



**Adrian GROZA**

# **INTELIGENȚĂ ARTIFICIALĂ EXERCII**



**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](mailto: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

Copyright © 2019 Editura U.T.PRESS

Reproducerea integrală sau parțială a textului sau ilustrațiilor din această carte este posibilă numai cu acordul prealabil scris al editurii U.T.PRESS.

ISBN 978-606-737-374-5

# Cuprins

<b>I</b>	<b>Exerciții</b>	<b>1</b>
1	Întrebări cu răspuns multiplu	2
2	Rebusuri în inteligență artificială	41
<b>II</b>	<b>Soluții și explicații</b>	<b>64</b>
3	Întrebări cu răspuns multiplu	65
4	Rebusuri în inteligență artificială	129

# Cuvânt înainte

Despre inteligența artificială se vorbește din ce în ce mai mult. Ca urmare, cei care încep să studieze inteligența artificială sunt mai eterogeni și au metode și profile de învățare diferite. Exercițiile din această carte vin sub forma unor jocuri din două motive. Pe de o parte, limbajul universal al jocului permite mai multor tipuri de studenți să se apropie de domeniul vast și deloc simplu al inteligenței artificiale. Pe de altă parte, jocul este un mecanism încă eficient în creșterea motivației studenților care rezonază tot mai puțin cu învățarea doar pe baza unui curs. Jocurile se potrivesc profilului studentului contemporan care nu respinge competiția, care dorește un câștig imediat chiar dacă este mic sau care este adeseori ghidat de imperativul distracției.

Această carte este un instrument ajutător celor care studiază inteligența artificială utilizând manualul *Artificial Intelligence - A Modern Approach* (AIMA) a lui Russell și Norvig [1]. AIMA este văzută de către studenți ca extrem de densă. Pentru a sparge această nucă tare numită AIMA, am dorit oferirea către studenți a unui instrument complementar care să-i ajute în procesul de construire a hărții cognitive proprii pe domeniul inteligenței 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](http://www.kahoot.com)), respectiv 2) rebusuri în inteligență artificială.

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

Jocuri de tip rebus au fost dezvoltate pentru fiecare capitol din AIMA. Studenții 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 inteligența artificială care sunt prezentate doar într-o singură propoziție în AIMA. Aceste metode ar putea trece neobservate de către studenții care citeșc din AIMA doar părțile relevante pentru înțelegerea prezentărilor din timpul cursului.

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

# Partea I

## Exerciții

# Capitolul 1

## Întrebări cu răspuns multiplu

## 1. Introducere în inteligență artificială

- (a) Când s-a născut inteligența 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 raționare logică
- (b) Testul lui Turing tratează IA din perspectiva (30s)
- gândirii umane
  - gândirii raționale
  - comportamentului uman
  - comportamentului rațional
- (c) Perspectiva AIMA asupra IA impune agenților să (30s)
- gândească uman
  - gândească rațional
  - acționeze uman
  - acționeze rațional
- (d) Numele primului calculator care a învins campionul mondial la șah? (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)
- învățare automată și viziune computerizată
  - viziune computerizată și robotică
  - procesarea limbajului natural
  - înțelegerea limbajului și robotică
- (g) Predicții legate de testul Turing (30s)
- până în 2000, 30% șanse ca cineva să fie indus în eroare pentru 5 minute
  - până în 2000, 30% șanse ca un expert să fie indus în eroare
  - până în 2000, să fie păcălită orice persoană
  - 20% șanse 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 supremația a fost

(30s)

- Șah
- Go
- Spinner
- Game of Thrones



## 2. Agenți

- (a) Raționalitatea implică (30s)
- omnisciență, învățare, autonomie
  - explorare, învățare, autonomie
  - culegere de informații, învățare, omnisciență
  - succes, învățare, 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 și funcția de performanță se schimbă
  - mediul se schimbă, funcția de performanță nu
  - mediul nu se schimbă, funcția de performanță se schimbă
  - mediul și funcția de performanță nu se schimbă
- (d) În medii nondeterministe stărilor succesoare le sunt atașate probabilități (30s)
- Adevărat
  - Fals
- (e) Mediul în jocurile de tip *rebus* este (30s)
- deterministic și continuu
  - static și parțial observabil
  - multi-agent și secvențial
  - nicio variantă din cele anterioare
- (f) Mediul pentru șah fără ceas este (30s)
- semi-static și complet observabil
  - static și complet observabil
  - continuu și secvențial
  - episodic și multi-agent
- (g) Mediul pentru șoferul de taxi este (30s)
- continuu și complet observabil
  - dinamic și complet observabil
  - continuu și secvențial
  - episodic și multi-agent
- (h) Un agent reflex bazat pe model menține o stare interioară (30s)
- Adevărat
  - Fals

- (i) Agenții bazați pe utilitate (30s)
- utilizează reguli de tip *conditie*  $\rightarrow$  *actiune*
  - aleg acțiunea care îi conduce la cea mai bună utilitate expectată
  - folosesc planificarea pentru a atinge starea țintă
- (j) Un agent care învață își poate îmbunătăți (30s)
- doar cunoștințele inițiale
  - doar funcția de evaluare a performanței
  - doar generatorul de probleme utilizat în învățare
  - toate elementele sale

### 3. Căutare neinformată

- (a) Dacă agentul nu știe care este starea lui inițială, 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 soluției
  - 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 găsește întotdeauna soluția dacă aceasta există este (30s)
- complet
  - optimal
- (d) Care este cea mai bună strategie de căutare pe un calculator cu memorie puțină? (30s)
- în adâncime
  - în lățime
  - cost uniform
- (e) Căutarea în adâncime găsește cea mai bună soluție (30s)
- adevărat
  - fals
- (f)  $A^*$  este un algoritm de tip greedy ghidat de funcția de estimare a costului  $f(n) =$  (30s)
- costul de la nodul  $n$  la țintă
  - costul de la starea inițială la nodul  $n$
  - maximul dintre costul de la starea inițială la  $n$  și costul de la  $n$  la țintă
  - 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 poziția finală
  - $h_2 =$  suma distanțelor de la fiecare piesă la poziția ei finală
  - $h_3 = \max(h_1, h_2)$
  - $h_4 = \text{sum}(h_1, h_2)$

#### 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* pornește 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 rulați în paralel. (30s)
- Adevărat
  - Fals
- (g) Care nu este un operator genetic în algoritmi genetici? (30s)
- ADN
  - mutație
  - încrucișare
  - clonare
- (h) Într-un nod de tip AND, ramificația este introdusă de (20s)
- alegerile mediului
  - alegerea făcută de agent
- (i) *Belief state* reprezintă: (30s)
- setul stărilor posibile în care un agent poate fi
  - spațiul de căutare al unui agent religios
  - cea mai probabilă stare în care agentul poate fi
  - spațiul de căutare al unui agent ateist

## 5. Căutare adversarială

- (a) Minimax execută în arborele jocului o căutare de tipul (20s)
- în adâncime
  - în lățime
  - cost uniform
  - A\*
- (b) Ordinea de analiză a mutărilor nu afectează eficiența eliminării nodurilor (pruning) (30s)
- fals
  - adevărat
- (c) Tăierea cu alpha-beta nu afectează rezultatul final (10s)
- adevărat
  - fals
- (d) În practică, pentru evaluarea poziției se utilizează o combinație de atribute (20s)
- liniară
  - nonliniară
- (e) Căutarea de tip *quiescent* analizează pozițiile 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 poziții în jocuri stocastice este (30s)
- media tuturor rezultatelor posibile ale nodurilor de tip *min* și *max*
  - suma tuturor rezultatelor posibile ale nodurilor de tip *frunză*
  - media tuturor rezultatelor posibile ale nodurilor de tip *șansă*
  - suma tuturor rezultatelor posibile ale nodurilor de tip *șansă*
- (h) În jocuri stocastice, comportamentul se prezervă pentru orice transformare a funcției de evaluare care este (30s)
- monotonă
  - nonmonotonă
  - liniar pozitivă
  - liniar negativă
- (i) În jocuri stocastice, căutarea la adâncimi mari este esențială (30s)
- adevărat
  - fals

## 6. Probleme de satisfacere a constrângerilor

- (a) Algoritmul de backtracking poate fi îmbunătățit prin: (30s)
- nu poate fi îmbunătățit!
  - sortarea variabilelor în ordine lexicografică
  - detectarea timpurie a eșecului 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 instanțiază la pasul următor
  - valorii din domeniu care se asignează variabilei curente
- (c) Euristică *valorile minime rămase* selectează (30s)
- variabilele cu cele mai multe constrângeri pe variabile rămase
  - variabila cu cele mai puține valori disponibile
  - valoarea care elimină cele mai puține valori ale variabilor învecinate
  - valorile care introduc cele mai multe constrângeri în variabile învecinate
- (d) Euristică *grad* selectează (30s)
- variabilele cu cele mai multe constrângeri pe variabile rămase
  - variabila cu cele mai puține valori disponibile
  - valoarea care elimină cele mai puține valori ale variabilelor învecinate
  - valorile care introduc cele mai multe constrângeri în variabilele învecinate
- (e) Euristică *cele mai puține constrângeri* selectează (30s)
- variabilele cu cele mai multe constrângeri pe variabile rămase
  - variabila cu cele mai puține valori disponibile
  - valoarea care elimină cele mai puține valori ale variabilelor învecinate
  - valorile care introduc cele mai multe constrângeri în variabilele învecinate
- (f) Pentru a detecta eșecul cât mai repede posibil se utilizează euristica (30s)
- valorile minime rămase
  - grad
  - cele mai puține constrângeri
- (g) Pentru a reduce factorul de ramificare se utilizează (30s)
- valorile minime rămase
  - grad
  - cele mai puține 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. Agenți logici

- (a) Lumea lui Wumpus este (30s)
- complet observabilă și deterministă
  - episodică și observabilă local
  - statică și include un singur agent
  - multi-agent și secvențială
- (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 și numai dacă (20s)
- $M(KB) \subseteq M(\alpha)$
  - $M(\alpha) \subseteq M(KB)$
- (d) Propoziția  $p \rightarrow q \equiv \neg q \rightarrow \neg p$  este (30s)
- falsă
  - adevărată
  - depinde doar de  $p$
  - depinde doar de  $q$
- (e) Contrapозиția 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 implicației conține (10s)
- 4 de zero
  - 3 de zero
  - 2 de zero
  - 1 singur zero
- (g) Care din următoarele propoziții 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)
- $a \rightarrow b$
  - $a \rightarrow \neg b$



$a \rightarrow b \wedge c$

$a \rightarrow b \vee c$

(i) Raționarea înainte este (30s)

ghidată de țel

ghidată de date

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

$a \rightarrow b$

$a \wedge b$

$\neg a \vee b$

$\neg(a \wedge b)$

(k) Pentru a demonstra  $\alpha$ , rezoluția 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, relații, funcții
  - timp, credințe, cunoștințe
  - constrângeri
  - aceeași expresivitate cu logica propozițională
- (d) *Fiecăruia îi place înghețata* 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 iubeste 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 isteț* 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) Relația *sibling* este simetrică (20s)
- $Sibling(x)$

- $\forall x, y \text{ Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x)$
- $\text{Sibling}(x, y) \vee \text{Sibling}(y, x)$
- $\exists x, y \text{ Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x)$

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

- $\exists x, y \text{ Mother}(x, y) \equiv (\text{Female}(x) \wedge \text{Parent}(x, y))$
- $\forall x, y \text{ Mother}(x, y) \equiv (\text{Female}(x) \wedge \text{Parent}(x, y))$
- $\forall x, y \text{ Mother}(x, y) \equiv \text{Parent}(x, y)$
- $\forall x, y \text{ Mother}(x, y) \wedge (\text{Female}(x) \wedge \text{Parent}(x, y))$

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

- $\forall x \exists y, \text{iubeste}(x, y)$
- $\exists x \forall y, \text{iubeste}(x, y)$
- $\forall x \exists y, \text{iubeste}(y, x)$
- $\exists x \forall y, \text{iubeste}(y, x)$

## 9. Inferență în logica de ordinul întâi

- (a) Un termen *ground* este (20s)
- un termen cu cel puțin o constantă
  - un termen fără variabile
  - un sol unde un arbore binar poate fi plantat
  - niciuna dintre aceste opțiuni
- (b) *Lifting* înseamnă (20s)
- transformarea din logică propozițională în logica de ordinul întâi
  - eliminarea constantelor Skolem
  - introducerea constantelor Skolem
- (c) Datalog = (10s)
- Prolog + funcții
  - baze de date extinse cu reguli
  - clauze în logica de ordinul întâi plus funcții
  - clauze în logica de ordinul întâi fără funcții
- (d) O bază de date deductivă (20s)
- utilizează inferența pentru a răspunde la interogări
  - combină programarea logică cu baze de date relaționale
  - mai expresivă decât o bază de date, dar mai puțin decât un limbaj logic
- (e) Skolemizarea este procesul de (10s)
- eliminare a cuantificatorilor universali
  - eliminare a cuantificatorilor existențiali
  - introducere a cuantificatorilor universali
  - introducere a cuantificatorilor existențiali
- (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 conținută este o (20s)
- disjuncție de literari
  - conjuncție de literari
- (i) Care nu este o metodă de inferență logică? (10s)

- raționarea înainte
- raționarea înapoi
- skolemizarea
- rezoluția

(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 condiție deschisă este o precondiție (10s)
- cu un termen care nu are toate variabilele instanțiate
  - cu prezumția de lume deschisă
  - a unui pas fără legături cauzale
- (c) Un *clobber* (20s)
- validează o precondiție obținută datorită unui efect al unei acțiuni
  - distruge o precondiție obținută printr-o legătură cauzală
  - introduce o relație de ordine parțială între acțiuni
- (d) Anomalia Sussman (20s)
- ilustrează dezavantajele planificării greedy
  - oprește algoritmul de planificare, chiar dacă un plan există.
  - poate fi rezolvată cu ajutorul *clobberilor*
- (e) O acțiune este relevantă dacă (20s)
- poate fi următorul pas într-un plan
  - cel puțin unul din efecte se unifică cu un element din starea finală
  - are toate precondițiile satisfăcute
- (f) Care sunt euristici admisibile pentru planificare? (20s)
- $h_1$  = ignorarea precondițiilor
  - $h_2$  = ignorarea stării inițiale
  - $h_3$  = ignorarea efectelor care elimină fapte din starea curentă
  - $h_4$  = ignorarea stării finale
- (g) Ce se elimină pentru  $h = \text{Manhattan}$  în:  $on(t, s1)$ ,  $tile(t)$ ,  $blank(s2)$ ,  $adj(s1, s2)$ ? (30s)
- $blank(s2)$
  - $adj(s1, s2)$
  - $on(t, s1)$
  - toate precondițiile
- (h) Ce trebuie eliminat pentru  $h = \text{numărul de piese plasate greșit}$  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 precondițiile

## 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 obținerea soluției optime
  - nicio variantă din cele de mai sus
- (e) Este mai bine să avem efecte condiționate decât o acțiune 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 și cea contingentă
  - nicio opțiune din cele anterioare
- (g) Pentru identificarea succesului accidental (serendipity) agentul monitorizează din mediu: (20s)
- acțiunile
  - planul
  - țelul
- (h) La pierderea semnalului GPS, navigatorul e mai bine să execute (20s)
- replanificare

- repararea planului
  - depinde de lungimea planului și perioadă de pierdere a semnalului
  - se oprește și așteaptă instrucțiuni de la agentul uman
- (i) Acțiunile concurente sunt necesare în cazul (20s)
- monitorizării planului
  - planificării cu contingente
  - planificării multi-agent
  - planificării online
- (j) În cazul informațiilor incomplete se utilizează (20s)
- planificarea condițională
  - planificarea cu contingente
  - planificarea fără senzori
  - monitorizarea execuției și replanificare
- (k) În cazul informațiilor eronate se utilizează (20s)
- planificarea condițională
  - planificarea contingentă
  - planificarea fără senzori
  - monitorizarea execuției și replanificare



## 12. Calculul evenimentelor

- (a) Care nu este o sarcină de raționare în calculul evenimentelor? (30s)
- postdicția
  - rezoluția
  - abducția
  - predicția
- (b) Se poate face planificare în calculul evenimentelor prin (30s)
- predicție
  - abducție
  - postdicție
  - doar limbajul PDDL poate fi utilizat pentru planificare
- (c)  $\neg HoldsAt(Awake(Nathan), 0)$  este (30s)
- o observație
  - axiomă cu efect pozitiv
  - axiomă cu efect negativ
  - apariția unui eveniment
- (d)  $Happens(Awake(Nathan), 1)$  (30s)
- o observație
  - axiomă cu efect pozitiv
  - axiomă cu efect negativ
  - apariția 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 relație injectivă
  - Ocupa* este o relație surjectivă
- (f)  $\neg HoldsAt(On(o, o), t)$  (30s)
- On* este o relație reflexivă
  - On* este o relație 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 însuși
  - telefonul  $p$  nu sună la timpul  $t$
- (h)  $\neg HoldsAt(Broken(d), t) \rightarrow Initiates(TurnOn(a, d), On(d), t)$  (30s)
- condiție de tip fluent
  - condiție de tip acțiune
- (i)  $Happens(WalkThroughDoor(a, d), t) \rightarrow HoldsAt(Near(a, d), t)$  (30s)
- condiție de tip fluent
  - condiție de tip acțiune

### 13. Probabilități condiționale

	Joacă fotbal	Nu joacă	Total
Băieți	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)  $P(\text{baiat}|\text{joacaFotbal}) =$  (20s)
- 42/110
  - 42/54
  - 42/75
  - 54/110
- (c)  $P(\text{baiat} \wedge \text{joacaFotbal}) =$  (20s)
- 42/110
  - 42/54
  - 42/75
  - 54/110
- (d)  $P(\text{baiat} \vee \text{joacaFotbal}) =$  (20s)
- 42/110
  - 54/110
  - 75/110
  - 87/110
- (e) Sunt variabilele *baiat* și *joacaFotbal* independente? (60s)
- Da
  - Nu
- (f) Care este probabilitatea de a extrage o carte roșie 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 cuvântul ARTIFICIAL? (20s)
- 1/5
  - 3/10

1/10

1/2

(h) Care este probabilitatea de a obține un multiplu de 3 dintr-o aruncare cu zarul și stema unei monede? (20s)

1/6

2/6

1/4

33%

#### 14. Rețele 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 dați părinții + copiii + copiii părinților
  - este echivalenta cu tabelul complet al probabilităților
  - nodurile sunt independente de nondescendenți, fiind dați părinții
- (d) Noisy-OR consideră (20s)
- toate cauzele posibile sunt incluse
  - probabilitățile de eșec 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 rețea Bayesiană
  - acoperă toate cauzele de tipul *și altele* într-o relație Noisy-OR
  - un nod condițional independent față de rădăcina rețelei Bayesiene.
- (f) O rețea Bayesiană hibridă (20s)
- este o rețea bayesiană modificată de un algoritm genetic
  - conține atât variabile aleatoare booleene, cât și nonbooleene
  - conține atât variabile discrete, cât și continue
  - modelează cel puțin două domenii

### 15. Inferență în rețele Bayesiene

- (a) Care nu este un algoritm de inferență exactă? (20s)
- inferență prin enumerare
  - inferență prin eliminarea variabilelor
  - inferență prin simulare stocastică
  - inferență 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 folosește factori? (20s)
- inferență prin enumerare
  - inferență prin eliminarea variabilelor
  - inferență prin simulare stocastică
  - inferență prin Markov Chain Monte Carlo
- (e) Metoda *Rejection Sampling* generează doar evenimente consistente cu evidențele  $e$  (20s)
- Adevărat
  - Fals
- Eșantionarea prin revocare (rejection sampling) este o metodă de raționare aproximativă în rețele Bayesiene. Metoda generează eşantioane conform distribuției de probabilitate specificate. Dintre aceste eşantioane le păstrează doar pe cele consistente cu evidențele date. Probabilitatea condiționată este obținută prin numărarea cazurilor favorabile din eşantionul rămas.
- (f) Metoda *MCMC* generează eşantioane integrale (20s)
- Fals
  - Adevărat
- (g) Algoritmul *d-separare* păstrează (20s)
- toate variabilele menționate în interogare și toți părinții acestora
  - toate variabilele menționate în interogare și toți ancesorii acestora
  - toate variabilele menționate în interogare și toți copiii acestora
  - toate variabilele menționate în interogare și toți descendenții acestora

(h) Metoda *likelihood weighting*

(20s)

- folosește evidențele pentru a pondera eşantioanele generate
- generează eşantioane dintr-o rețea goală
- respinge eşantioanele inconsistente cu evidențele

## 16. Raționare probabilistică cu timp

- (a) În asumptia Markov, starea curentă depinde de (20s)
- starea anterioară
  - un număr finit și fix de stări anterioare
  - de părinți + copii + copiii părinților
- (b) Un proces staționar 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) Asumptia 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)
- $P(X_t|e_{1:t})$
  - $P(X_{t+k}|e_{1:t})$
  - $P(X_k|e_{1:t}), k < t$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_{1:t})$
- (e) Netezirea (smoothing) este formalizată cu (30s)
- $P(X_t|e_{1:t})$
  - $P(X_{t+k}|e_{1:t})$
  - $P(X_k|e_{1:t}), k < t$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_{1:t})$
- (f) Predicția este formalizată cu (30s)
- $P(X_t|e_{1:t})$
  - $P(X_{t+k}|e_{1:t})$
  - $P(X_k|e_{1:t}), k < t$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_{1:t})$
- (g) Identificarea celei mai probabile explicații (30s)
- $\operatorname{argmax}(X_{1:t})P(X_t|e_{1:t})$
  - $\operatorname{argmax}(X_{1:t})P(X_k|e_{1:t}), k < t$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_t)$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_{1:t})$
- (h) Care din următoarele metode nu este o tehnică de raționare în modele temporale? (20s)
- rezoluția
  - predicția

- netezirea
  - identificarea celei mai probabile explicații
- (i) Pentru a putea raționa într-o rețea Bayesiană dinamică nu e nevoie de (20s)
- modelul stării inițiale
  - modelul de tranziție
  - modelul senzorilor
  - distribuția Gaussiană
- (j) Învățarea unei rețele Bayesiene dinamice se referă la (60s)
- a învăța din observații modelul de tranziții
  - a interpreta orice nod din rețea ca un neuron
  - estimarea stării, filtrare și netezire
  - a învăța din observații modelul senzorilor



## 17. Decizii simple

- (a) Micromortul este (20s)
- unitatea de măsură pentru utilitate
  - 1/1,000,000 șanse de deces
  - măsura pentru metrica QUALY (Quality Adjusted Life Years)
  - identic cu microprobabilitatea
- (b) Sistemul de asigurări are la bază faptul că agenții umani sunt față de risc (30s)
- neutri
  - împotriva
  - în căutarea riscului
- (c) Funcția de utilitate a banilor este (30s)
- liniar crescătoare
  - constantă
  - logaritmică
  - exponențială
- (d) La ce este utilizat principiul MEU? (30s)
- Minimize Expected Utility
  - calcularea utilității
  - selectarea acțiunii în starea curentă
  - explicarea comportamentului irațional
- (e) Cum se dorește să acționeze agenții software (perspectiva AIMA)? (30s)
- deterministic
  - rațional
  - cât mai uman
  - rapid
- (f) Cu ce se extinde o rețea Bayesiană pentru a deveni o rețea de decizie? (30s)
- noduri de tip acțiune și utilitate
  - noduri utilitate
  - noduri MEU
  - Cu nimic. O rețea Bayesiană este deja o rețea de decizie
- (g) Valoarea expectată a informației este (30s)
- negativă
  - aditivă
  - depinde de ordinea în care sunt primite informațiile
  - nicio versiune din cele anterioare
- (h) Eroare cognitivă în care deciziile sunt afectate de o informația inițială (30s)
- efectul certitudinii

- încadrarea (framing)
  - efectul de ancoră
- (i) Eroare cognitivă în care deciziile sunt afectate de modul în care opțiunile sunt prezentate: pozitiv sau negativ (30s)
- efectul certitudinii
  - încadrarea (framing)
  - efectul de ancoră
- (j) Teoria deciziilor este o teorie (10s)
- normativă
  - descriptivă

