 5], D_2 = [0, 8], D_1 + D_2 = 10$ (30s)

- ◯ $D_1 = [5, 5], D_2 = [2, 8]$
- ◯ $D_1 = [0, 5], D_2 = [2, 8]$
- ◯ $D_1 = [2, 5], D_2 = [5, 8]$
- ◯ nicio variantă

7. **Agenţi logici**

(a) Lumea lui Wumpus este (30s)

- ◯ complet observabilă şi deterministă
- ◯ episodică şi observabilă local
- ◯ statică şi include un singur agent
- ◯ multi-agent şi secvenţială

(b) Care expresie nu este satisfiabilă? (30s)

- ◯ $\neg a \vee b$
- ◯ $a \rightarrow \neg a$
- ◯ $a \vee \neg a \rightarrow a \wedge \neg a$
- ◯ $a \wedge \neg a \vee b$

(c) $KB \models \alpha$ dacă şi numai dacă (20s)

- ◯ $M(KB) \subseteq M(\alpha)$
- ◯ $M(\alpha) \subseteq M(KB)$

(d) Propoziţia $p \rightarrow q \equiv \neg q \rightarrow \neg p$ este (30s)

- ◯ falsă
- ◯ adevarată
- ◯ depinde doar de p
- ◯ depinde doar de q

(e) Contrapoziţia spune că: (30s)

- ◯ $(a \rightarrow b) \equiv \neg b \rightarrow \neg a$
- ◯ $\neg(\neg a) \equiv a$
- ◯ $\neg(a \wedge b) \equiv (\neg a \vee \neg b)$
- ◯ $\neg(a \vee b) \equiv (\neg a \wedge \neg b)$

(f) Tabelul de adevăr al implicaţiei conţine (10s)

- ◯ 4 de zero
- ◯ 3 de zero
- ◯ 2 de zero
- ◯ 1 singur zero

(g) Care din următoarele propoziţii sunt tautologii? (20s)

- ◯ $a \wedge \neg a$
- ◯ $a \vee \neg a$
- ◯ $a \vee \neg a$
- ◯ $\neg a$

(h) Care din următoarele nu este o clauză Horn? (10s)

- ◯ $a \rightarrow b$
- ◯ $a \rightarrow \neg b$

◯ $a \to b \wedge c$

◯ $a \to b \vee c$

(i) Raţionarea înainte este (30s)

◯ ghidată de ţel

◯ ghidată de date

(j) Care expresie este în forma normal conjunctivă? (10s)

◯ $a \to b$

◯ $a \wedge b$

◯ $\neg a \vee b$

◯ $\neg(a \wedge b)$

(k) Pentru a demonstra α, rezoluţia arată că $KB \wedge \neg\alpha$ este (10s)

◯ nesatisfiabilă

◯ satisfiabilă

8. **Logica de ordinul întâi**

(a) Ce este FOL? (10s)

- ◯ First Order Language
- ◯ First order logic
- ◯ Focus on logic
- ◯ First order (Star Wars)

(b) Predicatul $between(x, y)$ este (20s)

- ◯ unar
- ◯ binar
- ◯ ternar

(c) Logica de ordinul include (20s)

- ◯ obiecte, relaţii, funcţii
- ◯ timp, credinţe, cunoştinţe
- ◯ constrângeri
- ◯ aceeaşi expresivitate cu logica propoziţională

(d) *Fiecăruia îi place îngheţata* este formalizată în FOL (20s)

- ◯ $place(x, inghetata)$
- ◯ $place \wedge inghetata$
- ◯ $\exists x, place(x, inghetata)$
- ◯ $\forall x, place(x, inghetata)$

(e) *Există o persoană care iubeşte pe toată lumea* este formalizată în FOL (20s)

- ◯ $\forall x\ \exists y, iubeste(x, y)$
- ◯ $\exists x\ \forall y, iubeste(x, y)$
- ◯ $\forall x\ iubeste(x, Persoana)$
- ◯ $\exists x\ iubeste(x, ToataLumea)$

(f) *Brothers are siblings* este formalizată în FOL (20s)

- ◯ $\exists x\ \exists y Brother(x, y) \rightarrow Sibling(x, y)$
- ◯ $\forall x\ \exists y Brother(x, y) \wedge Sibling(x, y)$
- ◯ $Brother(x) \rightarrow Sibling(y)$
- ◯ $\forall x\ \forall y Brother(x, y) \rightarrow Sibling(x, y)$

(g) *Oricine de la UTCN este isteţ* este formalizată în FOL (20s)

- ◯ $\forall x\ la(x, UTCN) \wedge istet(x)$
- ◯ $istet(x, TUCN)$
- ◯ $\forall x\ la(x, UTCN) \Rightarrow istet(x)$
- ◯ $\exists x\ la(x, UTCN) \Rightarrow istet(x)$

(h) Relaţia *sibling* este simetrică (20s)

- ◯ $Sibling(x)$

◯ $\forall x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$
◯ $Sibling(x, y) \vee Sibling(y, x)$
◯ $\exists x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$

(i) *O mamă este părintele de sex feminin* (30s)

◯ $\exists x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
◯ $\forall x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
◯ $\forall x, y\ Mother(x, y) \equiv Parent(x, y)$
◯ $\forall x, y\ Mother(x, y) \wedge (Female(x) \wedge Parent(x, y))$

(j) *Oricine iubeşte pe cineva* este formalizată în FOL (20s)

◯ $\forall x \exists y, iubeste(x, y)$
◯ $\exists x \forall y, iubeste(x, y)$
◯ $\forall x \exists y, iubeste(y, x)$
◯ $\exists x \forall y, iubeste(y, x)$

9. **Inferenţă în logica de ordinul întâi**

(a) Un termen *ground* este (20s)

- ◯ un termen cu cel puţin o constantă
- ◯ un termen fără variabile
- ◯ un sol unde un arbore binar poate fi plantat
- ◯ niciuna dintre aceste opţiuni

(b) *Lifting* înseamnă (20s)

- ◯ transformarea din logică propoziţională în logica de ordinul întâi
- ◯ eliminarea constantelor Skolem
- ◯ introducerea constantelor Skolem

(c) Datalog = (10s)

- ◯ Prolog + funcţii
- ◯ baze de date extinse cu reguli
- ◯ clauze în logica de ordinul întâi plus funcţii
- ◯ clauze în logica de ordinul întâi fără funcţii

(d) O bază de date deductivă (20s)

- ◯ utilizează inferenţa pentru a răspunde la interogări
- ◯ combină programarea logică cu baze de date relaţionale
- ◯ mai expresivă decât o bază de date, dar mai puţin decât un limbaj logic

(e) Skolemizarea este procesul de (10s)

- ◯ eliminare a cuantificatorilor universali
- ◯ eliminare a cuantificatorilor existenţiali
- ◯ introducere a cuantificatorilor universali
- ◯ introducere a cuantificatorilor existenţiali

(f) Logica de ordinul întâi a fost inventată de: (10s)

- ◯ Aristotel (384-322 BC)
- ◯ Boole (1815-1864)
- ◯ Ludwig Wittgenstein (1889-1951)
- ◯ Gottlob Frege (1848-1925)

(g) În Prolog nu există nicio modalitate de a aserta un fapt negativ (e.g. $\neg father(a, b)$) (10s)

- ◯ Fals
- ◯ Adevărat

(h) În forma normal conjunctivă, fiecare clauză conţinută este o (20s)

- ◯ disjuncţie de literari
- ◯ conjuncţie de literari

(i) Care nu este o metodă de inferenţă logică? (10s)

- ◯ raţionarea înainte
- ◯ raţionarea înapoi
- ◯ skolemizarea
- ◯ rezoluţia

(j) Utilizări practice ale demonstratoarelor de teoreme includ (20s)

- ◯ verificare hardware
- ◯ verificare software

10. **Planificare clasică**

(a) PDDL provine de la (10s)

- ◯ Planning Domain Description Language
- ◯ Planning Description Domain Language
- ◯ Planning Domain Definition Language
- ◯ nicio variantă din cele de mai sus

(b) O condiţie deschisă este o precondiţie (10s)

- ◯ cu un termen care nu are toate variabilele instanţiate
- ◯ cu prezumţia de lume deschisă
- ◯ a unui pas fără legături cauzale

(c) Un *clobber* (20s)

- ◯ validează o precondiţie obţinută datorită unui efect al unei acţiuni
- ◯ distruge o precondiţie obţinută printr-o legătură cauzală
- ◯ introduce o relaţie de ordine parţială între acţiuni

(d) Anomalia Sussman (20s)

- ◯ ilustrează dezavantajele planificării greedy
- ◯ opreşte algoritmul de planificare, chiar dacă un plan există.
- ◯ poate fi rezolvată cu ajutorul *clobberilor*

(e) O acţiune este relevantă dacă (20s)

- ◯ poate fi următorul pas într-un plan
- ◯ cel puţin unul din efecte se unifică cu un element din starea finală
- ◯ are toate precondiţiile satisfăcute

(f) Care sunt euristici admisibile pentru planificare? (20s)

- ◯ h_1= ignorarea precondiţiilor
- ◯ h_2= ignorarea stării iniţiale
- ◯ h_3 = ignorarea efectelor care elimină fapte din starea curentă
- ◯ h_4= ignorarea stării finale

(g) Ce se elimină pentru h= *Manhattan* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

- ◯ $blank(s2)$
- ◯ $adj(s1, s2)$
- ◯ $on(t, s1)$
- ◯ toate precondiţiile

(h) Ce trebuie eliminat pentru h = *numărul de piese plasate greşit* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

- ◯ $blank(s2), on(t, s1)$
- ◯ $blank(s2), adj(s1, s2)$
- ◯ $on(t, s1), tile(t)$
- ◯ toate precondiţiile

11. **Planificare în lumea reală**

(a) Complexitatea algoritmului căii critice este (20s)

- ◯ $O(n)$
- ◯ $O(1)$
- ◯ $O(Nb)$
- ◯ $O(N^b)$

(b) $ES(B) =$ (30s)

- ◯ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- ◯ $min_{B \succ A} LS(B) - Duration(A)$

(c) $LS(A) =$ (30s)

- ◯ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- ◯ $min_{B \succ A} LS(B) - Duration(A)$

(d) Euristica *minimum slack* (20s)

- ◯ este similară heuristicii *numărul de valori minime* specifică CSP
- ◯ garantează obţinerea soluţiei optime
- ◯ nicio variantă din cele de mai sus

(e) Este mai bine să avem efecte condiţionate decât o acţiune care nu se poate aplica (20s)

- ◯ întotdeauna
- ◯ niciodată
- ◯ în cazul planificării fără senzori
- ◯ în cazul planificării contingente

(f) Schemele de percepte sunt utilizate în (20s)

- ◯ planificarea fără senzori
- ◯ planificarea contingentă
- ◯ atât planificarea fără senzori, cât şi cea contingentă
- ◯ nicio opţiune din cele anterioare

(g) Pentru identificarea succesului accidental (serendipity) agentul monitorizează din mediu: (20s)

- ◯ acţiunile
- ◯ planul
- ◯ ţelul

(h) La pierderea semnalului GPS, navigatorul e mai bine să execute (20s)

- ◯ replanificare

- ◯ repararea planului
- ◯ depinde de lungimea planului şi perioadă de pierdere a semnalului
- ◯ se opreşte şi asteaptă instrucţiuni de la agentul uman

(i) Acţiunile concurente sunt necesare în cazul (20s)

- ◯ monitorizării planului
- ◯ planificării cu contingenţe
- ◯ planificării multi-agent
- ◯ planificării online

(j) În cazul informaţiilor incomplete se utilizează (20s)

- ◯ planificarea condiţională
- ◯ planificarea cu contingeţe
- ◯ planificarea fără senzori
- ◯ monitorizarea execuţiei şi replanificare

(k) În cazul informaţiilor eronate se utilizează (20s)

- ◯ planificarea conditională
- ◯ planificarea contingentă
- ◯ planificarea fără senzori
- ◯ monitorizarea execuţiei şi replanificare

12. **Calculul evenimentelor**

(a) Care nu este o sarcină de raţionare în calculul evenimentelor? (30s)

- ◯ postdicţia
- ◯ rezoluţia
- ◯ abducţia
- ◯ predicţia

(b) Se poate face planificare în calculul evenimentelor prin (30s)

- ◯ predicţie
- ◯ abducţie
- ◯ postdicţie
- ◯ doar limbajul PDDL poate fi utilizat pentru planificare

(c) $\neg HoldsAt(Awake(Nathan), 0)$ este (30s)

- ◯ o observaţie
- ◯ axiomă cu efect pozitiv
- ◯ axiomă cu efect negativ
- ◯ apariţia unui eveniment

(d) $Happens(Awake(Nathan), 1)$ (30s)

- ◯ o observaţie
- ◯ axiomă cu efect pozitiv
- ◯ axiomă cu efect negativ
- ◯ apariţia unui eveniment

(e) $HoldsAt(Ocupa(p_1, s), t) \wedge HoldsAt(Occupies(p_2, s), t) \rightarrow p_1 = p_2$ (30s)

- ◯ *Ocupa* este o relaţie injectivă
- ◯ *Ocupa* este o relaţie surjectivă

(f) $\neg HoldsAt(On(o, o), t)$ (30s)

- ◯ *On* este o relaţie reflexivă
- ◯ *On* este o relaţie ireflexivă

(g) $\neg HoldsAt(Ringing(p, p), t)$ (30s)

- ◯ un telefon nu poate suna un alt telefon
- ◯ un telefon nu se poate suna pe el însuşi
- ◯ telefonul p nu sună la timpul t

(h) $\neg HoldsAt(Broken(d), t) \rightarrow Initiates(TurnOn(a, d), On(d), t)$ (30s)

- ◯ precondiţie de tip fluent
- ◯ precondiţie de tip acţiune

(i) $Happens(WalkThroughDoor(a, d), t) \rightarrow HoldsAt(Near(a, d), t)$ (30s)

- ◯ precondiţie de tip fluent
- ◯ precondiţie de tip acţiune

13. **Probabilităţi condiţionale**

	Joacă fotbal	Nu joacă	Total
Băieţi	42	33	75
Fete	12	23	35
Total	54	56	110

(a) Care este probabilitatea ca o persoană selectată aleator să fie baiat? (20s)

- ◯ 42/110
- ◯ 54/110
- ◯ 75/110
- ◯ 42/75

(b) $P(baiat|joacaFotbal) =$ (20s)

- ◯ 42/110
- ◯ 42/54
- ◯ 42/75
- ◯ 54/110

(c) $P(baiat \wedge joacaFotbal)=$ (20s)

- ◯ 42/110
- ◯ 42/54
- ◯ 42/75
- ◯ 54/110

(d) $P(baiat \vee joacaFotbal)=$ (20s)

- ◯ 42/110
- ◯ 54/110
- ◯ 75/110
- ◯ 87/110

(e) Sunt variabilele *baiat* şi *joacaFotbal* independente? (60s)

- ◯ Da
- ◯ Nu

(f) Care este probabilitatea de a extrage o carte roşie sau un as dintr-un pachet? (20s)

- ◯ 16/50
- ◯ 1/2
- ◯ 15/26
- ◯ 7/13

(g) Care este probabilitatea de a alege o vocală din cuvintul ARTIFICIAL? (20s)

- ◯ 1/5
- ◯ 3/10

◯ 1/10
◯ 1/2

(h) Care este probabilitatea de o obţine un miltiplu de 3 dintr-o aruncare cu zarul şi stema unei monede? (20s)

◯ 1/6
◯ 2/6
◯ 1/4
◯ 33%

14. **Reţele Bayesiene**

(a) $P(A) = \sum P(A, B)$ este (20s)

- ◯ formula lui Bayes
- ◯ marginalizare
- ◯ chain rule
- ◯ joint probability

Marginalizarea reprezintă însumarea pe toate valorile posibile ale variabile noi introduse B.

(b) Formula lui Bayes este (20s)

- ◯ $P(A, B) = P(A)P(B)$
- ◯ $P(B|A) = P(A|B) * P(B)/P(A)$
- ◯ $P(A) = \sum P(A, B)$
- ◯ $P(B|A) = (P(A|B) * P(A))/P(B)$

(c) Pătura lui Markov (20s)

- ◯ nodurile sunt independente fiind daţi părinţii + copiii + copiii părinţilor
- ◯ este echivalenta cu tabelul complet al probabilităţilor
- ◯ nodurile sunt independente de nondescendenţi, fiind daţi părinţii

(d) Noisy-OR consideră (20s)

- ◯ toate cauzele posibile sunt incluse
- ◯ probabilităţile de eşec ale fiecărei clauze sunt independente
- ◯ nicio variantă din cele anterioare

(e) Nod de tip rezidual (leak node) este (20s)

- ◯ o pagină pe Wikileaks
- ◯ un nod probabilistic dintr-o reţea Bayesiană
- ◯ acoperă toate cauzele de tipul *şi altele* într-o relaţie Noisy-OR
- ◯ un nod condiţional independent faţă de rădăcina reţelei Bayesiene.

(f) O reţea Bayesiană hibridă (20s)

- ◯ este o reţea bayesiană modificată de un algoritm genetic
- ◯ conţine atât variabile aleatoare booleene, cât şi nonbooleene
- ◯ conţine atât variabile discrete, cıat şi continue
- ◯ modelează cel puţin două domenii

15. **Inferenţă în reţele Bayesiene**

(a) Care nu este un algoritm de inferenţă exactă? (20s)

- ◯ inferenţă prin enumerare
- ◯ inferenţă prin eliminarea variabilelor
- ◯ inferenţă prin simulare stocastică
- ◯ inferenţă prin Markov Chain Monte Carlo

(b) Care sunt variabilele ascunse din întrebarea $P(A|j,m)$ în scenariul alarmei? (20s)

- ◯ j,m
- ◯ b,e
- ◯ a
- ◯ j,a,m

(c) Care sunt variabilele evidentă din întrebarea $P(A|j,m)$ în scenariul alarmei? (20s)

- ◯ j,m
- ◯ b,e
- ◯ a
- ◯ j,a,m

(d) Care algoritm foloseşte factori? (20s)

- ◯ inferenţă prin enumerare
- ◯ inferenţă prin eliminarea variabilelor
- ◯ inferenţă prin simulare stocastică
- ◯ inferenţă prin Markov Chain Monte Carlo

(e) Metoda *Rejection Sampling* generează doar evenimente consistente cu evidenţele e (20s)

- ◯ Adevărat
- ◯ Fals

Eşantionarea prin revocare (rejection sampling) este o metodă de raţionare aproximativă în reţele Bayesiene. Metoda generează eşantione conform distribuţiei de probabilitate specificate. Dintre aceste eşantioane le păstrează doar pe cele consistente cu evidenţele date. Probabilitatea condiţionată este obţinută prin numărarea cazurilor favorabile din eşantionul rămas.

(f) Metoda *MCMC* generează eşantioane integrale (20s)

- ◯ Fals
- ◯ Adevărat

(g) Algoritmul *d-separare* păstrează (20s)

- ◯ toate variabilele menţionate în interogare şi toţi părinţii acestora
- ◯ toate variabilele menţionate în interogare şi toţi ancestorii acestora
- ◯ toate variabilele menţionate în interogare şi toţi copiii acestora
- ◯ toate variabilele menţionate în interogare şi toţi descendenţii acestora

(h) Metoda *likelihood weighting* (20s)

- ◯ foloseşte evidenţele pentru a pondera eşantionale generate
- ◯ generează eşantioane dintr-o reţea goală
- ◯ respinge eşantioanele inconsistente cu evidenţele

16. **Raţionare probabilistică cu timp**

(a) În asumpţia Markov, starea curentă depinde de (20s)

- ◯ starea anterioară
- ◯ un număr finit şi fix de stări anterioare
- ◯ de părinţi + copii + copiii părinţilor

(b) Un proces staţionar este un (20s)

- ◯ proces în schimbare, dar legile acestei schimbări nu se modifică
- ◯ proces static (i.e. starea nu se schimbă)
- ◯ niciuna dintre aceste variante

(c) Asumpţia Markov a senzorilor (60s)

- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t, E_{0:t-1})$
- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{0:t})$
- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$
- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{t-1})$

(d) Filtrarea sau estimarea stării curente este formalizată cu (30s)

- ◯ $P(X_t|e_{1:t})$
- ◯ $P(X_{t+k}|e_{1:t})$
- ◯ $P(X_k|e_{1:t}), k < t$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

(e) Netezirea (smoothing) este formalizată cu (30s)

- ◯ $P(X_t|e_{1:t})$
- ◯ $P(X_{t+k}|e_{1:t})$
- ◯ $P(X_k|e_{1:t}), k < t$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

(f) Predicţia este formalizată cu (30s)

- ◯ $P(X_t|e_{1:t})$
- ◯ $P(X_{t+k}|e_{1:t})$
- ◯ $P(X_k|e_{1:t}), k < t$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

(g) Identificarea celei mai probabile explicaţii (30s)

- ◯ $argmax(X_{1:t})P(X_t|e_{1:t})$
- ◯ $argmax(X_{1:t})P(X_k|e_{1:t}),\ k < t$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_t)$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

(h) Care din următoarele metode nu este o tehnică de raţionare în modele temporale? (20s)

- ◯ rezoluţia
- ◯ predicţia

- ◯ netezirea
- ◯ identificarea celei mai probabile explicaţii

(i) Pentru a putea raţiona într-o reţea Bayesiană dinamică nu e nevoie de (20s)

- ◯ modelul stării iniţiale
- ◯ modelul de tranziţie
- ◯ modelul senzorilor
- ◯ distribuţia Gaussiană

(j) Învăţarea unei reţele Bayesiene dinamice se referă la (60s)

- ◯ a învăţa din observaţii modelul de tranziţii
- ◯ a interpreta orice nod din reţea ca un neuron
- ◯ estimarea stării, filtrare şi netezire
- ◯ a învăţa din observaţii modelul senzorilor

17. **Decizii simple**

(a) Micromortul este (20s)

- ◯ unitatea de măsură pentru utilitate
- ◯ 1/1,000,000 şanse de deces
- ◯ măsura pentru metrica QUALY (Quality Adjusted Life Years)
- ◯ identic cu microprobabilitatea

(b) Sistemul de asigurări are la bază faptul că agenţii umani sunt faţă de risc (30s)

- ◯ neutri
- ◯ împotrivă
- ◯ în căutarea riscului

(c) Funcţia de utilitate a banilor este (30s)

- ◯ liniar crescătoare
- ◯ constantă
- ◯ logaritmică
- ◯ exponenţială

(d) La ce este utilizat principiul MEU? (30s)

- ◯ Minimize Expected Utility
- ◯ calcularea utilităţii
- ◯ selectarea acţiunii în starea curentă
- ◯ explicarea comportamentului iraţional

(e) Cum se doreşte să acţioneze agenţii software (perspectiva AIMA)? (30s)

- ◯ deterministic
- ◯ raţional
- ◯ cât mai uman
- ◯ rapid

(f) Cu ce se extinde o reţea Bayesiană pentru a deveni o reţea de decizie? (30s)

- ◯ noduri de tip acţiune şi utilitate
- ◯ noduri utilitate
- ◯ noduri MEU
- ◯ Cu nimic. O reţea Bayesiană este deja o reţea de decizie

(g) Valoarea expectată a informaţiei este (30s)

- ◯ negativă
- ◯ aditivă
- ◯ depinde de ordinea în care sunt primite informaţiile
- ◯ nicio versiune din cele anterioare

(h) Eroare cognitivă în care deciziile sunt afectate de o informaţia iniţială (30s)

- ◯ efectul certitudinii

◯ încadrarea (framing)
◯ efectul de ancoră

(i) Eroare cognitivă în care deciziile sunt afectate de modul în care opţiunile sunt prezentate: pozitiv sau negativ (30s)
◯ efectul certitudinii
◯ încadrarea (framing)
◯ efectul de ancoră

(j) Teoria deciziilor este o teorie (10s)
◯ normativă
◯ descriptivă

18. **Decizii secvenţiale**

(a) Problemele de decizii secvenţiale (SDP) (30s)

- ◯ includ utilităţi, incertitudine şi captarea informaţiilor din mediu
- ◯ utilitatea agentului depinde de o secvenţă de decizii
- ◯ sunt cazuri particulare ale problemelor de planificare
- ◯ sunt cazuri particulare de probleme de decizie Markoviene

(b) Problemele de decizie Markoviene sunt probleme secvenţiale pentru mediile (60s)

- ◯ complet observabile şi stocastice
- ◯ complet observabile şi deterministe
- ◯ cu modele de tranziţie Markoviene şi funcţii aditive de recompensă
- ◯ parţial observabile şi stocastice

(c) O politică optimă $\pi(s)$ (30s)

- ◯ minimizează suma expectată a recompenselor
- ◯ specifică cea mai bună acţiune pentru fiecare stare s
- ◯ are cea mai mare valoare expectată a utilităţii

(d) O politică optimală pentru un orizont finit este (20s)

- ◯ nonstaţionară
- ◯ staţionară

(e) Dacă factorul de discount tinde spre 0, atunci recompensele pe termen lung sunt (30s)

- ◯ nesemnificative
- ◯ semnificative

(f) Ecuaţia lui Bellman este (60s)

- ◯ $U(s) = R(s) + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s)$
- ◯ $U(s) = R(s) + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s')$
- ◯ $U(s) = R(s) + \gamma max_{a \in A(s)} \sum_{s'} P(s'|a, s)U(s')$
- ◯ $U(s) = R(s') + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s')$

(g) Care din următoarele licitaţii au caracter public şi sunt descendente? (30s)

- ◯ englezeşti
- ◯ olandeze
- ◯ cu plic inchis
- ◯ Vickrey

(h) O soluţie este Pareto optimală dacă (30s)

- ◯ nu există o altă soluţie pe care unul dintre agenţi o preferă
- ◯ nu există o altă soluţie pe care toţi agenţii să o prefere
- ◯ există o altă soluţie pe care unul dintre agenţi o preferă
- ◯ există o altă soluţie pe care toţi agenţii să o prefere

19. **Arbori de decizie**

(a) Dacă predicatul ţintă are un set finit de valori, atunci problema de învăţare este (10s)

- ◯ regresie
- ◯ clasificare

(b) Principiul *Ockham's razor* preferă (20s)

- ◯ cea mai buna ipoteză consistentă cu datele
- ◯ cea mai simplă ipoteză consistentă cu datele
- ◯ $X^2 + 1/3$ faţă de X^3

(c) O problemă de învăţare este realizabilă dacă (30s)

- ◯ spaţiul ipotezelor conţine funcţia adevărată
- ◯ datele sunt consistente şi fără zgomote
- ◯ se aplică un algoritm de învăţare potrivit pentru date

(d) Arborele de decizie este construit pe baza unei strategii de căutare de tipul (30s)

- ◯ căutare în adâncime
- ◯ căutare în lăţime
- ◯ greedy şi desparte-şi-cucereşte
- ◯ aleator

(e) Dacă s-au analizat toate atributele, dar rămân exemple neclasificate atunci (60s)

- ◯ există erori sau zgomote în datele de antrenament
- ◯ domeniul în care se aplică învăţarea nu este deterministic
- ◯ nu există acces la un atribut prin care s-ar putea distinge între instanţe
- ◯ nici una din opţiunile anterioare

(f) Mai multă informaţie înseamnă (10s)

- ◯ mai multă entropie
- ◯ mai puţină entropie

(g) Suprapotrivirea (overfitting) este (10s)

- ◯ de dorit
- ◯ de evitat

(h) Suprapotrivirea (overfitting) (20s)

- ◯ violează principiul lamei lui Ockham
- ◯ este sinonimă cu supraantrenarea
- ◯ include mai mulţi parametri decât cei necesari
- ◯ toate variantele anterioare

(i) Validarea de tip hold-out nu utilizează în antrenare toate datele disponibile. (20s)

- ◯ Adevărat
- ◯ Fals

(j) Performanţa învăţării este dată de acurateţea prezicerii pe setul de (30s)

- ◯ antrenare
- ◯ validare
- ◯ testare
- ◯ media acurateţii pe cele 3 seturi

(k) *Peeking* apare dacă (60s)

- ◯ ipoteza se selectează şi pe baza erorii pe setul de testare
- ◯ nu se utilizează set pentru validare
- ◯ curba de învăţare (ROC) este un *graf vesel*
- ◯ atributul ţintă este continuu

20. **Cunoştinţe în învaţăre**

(a) Care este diferenţa dintre regresie şi clasificare? (30s)

- ◯ natura atributului ţintă
- ◯ o metodă este supervizată, iar cealaltă nesupervizată
- ◯ o metodă utilizează statistica, iar cealaltă nu
- ◯ nu e nicio diferenţă

(b) Dacă ipoteza clasifică o instanţă ca negativă dar corect ar fi pozitivă, avem (20s)

- ◯ fals pozitiv
- ◯ fals negativ
- ◯ adevărat pozitiv
- ◯ adevărat negativ

(c) Instanţele fals pozitive sunt rezolvate prin (30s)

- ◯ generalizare
- ◯ specializare

(d) Generalizarea are loc prin (30s)

- ◯ adăugarea de condiţii
- ◯ eliminarea unor condiţii

(e) Ce nu este adevărat legat de căutarea current-best-hypothesis? (30s)

- ◯ menţine o singură ipoteză care se ajustează cu fiecare exemplu
- ◯ este deterministică
- ◯ modificarea ipotezei impune verificarea instanţelor anterioare
- ◯ execută mult backtracking

(f) Care nu este o condiţie de stop a algoritmului *version space*? (60s)

- ◯ rămâne doar o ipoteză în spaţiul soluţiilor
- ◯ spaţiul soluţiilor colapsează ($\mathcal{G}$ sau $\mathcal{S}$ devin vide)
- ◯ rămân mai multe ipoteze şi nu mai există instanţe de analizat
- ◯ complexitatea ipotezei este mai mare decât un prag

(g) Algoritmul FOIL evită suprapotrivirea (overfitting) prin (30s)

- ◯ validare încrucişată
- ◯ penalizând ipotezele mari
- ◯ folosind cunoştinţe din domeniu

(h) Învăţarea bazată pe relevanţă (30s)

- ◯ extrage reguli generale dintr-un singur exemplu
- ◯ foloseşte cunoştinţe anterioare

(i) Ce nu este adevărat la *version space*? (30s)

- ◯ este o reprezentare hierahică a cunoştinţelor

◯ valorile atributelor sunt continue
◯ utilizează arbori de generalizări şi specializări
◯ presupune ca datele sunt corecte

21. **Reţele neurale artificiale**

(a) Corespondenţele pentru ⟨neuron, sinapse, dendrite, axon⟩ sunt: (30s)

- ◯ ⟨ponderi, nod, intrare, ieşire⟩
- ◯ ⟨nod, ponderi, intrare, ieşire⟩
- ◯ ⟨intrare, nod 3, ponderi 4, ieşire⟩
- ◯ ⟨intrare, ponderi, nod, ieşire⟩

(b) 20 Reţelele neurale sunt metode de învăţare

- ◯ supervizată
- ◯ nesupervisaă
- ◯ reinforsată
- ◯ depinde de tipul reţelei neurale

(c) Neuronul cu $w_0 = 0.5$, $w_1 = 1$ şi $w_2 = 1$ implementează poarta logică (60s)

- ◯ and
- ◯ or
- ◯ xor
- ◯ not

(d) Neuronul cu $w_1 = 1$ şi $w_2 = 1$ implementează poarta logică AND dacă (60s)

- ◯ $w_0 = 0.5$
- ◯ $w_0 = -0.5$
- ◯ $w_0 = 1$
- ◯ $w_0 = 1.5$

(e) Perceptronul nu poate modela funcţia XOR (20s)

- ◯ Adevărat
- ◯ Fals

(f) Învăţarea are loc prin ajustarea ponderilor pentru a reduce eroarea pe setul de test (20s)

- ◯ Adevărat
- ◯ Fals

(g) Ce aţi răspunde unui medic legat de similaritatea dintre creier şi ANN? (20s)

- ◯ sunt similare
- ◯ nu diferite

(h) SVM crează un clasificator (20s)

- ◯ neliniar
- ◯ liniar

(i) Numărul vectorilor suport este mai mare decât setul de antrenament (20s)

- ◯ Adevărat
- ◯ Fals

(j) *Boosting* este (20s)

- ◯ o metodă de învăţare cu ansambluri de clasificatori
- ◯ un caz particular de arbore de decizie
- ◯ o metodă de eficientizare a învăţării prin paralelizare

(k) Învăţarea cu ansambluri poate utiliza clasificatori diferiţi (e.g. ANN, SVM, DT) (20s)

- ◯ Adevărat
- ◯ Fals

22. **Clusterizare. K-means. Clusterizare ierarhică**

(a) K-means este un algoritm de (10s)

- ◯ clusterizare herarhică
- ◯ clusterizare prin partiţionare

(b) K-means are nevoie de date normalizate şi nenominale (20s)

- ◯ True
- ◯ False

(c) În suma erorilor pătratice (SSE) pentru fiecare instanţă se (30s)

- ◯ însumează distanţa până la cel mai apropiat centroid
- ◯ ridică la patrat distanţa până la cel mai apropiat centroid şi se însumează rezultatele
- ◯ însumează distanţele la toate celelalte puncte din cluster
- ◯ ridică la patrat distanţele faţă de toate celelalte puncte din cluster şi se însumează rezultatele.

