Restul

Acum, voi detalia conceptele din materialul de curs privind **Modelele Bazate pe Stări** (State-based Models) pentru problemele de decizie și **Strategiile de Căutare Neinformate** (Uninformed Search Strategies).

🧭 Modele Bazate pe Stări (State-based Models)

Această abordare este esențială pentru dezvoltarea sistemelor AI raționale.

1. Definirea Problemei și a Scopului

Un sistem AI rațional urmărește să atingă obiectivele (să rezolve instanțele) pe baza unei probleme și a scopurilor definite.

- **Problema (Problem):** O clasă generală de sarcini (ex: "Avem două numere, A și B. Care este mai mare?").
- **Instanța (Instance):** O problemă specifică cu valori concrete (ex: A=?, B=?).
- **Soluția:** Trebuie să producă răspunsul corect pentru **toate** instanțele posibile.

2. Clasificarea Problemelor (Complexitatea)

Definirea problemei necesită înțelegerea complexității sale (clasele de complexitate).

- **P (Polynomial Time):** Setul de probleme care pot fi rezolvate/verificate într-un timp polinomial de către o Mașină Turing deterministă (TM).
- **NP (Non-deterministic Polynomial Time):** Setul de probleme care pot fi rezolvate în timp polinomial de către o Mașină Turing non-deterministă.
    - **Notă:** Validarea unei soluții NP este o problemă P.
- **NP-Complete:** Submulțimea NP cu proprietatea că orice problemă din NP poate fi redusă la o problemă NP-Complet în timp polinomial.
- **NP-Hard:** Setul de probleme cu proprietatea că orice problemă din NP poate fi redusă la o problemă NP-Hard în timp polinomial. (Include probleme care pot fi rezolvate și probleme care sunt indecidabile, ex. Problema opririi lui Turing).
- **Problema P=NP?** Este o întrebare nerezolvată care ar stabili dacă rezolvarea unei probleme este la fel de rapidă ca și verificarea ei.

3. Etapele Modelării Bazate pe Stări

Pentru a rezolva o problemă (ex. găsirea unei căi într-un graf), trebuie să descriem un model urmând acești pași:

1. Descrie o stare (State) : Toate datele necesare pentru a continua căutarea unei soluții.
2. Identifică stările speciale și spațiul problemei.

3. Descrie tranzițiile (Transitions) și validează-le.
4. Specifică o strategie de căutare (Search Strategy).

🧩 Reprezentarea Stării (State Representation)

Alegerea reprezentării stării este cel mai important și dificil pas.

- **Cerințe pentru o reprezentare a stării:**
  - Să fie **fără ambiguitate** (No ambiguity).
  - Să fie suficient de **expresivă** pentru a include toate datele necesare.
  - Să fie suficient de **compactă** pentru a fi ușor de stocat și modificat.
- **Stări Speciale (Special States):**
  - **Starea Inițială (Initial State):** Punctul de plecare al căutării. Poate exista una sau mai multe stări inițiale, dar trebuie să existe cel puțin una.
  - **Starea Finală (Final State(s)):** Starea sau setul de stări care reprezintă soluția problemei. Poate exista una sau mai multe stări finale, dar trebuie să existe cel puțin una.
- **Spațiul Problemei (Problem Space):** Reprezintă toate stările posibile (valide) pe care le poate lua sistemul, incluzând starea inițială și stările finale.
- **Tranzițiile (Transitions):** Reprezintă acțiunile posibile care schimbă starea curentă.
  - **Tranziții Valide (Valid Transitions):** Mișcările trebuie să respecte constrângerile problemei (ex. pentru Turnurile Hanoi: nicio piesă mai mică nu poate fi așezată deasupra unei piese mai mari).

🔎 Strategii de Căutare (Search Strategies)

Strategiile de căutare sunt algoritmii folosiți pentru a naviga prin spațiul problemei și a găsi o soluție (o secvență de tranziții). Acestea sunt împărțite în două categorii:

A. Strategii Neinformate (Uninformed Search)

Aceste strategii nu fac distincție între stări, adică nu folosesc nicio informație despre cât de "aproape" este o stare de scop.