## 18. Decizii secvențiale

- (a) Problemele de decizii secvențiale (SDP) (30s)
- includ utilități, incertitudine și captarea informațiilor din mediu
  - utilitatea agentului depinde de o secvență de decizii
  - sunt cazuri particulare ale problemelor de planificare
  - sunt cazuri particulare de probleme de decizie Markoviene
- (b) Problemele de decizie Markoviene sunt probleme secvențiale pentru mediile (60s)
- complet observabile și stocastice
  - complet observabile și deterministe
  - cu modele de tranziție Markoviene și funcții aditive de recompensă
  - parțial observabile și stocastice
- (c) O politică optimă  $\pi(s)$  (30s)
- minimizează suma expectată a recompenselor
  - specifică cea mai bună acțiune pentru fiecare stare  $s$
  - are cea mai mare valoare expectată a utilității
- (d) O politică optimală pentru un orizont finit este (20s)
- nonstaționară
  - staționară
- (e) Dacă factorul de discount tinde spre 0, atunci recompensele pe termen lung sunt (30s)
- nesemnificative
  - semnificative
- (f) Ecuația 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 licitații au caracter public și sunt descendente? (30s)
- englezești
  - olandeze
  - cu plic închis
  - Vickrey
- (h) O soluție este Pareto optimală dacă (30s)
- nu există o altă soluție pe care unul dintre agenți o preferă
  - nu există o altă soluție pe care toți agenții să o prefere
  - există o altă soluție pe care unul dintre agenți o preferă
  - există o altă soluție pe care toți agenții să o prefere

## 19. Arbori de decizie

- (a) Dacă predicatul țintă are un set finit de valori, atunci problema de învățare 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
  - $X^2 + 1/3$  față de  $X^3$
- (c) O problemă de învățare este realizabilă dacă (30s)
- spațiul ipotezelor conține funcția adevărată
  - datele sunt consistente și fără zgomote
  - se aplică un algoritm de învățare potrivit pentru date
- (d) Arborele de decizie este construit pe baza unei strategii de căutare de tipul (30s)
- căutare în adâncime
  - căutare în lățime
  - greedy și desparte-și-cucerește
  - 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 învățarea nu este deterministic
  - nu există acces la un atribut prin care s-ar putea distinge între instanțe
  - nici una din opțiunile anterioare
- (f) Mai multă informație înseamnă (10s)
- mai multă entropie
  - mai puțină 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 mulți parametri decât cei necesari
  - toate variantele anterioare
- (i) Validarea de tip hold-out nu utilizează în antrenare toate datele disponibile. (20s)
- Adevărat
  - Fals

- (j) Performanța învățării este dată de acuratețea prezicerii pe setul de (30s)
- antrenare
  - validare
  - testare
  - media acurateții pe cele 3 seturi
- (k) *Peeking* apare dacă (60s)
- ipoteza se selectează și pe baza erorii pe setul de testare
  - nu se utilizează set pentru validare
  - curba de învățare (ROC) este un *graf vesel*
  - atributul țintă este continuu

## 20. Cunoștințe în învățare

- (a) Care este diferența dintre regresie și clasificare? (30s)
- natura atributului țintă
  - o metodă este supervizată, iar cealaltă nesupervizată
  - o metodă utilizează statistica, iar cealaltă nu
  - nu e nicio diferență
- (b) Dacă ipoteza clasifică o instanță ca negativă dar corect ar fi pozitivă, avem (20s)
- fals pozitiv
  - fals negativ
  - adevărat pozitiv
  - adevărat negativ
- (c) Instanțele fals pozitive sunt rezolvate prin (30s)
- generalizare
  - specializare
- (d) Generalizarea are loc prin (30s)
- adăugarea de condiții
  - eliminarea unor condiții
- (e) Ce nu este adevărat legat de căutarea current-best-hypothesis? (30s)
- menține o singură ipoteză care se ajustează cu fiecare exemplu
  - este deterministică
  - modificarea ipotezei impune verificarea instanțelor anterioare
  - execută mult backtracking
- (f) Care nu este o condiție de stop a algoritmului *version space*? (60s)
- rămâne doar o ipoteză în spațiul soluțiilor
  - spațiul soluțiilor colapsează ( $\mathcal{G}$  sau  $\mathcal{S}$  devin vide)
  - rămân mai multe ipoteze și nu mai există instanțe de analizat
  - complexitatea ipotezei este mai mare decât un prag
- (g) Algoritmul FOIL evită suprapotrivirea (overfitting) prin (30s)
- validare încrucișată
  - penalizând ipotezele mari
  - folosind cunoștințe din domeniu
- (h) Învățarea bazată pe relevanță (30s)
- extrage reguli generale dintr-un singur exemplu
  - folosește cunoștințe anterioare
- (i) Ce nu este adevărat la *version space*? (30s)
- este o reprezentare hierahică a cunoștințelor

- valorile atributelor sunt continue
- utilizează arbori de generalizări și specializări
- presupune ca datele sunt corecte

## 21. Rețele neurale artificiale

- (a) Corespondențele pentru ⟨neuron, sinapse, dendrite, axon⟩ sunt: (30s)
- ⟨ponderi, nod, intrare, ieșire⟩
  - ⟨nod, ponderi, intrare, ieșire⟩
  - ⟨intrare, nod 3, ponderi 4, ieșire⟩
  - ⟨intrare, ponderi, nod, ieșire⟩
- (b) 20 Rețelele neurale sunt metode de învățare
- supervizată
  - nesupervizată
  - reinforșată
  - depinde de tipul rețelei neurale
- (c) Neuronul cu  $w_0 = 0.5$ ,  $w_1 = 1$  și  $w_2 = 1$  implementează poarta logică (60s)
- and
  - or
  - xor
  - not
- (d) Neuronul cu  $w_1 = 1$  și  $w_2 = 1$  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 funcția XOR (20s)
- Adevărat
  - Fals
- (f) Învățarea are loc prin ajustarea ponderilor pentru a reduce eroarea pe setul de test (20s)
- Adevărat
  - Fals
- (g) Ce ați răspunde unui medic legat de similaritatea dintre creier și 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 învățare cu ansambluri de clasificatori
  - un caz particular de arbore de decizie
  - o metodă de eficientizare a învățării prin paralelizare
- (k) Învățarea cu ansambluri poate utiliza clasificatori diferiți (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 partiționare
- (b) K-means are nevoie de date normalizate și nenominale (20s)
- True
  - False
- (c) În suma erorilor pătratice (SSE) pentru fiecare instanță se (30s)
- însumează distanța până la cel mai apropiat centroid
  - ridică la patrat distanța până la cel mai apropiat centroid și se însumează rezultatele
  - însumează distanțele la toate celelalte puncte din cluster
  - ridică la patrat distanțele față de toate celelalte puncte din cluster și se însumează rezultatele.
- (d) Dacă  $K$  crește atunci SSE (10s)
- scade
  - crește
- (e) Dacă setul de date conține 3 grupuri reale, care sunt șansele 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 soluție la problema inițializării algoritmului K-means? (30s)
- rulări multiple
  - Bisecting K-means
  - clusterizare ierarhică
  - preprocesare
- (g) Care din următoarele nu este o metrică pentru distanța dintre cluster? (30s)
- min (single link)
  - media grupului
  - dendograma
  - distanța dintre centroizi
- (h) Care metodă de clusterizare nu are nevoie de predefinirea numărului de cluster? (20s)
- Bisecting K-means
  - K-means
  - clusterizare hierarhică

- toate menționate anterior
- (i) Clusterizarea ierarhică cu metrica min (single link) (30)
  - nu este susceptibilă la excepții
  - 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 excepții

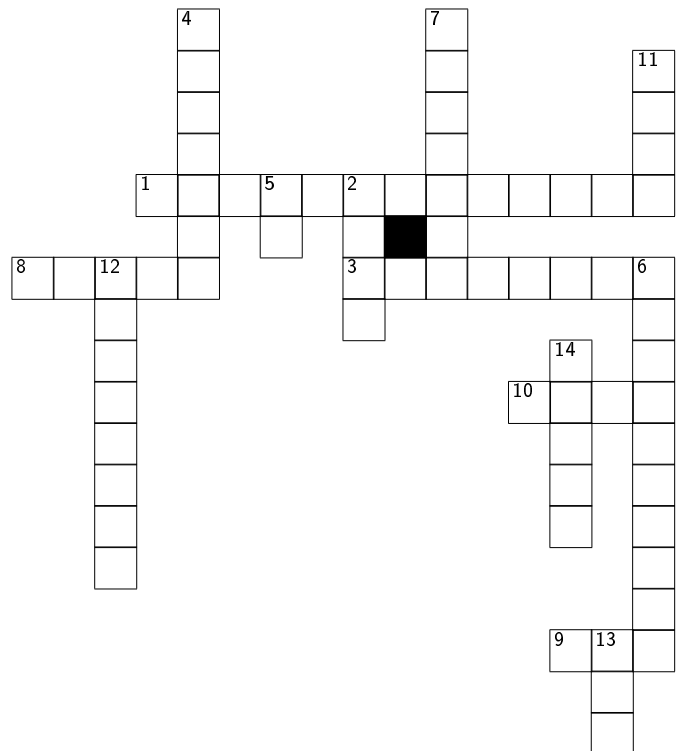
### 23. Reguli de asociație. Algoritmul Apriori

- (a) Este suportul o măsură simetrică? (10s)  
 nu  
 da
- (b) Este confidența 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 depășește suportul submulțimilor 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 mulțime de itemi frecvenți  $L$  ( $|L| = k$ )? (30s)  
  $2^k$   
  $2^{k-2}$   
  $2^k - 2$   
  $k!$
- (h) Confidența 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 inteligență artificială

## Introducere în inteligență artificială



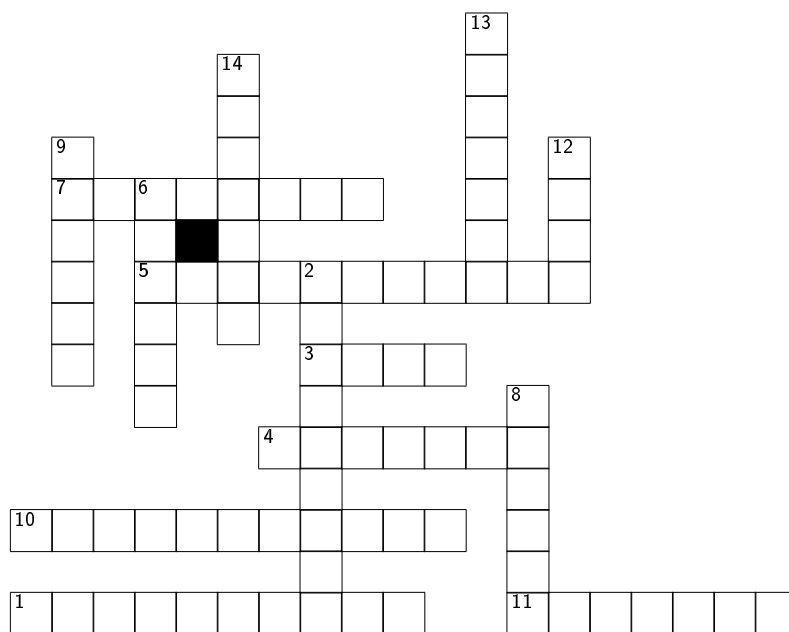
### Orizontal

- 1 Când IA depășește inteligența umană.
- 3 Formalizat de Aristotel
- 8 Vechi sistem expert în medicină
- 9 Mulți agenți
- 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 operațional programabil
- 12 Turing părea mic în preajma lui
- 13 Primul calculator electronic
- 14 Testul lui Turing extins cu viziune computerizată și robotică

## Agenți inteligenți



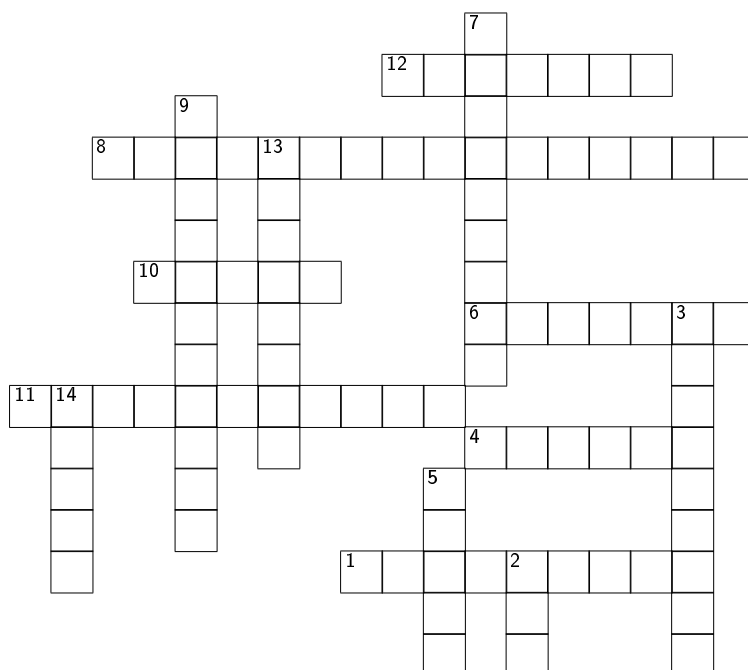
### Orizontal

- 1 Agent care știe tot ce se va întâmpla
- 3 Descrie performanța, acțiunile, senzorii agentului și caracterizează mediul (abr)
- 4 Agent software
- 5 Opuse jocurilor competitive
- 7 Mediul în care acțiunile nu depind de stările anterioare
- 10 Mediul nu se schimbă dar performanța agentului scade în timpul deliberării
- 11 Agent conversațional

### Vertical

- 2 Colectare de informație 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 percepție
- 14 Agent care se descurcă singur

## Rezolvarea problemelor prin căutare



### Orizontal

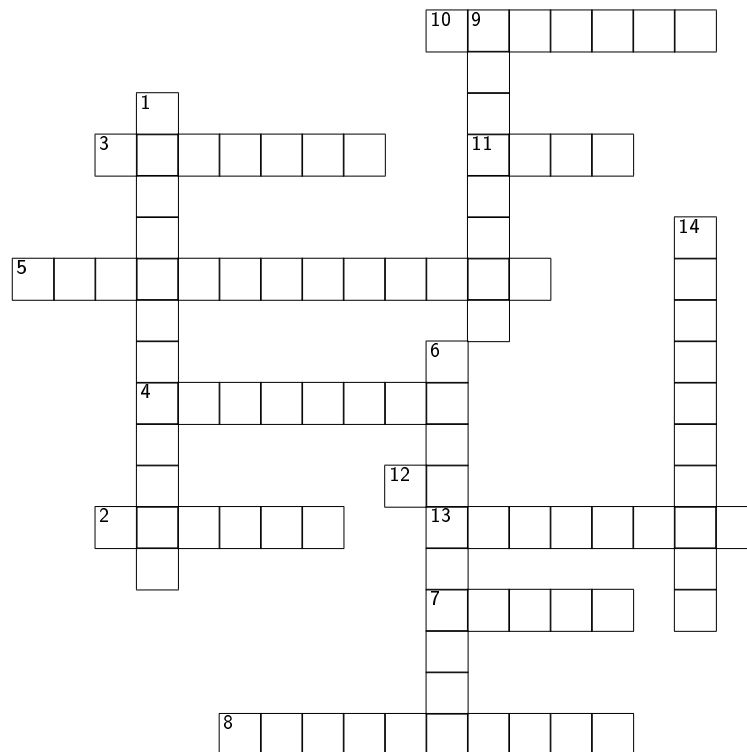
- 1 Stocheză nodurile frunză disponibile pentru expandare
- 4 Cea mai bună soluție
- 6 Algoritm care găsește soluția dacă aceasta există
- 8 Căutare în două sensuri
- 10 Algoritm informat
- 11 Euristici monotone
- 12 Taiere de arbori de căutare (en)

### Vertical

- 2 Problema comis voiajorului (abr, en)
- 3 Aplicarea acțiunilor legale pe starea curentă
- 5 Lista cu noduri explorate (en)
- 7 Utile în căutarea informată
- 9 Euristică optimistă
- 13 Problemă cu mai puține restricții
- 14 Căutare neinformată



## Căutare locală



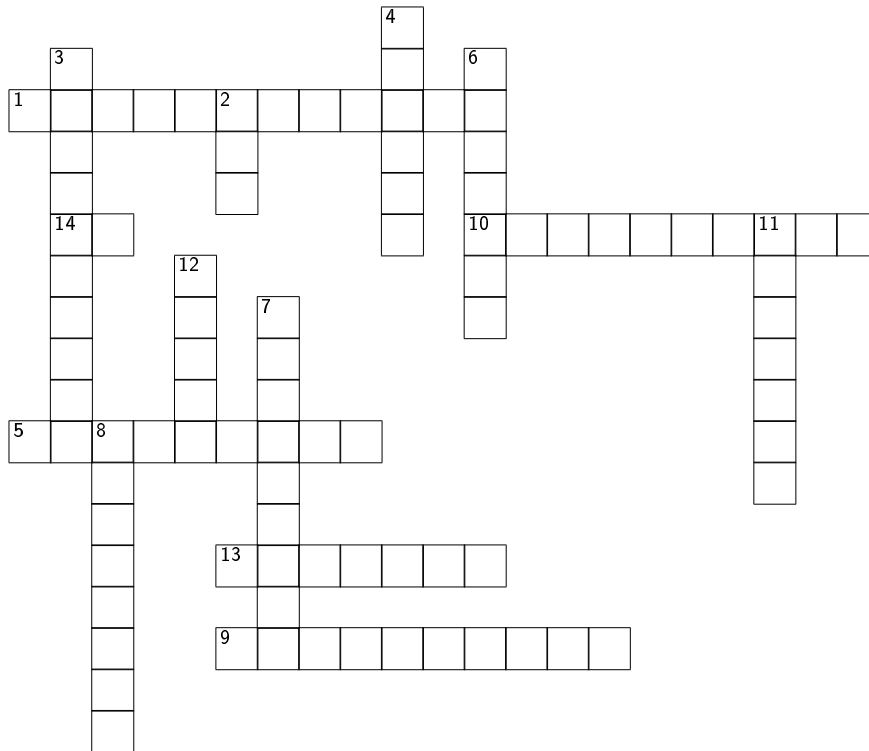
### Orizontal

- 2 Căutare în mediu necunoscut (en)
- 3 Funcție care evaluează indivizi (en)
- 4 Operator genetic
- 5 Operator genetic
- 7 Arbore de căutare pentru acțiuni 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 și ieșire

### 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 populație

## Căutare adversarială



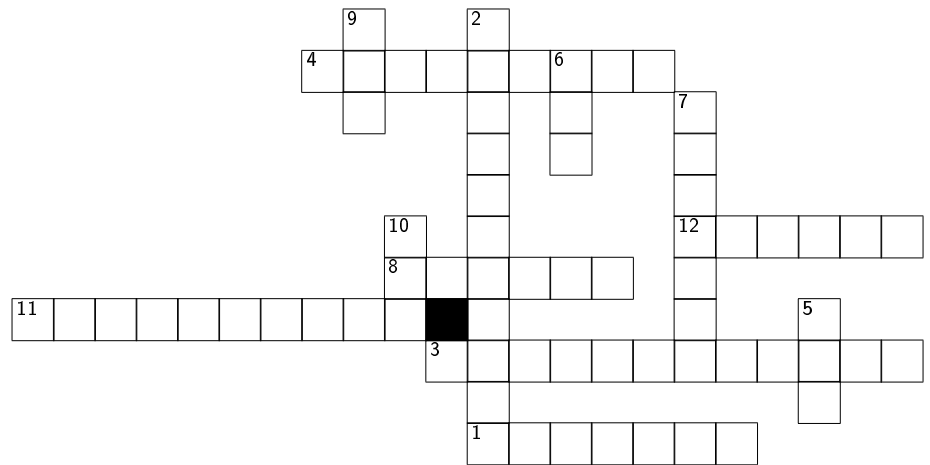
### Orizontal

- 1 Permutații ale mutărilor care conduc în aceeași stare
- 5 Tehnică de tăiere a arborilor
- 9 Jocuri cu elemente aleatoare
- 10 Simulare pentru evaluarea poziției
- 13 Jocul Othello
- 14 Joc popular în Asia cu factor mare de ramificație

### Vertical

- 2 O jumătate de mutare
- 3 Șah cu tabla parțial observabilă
- 4 Test care decide când se aplică funcția de evaluare a poziției (en)
- 6 Algoritm de căutare adversarială
- 7 Poziții care nu par să își schimbe valoarea în viitorul apropiat
- 8 Funcție care primește o stare și întoarce cea mai bună mutare
- 11 Simulare Monte Carlo pentru jocuri cu zaruri
- 12 Succesoria mitologică a lui Deep Blue

## Probleme de satisfacere a constrângerilor



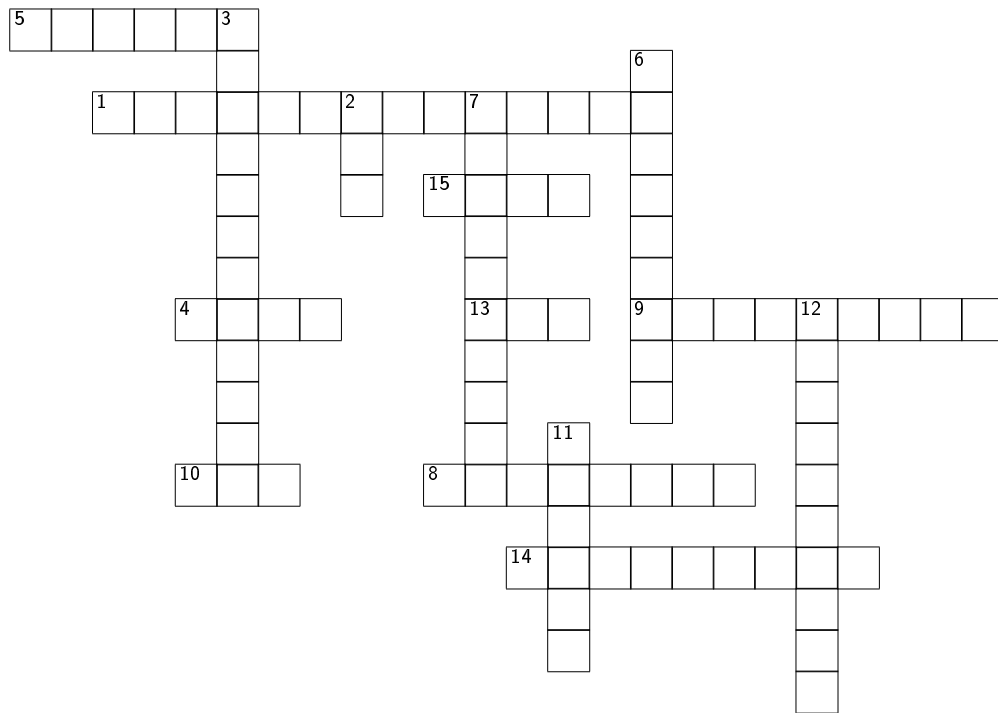
### Orizontal

- 1 Constrângere cu multe variabile
- 3 Euristică la CSP cu căutare locală (en)
- 4 Inferență 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 instanțieri a unei variabile din setul conflict
- 5 Algoritm pentru consistența muchiilor dezvoltat în 1977 de Mackworth
- 6 Algoritm pentru consistența arcului
- 7 Tip de constrângere globală aplicată și la Sudoku
- 9 Euristică celor mai puține valori rămase
- 10 Algoritm pentru menținerea consistenței arcului

## Agenți logici



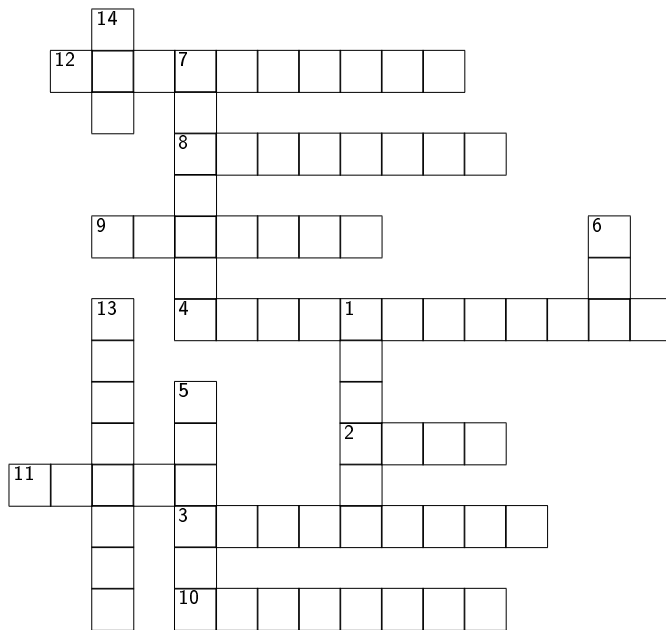
### Orizontal

- 1  $a \rightarrow b \equiv \neg b \rightarrow \neg a$
- 4 Propoziție cu un singur literar pozitiv
- 5 Celebru monstru care duhnește în labirint
- 8  $\neg(a \wedge b) \equiv \neg a \vee \neg b$
- 9 Deducerea de propoziții 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 Propoziție adevărată în cel puțin o lume posibilă
- 6 Înțelesul propozițiilor dintr-o logică
- 7 Operator logic pentru  $\neg a \vee b$
- 11 Regulă de inferență (lt)
- 12 Algoritm complet de inferență care folosește reducerea la absurd