(d) Dacă K creşte atunci SSE (10s)

- ◯ scade
- ◯ creşte

(e) Dacă setul de date conţine 3 grupuri reale, care sunt şansele ca algoritmul K-means să genereze aleator câte un centroid în fiecare cluster? (60s)

- ◯ rulări multiple
- ◯ Bisecting K-means
- ◯ clusterizare ierarhică
- ◯ preprocesare

(f) Care nu este o soluţie la problema iniţializării algoritmului K-means? (30s)

- ◯ rulări multiple
- ◯ Bisecting K-means
- ◯ clusterizăre ierarhică
- ◯ preprocesare

(g) Care din următorele nu este o metrică pentru distanţa dintre clustere? (30s)

- ◯ min (single link)
- ◯ media grupului
- ◯ dendograma
- ◯ distanţa dintre centroizi

(h) Care metodă de clusterizare nu are nevoie de predefinirea numărului de clustere? (20s)

- ◯ Bisecting K-means
- ◯ K-means
- ◯ clusterizare hierarhică

◯ toate menţionate anterior

(i) Clusterizarea ierarhică cu metrica min (single link) (30)

◯ nu este susceptibilă la excepţii
◯ nu e sensibilă la zgomote
◯ nicio varianta din cele anterioare

(j) Clusterizarea ierarhică cu metrica max (complete link) (30)

◯ tinde să spargă grupurile mari,
◯ tinde să genereze clustere de formă globulară
◯ susceptibil la zgomote
◯ susceptibil la excepţii

23. **Reguli de asociaţie. Algoritmul Apriori**

(a) Este suportul o măsură simetrică? (10s)

- ◯ nu
- ◯ da

(b) Este confidenţa o măsură simetrică? (10s)

- ◯ nu
- ◯ da
- ◯ poate

(c) Câte seturi de itemi pot fi construite cu d itemi unici? (20s)

- ◯ 2^d
- ◯ d^2
- ◯ $d!$
- ◯ $(d-1)!$

(d) Câte reguli pot fi generate cu d itemi unici? (30s)

- ◯ $3^{d+1} - 2^d + 1$
- ◯ $3^{d+1} + 2^{d+1}$
- ◯ $3^{d+1} - 2^{d+1} + 1$
- ◯ $3^{d+1} + 2^{d+1}$

(e) Suportul pentru un set de itemi nu depăşeşte suportul submulţimilor sale. (20s)

- ◯ fals
- ◯ adevărat

(f) $\forall X, Y \ X \subseteq Y \rightarrow s(X) \leq s(Y)$ (20s)

- ◯ adevărat
- ◯ fals

(g) Câte reguli candidate pot fi generate dintr-o mulţime de itemi frecvenţî L ($|L| = k$)? (30s)

- ◯ 2^k
- ◯ 2^{k-2}
- ◯ $2^k - 2$
- ◯ $k!$

(h) Confidenţa este anti-monotonă cu privire la numărul de itemi din partea dreaptă (20s)

- ◯ adevărat
- ◯ fals

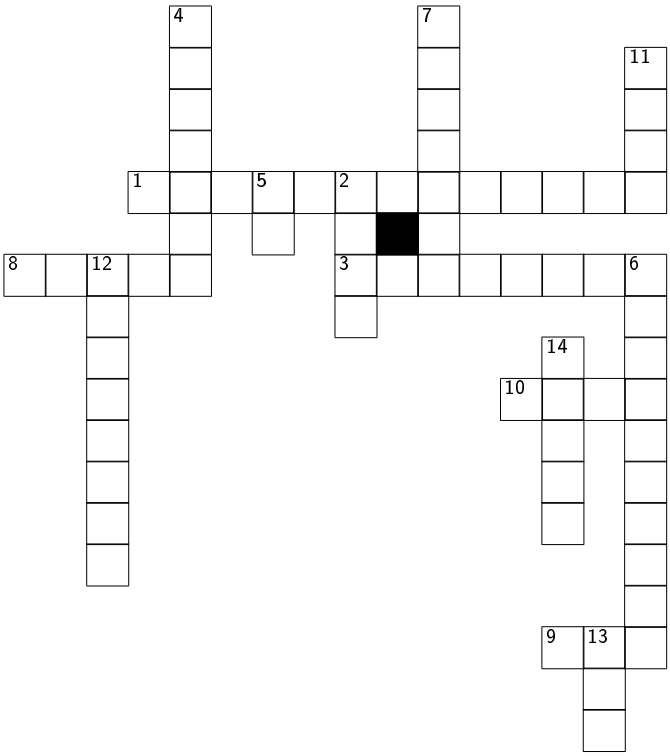
(i) Metoda $F_{k-1} \times F_{k-1}$: Care pereche nu se va agrega? (30s)

- ◯ ABC, ABD
- ◯ ABD, ABE
- ◯ ABC, ACD
- ◯ ABC, ABE

Capitolul 2

Rebusuri în inteligenţă artificială

Introducere în inteligenţă artificială



Orizontal

1 Când IA depăşeşte inteligenţa umană.

3 Formalizat de Aristotel

8 Vechi sistem expert în medicină

9 Mulţi agenţi

10 Vechi sistem multiagent cognitiv

Vertical

2 Limbaj cu paranteze

4 Veche metodă pentru actualizarea ponderilor între neuroni

5 Algoritmi Genetici! (en)

6 Lumile lui Minsky (en)

7 Vechi sistem expert în chimie

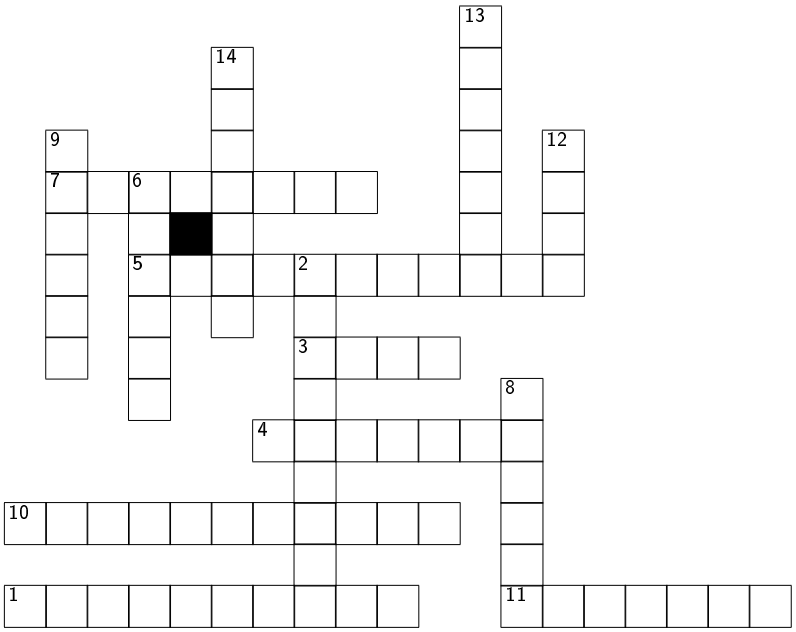
11 Inventatorul primului computer operaţional programabil

12 Turing părea mic în preajma lui

13 Primul calculator electronic

14 Testul lui Turing extins cu viziune computerizată şi robotică

Agenţi inteligenţi



Orizontal

1 Agent care ştie tot ce se va întampla

3 Descrie performanţa, acţiunile, senzorii agentului şi caracterizează mediul (abr)

4 Agent software

5 Opuse jocurilor competitive

7 Mediul în care acţiunile nu depind de stările anterioare

10 Mediul nu se schimbă dar performanţa agentului scade în timpul deliberării

11 Agent conversaţional

Vertical

2 Colectare de informaţie despre mediu

6 Mediu incomplet observabil sau nondeterministic

8 Mediul nu se schimbă în timp ce agentul deliberează

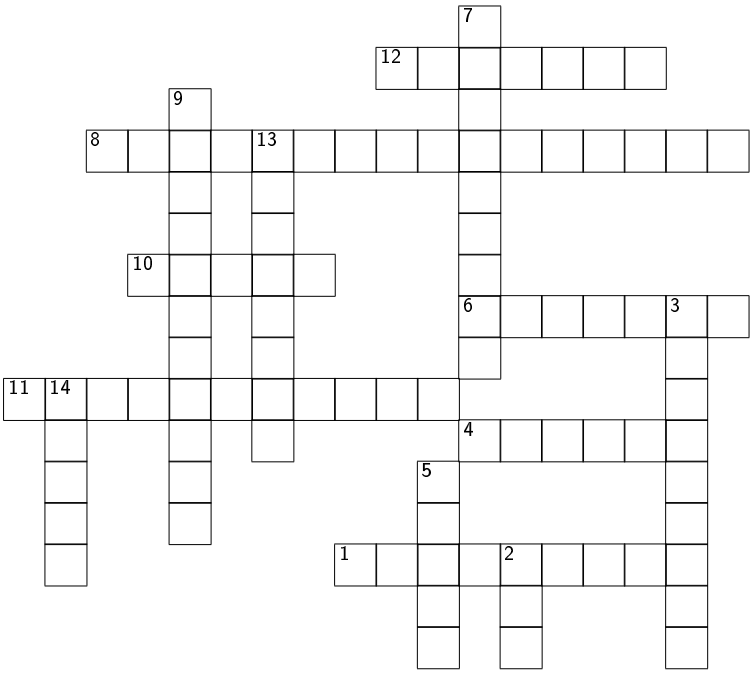
9 Cel mai simplu agent

12 Sistem multi-agent în Java

13 Utili în percepţie

14 Agent care se descurcă singur

Rezolvarea problemelor prin căutare



Orizontal

1 Stochează nodurile frunză disponibile pentru expandare

4 Cea mai bună soluţie

6 Algoritm care găseşte soluţia dacă aceasta există

8 Căutare în două sensuri

10 Algoritm informat

11 Euristici monotone

12 Taiere de arbori de căutare (en)

Vertical

2 Problema comis voiajorului (abr, en)

3 Aplicarea acţiunilor legale pe starea curentă

5 Lista cu noduri explorate (en)

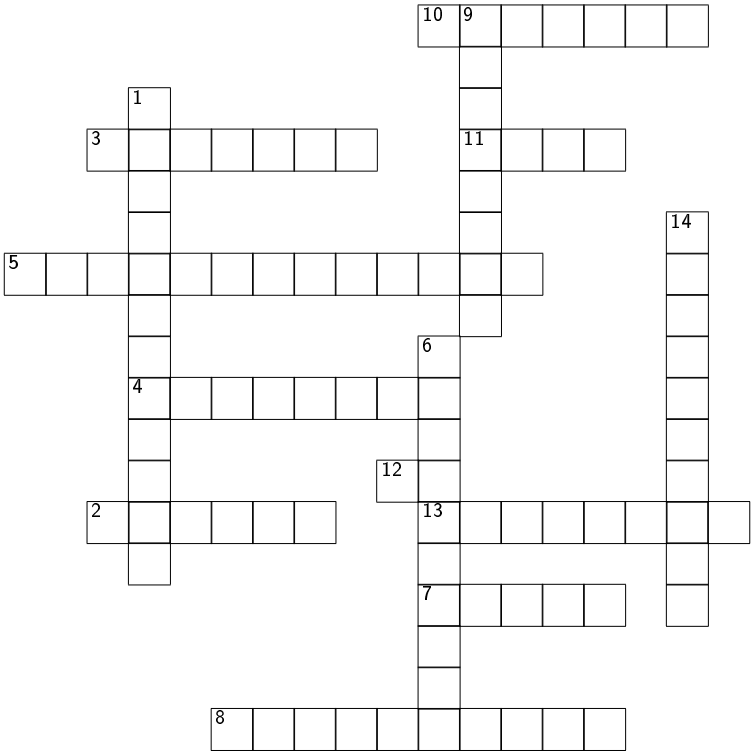
7 Utile în căutarea informată

9 Euristică optimistă

13 Problemă cu mai puţine restricţii

14 Căutare neinformată

Căutare locală



Orizontal

2 Căutare în mediu necunoscut (en)

3 Funcţie care evaluează indivizi (en)

4 Operator genetic

5 Operator genetic

7 Arbore de căutare pentru acţiuni nondeterministe

8 Problemă fără senzori (en)

10 Căutare în care mediul este cunoscut

11 Căutare cu stări interzise

12 Algoritm genetic în care indivizii sunt programe

13 Graf în care toate nodurile au număr egal de muchii de intrare şi ieşire

Vertical

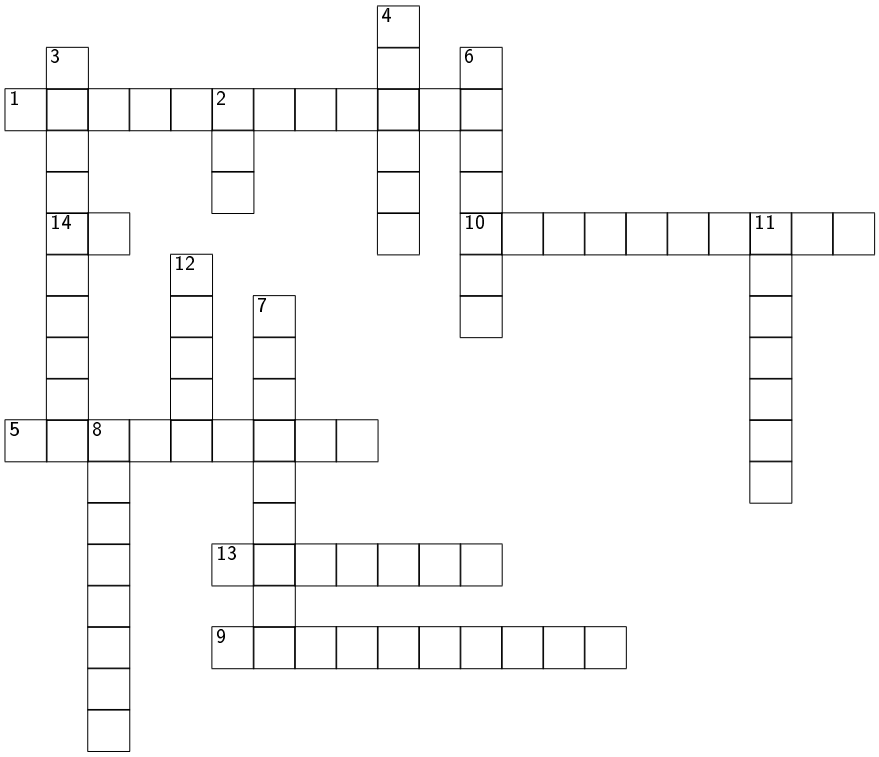
1 Local beam search cu $k = 1$

6 Scade în *simulating annealing*

9 Estimarea stării curente

14 Algoritm genetic cu un singur individ în populaţie

Căutare adversarială



Orizontal

1 Permutaţii ale mutărilor care conduc în aceeaşi stare

5 Tehnică de tăiere a arborilor

9 Jocuri cu elemente aleatoare

10 Simulare pentru evaluarea poziţiei

13 Jocul Othello

14 Joc popular în Asia cu factor mare de ramificaţie

Vertical

2 O jumătate de mutare

3 Şah cu tabla parţial observabilă

4 Test care decide când se aplică funcţia de evaluare a poziţiei (en)

6 Algoritm de căutare adversarială

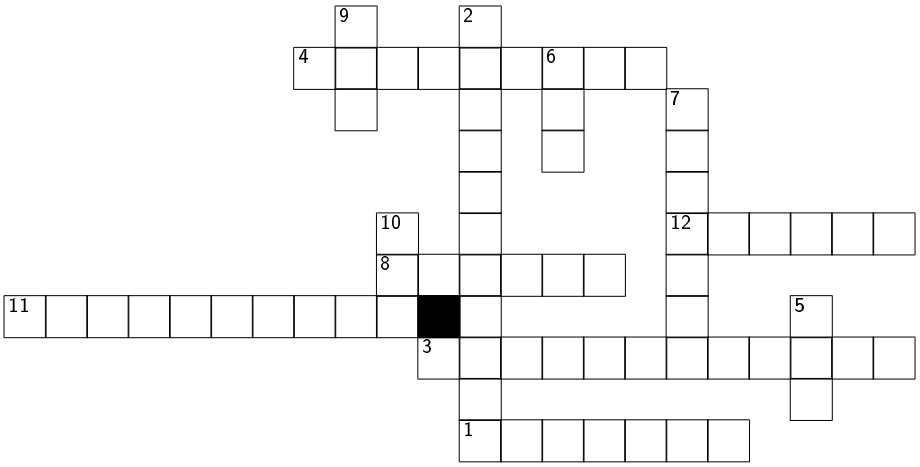
7 Poziţii care nu par să îşi schimbe valoarea în viitorul apropiat

8 Funcţie care primeşte o stare şi întoarce cea mai bună mutare

11 Simulare Monte Carlo pentru jocuri cu zaruri

12 Succesoarea mitologică a lui Deep Blue

Probleme de satisfacere a constrângerilor



Orizontal

1 Constrângere cu multe variabile

3 Euristică la CSP cu căutare locală (en)

4 Inferenţă pe constrângeri

8 Tip de constrângere globală

11 Tip de backtracking în care cea mai recentă decizie este reconsiderată

12 Euristică pe număr de constrângeri (en)

Vertical

2 Schimbarea celei mai recente instanţieri a unei variabile din setul conflict

5 Algoritm pentru consistenţa muchiilor dezvoltat în 1977 de Mackworth

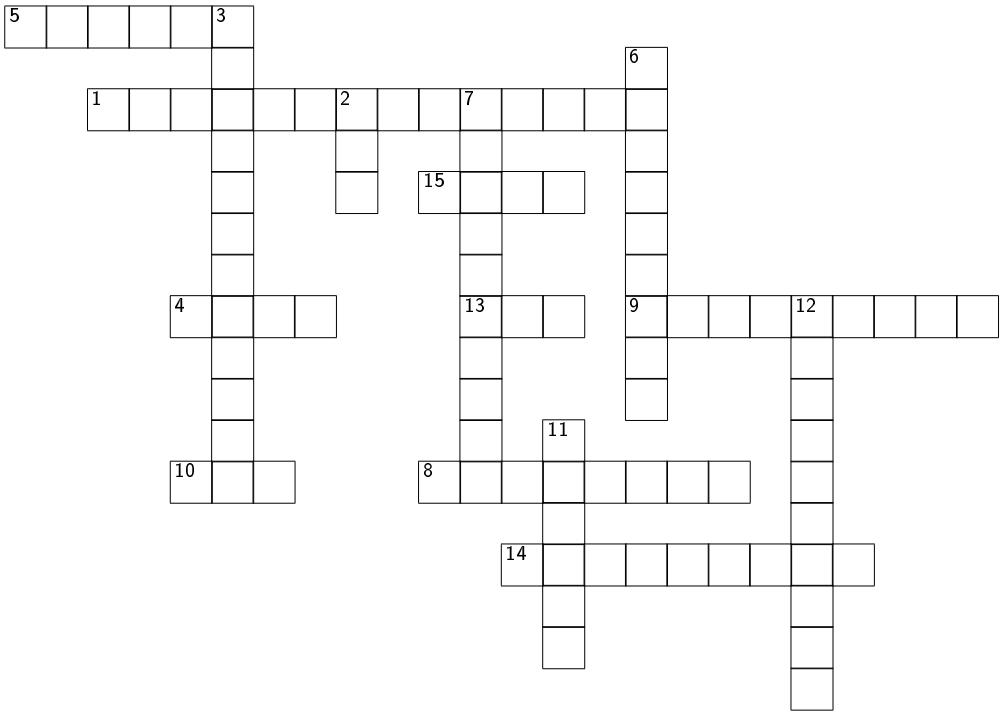
6 Algoritm pentru consistenţa arcului

7 Tip de constrângere globală aplicată şi la Sudoku

9 Euristica celor mai puţine valori rămase

10 Algoritm pentru menţinerea consistenţei arcului

Agenţi logici



Orizontal

1 $a \to b \equiv \neg b \to \neg a$

4 Propoziţie cu un singur literar pozitiv

5 Celebru monstru care duhneşte în labirint

8 $\neg(a \wedge b) \equiv \neg a \vee negb$

9 Deducerea de propoziţii noi pe baza celor vechi

10 Problemă de satisfiabilitate

13 Forma normal conjunctivă (en)

14 Clauză Horn cu exact un literar pozitiv

15 Algoritm pentru satisfiabilitate

Vertical

2 Simbol care nu apare negat sau apare doar negat

3 Propoziţie adevărată în cel puţin o lume posibilă

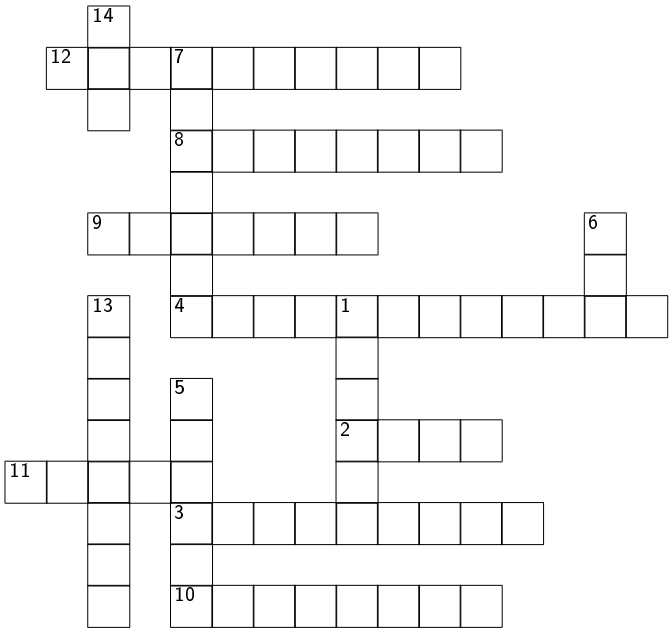
6 Înţelesul propoziţiilor dintr-o logică

7 Operator logic pentru $\neg a \vee b$

11 Regulă de înferenţă (lt)

12 Algoritm complet de inferenţă care foloseşte reducerea la absurd

Logica de ordinul întâi



Orizontal

2 Propoziţie atomică pe scurt

3 Cuantificator în logica de ordinul întâi

4 Cuantificator în logica de ordinul întâi

8 Formalizează un domeniu particular

9 Introduce cuantificatorii în 1879

10 Două axiome îi poartă numele

11 Dezvoltă sintaxa logicii de ordinul întâi

12 Propoziţie validă

Verical

1 Funcţii care au valori pentru fiecare intrare din domeniul specificat

5 Termen fără variabile (en)

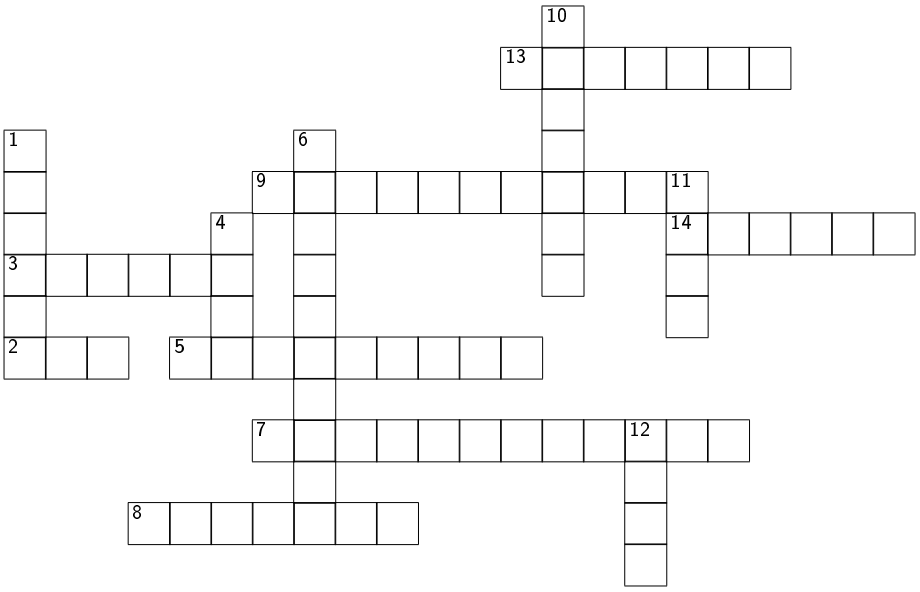
6 Asumpţie în care propoziţiile despre care nu ştim că sunt adevărate sunt false (en)

7 Propozitţii care nu sunt axiome

13 Regulă care leagă o cauză de efectul ei

14 Primul sistem de raţionare în logica de ordinul întâi

Inferenţă în logica de ordinul întâi



Orizontal

2 Cel mai general unificator de clauze (en)

3 Structură cu relaţii de ordine parţială

5 Baze de date care permit inferenţă

7 Regulă de inferenţă în logica de ordiul întâi

8 Demonstrator de teoreme succesor al lui Otter

9 Modus ponens pentru logica de ordinul întâi

13 Baze de cunoştinţe fără funcţii

14 Limbaj de programare logică

Vertical

1 Constante introduse pentru variabilele cuantificate existenţial

4 Algoritm eficient de raţionare cu reguli

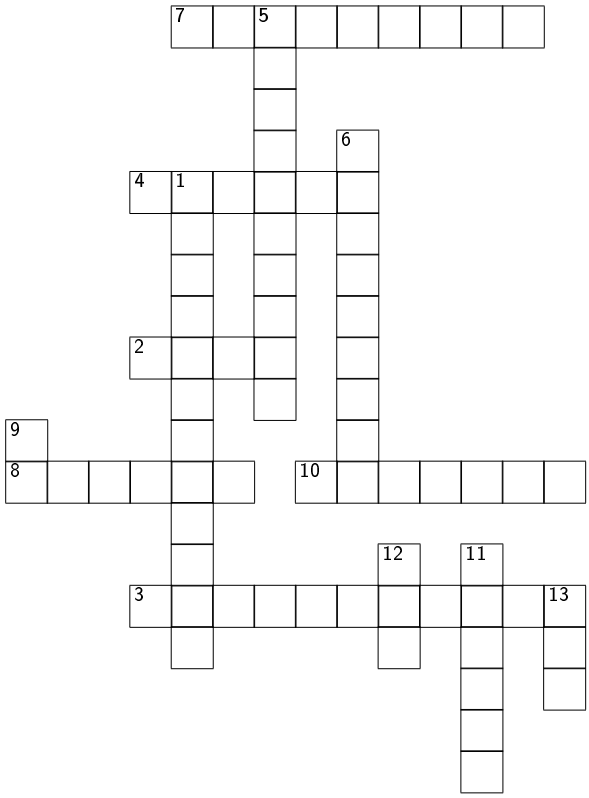
6 Regulă de inferenţă în logica de ordiul întâi

10 Campion între demonstratoarele de teoreme

11 Colecţie de teoreme (en)

12 Demonstrator de teoreme care poate proiecta circuite

Planificare clasică



Orizontal

2 Limbaj specializat pentru probleme de planificare

3 Prezente la acţiuni

4 Prezente la acţiuni

7 Algoritm de planificare

8 Relaţie care variază de la o stare la alta

10 Planificator care foloseşte satisfiabilitate

Vertical

1 Planificator cu euristici în Python

5 Acţiune cu toate precondiţiile satisfăcute

6 Acţiune cu un efect care se regăseşte în starea finală

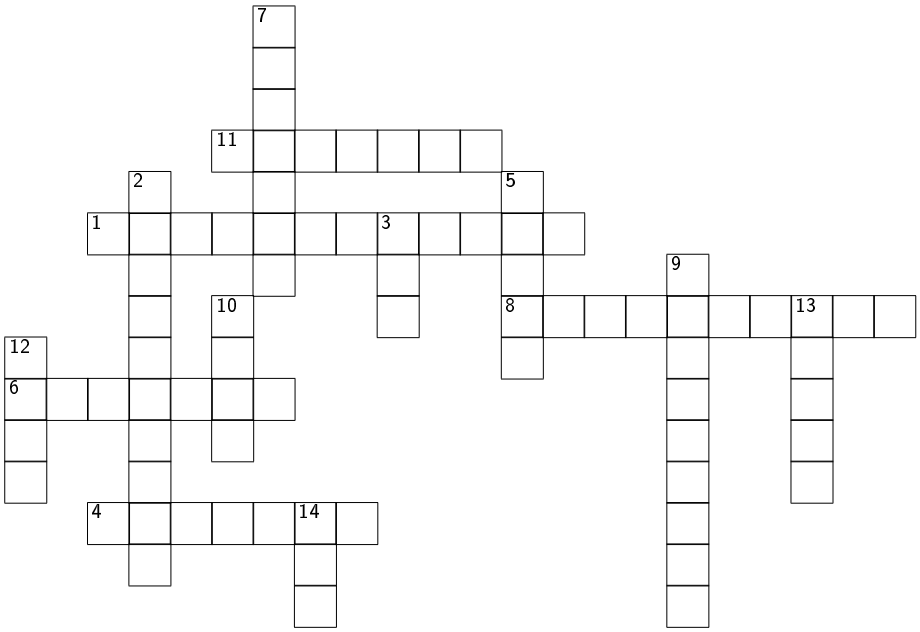
9 Planificator pe repede-înainte

11 Limbaj de planificare inclus in PDDL

12 Diagrame de decizie pentru reprezentarea eficientă a planurilor

13 Competiţie de planificare

Planificare în lumea reală



Orizontal

1 Efecte prefixate cu *when*

4 Cea mai lungă cale

6 Nondeterminism generat de un alt agent

8 Planificare fără senzori (en)

11 Consumate de acţiuni

Vertical

2 Necesară în planificarea multi-agent

3 Asumpţie în care starea conţine atât fluenţi negativi cât şi pozitivi

5 $LS - ES$ (en)