- **Căutarea în Lățime (Breadth First Search - BFS):**
  - Vizitează stările în ordinea distanței (numărul de tranziții) față de starea inițială.
  - Explorează toți vecinii imediați înainte de a trece la nivelul următor.
  - **Găsește cea mai scurtă cale (soluție optimă).**
  - Costisitoare din punct de vedere al memoriei, deoarece trebuie să memoreze fiecare stare generată.
- **Căutarea Uniformă a Costului (Uniform Cost Search):** O variantă a BFS. Dacă opțiunile au costuri diferite, explorează mai întâi căile mai ieftine.

- **Căutarea în Adâncime (Depth First Search - DFS):**
    - Vizitează un vecin imediat al stării curente și continuă în adâncime până găsește starea finală sau ajunge la un fund de sac.
    - Poate să nu se termine (să intre în buclă) dacă în spațiul problemei există cicluri.
- **Căutarea prin Adâncime Iterativă (Iterative Deepening Search - IDS):** O variantă a DFS care explorează doar până la o adâncime crescătoare, evitând căile infinite. Găsește soluția optimă.
- **Backtracking (BKT):**
    - Nu este același lucru cu DFS.
    - Stabilește o ordine pentru stările vecine.
    - **Nu trebuie să memoreze stările vizitate** și poate evita buclele fără a stoca stări.
    - Poate elimina ramurile invalide mai devreme (pruning) dacă ordinea este optimă.
- **Căutarea Bidirecțională (Bidirectional Search):**
    - Începe explorarea simultan din starea inițială și din starea finală.
    - Găsește cea mai scurtă cale (soluție optimă).

B. Strategii Informate (Informed Search)

Aceste strategii folosesc **euristici** pentru a distinge între stări și pentru a estima cât de aproape este o stare de scop. (Acestea vor fi probabil abordate în săptămânile următoare ale cursului).

- Greedy best-first
- Hillclimbing and Simulated Annealing
- A* and IDA*

Voi detalia conceptul de **Probleme de Satisfacere a Constrângerilor (Constraint Satisfaction Problems - CSP)**, care face parte din abordarea bazată pe **Modelele Bazate pe Variabile** [1] și este programat pentru Săptămâna 4 a cursului[2].

🧩 Probleme de Satisfacere a Constrângerilor (CSP)

CSP-urile reprezintă un tip de problemă în care soluția trebuie să satisfacă un set de condiții sau restricții (constrângeri). Ele fac parte din clasa modelelor bazate pe variabile[3].

Definirea unui CSP

O Problemă de Satisfacere a Constrângerilor este definită de trei componente principale:

1. **Variabile (V):** O mulțime finită de variabile pe care trebuie să le instanțiem (să le atribuim valori)[4].

2. **Domenii (D):** Pentru fiecare variabilă, există o mulțime finită de valori posibile pe care le poate lua (domeniul variabilei)[5].
3. **Constrângeri (C):** O mulțime finită de relații care limitează valorile pe care le pot lua simultan variabilele[6].

O **soluție** la un CSP este o atribuire consistentă de valori din domenii către variabile, astfel încât toate constrângerile sunt satisfăcute[7].

Abordarea Bazată pe Variabile

În contextul AI, CSP-urile folosesc abordarea bazată pe variabile[8]:

- **Întrebarea Centrală:** Ce formulă (funcție) poate fi aplicată datelor problemei pentru a genera obiectivul? [9]
- **Mecanism:** Startul se face de la formule arbitrare, care sunt testate pe rezultatele așteptate și ajustate[10]. CSP-urile se concentrează pe găsirea valorilor care se potrivesc într-un cadru definit de relațiile (constrângerile) dintre ele[11].