## Logica de ordinul întâi



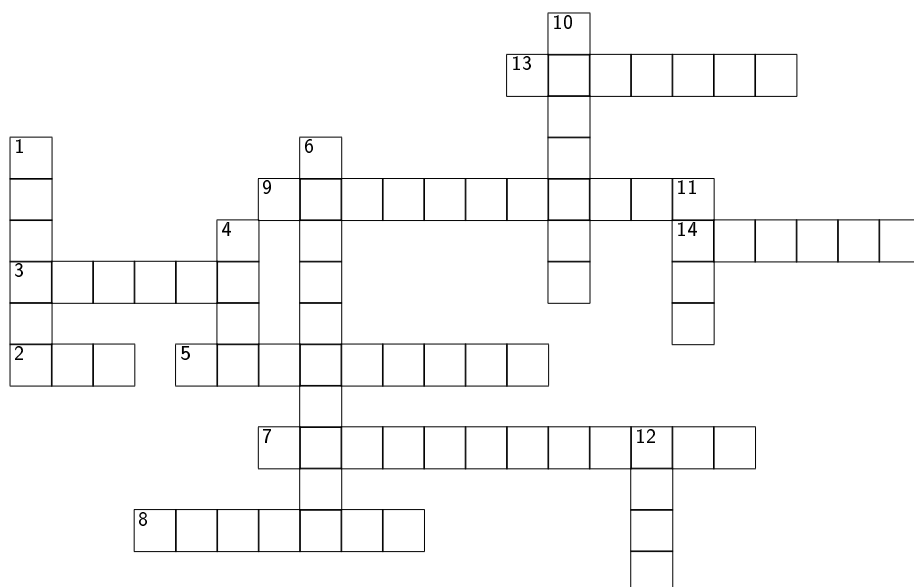
### Orizontal

- 2 Propoziție 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 Propoziție validă

### Vertical

- 1 Funcții care au valori pentru fiecare intrare din domeniul specificat
- 5 Termen fără variabile (en)
- 6 Asumpție în care propozițiile despre care nu știm că sunt adevărate sunt false (en)
- 7 Propozitții care nu sunt axiome
- 13 Regulă care leagă o cauză de efectul ei
- 14 Primul sistem de raționare în logica de ordinul întâi

## Inferență în logica de ordinul întâi



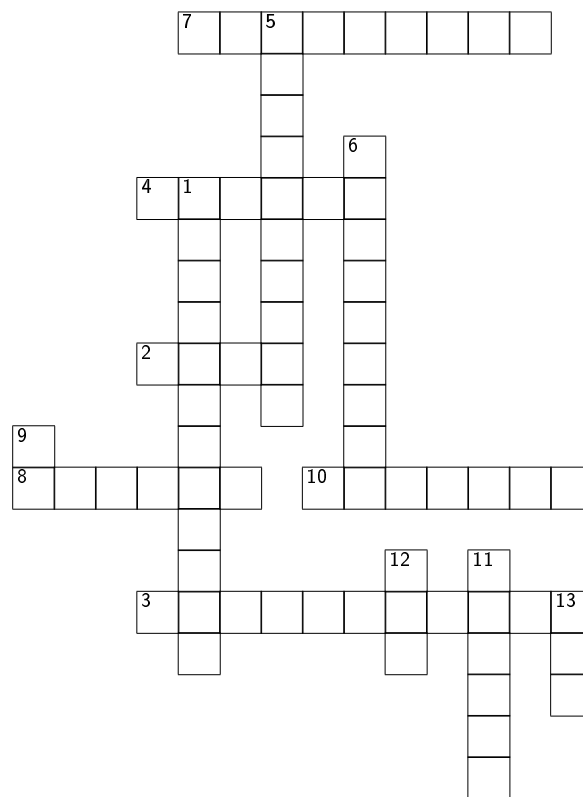
### Orizontal

- 2 Cel mai general unificator de clauze (en)
- 3 Structură cu relații de ordine parțială
- 5 Baze de date care permit inferență
- 7 Regulă de inferență în logica de ordinul întâi
- 8 Demonstrator de teoreme succesori al lui Otter
- 9 Modus ponens pentru logica de ordinul întâi
- 13 Baze de cunoștințe fără funcții
- 14 Limbaj de programare logică

### Vertical

- 1 Constante introduse pentru variabilele cuantificate existențial
- 4 Algoritm eficient de raționare cu reguli
- 6 Regulă de inferență în logica de ordinul întâi
- 10 Campion între demonstratoarele de teoreme
- 11 Colecție de teoreme (en)
- 12 Demonstrator de teoreme care poate proiecta circuite

## Planificare clasică



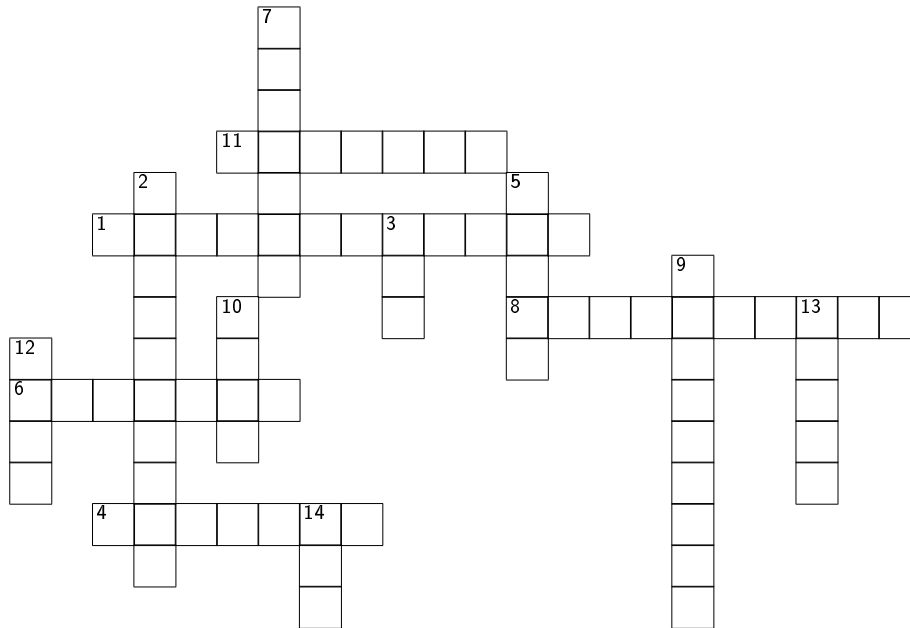
### Orizontal

- 2 Limbaj specializat pentru probleme de planificare
- 3 Prezente la acțiuni
- 4 Prezente la acțiuni
- 7 Algoritm de planificare
- 8 Relație care variază de la o stare la alta
- 10 Planificator care folosește satisfiabilitate

### Vertical

- 1 Planificator cu euristici în Python
- 5 Acțiune cu toate condițiile satisfăcute
- 6 Acțiune cu un efect care se regăsește în starea finală
- 9 Planificator pe repede-înainte
- 11 Limbaj de planificare inclus în PDDL
- 12 Diagrame de decizie pentru reprezentarea eficientă a planurilor
- 13 Competiție de planificare

## Planificare în lumea reală



### Orizontal

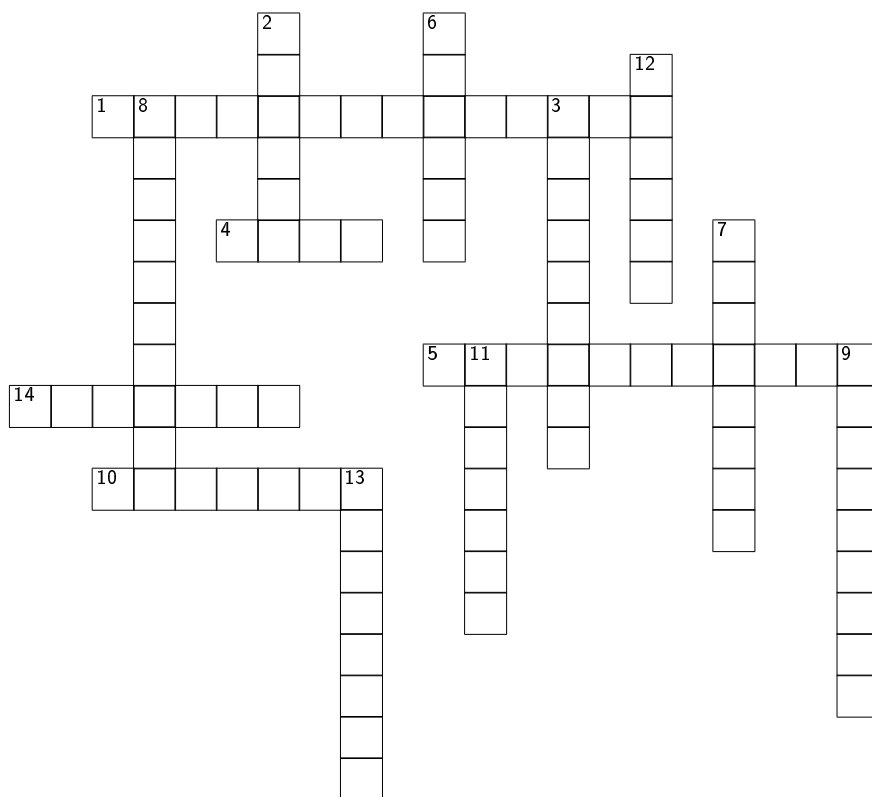
- 1 Efecte prefixate cu *when*
- 4 Cea mai lungă cale
- 6 Nondeterminism generat de un alt agent
- 8 Planificare fără senzori (en)
- 11 Consumate de acțiuni

### Vertical

- 2 Necesară în planificarea multi-agent
- 3 Asumpție în care starea conține atât fluenți negativi cât și pozitivi
- 5 *LS – ES* (en)
- 7 Nondeterminism generat de agentul curent
- 9 Mod de a selecta planul curent comun pentru mai mulți agenți
- 10 Agent pasăre (en)
- 12 Folosite de către planificatorul conformant HSCP
- 13 Conferință anuală pentru agenți
- 14 Determină posibilele momente de început și sfârșit ale unei acțiuni



## Reprezentarea cunoștințelor. Calculul evenimentelor



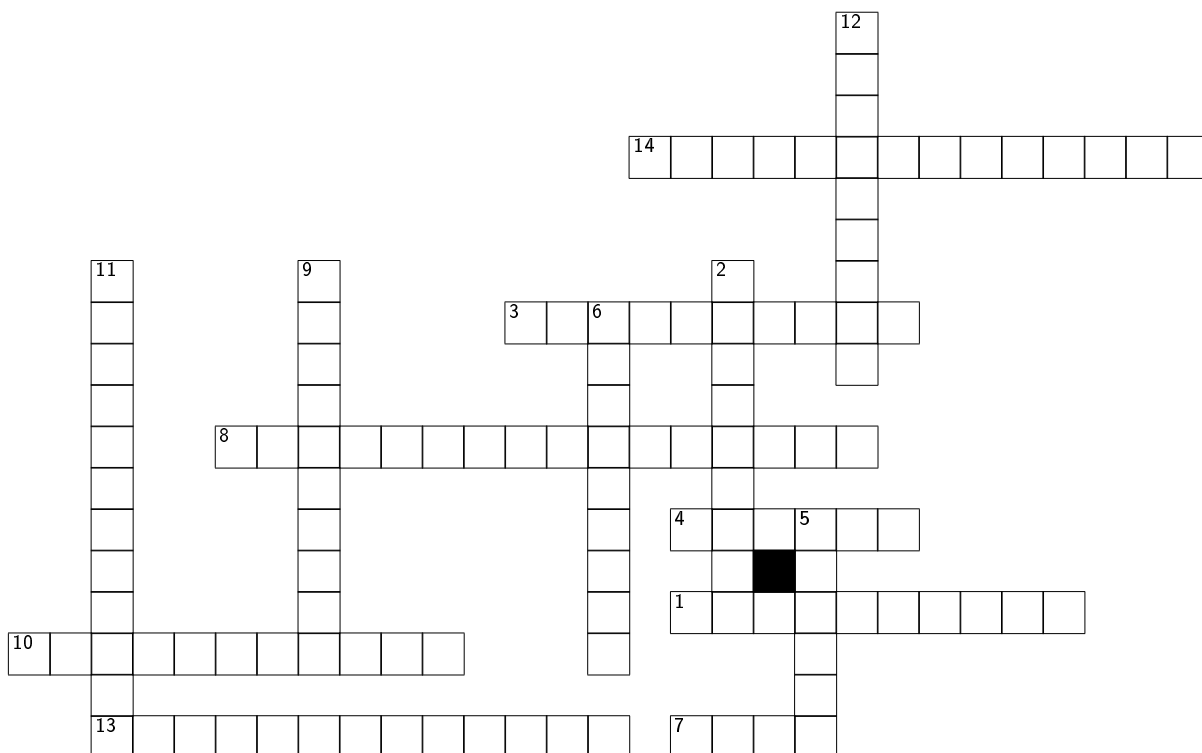
### Orizontal

- 1 Mai puternică decât prezumția de lume închisă
- 4 Sistem de menținere a adevărului bazat pe justificări
- 5 Logici în care o concluzie derivată poate fi retrasă
- 10 Evenimente care împărțite dau același eveniment
- 14 Ontologia wikipediei

### Vertical

- 2 Tipul lui  $f$  în  $T(f, t)$
- 3 Ontologie doar cu relații de tip *subclasă*
- 6 Predicat pe intervale temporale (en)
- 7 Decompoziție exhaustivă disjunctă
- 8 Proprietăți 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 consecințe dintr-o bază de cunoștințe

## Probabilități



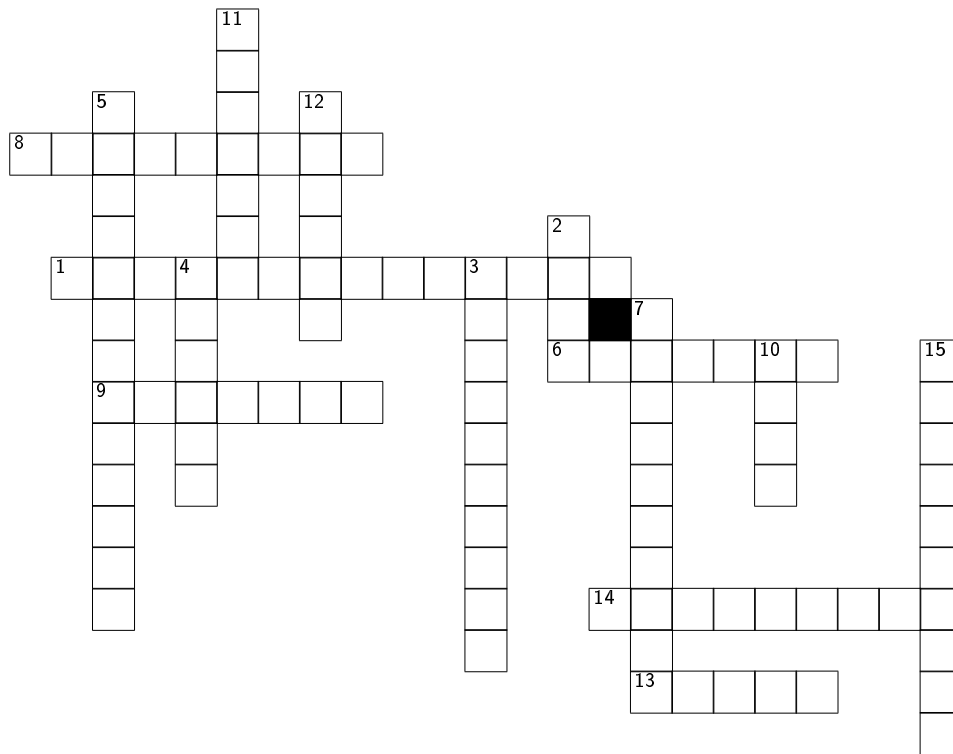
### Orizontal

- 1 Teoria bazată pe teoria probabilităților și a utilităților
- 3 Rezultate ponderate cu probabilitatea lor de apariție
- 4 Spațiu pentru toate lumile posibile (en)
- 7 Funcții densitate de probabilitate (en)
- 8 Teorie pentru grade de certitudine
- 10 Interpretare conform căreia incertitudinea provine din ignoranța condițiilor inițiale
- 13 Însurarea probabilităților pentru fiecare valoare posibilă a celorlalte variabile
- 14 Interpretare conform căreia probabilitățile provin doar din experimente

### Vertical

- 2 Calitatea de a fi util
- 5 Probabilități neconditionate (en)
- 6 Probabilități conditionate (en)
- 9 Matematician rus care a dezvoltat teoria probabilităților
- 11 Interpretare conform căreia probabilitățile caracterizează cunoștințele unui agent
- 12 Calcularea probabilităților posterioare pe baza observațiilor/evidențelor curente

## Raționare în rețele Bayesiene



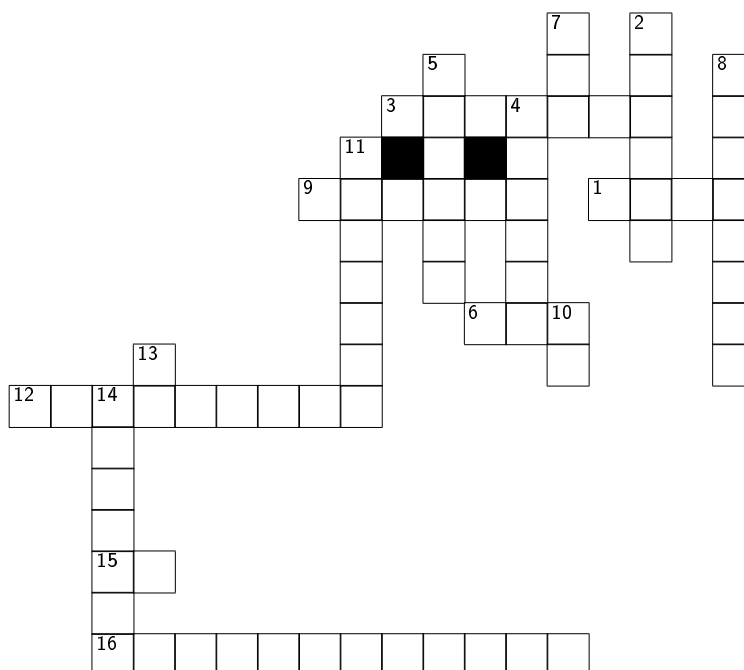
### Orizontal

- 1 Rețea Bayesiană
- 6 Rețea Bayesiană
- 8 Criteriu topologic pentru a decide dacă un set de noduri  $X$  este condițional independent de  $Y$  fiind dat  $Z$
- 9 Generalizare a funcției OR pentru relații incerte (en)
- 13 Distribuție de probabilitate (en)
- 14 Distribuție de probabilitate

### Vertical

- 2 Markov Chain Monte Carlo!
- 3 Semantică în care variabilele sunt condițional independente de nondescendenții lor, fiind dați părinții
- 4 Rețea Bayesiană (en)
- 5 Nod cu valoarea specificată exact de către unul dintre părinți
- 7 Dezvoltator al rețelelor Bayesiene și rezident în Los Angeles
- 10 Nod pentru tot felul de cauze (en)
- 11 Rețea cu variabile atât discrete, cât și continue
- 12 Distribuție de probabilitate (en)
- 15 Algoritmi de eșantionare aleatoare

## Rețele Bayesiene



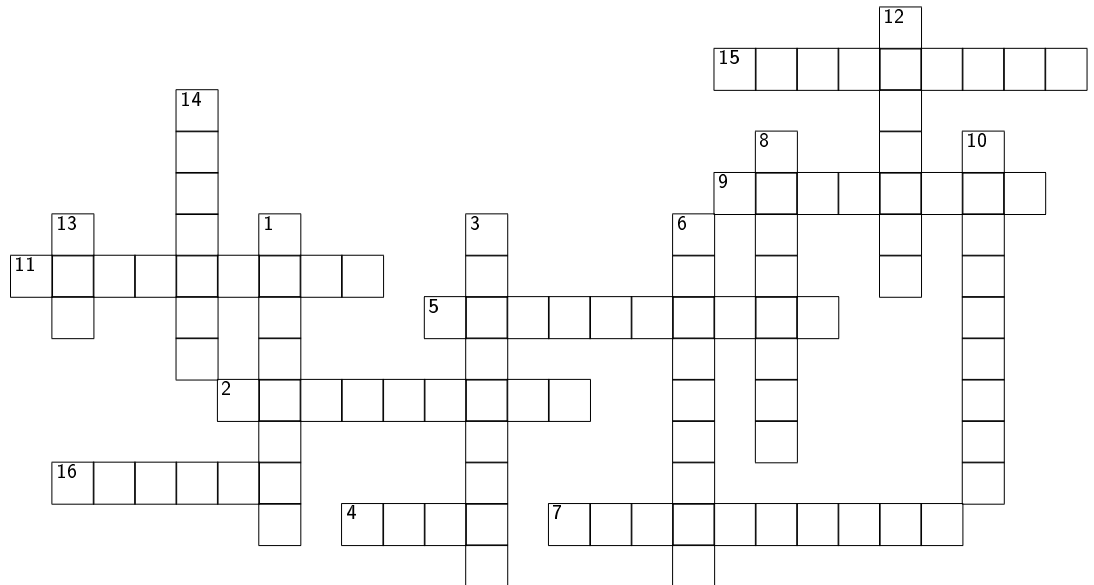
### Orizontal

- 1 Model Bayesian în care o singură cauză influențează efecte independente)
- 3 Autor al primei analize sistematice a probabilităților în anul 1565
- 6 Prezumția de nume unic (en)
- 9 Asumpție conform căreia starea curentă depinde de un număr finit și fix de stări anterioare
- 12 Algoritm care decide dacă două variabile sunt independente într-o rețea Bayesiană
- 15 Dempster-Shafer!
- 16 Marginalizare cu probabilități condiționate în locul probabilitatilor cuplate (joint)

### Vertical

- 2 Prenumele lui Bayes
- 4 Autorul cărții *Partida neterminată (The Unfinished Game)*, în care se discută celebra corespondență dintre Pascal și Fermat din 1654
- 5 Criteriu (pătură) pentru stabilirea independenței condiționale
- 7 Prezumție de lume închisă
- 8 Primul sistem expert bazat pe rețele Bayesiene
- 10 Statistician 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 și nu există cicluri stricte periodice

## Filtru Kalman



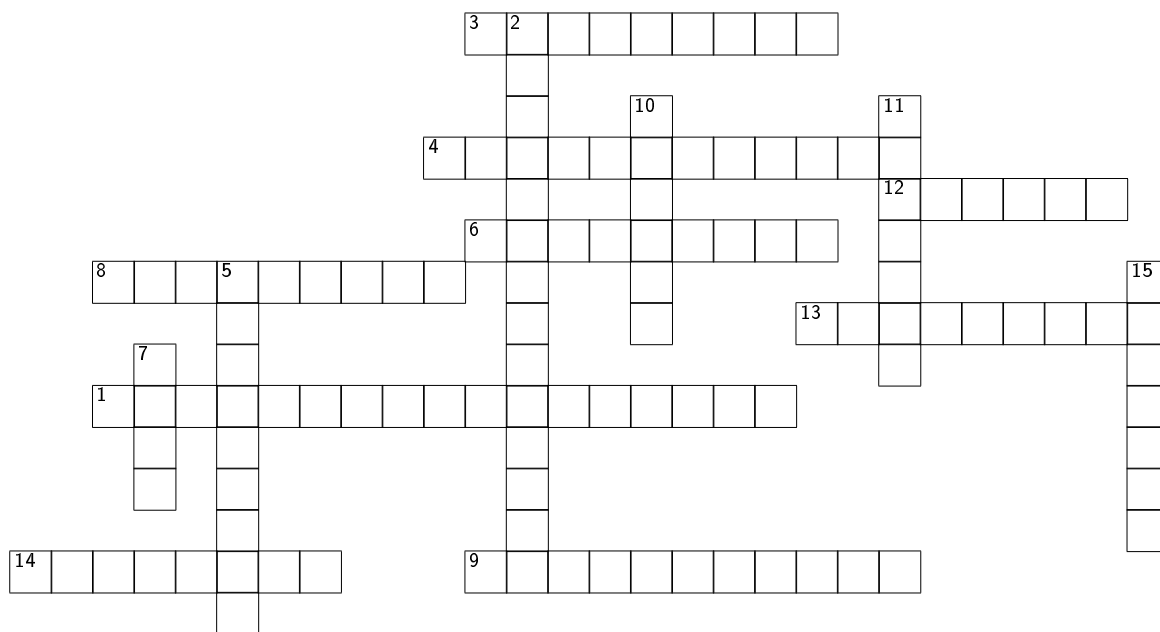
### Orizontal

- 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 distribuției posterioare pe starea curentă, fiind date evidențele până la momentul curent
- 11 Calculul distribuției posterioare pe o stare trecută, pe baza evidențelor până la momentul curent (en)
- 15 Proces în care schimbarea se supune unor legi care nu se schimbă în timp
- 16 Asumpția  $P(E_t|X_{0..t}, E_{0..t-1}) = P(E_t|X_t)$