7 Nondeterminism generat de agentul curent

9 Mod de a selecta planul curent comun pentru mai mulţi agenţi

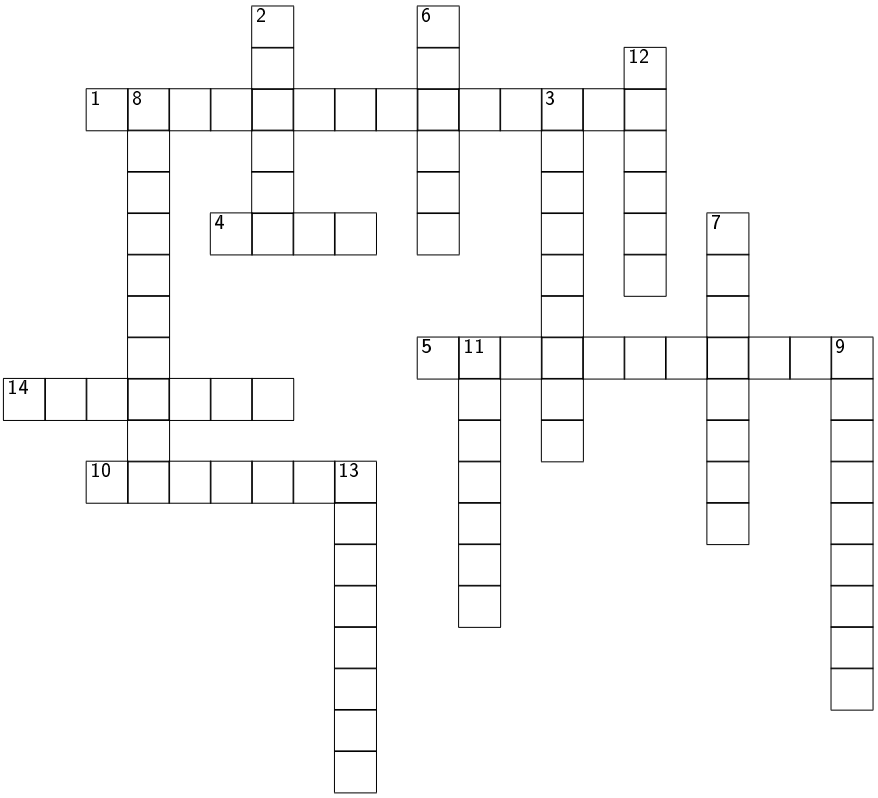
10 Agent pasăre (en)

12 Folosite de către planificatorul conformant HSCP

13 Conferinţă anuală pentru agenţi

14 Determină posibilele momente de început şi sfârşit ale unei acţiuni

Reprezentarea cunoştinţelor. Calculul evenimentelor



Orizontal

1 Mai puternică decât prezumţia de lume închisă

4 Sistem de menţinere a adevărului bazat pe justificări

5 Logici în care o concluzie derivată poate fi retrasă

10 Evenimente care împarţite dau acelaşi eveniment

14 Ontologia wikipediei

Vertical

2 Tipul lui f în $T(f,t)$

3 Ontologie doar cu relaţii de tip *subclasă*

6 Predicat pe intervale temporale (en)

7 Decompoziţie exhaustivă disjunctă

8 Proprietăţi interne ale obiectului

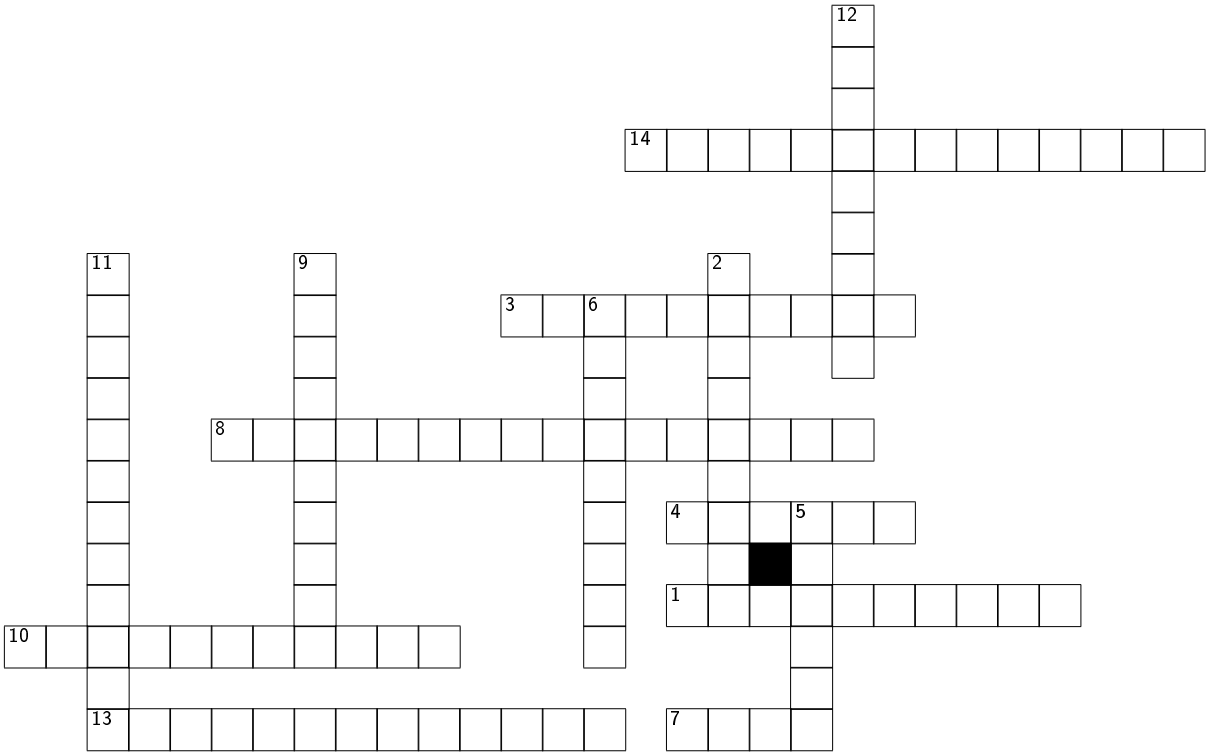
9 Tipul lui e în $Happens(e,i)$

11 Predicat pe intervale temporale (en)

12 Predicat pe intervale temporale (en)

13 Setul maximal de consecinţe dintr-o bază de cunoştinţe

Probabilităţi



Orizontal

1 Teoria bazată pe teoria probabilităţilor şi a utilităţilor

3 Rezultate penderate cu probabilitatea lor de apariţie

4 Spaţiu pentru toate lumile posibile (en)

7 Funcţii densitate de probabilitate (en)

8 Teorie pentru grade de certitudine

10 Interpretare conform căreia incertitudinea provine din ignoranţa condiţiilor iniţiale

13 Însumarea probabilităţilor pentru fiecare valoare posibilă a celorlalte variabile

14 Interpretare conform căreia probabilităţile provin doar din experimente

Vertical

2 Calitatea de a fi util

5 Probabilităţi neconditionate (en)

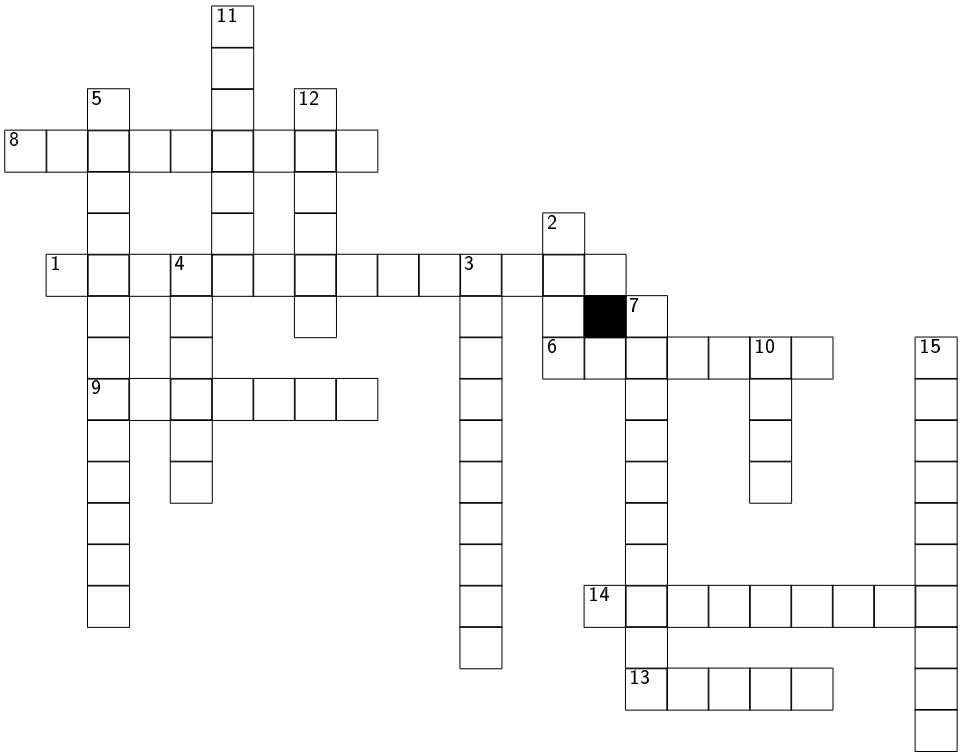
6 Probabilităţi conditionate (en)

9 Matematician rus care a dezvoltat teoria probabilităţilor

11 Interpretare conform căreia probabilităţile caracterizează cunoştinţele unui agent

12 Calcularea probabilităţilor posterioare pe baza observaţiilor/evidenţelor curente

Raţionare în reţele Bayesiene



Orizontal

1 Reţea Bayesiană

6 Reţea Bayesiană

8 Criteriu topologic pentru a decide dacă un set de noduri X este condiţional independent de Y fiind dat Z

9 Generalizare a funcţiei OR pentru relaţii incerte (en)

13 Distribuţie de probabilitate (en)

14 Distribuţie de probabilitate

Vertical

2 Markov Chain Monte Carlo!

3 Semantică în care variabilele sunt condiţional independente de nondescendenţii lor, fiind daţi părinţii

4 Reţea Bayesiană (en)

5 Nod cu valoarea specificată exact de către unul dintre părinţi

7 Dezvoltator al reţelelor Bayesiene şi rezident în Los Angeles

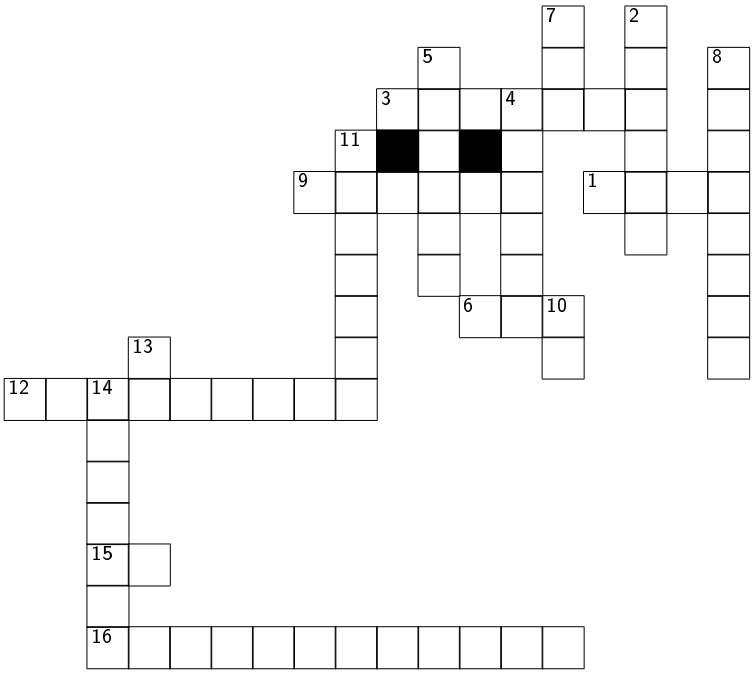
10 Nod pentru tot felul de cauze (en)

11 Reţea cu variabile atât discrete, cât şi continue

12 Distribuţie de probabilitate (en)

15 Algoritmi de eşantionare aleatoare

Reţele Bayesiene



Orizontal

1 Model Bayesian în care o singură cauză influenţează efecte independente)

3 Autor al primei analize sistematice a probabilităţilor în anul 1565

6 Prezumţia de nume unic (en)

9 Asumpţie conform căreia starea curentă depinde de un număr finit şi fix de stări anterioare

12 Algoritm care decide dacă două variabile sunt independente într-o reţea Bayesiană

15 Dempster-Shafer!

16 Marginalizare cu probabilităţi condiţionate în locul probabilitatilor cuplate (joint)

Vertical

2 Prenumele lui Bayes

4 Autorul cărtii *Partida neterminată (The Unfineshed Game)*, în care se discută celebra corespondenţă dintre Pascal şi Fermat din 1654

5 Criteriu (pătură) pentru stabilirea independenţei condiţionale

7 Prezumţie de lume închisă

8 Primul sistem expert bazat pe reţele Bayesiene

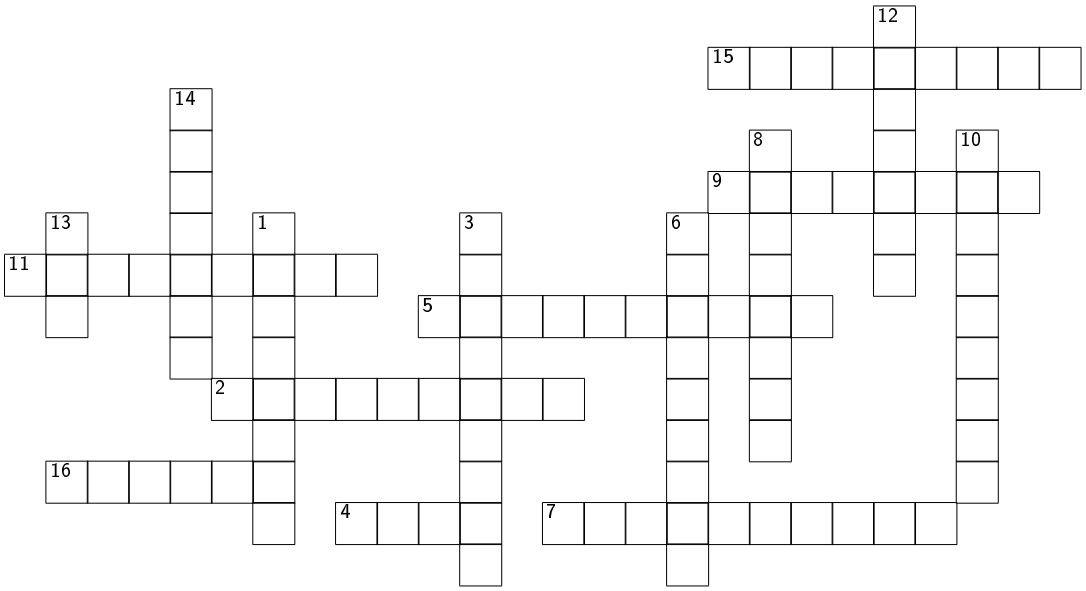
10 Stastistician rus (1856-1922)

11 A descoperit independent de Bayes regula lui Bayes

13 Jude Pearl!

14 Sistem în care fiecare stare este accesibilă din oricare alta şi nu există cicluri stricte periodice

Filtru Kalman



Orizontal

2 $P(X_{t+k}|e_{0..t})$

4 Inventatorul Algoritmului Ungar

5 Sisteme potrivite pentru filtrul Kalman extins

7 Eroare a senzorului care trimite ocazional valori eronate

9 Calculul distribuţiei posterioare pe starea curentă, fiind date evidenţele până la momentul curent

11 Calculul distribuţiei posterioare pe o stare trecută, pe baza evidenţelor până la momentul curent (en)

15 Proces în care schimbarea se supune unor legi care nu se schimbă în timp

16 Asumpţia $P(E_t|X_{0..t}, E_{0..ti-1} = P(E_t|X_t)$

Vertical

1 $P(X_t|e_{0..t})$

3 $P(X_k|e_{0..t})$, $k < t$ (en)

6 Multiple filtre Kalman

8 Estimarea stării curente

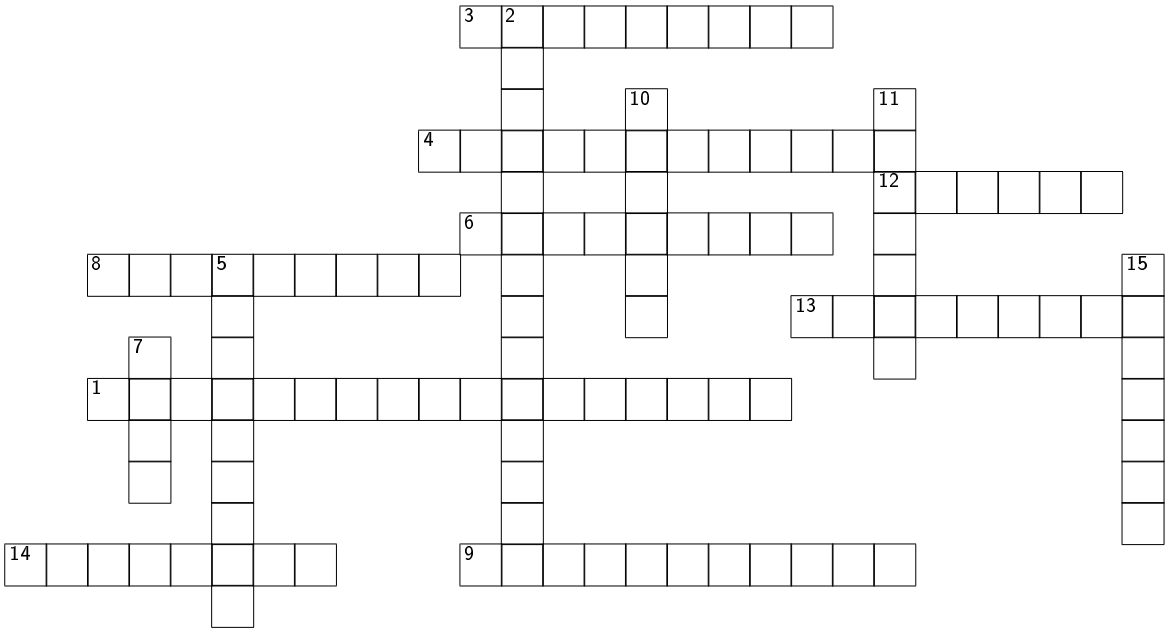
10 Calculul distribuţiei posterioare pe o stare viitoare, fiind date evidenţele până la momentul curent

12 Algoritm pentru identificarea celei mai probabile secvenţe

13 Hidden Markov Model!

14 Alarmă falsă (en)

Decizii simple



Orizontal

1 Constrângere pe loterii

3 Funcţie care modelează preferinţele agenţilor

4 Constrângere pe loterii

6 Măsură utilizată în domeniul medical

8 Teorie care descrie cum trebuie un agent sa acţioneze

9 Teorie care descrie cum se comportă în realitate un agent

12 Ovale în reţele de decizie (en)

13 Romburi în reţele de decizie

14 Functie acţiune-utilitate în învaţarea reinforsată

Vertical

2 Constrângere pe loterii

5 Constrângere pe loterii

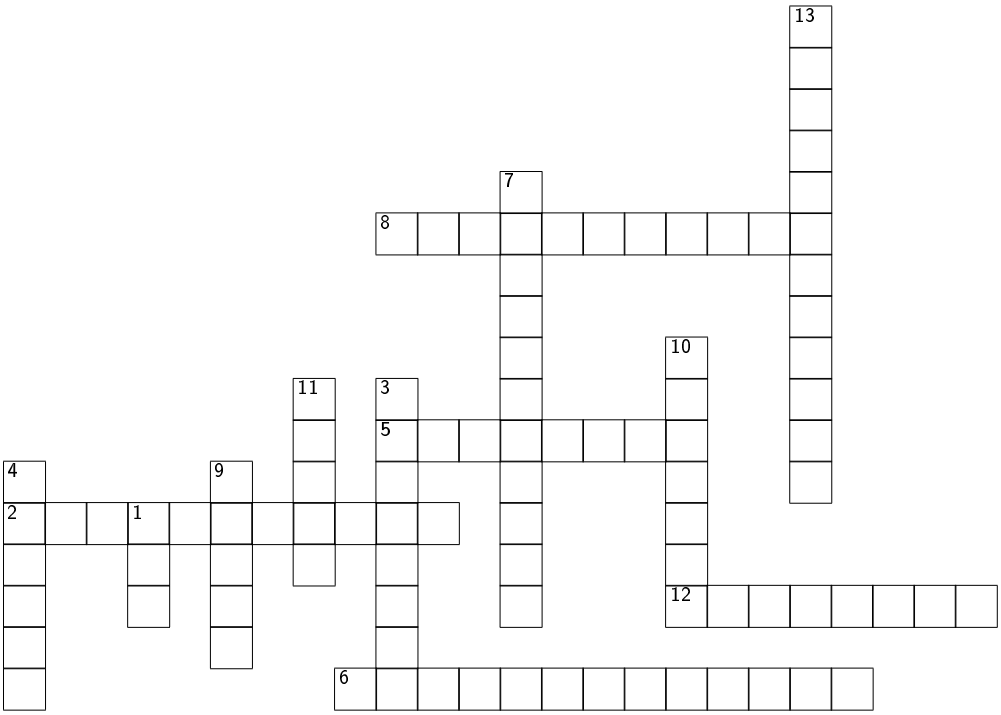
7 Ultimul cuvânt din abrevierea QUALY (en)

10 Bias cognitiv

11 Reţea cu noduri de tip variabilă aleatoare, decizii şi utilităţi

15 Dezvoltatorul calculului hedonic

Învăţare supervizată. Arbori de decizie



Orizontal

2 Învăţare a unui atribut având un set finit de valori

5 Măsură a importanţei unui atribut în luarea deciziei

6 Tip de învăţare

8 Tip de învăţare

12 Vector de derivate parţiale

Vertical

1 Algoritm de învăţare supervizată

3 Învăţare a unui atribut având un set continuu de valori

4 Filozof englez căruia îi plac lucrurile simple

7 Tip de învăţare

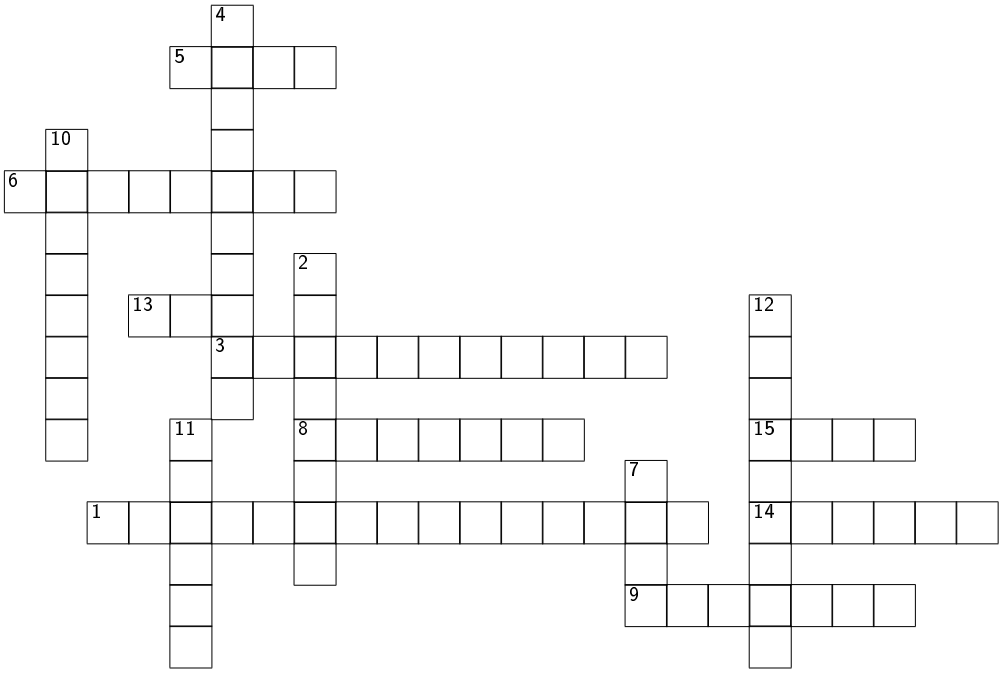
9 Variantă de validare încrucişată

10 Când setul de test influenţează parametrii învăţării (en)

11 Validare încrucişată cu $k = n$

13 Proces de penalizare explicită a ipotezelor complexe

Reţele neurale



Orizontal

1 Algoritm de învăţare dezvoltat de Werbos în 1975 (nn)

3 De evitat la învăţare (en)

5 Reţea cu multe niveluri ascunse

6 Vector de derivate parţiale

8 Funcţie de activare simplu de derivat

9 Funcţie de activare

13 Funcţie logică neagreată de perceptron

14 Algoritm de extindere a unei reţele iniţiale (en)

15 Funcţie liniară de activare a neuronului

Vertical

2 Medie armonică între precizie şi recall (en)

4 Reţea fără niveluri ascunse

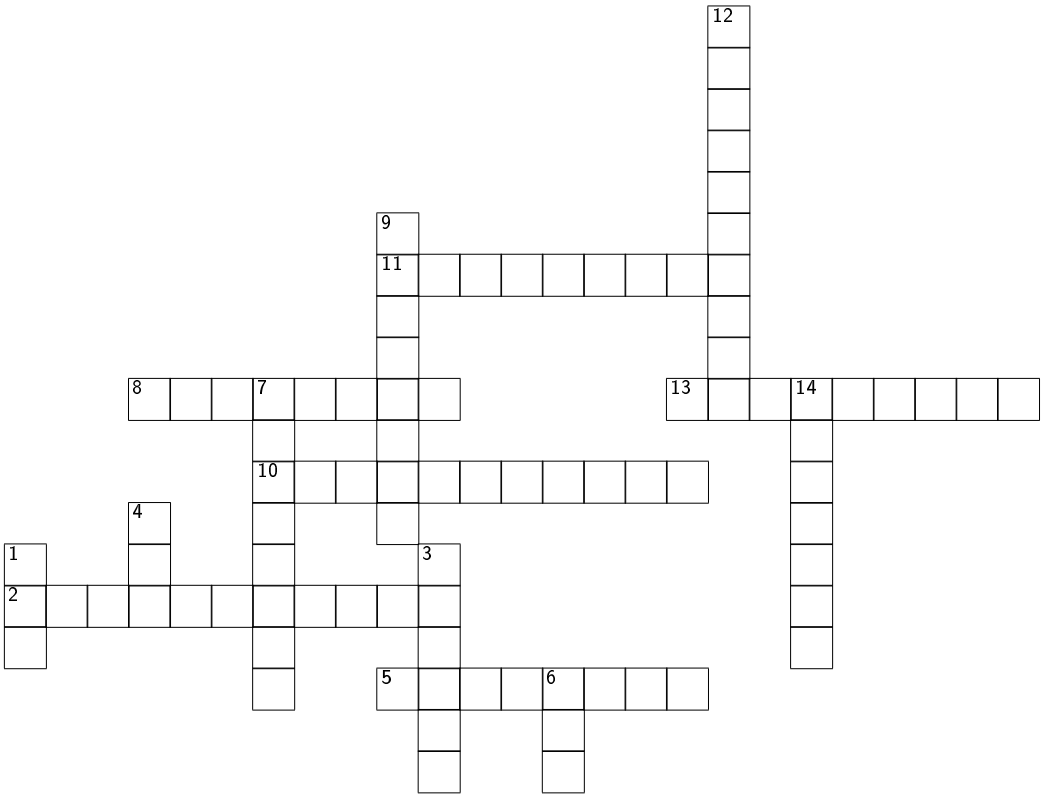
7 Funcţie de evaluare a erorii (en)

10 Metrică de evaluare

11 Metrică de evaluare

12 Metrică de evaluare

Ansambluri de clasificatori



Orizontal

2 Prepocesare necesară la KNN

5 Mulţi clasificatori

8 Algoritm adaptiv de învăţare cu ansambluri

10 De evitat la învăţare (en)

11 Distantă între două instanţe

13 Distanţă Minkowski cu $p = 1$

Vertical

1 Algoritm nonparametric

3 În centrul SVM

4 Algoritm nonparametric

6 Mulţi agenţi

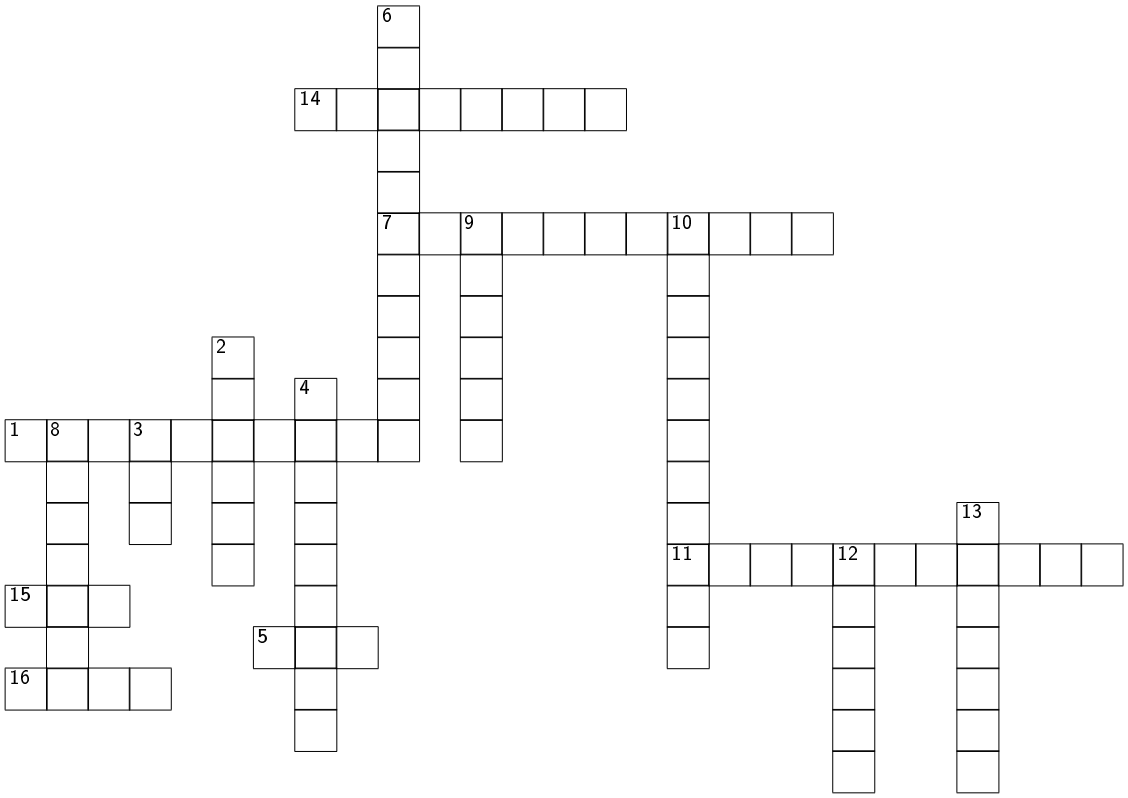
7 Metodă în învăţarea cu ansambluri de clasificatori