Concepte Corelate

1. **Constrângeri Soft (Soft Constraints):**
   - În CSP-urile standard, o constrângere este fie satisfăcută, fie încălcată.
   - Constrângerile Soft introduc noțiunea de **preferință** sau **cost.** Dacă nu toate constrângerile pot fi satisfăcute, scopul este de a le încălca pe cele mai puțin importante sau de a minimiza costul total al încălcărilor[12].
2. **Optimizări (Optimisations):**
   - Dacă un CSP are mai multe soluții, tehnicile de optimizare caută cea mai bună soluție în funcție de un anumit criteriu (de exemplu, soluția cu cel mai mic cost total sau soluția care maximizează o funcție de utilitate)[13].

Exemplu (Sudoku)

Un exemplu clasic de CSP este jocul Sudoku[14]:

- **Variabile (V):** Cele 81 de celule din grilă.
- **Domenii (D):** Pentru fiecare celulă, valorile posibile sunt $\{1, 2, ..., 9\}$.
- **Constrângeri (C):**
  - Toate cele 9 celule din fiecare rând trebuie să aibă valori unice.
  - Toate cele 9 celule din fiecare coloană trebuie să aibă valori unice.
  - Toate cele 9 celule din fiecare sub-grilă 3×3 trebuie să aibă valori unice.

Întrebarea Cursului

- **"Can the computer satisfy constraints over variables in a model?"** (Poate computerul să satisfacă constrângerile asupra variabilelor dintr-un model?)[15]
    - Răspunsul este da, prin algoritmi de căutare specializați, cum ar fi **backtracking**-ul cu tehnici de optimizare (cum ar fi propagarea constrângerilor și ordonarea euristică a variabilelor).

Da, bineînțeles. Următoarea secțiune din sinopsisul cursului, programată pentru **Săptămâna 5**, se axează pe **Jocuri** (Games).

🎲 Jocuri (Games)

Această secțiune analizează modul în care Inteligența Artificială abordează jocurile, în special jocurile competitive (sau adversarial games).

Concepte Principale

- **Tipuri de Jocuri (Types of Games):** Jocurile sunt clasificate în funcție de mai mulți factori, cum ar fi:
    - **Jocuri Deterministe vs. Stochastice (Probabilistice):** Jocurile deterministe (ex. Șah, Dame) nu implică elemente de șansă; starea viitoare depinde doar de starea curentă și de acțiunea aleasă. Jocurile stochastice (ex. table, Poker) implică elemente aleatorii (zaruri, cărți).
    - **Informație Perfectă vs. Imperfectă:** Jocurile cu informație perfectă (ex. Șah) permit ambilor jucători să vadă toată starea jocului. Jocurile cu informație imperfectă (ex. Poker, Black-jack) ascund o parte din informație.
    - **Jocuri Zero-Sum vs. Non-Zero-Sum:** În jocurile zero-sum, câștigul unui jucător este egal cu pierderea celuilalt (interese opuse).
- **Teoria Jocurilor (Games Theory):**
    - Este studiul strategic al interacțiunilor dintre agenții raționali.
    - Se concentrează pe prezicerea comportamentului adversarilor și pe găsirea celor mai bune decizii (strategii).
    - Conceptele cheie includ **Echilibrul Nash** (o stare stabilă în care niciun jucător nu poate beneficia de pe urma schimbării unilaterale a strategiei).
- **Strategii (Strategies):**
    - Sistemele AI folosesc algoritmi de căutare avansați pentru a explora spațiul jocului.
    - Cea mai comună metodă pentru jocurile deterministe cu informație perfectă (cum ar fi Șahul) este algoritmul **Minimax** și optimizarea sa, **Alpha-Beta Pruning**. Acestea permit AI-ului să aleagă mutarea care maximizează câștigul său, presupunând că adversarul (Min) va încerca să minimizeze acest câștig (adică, adversarul joacă optim).