### 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 distribuției posterioare pe o stare viitoare, fiind date evidențele până la momentul curent
- 12 Algoritm pentru identificarea celei mai probabile secvențe
- 13 Hidden Markov Model!
- 14 Alarmă falsă (en)

## Decizii simple



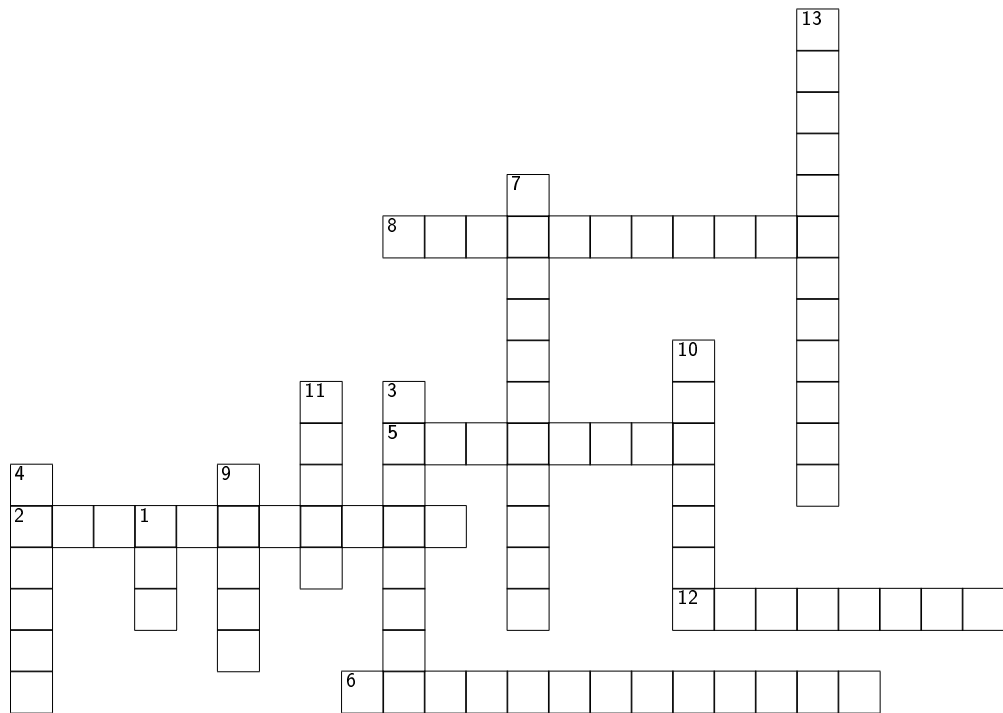
### Orizontal

- 1 Constrângere pe loterii
- 3 Funcție care modelează preferințele agenților
- 4 Constrângere pe loterii
- 6 Măsură utilizată în domeniul medical
- 8 Teorie care descrie cum trebuie un agent să acționeze
- 9 Teorie care descrie cum se comportă în realitate un agent
- 12 Ovale în rețele de decizie (en)
- 13 Romburi în rețele de decizie
- 14 Funcție acțiune-utilitate în învățarea reînforțată

### Vertical

- 2 Constrângere pe loterii
- 5 Constrângere pe loterii
- 7 Ultimul cuvânt din abrevierea QUALY (en)
- 10 Bias cognitiv
- 11 Rețea cu noduri de tip variabilă aleatoare, decizii și utilități
- 15 Dezvoltatorul calculului hedonic

## Învățare supervizată. Arbori de decizie



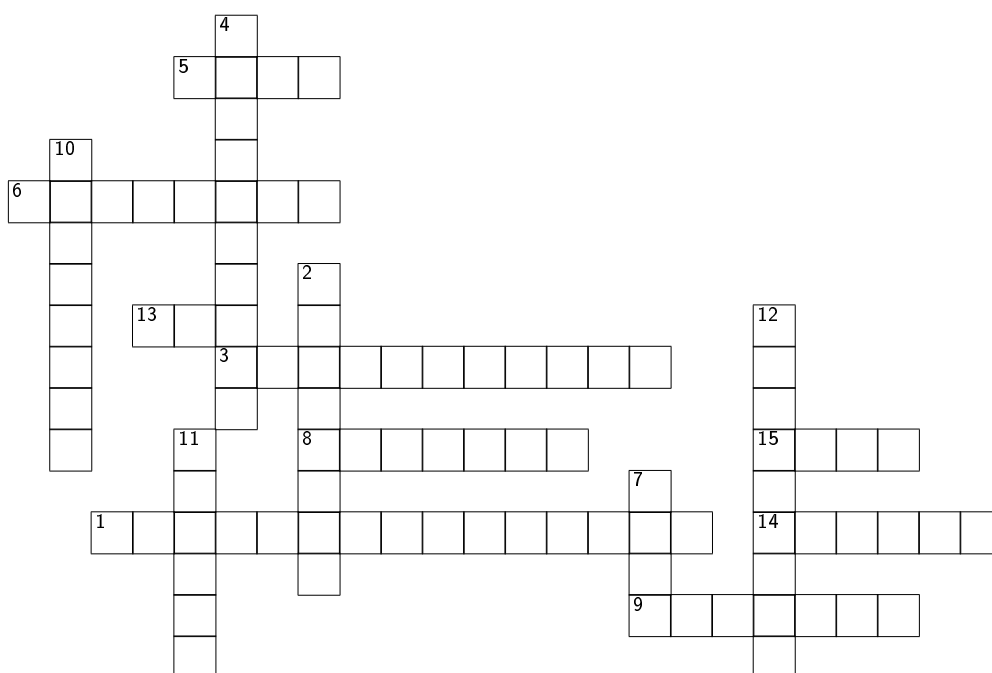
### Orizontal

- 2 Învățare a unui atribut având un set finit de valori
- 5 Măsură a importanței unui atribut în luarea deciziei
- 6 Tip de învățare
- 8 Tip de învățare
- 12 Vector de derivate parțiale

### Vertical

- 1 Algoritm de învățare supervizată
- 3 Învățare a unui atribut având un set continuu de valori
- 4 Filozof englez căruia îi plac lucrurile simple
- 7 Tip de învățare
- 9 Variantă de validare încrucișată
- 10 Când setul de test influențează parametrii învățării (en)
- 11 Validare încrucișată cu  $k = n$
- 13 Proces de penalizare explicită a ipotezelor complexe

## Rețele neurale



### Orizontal

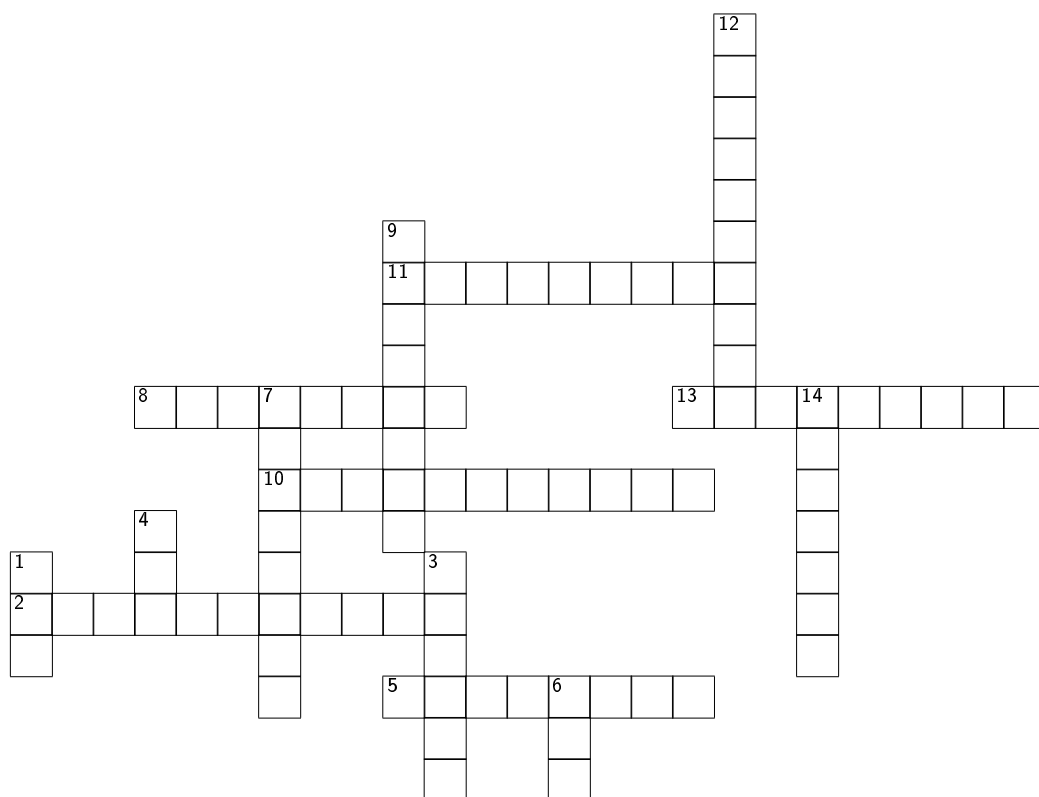
- 1 Algoritm de învățare dezvoltat de Werbos în 1975 (nn)
- 3 De evitat la învățare (en)
- 5 Rețea cu multe niveluri ascunse
- 6 Vector de derivate parțiale
- 8 Funcție de activare simplu de derivat
- 9 Funcție de activare
- 13 Funcție logică neagreată de perceptron
- 14 Algoritm de extindere a unei rețele inițiale (en)
- 15 Funcție liniară de activare a neuronului

### Vertical

- 2 Medie armonică între precizie și recall (en)
- 4 Rețea fără niveluri ascunse
- 7 Funcție de evaluare a erorii (en)
- 10 Metrică de evaluare
- 11 Metrică de evaluare
- 12 Metrică de evaluare



## Ansambluri de clasificatori



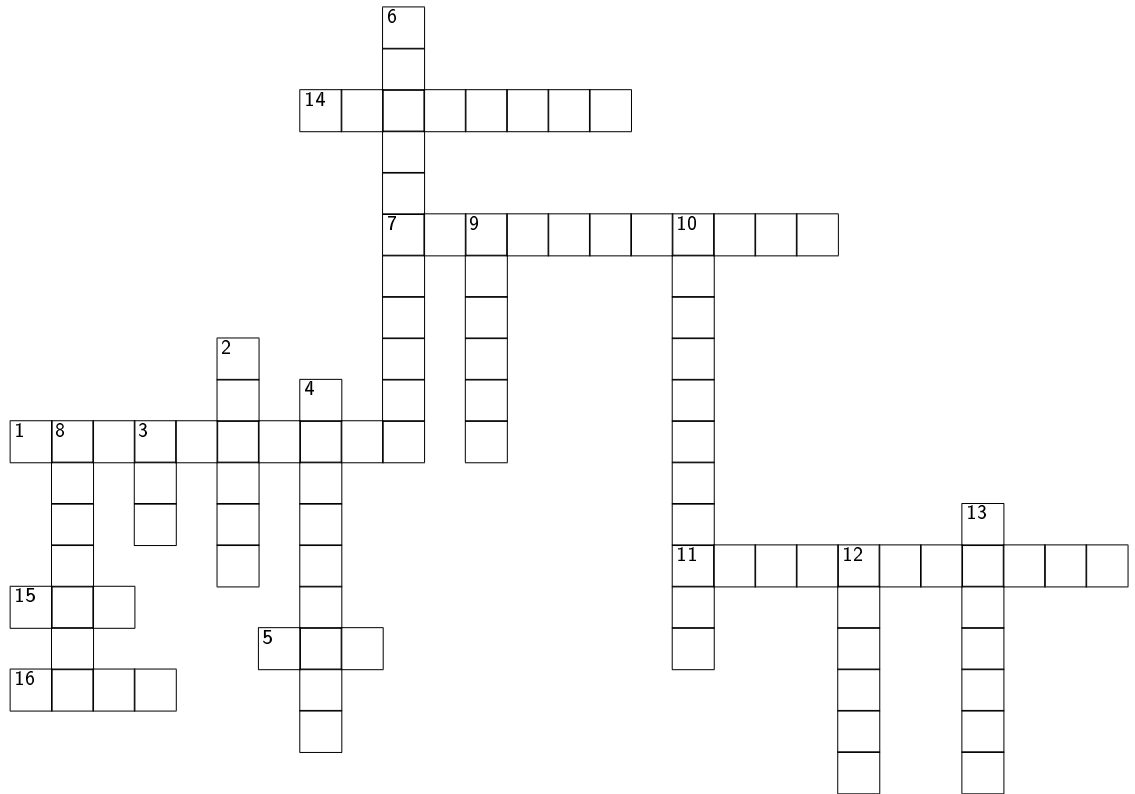
### Orizontal

- 2 Preprocesare necesară la KNN
- 5 Mulți clasificatori
- 8 Algoritm adaptiv de învățare cu ansambluri
- 10 De evitat la învățare (en)
- 11 Distanță între două instanțe
- 13 Distanță Minkowski cu  $p = 1$

### Vertical

- 1 Algoritm nonparametric
- 3 În centrul SVM
- 4 Algoritm nonparametric
- 6 Mulți agenți
- 7 Metodă în învățarea cu ansambluri de clasificatori
- 9 Medie armonică între precizie și recall (en)
- 12 Distanță Minkowski cu  $p = 2$
- 14 Distanță pe attribute booleene

## Învățare nesupervizată



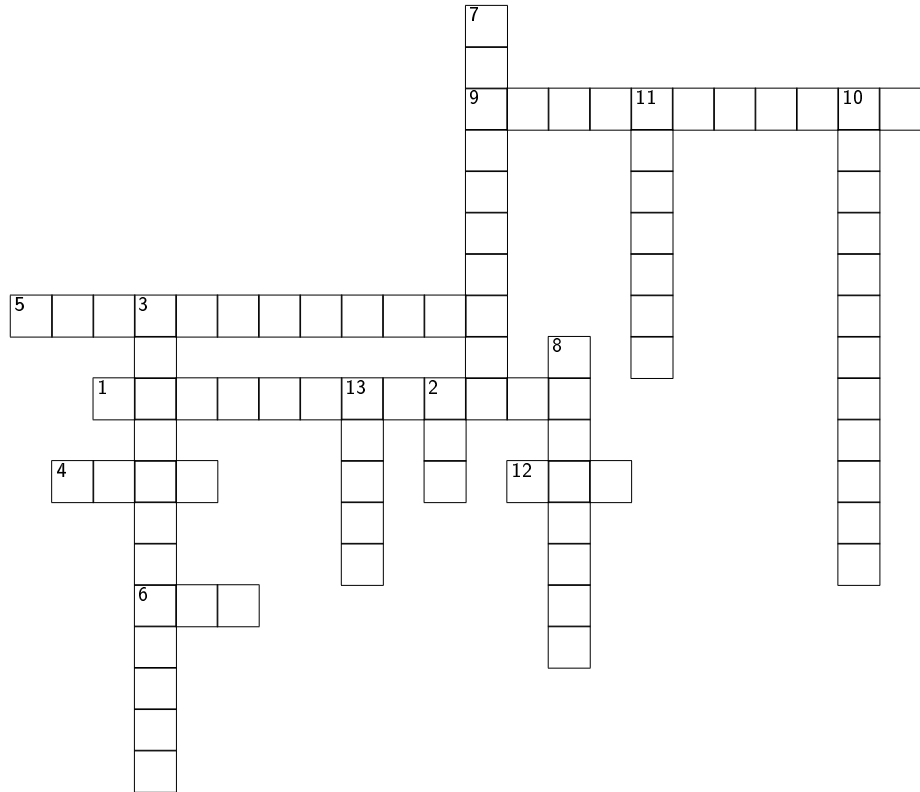
### Orizontal

- 1 Metodă de învățare nesupervizată
- 5 Metrică pentru distanța dintre clustere sensibilă la zgomote
- 7 Primul cuvânt din abrevierea i.i.d.
- 11 Tip de clusterizare ierarhică
- 14 Medie armonică între precizie și recall (en)
- 15 Metrică pentru distanța dintre clustere
- 16 Metrică pentru distanța dintre clustere bazată pe SSE

### Vertical

- 2 Algoritm de clusterizare de tip partiție
- 3 Metrică de evaluare a clusterizării
- 4 Variantă de K-means
- 6 De evitat la învățare (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

## Învățare bazată pe cunoștințe



### Orizontal

- 1 Modificare a ipotezei pentru a gestiona instanțe fals negative
- 4 Învățare inductivă bazată pe cunoștințe (en)
- 5 Algoritmi care generează noi predicate prin inducție
- 6 Algoritm dezvoltat de Quinlan pentru inducția regulilor
- 9 Salvarea unor rezultate intermediare de calcul (en)
- 12 Algoritm dezvoltat de Clark și Niblett pentru inducția regulilor

### Vertical

- 2 Primitivul din capitol
- 3 Modificare a ipotezei pentru a gestiona instanțe fals pozitive
- 7 Tip de învățare în care agenții învață mai bine pe măsură ce au mai multe cunoștințe
- 8 Brazilianul din capitol
- 10 Țeluri ușor de rezolvat
- 11 Rezoluție la programarea logică inductivă
- 13 Translator din FOL în logica propozițională

# Partea II

## Soluții și explicații

## Capitolul 3

### Întrebări cu răspuns multiplu

## 24. Introducere în inteligență artificială

(a) Când s-a născut inteligența 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 raționare logică

**Explicație:** Conferința de la Colegiul Dartmouth, USA a fost organizată de Marvin Minsky, John McCarthy, Claude Shannon și Nathan Rochester. Viziunea organizatorilor era ca orice aspect al inteligenței poate fi formalizat atât de precis încât o mașină îl poate simula. Termenul de *inteligență artificială* a venit la propunerea lui McCarthy.

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

- gândirii umane
- gândirii raționale
- comportamentului uman**
- comportamentului rațional

**Explicație:** Criteriul propus de Turing consideră un agent inteligent dacă acesta acționează similar unui agent uman. Criteriul nu impune ca similaritatea comportamentală să fie obținută prin mimarea modului de gândire a agentului uman.

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

- gândească uman
- gândească rațional
- acționeze uman
- acționeze rațional**

**Explicație:** Se dorește proiectarea unor agenți raționali care acționează pentru a obține cel mai bun rezultat.

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

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

**Explicație:** Deep Blue l-a învins pe Garry Kasparov în 1997, câștigând 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)**

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

- (f) Testul Turing total include (30s)
- învățare automată și viziune computerizată
  - viziune computerizată și robotică**
  - procesarea limbajului natural
  - înțelegerea limbajului și robotică

**Explicație:** Testul extinde versiunea clasică cu recunoașterea scenei și interacțiune fizică. Sunt necesare astfel viziunea computerizată și robotica.

- (g) Predicții legate de testul Turing (30s)
- până în 2000, 30% șanse ca cineva să fie indus în eroare pentru 5 minute**
  - până în 2000, 30% șanse ca un expert să fie indus în eroare
  - până în 2000, să fie păcălită orice persoană
  - 20% șanse să fie păcălită orice persoană

**Explicație:** Predicția 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**

**Explicație:** 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 supremația a fost

(30s)

- Șah
- Go**
- Spinner
- Game of Thrones

**Explicație:** Versiunile programului AlphaGo caută cea mai bună mutare folosind metode de tip Monte Carlo și modele învățate cu rețele neurale sau învățare re-inforșată.



## 25. Agenți

(a) Raționalitatea implică (30s)

- omnisciență, învățare, autonomie
- explorare, învățare, autonomie**
- culegere de informații, învățare, omnisciență
- succes, învățare, autonomie

**Explicație:** Omnisciența presupune că agenții pot deduce toate consecințele logice din bazelor lor de cunoștințe. Omnisciența este o proprietate prea puternică pentru agenții software. Similar, faptul că un agent ia decizii raționale 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**

**Explicație:** Agentul percepe *mediul* prin *senzori* și acționează asupra acestuia prin *actuatori*. Cât de bine evoluează agentul este măsurat de o funcție de performanță.

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

- mediul și funcția de performanță se schimbă
- mediul se schimbă, funcția de performanță nu
- mediul nu se schimbă, funcția de performanță se schimbă**
- mediul și funcția de performanță nu se schimbă

**Explicație:** Un exemplu de mediu semi-static este jocul de șah cu ceas. În timpul deliberării poziția de pe tablă nu se modifică. În schimb, funcția de evaluare întoarce valori mai mici odată cu trecerea timpului.

(d) În medii nondeterministe stărilor succesoare le sunt atașate probabilități (30s)

- Adevărat
- Fals**

**Explicație:** Sunt specificate toate stările succesoare *posibile* unei acțiuni nedeterministe. Acestor stări nu le sunt atașate probabilități.

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

- deterministic și continuu

- static și parțial observabil
- multi-agent și secvențial
- ✓ **nicio variantă din cele anterioare**

**Explicație:** Doar ultima variantă este corectă deoarece: (1) Stările succesoare sunt generate prin completarea unei litere. Mediul este așadar discret. (2) Definițiile cuvintelor și poziția 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 șah fără ceas este (30s)
- semi-static și complet observabil
  - ✓ **static și complet observabil**
  - continuu și secvențial
  - episodic și multi-agent

**Explicație:** Doar variantă (2) este corectă, astfel: (1) Deoarece nu există ceas, performanța agentului nu scade în timp ce acesta deliberează. (3) Pozițiile succesoare sunt generate cu fiecare mutare - mediul este discret. (4) Acțiunile curente ale agenților depind de mutările efectuate anterior - mediul este secvențial și nu episodic.

- (g) Mediul pentru șoferul de taxi este (30s)
- continuu și complet observabil
  - dinamic și complet observabil
  - ✓ **continuu și secvențial**
  - episodic și multi-agent

**Explicație:** Doar variantă (3) este corectă, deoarece: (1,2) Anumite variabile nu pot fi observate (e.g. numărul de mașini pe o anumită rută) - mediul este doar parțial observabil. (4) Acțiunea din starea curentă depinde de o acțiunile executate anterior - mediul este secvențial și nu episodic.

- (h) Un agent reflex bazat pe model menține o stare interioară (30s)
- ✓ **Adevărat**
  - Fals

**Explicație:** Modelul reprezintă cunoștințele agentului despre cum funcționează lumea sau mediul în care acționează.

- (i) Agenții bazați pe utilitate (30s)

- utilizează reguli de tip *conditie*  $\rightarrow$  *actiune*
- aleg acțiunea care îi conduce la cea mai bună utilitate expectată**
- folosesc planificarea pentru a atinge starea țintă

**Explicație:** Suplimentar unui model al lumii, se utilizează o funcție de utilitate care măsoara performanța agentului în fiecare stare. Pe baza funcției de utilitate, agentul alege cea mai bună acțiune ponderând utilitatea stării succesoare cu probabilitatea de a ajunge în respectiva stare.

- (j) Un agent care învață își poate îmbunătăți (30s)
- doar cunoștințele inițiale
  - doar funcția de evaluare a performanței
  - doar generatorul de probleme utilizat în învățare
  - toate elementele sale**

**Explicație:** Algoritmii de învățare se pot aplica pe orice componentă din arhitectura agentului.

## 26. Căutare neinformată

- (a) Dacă agentul nu știe care este starea lui inițială, problema de rezolvat este de tipul (30s)
- stare unică (single state)
  - conformant**
  - contingent

**Explicație:** În *conformant problems*, percepțiile agentului nu sunt suficiente pentru a decide starea în care se află. Totuși un astfel de agent poate executa acțiuni care îl duc într-o stare cunoscută. Pornind de la un set stări posibile în care se află, agentul execută acțiuni 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 soluției
  - numărul mediu de noduri în memorie
  - numărul total de noduri generate/expandate**
  - numărul maxim de noduri în memorie în timpul căutării

**Explicație:** Complexitatea în timp a unui algoritm este estimată prin numărul de operații efectuate. În cazul căutării operațiile se referă la crearea de noduri în arborele de căutare.