9 Medie armonică între precizie şi recall (en)

12 Distanţă Minkowski cu $p = 2$

14 Distanţă pe atribute booleene

Învăţare nesupervizată



Orizontal

1 Metodă de învăţare nesupervizată

5 Metrică pentru distanţa dintre clustere sensibilă la zgomote

7 Primul cuvânt din abrevierea i.i.d.

11 Tip de clusterizare ierarhică

14 Medie armonică între precizie şi recall (en)

15 Metrică pentru distanţa dintre clustere

16 Metrică pentru distanţa dintre clustere bazată pe SSE

Vertical

2 Algoritm de clusterizare de tip partiţie

3 Metrică de evaluare a clusterizării

4 Variantă de K-means

6 De evitat la învăţare (en)

8 Complexitatea algoritmului K-means

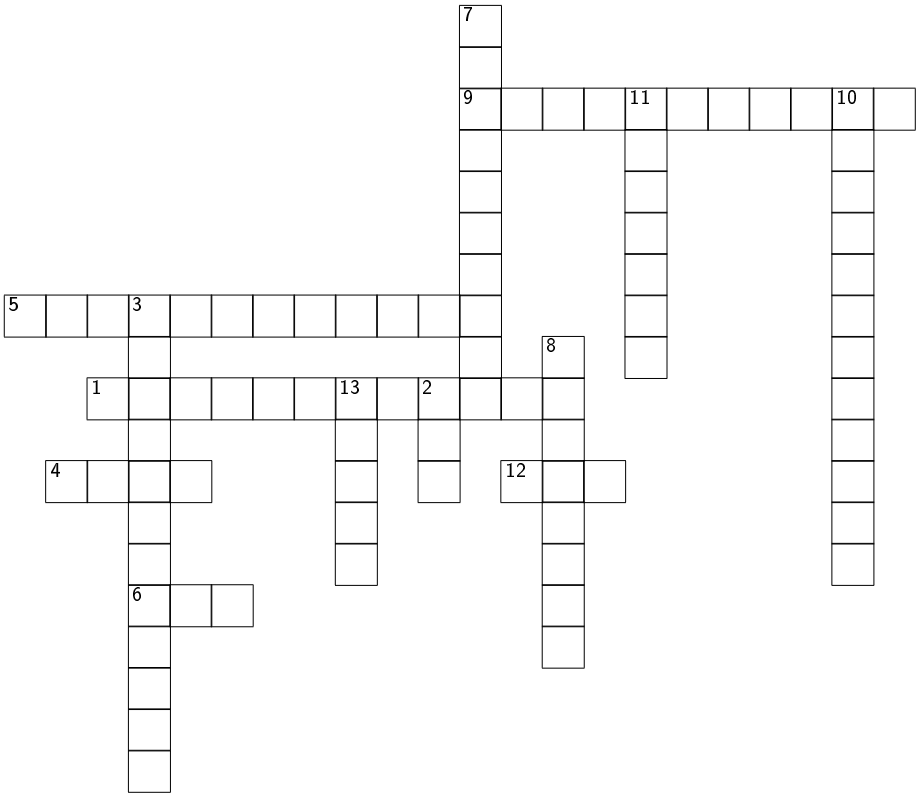
9 Algoritm de clusterizare bazat pe densitatea punctelor

10 Reprezentare grafică în clusterizarea ierarhică

12 Centroid membru în setul de date

13 Metrică pentru similaritatea a două seturi de date

Învăţare bazată pe cunoştinţe



Orizontal

1 Modificare a ipotezei pentru a gestiona instanţe fals negative

4 Învăţare inductivă bazată pe cunoştinţe (en)

5 Algoritmi care generează noi predicate prin inducţie

6 Algoritm dezvoltat de Quinlan pentru inducţia regulilor

9 Salvarea unor rezultate intermediare de calcul (en)

12 Algoritm dezvoltat de Clark şi Niblett pentru inducţia regulilor

Vertical

2 Primitivul din capitol

3 Modificare a ipotezei pentru a gestiona instanţe fals pozitive

7 Tip de învăţare în care agenţii învaţă mai bine pe măsură ce au mai multe cunoştinţe

8 Brazilianul din capitol

10 Ţeluri uşor de rezolvat

11 Rezoluţie la programarea logică inductivă

13 Translator din FOL în logica propoziţională

Partea II

Soluţii şi explicaţii

Capitolul 3

Întrebări cu răspuns multiplu

24. **Introducere în inteligenţă artificială**

(a) Când s-a născut inteligenţa artificială? (30s)

- √ **1956 - întâlnirea de la Dartmouth**
- ◯ 1943 - modelarea creierului cu circuite booleene
- ◯ 1950 - articolul lui Turing *Computing Machinery and Intelligence*
- ◯ 1956 - algoritmul complet al lui Robinson pentru raţionare logică

Explicaţie: Conferinţa de la Colegiul Dartmouth, USA a fost organizată de Marvin Minsky, John McCarthy, Clause Shannon şi Nathan Rochester. Viziunea organizatorilor era ca orice aspect al inteligenţei poate fi formalizat atât de precis încât o maşină îl poate simula. Termenul de *inteligenţă artificială* a venit la propunerea lui McCarthy.

(b) Testul lui Turing tratează IA din perspectiva (30s)

- ◯ gândirii umane
- ◯ gândirii raţionale
- √ **comportamentului uman**
- ◯ comportamentului raţional

Explicaţie: Criteriul propus de Turing consideră un agent inteligent dacă acesta acţionează similar unui agent uman. Criterul nu impune ca similaritatea comportamentală să fie obţinută prin mimarea modului de gândire a agentului uman.

(c) Perspectiva AIMA asupra IA impune agenţilor să (30s)

- ◯ gândească uman
- ◯ gândească raţional
- ◯ acţioneze uman
- √ **acţioneze raţional**

Explicaţie: Se doreşte proiectarea unor agenţi raţionali care acţionează pentru a obţine cel mai bun rezultat.

(d) Numele primului calculator care a învins campionul mondial la şah? (30s)

- ◯ Deep Purple
- ◯ Windows Azure
- √ **Deep Blue**
- ◯ Black Friday

Explicaţie: Deep Blue l-a învins pe Garry Kasparov în 1997, câştigând meciul de 6 partide cu scorul $3\frac{1}{2} - 2\frac{1}{2}$. Algoritmul executa o căutare *alpha-beta* în paralel.

(e) Cine nu apare pe coperta AIMA? (30s)

◯ Alan Turing (1912–1954)
◯ Aristotle (384–322 I.C.),
◯ Thomas Bayes (1702–1761)
√ **Marvin Minski (1927–2016)**

Explicaţie: Aristotel a formalizat procesele de raţionare din mintea umană care permit deducerea concluzilor pe baza unor premise. Turing a propus în 1950 un criteriu de a decide dacă o maşină este inteligentă. Reverendul Bayes apare pe copertă datorită contribuţiilor legate de probabilităţi, în special renumita formula a lui Bayes. Cel care nu apare este Marvin Minski, unul din organizatorii conferinţei de la colegiul Dartmouth din 1956. O carte recentă a acestuia este *The emotion machine* (2006).

(f) Testul Turing total include (30s)

◯ învăţare automată şi viziune computerizată
√ **viziune computerizată şi robotică**
◯ procesarea limbajului natural
◯ înţelegerea limbajului şi robotică

Explicaţie: Testul extinde versiunea clasică cu recunoaşterea scenei şi interacţiune fizică. Sunt necesare astfel viziunea computerizată şi robotica.

(g) Predicţii legate de testul Turing (30s)

√ **până în 2000, 30% şanse ca cineva să fie indus în eroare pentru 5 minute**
◯ până în 2000, 30% şanse ca un expert să fie indus în eroare
◯ până în 2000, să fie păcălită orice persoană
◯ 20% şanse să fie păcălită orice persoană

Explicaţie: Predicţia a fost făcută de Turing în *Computing Machinery and Intelligence* (1950)

(h) Sistemele multi-agent s-au dezvoltat începând cu (30s)

◯ 1989
◯ 2001
◯ 2003
√ **1995**

Explicaţie: Un articol pe linia aceasta este Wooldridge, Michael, and Nicholas R. Jennings. "Intelligent agents: Theory and practice". The knowledge engineering review 10.2 (1995): 115-152.

(i) Ultimul joc la care agentul uman a avut supremaţia a fost (30s)

- ◯ Şah
- √ **Go**
- ◯ Spinner
- ◯ Game of Thrones

Explicaţie: Versiunile programului AlphaGo caută cea mai bună mutare folosind metode de tip Monte Carlo şi modele învaţate cu reţele neurale sau învăţare reinforsată.

25. **Agenţi**

(a) Raţionalitatea implică (30s)

- ◯ omniscienţă, învăţare, autonomie
- √ **explorare, învăţare, autonomie**
- ◯ culegere de informaţii, învăţare, omniscienţă
- ◯ succes, învăţare, autonomie

Explicaţie: Omniscienţa presupune că agenţii pot deduce toate consecinţele logice din bazelor lor de cunoştinţe. Ominisicenţa este o proprietate prea puternică pentru agenţii software. Similar, faptul că un agent ia decizii raţionale nu îi garantează succesul.

(b) PEAS provine de la (30s)

- ◯ Planning, Environment, Actuators, Sensors
- ◯ Performance measure, Environment, Actuators, Search
- ◯ Performance measure, Environment, Agents, Sensors
- √ **Performance measure, Environment, Actuators, Sensors**

Explicaţie: Agentul percepe *mediul* prin *senzori* şi acţionează asupra acestuia prin *actuatori*. Cât de bine evoluează agentul este măsurat de o funcţie de performanţă.

(c) Mediul este semi-static dacă (30s)

- ◯ mediul şi funcţia de performanţă se schimbă
- ◯ mediul se schimbă, funcţia de performanţă nu
- √ **mediul nu se schimbă, funcţia de performantă se schimbă**
- ◯ mediul şi funcţia de performanţă nu se schimbă

Explicaţie: Un eexmplu de mediu semi-static este jocul de şah cu ceas. În timpul deliberării poziţia de pe tablă nu se modifică. În schimb, funcţia de evaluare întoarce valori mai mici odată cu trecerea timpului.

(d) În medii nondeterminste stărilor succesoare le sunt ataşate probabilităţi (30s)

- ◯ Adevărat
- √ **Fals**

Explicaţie: Sunt specificate toate stările succesoare *posibile* unei acţiuni nedeterminste. Acestor stări nu le sunt ataşate probabilităţi.

(e) Mediul în jocurile de tip *rebus* este (30s)

- ◯ deterministic şi continuu

◯ static şi parţial observabil
◯ multi-agent şi secvenţial
√ **nicio variantă din cele anterioare**

Explicaţie: Doar ultima variantă este corectă deoarece: (1) Stările succesoare sunt generate prin completarea unei litere. Mediul este aşadar discret. (2) Definiţiile cuvintelor şi poziţia curentă sunt complet observabile de către agent. Mediul la rebus este deci complet observabil. (3) Există un singur agent care completează cuvintele.

(f) Mediul pentru şah fără ceas este (30s)

◯ semi-static şi complet observabil
√ **static şi complet observabil**
◯ continuu şi secvenţial
◯ episodic şi multi-agent

Explicaţie: Doar variantă (2) este corectă, astfel: (1) Deoarece nu există ceas, performanţă agentului nu scade în timp ce acesta deliberează. (3) Poziţiile succesoare sunt generate cu fiecare mutare - mediul este discret. (4) Acţiunile curente ale agenţolor depind de mutările efectuate anterior - mediul este secvenţial şi nu episodic.

(g) Mediul pentru şoferul de taxi este (30s)

◯ continuu şi complet observabil
◯ dinamic şi complet observabil
√ **continuu şi secvenţial**
◯ episodic şi multi-agent

Explicaţie: Doar variantă (3) este corectă, deoarece: (1,2) Anumite variabile nu pot fi observate (e.g. numărul de măsini pe o anumită rută) - mediul este doar parţial observabil. (4) Acţiunea din starea curentă depinde de o acţiunile executate anterior - mediul este secvenţial şi nu episodic.

(h) Un agent reflex bazat pe model menţine o stare interioară (30s)

√ **Adevărat**
◯ Fals

Explicaţie: Modelul reprezintă cunoştinţele agentului despre cum funcţionează lumea sau mediul în care acţionează.

(i) Agenţii bazaţi pe utilitate (30s)

◯ utilizează reguli de tip *conditie* → *actiune*

√ **aleg acţiunea care îi conduce la cea mai bună utilitate expectată**

◯ folosesc planificarea pentru a atinge starea ţintă

Explicaţie: Suplimentar unui model al lumii, se utilizează o funcţie de utilitate care măsoara performanţa agentului în fiecare stare. Pe baza funcţiei de utilitate, agentul alege cea mai bună acţiune ponderând utilitatea stării succesoare cu probabilitatea de a ajunge în respectiva stare.

(j) Un agent care învaţă îşi poate îmbunătăţi (30s)

◯ doar cunoştinţele iniţiale

◯ doar funcţia de evaluare a performanţei

◯ doar generatorul de probleme utilizat în învăţare

√ **toate elementele sale**

Explicaţie: Algoritmii de învăţare se pot aplica pe orice componentă din arhitectura agentului.

26. **Căutare neinformată**

(a) Dacă agentul nu ştie care este starea lui iniţială, problema de rezolvat este de tipul (30s)

- ◯ stare unică (single state)
- √ **conformant**
- ◯ contingent

Explicaţie: În *conformant problems*, perceptele agentului nu sunt sufieciente pentru a decide starea în care se află. Totuşi un astfel de agent poate executa acţiuni care îl duc intr-o stare cunoscută. Pornind de la un set stări posibile în cre se află, agentul execută acţiuni cu scopul de a reduce setul la o singură stare. Probleme contingente apar în medii nondeterministe.

(b) Complexitatea în timp a unui algoritm reprezintă (30s)

- ◯ timpul în secunde pentru calcularea soluţiei
- ◯ numărul mediu de noduri în memorie
- √ **numărul total de noduri generate/expandate**
- ◯ numărul maxim de noduri în memorie în timpul căutării

Explicaţie: Complexitatea în timp a unui algoritm este estimată prin numărul de operaţii efectuate. În cazul căutării operaţiile se referă la crearea de noduri în arborele de căutare.

(c) Un algoritm care găseşte întotdeauna soluţia dacă aceasta există este (30s)

- ◯ complet
- √ **optimal**

Explicaţie: Bineînteles, completitudinea nu implică optimalitate.

(d) Care este cea mai bună strategie de căutare pe un calculator cu memorie puţină? (30s)

- √ **în adâncime**
- ◯ în lăţime
- ◯ cost uniform

Explicaţie: Căutarea în lăţime păstrează în memorie $O(b^{d+1})$ noduri, unde b este factorul de ramificare, iar d adâncimea arborelui de căutare. Practic toate nodurile sunt păstrate în memorie. Spaţiul de memorare este aşadar o problemă pentru căutarea în lăţime. Căutarea cu cost uniform expandează nodul cu cel mai mic cost. Când costurile sunt identice, căutarea cu cost uniform este similară celei în adâncime. Căutarea în adâncime păstrează doar $O(bm)$ noduri, unde m este adâncimea la care se găseşte soluţia. Aşadar, spaţiul este liniar raportat la factorul de ramificare b.

(e) Căutarea în adâncime găseşte cea mai bună soluţie (30s)

◯ adevărat

√ **fals**

Explicaţie: Căutarea în adâncime nu este optimală. De exemplu, o soluţie aflată la adâncimea 5 în arborele de căutare poate fi găsită înaintea unei soluţii aflate la adâncimea 3.

(f) A* este un algoritm de tip greedy ghidat de functia de estimare a costului $f(n) =$ (30s)

◯ costul de la nodul n la ţintă

◯ costul de la starea iniţială la nodul n

◯ maximul dintre costul de la starea iniţială la n şi costul de la n la ţintă

√ **nicio variantă din cele enumerate**

Explicaţie: La A*, $f(n) = g(n) + h(n)$. Pe lângă costul $g(n)$ de la starea iniţială la nodul curent n, algoritmul A* ţine cont şi de estimarea costului de la nodul curent la ţintă. Această estimare este data de euristica $h(n)$.

(g) Care nu este un algoritm de căutare informată? (20s)

√ **cost uniform**

◯ greedy-best search

◯ A*

Explicaţie: Căutarea informată este ghidată de euristici $h(n)$. A* selectează nodul cu cea mai bună valoare $g(n)+h(n)$. Greedy best first search este un caz particular de A* în care funcţia $g(n)$ nu este luată în calcul. Decizia depinde doar de valoarea estimării $h(n)$. Atât A*, cât şi *greedy-best first* se bazează pe euristici.

(h) În problema jocului culisant, care nu sunt euristici admisibile? (30s)

◯ $h_1 =$ numărul de piese care nu se află în poziţia finală

◯ $h_2 =$ suma distanţelor de la fiecare piesă la poziţia ei finală

◯ $h_3 = max(h_1, h_2)$

√ $h_4 = sum(h_1, h_2)$

Explicaţie: h_1 şi h_2 sunt admisibile deoarece nu supraestimează costul real. Maximul dintre două euristici admisibile rămâne o euristică admisibilă. În schimb, suma valorilor a două euristici admisibile poate depăşi costul real.

27. **Dincolo de căutarea clasică**

(a) Care dintre următorii algoritmi este complet? (30s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- ◯ first choice hill climbing
- √ **random-restart hill climbing**

Explicaţie: Un algoritm complet garantează găsirea soluţiei dacă aceasta există. Algorimul hill-climbing poate rămâne blocat într-un maxim local (i.e. un agent nu poate ieşi dintr-o stare care este evaluată mai bine decât toate stările succesoare posibile). În varianta stocastică se selectează aleatoriu o stare din setul de stări evaluate mai bine decât starea curentă. Probabilitatea de selecţie poate depinde de diferenţa de utilitatea între starea succesoare şi cea curentă. În versiunea *first choice hill climbing*, prima stare identificată ca având o utilitate mai bună va fi aleasă. Toate aceste metode nu garantează găsirea soluţiei. Varianta hill climbing cu reporniri aleatoare rulează algoritmul pornind din mai multe stări iniţiale aleatoare. Cu un număr suficient de mare de rulări, probabilitatea de a găsi soluţia se apropie de 1.

(b) Când o stare are mii de stări succesoare, vei utiliza în căutare (60s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- √ **first choice hill climbing**

Explicaţie: Versiunea stocastică a algoritmului hill climbing alege aleatoriu una dintre stările succesoare evaluate mai bine decât starea curentă. Algoritmul evaluează toate stările succesoare. La un factor de ramificaţie (branching factor) aşa mare e ineficientă evaluarea tuturor stărilor succesoare. Algoritmul *first choice hill climbing* selectează prima stare mai bună, în urma evaluării unei stări succesoare alese aleator. Se elimină astfel evaluarea tuturor stărilor succesoare.

(c) Algoritmul de *răcire controlată* păstrează în memorie (20s)

- √ **1 stare**
- ◯ k stări, $k > 1$

Explicaţie: Algoritmul alege *o stare* succesoare s aleator. Dacă în urma evaluării se dovedeşte mai bună decât starea curentă ($E_s > E_c$), starea s este selectată. Dacă starea aleasă este evaluată mai slab ($E_s < E_c$), s poate fi totuşi selectată cu o anumită probabilitate. Probabilitatea depinde de doi factori: 1) de diferenţa de valoare dintre starea curentă c şi starea selectată s ($\Delta_E = |E_s - E_c|$) precum şi 2) de temperatura sistemului, rezultând

$$P = \frac{1}{e^{\frac{\Delta E}{T}}}$$

Iniţial temperatura este mai mare. Rezultă o probabilitate mai mare de selecţie a unei stări mai slab evaluate. Pe parcursul căutării temperatura scade conform unui program prestabilit. Această scădere conduce la o probabilitate mai mică de alegere a unei stări proaste pe măsură ce căutarea avansează.

(d) Algoritmul de *răcire controlată* porneşte cu (20s)

- √ **temperatură ridicată**
- ◯ temperatură joasă

Explicaţie: Temperatura iniţiala mare permite explorarea spaţiului de căutare. Pe măsură ce căutarea avansează, probabilitatea de selectare a unei stări mai slabe scade. Când termperatura ajunge la 0, algoritmul devine similar cu hill climbing.

(e) Algoritmul de *răcire controlată* (30s)

- ◯ nu permite selectarea stărilor cu utilitate mai mică decât cea curentă
- √ **permite selectarea stărilor mai proaste cu probabilitatea** $P = \frac{1}{e^{\frac{\Delta E}{T}}}$
- ◯ permite selectarea stărilor mai proaste cu probabilitatea $P = \frac{1}{e^T}$
- ◯ nicio variantă dintre acestea

Explicaţie: Probabilitatea depinde de doi factori: 1) de diferenţa de valoare dintre starea curentă c şi starea selectată s ($\Delta_E = |E_s - E_c|$), precum şi 2) de temperatura sistemului.

(f) *Local beam search* cu k stări este similar cu k algoritmi hill climbing rulaţi în paralel. (30s)

- ◯ Adevărat
- √ **Fals**

Explicaţie: În cazul *local beam*, căutarea începe cu k stări generate aleator. La fiecare pas sunt generaţi toţi succesorii celor k stări. Din această mulţime se selectează cei mai buni k succesori. Astfel, o stare din cele k poate contribui cu mai mulţi succesori în generaţia următoare. Acest lucru nu apare în cazul rulării în paralel a k algoritmi hill climbing.

(g) Care nu este un operator genetic în algoritmii genetici? (30s)

- √ **ADN**
- ◯ mutaţie
- ◯ încrucişare

◯ clonare

Explicaţie: Pentru a construi generaţia următoare, algoritmii genetici aplică următorii operatori pe indivizii din generaţia curentă: 1) mutaţie: gene din cromozom sunt modificate aleator; 2) încrucişare: secvenţe de gene de la un individ sunt încrucişate cu secvenţe de gene de la alt individ; 3) clonare: indivizii foarte buni pot fi copiaţi direct în generaţia următoare.

(h) Într-un nod de tip AND, ramificaţia este introdusă de (20s)

√ **alegerile mediului**

◯ alegerea facută de agent

Explicaţie: Arborii de căutare de tip AND-OR conţin două tipuri de noduri. Nodurile de tip OR reprezintă alegerile agentului. Agentul decide asupra unei singure acţiuni pe care o va executa. Nodurile de tip AND reprezintă răspunsurile posibile ale mediului nondeterministic. Agentul trebuie să trateze toate ramurile unui nod de tip AND.

(i) *Belief state* reprezintă: (30s)

√ **setul stărilor posibile în care un agent poate fi**

◯ spaţiul de căutare al unui agent religios

◯ cea mai probabilă stare în care agentul poate fi

◯ spaţiul de căutare al unui agent ateist

Explicaţie: Mulţimea include toate stările posibile. Acestor stări nu le sunt ataşate probabilităţi.

28. **Căutare adversarială**

(a) Minimax execută în arborele jocului o căutare de tipul (20s)

- √ **în adâncime**
- ◯ în lăţime
- ◯ cost uniform
- ◯ A*

Explicaţie: Algoritmul explorează complet (până la frunze) arborele jocului printr-o căutare în adâncime: pentru mutarea curentă se consideră un posibil răspuns al adversarului căruia i se calculează o mutare posibilă şi aşa mai departe.

(b) Ordinea de analiză a mutărilor nu afecteză eficienţa eliminării nodurilor (pruning) (30s)

- √ **fals**
- ◯ adevărat

Explicaţie: O bună ordonare a mutărilor ajută la oprirea mai rapidă a căutării în zonele care funcţia de evaluare ar întoarce valori mai mici decât cele dorite. Identificarea unei ordini bune de analiză a mutărilor este însă dificilă.

(c) Tăierea cu alpha-beta nu afectează rezultatul final (10s)

- √ **adevărat**
- ◯ fals

Explicaţie: Cu *alpha-beta pruning* se obţine acelaşi rezultat, dar în urma unei căutari mai reduse.

(d) În practică, pentru evaluarea poziţiei se utilizează o combinaţie de atribute (20s)

- ◯ liniară
- √ **nonliniară**

Explicaţie: Combinaţia liniară presupune că atributele considerate în evaluarea poziţiei sunt independente. De exemplu, la şah, valori posibile pentru cal, nebun şi damă sunt 3, 3, respectiv 10 pioni. Dar combinaţia cal-damă este mai puternică decât nebun-damă, ceea ce nu poate fi modelat printr-o combinaţie liniară.

(e) Căutarea de tip *quiescent* analizează poziţiile interesante la o adâncime mai mare (10s)

- ◯ fals
- √ **adevărat**

Explicaţie: Oprirea căutării la o adâncime constantă este riscantă datorită *efectului de orizont*. Presupunem că adâncimea de căutare este 8, iar în ultimul *ply* albul capturează un cal cu dama. Funcţia de evaluare aplicată pe această poziţie ar fi favorabilă albului. Poziţia nu este una linistită, deoarece este puţin probabil ca negrul să nu aibă un răspuns simetric prin care să captureze o piesă de valoare similară. O astfel de poziţie e recomandat sa fie analizată la o adâncime mai mare. Căutarea de tip *quiescent* aplică funcţia de evaluare doar pe poziţii liniştite. Poziţiile care nu sunt liniştite vor fi explorate la o adâncime mai mare.

(f) Ce se poate utiliza pentru a implementa *tăierea înainte (forward pruning)*? (30s)

- ◯ hill climbing
- √ **beam search**
- ◯ hill climbing stocastic
- ◯ răcire controlată (simulating annealing)

Explicaţie: *Tăierea înainte* presupune că o bună parte din mutările legale nu sunt analizate deloc. Agentul uman identifică imediat care mutări legale nu au nicio şansă să ducă la poziţii bune. Căutarea se aplică doar pe câteva mutări considerate promiţătoare. Algoritmul *beam search* funcţionează similar: la fiecare mutare se consideră doar un subset k din cele n mutări legale. Subsetul de k mutări promiţătoare este ales pe baza funcţiei de evaluare.

(g) Valoarea expectată a unei poziţii în jocuri stocastice este (30s)

- ◯ media tutoror rezultatelor posibile ale nodurilor de tip *min* şi *max*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *frunză*
- √ **media tutoror rezultatelor posibile ale nodurilor de tip** *şansă*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *şansă*

Explicaţie: Pentru calcularea valorii expectate se înmulţeşte câştigul posibil cu probabilitatea de obţinere a acestuia. Acest calcul se efectuează pentru fiecare instanţiere posibilă a nodurilor şansă.

(h) În jocuri stocastice, comportamenul se prezervă pentru orice transformare a funcţiei de evaluare care este (30s)

- ◯ monotonă
- ◯ nonmonotonă
- √ **liniar pozitivă**
- ◯ liniar negativă

Explicaţie: Monotonia nu este suficientă pentru a păstra ordinea între mutări. Fie o transformare monotonă pentru două poziţii succesoare de la evaluarea iniţială de 2 şi 3 la 20 respectiv 300. Probabilitatea de atingere a primei poziţii este 0.9, iar a celei de a doua poziţii 0.1. Valoarea expectată a poziţiei curente s_1 în primul caz este $2 \times 0.9 + 3 \times 0.1 = 2.1$. Valoarea expectată după transformare este $20 \times 0.9 + 300 \times 0.1 = 18 + 270 = 288$. Presupunem că există în arbore o altă poziţie s_2 evaluată la 5 inainte de transformare şi la 50 dupa transformare. Inainte de transformare s_2 pare mai bună. După transformare s_1 este mai bună. Dacă transformarea ar fi monoton liniară, ierarhia poziţiilor se păstrează.

(i) În jocuri stocastice, căutarea la adâncimi mari este esenţială (30s)

◯ adevărat

√ **fals**

Explicaţie: În jocurile stocastice factorul de ramificare este mare. Cu cât adâncimea este mai mare cu atât probabilitatea de atingere a acelei poziţii este mai mică. La adâncimi mari se analizează practic doar poziţii cu probabilităţi extrem de mici. La jocuri stocastice accentul cade nu pe analiza multor mutări în avans, ci pe construirea de funcţii corecte pentru evaluarea poziţiei.

29. **Probleme de satisfacere a constrângerilor**

(a) Algoritmul de backtracking poate fi îmbunătăţit prin: (30s)

- ◯ nu poate fi imbunătăţit!
- ◯ sortarea variabilelor în ordine lexicografică
- √ **detectarea timpurie a eşecului inevitabil**
- √ **exploatând structura problemei**

> **Explicaţie:** Prin detectarea timpurie a eşecului se elimină instanţierea unor variabile cu valori pentru care nu există soluţie. Prin explorarea structurii problemei se poate reduce domeniul de valori al unei variabile. În ambele cazuri, arbore rezultat în urma backtrackingului va fi mai mic.

(b) Algoritmul de backtracking poate beneficia de euristici în alegerea (30s)

- ◯ domeniului unei variabile
- ◯ constrângerilor care să fie eliminate
- √ **variabilei care se instanţiază la pasul următor**
- √ **valorii din domeniu care se asignează variabilei curente**

> **Explicaţie:** Atât domeniul variabilelor cât şi constrângerile sunt date de intrar eîn problema, deci nu pot fi modificate. Ordinea în care se analizeaza variabilele, respectiv ordinea în care se asignează valori din domeniu variabilei selectate influenţează semnificativ performanţa căutării.

(c) Euristica *valorile minime rămase* selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- √ **variabila cu cele mai puţine valori disponible**
- ◯ valoarea care elimină cele mai puţine valori ale variabilor învecinate
- ◯ valorile care introduc cele mai multe constrângeri în variabile învecinate

> **Explicaţie:** Sunt două tipuri de euristici în acest context: euristici care selectează variabila, respectiv euristici care selecteză valori din domeniul unei variabile. Euristica *valorile minime rămase* ajută la selectarea variabilei curente. Euristica are rolul de a selecta variabila cu domeniul de valori posibile cel mai mic la momentul curent. Se urmăreşte maximizarea şanselor de a putea asigna valori variabilelor rămase neinstanţiate. Variabilele învecinate unei variabile X se referă la varibilele cu care X are relaţii de constrângere. Modificarea domeniului variabilei X se propagă domeniilor variabilelor învecinate.

(d) Euristica grad selectează (30s)

- √ **variabilele cu cele mai multe constrângeri pe variabile rămase**
- ◯ variabila cu cele mai puţine valori disponibile

◯ valoarea care elimină cele mai puţine valori ale variabilelor învecinate
◯ valorile care introduc cele mai multe constrângeri în variabilele învecinate

Explicaţie: Euristica *grad* ajută la selectarea variabilei curente. Prin selectarea variabilei cu cele mai multe constrângeri se urmăreşte identificarea eşecului inevitabil cât mai curând posibil.