Întrebarea Cursului

- **"Can the computer play games competitively?"** (Poate computerul să joace jocuri competitiv?)
    - **Răspunsul este da.** În multe domenii, AI-ul depășește performanța umană (ex. Deep Blue la șah, AlphaGo la Go). În funcție de tipul de joc, AI-ul folosește fie strategii bazate pe căutare (Minimax), fie învățare prin consolidare (Reinforcement Learning), fie o combinație a celor două.

Secțiunea următoare din sinopsisul cursului de Inteligență Artificială abordează două subiecte majore, adesea interconectate: **Rețelele Neurale (Neural Networks)** și **Învățarea prin Consolidare (Reinforcement Learning).**

🧠 Săptămâna 6: Rețele Neurale (Neural Networks)

Această secțiune introduce fundamentele **Machine Learning** din perspectiva modelelor inspirate de creierul biologic.

Concepte Principale

- **Rețele Neurale:** Sunt sisteme de calcul inspirate de rețelele neuronale biologice din creier. Sunt alcătuite din **noduri (neuroni)** organizate în straturi interconectate (Intrare, Ascuns, Ieșire)[1].
- **Perceptroni:** Reprezintă cel mai simplu tip de rețea neuronală[2]. Un perceptron este un singur nod care ia intrări, le ponderă, le însumează și aplică o funcție de activare pentru a produce o ieșire (de obicei o clasificare binară).
- **Machine Learning (Învățarea Automată):** Rețelele neurale sunt instrumente fundamentale ale Machine Learning[3]. Procesul implică antrenarea rețelei cu date pentru a ajusta ponderile (greutățile) dintre noduri, permițând astfel rețelei să învețe tipare și să facă predicții.
- **Aplicații:** Rețelele neurale au aplicații extinse în jocuri și în **Procesarea Limbajului Natural (NLP)**[4].

Întrebarea Cursului

- **"Can the computer learn anything?"** (Poate computerul să învețe ceva?)[5]
    - **Răspunsul este da**, prin ajustarea iterativă a parametrilor (ponderilor) modelului pe baza datelor, permițându-i să generalizeze tiparele și să facă predicții.

Shutterstock

🔁 Săptămânile 6-7-10: Învățarea prin Consolidare (Reinforcement Learning - RL)

RL este o paradigmă de învățare bazată pe modul în care un agent învață să acționeze într-un mediu pentru a maximiza o recompensă cumulativă[6].

Concepte Principale

- **Agent, Mediu, Acțiune, Recompensă:** Un agent[7] ia o **Acțiune** în **Mediu**, trece într-o nouă **Stare** și primește o **Recompensă** sau o penalizare. Scopul este de a găsi o **Politică** optimă care să maximizeze recompensa totală pe termen lung.
- **Proces Markov de Decizie (Markov Decision Process - MDP):** Este cadrul matematic folosit pentru modelarea RL[8]. Un MDP este definit de stări, acțiuni, probabilități de tranziție (proba de a ajunge într-o nouă stare după o acțiune) și funcții de recompensă. Proprietatea Markov stipulează că următoarea stare depinde doar de starea și acțiunea curentă, nu de istoricul precedent.
- **Q-Learning:** Este un algoritm popular de RL fără model (model-free)[9].
    - **Mecanism:** Agentul învață o **Funcție Q (Quality Function)**, notată $Q(\text{stare}, \text{acțiune})$, care estimează valoarea (recompensa așteptată) a efectuării unei anumite **acțiuni** într-o anumită **stare**.
    - **Scop:** Prin explorare și exploatare, agentul actualizează valorile Q utilizând ecuația Bellman, învățând astfel ce acțiune este optimă în fiecare stare.

rmătoarea secțiune din sinopsisul cursului, programată pentru **Săptămânile 9-11**, se concentrează pe **Reprezentarea Cunoștințelor** (Knowledge Representation) și **Procesarea Limbajului Natural (NLP)**.