- (c) Un algoritm care găsește întotdeauna soluția dacă aceasta există este (30s)
- complet
  - optimal**

**Explicație:** Bineînțeles, completitudinea nu implică optimalitate.

- (d) Care este cea mai bună strategie de căutare pe un calculator cu memorie puțină? (30s)
- în adâncime**
  - în lățime
  - cost uniform

**Explicație:** Căutarea în lățime 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. Spațiul de memorare este așadar o problemă pentru căutarea în lățime. 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 găsește soluția. Așadar, spațiul este liniar raportat la factorul de ramificare  $b$ .

- (e) Căutarea în adâncime găsește cea mai bună soluție (30s)
- adevărat
  - fals

**Explicație:** Căutarea în adâncime nu este optimală. De exemplu, o soluție aflată la adâncimea 5 în arborele de căutare poate fi găsită înaintea unei soluții aflate la adâncimea 3.

- (f) A\* este un algoritm de tip greedy ghidat de funcția de estimare a costului  $f(n) =$  (30s)
- costul de la nodul  $n$  la țintă
  - costul de la starea inițială la nodul  $n$
  - maximul dintre costul de la starea inițială la  $n$  și costul de la  $n$  la țintă
  - nicio variantă din cele enumerate**

**Explicație:** La A\*,  $f(n) = g(n) + h(n)$ . Pe lângă costul  $g(n)$  de la starea inițială la nodul curent  $n$ , algoritmul A\* ține cont și de estimarea costului de la nodul curent la țintă. Această estimare este data de euristica  $h(n)$ .

- (g) Care nu este un algoritm de căutare informată? (20s)
- cost uniform**
  - greedy-best search
  - A\*

**Explicație:** 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 funcția  $g(n)$  nu este luată în calcul. Decizia depinde doar de valoarea estimării  $h(n)$ . Atât A\*, cât și *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 poziția finală
  - $h_2 =$  suma distanțelor de la fiecare piesă la poziția ei finală
  - $h_3 = \max(h_1, h_2)$
  - $h_4 = \text{sum}(h_1, h_2)$

**Explicație:**  $h_1$  și  $h_2$  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 depăși 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**

**Explicație:** Un algoritm complet garantează găsirea soluției dacă aceasta există. Algoritmul hill-climbing poate rămâne blocat într-un maxim local (i.e. un agent nu poate ieși 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 selecție poate depinde de diferența de utilitatea între starea succesoare și 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 soluției. Varianta hill climbing cu reporniri aleatoare rulează algoritmul pornind din mai multe stări inițiale aleatoare. Cu un număr suficient de mare de rulări, probabilitatea de a găsi soluția 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**

**Explicație:** 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 ramificație (branching factor) așa 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,  $k > 1$

**Explicație:** Algoritmul alege o stare succesoare  $s$  aleator. Dacă în urma evaluării se dovedește 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 totuși selectată cu o anumită probabilitate. Probabilitatea depinde de doi factori: 1) de diferența de valoare dintre starea curentă  $c$  și starea selectată  $s$  ( $\Delta_E = |E_s - E_c|$ ) precum și 2) de temperatura sistemului, rezultând

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

Inițial temperatura este mai mare. Rezultă o probabilitate mai mare de selecție 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* pornește cu (20s)

- temperatură ridicată**  
 temperatură joasă

**Explicație:** Temperatura inițială mare permite explorarea spațiului de căutare. Pe măsură ce căutarea avansează, probabilitatea de selectare a unei stări mai slabe scade. Când temperatura 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

**Explicație:** Probabilitatea depinde de doi factori: 1) de diferența de valoare dintre starea curentă  $c$  și starea selectată  $s$  ( $\Delta E = |E_s - E_c|$ ), precum și 2) de temperatura sistemului.

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

- Adevărat  
 **Fals**

**Explicație:** În cazul *local beam*, căutarea începe cu  $k$  stări generate aleator. La fiecare pas sunt generați toți succesorii celor  $k$  stări. Din această mulțime se selectează cei mai buni  $k$  succesori. Astfel, o stare din cele  $k$  poate contribui cu mai mulți succesori în generația următoare. Acest lucru nu apare în cazul rulării în paralel a  $k$  algoritmi hill climbing.

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

- ADN**  
 mutație  
 încrucișare

clonare

**Explicație:** Pentru a construi generația următoare, algoritmi genetici aplică următorii operatori pe indivizii din generația curentă: 1) mutație: gene din cromozom sunt modificate aleator; 2) încrucișare: secvențe de gene de la un individ sunt încrucișate cu secvențe de gene de la alt individ; 3) clonare: indivizii foarte buni pot fi copiați direct în generația următoare.

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

**alegerile mediului**

alegerea făcută de agent

**Explicație:** Arborii de căutare de tip AND-OR conțin două tipuri de noduri. Nodurile de tip OR reprezintă alegerile agentului. Agentul decide asupra unei singure acțiuni 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**

spațiul de căutare al unui agent religios

cea mai probabilă stare în care agentul poate fi

spațiul de căutare al unui agent ateist

**Explicație:** Mulțimea include toate stările posibile. Acestor stări nu le sunt atașate probabilități.



## 28. Căutare adversarială

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

în adâncime

în lățime

cost uniform

A\*

**Explicație:** 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 și așa mai departe.

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

fals

adevărat

**Explicație:** O bună ordonare a mutărilor ajută la oprirea mai rapidă a căutării în zonele care funcția 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

**Explicație:** Cu *alpha-beta pruning* se obține același rezultat, dar în urma unei căutari mai reduse.

- (d) În practică, pentru evaluarea poziției se utilizează o combinație de atribute (20s)

liniară

nonliniară

**Explicație:** Combinația liniară presupune că atributele considerate în evaluarea poziției sunt independente. De exemplu, la șah, valori posibile pentru cal, nebun și damă sunt 3, 3, respectiv 10 pioni. Dar combinația cal-damă este mai puternică decât nebun-damă, ceea ce nu poate fi modelat printr-o combinație liniară.

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

fals

adevărat

**Explicație:** 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. Funcția de evaluare aplicată pe această poziție ar fi favorabilă albului. Poziția nu este una liniștită, deoarece este puțin probabil ca negrul să nu aibă un răspuns simetric prin care să captureze o piesă de valoare similară. O astfel de poziție e recomandat să fie analizată la o adâncime mai mare. Căutarea de tip *quiesscent* aplică funcția de evaluare doar pe poziții liniștite. Pozițiile care nu sunt liniștite 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)

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

- (g) Valoarea expectată a unei poziții în jocuri stocastice este (30s)
- media tuturor rezultatelor posibile ale nodurilor de tip *min* și *max*
  - suma tuturor rezultatelor posibile ale nodurilor de tip *frunză*
  - media tuturor rezultatelor posibile ale nodurilor de tip șansă**
  - suma tuturor rezultatelor posibile ale nodurilor de tip *șansă*

**Explicație:** Pentru calcularea valorii expectate se înmulțește câștigul posibil cu probabilitatea de obținere a acestuia. Acest calcul se efectuează pentru fiecare instanțiere posibilă a nodurilor șansă.

- (h) În jocuri stocastice, comportamentul se prezervă pentru orice transformare a funcției de evaluare care este (30s)
- monotonă
  - nonmonotonă
  - liniar pozitivă**
  - liniar negativă

**Explicație:** Monotonia nu este suficientă pentru a păstra ordinea între mutări. Fie o transformare monotonă pentru două poziții succesive de la evaluarea inițială de 2 și 3 la 20 respectiv 300. Probabilitatea de atingere a primei poziții este 0.9, iar a celei de a doua poziții 0.1. Valoarea expectată a poziției 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 poziție  $s_2$  evaluată la 5 înainte de transformare și la 50 după transformare. Înainte de transformare  $s_2$  pare mai bună. După transformare  $s_1$  este mai bună. Dacă transformarea ar fi monoton liniară, ierarhia pozițiilor se păstrează.

(i) În jocuri stocastice, căutarea la adâncimi mari este esențială

(30s)

adevărat

fals

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

## 29. Probleme de satisfacere a constrângerilor

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

- nu poate fi îmbunătățit!
- sortarea variabilelor în ordine lexicografică
- detectarea timpurie a eșecului inevitabil**
- exploatând structura problemei**

**Explicație:** Prin detectarea timpurie a eșecului se elimină instanțierea unor variabile cu valori pentru care nu există soluție. 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 instanțiază la pasul următor**
- valorii din domeniu care se asignează variabilei curente**

**Explicație:** Atât domeniul variabilelor cât și constrângerile sunt date de intrare în problema, deci nu pot fi modificate. Ordinea în care se analizează variabilele, respectiv ordinea în care se asignează valori din domeniu variabilei selectate influențează semnificativ performanța căutării.

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

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

**Explicație:** Sunt două tipuri de euristici în acest context: euristici care selectează variabila, respectiv euristici care selectează valori din domeniul unei variabile. Euristică *valorile minime rămase* ajută la selectarea variabilei curente. Euristică are rolul de a selecta variabila cu domeniul de valori posibile cel mai mic la momentul curent. Se urmărește maximizarea șanselor de a putea asigna valori variabilelor rămase neinstanțiate. Variabilele învecinate unei variabile  $X$  se referă la variabilele cu care  $X$  are relații de constrângere. Modificarea domeniului variabilei  $X$  se propagă domeniilor variabilelor învecinate.

(d) Euristică grad selectează (30s)

- variabilele cu cele mai multe constrângeri pe variabile rămase**
- variabila cu cele mai puține valori disponibile

- valoarea care elimină cele mai puține valori ale variabilelor învecinate
- valorile care introduc cele mai multe constrângeri în variabilele învecinate

**Explicație:** Euristică *grad* ajută la selectarea variabilei curente. Prin selectarea variabilei cu cele mai multe constrângeri se urmărește identificarea eșecului inevitabil cât mai curând posibil.

(e) Euristică *cele mai puține constrângeri* selectează (30s)

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

**Explicație:** Euristică urmărește selectarea unei valori din domeniul variabilei deja selectate. Deci (1) și (2) nu se aplică. Prin selectarea valorii care afectează cel mai puțin vecinii se dorește maximizarea șanselor de a putea asigna valori variabilelor rămase neinstantiate.

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

- valorile minime rămase**
- grad
- cele mai puține constrângeri

**Explicație:** Eșecul apare în momentul în care domeniul unei variabile este nul. Euristică *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 puține constrângeri

**Explicație:** Euristică *grad* alege variabila cu cele mai multe constrângeri. La instanțierea acesteia, constrângerile rezultate se propagă în cele mai multe variabile învecinate. Prin reducerea domeniului a cât mai multor variabile se micșorează 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

**Explicație:** Verificarea înainte reduce domeniul variabilelor neinstantiate prin eliminarea valorilor pentru care nu există soluție. Pentru respectivele valori, algoritmul de backtracking ar identifica eșecul doar după instanțierea variabilelor.

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

**Explicație:**  $Y < Z$  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

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

### 30. Agenți logici

- (a) Lumea lui Wumpus este (30s)
- complet observabilă și deterministă
  - episodică și observabilă local
  - statică și include un singur agent**
  - multi-agent și secvențială

**Explicație:** (1) Greșită deoarece agentul are acces la informații doar din căsuțele învecinate. Deci mediul nu este complet observabil. (2) Greșită deoarece acțiunea curentă a agentului depinde de acțiunile 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 își adaptează comportamentul în funcție de acțiunile personajului. (4) Greșită deoarece sistemul include un singur agent.

- (b) Care expresie nu este satisfiabilă? (30s)
- $\neg a \vee b$
  - $a \rightarrow \neg a$
  - $a \vee \neg a \rightarrow a \wedge \neg a$
  - $a \wedge \neg a \vee b$

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

- (c)  $KB \models \alpha$  dacă și numai dacă (20s)
- $M(KB) \subseteq M(\alpha)$
  - $M(\alpha) \subseteq M(KB)$

**Explicație:** Pentru a deduce propoziția  $\alpha$  din baza de cunoștințe  $KB$  ( $KB \models \alpha$ ) mulțimea modelelor bazei de cunoștințe trebuie să fie inclusă în mulțimea modelelor în care  $\alpha$  este adevărată.

- (d) Propoziția  $p \rightarrow q \equiv \neg q \rightarrow \neg p$  este (30s)
- falsă
  - adevărată**
  - depinde doar de  $p$
  - depinde doar de  $q$

**Explicație:** Prin eliminarea implicației obținem în partea stângă  $\neg p \vee \neg q$ , iar în partea dreaptă  $\neg q \vee \neg p$ . Cele două disjunții sunt echivalente  $\forall p, q$ .

(e) Contrapozitia spune că: (30s)

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

**Explicație:** Variantele prezentate reprezintă următoarele axiome: (1) Contrapozitie; (2) Dublă negație; (3, 4) de Morgan.

(f) Tabelul de adevăr al implicației conține (10s)

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

**Explicație:** Implicația este falsă doar pentru  $1 \rightarrow 0$ .

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

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

**Explicație:** O propoziție este tautologie dacă are valoarea 1 indiferent de instanțierile posibile ale variabilelor. (1) are valoarea 0 pentru  $a = 0$ ; (2) tautologie; (3) prin eliminarea implicației se obține  $\neg a \vee a$ . (4) are valoarea 0 pentru  $a = 1$ .

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

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

**Explicație:** O clauză Horn este o disjuncție de literali cu maxim un literar pozitiv. Prin eliminarea implicației se obține: (1)  $\neg a \vee b$ . Doar  $b$  pozitiv, deci este clauză Horn. (2)  $\neg a \vee \neg b$  este o clauza Horn. Totuși, o astfel de expresie nu poate apărea în Prolog. Prologul conține 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 disjuncție. 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 implicației formula inițială poate fi transformă în două clauze Horn:  $(a \rightarrow b)$  și  $(a \rightarrow c)$ . (4)  $\neg a \vee b \vee c$  conține 2 literari pozitivi. Astfel de formule pot fi modelate în programarea logică disjunctivă.

- (i) Raționarea înainte este (30s)
- ghidată de țel
  - ghidată de date**

**Explicație:** Raționarea înainte pornește de la faptele existente pentru a identifica care reguli se pot activa.

- (j) Care expresie este în forma normal conjunctivă? (10s)
- $a \rightarrow b$
  - $a \wedge b$
  - $\neg a \vee b$
  - $\neg(a \wedge b)$

**Explicație:** Forma normal conjunctivă (CNF) este o conjuncție de disjuncții. În astfel de expresii nu apar implicații, respectiv negația apare doar pe literari atomici. (1) Nu este în CNF deoarece apare implicația; (2) Este în CNF; (3) Este o disjuncție în loc de conjuncție; (4) Negația apare pe expresii și nu pe atomi.

- (k) Pentru a demonstra  $\alpha$ , rezoluția arată că  $KB \wedge \neg\alpha$  este (10s)
- nesatisfiabilă**
  - satisfiabilă

**Explicație:** Rezoluția folosește metoda reducerii la absurd. Pentru a demonstra o propoziție, rezoluția arată că propoziția negată introduce o inconsistență în baza de cunoștințe. Altfel spus, baza de cunoștințe împreună cu propoziția 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)

**Explicație:** 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

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

- (c) Logica de ordinul include (20s)
- obiecte, relații, funcții**
  - timp, credințe, cunoștințe
  - constrângeri
  - aceeași expresivitate cu logica propozițională

**Explicație:** De exemplu, logica propozițională nu include variabile sau cuantificatori.

- (d) *Fiecăruia îi place înghețata* este formalizată în FOL (20s)
- $place(x, inghetata)$
  - $place \wedge inghetata$
  - $\exists x, place(x, inghetata)$
  - $\forall x, place(x, inghetata)$

**Explicație:** Prima variantă este greșită deoarece în FOL toate variabile trebuie cuantificate. A doua variantă reprezintă o formalizare în logica propozițională, logică care nu include variabile. În a treia varianta variabila  $x$  este cuantifică existențial și nu universal ca în varianta corectă.

- (e) *Există o persoană care iubeste pe toată lumea* este formalizată în FOL (20s)
- $\forall x \exists y, iubeste(x, y)$
  - $\exists x \forall y, iubeste(x, y)$

- $\forall x \text{ iubeste}(x, \text{Persoana})$
- $\exists x \text{ iubeste}(x, \text{ToataLumea})$

**Explicație:** Prima variantă este formalizarea propoziției: *Toată lumea iubeste pe cineva*. Variantele 3 și 4 includ constante: fiecare constantă referă un singur obiect.

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

**Explicație:** Greșeala din prima variantă provine de la utilizarea cuatificatorului existențial în locul celui universal. Varianta a doua folosește conjuncția în locul implicației. În varianta a treia, variabile  $x$  și  $y$  sunt libere - nu se va realiza nicio potrivire între acestea.

- (g) *Oricine de la UTCN este isteț* este formalizată în FOL (20s)
- $\forall x \text{ la}(x, \text{UTCN}) \wedge \text{istet}(x)$
  - $\text{istet}(x, \text{TUCN})$
  - $\forall x \text{ la}(x, \text{UTCN}) \Rightarrow \text{istet}(x)$
  - $\exists x \text{ la}(x, \text{UTCN}) \Rightarrow \text{istet}(x)$

**Explicație:** În multe cazuri, variabile cuantificate universal sunt folosite în implicații, iar cele cuantificate existențial în conjuncții.

- (h) Relația *sibling* este simetrică (20s)
- $\text{Sibling}(x)$
  - $\forall x, y \text{ Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x)$
  - $\text{Sibling}(x, y) \vee \text{Sibling}(y, x)$
  - $\exists x, y \text{ Sibling}(x, y) \leftrightarrow \text{Sibling}(y, x)$

**Explicație:** În (1) și (3) variabilele nu sunt cuantificate. În (4) cuantificatorul existențial este folosit în locul celui corect ( $\forall$ ).

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

$$\bigcirc \forall x, y \text{ Mother}(x, y) \wedge (\text{Female}(x) \wedge \text{Parent}(x, y))$$

**Explicație:** Definiția trebuie să fie validă pentru toate mamele, fiind necesară utilizarea cuantificatorului universal.

(j) *Oricine iubește pe cineva* este formalizată în FOL

(20s)

$\forall x \exists y, \text{iubeste}(x, y)$

$\exists x \forall y, \text{iubeste}(x, y)$

$\forall x \exists y, \text{iubeste}(y, x)$

$\exists x \forall y, \text{iubeste}(y, x)$

**Explicație:** (2) este formalizarea pentru *Cineva iubește pe toată lumea*. (3) este formalizarea propoziției *Pentru orice persoană există cineva care să iubească acea persoană*. (4) este formalizarea pentru *Există o persoană iubită de toată lumea*. Rețineți că ordinea cuantificatorilor este importantă.

### 32. Inferență în logica de ordinul întâi

- (a) Un termen *ground* este (20s)
- un termen cu cel puțin o constantă
  - un termen fără variabile**
  - un sol unde un arbore binar poate fi plantat
  - niciuna dintre aceste opțiuni

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

- (b) *Lifting* înseamnă (20s)
- transformarea din logică propozițională în logica de ordinul întâi**
  - eliminarea constantelor Skolem
  - introducerea constantelor Skolem

**Explicație:** Algoritmii de raționare în logica propozițională (înlanțuire înainte, înlanțuire înapoi și rezoluție) au variante urcate (*lifted*) în logica de ordinul întâi. Modus ponens generalizat (GMP) este modus ponens (MP) în care s-au introdus variabile:  $GMP = lifted MP$ . Operația inversă de transformare din logica de ordinul întâi în logica propozițională se realizează prin instanțierea cuantificatorilor universal (UI) și existențial (EI). Operația de eliminare a cuantificatorului existențial se numește skolemizare. Variabilele cuantificate existențial sunt înlocuite cu constante Skolem sau funcții Skolem.

- (c) Datalog = (10s)
- Prolog + funcții
  - baze de date extinse cu reguli
  - clauze în logica de ordinul întâi plus funcții
  - clauze în logica de ordinul întâi fără funcții**

**Explicație:** În Datalog sunt permise clauzele define dar fără funcții. Algoritmii de inferență este complet.

- (d) O bază de date deductivă (20s)
- utilizează inferența pentru a răspunde la interogări**
  - combină programarea logică cu baze de date relaționale**
  - mai expresivă decât o bază de date, dar mai puțin decât un limbaj logic**

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

- (e) Skolemizarea este procesul de (10s)
- eliminare a cuantificatorilor universali
  - eliminare a cuantificatorilor existențiali**
  - introducere a cuantificatorilor universali
  - introducere a cuantificatorilor existențiali

**Explicație:** În urma skolemizării, variabile cuantificate existențial sunt înlocuite cu constante Skolem sau funcții Skolem, iar cuantificatorul existențial 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)**

**Explicație:** Dezvoltarea logicii de ordinul întâi s-a datorat introducerii de către Frege a cuantificatorilor universal și existențial.

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

**Explicație:** Prologul utilizează clauze Horn definite (i.e. exact un literal pozitiv). Faptele negative nu se încadrează în aceasta definiție. În aceeași linie, concluziile clauzelor nu pot fi negate.

- (h) În forma normal conjunctivă, fiecare clauză conținută este o (20s)
- disjuncție de literari**
  - conjuncție de literari

**Explicație:** Forma normal conjunctivă conține clauze legate prin operatorul AND ( $\wedge$ ). În schimb, aceste clauze sunt formate din literari legați doar prin disjuncții ( $\vee$ ).

- (i) Care nu este o metodă de inferență logică? (10s)
- raționarea înainte
  - raționarea înapoi
  - skolemizarea**

○ rezoluția

**Explicație:** Skolemizarea reprezintă etapa de eliminare a cuantificatorilor existențiali. Etapa apare în momentul conversiei unei formule la forma normal conjunctivă. Această formă este necesară în mecanismul de rezoluție. Alături de rezoluție, raționarea înainte și înapoi sunt metode prin care se pot demonstra concluzii.

(j) Utilizări practice ale demonstratoarelor de teoreme includ

(20s)

✓ **verificare hardware**

✓ **verificare software**

**Explicație:** Pe de o parte, specificațiile hardware sau software sunt translatate ca teoreme. Pe de altă parte, modelul hardware sau software este formalizat și interpretat ca axiome. Demonstrarea teoremei pe baza axiomelor reprezintă verificarea ca sistemul respectă specificațiile.

### 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

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

- (b) O condiție deschisă este o precondiție (10s)
- cu un termen care nu are toate variabilele instanțiate
  - cu prezumția de lume deschisă
  - a unui pas fără legături cauzale**

**Explicație:** Algoritmii de planificare cu relații de ordine parțială mențin un set de precondiții *deschise* conținând acele precondiții care nu sunt îndeplinite de nicio acțiune la momentul curent.

- (c) Un *clobber* (20s)
- validează o precondiție obținută datorită unui efect al unei acțiuni
  - distruge o precondiție obținută printr-o legătură cauzală**
  - introduce o relație de ordine parțială între acțiuni

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

- (d) Anomalia Sussman (20s)
- ilustrează dezavantajele planificării greedy**
  - oprește algoritmul de planificare, chiar dacă un plan există.
  - poate fi rezolvată cu ajutorul *clobberilor***

**Explicație:** Fiind dat un țel  $g$  acesta poate fi împărțit în două subobiective  $g_1$  și  $g_2$ . Presupunem că în timpul căutării s-a identificat un plan pentru  $g_1$ . Anomalia Sussman apare când nu se poate avansa spre obținerea țelului global  $g$  fără a șterge planul obținut pentru  $g'$ .

- (e) O acțiune este relevantă dacă (20s)



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

**Explicație:** Acțiunile *relevante* sunt cele care au ca efect cel puțin unul din subțelurile obiectivului din starea finală. Acțiunile care au toate condițiile îndeplinite se numesc *aplicabile*.

- (f) Care sunt euristici admisibile pentru planificare? (20s)
- $h_1 =$  ignorarea condițiilor
  - $h_2 =$  ignorarea stării inițiale
  - $h_3 =$  ignorarea efectelor care elimină fapte din starea curentă
  - $h_4 =$  ignorarea stării finale

**Explicație:**  $h_1$  și  $h_3$  lucrează pe o simplificare a problemei inițiale. 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 condițiile

**Explicație:** În jocul culisant distanța Manhattan reprezintă suma distanțelor de la fiecare piesă la poziția ei finală. Distanța Manhattan reprezintă problema relaxată în care nicio altă piesă nu incomodează traseul de la poziția curentă la cea finală. Singura condiție necesară pentru efectuarea mutării este ca locațiile 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 greșit 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 condițiile

**Explicație:** În jocul culisant euristica  $h =$  numărul de piese plasate greșit cuantifică numărul de piese care nu se află încă pe pozițiile specificate în țel. Euristica  $h$  reprezintă problema relaxată în care o piesă poate fi scoasă de pe tablă și

așezată direct pe poziția finală. Pentru a se executa o astfel de acțiune nu sunt necesare condițiile ca locația succesoră să fie neocupată de o altă piesă (i.e. *blank(s2)*) și nici adiacentă cu poziția curentă (i.e. *adj(s1, s2)*).