(e) Euristica *cele mai puţine constrângeri* selectează (30s)

◯ variabilele cu cele mai multe constrângeri pe variabile rămase
◯ variabila cu cele mai puţine valori disponibile
√ **valoarea care elimină cele mai puţine valori ale variabilelor învecinate**
◯ valorile cure introduc cele mai multe constrângeri în variabilele învecinate

Explicaţie: Euristica urmăreşte selectarea unei valori din domeniul variabilei deja selectate. Deci (1) şi (2) nu se aplică. Prin selectarea valorii care afectează cel mai puţin vecinii se doreşte maximizarea şanselor de a putea asigna valori variabilelor rămase neinstanţiate.

(f) Pentru a detecta eşecul cât mai repede posibil se utilizează euristica (30s)

√ **valorile minime rămase**
◯ grad
◯ cele mai puţine constrângeri

Explicaţie: Eşecul apare în momentul în care domeniul unei variabile este nul. Euristica *valorile minime rămase* analizează în primul rând variabilele cu cel mai mic domeniu.

(g) Pentru a reduce factorul de ramificare se utilizează (30s)

◯ valorile minime rămase
√ **grad**
◯ cele mai puţine constrângeri

Explicaţie: Euristica *grad* alege variabila cu cele mai multe constângeri. La instanţierea acesteia, constrângerile rezultate se propagă în cele mai multe variabile învecinate. Prin reducerea domeniului a cât mai multor variabile se micşoreaza factorul de ramificare al algoritmului de backtracking. Scopul este de a reduce cât mai mult domeniul variabilelor prin propagarea constrângerilor înainte de aplicarea backtrackingului.

(h) Verificarea înainte (forward checking) evită detectarea târzie a conflictului (20s)

◯ adevărat

◯ fals

Explicaţie: Verificarea înainte reduce domeniul variabilelor neintanţiate prin eliminarea valorilor pentru care nu există soluţie. Pentru respectivele valori, algoritmul de backtracking ar identifica eşecul doar după instanţierea variabilelor.

(i) $X, Y, Z \in \{0, 1, 2, 3\}, X < Y < Z$ (30s)

◯ $X \in \{0, 1, 2\}$
◯ $X \in \{2, 3\}$
◯ $X \in \{1, 2\}$
√ $X \in \{0, 1\}$

Explicaţie: $Y < Z$ implică $Y \in \{0, 1, 2\}$. $X < Y$ implică $X \in \{0, 1\}$.

(j) Bound propagation $D_1 = [0, 5], D_2 = [0, 8], D_1 + D_2 = 10$ (30s)

◯ $D_1 = [5, 5], D_2 = [2, 8]$
◯ $D_1 = [0, 5], D_2 = [2, 8]$
√ $D_1 = [2, 5], D_2 = [5, 8]$
◯ nicio variantă

Explicaţie: Înainte de instanţierea variabilelor se urmăreşte reducerea pe cât posibil a domeniului acestora. Daca D_1 este maximă ($D_1 = 5$), atunci D_1 are valoarea minimă posibilă 5. Domeniul variabilei devine acum $D_2 = [5, 8]$. Daca D_2 este maximă ($D_2 = 8$), atunci D_2 are valoarea minimă posibilă 2. Domeniul variabilei devine acum $D_1 = [2, 5]$.

30. **Agenţi logici**

(a) Lumea lui Wumpus este (30s)

- ◯ complet observabilă şi determinstă
- ◯ episodică şi observabilă local
- √ **statică şi include un singur agent**
- ◯ multi-agent şi secvenţială

Explicaţie: (1) Greşită deoarece agentul are acces la informaţii doar din căsuţele învecinate. Deci mediul nu este complet observabil. (2) Greşită deoarece acţiunea curentă a agentului depinde de acţiunile anterioare. Deci mediul nu este episodic. (3) Corectă. Mediul este static deoarece nu se schimbă în timp ce agentul deliberează. Wumpus nu este considerat agent deoarece nu îşi adaptează comportamentul în funcţie de acţiunile personajului. (4) Greşită deoarece sistemul include un singur agent.

(b) Care expresie nu este satisfiabilă? (30s)

- ◯ $\neg a \vee b$
- ◯ $a \to \neg a$
- √ $a \vee \neg a \to a \wedge \neg a$
- ◯ $a \wedge \neg a \vee b$

Explicaţie: (1) Satisfiabilă cu $a = 0$ şi $b = 1$. (2) Satisfiabilă cu $a = 0$. (3) Nesatisfiabilă. Având o singură variabilă există două lumi posibile. Pentru $a = 0$ avem $1 \to 0 \equiv 0$. Pentru $a = 1$ avem $1 \to 0 \equiv 0$. (4) Satisfiabilă pentru $b = 1$.

(c) $KB \vDash \alpha$ dacă şi numai dacă (20s)

- √ $M(KB) \subseteq M(\alpha)$
- ◯ $M(\alpha) \subseteq M(KB)$

Explicaţie: Pentru a deduce propoziţia α din baza de cunoştinţe KB ($KB \vDash \alpha$) mulţimea modelelor bazei de cunoştinte trebuie să fie inclusă în mulţimea modelelor în care α este adevarată.

(d) Propoziţia $p \to q \equiv \neg q \to \neg p$ este (30s)

- ◯ falsă
- √ **adevarată**
- ◯ depinde doar de p
- ◯ depinde doar de q

Explicaţie: Prin eliminarea implicaţiei obţinem în partea stângă $\neg p \vee \neg q$, iar în partea dreaptă $\neg q \vee \neg p$. Cele două disjuncţii sunt echivalente $\forall p, q$.

(e) Contrapoziţia spune că: (30s)

√ $(a \to b) \equiv \neg b \to \neg a$

◯ $\neg(\neg a) \equiv a$

◯ $\neg(a \wedge b) \equiv (\neg a \vee \neg b)$

◯ $\neg(a \vee b) \equiv (\neg a \wedge \neg b)$

Explicaţie: Variantele prezentate reprezintă următoarele axiome: (1) Contrapoziţie; (2) Dublă negaţie; (3, 4) de Morgan.

(f) Tabelul de adevăr al implicaţiei conţine (10s)

◯ 4 de zero

◯ 3 de zero

◯ 2 de zero

√ **1 singur zero**

Explicaţie: Implicaţia este falsă doar pentru $1 \to 0$.

(g) Care din următoarele propoziţii sunt tautologii? (20s)

◯ $a \wedge \neg a$

√ $a \vee \neg a$

√ $a \vee \neg a$

◯ $\neg a$

Explicaţie: O propoziţie este tautologie dacă are valoarea 1 indiferent de instanţierile posibile ale variabilelor. (1) are valoarea 0 pentru $a = 0$; (2) tautologie; (3) prin eliminarea implicaţiei se obţine $\neg a \vee a$. (4) are valoarea 0 pentru $a = 1$.

(h) Care din următoarele nu este o clauză Horn? (10s)

◯ $a \to b$

◯ $a \to \neg b$

◯ $a \to b \wedge c$

√ $a \to b \vee c$

Explicaţie: O clauză Horn este o disjuncţie de literali cu maxim un literar pozitiv. Prin eliminarea implicaţiei se obţine: (1) $\neg a \vee b$. Doar b pozitiv, deci este clauză Horn. (2) $\neg a \vee b$ este o clauza Horn. Totuşi, o astfel de expresie nu poate apărea în Prolog. Prologul conţine doar clauze Horn definite. Acestea sunt clauze Horn cu exact un literar pozitiv. (3) $\neg a \vee (b \wedge c)$ nu este o clauză Horn deoarece nu e disjuncţie. Expresia reprezintă o *formulă Horn*. Formula

Horn este o formula în forma normal conjunctivă în care toate clauzele sale sunt clauze Horn. Într-adevăr $\neg a \vee (b \wedge c)$ poate fi convertită la $(\neg a \vee b) \wedge (\neg a \vee c)$. Prin introducerea implicaţiei formula iniţială poate fi transformă în două clauze Horn: $(a \rightarrow b)$ şi $(a \rightarrow c)$. (4) $\neg a \vee b \vee c$ conţine 2 literari pozitivi. Astfel de formule pot fi modelate în programarea logică disjunctivă.

(i) Raţionarea înainte este (30s)

◯ ghidată de ţel

√ **ghidată de date**

Explicaţie: Raţionarea înainte porneşte de la faptele existente pentru a identifica care reguli se pot activa.

(j) Care expresie este în forma normal conjunctivă? (10s)

◯ $a \rightarrow b$

√ $a \wedge b$

◯ $\neg a \vee b$

◯ $\neg(a \wedge b)$

Explicaţie: Forma normal conjunctivă (CNF) este o conjuncţie de disjuncţii. 'In astfel de expresii nu apar implicaţii, respectiv negaţia apare doar pe literari atomici. (1) Nu este în CNF deaorece apare implicaţia; (2) Este în CNF; (3) Este o disjuncţie în loc de conjuncţie; (4) Negaţia apare pe expresii şi nu pe atomi.

(k) Pentru a demonstra α, rezoluţia arată că $KB \wedge \neg\alpha$ este (10s)

√ **nesatisfiabilă**

◯ satisfiabilă

Explicaţie: Rezoluţia foloseşte metoda reducerii la absurd. Pentru a demonstra o propoziţie, rezoluţia arată că propoziţia negată introduce o inconsistenţă în baza de cunoştinţe. Altfel spus, baza de cunoştinţe împreună cu propoziţia negată (i.e. $KB \wedge \neg\alpha$) conduc la o formulă nesatisfiabilă.

31. **Logica de ordinul întâi**

(a) Ce este FOL? (10s)

- ◯ First Order Language
- √ **First order logic**
- ◯ Focus on logic
- ◯ First order (Star Wars)

> **Explicaţie:** Alte denumiri pentru logica de ordinul întâi sunt *logica predicatelor* sau *calculul predicatelor de ordinul întâi.*

(b) Predicatul $between(x, y)$ este (20s)

- ◯ unar
- √ **binar**
- ◯ ternar

> **Explicaţie:** Aritatea unui predicat este dată de numărul său de argumente.

(c) Logica de ordinul include (20s)

- √ **obiecte, relaţii, funcţii**
- ◯ timp, credinţe, cunoştinţe
- ◯ constrângeri
- ◯ aceeaşi expresivitate cu logica propoziţională

> **Explicaţie:** De exemplu, logica propoziţională nu include variabile sau cuantificatori.

(d) *Fiecăruia îi place îngheţata* este formalizată în FOL (20s)

- ◯ $place(x, inghetata)$
- ◯ $place \wedge inghetata$
- ◯ $\exists x, place(x, inghetata)$
- √ $\forall x, place(x, inghetata)$

> **Explicaţie:** Prima variantă este greşită deoarece în FOL toate variabile trebuie cuantificate. A doua variantă reprezintă o formalizare în logica propoziţională, logică care nu include variabile. În a treia varianta variabila x este cuantifică existenţial şi nu universal ca în varianta corectă.

(e) *Există o persoană care iubeşte pe toată lumea* este formalizată în FOL (20s)

- ◯ $\forall x\ \exists y, iubeste(x, y)$
- √ $\exists x\ \forall y, iubeste(x, y)$

◯ $\forall x\ iubeste(x, Persoana)$
◯ $\exists x\ iubeste(x, ToataLumea)$

Explicaţie: Prima variantă este formalizarea propoziţiei: *Toată lumea iubeşte pe cineva*. Variantele 3 şi 4 includ constante: fiecare constantă referă un singur obiect.

(f) *Brothers are siblings* este formalizată în FOL (20s)
◯ $\exists x\ \exists y Brother(x, y) \rightarrow Sibling(x, y)$
◯ $\forall x\ \exists y Brother(x, y) \wedge Sibling(x, y)$
◯ $Brother(x) \rightarrow Sibling(y)$
√ $\forall x\ \forall y Brother(x, y) \rightarrow Sibling(x, y)$

Explicaţie: Greşeala din prima variantă provine de la utilizarea cuatificatorului existenţial în locul celui universal. Varianta a doua foloseşte conjuncţia în locul implicaţiei. În varianta a treia, variabile x şi y sunt libere - nu se va realiza nicio potrivire între acestea.

(g) *Oricine de la UTCN este isteţ* este formalizată în FOL (20s)
◯ $\forall x\ la(x, UTCN) \wedge istet(x)$
◯ $istet(x, TUCN)$
√ $\forall x\ la(x, UTCN) \Rightarrow istet(x)$
◯ $\exists x\ la(x, UTCN) \Rightarrow istet(x)$

Explicaţie: În multe cazuri, variabile cuantificate universal sunt folosite în implicaţii, iar cele cuantificate existenţial în conjuncţii.

(h) Relaţia *sibling* este simetrică (20s)
◯ $Sibling(x)$
√ $\forall x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$
◯ $Sibling(x, y) \vee Sibling(y, x)$
◯ $\exists x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$

Explicaţie: În (1) şi (3) variabilele nu sunt cuantificate. În (4) cuantificatorul existenţial este folosit în locul celui corect ($\forall$).

(i) *O mamă este părintele de sex feminin* (30s)
◯ $\exists x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
√ $\forall x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
◯ $\forall x, y\ Mother(x, y) \equiv Parent(x, y)$

◯ $\forall x, y\ Mother(x, y) \wedge (Female(x) \wedge Parent(x, y))$

Explicaţie: Definiţia trebuie să fie validă pentru toate mamele, fiind necesară utilizarea cuantificatorului universal.

(j) *Oricine iubeşte pe cineva* este formalizată în FOL (20s)

√ $\forall x \exists y, iubeste(x, y)$

◯ $\exists x \forall y, iubeste(x, y)$

◯ $\forall x \exists y, iubeste(y, x)$

◯ $\exists x \forall y, iubeste(y, x)$

Explicaţie: (2) este formalizarea pentru *Cineva iubeşte pe toată lumea.* (3) este formalizarea propoziţiei *Pentru orice persoană există cineva care să iubească acea persoana.* (4) este formalizarea pentru *Există o persoană iubită de toată lumea.* Reţineţi că ordinea cuantificatorilor este importantă.

32. **Inferenţă în logica de ordinul întâi**

(a) Un termen *ground* este (20s)

- ◯ un termen cu cel puţin o constantă
- √ **un termen fără variabile**
- ◯ un sol unde un arbore binar poate fi plantat
- ◯ niciuna dintre aceste opţiuni

Explicaţie: Un termen *ground* este un termen fără variabile. Un astfel de termen include: (1) toate constantele, (2) funcţii aplicate pe constante, 3) respectiv funcţii aplicate de număr finit de ori pe (1) sau (2). A se distinge între termen *ground* şi predicat *ground*. Un predicat (sau literal) ground are toate argumentele termeni *ground*.

(b) *Lifting* înseamnă (20s)

- √ **transformarea din logică propoziţională în logica de ordinul întâi**
- ◯ eliminarea constantelor Skolem
- ◯ introducerea constantelor Skolem

Explicaţie: Algoritmii de raţionare în logica propoziţională (înlănţuire înainte, înlănţuire înapoi şi rezoluţie) au variante urcate ($lifted$) în logica de ordinul întâi. Modus ponens generalizat (GMP) este modus ponens (MP) în care s-au introdus variabile: $GMP = lifted\ MP$. Operaţia inversă de transformare din logica de ordinul întâi în logica propoziţională se realizezează prin instanţierea cuantificatorilor universal (UI) şi existenţial (EI). Operaţia de eliminare a cuantificatorului existenţial se numeşte skolemizare. Variabilele cuantificate existenţial sunt înlocuite cu constante Skolem sau funcţii Skolem.

(c) Datalog = (10s)

- ◯ Prolog + funcţii
- ◯ baze de date extinse cu reguli
- ◯ clauze în logica de ordinul întâi plus funcţii
- √ **clauze în logica de ordinul întâi fără funcţii**

Explicaţie: In Datalog sunt permise clauzele define dar fără funcţii. Algoritmul de inferenţă este complet.

(d) O bază de date deductivă (20s)

- √ **utilizează inferenţa pentru a răspunde la interogări**
- √ **combină programarea logică cu baze de date relaţionale**
- √ **mai expresivă decât o bază de date, dar mai puţin decât un limbaj logic**

Explicaţie: Datalog este un limbaj utilizat în bazele de date deductive.

(e) Skolemizarea este procesul de (10s)

- ◯ eliminare a cuantificatorilor universali
- √ **eliminare a cuantificatorilor existenţiali**
- ◯ introducere a cuantificatorilor universali
- ◯ introducere a cuantificatorilor existenţiali

Explicaţie: În urma skolemizării, variabile cuantificate existenţial sunt înlocuite cu constante Skolem sau funcţii Skolem, iar cuantificatorul existenţial dispare.

(f) Logica de ordinul întâi a fost inventată de: (10s)

- ◯ Aristotel (384-322 BC)
- ◯ Boole (1815-1864)
- ◯ Ludwig Wittgenstein (1889-1951)
- √ **Gottlob Frege (1848-1925)**

Explicaţie: Dezvoltarea logicii de ordinul întâi s-a datorat introducerii de către Frege a cuantificatorilor universal şi existenţial.

(g) În Prolog nu există nicio modalitate de a aserta un fapt negativ (e.g. $\neg father(a, b)$) (10s)

- ◯ Fals
- √ **Adevărat**

Explicaţie: Prologul utilizează clauze Horn definite (i.e. exact un literar pozitiv). Faptele negative nu se încadrează în aceasta definiţie. În aceeaşi linie, concluziile clauzelor nu pot fi negate.

(h) În forma normal conjunctivă, fiecare clauză conţinută este o (20s)

- √ **disjuncţie de literari**
- ◯ conjuncţie de literari

Explicaţie: Forma normal conjunctivă conţine clauze legate prin operatorul AND ($\wedge$). În schimb, aceaste clauze sunt formate din literari legaţi doar prin disjuncţii ($\vee$).

(i) Care nu este o metodă de inferenţă logică? (10s)

- ◯ raţionarea înainte
- ◯ raţionarea înapoi
- √ **skolemizarea**

◯ rezoluţia

Explicaţie: Skolemizarea reprezintă etapa de eliminare a cuantificatorilor existenţiali. Etapa apare în momentul conversiei unei formule la forma normal conjunctivă. Această formă este necesară în mecanismul de rezoluţie. Alături de rezoluţie, raţionarea înainte şi înapoi sunt metode prin care se pot demonstra concluzii.

(j) Utilizări practice ale demonstratoarelor de teoreme includ (20s)

√ **verificare hardware**

√ **verificare software**

Explicaţie: Pe de o parte, specificaţiile harware sau software sunt translatate ca teoreme. Pe de altă parte, modelul hardware sau software este formalizat şi interpretat ca axiome. Demonstrarea teoremei pe baza axiomelor reprezintă verificarea ca sistemul respectă specificaţiile.

33. **Planificare clasică**

(a) PDDL provine de la (10s)

- ◯ Planning Domain Description Language
- ◯ Planning Description Domain Language
- √ **Planning Domain Definition Language**
- ◯ nicio variantă din cele de mai sus

Explicaţie: Limbajul de definire a problemelor de planificare (PDDL) a fost dezvoltat în 1998, fiind limbajul folosit la IPC (International Planning Competition).

(b) O condiţie deschisă este o precondiţie (10s)

- ◯ cu un termen care nu are toate variabilele instanţiate
- ◯ cu prezumţia de lume deschisă
- √ **a unui pas fără legături cauzale**

Explicaţie: Algoritmul de planificare cu relaţii de ordine parţială menţine un set de precondiţii *deschise* conţinând acele precondiţii care nu sunt îndeplinite de nicio acţiune la momentul curent.

(c) Un *clobber* (20s)

- ◯ validează o precondiţie obţinută datorită unui efect al unei acţiuni
- √ **distruge o precondiţie obţinută printr-o legătură cauzală**
- ◯ introduce o relaţie de ordine parţială între acţiuni

Explicaţie: Dupa identificarea acestor cazuri de eliminare a legăturilor cauzale, acţiunile de acest tip (*clobber*) sunt promovate sau retrograte în planul curent pentru ca efectele lor (i.e., stergerea legăturilor cauzele construite până la momentul curent) să nu aibă loc.

(d) Anomalia Sussman (20s)

- √ **ilustrează dezavantajele planificării greedy**
- ◯ opreşte algoritmul de planificare, chiar dacă un plan există.
- √ **poate fi rezolvată cu ajutorul** *clobberilor*

Explicaţie: Fiind dat un ţel g acesta poate fi împărţit în două subobiective g_1 şi g_2. Presupunem că în timpul căutării s-a identificat un plan pentru g_1. Anomalia Sussman apare când nu s epoate avansa spre obţinerea ţelului global g fără a şterge planul obţinut pentru g'.

(e) O acţiune este relevantă dacă (20s)

◯ poate fi următorul pas într-un plan

√ **cel puţin unul din efecte se unifică cu un element din starea finală**

◯ are toate precondiţiile satisfăcute

Explicaţie: Acţiunile *relevante* sunt cele care au ca efect cel puţin unul din subţelurile obiectivului din starea finală. Acţiunile care au toate precondiţiile îndeplinite se numesc *aplicabile*.

(f) Care sunt euristici admisibile pentru planificare? (20s)

√ h_1= **ignorarea precondiţiilor**

◯ h_2= ignorarea stării iniţiale

√ h_3 = **ignorarea efectelor care elimină fapte din starea curentă**

◯ h_4= ignorarea stării finale

Explicaţie: h_1 şi h_3 lucrează pe o simplificare a problemei iniţile. De aici admisibilitatea.

(g) Ce se elimină pentru h= *Manhattan* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

√ $blank(s2)$

◯ $adj(s1, s2)$

◯ $on(t, s1)$

◯ toate precondiţiile

Explicaţie: În jocul culisant distanţa Manhattan reprezintă suma distanţelor de la fiecare piesă la poziţia ei finală. Distanţa Manhattan reprezintă problema relaxată în care nicio altă piesă nu incomodează traseul de la poziţia curentă la cea finală. Singura condiţie necesară pentru efectuarea mutării este ca locaţiile să fie adiacente (i.e. $adj(s1, s2)$). Astfel, pentru a ajunge într-o stare succesoare nu este necesar ca aceasta să fie liberă (i.e. $blank(s2)$).

(h) Ce trebuie eliminat pentru h = *numărul de piese plasate greşit* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

◯ $blank(s2), on(t, s1)$

√ $blank(s2), adj(s1, s2)$

◯ $on(t, s1), tile(t)$

◯ toate precondiţiile

Explicaţie: În jocul culisant euristica h = *numărul de piese plasate greşit* cuantifică numărul de piese care nu se află încă pe poziţiile specificate în ţel. Euristica h repreintă problema relaxată în care o piesă poate fi scoasă de pe tablă şi

aşezată direct pe poziţia finală. Pentru a se executa o astfel de acţiune nu sunt necesare precondiţiile ca locaţia succesoare să fie neocupată de o altă piesă (i.e. $blank(s2)$) şi nici adiacentă cu poziţia curentă (i.e $adj(s1, s2)$).

34. **Planificare în lumea reală**

(a) Complexitatea algoritmului căii critice este (20s)

- ◯ $O(n)$
- ◯ $O(1)$
- √ $O(Nb)$
- ◯ $O(N^b)$

> **Explicaţie:** Aici N reprezintă numărul de acţiuni, iar b este factorul maxim de ramificare al acţiunilor (indiferent dacă reprezintă intrare sau ieşire). Operaţiile LS şi ES sunt efectuate o singură dată pentru fiecare acţiune. Fiecare astfel de calcul este iterat pentru fiecare intrare/iesire din acţiunea curentă.

(b) $ES(B) =$ (30s)

- √ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- ◯ $min_{B \succ A} LS(B) - Duration(A)$

> **Explicaţie:** $ES(B)$ reprezintă cel mai devreme moment la care acţiunea B poate începe. Dintre toate acţiunile A premergătoare lui B (i.e. $A \prec B$) se alege cea cu ES maxim. La această valoare se adaugă durata acţiunii A selectate. Rezultă $ES(B) = max_{A \prec B} ES(A) + Duration(A)$.

(c) $LS(A) =$ (30s)

- ◯ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- √ $min_{B \succ A} LS(B) - Duration(A)$

> **Explicaţie:** $LS(A)$ reprezintă cel mai târziu moment la care acţiunea A poate începe. Dintre toate acţiunile B care îi urmează lui A (i.e. $B \succ A$) se alege cea cu LS minim. Din această valoare se scade durata acţiunii curente A. Rezultă $LS(A) = min_{B \succ A} LS(B) - Duration(A)$.

(d) Euristica *minimum slack* (20s)

- √ **este similară heuristicii** *numărul de valori minime* **specifică CSP**
- ◯ garantează obţinerea soluţiei optime
- ◯ nicio variantă din cele de mai sus

Explicaţie: Euristica *marja minimă (minimum slack)* alege acţiunea care poate fi începută cel mai drevreme dintre toate acţiunile cu precondiţiile îndeplinite şi care are cea mai mică marjă temporală. Euristica nu garantează soluţia optimă. Similar cu euristica *numărul de valori minime* (MRV) din problemele de satisfacere a constrângerilor (CSP) este selectată variabila (în cazul nostru acţiunea) care are domeniul cu cele mai puţine valori (în cazul nostru marja cea mai mică de alegere a momentului de început a acţiunii).

(e) Este mai bine să avem efecte condiţionate decât o acţiune care nu se poate aplica (20s)

- ◯ întotdeauna
- ◯ niciodată
- √ **în cazul planificării fără senzori**
- ◯ în cazul planificării contingente

Explicaţie: O acţiune nu se poate aplica dacă are cel puţin o precondiţie neîndeplinită În cazul planificării fără senzori, în lipsa informaţiilor din mediu, îndeplinirea sau nu a unor precondiţii nu poate fi validată. Pentru a se putea genera un plan se preferă definirea de acţiuni fără precondiţii. În schimb, efectele acţiunii vor fi avea loc doar dacă anumite condiţii sunt îndeplinite.

(f) Schemele de percepte sunt utilizate în (20s)

- ◯ planificarea fără senzori
- √ **planificarea contingentă**
- ◯ atât planificarea fără senzori, cât şi cea contingentă
- ◯ nicio opţiune din cele anterioare

Explicaţie: Planificarea cu contingenţe se aplică în medii parţial observabile şi nondeterministe. Agentul are nevoie să simtă starea în care a ajuns în urma unui răspuns nondeterminist al mediului. Agentul include în planul său printre acţiunile clasice din PDDL şi acţiuni de tipul percepte. Schemele de percepte extind PDDL şi au nevoie de date de la senzori pentru a fi activate.

(g) Pentru identificarea succesului accidental (serendipity) agentul monitorizează din mediu: (20s)

- ◯ acţiunile
- √ **planul**
- ◯ ţelul

Explicaţie: Procesul de monitorizare a acţiunilor cere ca agentul să verifice dacă precondiţiile următoarei acţiuni din plan sunt încă îndeplinite. Fără o

astfel de verificare agentul se bazează doar pe computaţiile făcute la momentul derivării planului. Monitorizarea planului cere ca agentul să verifice inainte de executarea următoarei acţiuni dacă intreg planul este încă realizabil. Într-un mediu nondeterminist, o precondiţie necesară unei acţiuni viitoare poate să nu mai fie îndeplinită la momentul curent. Monitorizarea ţelului impune agentului ca înainte de executarea acţiunii următoare să verifice dacă nu este disponibil între timp un set de ţeluri mai bine evaluat (conform funcţiei de utilitate a agentului). Apariţia unui ţel mai bun face ca planul curent să fie înlocuit de unul care are ca ţintă noile obiective.

(h) La pierderea semnalului GPS, navigatorul e mai bine să execute (20s)

- ◯ replanificare
- ◯ repararea planului
- √ **depinde de lungimea planului şi perioadă de pierdere a semnalului**
- ◯ se opreşte şi asteaptă instrucţiuni de la agentul uman

Explicaţie: Deoarece posibila deviaţie de la planul curent este relativ mică raportată la lungimea planului, este de preferat o simplă reparare (i.e. generarea unui plan mic care are ca ţel una din stările traseului iniţial). Dacă pierderea semnalului a fost pe o perioadă semnficativă, poziţia curentă a agentului poate fi la o distanţă mare fată de de planul iniţial. Se poate identifica un traseu pornind din starea curentă care să nu aibă puncte comune cu traseul iniţial şi care să fie mai bun decât planul pentru readucere la traseul iniţial plus finalizarea acestuia. În acest caz se recomandă replanificare. “in acest scenariu cu GPS replanificarea este mai sigură fiind dat costul redus în timp al planificarii reportat la timpul pierdut executării unui plan neoptim.

(i) Acţiunile concurente sunt necesare în cazul (20s)

- ◯ monitorizării planului
- ◯ planificării cu contingenţe
- √ **planificării multi-agent**
- ◯ planificării online

Explicaţie: Pentru a se sincroniza în executarea unei acţiuni comune, este necesar ca limbajul să permită în sisteme multi-agent modelarea acţiunilor concurente.

(j) În cazul informaţiilor incomplete se utilizează (20s)

- √ **planificarea condiţională**
- √ **planificarea cu contingeţe**

◯ planificarea fără senzori
◯ monitorizarea execuţiei şi replanificare

Explicaţie:

(k) În cazul informaţiilor eronate se utilizează (20s)

◯ planificarea conditională
◯ planificarea contingentă
◯ planificarea fără senzori
√ **monitorizarea execuţiei şi replanificare**

Explicaţie: Planificarea cu contingenţe (sau condiţională) se aplică în medii parţial observabile şi nondeterministe.

35. **Calculul evenimentelor**

(a) Care nu este o sarcină de raţionare în calculul evenimentelor? (30s)

◯ postdicţia
√ **rezoluţia**
◯ abducţia
◯ predicţia

Explicaţie: Rezoluţia este o metodă de a demonstra propoziţii prin reducerea la absurd.

(b) Se poate face planificare în calculul evenimentelor prin (30s)

◯ predicţie
√ **abducţie**
◯ postdicţie
◯ doar limbajul PDDL poate fi utilizat pentru planificare

Explicaţie: Fiind date starea iniţială şi cea finală, abducţia poate fi utilizată pentru a genera posibilele acţiuni care pot explica cum s-a ajuns în starea finală pornind dintr-o stare iniţială dată.

(c) $\neg HoldsAt(Awake(Nathan), 0)$ este (30s)

√ **o observaţie**
◯ axiomă cu efect pozitiv
◯ axiomă cu efect negativ
◯ apariţia unui eveniment

Explicaţie: Predicatul *HoldsAt* specifică valori unor fluenţi la momente diferite de timp.

(d) $Happens(Awake(Nathan), 1)$ (30s)

◯ o observaţie
◯ axiomă cu efect pozitiv
◯ axiomă cu efect negativ
√ **apariţia unui eveniment**

Explicaţie: Predicatul *Happens* semnalizează apariţia unor evenimente.

(e) $HoldsAt(Ocupa(p_1, s), t) \wedge HoldsAt(Occupies(p_2, s), t) \rightarrow p_1 = p_2$ (30s)

√ *Ocupa* **este o relaţie injectivă**
◯ *Ocupa* este o relaţie surjectivă

Explicaţie: Dacă obiectele p_1 şi p_2 ocupă acelaşi spaţiu s la momentul t, atunci cele două obiecte sunt identice. Aceeaşi valoare a funcţiei nu poate sa aibă două intrări diferite. Deci relaţia este injectivă.

(f) $\neg HoldsAt(On(o, o), t)$ (30s)

◯ On este o relaţie reflexivă

√ On **este o relaţie ireflexivă**

Explicaţie: Orice obiect o nu poate să fie pe el însuşi. Relaţia on este deci ireflexivă.