🧠 Reprezentarea Cunoștințelor (Knowledge Representation)

Acest domeniu se ocupă de modul în care informațiile din lumea reală sunt structurate și stocate în interiorul unui sistem AI pentru a permite raționamentul.

- **Ontologii (Ontologies):** Sunt structuri formale care definesc tipurile, proprietățile și relațiile dintre concepte dintr-un anumit domeniu.
    - **Scop:** Oferă o bază de cunoștințe explicită, care permite computerului să înțeleagă semantica și să facă inferențe logice.
    - **Structură:** O ontologie tipică folosește noduri (concepte/entități) și arce etichetate (relații, ex. "is a" sau "has") pentru a lega entitățile (de exemplu, o entitate Model `has` Model Data, iar Model Data `is` Discharge).

🗣️ Procesarea Limbajului Natural (Natural Language Processing - NLP)

NLP este subdomeniul AI care se concentrează pe interacțiunea dintre computere și limbajul uman (natural).

- **Scop:** Permite computerului să comunice cu noi folosind propria noastră limbă.
- **Înțelegerea Limbajului Natural (Understanding Natural Language):** Reprezintă capacitatea sistemului AI de a extrage sensul și intenția din textul sau vorbirea umană.
- **Ambiguitatea Limbajului (Language Ambiguity):** Este una dintre cele mai mari provocări în NLP. Cuvintele sau frazele pot avea sensuri multiple în funcție de context, sintaxă sau semantică (ex. "time flies like an arrow"). Computerul trebuie să rezolve această ambiguitate pentru a înțelege corect mesajul.

Întrebarea Cursului

- **"Can the computer talk to us using our language?"** (Poate computerul să vorbească cu noi folosind limba noastră?)
    - **Răspunsul este da**, iar progresele în reprezentarea cunoștințelor și NLP, în special prin modele de limbaj mari (LLM), au îmbunătățit semnificativ această capacitate.

Perfect. Următoarea secțiune din sinopsisul cursului de Inteligență Artificială, programată pentru **Săptămâna 12**, este dedicată **Rețelelor Bayes** (Bayesian Networks) și raționamentului cu incertitudine.

🎲 Săptămâna 12: Rețele Bayes (Bayesian Networks)

Rețelele Bayes oferă un cadru formal pentru a raționa sub incertitudine, combinând teoria probabilităților cu conceptele de grafuri.

Concepte Principale

1. **Raționament cu Probabilități (Reasoning with Probabilities):**
    - Rețelele Bayes sunt **modele grafice probabilistice** care reprezintă un set de variabile și relațiile de **independență condiționată** dintre ele[1].
    - Ele permit unui sistem AI să facă **inferențe probabilistice** (adică, să calculeze probabilitatea unui eveniment având în vedere apariția altor evenimente)[2].
2. **Independență și Independență Condiționată (Independence and Conditional Independence):**

    - **Independență:** Două evenimente (A și B) sunt independente dacă apariția unuia nu afectează probabilitatea apariției celuilalt ($P(A|B) = P(A)$)[3].
    - **Independență Condiționată:** Două evenimente (A și B) pot deveni independente dacă se cunoaște starea unui al treilea eveniment (C). Această proprietate este crucială deoarece permite simplificarea modelului (nu trebuie să calculăm toate probabilitățile posibile)[4].
3. **Structura Rețelei (Complete Bayesian Network):**
    - Rețeaua este un **Graf Acyclic Dirijat (DAG).**
    - **Nodurile** reprezintă variabilele aleatoare (evenimentele)[5].
    - **Arcele (săgețile)** reprezintă relațiile de influență directă (cauzalitate)[6].
    - Fiecărui nod i se asociază o **Tabelă de Probabilități Condiționate (Conditional Probability Table - CPT)**, care specifică probabilitatea ca nodul să ia o anumită valoare, având în vedere valorile părinților săi (nodurile care indică spre el)[7].