### 34. Planificare în lumea reală

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

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

**Explicație:** Aici  $N$  reprezintă numărul de acțiuni, iar  $b$  este factorul maxim de ramificare al acțiunilor (indiferent dacă reprezintă intrare sau ieșire). Operațiile  $LS$  și  $ES$  sunt efectuate o singură dată pentru fiecare acțiune. Fiecare astfel de calcul este iterat pentru fiecare intrare/ieșire din acțiunea curentă.

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

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

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

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

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

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

(d) Euristică *minimum slack* (20s)

- este similară heuristicii numărul de valori minime specifică CSP
- garantează obținerea soluției optime
- nicio variantă din cele de mai sus

**Explicație:** Euristică *marja minimă* (*minimum slack*) alege acțiunea care poate fi începută cel mai drep vreme dintre toate acțiunile cu condițiile îndeplinite și care are cea mai mică marjă temporală. Euristică nu garantează soluția optimă. Similar cu euristică *numărul de valori minime* (MRV) din problemele de satisfacere a constrângerilor (CSP) este selectată variabila (în cazul nostru acțiunea) care are domeniul cu cele mai puține valori (în cazul nostru marja cea mai mică de alegere a momentului de început a acțiunii).

- (e) Este mai bine să avem efecte condiționate decât o acțiune care nu se poate aplica (20s)
- întotdeauna
  - niciodată
  - în cazul planificării fără senzori**
  - în cazul planificării contingente

**Explicație:** O acțiune nu se poate aplica dacă are cel puțin o condiție neîndeplinită. În cazul planificării fără senzori, în lipsa informațiilor din mediu, îndeplinirea sau nu a unor condiții nu poate fi validată. Pentru a se putea genera un plan se preferă definirea de acțiuni fără condiții. În schimb, efectele acțiunii vor fi avea loc doar dacă anumite condiții sunt îndeplinite.

- (f) Schemele de percepție sunt utilizate în (20s)
- planificarea fără senzori
  - planificarea contingentă**
  - atât planificarea fără senzori, cât și cea contingentă
  - nicio opțiune din cele anterioare

**Explicație:** Planificarea cu contingente se aplică în medii parțial observabile și nondeterministe. Agentul are nevoie să simtă starea în care a ajuns în urma unui răspuns nondeterminist al mediului. Agentul include în planul său printre acțiunile clasice din PDDL și acțiuni de tipul percepție. Schemele de percepție extind PDDL și au nevoie de date de la senzori pentru a fi activate.

- (g) Pentru identificarea succesului accidental (serendipity) agentul monitorizează din mediu: (20s)
- acțiunile
  - planul**
  - țelul

**Explicație:** Procesul de monitorizare a acțiunilor cere ca agentul să verifice dacă condițiile următoarei acțiuni din plan sunt încă îndeplinite. Fără o

astfel de verificare agentul se bazează doar pe computațiile făcute la momentul derivării planului. Monitorizarea planului cere ca agentul să verifice înainte de executarea următoarei acțiuni dacă întreg planul este încă realizabil. Într-un mediu nondeterminist, o condiție necesară unei acțiuni viitoare poate să nu mai fie îndeplinită la momentul curent. Monitorizarea țelului impune agentului ca înainte de executarea acțiunii următoare să verifice dacă nu este disponibil între timp un set de țeluri mai bine evaluat (conform funcției de utilitate a agentului). Apariția unui țel mai bun face ca planul curent să fie înlocuit de unul care are ca țintă noile obiective.

- (h) La pierderea semnalului GPS, navigatorul e mai bine să execute (20s)
- replanificare
  - repararea planului
  - depinde de lungimea planului și perioadă de pierdere a semnalului**
  - se oprește și așteaptă instrucțiuni de la agentul uman

**Explicație:** Deoarece posibila deviație 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 țel una din stările traseului inițial). Dacă pierderea semnalului a fost pe o perioadă semnificativă, poziția curentă a agentului poate fi la o distanță mare față de de planul inițial. Se poate identifica un traseu pornind din starea curentă care să nu aibă puncte comune cu traseul inițial și care să fie mai bun decât planul pentru readucere la traseul inițial plus finalizarea acestuia. În acest caz se recomandă replanificare. “în acest scenariu cu GPS replanificarea este mai sigură fiind dat costul redus în timp al planificării raportat la timpul pierdut executării unui plan neoptim.

- (i) Acțiunile concurente sunt necesare în cazul (20s)
- monitorizării planului
  - planificării cu contingente
  - planificării multi-agent**
  - planificării online

**Explicație:** Pentru a se sincroniza în executarea unei acțiuni comune, este necesar ca limbajul să permită în sisteme multi-agent modelarea acțiunilor concurente.

- (j) În cazul informațiilor incomplete se utilizează (20s)
- planificarea condițională**
  - planificarea cu contingente**

- planificarea fără senzori
- monitorizarea execuției și replanificare

**Explicație:**

(k) În cazul informațiilor eronate se utilizează

(20s)

- planificarea condițională
- planificarea contingentă
- planificarea fără senzori
- monitorizarea execuției și replanificare**

**Explicație:** Planificarea cu contingente (sau condițională) se aplică în medii parțial observabile și nondeterministe.

### 35. Calculul evenimentelor

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

- postdicția
- rezoluția**
- abducția
- predicția

**Explicație:** Rezoluția este o metodă de a demonstra propoziții prin reducerea la absurd.

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

- predicție
- abducție**
- postdicție
- doar limbajul PDDL poate fi utilizat pentru planificare

**Explicație:** Fiind date starea inițială și cea finală, abducția poate fi utilizată pentru a genera posibilele acțiuni care pot explica cum s-a ajuns în starea finală pornind dintr-o stare inițială dată.

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

- o observație**
- axiomă cu efect pozitiv
- axiomă cu efect negativ
- apariția unui eveniment

**Explicație:** Predicatul *HoldsAt* specifică valori unor fluenți la momente diferite de timp.

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

- o observație
- axiomă cu efect pozitiv
- axiomă cu efect negativ
- apariția unui eveniment**

**Explicație:** Predicatul *Happens* semnalizează apariția unor evenimente.

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

- Ocupa* este o relație injectivă**
- Ocupa* este o relație surjectivă

**Explicație:** Dacă obiectele  $p_1$  și  $p_2$  ocupă același spațiu  $s$  la momentul  $t$ , atunci cele două obiecte sunt identice. Aceeași valoare a funcției nu poate să aibă două intrări diferite. Deci relația este injectivă.

- (f)  $\neg HoldsAt(On(o, o), t)$  (30s)
- $On$  este o relație reflexivă
  - $On$  este o relație ireflexivă

**Explicație:** Orice obiect  $o$  nu poate să fie pe el însuși. Relația  $on$  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 însuși
  - telefonul  $p$  nu sună la timpul  $t$

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

- (h)  $\neg HoldsAt(Broken(d), t) \rightarrow Initiates(TurnOn(a, d), On(d), t)$  (30s)
- condiție de tip fluent
  - condiție de tip acțiune

**Explicație:** Dacă condiția  $\neg HoldsAt(Broken(d), t)$  are loc, atunci se poate executa acțiunea  $TurnOn(a, d)$ . Efectul acestei acțiuni va fi  $On(d)$ . Condiția este specificată prin  $HoldsAt$ , deci  $Broken(d)$  este un fluent. Altfel spus, dacă fluentul  $Broken(d)$  are loc la momentul  $t$ , atunci acțiunea se poate executa.

- (i)  $Happens(WalkThroughDoor(a, d), t) \rightarrow HoldsAt(Near(a, d), t)$  (30s)
- condiție de tip fluent
  - condiție de tip acțiune

**Explicație:** Condiția  $Happens(WalkThroughDoor(a, d), t)$  este de tip acțiune. Apariția evenimentului  $WalkThroughDoor(a, d)$  duce la apariția fluentului  $Near(a, d)$  la momentul  $t$ .



### 36. Probabilități condiționale

	Joacă fotbal	Nu joacă	Total
Băieți	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

**Explicație:** Din definiția probabilității: cazuri favorabile (75) supra cazuri posibile (110).

- (b)  $P(\text{baiat} | \text{joacaFotbal}) =$  (20s)
- 42/110  
 **42/54**  
 42/75  
 54/110

**Explicație:** Din 54 jucători de fotbal, 42 sunt băieți.

- (c)  $P(\text{baiat} \wedge \text{joacaFotbal}) =$  (20s)
- 42/110**  
 42/54  
 42/75  
 54/110

**Explicație:** Sunt 42 băieți care joacă fotbal din totalul de 110 persoane.

- (d)  $P(\text{baiat} \vee \text{joacaFotbal}) =$  (20s)
- 42/110  
 54/110  
 75/110  
 **87/110**

**Explicație:** Cazurile favorabile sunt date de 75 băieți la care se adaugă 12 fete.

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

- Da  
 Nu

**Explicație:**

$$P(\text{baiat}) = 75/110,$$

$$P(\text{joacaFotbal}) = 54/110,$$

$$P(\text{baiat} \wedge \text{joacaFotbal}) = 42/110$$

Deoarece  $P(\text{baiat} \wedge \text{joacaFotbal}) \neq P(\text{baiat})P(\text{joacaFotbal})$ , variabile nu sunt independente.

- (f) Care este probabilitatea de a extrage o carte roșie sau un as dintr-un pachet? (20s)
- 16/50  
 1/2  
 15/26  
 7/13

**Explicație:**

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

- (g) Care este probabilitatea de a alege o vocală din cuvântul ARTIFICIAL? (20s)
- 1/5  
 3/10  
 1/10  
 1/2

**Explicație:** Sunt 5 cazuri favorabile din 10 posibile.

- (h) Care este probabilitatea de a obține un multiplu de 3 dintr-o aruncare cu zarul și stema unei monede? (20s)
- 1/6  
 2/6  
 1/4  
 33%

**Explicație:** Cele 2 probabilități sunt independente. Astfel  $P(3k \wedge \text{stema}) = P(3k) * P(\text{stema}) = 2/6 * 1/2 = 2/12$ .

### 37. Rețele 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  $B$ .

**Explicație:**

- (b) Formula lui Bayes este (20s)
- $P(A, B) = P(A)P(B)$
  - $P(B|A) = P(A|B) * P(B)/P(A)$
  - $P(A) = \sum P(A, B)$
  - $P(B|A) = (P(A|B) * P(A))/P(B)$

**Explicație:** Formula lui Bayes este derivată direct din definiția probabilităților conditionate:

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

- (c) Pătura lui Markov (20s)
- nodurile sunt independente fiind dați părinții + copiii + copiii părinților**
  - este echivalenta cu tabelul complet al probabilităților
  - nodurile sunt independente de nondescendenți, fiind dați părinții

**Explicație:** Fiind date nodurile din pătura lui Markov (MB) a unui nod  $A$ ,  $A$  este independent de toate nodurile rămase în rețeaua Bayesiană.

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

- (d) Noisy-OR consideră (20s)
- toate cauzele posibile sunt incluse**
  - probabilitățile de eșec ale fiecărei clauze sunt independente**
  - nicio variantă din cele anterioare

**Explicație:** Noisy-OR este o generalizare a operatorului logic OR pentru a modela relații cu incertitudine. Incertitudinea se manifestă când efectul unei cauze nu se activează chiar dacă cauza a devenit adevărată. Noisy-OR are două asumptii: toate cauzele sunt enumerate, respectiv probabilitatea de inhibiție 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 rețea Bayesiană
  - acoperă toate cauzele de tipul și altele într-o relație Noisy-OR**
  - un nod condițional independent față de rădăcina rețelei Bayesiene.

**Explicație:** Deoarece Noisy-OR cere ca toate cauzele să fie menționate este necesar un astfel de nod.

- (f) O rețea Bayesiană hibridă (20s)
- este o rețea bayesiană modificată de un algoritm genetic
  - conține atât variabile aleatoare booleene, cât și nonbooleene
  - conține atât variabile discrete, cât și continue**
  - modelează cel puțin două domenii

**Explicație:** Distribuția de probabilitate în cazul variabilelor continue va fi dată de funcții distribuții de probabilitate.

### 38. Inferență în rețele Bayesiene

- (a) Care nu este un algoritm de inferență exactă? (20s)
- inferență prin enumerare
  - inferență prin eliminarea variabilelor
  - inferență prin simulare stocastică**
  - inferență prin Markov Chain Monte Carlo**

**Explicație:** Primii doi sunt algoritmi de raționare exactă. Ultimii doi sunt algoritmi de raționare aproximativă în rețele Bayesiene.

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

**Explicație:** Intr-o rețea Bayesiană sunt 3 tipuri de variabile: întrebare ( $X$  în cazul curent), evidențe ( $j, m$ ), respectiv variabile ascunse - restul nodurilor din rețea (i.e.  $b$  și  $e$ ).

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

**Explicație:** Evidențele sunt  $j$  și  $m$ . Variabilele aleatoare neinstantiate apar cu litere mari, în timp ce observațiile cu litere mici.

- (d) Care algoritm folosește factori? (20s)
- inferență prin enumerare
  - inferență prin eliminarea variabilelor**
  - inferență prin simulare stocastică
  - inferență prin Markov Chain Monte Carlo

**Explicație:** Factorii evită calcularea repetată a aceluiași produs de probabilități, aspect care apare frecvent la algoritmul de eliminare a variabilelor.

- (e) Metoda *Rejection Sampling* generează doar evenimente consistente cu evidențele  $e$  (20s)
- Adevărat

**Fals**

Eșantionarea prin revocare (rejection sampling) este o metodă de raționare aproximativă în rețele Bayesiene. Metoda generează eșantioane conform distribuției de probabilitate specificate. Dintre aceste eșantioane le păstrează doar pe cele consistente cu evidențele date. Probabilitatea condiționată este obținută prin numărarea cazurilor favorabile din eșantionul rămas.

**Explicație:**

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

**Fals**

**Adevărat**

**Explicație:** Markov Chain Monte Carlo este o metodă de raționare aproximativă în rețele Bayesiene. În loc de a genera de fiecare dată eșantioane complete noi, metoda pornește cu un eșantion căruiia îi modifică aleator o singură valoare la pasul următor. Așadar eșantioanele nu sunt generate integral, ci reprezintă modificări ale ultimei generări.

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

toate variabilele menționate în interogare și toți părinții acestora

**toate variabilele menționate în interogare și toți ancesorii acestora**

toate variabilele menționate în interogare și toți copiii acestora

toate variabilele menționate în interogare și toți descendenții acestora

**Explicație:** Algoritmul determină dacă două variabile sunt independente într-o rețea Bayesiană, fiind date anumite evidențe. La primul pas se selectează din rețeaua Bayesiană doar variabilele menționate în interogare și toți ancesorii acestora.

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

**folosește evidențele pentru a pondera eșantioanele generate**

generează eșantioane dintr-o rețea goală

respinge eșantioanele inconsistente cu evidențele

**Explicație:** Metodă generează valori doar pentru variabilele care nu sunt evidențe. Se asigură astfel că fiecare eșantion este consistent cu evidențele. Fiecare eșantion este ponderat cu probabilitatea de a apărea în asociere cu evidențele date.

### 39. Raționare probabilistică cu timp

- (a) În asumptia Markov, starea curentă depinde de (20s)
- starea anterioară
  - un număr finit și fix de stări anterioare**
  - de părinți + copii + copiii părinților

**Explicație:** Doar în lanțuri Markov de ordinul I starea curentă depinde doar de starea anterioară.

- (b) Un proces staționar 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

**Explicație:** În cazul unui proces staționar, modelul de tranziție și cel al senzorilor nu variază în timp.

- (c) Asumptia Markov a senzorilor (60s)
- $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t, E_{0:t-1})$
  - $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{0:t})$
  - $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_t)$
  - $P(E_t|X_{0:t}, E_{0:t-1}) = P(E_t|X_{t-1})$

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

- (d) Filtrarea sau estimarea stării curente este formalizată cu (30s)
- $P(X_t|e_{1:t})$
  - $P(X_{t+k}|e_{1:t})$
  - $P(X_k|e_{1:t}), k < t$
  - $\operatorname{argmax}(X_{1:t})P(X_{1:t}|e_{1:t})$

**Explicație:** Se estimează distribuția de probabilitate a stării curente  $X$  la momentul  $t$  pe baza observațiilor  $e_{1:t}$  obținute până acum.

- (e) Netezirea (smoothing) este formalizată cu (30s)
- $P(X_t|e_{1:t})$

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

**Explicație:** Se îmbunătățește estimarea unei stări anterioare  $X_k$  în lumina noilor date obținute de la momentul  $k$  până la momentul curent  $t$ .

(f) Predicția este formalizată cu (30s)

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

**Explicație:** Se prezice distribuția de probabilitate a variabilei de stare  $X$  la un moment ulterior  $t + k$  pe baza datelor obținute până la momentul curent  $t$ .

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

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

**Explicație:** Ne interesează cea mai probabilă secvență de instanțieri a variabilei aleatoare  $X$  de la momentul inițial până la cel curent ( $\text{argmax}(X_{1:t})$ ) pe baza tuturor măsurătorilor disponibile  $e_{1:t}$

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

- rezoluția**
- predicția
- netezirea
- identificarea celei mai probabile explicații

**Explicație:** Rezoluția este o metodă de raționare pe anumite logici. De exemplu, rezoluția în logica de ordinul întâi este utilă în demonstrarea de teoreme.

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

- modelul stării inițiale
- modelul de tranziție
- modelul senzorilor
- distribuția Gaussiană**



**Explicație:** Trei informații sunt necesare pentru a rula algoritmi de predicție, netezire, filtrare sau de identificare a celei mai probabile explicații. Aceste informații sunt: 1) modelul stării inițiale, 2) modelul de tranziție, 3) modelul senzorilor.

- (j) Învățarea unei rețele Bayesiene dinamice se referă la (60s)
- a învăța din observații modelul de tranziții
  - a interpreta orice nod din rețea ca un neuron
  - estimarea stării, filtrare și netezire
  - a învăța din observații modelul senzorilor

**Explicație:** Atât modelul senzorilor cât și cel de tranziție pot fi învățate.

#### 40. Decizii simple

- (a) Micromortul este (20s)
- unitatea de măsură pentru utilitate
  - 1/1,000,000 șanse de deces**
  - măsura pentru metrica QUALY (Quality Adjusted Life Years)
  - identic cu microprobabilitatea

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

- (b) Sistemul de asigurări are la bază faptul că agenții umani sunt față de risc (30s)
- neutri
  - împotriva**
  - în căutarea riscului

**Explicație:** Aversiunea față de risc înseamnă că agenții umani evaluează fiecare ban în plus mai puțin dect predecesorul.

- (c) Funcția de utilitate a banilor este (30s)
- liniar crescătoare
  - constantă
  - logaritmică**
  - exponențială

**Explicație:** Până la o anumită valoare utilitatea crește proporțional cu banii. Peste un anumit prag, utilitatea crește mult mai puțin relativ la creșterea masei monetare.

- (d) La ce este utilizat principiul MEU? (30s)
- Minimize Expected Utility
  - calcularea utilității
  - selectarea acțiunii în starea curentă**
  - explicarea comportamentului irațional

**Explicație:** MEU (Maximize Expected Utility) urmărește maximizarea utilității expectate prin selectarea celei mai promițătoare acțiuni din starea curentă.

- (e) Cum se dorește să acționeze agenții software (perspectiva AIMA)? (30s)
- deterministic
  - rațional**

- cât mai uman
- rapid

**Explicație:** Agenții echipați cu teoria utilităților nu urmăresc imitarea comportamentului uman, ci un comportament rațional.

- (f) Cu ce se extinde o rețea Bayesiană pentru a deveni o rețea de decizie? (30s)
- noduri de tip acțiune și utilitate**
  - noduri utilitate
  - noduri MEU
  - Cu nimic. O rețea Bayesiană este deja o rețea de decizie

**Explicație:** Rețelele de decizie conțin 3 tipuri de noduri pentru: i) variabile aleatoare, ii) acțiuni, iii) utilități.

- (g) Valoarea expectată a informației este (30s)
- negativă
  - aditivă
  - depinde de ordinea în care sunt primite informațiile
  - nicio versiune din cele anterioare**

**Explicație:** Valoarea informației este dată de creșterea expectată a utilității comparată cu cazul în care decizia se ia fără a avea informația respectivă. Fie informația curentă  $E$ , cea mai bună acțiune curentă  $\alpha$ , și potențiala informație  $e_j$ .

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

Valoarea celei mai bune acțiuni după obținerea informației  $E_j = e_j$  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 aleatoare a cărei valoare nu este cunoscută momentan, se calculează câștigul expectat peste toate valorile posibile  $e_{jk}$  ale informației  $E_j$ .

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

Asumpția este că înainte de obținerea informației  $E_j$  avem o distribuție de probabilitate asupra posibilelor sale valori. Valoarea expectată a informației are următoarele proprietăți:

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 față 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 informația inițială (30s)

- efectul certitudinii
- încadrarea (framing)
- efectul de ancoră**

**Explicație:** Informația inițială acționează 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 opțiunile sunt prezentate: pozitiv sau negativ (30s)

- efectul certitudinii
- încadrarea (framing)**
- efectul de ancoră

**Explicație:** Considerați următoarele opțiuni:  $O_1$ : Aceasta operație are 90% șanse de succes, respectiv  $O_2$ : Aceasta operație are 10% șanse de eșec. Numărul pacienților care acceptă operație prezentată ca  $O_1$  este dublu față de cei care au primit informația  $O_2$ . Similar:  $O_3$  Această pastă de dinți conține 98% elemente naturale., respectiv  $O_4$  Această pastă de dinți conține 2% elemente sintetice.

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

- normativă**
- descriptivă

**Explicație:** Teoria deciziilor este o teorie normativă deoarece descrie cum ar trebui un agent rațional să acționeze. O teorie descriptivă descrie sau explică cum acționează în realitate un agent.

#### 41. Decizii secvențiale

- (a) Problemele de decizii secvențiale (SDP) (30s)
- includ utilități, incertitudine și captarea informațiilor din mediu**
  - utilitatea agentului depinde de o secvență de decizii**
  - sunt cazuri particulare ale problemelor de planificare
  - sunt cazuri particulare de probleme de decizie Markoviene

**Explicație:** Problemele de căutare, planificare și cele Markoviene sunt cazuri particulare ale problemelor de decizii secvențiale.

- (b) Problemele de decizie Markoviene sunt probleme secvențiale pentru mediile (60s)
- complet observabile și stocastice**
  - complet observabile și deterministe
  - cu modele de tranziție Markoviene și funcții aditive de recompensă**
  - parțial observabile și stocastice

**Explicație:** Dacă mediul este parțial 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 acțiune pentru fiecare stare  $s$**
  - are cea mai mare valoare expectată a utilității**

**Explicație:** Politica  $\pi$  este o funcție care primește o stare  $s$  și întoarce cea mai bună acțiune din starea respectivă.

- (d) O politică optimală pentru un orizont finit este (20s)
- nonstaționară**
  - staționară

**Explicație:** În politicile nonstaționare  $\pi(s)$  depinde de timpul rămas. În cazul unui orizont de timp infinit, nu există niciun motiv pentru ca agentul să se comporte diferit într-o aceeași stare.

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

**Explicație:** Factorul de discount este între 0 și 1. Cu valori apropiate de 1, recompensele pe termen lung au același impact precum recompensele pe termen scurt.

(f) Ecuația lui Bellman este (60s)

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

**Explicație:** Prima variantă consideră acțiunile posibile din starea succesoare ( $\max_{a \in A(s')}$ ), și nu pe cele din starea curentă ( $\max_{a \in A(s)}$ ). În plus se consideră utilitatea stării curente  $U(s)$  și nu utilitatea stării succesoare. A doua și a patra versiune consideră acțiunile 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 licitații au caracter public și sunt descendente? (30s)

- englezești
- olandeze**
- cu plic închis
- Vickrey

**Explicație:** Licitațiile englezești sunt publice, dar prețul are caracter ascendent. Licitațiile cu plic închis au caracter privat: agenții nu au acces la valorile licitate de agenții concurenți. Licitațiile de tip Vickrey sunt un caz particular al licitațiilor cu plic închis în care agentul care licitează cel mai mult este câștigător, dar la cea de-a doua valoare licitată de către alt agent. În licitația olandeză agentul vânzător pornește cu un preț inițial cunoscut de către toți participanții și care este scăzut la fiecare pas până când devine atrăgător pentru un cumpărător. Singurul protocol din cele enumerate cu caracter public și preț descendent este deci licitația olandeză.