(g) $\neg HoldsAt(Ringing(p, p), t)$ (30s)

◯ un telefon nu poate suna un alt telefon

√ **un telefon nu se poate suna pe el însuşi**

◯ telefonul p nu sună la timpul t

Explicaţie: $Ringing(p, p)$ este un fluent specificat ca fals la momentul t.

(h) $\neg HoldsAt(Broken(d), t) \rightarrow Initiates(TurnOn(a, d), On(d), t)$ (30s)

√ **precondiţie de tip fluent**

◯ precondiţie de tip acţiune

Explicaţie: Dacă precondiţia $\neg HoldsAt(Broken(d), t)$ are loc, atunci se poate executa acţiunea $TurnOn(a, d)$. Efectul acestei acţiuni va fi $On(d)$. Precondiţia este specificată prin $HoldsAt$, deci $Broken(d)$ este un fluent. Altfel spus, dacă fluentul $Broken(d)$ are loc la momentul t, atunci acţiunea se poate executa.

(i) $Happens(WalkThroughDoor(a, d), t) \rightarrow HoldsAt(Near(a, d), t)$ (30s)

◯ precondiţie de tip fluent

√ **precondiţie de tip acţiune**

Explicaţie: Precondiţia $Happens(WalkThroughDoor(a, d), t)$ este de tip acţiune. Apariţia evenimentului $WalkThroughDoor(a, d)$ duce la apariţia fluentului $Near(a, d)$ la momentul t.

36. **Probabilităţi condiţionale**

	Joacă fotbal	Nu joacă	Total
Băieţi	42	33	75
Fete	12	23	35
Total	54	56	110

(a) Care este probabilitatea ca o persoană selectată aleator să fie baiat? (20s)

- ◯ 42/110
- ◯ 54/110
- √ **75/110**
- ◯ 42/75

Explicaţie: Din definiţia probabilităţii: cazuri favorabile (75) supra cazuri posibile (110).

(b) $P(baiat|joacaFotbal) =$ (20s)

- ◯ 42/110
- √ **42/54**
- ◯ 42/75
- ◯ 54/110

Explicaţie: Din 54 jucători de fotbal, 42 sunt băieţi.

(c) $P(baiat \wedge joacaFotbal)=$ (20s)

- √ **42/110**
- ◯ 42/54
- ◯ 42/75
- ◯ 54/110

Explicaţie: Sunt 42 băieţi care joacă fotbal din totalul de 110 persoane.

(d) $P(baiat \vee joacaFotbal)=$ (20s)

- ◯ 42/110
- ◯ 54/110
- ◯ 75/110
- √ **87/110**

Explicaţie: Cazurile favorabile sunt date de 75 băieţi la care se adaugă 12 fete.

(e) Sunt variabilele *baiat* şi *joacaFotbal* independente? (60s)

◯ Da

√ **Nu**

Explicaţie:
$P(baiat) = 75/110$,
$P(joacaFotbal) = 54/110$,
$P(baiat \wedge joacaFotbal) = 42/110$
Deoarece $P(baiat \wedge joacaFotbal) \neq P(baiat)P(joacaFotbal)$, variabile nu sunt independente.

(f) Care este probabilitatea de a extrage o carte roşie sau un as dintr-un pachet? (20s)

◯ 16/50

◯ 1/2

◯ 15/26

√ **7/13**

Explicaţie:

$$\begin{aligned} P(rosu \vee as) &= P(rosu) + P(as) - P(rosu)P(as) \\ &= P(26/52) + P(4/52) - P(26/52) + P(4/52) \\ &= 1/2 + 1/13 - 1/21/13 \\ &= 14/26 \end{aligned}$$

(g) Care este probabilitatea de a alege o vocală din cuvintul ARTIFICIAL? (20s)

◯ 1/5

◯ 3/10

◯ 1/10

√ **1/2**

Explicaţie: Sunt 5 cazuri favorabile din 10 posibile.

(h) Care este probabilitatea de o obţine un miltiplu de 3 dintr-o aruncare cu zarul şi stema unei monede? (20s)

√ **1/6**

◯ 2/6

◯ 1/4

◯ 33%

Explicaţie: Cele 2 probabilităţi sunt independente. Astfel $P(3k \wedge stema) = P(3k) * P(stema) = 2/6 * 1/2 = 2/12$.

37. **Reţele Bayesiene**

(a) $P(A) = \sum P(A, B)$ este (20s)

- ◯ formula lui Bayes
- √ **marginalizare**
- ◯ chain rule
- ◯ joint probability

Marginalizarea reprezintă însumarea pe toate valorile posibile ale variabile noi introduse B.

Explicaţie:

(b) Formula lui Bayes este (20s)

- ◯ $P(A, B) = P(A)P(B)$
- √ $P(B|A) = P(A|B) * P(B)/P(A)$
- ◯ $P(A) = \sum P(A, B)$
- ◯ $P(B|A) = (P(A|B) * P(A))/P(B)$

Explicaţie: Formula lui Bayes este derivată direct din definiţia proabilităţilor conditionate:

$$\frac{P(A|B)}{P(A)} = \frac{P(B|A)}{P(B)}$$

(c) Pătura lui Markov (20s)

- √ **nodurile sunt independente fiind daţi părinţii + copiii + copiii părinţilor**
- ◯ este echivalenta cu tabelul complet al probabilităţilor
- ◯ nodurile sunt independente de nondescendenţi, fiind daţi părinţii

Explicaţie: Fiind date nodurile din pătura lui Markov (MB) a unui nod A, A este independent de toace nodurile rămase în reteaua Bayesiană.

$$MB(A) = Parinti(A) + Copii(A) + Parinti(Copii(A))$$

(d) Noisy-OR consideră (20s)

- √ **toate cauzele posibile sunt incluse**
- √ **probabilităţile de eşec ale fiecărei clauze sunt independente**
- ◯ nicio variantă din cele anterioare

Explicaţie: Noisy-OR este o generalizare a operatorului logic OR pentru a modela relaţii cu incertitudine. Incertitudinea se manifestă când efectul unei cauze nu se activează chiar dacă cauza a devenit adevărată. Noisy-OR are doua asumpţii: toate cauzele sunt enumerate, respectiv probabilitatea de inhibiţie a unei cauze este independentă de a oricărei alte cauze incidente în nodul Noisy-OR.

(e) Nod de tip rezidual (leak node) este (20s)

- ◯ o pagină pe Wikileaks
- ◯ un nod probabilistic dintr-o reţea Bayesiană
- √ **acoperă toate cauzele de tipul** *şi altele* **într-o relaţie Noisy-OR**
- ◯ un nod condiţional independent faţă de rădăcina reţelei Bayesiene.

Explicaţie: Deoarece Noisy-OR cere ca toate cauzele să fie menţionate este necesar un astfel de nod.

(f) O reţea Bayesiană hibridă (20s)

- ◯ este o reţea bayesiană modificată de un algoritm genetic
- ◯ conţine atât variabile aleatoare booleene, cât şi nonbooleene
- √ **conţine atât variabile discrete, cıat şi continue**
- ◯ modelează cel puţin două domenii

Explicaţie: Distribuţia de probabilitate în cazul variabilelor continue va fi dată de funcţii distribuţii de probabilitate.

38. **Inferenţă în reţele Bayesiene**

(a) Care nu este un algoritm de inferenţă exactă? (20s)

- ◯ inferenţă prin enumerare
- ◯ inferenţă prin eliminarea variabilelor
- √ **inferenţă prin simulare stocastică**
- √ **inferenţă prin Markov Chain Monte Carlo**

> **Explicaţie:** Primii doi sunt algoritmi de raţionare exactă. Ultimii doi sunt algoritmi de raţionare aproximativă în reţele Bayesiene.

(b) Care sunt variabilele ascunse din întrebarea $P(A|j,m)$ în scenariul alarmei? (20s)

- ◯ j,m
- √ **b,e**
- ◯ a
- ◯ j,a,m

> **Explicaţie:** Intr-o reţea Bayesiană sunt 3 tipuri de variabile: întrebare (X în cazul curent), evidenţe (j, m), respectiv variabile ascunse - restul nodurilor din reţea (i.e. b şi e).

(c) Care sunt variabilele evidentă din întrebarea $P(A|j,m)$ în scenariul alarmei? (20s)

- √ **j,m**
- ◯ b,e
- ◯ a
- ◯ j,a,m

> **Explicaţie:** Evidenţele sunt j şi m. Variabilele aleatoare neinstanţiate apar cu litere mari, în timp ce observaţiile cu litere mici.

(d) Care algoritm foloseşte factori? (20s)

- ◯ inferenţă prin enumerare
- √ **inferenţă prin eliminarea variabilelor**
- ◯ inferenţă prin simulare stocastică
- ◯ inferenţă prin Markov Chain Monte Carlo

> **Explicaţie:** Factorii evită calcularea repetată a aceluiaşi produs de probabilităţi, aspect care apare frecvent la algoritmul de eliminare a variabilelor.

(e) Metoda *Rejection Sampling* generează doar evenimente consistente cu evidenţele e (20s)

- ◯ Adevărat

√ **Fals**

Eşantionarea prin revocare (rejection sampling) este o metodă de raţionare aproximativă în reţele Bayesiene. Metoda generează eşantione conform distribuţiei de probabilitate specificate. Dintre aceste eşantioane le păstrează doar pe cele consistente cu evidenţele date. Probabilitatea condiţionată este obţinută prin numărarea cazurilor favorabile din eşantionul rămas.

Explicaţie:

(f) Metoda *MCMC* generează eşantioane integrale (20s)

√ **Fals**

◯ Adevărat

Explicaţie: Markov Chain Monte Carlo este o metodă de raţionare aproximativă în reţele Bayesiene. În loc de a genera de fiecare dată esăntioane complet noi, metoda porneşte cu un eşantion căruia îi modifică aleator o singură valoare la pasul următor. Aşadar eşantionele nu sunt generate integral, ci reprezintă modificări ale ultimei generări.

(g) Algoritmul *d-separare* păstrează (20s)

◯ toate variabilele menţionate în interogare şi toţi părinţii acestora

√ **toate variabilele menţionate în interogare şi toţi ancestorii acestora**

◯ toate variabilele menţionate în interogare şi toţi copiii acestora

◯ toate variabilele menţionate în interogare şi toţi descendenţii acestora

Explicaţie: Algoritmul determină dacă două variabile sunt independente într-o reţea Bayesiană, fiind date anumite evidenţe. La primul pas se selectează din reţeaua Bayesiană doar variabilele menţionate în interogare şi toţi ancestorii acestora.

(h) Metoda *likelihood weighting* (20s)

√ **foloseşte evidenţele pentru a pondera eşantionale generate**

◯ generează eşantioane dintr-o reţea goală

◯ respinge eşantioanele inconsistente cu evidenţele

Explicaţie: Metodă generează valori doar pentru variabilele care nu sunt evidenţe. Se asigură astfel că fiecare eşantion este consistent cu evidenţele. Fiecare eşantion este ponderat cu probabilitatea de a apărea în asociere cu evidenţele date.

39. **Raţionare probabilistică cu timp**

(a) În asumpţia Markov, starea curentă depinde de (20s)

- ◯ starea anterioară
- √ **un număr finit şi fix de stări anterioare**
- ◯ de părinţi + copii + copiii părinţilor

> **Explicaţie:** Doar în lanţuri Markov de ordinul I starea curentă depinde doar de starea anterioară.

(b) Un proces staţionar este un (20s)

- √ **proces în schimbare, dar legile acestei schimbări nu se modifică**
- ◯ proces static (i.e. starea nu se schimbă)
- ◯ niciuna dintre aceste variante

> **Explicaţie:** În cazul unui proces staţionar, modelul de tranziţie şi cel al senzorilor nu variază în timp.

(c) Asumpţia Markov a senzorilor (60s)

- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t, E_{0:t-1})$
- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{0:t})$
- √ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$
- ◯ $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{t-1})$

> **Explicaţie:** În primul caz, observaţiile la momentul curent depind de toate observaţiile anterioare ($P(E_t|X_t, E_{0:t-1})$). În al doilea caz, observaţiile la momentul curent depind de toate stările anterioare ($P(E_t|X_{0:t})$). În ultimul caz, observaţiile la momentul curent t depind de starea anterioa, la momentul $t-1$ ($P(E_t|X_{t-1})$).

(d) Filtrarea sau estimarea stării curente este formalizată cu (30s)

- √ $P(X_t|e_{1:t})$
- ◯ $P(X_{t+k}|e_{1:t})$
- ◯ $P(X_k|e_{1:t}), k < t$
- ◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

> **Explicaţie:** Se estimează distribuţia de probabilitate a stării curente X la momentul t pe baza observaţiilor $e_{1:t}$ obţinute până acum.

(e) Netezirea (smoothing) este formalizată cu (30s)

- ◯ $P(X_t|e_{1:t})$

◯ $P(X_{t+k}|e_{1:t})$
√ $P(X_k|e_{1:t}), k < t$
◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

Explicaţie: Se îmbunătăţeşte estimarea unei stări anterioare X_k în lumina noilor date obţinute de la momentul k până la momentul curent t.

(f) Predicţia este formalizată cu (30s)
◯ $P(X_t|e_{1:t})$
√ $P(X_{t+k}|e_{1:t})$
◯ $P(X_k|e_{1:t}), k < t$
◯ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

Explicaţie: Se prezice distribuţia de probabilitate a variabilei de stare X la un moment ulterior $t + k$ pe baza datelor obţinute până la momentul curent t.

(g) Identificarea celei mai probabile explicaţii (30s)
◯ $argmax(X_{1:t})P(X_t|e_{1:t})$
◯ $argmax(X_{1:t})P(X_k|e_{1:t})$, $k < t$
◯ $argmax(X_{1:t})P(X_{1:t}|e_t)$
√ $argmax(X_{1:t})P(X_{1:t}|e_{1:t})$

Explicaţie: Ne interesează cea mai probabilă secvenţă de instanţieri a variabilei aleatoare X de la momentul iniţial până la cel curent ($argmax(X_{1:t})$) pe baza tutoror măsurătorilor disponibile $e_{1:t}$

(h) Care din următoarele metode nu este o tehnică de raţionare în modele temporale? (20s)
√ **rezoluţia**
◯ predicţia
◯ netezirea
◯ identificarea celei mai probabile explicaţii

Explicaţie: Rezoluţia este o metodă de raţionare pe anumite logici. De exemplu, rezoluţia în logica de ordinul întâi este utilă în demonstrarea de teoreme.

(i) Pentru a putea raţiona într-o reţea Bayesiană dinamică nu e nevoie de (20s)
◯ modelul stării iniţiale
◯ modelul de tranziţie
◯ modelul senzorilor
√ **distribuţia Gaussiană**

Explicaţie: Trei informaţii sunt necesare pentru a rula algoritmii de predicţie, netezire, filtrare sau de identificare a celei mai probabile explicaţii. Aceste informaţii sunt: 1) modelul stării iniţiale, 2) modelul de tranziţie, 3) modelul senzorilor.

(j) Învăţarea unei reţele Bayesiene dinamice se referă la (60s)

- √ **a învăţa din observaţii modelul de tranziţii**
- ◯ a interpreta orice nod din reţea ca un neuron
- ◯ estimarea stării, filtrare şi netezire
- ◯ a învăţa din observaţii modelul senzorilor

Explicaţie: Atât modelul senzorilor cât şi cel de tranziţie pot fi învăţate.

40. **Decizii simple**

(a) Micromortul este (20s)

- ◯ unitatea de măsură pentru utilitate
- √ **1/1,000,000 şanse de deces**
- ◯ măsura pentru metrica QUALY (Quality Adjusted Life Years)
- ◯ identic cu microprobabilitatea

Explicaţie: O microprobabilitate reprezintă 1/1,000,000 şansa de a se întâmpla un eveniment. Un micromort este deci microprobabilitatea de deces.

(b) Sistemul de asigurări are la bază faptul că agenţii umani sunt faţă de risc (30s)

- ◯ neutri
- √ **împotrivă**
- ◯ în căutarea riscului

Explicaţie: Aversiunea faţă de risc înseamnă că agenţii umani evaluează fiecare ban în plus mai puţin dect predecesorul.

(c) Funcţia de utilitate a banilor este (30s)

- ◯ liniar crescătoare
- ◯ constantă
- √ **logaritmică**
- ◯ exponenţială

Explicaţie: Până la o anumită valoare utilitatea creşte proporţional cu banii. Peste un anumit prag, utilitatea creşte mult mai puţin relativ la creşterea masei monetare.

(d) La ce este utilizat principiul MEU? (30s)

- ◯ Minimize Expected Utility
- ◯ calcularea utilităţii
- √ **selectarea acţiunii în starea curentă**
- ◯ explicarea comportamentului iraţional

Explicaţie: MEU (Maximize Expected Utility) urmăreşte maximizarea utilităţii expectate prin selectarea celei mai promiţătoare acţiuni din starea curentă.

(e) Cum se doreşte să acţioneze agenţii software (perspectiva AIMA)? (30s)

- ◯ deterministic
- √ **raţional**

◯ cât mai uman
◯ rapid

Explicaţie: Agenţii echipaţi cu teoria utilităţilor nu urmăresc imitarea comportamentului uman, ci un comportament raţional.

(f) Cu ce se extinde o reţea Bayesiană pentru a deveni o reţea de decizie? (30s)

√ **noduri de tip acţiune şi utilitate**
◯ noduri utilitate
◯ noduri MEU
◯ Cu nimic. O reţea Bayesiană este deja o reţea de decizie

Explicaţie: Reţelele de decizie conţin 3 tipuri de noduri pentru: i) variabile aleatorare, ii) acţiuni, iii) utilităţi.

(g) Valoarea expectată a informaţiei este (30s)

◯ negativă
◯ aditivă
◯ depinde de ordinea în care sunt primite informaţiile
√ **nicio versiune din cele anterioare**

Explicaţie: Valoarea informaţiei este dată de creşterea expectată a utilitaţii comparată cu cazul în care decizia se ia fără a avea informaţia respectivă.
Fie informaţia curentă E, cea mai bună acţiune curentă α, şi potenţiala informaţie e_j.

$$EU(\alpha|e) = \max_a \sum_{s'} P(Result(a) = s'|a, e)U(s')$$

Valoarea celei mai bune acţiuni după obţinerea informaţiei $E_j = e_j$ este

$$EU(\alpha_{e_j}|e, e_j) = \max_a \sum_{s'} P(Result(a) = s'|a, e, e_j)U(s')$$

Deoarece E_j este o variabilă aleatorare a cărei valoare nu este cunoscută momentan, se calculează câştigul expectat peste toate valorile posibile e_{jk} ale informaţiei E_j.

$$VPI_e(E_j) = (\sum_k P(E_j = e_{jk}|e)EU(\alpha_{e_{jk}}|e, E_j = e_{jk})) - EU(\alpha|E)$$

Asumpţia este că înainte de obţinerea informaţiei E_j avem o distribuţie de probabilitate asupra posibilelor sale valori. Valoarea expectată a informaţiei are următoarele proprietăţi:

1. nonnegativă: $\forall e, EVPI_e(E_j) \geq 0$
2. neaditivă: $VPI_e(E_j, E_k) \neq VPI_e(E_j) + VPI_e(E_k)$
3. independentă faţă de ordine: $VPI_e(E_j, E_k) = VPI_e(E_j) + VPI_{e,E_j}(E_k) = VPI_e(E_k) + VPI_{e,E_k}(E_j)$

(h) Eroare cognitivă în care deciziile sunt afectate de o informaţia iniţială (30s)

◯ efectul certitudinii
◯ încadrarea (framing)
√ **efectul de ancoră**

Explicaţie: Informaţia iniţială acţionează ca o ancoră, chiar dacă nu are nicio legătura cu datele necesare în luarea deciziei.

(i) Eroare cognitivă în care deciziile sunt afectate de modul în care opţiunile sunt prezentate: pozitiv sau negativ (30s)

◯ efectul certitudinii
√ **încadrarea (framing)**
◯ efectul de ancoră

Explicaţie: Consideraţi următoarele opţiuni: O_1*: Aceasta operaţie are 90% şanse de succes*, respectiv O_2*: Aceasta operaţie are 10% şanse de eşec.* Numărul pacienţilor care acceptø operaţie prezentată ca O_1 este dublu fată de cei care au primit informaţia O_2. Similar: O_3 *Această pastă de dinţi conţine 98% elemente naturale.*, respectiv O_4 *Această pastă de dinţi conţine 2% elemente sintetice.*

(j) Teoria deciziilor este o teorie (10s)

√ **normativă**
◯ descriptivă

Explicaţie: Teoria deciziilor este o teorie normativă deoarece descrie cum ar trebui un agent raţional să acţioneze. O teorie descriptivă descrie sau explică cum acţionează în realitate un agent.

41. **Decizii secvenţiale**

(a) Problemele de decizii secvenţiale (SDP) (30s)

- √ **includ utilităţi, incertitudine şi captarea informaţiilor din mediu**
- √ **utilitatea agentului depinde de o secvenţă de decizii**
- ◯ sunt cazuri particulare ale problemelor de planificare
- ◯ sunt cazuri particulare de probleme de decizie Markoviene

Explicaţie: Problemele de căutare, planificare şi cele Markoviene sunt cazuri particulare ale problemelor de decizii secvenţiale.

(b) Problemele de decizie Markoviene sunt probleme secvenţiale pentru mediile (60s)

- √ **complet observabile şi stocastice**
- ◯ complet observabile şi deterministe
- √ **cu modele de tranziţie Markoviene şi funcţii aditive de recompensă**
- ◯ parţial observabile şi stocastice

Explicaţie: Dacă mediul este parţial observabil vom avea POMDPs (Partially Observable Markov Decision Problems).

(c) O politică optimă $\pi(s)$ (30s)

- ◯ minimizează suma expectată a recompenselor
- √ **specifică cea mai bună acţiune pentru fiecare stare** s
- √ **are cea mai mare valoare expectată a utilităţii**

Explicaţie: Politica π este o funcţie care primeşte o stare s şi întoarce cea mai bună acţiune din starea respectivă.

(d) O politică optimală pentru un orizont finit este (20s)

- √ **nonstaţionară**
- ◯ staţionară

Explicaţie: În politicile nonstaţionare $\pi(s)$ depinde de timpul rămas. În cazul unui orizont de timp infinit, nu există niciun motiv pentru ca agentul să se comporte diferit într-o aceeaşi stare.

(e) Dacă factorul de discount tinde spre 0, atunci recompensele pe termen lung sunt (30s)

- √ **nesemnificative**
- ◯ semnificative

Explicaţie: Factorul de discount este între 0 şi 1. Cu valori apropiate de 1, recompensele pe termen lung au acelaşi impact precum recompensele pe termen scurt.

(f) Ecuaţia lui Bellman este (60s)

◯ $U(s) = R(s) + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s)$
◯ $U(s) = R(s) + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s')$
√ $U(s) = R(s) + \gamma max_{a \in A(s)} \sum_{s'} P(s'|a, s)U(s')$
◯ $U(s) = R(s') + \gamma max_{a \in A(s')} \sum_{s'} P(s'|a, s)U(s')$

Explicaţie: Prima variantă consideră acţiunile posibile din starea succesoare ($max_{a \in A(s')}$), şi nu pe cele din starea curentă ($max_{a \in A(s)}$). În plus se consideră utilitatea stării curente $U(s)$ şi nu utilitatea stării succesoare. A doua şi a patra versiune consideră acţiunile posibile din starea succesoare ($max_{a \in A(s')}$). În plus, a patra versiune consideră recompensa pe starea succesoare $R(s')$, în locul recompensei primite în starea curentă $R(s)$.

(g) Care din următoarele licitaţii au caracter public şi sunt descendente? (30s)

◯ englezeşti
√ **olandeze**
◯ cu plic inchis
◯ Vickrey

Explicaţie: Licitaţiile englezeşti sunt publice, dar preţul are caracter ascendent. Licitaţiile cu plic inchis au caracter privat: agenţii nu au acces la valorile licitate de agenţii concurenţi. Licitaţiile de tip Vickrey sunt un caz particular al licitaţiilor cu plic închis în care agentul care licitează cel mai mult este câştigător, dar la cea de-a doua valoare licitată de către alt agent. În licitaţia olandeză agentul vânzator porneşte cu un preţ iniţial cunoscut de către toţi participanţii şi care este scăzut la fiecare pas până când devine atragător pentru un cumpărător. Singurul protocol din cele enumerate cu caracter public şi preţ descendent este deci licitaţia olandeză.

(h) O soluţie este Pareto optimală dacă (30s)

◯ nu există o altă soluţie pe care unul dintre agenţi o preferă
√ **nu există o altă soluţie pe care toţi agenţii să o prefere**
◯ există o altă soluţie pe care unul dintre agenţi o preferă
◯ există o altă soluţie pe care toţi agenţii să o prefere

Explicaţie: Optimalitatea Pareto se referă la toţi agenţii implicaţi.

42. **Arbori de decizie**

(a) Dacă predicatul ţintă are un set finit de valori, atunci problema de învăţare este (10s)

- ◯ regresie
- √ **clasificare**

> **Explicaţie:** Dacă predicatul ţintă are o infinitate de valori (e.g. se doreşte prezicerea preţului unui apartment) avem o problemă de regresie. Într-un astfel de caz se vor utiliza arbori de regresie.

(b) Principiul *Ockham's razor* preferă (20s)

- ◯ cea mai buna ipoteză consistentă cu datele
- √ **cea mai simplă ipoteză consistentă cu datele**
- ◯ $X^2 + 1/3$ faţă de X^3

> **Explicaţie:** Principiul lamei lui Ockham favorizează simplitatea în faţa acurateţei. Principiul reprezintă o metodă de evitare a suprapotrivirii (overfitting).

(c) O problemă de învăţare este realizabilă dacă (30s)

- √ **spaţiul ipotezelor conţine funcţia adevărată**
- ◯ datele sunt consistente şi fără zgomote
- ◯ se aplică un algoritm de învăţare potrivit pentru date

> **Explicaţie:** Spaţiul de căutare a soluţiilor trebuie să conţină funcţia căutată.

(d) Arborele de decizie este construit pe baza unei strategii de căutare de tipul (30s)

- ◯ căutare în adâncime
- ◯ căutare în lăţime
- √ **greedy şi desparte-şi-cucereşte**
- ◯ aleator

> **Explicaţie:** Cel mai apetisant atribut este cel care aduce cel mai mare câştig de informaţie. Căutarea se bazează deci pe o metodă greedy.

(e) Dacă s-au analizat toate atributele, dar rămân exemple neclasificate atunci (60s)

- √ **există erori sau zgomote în datele de antrenament**
- √ **domeniul în care se aplică învăţarea nu este deterministic**
- √ **nu există acces la un atribut prin care s-ar putea distinge între instanţe**
- ◯ nici una din opţiunile anterioare

Explicaţie: Pentru arborii de decizie, oricare din primele 3 variante poate fi o cauză acestei situaţii.

(f) Mai multă informaţie înseamnă (10s)

◯ mai multă entropie

√ **mai puţină entropie**

Explicaţie: Cu cât un sistem este mai agitat, cu atât informaţia este mai puţină.

(g) Suprapotrivirea (overfitting) este (10s)

◯ de dorit

√ **de evitat**

Explicaţie: Cu cât un sistem este mai agitat, cu atât informaţia este mai puţină.

(h) Suprapotrivirea (overfitting) (20s)

√ **violează principiul lamei lui Ockham**

√ **este sinonimă cu supraantrenarea**

√ **include mai mulţi parametri decât cei necesari**

√ **toate variantele anterioare**

Explicaţie: Urmând principiul simplităţii suprapotrivirea poate fi evitată prin preferarea de clasificatori mai simpli, chiar dacă au o acurateţe mai slabă. O altă denumire pentru *overfitting* este *overtraining*. Practic, parametrii algoritmului de învăţare sunt reglaţi până când clasificatorul este potrivit perfect pe datele de antrenare. Această potrivire perfectă este dată de un clasificator complex, care necesită un număr mare de parametri ce pot fi ajustaţi. Datorită acestei specializări pe datele de antrenament clasificatorul îşi pierde capacitatea de generalizare pe date diferite de cele de antrenament.

(i) Validarea de tip hold-out nu utilizează în antrenare toate datele disponibile. (20s)

√ **Adevărat**

◯ Fals

Explicaţie: Datele disponibile sunt împărţite în două seturi distincte. Se aplică învăţarea doar pe datele de antrenament. Metoda *K-fold cross validation* elimină acest dezavantaj.

(j) Performanţa învăţării este dată de acurateţea prezicerii pe setul de (30s)

◯ antrenare
◯ validare
√ **testare**
◯ media acurateţii pe cele 3 seturi

Explicaţie: Setul de testare conţine date care au fost ţinute într-un seif pe durata învăţarii. Doar dupa terminarea învăţării se deschide seiful şi se evaluează performanţa modelului obţinut.

(k) *Peeking* apare dacă (60s)

√ **ipoteza se selectează şi pe baza erorii pe setul de testare**
√ **nu se utilizează set pentru validare**
◯ curba de învăţare (ROC) este un *graf vesel*
◯ atributul ţintă este continuu

Explicaţie: Setul de testare nu trebuie să influenţeze modelul învăţat pe datele de antrenament/validare. Acestea trebuie să fie distincte de cele utilizate în testare.

43. **Cunoştinţe în învaţăre**

(a) Care este diferenţa dintre regresie şi clasificare? (30s)

- √ **natura atributului ţintă**
- ◯ o metodă este supervizată, iar cealaltă nesupervizată
- ◯ o metodă utilizează statistica, iar cealaltă nu
- ◯ nu e nicio diferenţă

Explicaţie: Ambele clase ţin de învăţarea supervizată. La clasificare, atributul ţintă este discret/finit (e.g. două clase Spam/Not Spam). La regresie, atributul ţinta este continuu (e.g. viteza vehicului din faţă).

(b) Dacă ipoteza clasifică o instanţă ca negativă dar corect ar fi pozitivă, avem (20s)

- ◯ fals pozitiv
- √ **fals negativ**
- ◯ adevărat pozitiv
- ◯ adevărat negativ

Explicaţie: Instanţa este clasificată greşit ca fiind negativă. Avem deci un fals negativ.

(c) Instanţele fals pozitive sunt rezolvate prin (30s)

- ◯ generalizare
- √ **specializare**

Explicaţie: Instanţa este clasificată greşit ca pozitivă. Clasificatorul este deci prea general (i.e. etichetează prea multe instanţe ca pozitive). Se impune o specializare a acestuia.

(d) Generalizarea are loc prin (30s)

- ◯ adăugarea de condiţii
- √ **eliminarea unor condiţii**

Explicaţie: Cu cât are mai multe condiţii, ipoteza de clasificare este mai specifică. Generalizarea se obţine prin îndepărtarea unei condiţii din ipoteză.

(e) Ce nu este adevărat legat de căutarea current-best-hypothesis? (30s)

- ◯ menţine o singură ipoteză care se ajustează cu fiecare exemplu
- √ **este deterministică**
- ◯ modificarea ipotezei impune verificarea instanţelor anterioare
- ◯ execută mult backtracking

Explicaţie: Algoritmul lucrează cu o singură ipoteză. Modificarea ipotezei de către instanţe fals negative sau fals pozitive poate duce la o clasificare greşită a instanţelor deja analizate. Reverificarea acestora generează backtracking. Modificarea ipotezei prin generalizare sau specializare se face nondeterminist.