Exemplu (Rețeaua Burglar Alarm)

Un exemplu clasic de rețea Bayesiană este modelul unei alarme:

- **Variabile: Burglary** (Efracție), **Earthquake** (Cutremur), **Alarm** (Alarma se declanșează), **JohnCalls** (John sună), **MaryCalls** (Mary sună)[888888888888888888888888].
- **Relații (Cauzalitate):** Efracția și Cutremurul pot provoca declanșarea Alarmei. Alarma declanșată îl face pe John să sune și pe Mary să sune[999999999999999999999999].
- **Inferență:** Rețeaua permite calculul probabilităților, de exemplu, care este probabilitatea să fi fost Efracție, având în vedere că John a sunat și Mary a sunat, dar știm că nu a fost Cutremur.

Întrebarea Cursului

- **"Can the computer decide and learn on probabilistic data?"** (Poate computerul să decidă și să învețe pe date probabilistice?)
    - **Răspunsul este da**, iar Rețelele Bayes sunt unul dintre instrumentele cheie pentru a gestiona incertitudinea și a lua decizii informate în medii complexe și zgomotoase.

Ultima secțiune din sinopsisul cursului, programată pentru **Săptămâna 13**, este dedicată **Planificării (Planning).**

🗓️ Săptămâna 13: Planificare (Planning)

Planificarea în Inteligența Artificială se referă la procesul de alegere și organizare a unei secvențe de acțiuni pentru a atinge un obiectiv.

Concepte Principale

1. **STRIPS și PDDL:** Acestea sunt limbaje formale utilizate pentru a descrie problemele de planificare.
    - **STRIPS (STandard Representation of an AI Planning System):** Unul dintre primele și cele mai influente limbaje de reprezentare, care definește stările lumii, obiectivele și acțiunile.
    - **PDDL (Planning Domain Definition Language):** Un limbaj de domeniu standardizat, modern, folosit pentru a descrie domenii de planificare, inclusiv stările inițiale, efectele acțiunilor și obiectivele.
    - **Acțiunile (Actions):** Sunt definite prin:
        - **Precondiții:** Condițiile care trebuie să fie adevărate înainte ca acțiunea să poată fi executată.
        - **Efecte (Add/Delete Lists):** Modificările pe care acțiunea le aduce stării lumii (ce se adaugă sau se șterge din descrierea stării).
2. **Căutare Înainte și Căutare Înapoi (Forward and Backward Search):** Acestea sunt cele două strategii principale de a găsi o secvență de acțiuni (un plan).
    - **Căutare Înainte (Forward Search / State-Space Search):**
        - Începe din **starea inițială** și explorează acțiunile posibile (tranzițiile), mergând către starea scop.
        - Este similară cu căutarea în spațiul stărilor (ca BFS sau DFS), dar este ghidată de obiective.
    - **Căutare Înapoi (Backward Search / Goal-Directed Search):**
        - Începe de la **starea finală (scop)** și lucrează în sens invers, identificând precondițiile necesare pentru a ajunge la acea stare.
        - Procesul continuă înapoi până când se ajunge la o stare care este satisfăcută de starea inițială.

Întrebarea Cursului

- **"Can the computer find a plan which is guaranteed to succeed?"** (Poate computerul să găsească un plan care să fie garantat că va reușI?)
    - **Răspunsul este da**, cu condiția ca:
        1. Problema să fie definită într-un domeniu determinist (fără incertitudine).
        2. Modelul lumii să fie complet și corect.
        3. Algoritmul de planificare să exploreze spațiul de planificare în mod exhaustiv și eficient.

Am parcurs acum toate secțiunile principale din sinopsisul cursului de Inteligență Artificială. Iată o sinteză completă a tuturor conceptelor detaliate, organizată pe secțiuni:

🤖 Definirea Inteligenței Artificiale (AI)

Definirea AI se face din patru perspective majore, clasificate după proces și fidelitate (uman sau rațional):

- **Acționând Uman (Acting Humanly):** Concentrat pe comportament. Scopul este de a trece **Testul Turing**.
- **Gândind Uman (Thinking Humanly):** Concentrat pe procesul cognitiv. Încearcă să replice mecanismele biologice de gândire.
- **Gândind Rațional (Thinking Rationally):** Bazat pe logică. Folosește un sistem deductiv acceptat (reguli logice).
- **Acționând Rațional (Acting Rationally):** Bazat pe rezultate. Computerul produce rezultate raționale, funcționând ca un **agent rațional**.