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

- nu există o altă soluție pe care unul dintre agenți o preferă
- nu există o altă soluție pe care toți agenții să o prefere**
- există o altă soluție pe care unul dintre agenți o preferă
- există o altă soluție pe care toți agenții să o prefere

**Explicație:** Optimalitatea Pareto se referă la toți agenții implicați.

## 42. Arbori de decizie

- (a) Dacă predicatul țintă are un set finit de valori, atunci problema de învățare este (10s)
- regresie
  - clasificare**

**Explicație:** Dacă predicatul țintă are o infinitate de valori (e.g. se dorește prezicerea prețului unui apartament) 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**
  - $X^2 + 1/3$  față de  $X^3$

**Explicație:** Principiul lamei lui Ockham favorizează simplitatea în fața acurateții. Principiul reprezintă o metodă de evitare a suprapotrivirii (overfitting).

- (c) O problemă de învățare este realizabilă dacă (30s)
- spațiul ipotezelor conține funcția adevărată**
  - datele sunt consistente și fără zgomote
  - se aplică un algoritm de învățare potrivit pentru date

**Explicație:** Spațiul de căutare a soluțiilor trebuie să conțină funcția căutată.

- (d) Arborele de decizie este construit pe baza unei strategii de căutare de tipul (30s)
- căutare în adâncime
  - căutare în lățime
  - greedy și desparte-și-cucerește**
  - aleator

**Explicație:** Cel mai apetisant atribut este cel care aduce cel mai mare câștig de informație. 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 învățarea nu este deterministic**
  - nu există acces la un atribut prin care s-ar putea distinge între instanțe**
  - nici una din opțiunile anterioare

**Explicație:** Pentru arborii de decizie, oricare din primele 3 variante poate fi o cauză acestei situații.

- (f) Mai multă informație înseamnă (10s)
- mai multă entropie
  - mai puțină entropie**

**Explicație:** Cu cât un sistem este mai agitat, cu atât informația este mai puțină.

- (g) Suprapotrivirea (overfitting) este (10s)
- de dorit
  - de evitat**

**Explicație:** Cu cât un sistem este mai agitat, cu atât informația este mai puțină.

- (h) Suprapotrivirea (overfitting) (20s)
- violează principiul lamei lui Ockham**
  - este sinonimă cu supraantrenarea**
  - include mai mulți parametri decât cei necesari**
  - toate variantele anterioare**

**Explicație:** Urmând principiul simplității suprapotrivirea poate fi evitată prin preferarea de clasificatori mai simpli, chiar dacă au o acuratețe mai slabă. O altă denumire pentru *overfitting* este *overtraining*. Practic, parametrii algoritmului de învățare sunt reglați 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 ajustați. Datorită acestei specializări pe datele de antrenament clasificatorul își 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

**Explicație:** Datele disponibile sunt împărțite în două seturi distincte. Se aplică învățarea doar pe datele de antrenament. Metoda *K-fold cross validation* elimină acest dezavantaj.

- (j) Performanța învățării este dată de acuratețea prezicerii pe setul de (30s)



- antrenare
- validare
- testare**
- media acurateții pe cele 3 seturi

**Explicație:** Setul de testare conține date care au fost ținute într-un seif pe durata învățării. Doar după terminarea învățării se deschide seiful și se evaluează performanța modelului obținut.

(k) *Peeking* apare dacă

(60s)

- ipoteza se selectează și pe baza erorii pe setul de testare**
- nu se utilizează set pentru validare**
- curba de învățare (ROC) este un *graf vesel*
- atributul țintă este continuu

**Explicație:** Setul de testare nu trebuie să influențeze modelul învățat pe datele de antrenament/validare. Acestea trebuie să fie distincte de cele utilizate în testare.

### 43. Cunoștințe în învățare

(a) Care este diferența dintre regresie și clasificare? (30s)

- natura atributului țintă**
- o metodă este supervizată, iar cealaltă nesupervizată
- o metodă utilizează statistica, iar cealaltă nu
- nu e nicio diferență

**Explicație:** Ambele clase țin de învățarea supervizată. La clasificare, atributul țintă este discret/finit (e.g. două clase Spam/Not Spam). La regresie, atributul țintă este continuu (e.g. viteza vehiculului din față).

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

- fals pozitiv
- fals negativ**
- adevărat pozitiv
- adevărat negativ

**Explicație:** Instanța este clasificată greșit ca fiind negativă. Avem deci un fals negativ.

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

- generalizare
- specializare**

**Explicație:** Instanța este clasificată greșit ca pozitivă. Clasificatorul este deci prea general (i.e. etichetează prea multe instanțe ca pozitive). Se impune o specializare a acestuia.

(d) Generalizarea are loc prin (30s)

- adăugarea de condiții
- eliminarea unor condiții**

**Explicație:** Cu cât are mai multe condiții, ipoteza de clasificare este mai specifică. Generalizarea se obține prin îndepărtarea unei condiții din ipoteză.

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

- menține o singură ipoteză care se ajustează cu fiecare exemplu
- este deterministică**
- modificarea ipotezei impune verificarea instanțelor anterioare
- execută mult backtracking

**Explicație:** Algoritmul lucrează cu o singură ipoteză. Modificarea ipotezei de către instanțe fals negative sau fals pozitive poate duce la o clasificare greșită a instanțelor deja analizate. Reverificarea acestora generează backtracking. Modificarea ipotezei prin generalizare sau specializare se face nondeterminist.

- (f) Care nu este o condiție de stop a algoritmului *version space*? (60s)
- rămâne doar o ipoteză în spațiul soluțiilor
  - spațiul soluțiilor colapsează ( $\mathcal{G}$  sau  $\mathcal{S}$  devin vide)
  - rămân mai multe ipoteze și nu mai există instanțe de analizat
  - complexitatea ipotezei este mai mare decât un prag**

**Explicație:** Primele trei opțiuni sunt condiții de oprire a algoritmului. Dacă o mulțime devine vidă, exemplele de antrenament sunt inconsistente. Dacă învățarea se termină cu mai multe ipoteze de clasificare, toate acestea vor fi aplicate pe exemplele noi. Metode de tip *vote* pot fi aplicate în cazul în care aceste ipoteze nu sunt consistente între ele cu privire la clasa unei instanțe.

- (g) Algoritmul FOIL evită suprapotrivirea (overfitting) prin (30s)
- validare încrucișată
  - penalizând ipotezele mari**
  - folosind cunoștințe din domeniu

**Explicație:** Conform principiului lui Ockham.

- (h) Învățarea bazată pe relevanță (30s)
- extrage reguli generale dintr-un singur exemplu
  - folosește cunoștințe anterioare**

**Explicație:** Cunoștințele anterioare indică dacă ipotezele deduse prin inducție sunt relevante sau nu.

- (i) Ce nu este adevărat la *version space*? (30s)
- este o reprezentare hierahică a cunoștințelor
  - valorile atributelor sunt continue**
  - utilizează arbori de generalizări și specializări
  - presupune ca datele sunt corecte

**Explicație:** Metoda funcționează doar dacă atributele din setul de antrenament sunt discrete (nominale).

#### 44. Rețele neurale artificiale

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

- ⟨ponderi, nod, intrare, ieșire⟩
- ⟨**nod, ponderi, intrare, ieșire**⟩
- ⟨intrare, nod 3, ponderi 4, ieșire⟩
- ⟨intrare, ponderi, nod, ieșire⟩

**Explicație:** Într-o rețea neurală artificială nodurile reprezintă neuronii, iar sinapsele sunt modelate cu ponderi pe legăturile dintre neuroni. Dendritele reprezintă conexiunile de intrare în neuron, iar axonul reprezintă impulsul generat la activarea neuronului.

(b) 20 Rețele neurale sunt metode de învățare

- supervizată
- nesupervizată
- reinforșată
- depinde de tipul rețelei neurale**

**Explicație:** Rețelele neurale artificiale sunt metode supervizate. Mașinile Boltzmann sunt rețele neurale recurente stocastice în care învățarea este nesupervizată.

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

- and
- or**
- xor
- not

**Explicație:** Pentru intrarea ⟨0, 0⟩ ieșirea e  $-0.5 + 1 \cdot 0 + 1 \cdot 0 = -0.5$ . Pentru intrarea ⟨0, 1⟩ ieșirea e  $-0.5 + 1 \cdot 0 + 1 \cdot 1 = 0.5$ . Pentru intrarea ⟨1, 0⟩ ieșirea e  $-0.5 + 1 \cdot 1 + 1 \cdot 0 = 0.5$ . Pentru intrarea ⟨1, 1⟩ ieșirea e  $-0.5 + 1 \cdot 1 + 1 \cdot 1 = 1.5$ . Neuronul se activează în ultimele 3 cazuri. Pentru cele 4 perechi de intrare, ieșirile neuronului vor fi în ordine 0, 1, 1, 1, ceea ce corespunde porții logice OR.

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

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

**Explicație:** Pentru intrarea  $\langle 0, 0 \rangle$  ieșirea 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  $\langle 0, 1 \rangle$  ieșirea 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 funcția XOR (20s)

**Adevărat**

Fals

**Explicație:** XOR nu este liniar separabilă, iar perceptronul este un clasificator liniar.

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

Adevărat

**Fals**

**Explicație:** Învățarea are loc într-adevăr prin ajustarea ponderilor între neuroni. Dar acest lucru urmărește reducerea erorii pe setul de antrenament, nu pe cel de test. Setul de test nu trebuie să influențeze parametrii învățării!

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

sunt similare

**nu diferite**

**Explicație:** Asemănările între creierul uman și rețelele 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

**Explicație:** Mașinile cu suport vectorial crează un hiperplan pentru separarea a două clase. Prin introducerea unor coordonate suplimentare și modificarea valorilor inițiale pot fi clasificate și date nonliniare.

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

Adevărat

**Fals**

**Explicație:** Un număr foarte mic de vectori sunt suficienți pentru a separa cele două clase.

- (j) *Boosting* este (20s)
- o metodă de învățare cu ansambluri de clasificatori
  - un caz particular de arbore de decizie
  - o metodă de eficientizare a învățării prin paralelizare

**Explicație:** Un exemplu este algoritmul AdaBoost. Ansamblul de clasificatori poate fi format dintr-un set de arbori de decizie de dimensiune redusă.

- (k) Învățarea cu ansambluri poate utiliza clasificatori diferiți (e.g. ANN, SVM, DT) (20s)
- Adevărat
  - Fals

**Explicație:** Metode de tip votare se pot aplica pentru a rezolva situațiile în care clasificatorii au păreri diferite cu privire la eticheta unei instanțe.

#### 45. Clusterizare. K-means. Clusterizare ierarhică

- (a) K-means este un algoritm de (10s)
- clusterizare herarhică
  - clusterizare prin partiționare**

**Explicație:** Parametrul  $K$  reprezintă numărul de partiții ale setului de date.

- (b) K-means are nevoie de date normalizate și nenominale (20s)
- True**
  - False

**Explicație:** Fără normalizare anumite atribute nu vor avea contribuție relevantă în calcularea distanței între instanțe. Atributele nominale introduc dificultăți la calcularea distanței dintre instanțe.

- (c) În suma erorilor pătratice (SSE) pentru fiecare instanță se (30s)
- însumează distanța până la cel mai apropiat centroid
  - ridică la patrat distanța până la cel mai apropiat centroid și se însumează rezultatele**
  - însumează distanțele la toate celelalte puncte din cluster
  - ridică la patrat distanțele față de toate celelalte puncte din cluster și se însumează rezultatele.

**Explicație:** SSE este o metrică pentru evaluarea clusterizării. Pentru fiecare punct, eroarea este distanța până la cel mai apropiat centroid. Erorile pentru toate punctele din setul de date se ridică la pătrat și 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  $K$  crește atunci SSE (10s)
- scade**
  - crește

**Explicație:** Creșterea numărului de clustere duce la distanțe mai mici între puncte și centroizii care sunt mai numeroși.

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

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

**Explicație:** Dacă clusterelor au un număr egal de puncte atunci

$$P = \frac{\text{numărul de posibilități pentru a selecta un centroid din fiecare cluster}}{\text{numărul de posibilități de a selecta K centroizi}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

Pentru  $k = 3$ , probabilitatea este  $\frac{3!}{3^3} = \frac{2}{9}$ . Șansele devin extrem de mici pe măsură ce crește  $K$ . De exemplu pentru  $K = 10$  probabilitatea este  $10!/10^{10} = 0.00036$ .

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

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

**Explicație:** Rulările multiple ajută la creșterea șanselor de a avea centroizi în fiecare cluster real. În urma rulărilor succesive se preferă clusterizarea cu valoarea SSE cea mai mică. Algoritmii Bisecting K-means inițializează la fiecare pas doar 2 centroizi. Astfel, șansele ca aceștia să aparțină la clusteri distincți este mai mare comparativ cu  $k > 2$ . Clusterizarea ierarhică poate fi utilizată în generarea centrozilor inițiali.

(g) Care din următoarele nu este o metrică pentru distanța dintre cluster? (30s)

- min (single link)
- media grupului
- dendograma**
- distanța dintre centroizi

**Explicație:** Metricile utilizate pentru calcularea distanței între cluster sunt: (i) min (single link), (ii) max (complete link), (iii) distanța dintre centroide, (iv) media distanțelor dintre toate punctele din cluster, (v) rata de creștere a valorii SSE (metoda Ward). Dendograma este o reprezentare grafică a ordinii în care se face gruparea punctelor, precum și a distanței dintre cluster.

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

- Bisecting K-means**
- K-means



**clusterizare ierarhică**

toate menționate anterior

**Explicație:** În cazul Bisecting K-means și a clusterizării ierarhice, numărul de clustere se decide după rularea algoritmului.

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

nu este susceptibilă la excepții

nu e sensibilă la zgomote

**nicio varianta din cele anterioare**

**Explicație:** Clusterizarea cu metrica *min* are limitări legate de sensibilitatea la zgomote sau excepții.

(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 excepții

**Explicație:** Clusterizarea cu metrica *max* tinde să genereze clustere de dimensiune relativ egală. Dacă datele conțin grupuri de dimensiune diferite, metrica *max* nu este adecvata: puncte din grupurile mari vor fi înglobate în clusterelor mici.

#### 46. Reguli de asociație. Algoritmul Apriori

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

- nu  
 **da**

**Explicație:** Suportul este simetric dacă  $s(X \rightarrow Y) = s(Y \rightarrow X)$  pentru toate mulțimile  $X$  și  $Y$ . Suportul este dat de mulțimea  $X \cup Y$ , care este aceeași pentru ambele reguli.

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

- nu**  
 da  
 poate

**Explicație:** Confidența este simetrică dacă  $c(X \rightarrow Y) = c(Y \rightarrow X)$  pentru toate mulțimile  $X$  și  $Y$ . Considerați următorul exemplu:  $s(\{e\}) = 0.8$ ,  $s(\{b, d\}) = 0.2$ ,  $s(\{b, d, e\}) = 0.2$  Confidența

$$c(\{b, d\} \rightarrow \{e\}) = \frac{0.2}{0.8} = 0.4$$

iar

$$c(\{e\} \rightarrow \{b, d\}) = \frac{0.2}{0.2} = 1$$

Contraexemplul demonstrează că nu este o metrică simetrică.

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

- $2^d$   
  $d^2$   
  $d!$   
  $(d - 1)!$

**Explicație:** Cu  $d$  itemi se pot forma  $2^d$  mulțimi distincte (incluzând mulțimea vidă și mulțimea formată din toți cei  $d$  itemi).

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

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

**Explicație:** 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  $1 \leq k < d$ , numărul de posibilități de formare a părții 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 posibilitățile sunt date de produsul dintre numărul de combinații a părții stângi cu numărul de combinații a părții 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 depășește suportul submulțimilor sale. (20s)
- fals
- adevărat

**Explicație:** O mulțime mai mare nu se va regăsi mai des în setul de date decât oricare din submulțimile sale.

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

**Explicație:** Dacă un set de itemi este frecvent, atunci toate subseturile sale sunt frecvente. Altfel spus, suportul unui set nu depășește 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 mulțime de itemi frecvenți  $L$  ( $|L| = k$ )? (30s)
- $2^k$
- $2^{k-2}$
- $2^k - 2$
- $k!$

**Explicație:** Se exclud regulile cu 0 respectiv  $k$  itemi în partea stângă a regulii.

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

fals

**Explicație:** Cu cât numărul de itemi din partea dreaptă crește, cu atât scade confidența regulii (e.g.  $c(a \rightarrow bc) < c(a, b \rightarrow c)$ ).

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

(30s)

- ABC, ABD
- ABD, ABE
- ABC, ACD**
- ABC, ABE

**Explicație:** Seturile  $ABD$  și  $ACD$  au un prefix comun doar de lungime 1. Pentru gruparea a două mulțimi de dimensiune 3 este necesar un prefix comun de lungime 2. Toate celelalte perechi au un astfel de prefix.