(f) Care nu este o condiţie de stop a algoritmului *version space*? (60s)

- ◯ rămâne doar o ipoteză în spaţiul soluţiilor
- ◯ spaţiul soluţiilor colapsează ($\mathcal{G}$ sau $\mathcal{S}$ devin vide)
- ◯ rămân mai multe ipoteze şi nu mai există instanţe de analizat
- √ **complexitatea ipotezei este mai mare decât un prag**

Explicaţie: Primele trei opţiuni sunt condiţii de oprire a algoritmului. Dacă o mulţime devine vidă, exemplele de antrenament sunt inconsistente. Dacă învăţarea se termină cu mai multe ipoteze de clasificare, toate acestea vor fi aplicate pe exemplele noi. Metode de tip *votare* pot fi aplicate în cazul în care aceste ipoteze nu sunt consistente între ele cu privire la clasa unei instanţe.

(g) Algoritmul FOIL evită suprapotrivirea (overfitting) prin (30s)

- ◯ validare încrucişată
- √ **penalizând ipotezele mari**
- ◯ folosind cunoştinţe din domeniu

Explicaţie: Conform principiului lui Ockham.

(h) Învăţarea bazată pe relevanţă (30s)

- ◯ extrage reguli generale dintr-un singur exemplu
- √ **foloseşte cunoştinţe anterioare**

Explicaţie: Cunoştinţele anterioare indică dacă ipotezele deduse prin inducţie sunt relevante sau nu.

(i) Ce nu este adevărat la *version space*? (30s)

- ◯ este o reprezentare hierahică a cunoştinţelor
- √ **valorile atributelor sunt continue**
- ◯ utilizează arbori de generalizări şi specializări
- ◯ presupune ca datele sunt corecte

Explicaţie: Metoda funcţionează doar dacă atributele din setul de antrenament sunt discrete (nominale).

44. **Reţele neurale artificiale**

(a) Corespondenţele pentru ⟨neuron, sinapse, dendrite, axon⟩ sunt: (30s)

- ◯ ⟨ponderi, nod, intrare, ieşire⟩
- √ ⟨**nod, ponderi, intrare, ieşire**⟩
- ◯ ⟨intrare, nod 3, ponderi 4, ieşire⟩
- ◯ ⟨intrare, ponderi, nod, ieşire⟩

Explicaţie: Într-o reţea neurală artificială nodurile reprezintă neuronii, iar sinapsele sunt modelate cu ponderi pe legăturile dintre neuroni. Dendridele reprezintă conexiunile de intrare în neuron, iar axonul reprezintă impulsul generat la activarea neuronului.

(b) 20 Reţelele neurale sunt metode de învăţare

- ◯ supervizată
- ◯ nesupervisaă
- ◯ reinforsată
- √ **depinde de tipul reţelei neurale**

Explicaţie: Reţelele neurale artificiale sunt metode supervizate. Maşinile Boltzmann sunt reţele neurale recurente stocastice în care învăţarea este nesupervizată.

(c) Neuronul cu $w_0 = 0.5$, $w_1 = 1$ şi $w_2 = 1$ implementează poarta logică (60s)

- ◯ and
- √ **or**
- ◯ xor
- ◯ not

Explicaţie: Pentru intrarea $\langle 0, 0\rangle$ ieşirea e $-0.5 + 1 \cdot 0 + 1 \cdot 0 = -0.5$. Pentru intrarea $\langle 0, 1\rangle$ ieşirea e $-0.5 + 1 \cdot 0 + 1 \cdot 1 = 0.5$. Pentru intrarea $\langle 1, 0\rangle$ ieşirea e $-0.5 + 1 \cdot 1 + 1 \cdot 0 = 0.5$. Pentru intrarea $\langle 1, 1\rangle$ ieşirea e $-0.5 + 1 \cdot 1 + 1 \cdot 1 = 1.5$. Neuronul se activează în ultimele 3 cazuri. Pentru cele 4 perechi de intrare, ieşirile neuronului vor fi în ordine $0, 1, 1, 1$, ceea ce corespunde porţii logice OR.

(d) Neuronul cu $w_1 = 1$ şi $w_2 = 1$ implementează poarta logică AND dacă (60s)

- ◯ $w_0 = 0.5$
- ◯ $w_0 = -0.5$
- ◯ $w_0 = 1$
- √ $w_0 = 1.5$

Explicaţie: Pentru intrarea $\langle 0, 0\rangle$ ieşirea e $-w_0 + 1 \cdot 0 + 1 \cdot 0 = -w_0$. Neuronul nu trebuie să se activeze, deci $-w_0 < 0$ sau $w_0 > 0$.
Pentru intrarea $\langle 0, 1\rangle$ ieşirea e $-w_0 + 1 \cdot 0 + 1 \cdot 1 = 1 - w_0$. Neuronul nu trebuie să se activeze, deci $1 - w_0 < 0$ sau $w_0 > 1$. Ultima variantă rămâne singura corectă.

(e) Perceptronul nu poate modela funcţia XOR (20s)

√ **Adevărat**

◯ Fals

Explicaţie: XOR nu este liniar separabilă, iar perceptronul este un clasificator liniar.

(f) Învăţarea are loc prin ajustarea ponderilor pentru a reduce eroarea pe setul de test (20s)

◯ Adevărat

√ **Fals**

Explicaţie: Învăţarea are loc într-adevăr prin ajustarea ponderilor între neuroni. Dar acest lucru urmăreşte reducerea erorii pe setul de antrenament, nu pe cel de test. Setul de test nu trebuie să influenţeze parametrii învăţării!

(g) Ce aţi răspunde unui medic legat de similaritatea dintre creier şi ANN? (20s)

◯ sunt similare

√ **nu diferite**

Explicaţie: Asemănările între creierul uman şi reţelel neurale artificiale (ANN) nu pot fi duse prea departe. De exemplu, nu există dovezi că un mecanism similar cu cel al algoritmului de propagare înapoi are loc în creierul uman.

(h) SVM crează un clasificator (20s)

√ **neliniar**

◯ liniar

Explicaţie: Maşinile cu suport vectorial crează un hyperplan pentru separarea a două clase. Prin introducerea unor coordonate suplimentare şi modificarea valorilor iniţiale pot fi clasificate şi date nonliniare.

(i) Numărul vectorilor suport este mai mare decât setul de antrenament (20s)

◯ Adevărat

√ **Fals**

Explicaţie: Un număr foarte mic de vectori sunt suficienţi pentru a separa cele două clase.

(j) *Boosting* este (20s)

√ **o metodă de învăţare cu ansambluri de clasificatori**

◯ un caz particular de arbore de decizie

◯ o metodă de eficientizare a învăţării prin paralelizare

Explicaţie: Un exemplu este algoritmul AdaBoost. Ansamblul de clasificatori poate fi format dintr-un set de arbori de decizie de dimensiune redusă.

(k) Învăţarea cu ansambluri poate utiliza clasificatori diferiţi (e.g. ANN, SVM, DT) (20s)

√ **Adevărat**

◯ Fals

Explicaţie: Metode de tip votare se pot aplica pentru a rezolva situaţiile în care clasificatorii au păreri diferite cu privire la eticheta unei instanţe.

45. Clusterizare. K-means. Clusterizare ierarhică

(a) K-means este un algoritm de (10s)

◯ clusterizare herarhică

√ **clusterizare prin partiţionare**

Explicaţie: Parametrul K reprezintă numărul de partiţii ale setului de date.

(b) K-means are nevoie de date normalizate şi nenominale (20s)

√ **True**

◯ False

Explicaţie: Fară normalizare anumite atribute nu vor avea contribuţie relevantă în calcularea distanţei între instanţe. Atributele nominale introduc dificultăţi la calcularea distanţei dintre instanţe.

(c) În suma erorilor pătratice (SSE) pentru fiecare instanţă se (30s)

◯ însumează distanţa până la cel mai apropiat centroid

√ **ridică la patrat distanţa până la cel mai apropiat centroid şi se însumează rezultatele**

◯ însumează distanţele la toate celelalte puncte din cluster

◯ ridică la patrat distanţele faţă de toate celelalte puncte din cluster şi se însumează rezultatele.

Explicaţie: SSE este o metrică pentru evaluarea clusterizării. Pentru fiecare punct, eroarea este distanţa până la cel mai apropiat centroid. Erorile pentru toate punctele din setul de date se ridică la pătrat şi se însumează:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

unde x este un punct din clusterul C_i iar m_i este centroidul clusterului C_i.

(d) Dacă K creşte atunci SSE (10s)

√ **scade**

◯ creşte

Explicaţie: Creşterea numărului de clustere duce la distanţe mai mici între puncte şi centroizii care sunt mai numeroşi.

(e) Dacă setul de date conţine 3 grupuri reale, care sunt şansele ca algoritmul K-means să genereze aleator câte un centroid în fiecare cluster? (60s)

◯ rulări multiple
◯ Bisecting K-means
◯ clusterizare ierarhică
√ **preprocesare**

Explicaţie: Dacă clusterele au un număr egal de puncte atunci

$$P = \frac{\text{numărul de posibilităti pentru a selecta un centroid din fiecare cluster}}{\text{numărul de posibilităţi de a selecta K centroizi}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

Pentru $k = 3$, probabilitatea este $\frac{3!}{3^3} = \frac{2}{9}$. Şansele devin extrem de mici pe măsură ce creşte K. De exemplu pentru $K = 10$ probabilitatea este $10!/10^{10} = 0.00036$.

(f) Care nu este o soluţie la problema iniţializării algoritmului K-means? (30s)

◯ rulări multiple
◯ Bisecting K-means
◯ clusterizăre ierarhică
√ **preprocesare**

Explicaţie: Rulările multiple ajută la creşterea şanselor de a avea centroizi în fiecare cluster real. În urma rulărilor succesive se preferă clusterizarea cu valoarea SSE cea mai mică. Algorimul Bisecting K-means iniţializează la fiecare pas doar 2 centroizi. Astfel, şansele ca aceştia să aparţină la clusteri distincţi este mai mare comparativ cu $k > 2$. Clusterizarea ierarhică poate fi utilizată în generarea centroizilor iniţiali.

(g) Care din următorele nu este o metrică pentru distanţa dintre clustere? (30s)

◯ min (single link)
◯ media grupului
√ **dendograma**
◯ distanţa dintre centroizi

Explicaţie: Metrici utilizate pentru calcularea distanţei între clustere sunt: (i) min (single link), (ii) max (complete link), (iii) distanţa dintre centroide, (iv) media distanţelor dintre toate punctele din clustere, (v) rata de creştere a valorii SSE (metoda Ward). Dendograma este o reprezentare grafică a ordinii în care se face gruparea punctelor, precum şi a distanţei dintre clustere.

(h) Care metodă de clusterizare nu are nevoie de predefinirea numărului de clustere? (20s)

√ **Bisecting K-means**
◯ K-means

√ **clusterizare hierarhică**

◯ toate menţionate anterior

Explicaţie: În cazul Bisecting K-means şi a clusterizării ierarhice, numărul de clustere se decide după rularea algoritmului.

(i) Clusterizarea ierarhică cu metrica min (single link) (30)

◯ nu este susceptibilă la excepţii

◯ nu e sensibilă la zgomote

√ **nicio varianta din cele anterioare**

Explicaţie: Clusterizarea cu metrica *min* are limitări legate de sensibilitatea la zgomote sau excepţii.

(j) Clusterizarea ierarhică cu metrica max (complete link) (30)

√ **tinde să spargă grupurile mari,**

√ **tinde să genereze clustere de formă globulară**

◯ susceptibil la zgomote

◯ susceptibil la excepţii

Explicaţie: Clusterizarea cu metrica *max* tinde să genereze clustere de dimensiune relativ egală. Dacă datele conţin grupuri de dimensiune diferite, metrica *max* nu este adecvata: puncte din grupurile mari vor fi înglobate în clusterele mici.

46. **Reguli de asociaţie. Algoritmul Apriori**

(a) Este suportul o măsură simetrică? (10s)

◯ nu

√ **da**

Explicaţie: Suportul este simetric dacă $s(X \to Y) = s(Y \to X)$ pentru toate mulţimile X şi Y. Suportul este dat de mulţimea $X \cup Y$, care este aceeaşi pentru ambele reguli.

(b) Este confidenţa o măsură simetrică? (10s)

√ **nu**

◯ da

◯ poate

Explicaţie: Confidenţa este simetrică dacă $c(X \to Y) = c(Y \to X)$ pentru toate mulţimile X şi Y. Consideraţi următorul exemplu: $s(\{e\}) = 0.8$, $s(\{b,d\}) = 0.2$, $s(\{b,d,e\}) = 0.2$ Confidenţa

$$c(\{b,d\} \to \{e\}) = \frac{0.2}{0.8} = 0.4$$

iar

$$c(\{e\} \to \{b,d\}) = \frac{0.2}{0.2} = 1$$

Contraexemplul demonstrează că nu este o metrică simetrică.

(c) Câte seturi de itemi pot fi construite cu d itemi unici? (20s)

√ 2^d

◯ d^2

◯ $d!$

◯ $(d-1)!$

Explicaţie: Cu d itemi se pot forma 2^d mulţimi distincte (incluzând mulţimea vidă şi mulţimea formată din toţi cei d itemi.

(d) Câte reguli pot fi generate cu d itemi unici? (30s)

◯ $3^{d+1} - 2^d + 1$

◯ $3^{d+1} + 2^{d+1}$

√ $3^{d+1} - 2^{d+1} + 1$

◯ $3^{d+1} + 2^{d+1}$

Explicaţie: Partea stângă a regulii poate fi formată dintr-un număr k de itemi ($k \leq d$): 1 item, 2 itemi, respectiv $d-1$ itemi. Considerând k itemi, există un număr de C_d^k seturi în partea stângă a regulii. Deoarece $1 \leq k < d$, numarul de posibilităţi de formare a părţii stângi a regulii este $\sum_{k=1}^{d-1}$. Dacă k itemi apar în partea stângă, atunci $d-k$ itemi rămân în partea dreaptă a regulii. Cu $d-k$ itemi se pot forma C_{d-k}^i seturi de itemi, cu $1 \leq i \leq d-k$. Toate posibilităţile sunt date de produsul dintre numărul de combinaţii a părţii stângi cu numărul de combinaţii a părţii drepte a regulii:

$$R = \sum_{k=1}^{d} C_d^k \sum_{i=1}^{d-k} C_i^{d-k}$$

(e) Suportul pentru un set de itemi nu depăşeşte suportul submulţimilor sale. (20s)

◯ fals

√ **adevărat**

Explicaţie: O mulţime mai mare nu se va regăsi mai des în setul de date decât oricare din submulţimile sale.

(f) $\forall X, Y \ X \subseteq Y \rightarrow s(X) \leq s(Y)$ (20s)

◯ adevărat

√ **fals**

Explicaţie: Dacă un set de itemi este frecvent, atunci toate subseturile sale sunt frecvente. Altfel spus, suportul unui set nu depăşeşte suportul subseturilor sale. Formal:

$$\forall X, Y, X \subseteq Y \rightarrow s(X) \geq s(Y)$$

Suportul este deci anti-monoton.

(g) Câte reguli candidate pot fi generate dintr-o mulţime de itemi frecventî L ($|L| = k$)? (30s)

◯ 2^k

◯ 2^{k-2}

√ $2^k - 2$

◯ $k!$

Explicaţie: Se exclud regulile cu 0 respectiv k itemi în partea stângă a regulii.

(h) Confidenţa este anti-monotonă cu privire la numărul de itemi din partea dreaptă (20s)

√ **adevărat**

◯ fals

Explicaţie: Cu cât numărul de itemi din partea dreaptă creşte, cu atât scade confidenţa regulii (e.g. $c(a \to bc) < c(a, b \to c)$.

(i) Metoda $F_{k-1} \times F_{k-1}$: Care pereche nu se va agrega? (30s)

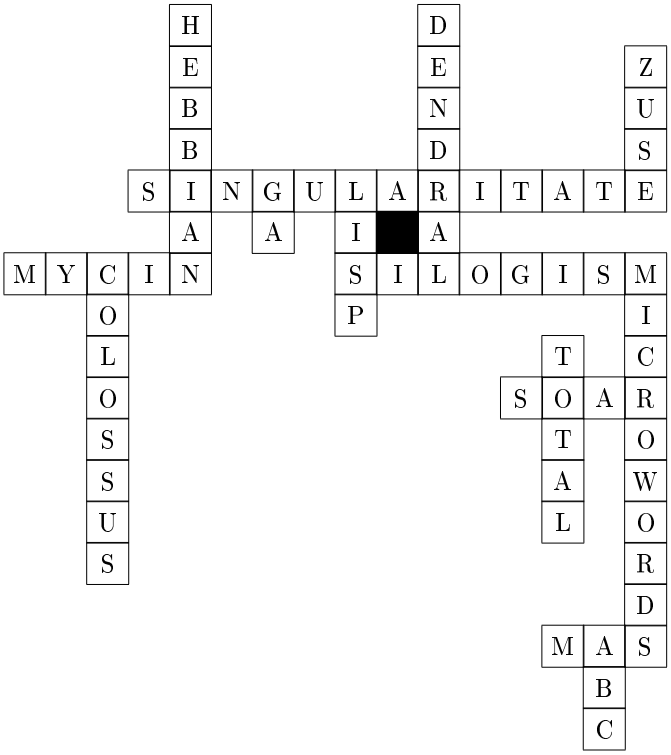
◯ ABC, ABD
◯ ABD, ABE
√ **ABC, ACD**
◯ ABC, ABE

Explicaţie: Seturile ABD şi ACD au un prefix comun doar de lungime 1. Pentru gruparea a două mulţimi de dimensiune 3 este necesar un prefix comun de lungime 2. Toate celelalte perechi au un astfel de prefix.

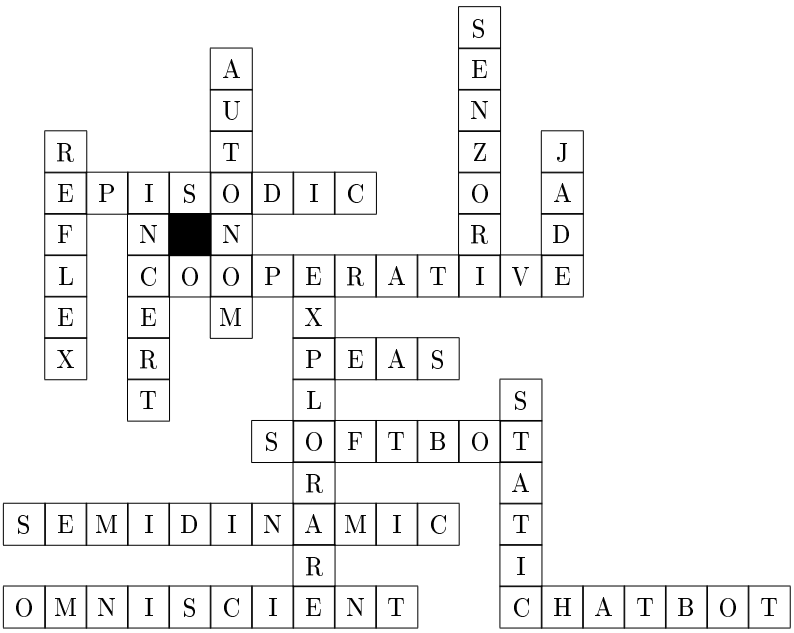
Capitolul 4

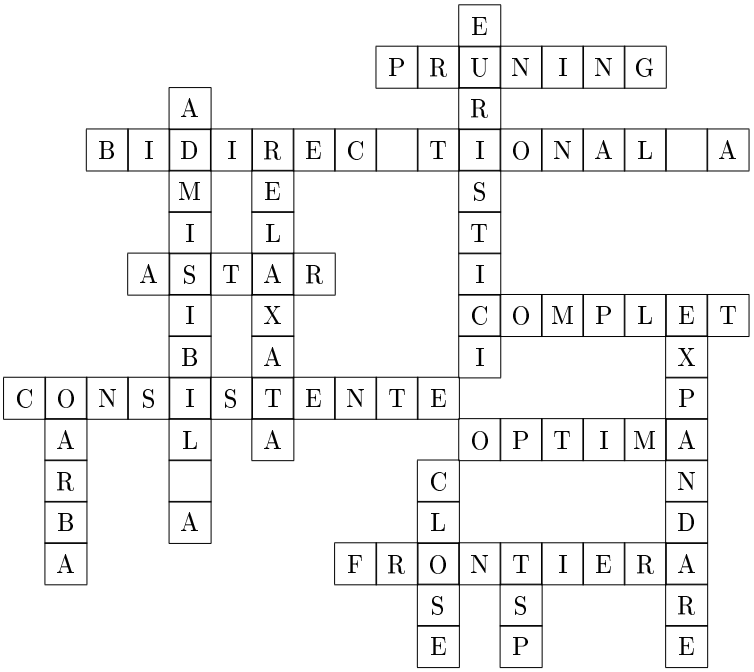
Rebusuri în inteligenţă artificială

Introducere în Inteligenţă Artificială

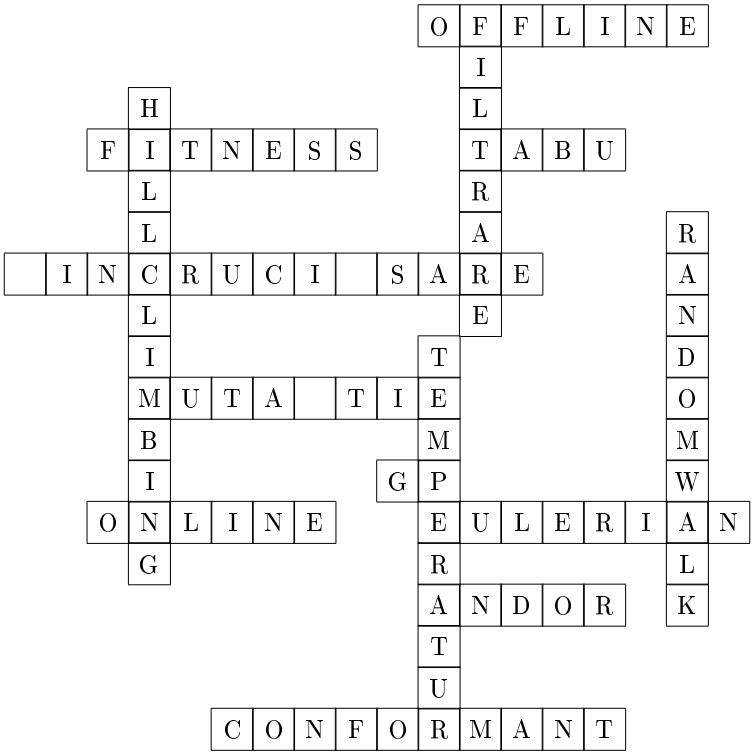


Agenţi inteligenţi

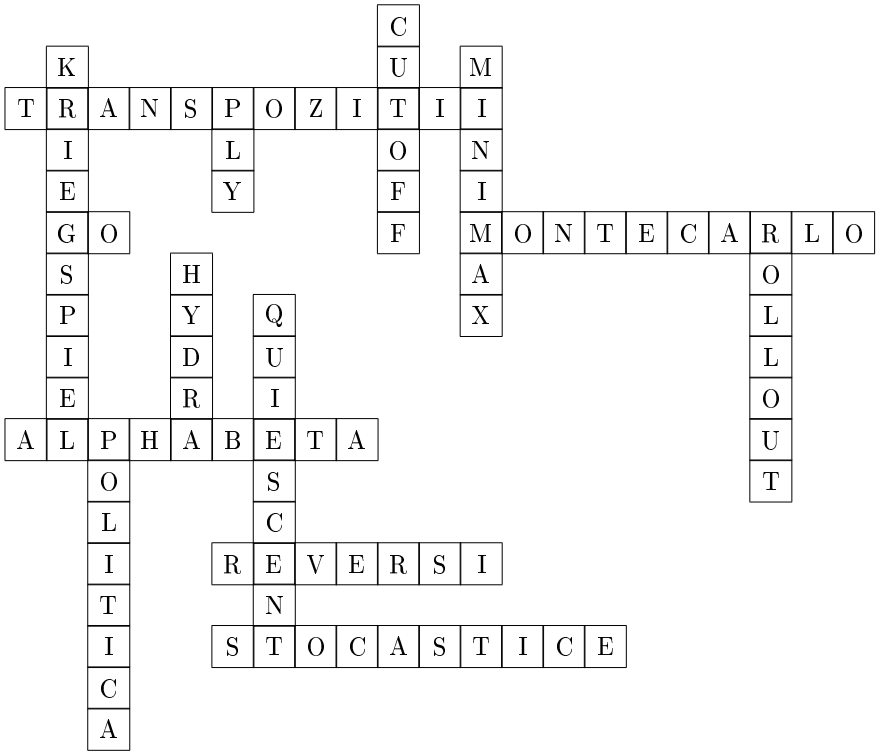



E
PRUNING
A R
BIDIRECTIONAL A
M E S
I L T
ASTAR I
I X COMPLETE
B A I X
CONSISTENTE P
A L A OPTIMA
R C N
B A L D
A FRONTIERA
S S R
E P E

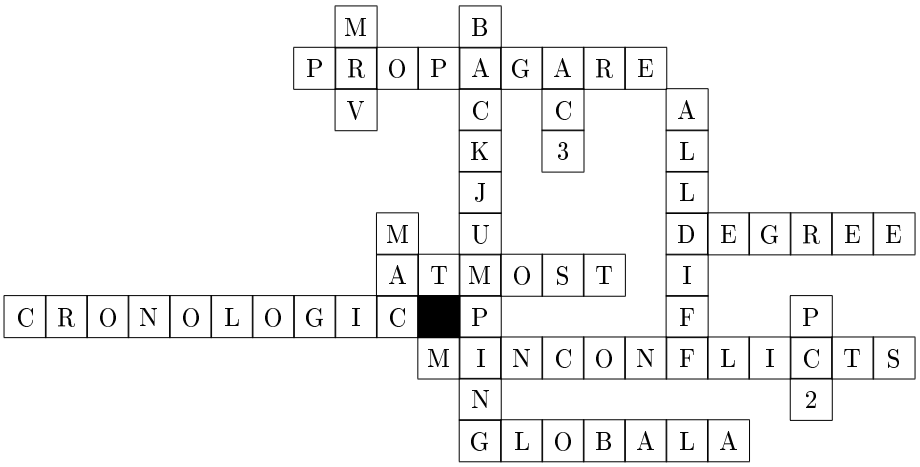
Căutare locală



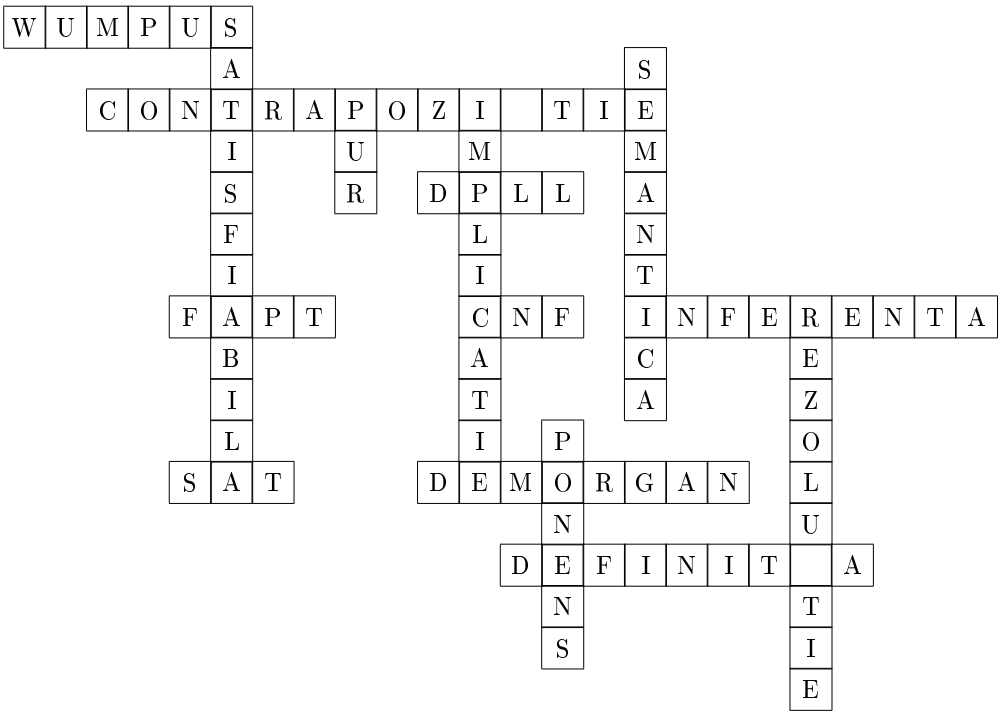
Căutare adversarială



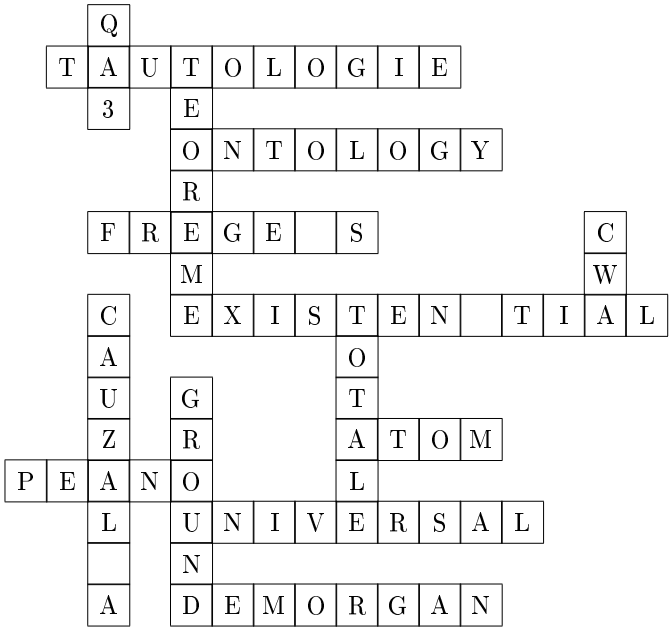
Problem de satisfacere a constrângerilor