⚙️ Pilonii AI și Abordări (Modele de Decizie)

Pilonii AI Moderne

1. **Modelare (Modeling):** Descrierea și lucrul cu datele din lumea reală (Structuri de Date, Baze de Date, Algoritmi de Grafuri, Sisteme Bazate pe Cunoștințe).
2. **Raționament (Reasoning):** Descoperirea de date noi (concepte, relații) din datele disponibile (Logică, Algoritmi Design, Algoritmi de Grafuri).
3. **Învățare (Learning):** Adaptarea modelului pentru contexte particulare (Probabilități și Statistică, Machine Learning, Rețele Neurale).

Abordări (Sisteme de Decizie)

- **Modele Bazate pe Stări (State-based Models):** Calculează o soluție ca o secvență de tranziții de la o stare inițială la o stare scop. Acoperă strategii de căutare și AI pentru jocuri.
- **Modele Bazate pe Variabile (Variable-based Models):** Caută o formulă/funcție care se aplică datelor pentru a obține obiectivul. Acoperă Machine Learning și CSP.

🔎 Strategii de Căutare Neinformate (Uninformed Search)

Strategiile de căutare nu fac distincție între stări, bazându-se pe ordinea de explorare:

- **Breadth First Search (BFS):** Vizitează stările în ordinea distanței față de starea inițială. Găsește cea mai scurtă cale (**optimă**).
- **Uniform Cost Search:** Explorează mai întâi căile mai ieftine. Găsește cea mai scurtă cale (**optimă**).
- **Depth First Search (DFS):** Vizitează un singur vecin imediat și continuă în adâncime. Poate să nu găsească soluția optimă și poate intra în bucle.

- **Iterative Deepening Search (IDS):** Explorează până la o adâncime crescătoare, evitând căile infinite. Găsește soluția optimă.
- **Backtracking (BKT):** Stabilește o ordine a vecinilor și nu trebuie să memoreze stările vizitate, putând evita buclele.
- **Bidirectional Search:** Explorează simultan de la starea inițială și de la starea finală. Găsește cea mai scurtă cale (**optimă**).

🧩 Sinopsisul Cursului

Săptămâna 4: Probleme de Satisfacere a Constrângerilor (CSP)

- **Definiție:** O problemă de alocare de valori variabilelor, sub un set de **constrângeri** (condiții).
- **Concepte:** Constrângeri Soft (costul încălcării), Optimizări.

Săptămâna 5: Jocuri (Games)

- **Concepte:** Tipuri de jocuri (deterministe, stochastice, perfect/imperfectă informație), **Teoria Jocurilor** (studiul interacțiunilor strategice), Strategii (ex. Minimax).

Săptămâna 6: Rețele Neurale (Neural Networks)

- **Concepte: Perceptroni** (celule de bază), Machine Learning, aplicații în jocuri și NLP.

Săptămânile 6, 7, 10: Învățarea prin Consolidare (Reinforcement Learning)

- **Concepte: Proces Markov de Decizie (MDP)** (cadrul matematic), **Q-Learning** (algoritm pentru a învăța valoarea acțiunilor în stări).

Săptămânile 9-11: Reprezentarea Cunoștințelor și NLP

- **Concepte: Ontologii** (structuri formale pentru cunoaștere), **Înțelegerea Limbajului Natural**, **Ambiguitatea Limbajului** (o provocare majoră).

Săptămâna 12: Rețele Bayes (Bayesian Networks)

- **Concepte: Raționament cu probabilități**, **Independență și independență condiționată** (simplifică inferența în prezența incertitudinii).