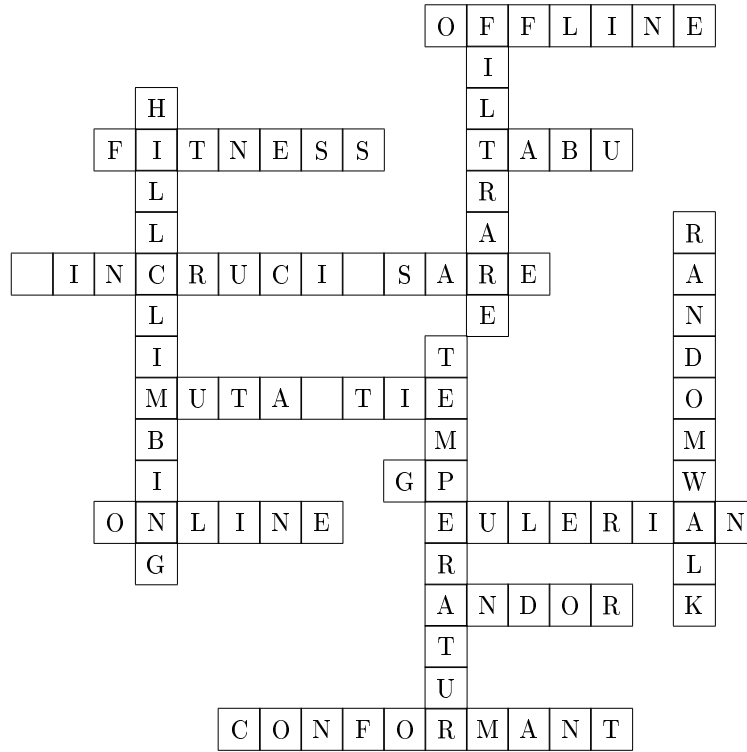
## Capitolul 4

# Rebusuri în inteligență artificială



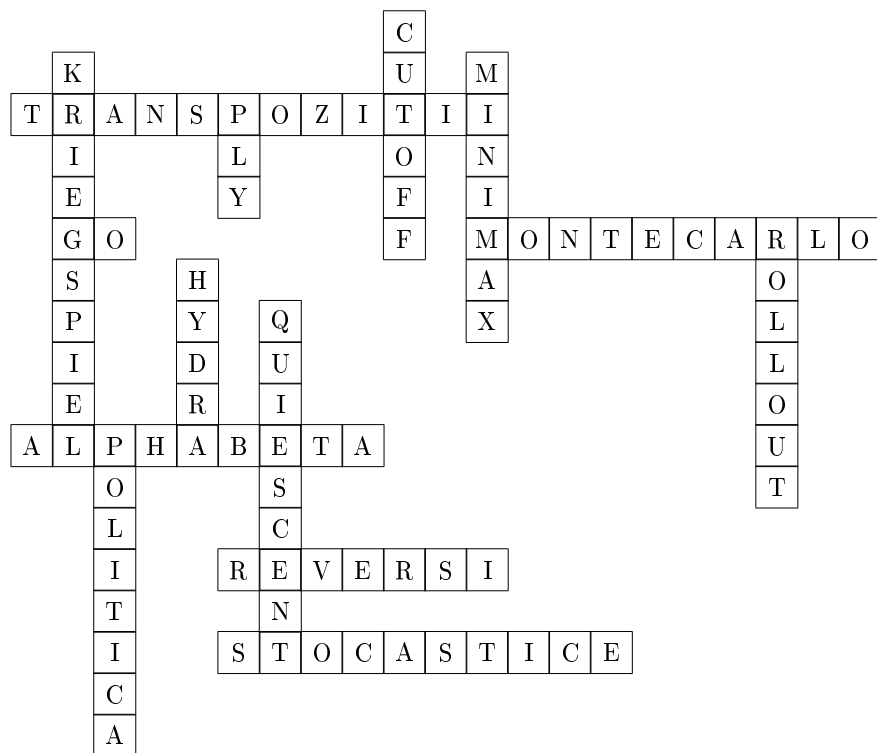


Căutare locală

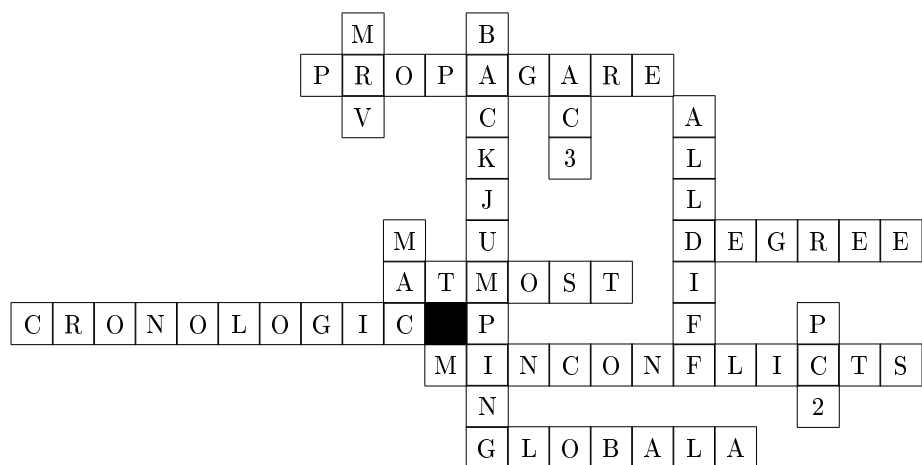




### Căutare adversarială

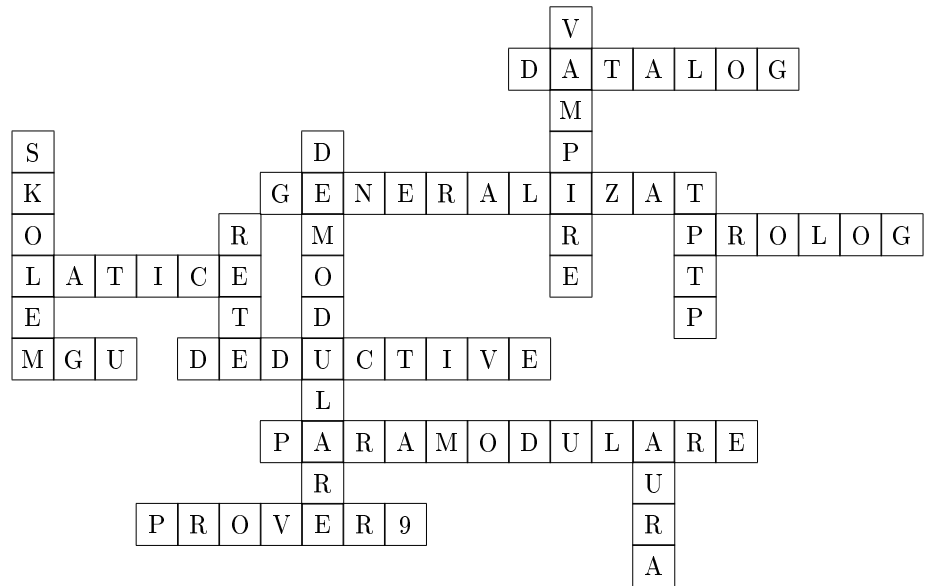


### Problem de satisfacere a constrângerilor

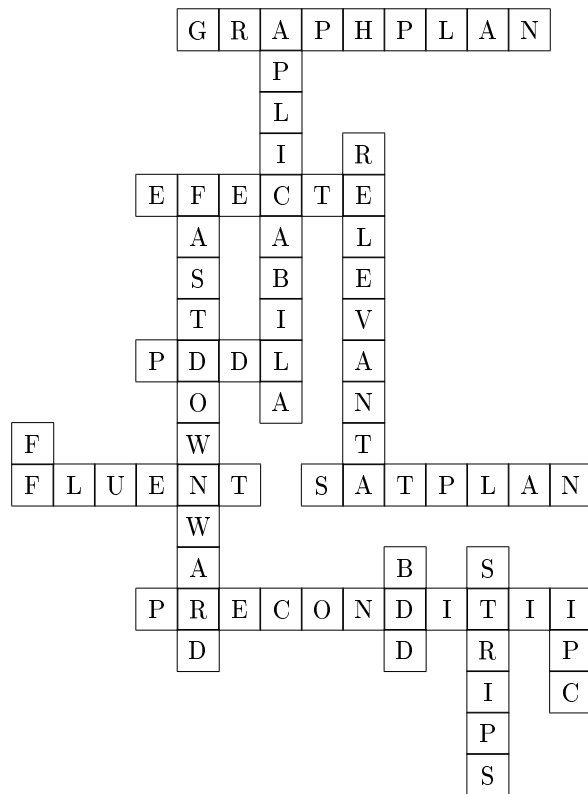




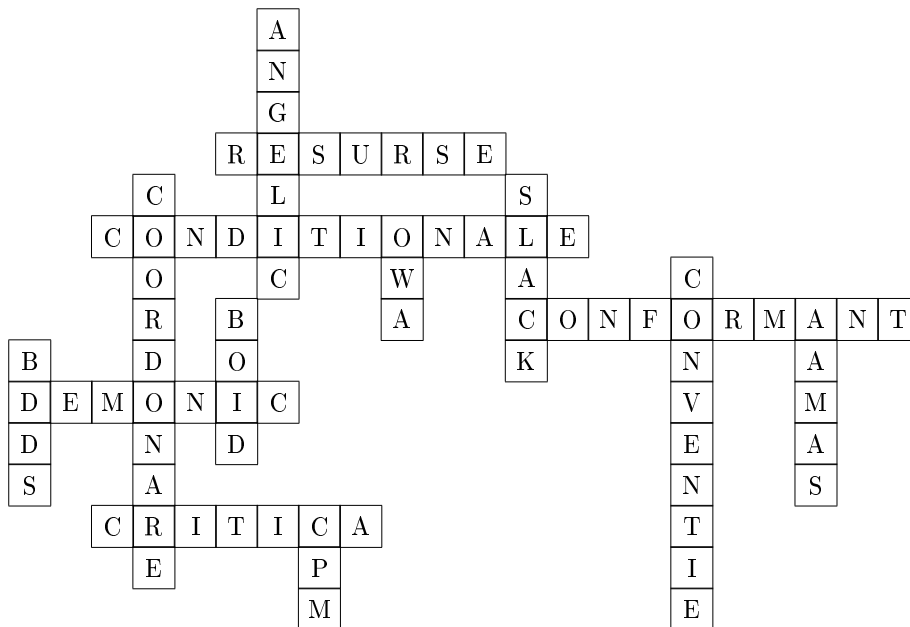
Inferență în logica de ordinul întâi



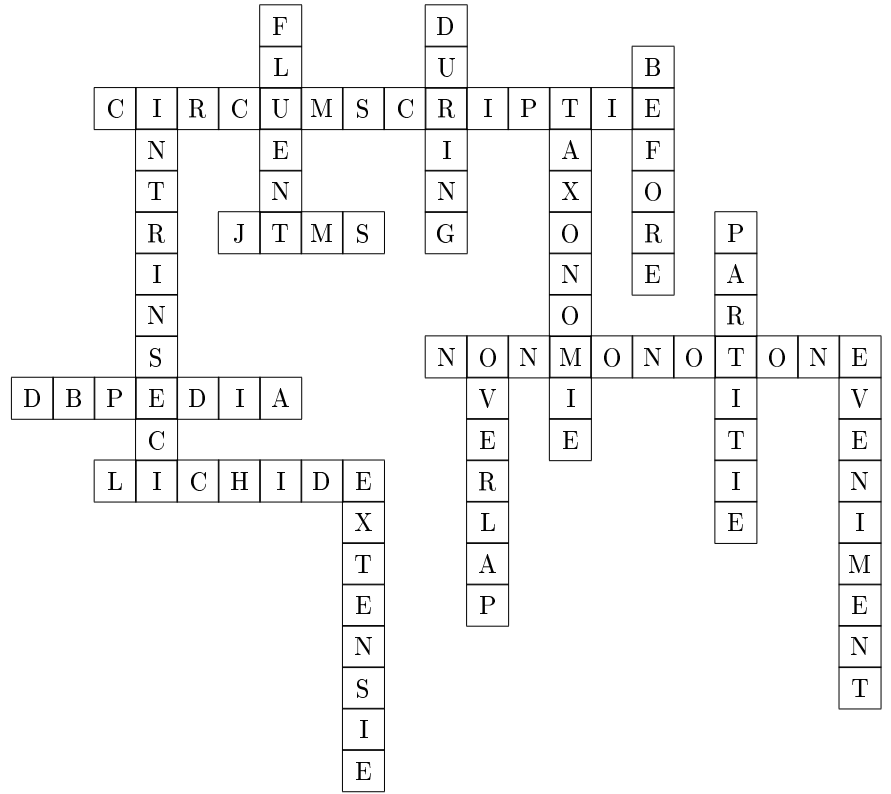
Planificare clasică



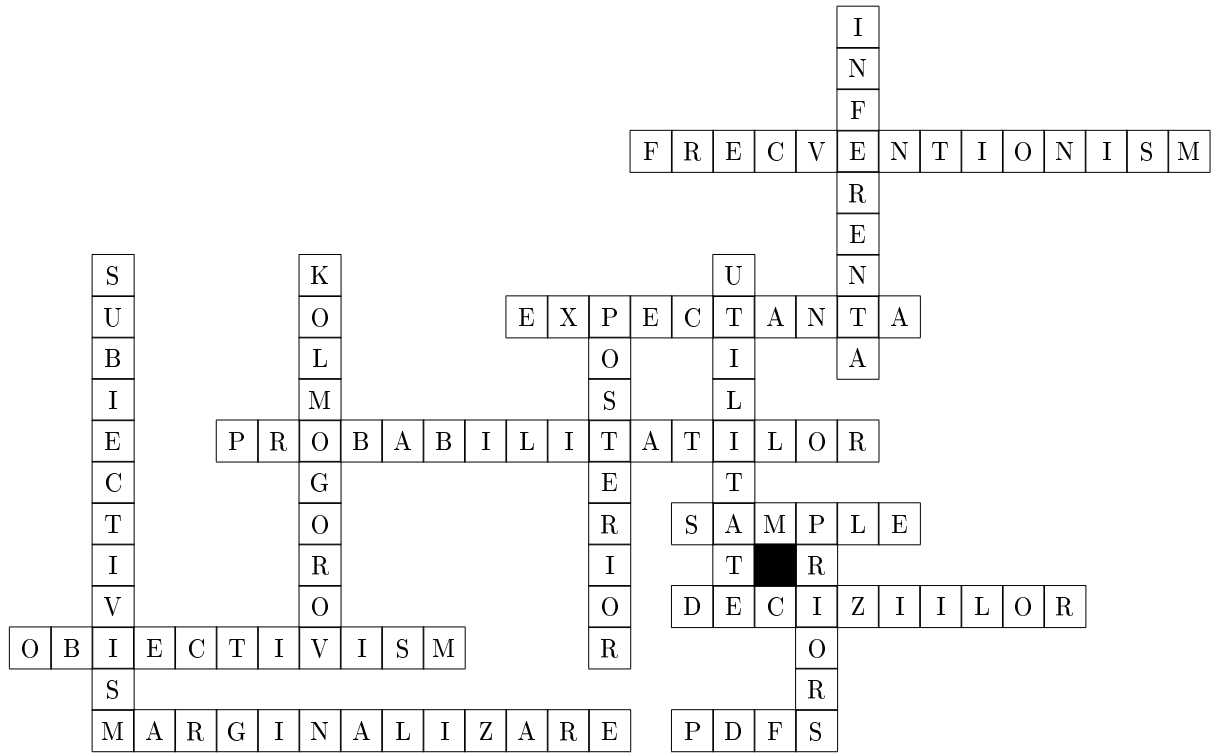
### Planificare în lumea reală



Reprezentarea cunoștințelor. Calculul evenimentelor



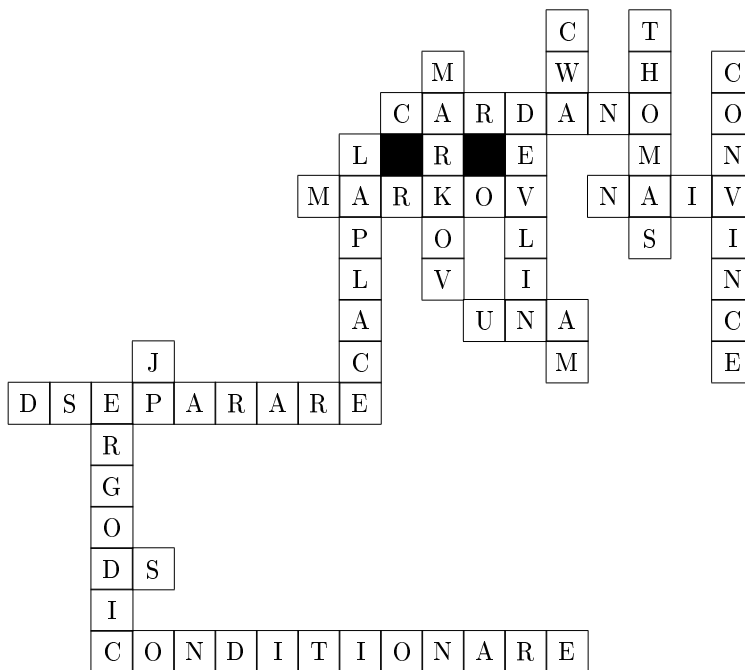
## Probabilități



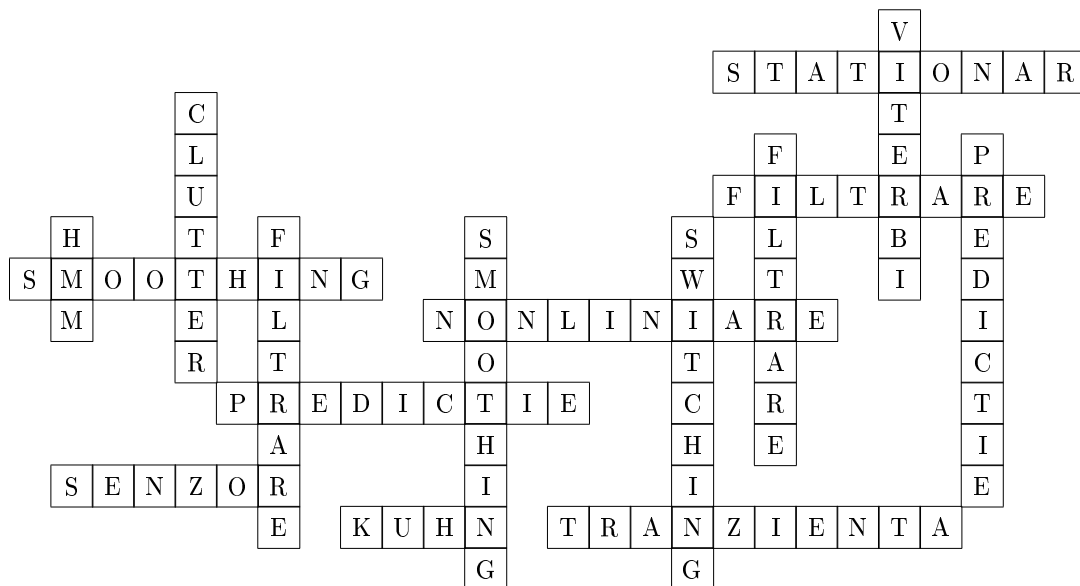




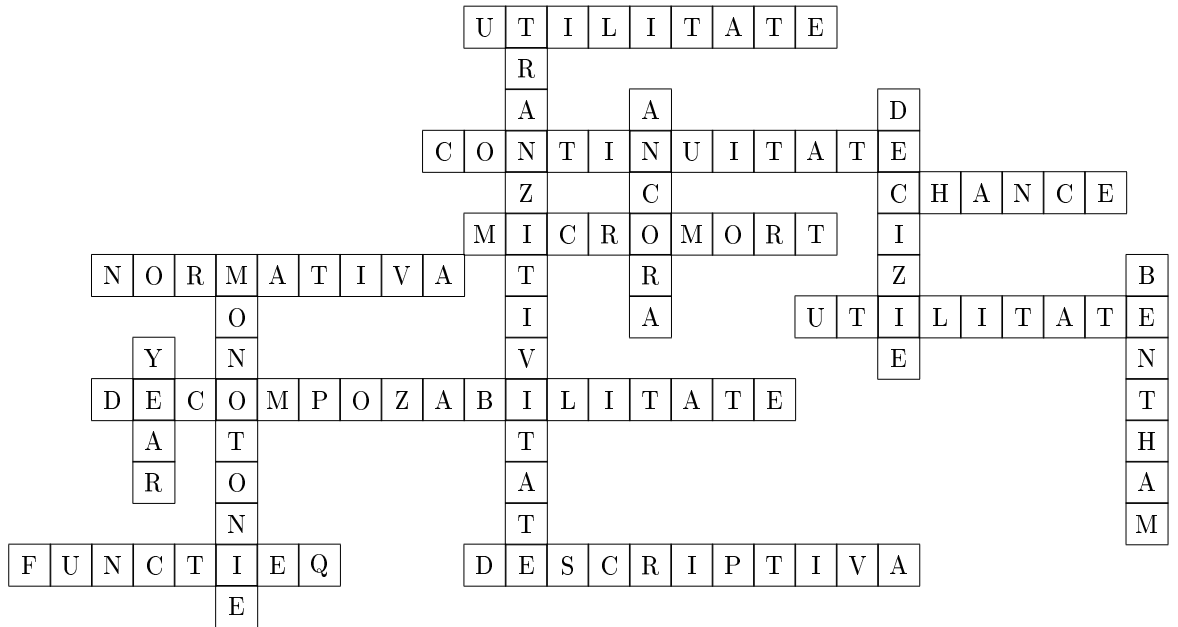
### Rețele Bayesiene



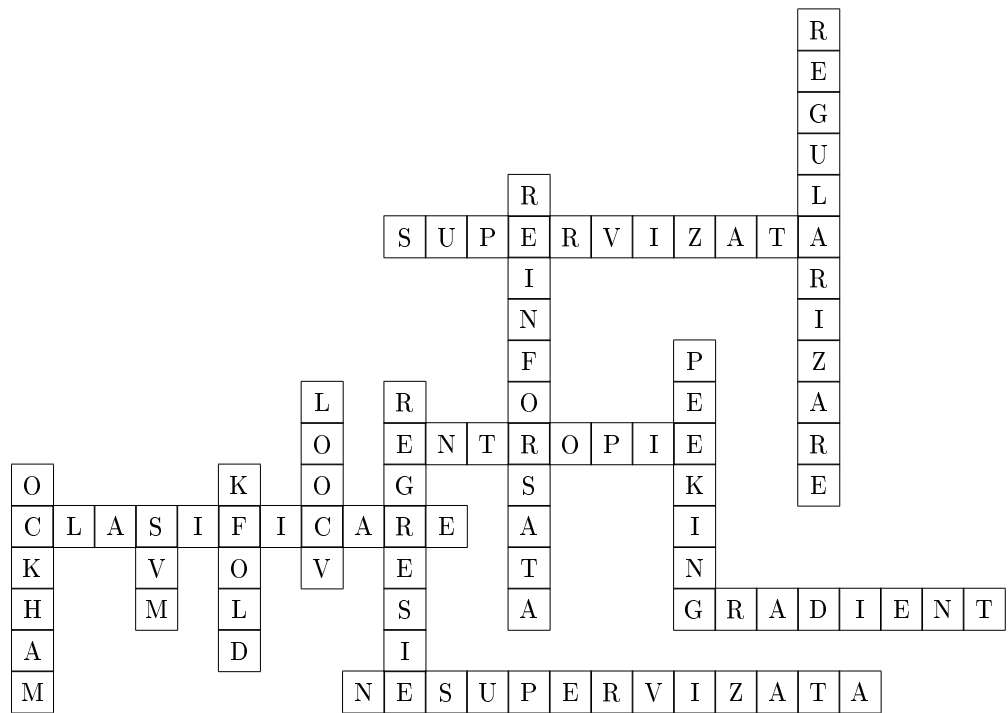
### Filtru Kalman



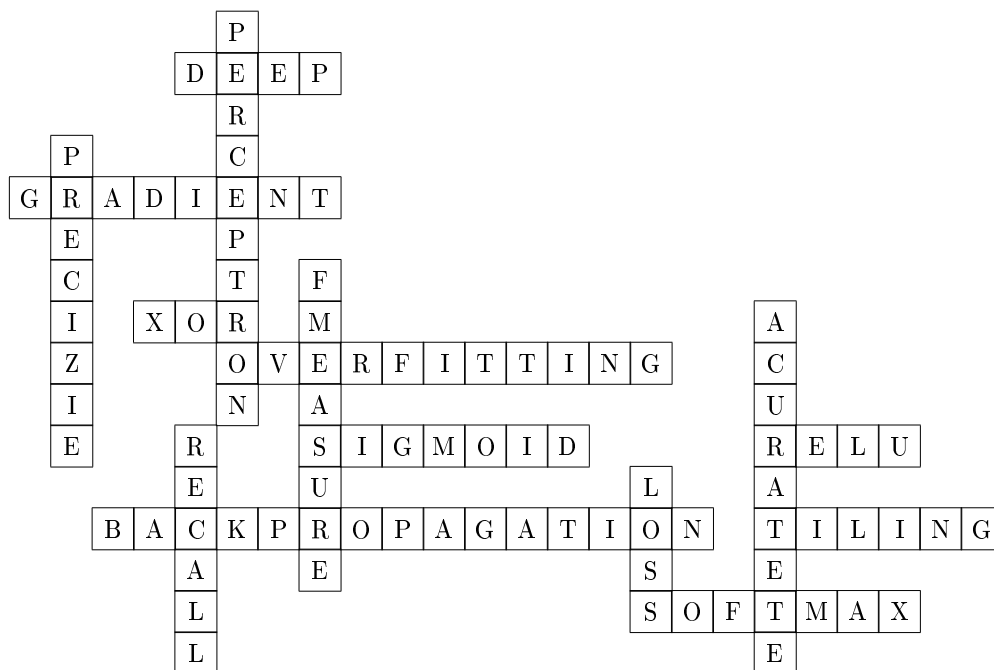
Decizii simple



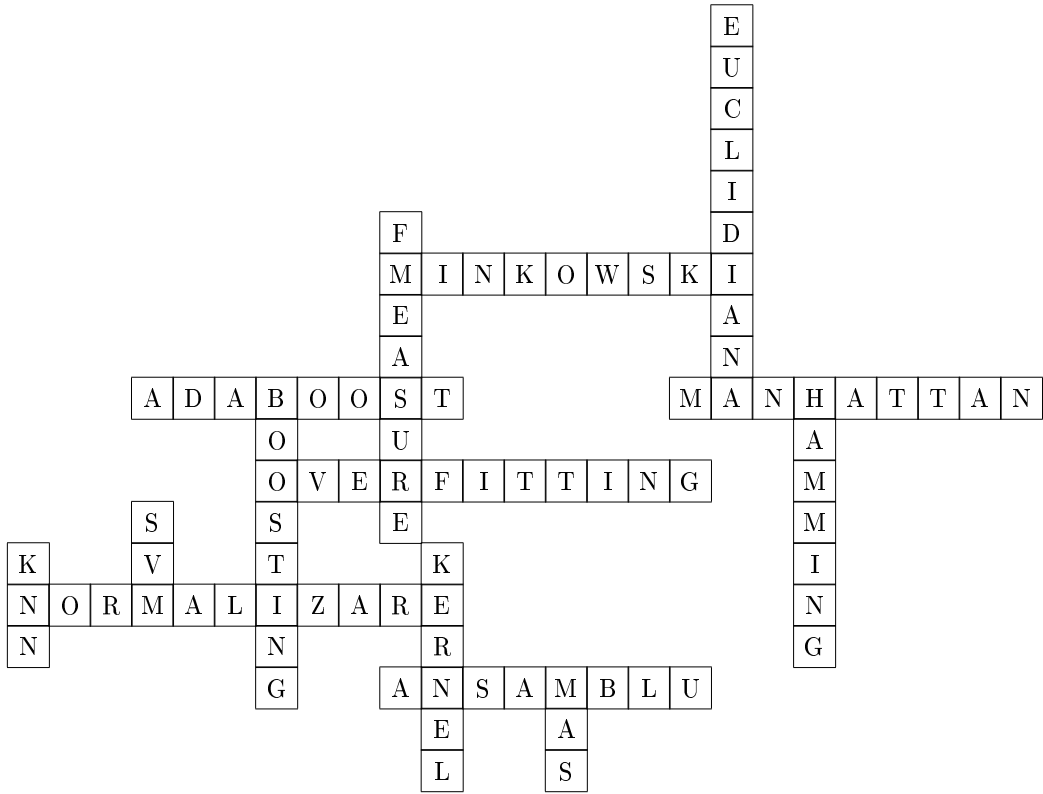
Învățare supervizată. Arbori de decizie



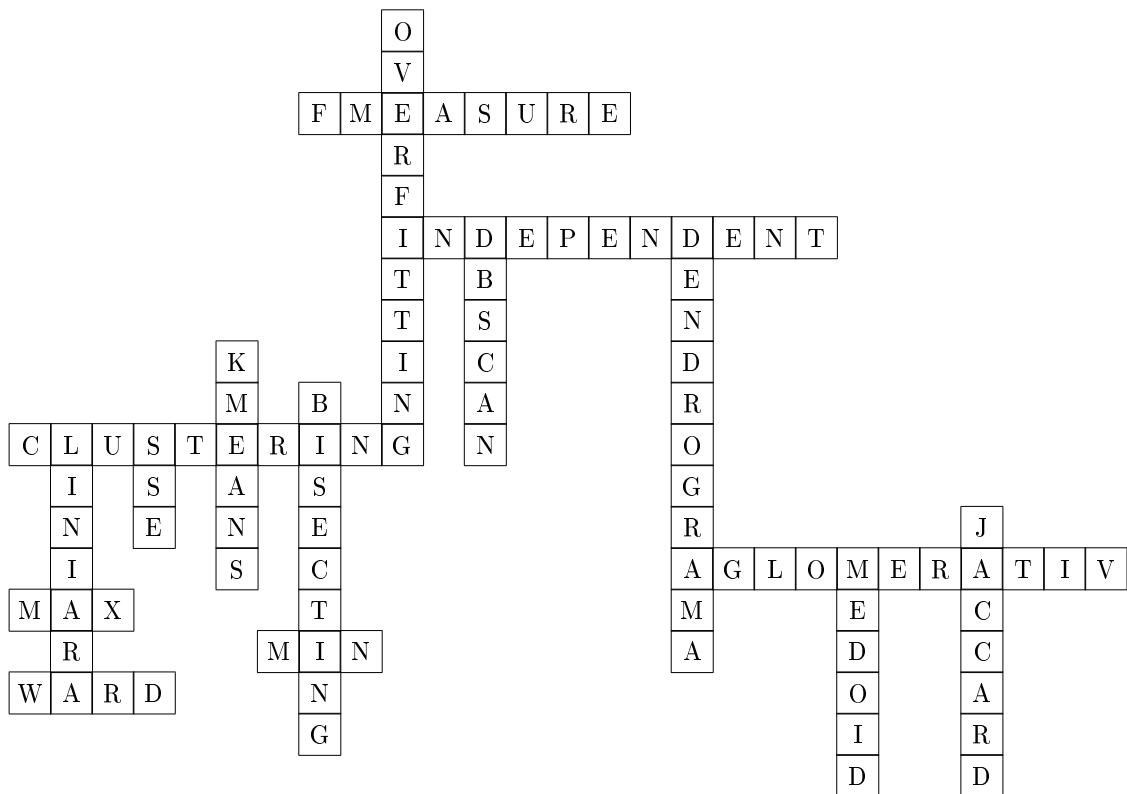
## Rețele neurale



Ansambluri de clasificatori



## Învățare nesupervizată





# Bibliografie

- [1] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson Education Limited, 2016.