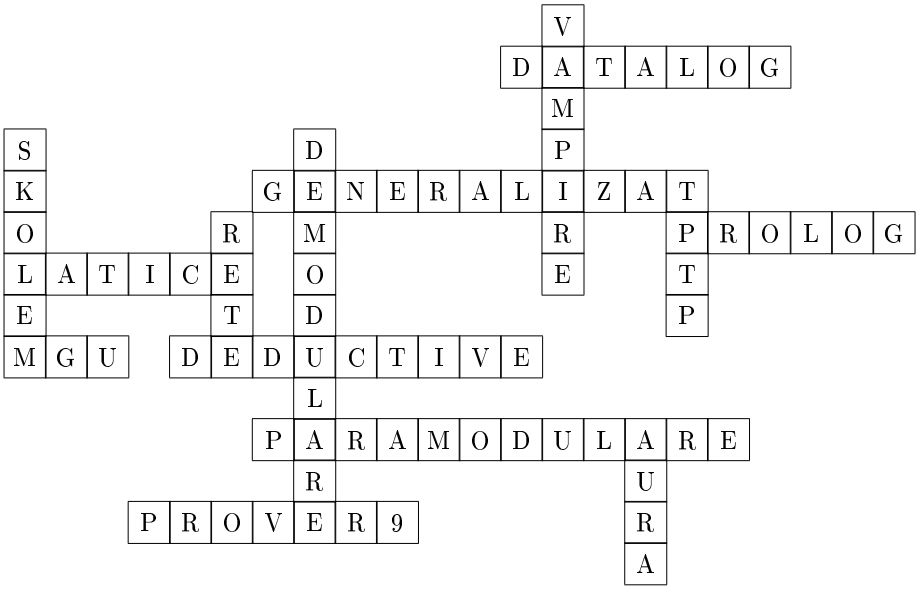


Agenţi logici

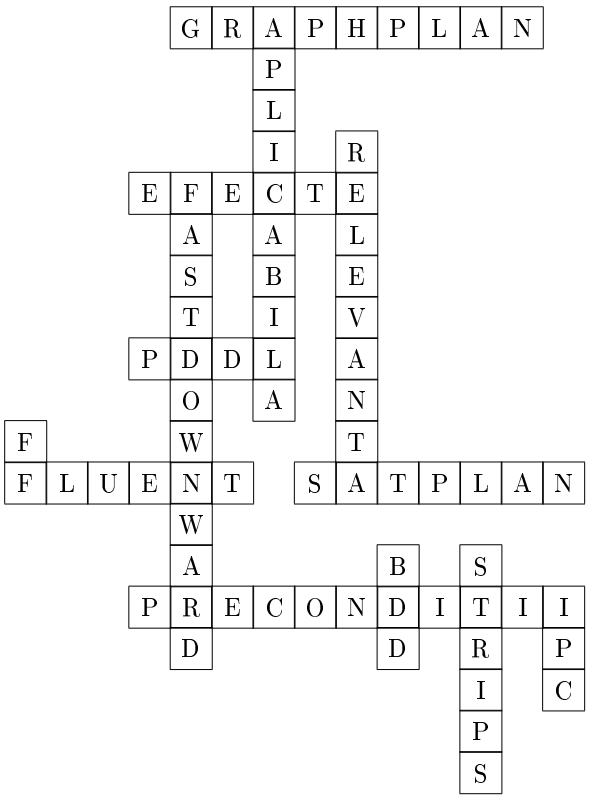


Logica de ordinul întâi

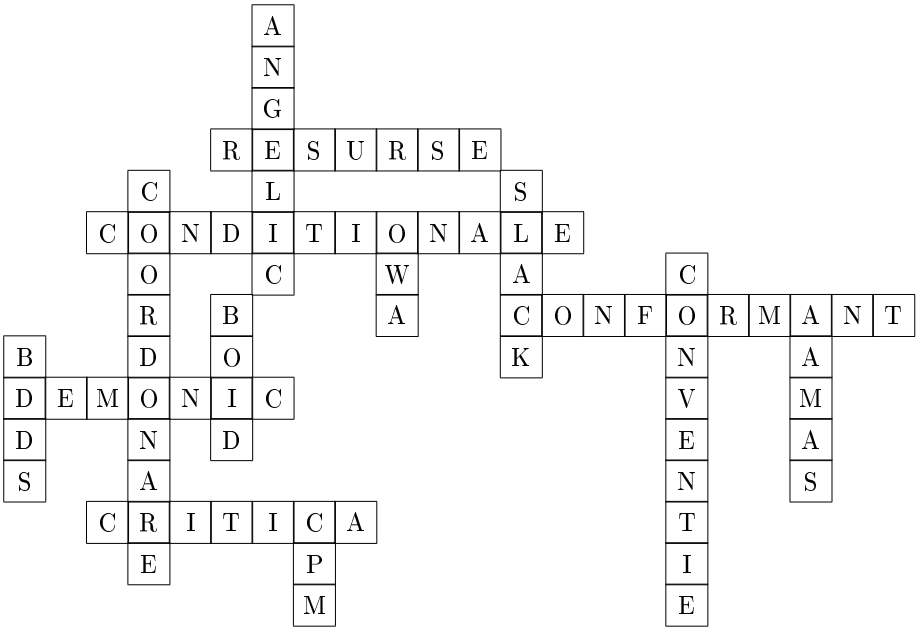



VDATALOG
MSDPKGENERALIZATORMRPROLOG
LATICEOETEDTPMGUDEDUCTIVE
LPARAMODULARE
RUPROVER9RA

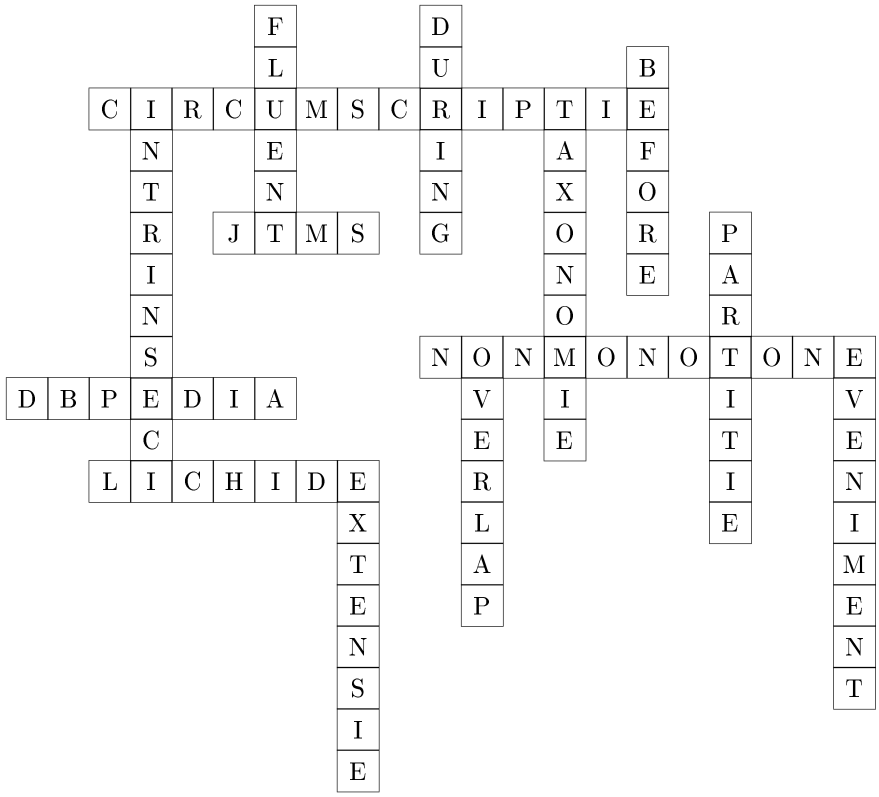
Planificare clasică



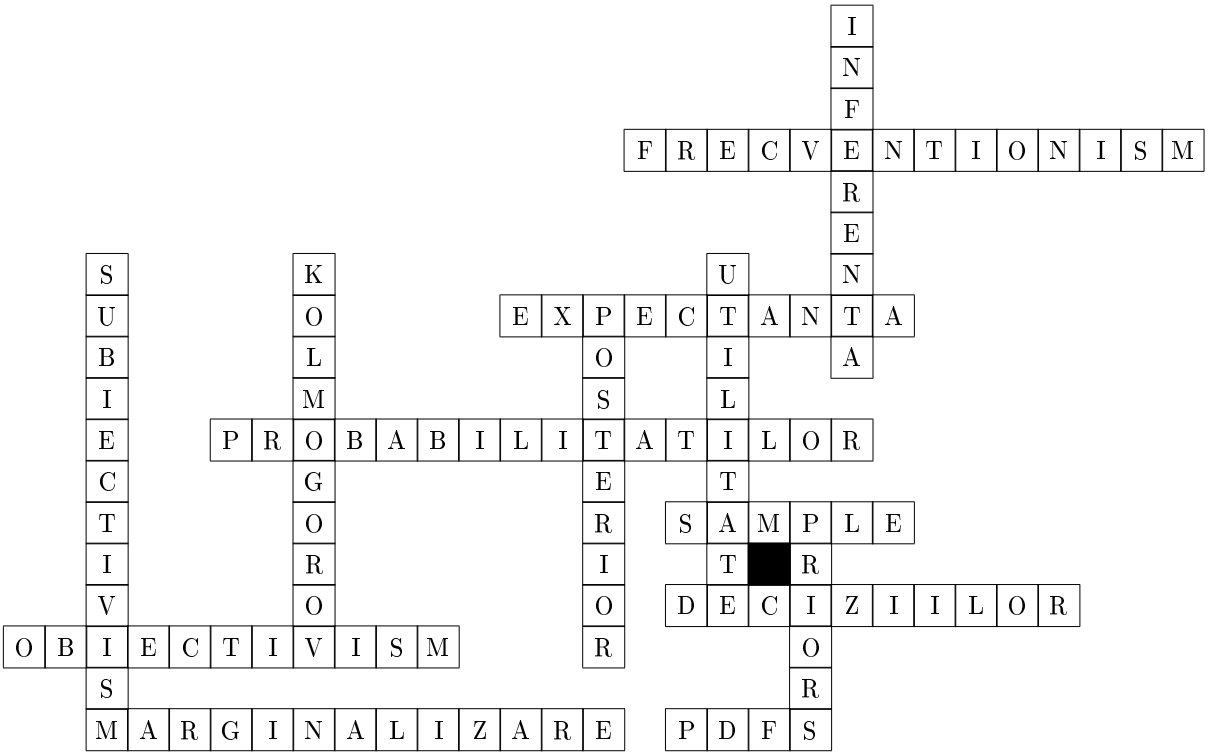
Planificare în lumea reală



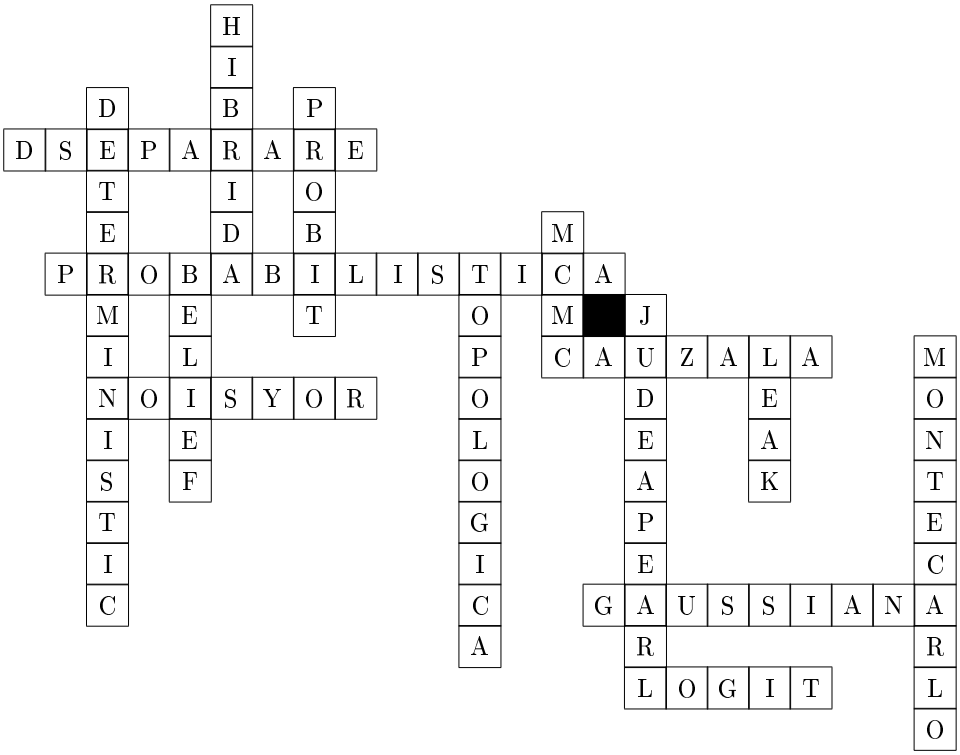
Reprezentarea cunoştinţelor. Calculul evenimentelor



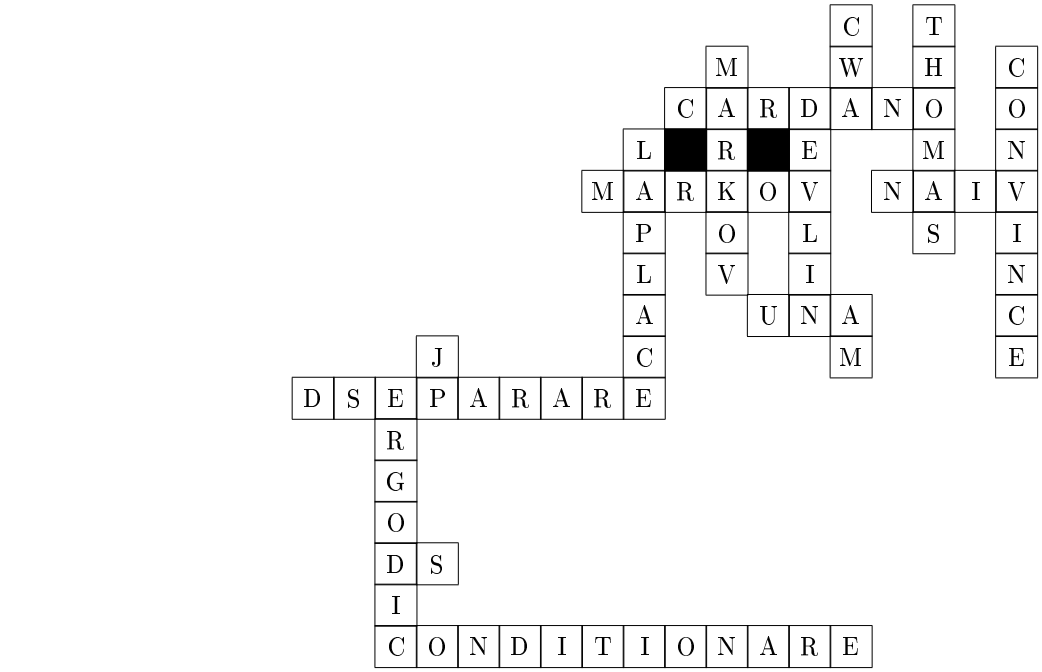
Probabilităţi



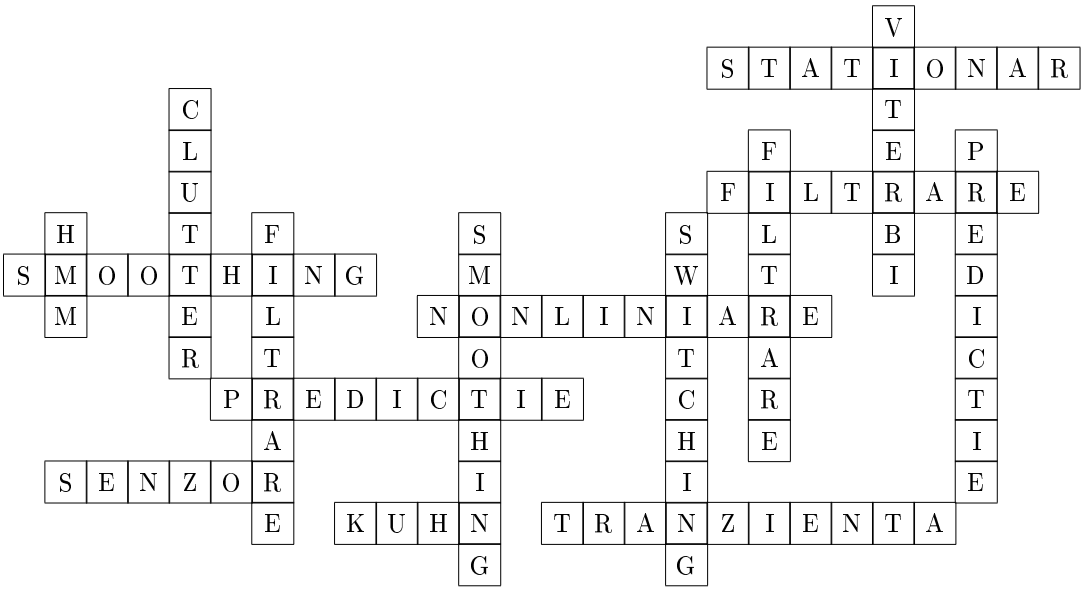
Raţionare în reţele Bayesiene



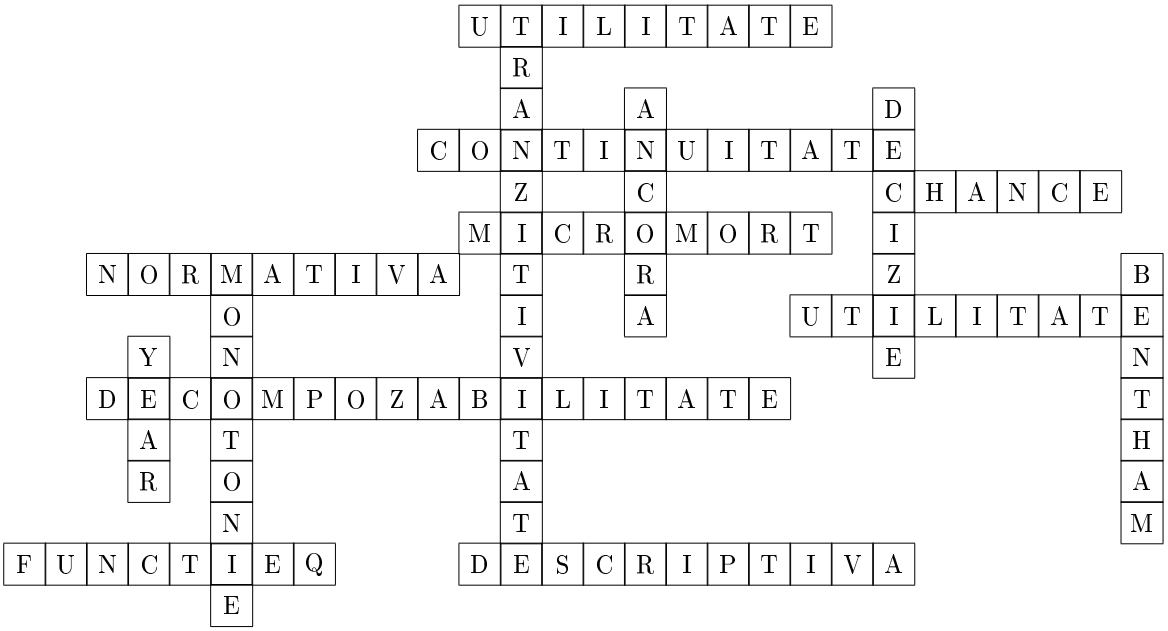
Reţele Bayesiene



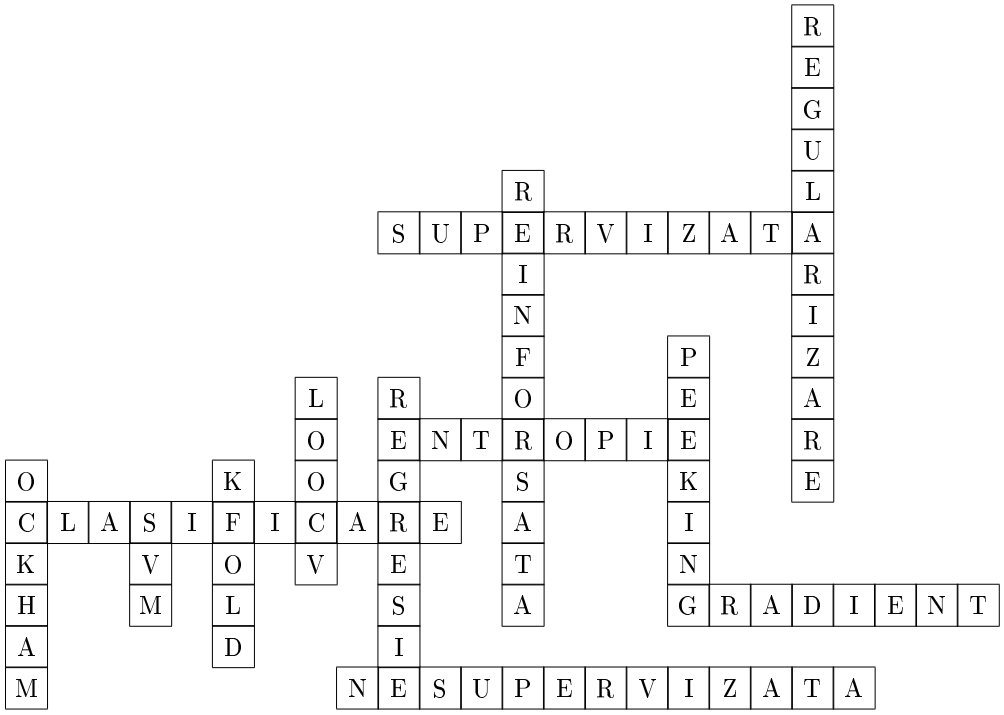
Filtru Kalman



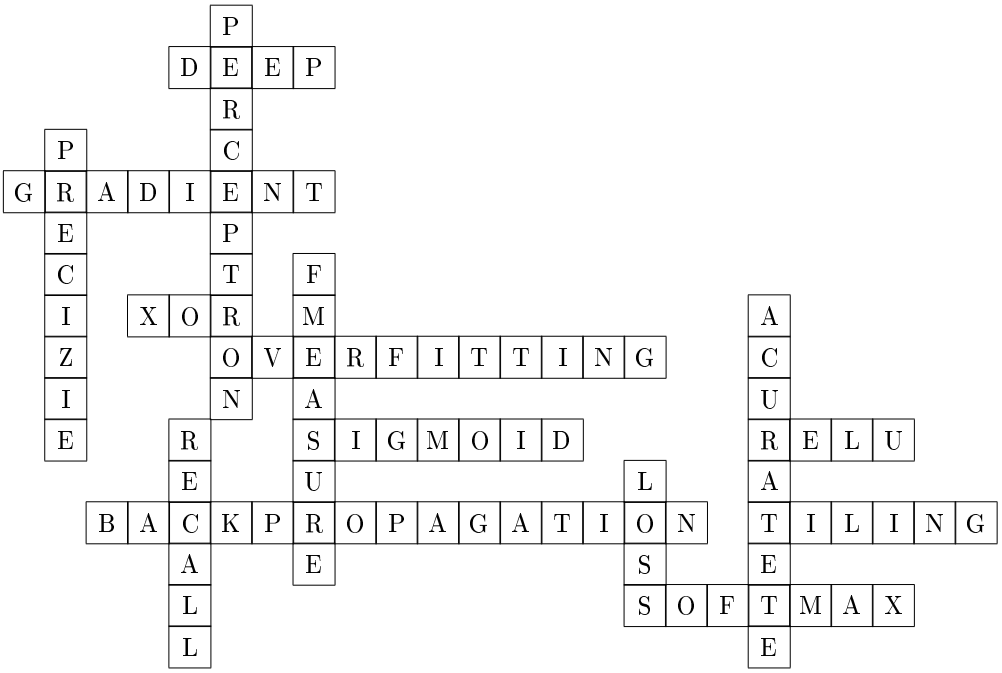
Decizii simple



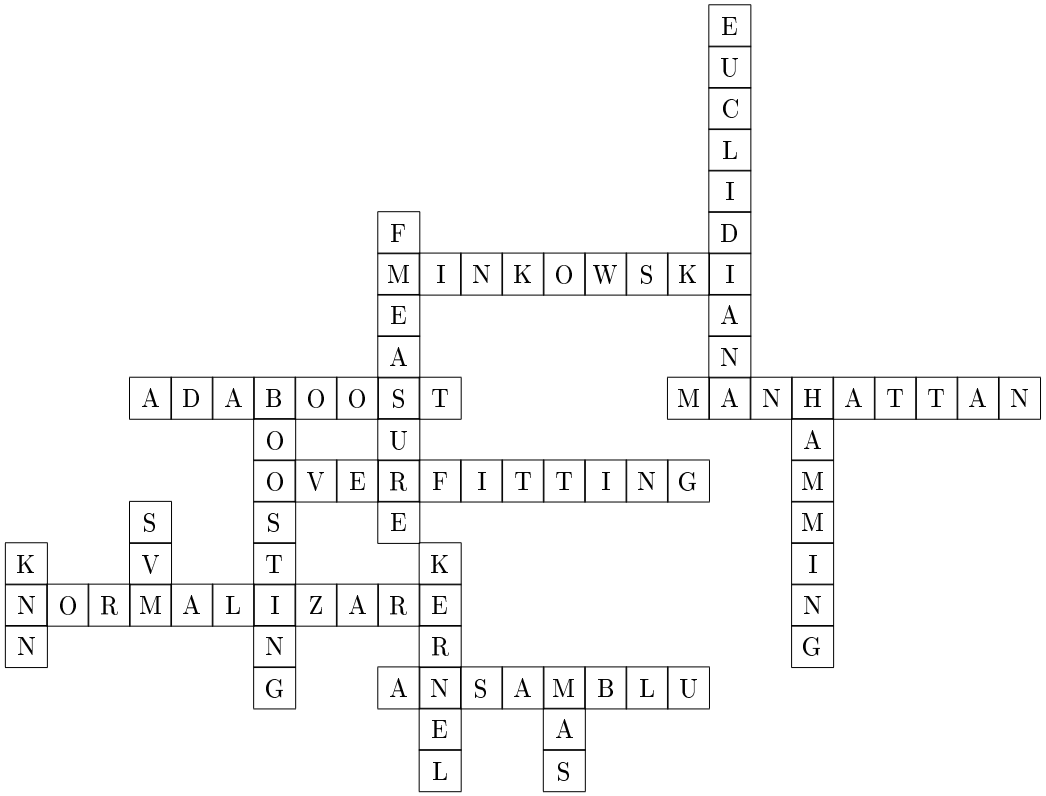
Învăţare supervizată. Arbori de decizie



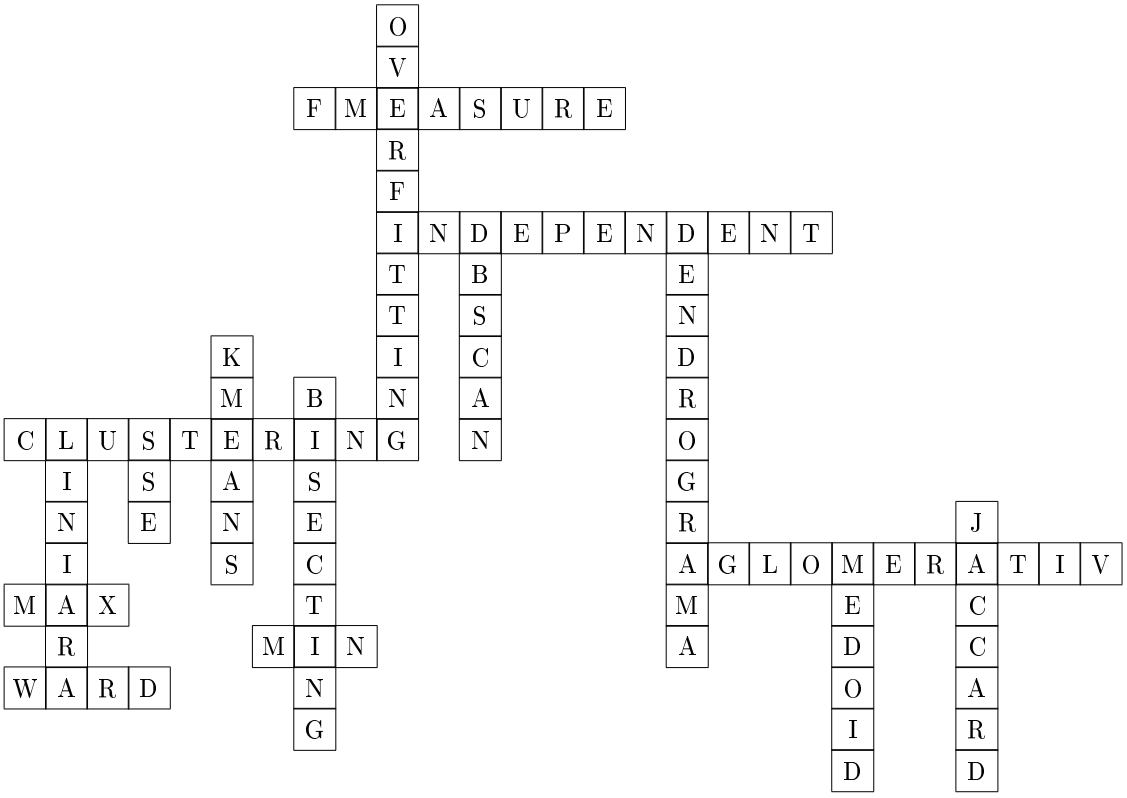
Reţele neurale



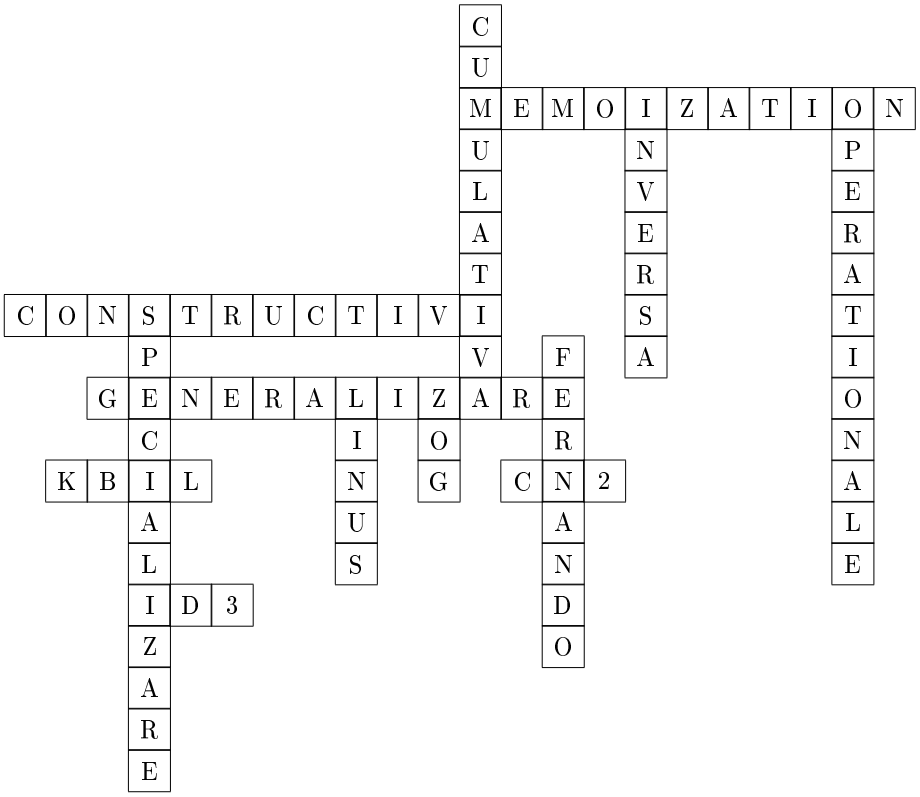
Ansambluri de clasificatori



Învăţare nesupervizată



Învăţare bazată pe cunoştinţe



Bibliografie

[1] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson Education Limited, 2016.