Săptămâna 13: Planificare (Planning)

- **Concepte: STRIPS și PDDL** (limbaje de definire a problemelor de planificare), **Căutare Înainte și Înapoi** (strategii de a găsi secvența de acțiuni).

Așadar, doriți să detaliez conceptele conexe din materialul de curs privind definirea și filozofia Inteligenței Artificiale. Iată o analiză aprofundată a acestor subiecte.

💡 Concepte Filozofice și Deosebiri

1. Strong AI (AI Puternică) vs. Weak AI (AI Slabă)

Această distincție se referă la obiectivul final al cercetării în AI și la natura inteligenței pe care o poate poseda o mașină[1].

- **Weak AI (AI Slabă):**
  - **Premisă:** Mașinile pot fi programate pentru a simula comportamentul inteligent și a rezolva probleme complexe, dar nu posedă cu adevărat o minte, o conștientizare, o conștiință sau o înțelegere reală, așa cum o are un om[2].
  - **Natura:** Este doar un instrument puternic pentru studierea minții (de exemplu, o simulare).
  - **Exemple:** Toate sistemele AI actuale, inclusiv ChatGPT, sunt considerate Weak AI[3].
- **Strong AI (AI Puternică):**
  - **Premisă:** O mașină care rulează un program inteligent nu doar simulează o minte, ci este *de facto* o minte, capabilă de gândire, conștiință și înțelegere în sensul uman al termenului[4].
  - **Natura:** O entitate cognitivă reală.

2. Testul Turing (Turing Test)

Testul Turing este un criteriu de evaluare propus de Alan Turing (1950) pentru a determina dacă o mașină poate manifesta un comportament inteligent, care să nu poată fi distins de cel al unui om[5].

- **Procedură:** Un om (interogatorul) poartă conversații scrise (prin terminale) cu două entități: un alt om și o mașină.
- **Obiectiv:** Întrebarea este dacă un om obișnuit poate distinge între om și computer pe baza răspunsurilor primite[6].
- **Critici:** Deși este o piatră de temelie a filozofiei AI, testul a fost criticat pentru că se concentrează doar pe **comportament** (acționând uman) și ignoră **procesul intern** (dacă mașina înțelege cu adevărat).

3. Argumentul Camerei Chineze (Chinese Room)

Acesta este un experiment mental propus de John Searle (1980) ca un contra-argument direct împotriva Strong AI[7].

- **Premisă:** O persoană care nu vorbește chineza este închisă într-o "cameră chineză" și primește simboluri (input) în chineză. Persoana folosește un set extins de reguli

(un "rule ledger" sau un program) pentru a manipula simbolurile și a trimite simboluri (output) înapoi, care sunt de fapt răspunsuri coerente în chineză[8888888].

- **Întrebarea:** Simplul fapt de a folosi reguli este echivalent cu înțelegerea?[9].
- **Concluzie (a lui Searle):** Deși persoana din cameră (sistemul AI) se comportă ca și cum ar înțelege chineza, ea doar manipulează simboluri fără a avea o înțelegere semantică reală. Prin urmare, simpla rulare a unui program nu este suficientă pentru a atinge Strong AI.

4. Connectionism vs. Computationalism

Acestea sunt două viziuni concurente asupra modului în care ar trebui realizată inteligența[10].

Criteriu	**Connectionism**	**Computationalism**
Natura Inteligenței	Inteligența este o **consecință** (a structurii)[13].	Inteligența este un **scop** (al funcțiilor)[1414].
Bază	Inteligența este un produs al **structurii**[15].	Inteligența este un produs al **funcțiilor**[16].
Model	Inspirat de creier (rețele neurale, conexiuni ponderate).	Inspirat de logică și programe (manipularea simbolurilor).
Întrebarea	Este comportamentul inteligent o consecință sau un scop?[17].	Este comportamentul inteligent o consecință sau un scop?[18].