**Universitatea Tehnică “Gheorghe Asachi” din Iași Facultatea de Automatică și Calculatoare Domeniul Calculatoare și Tehnologia Informației Specializarea Tehnologia Informației**

**INTELIGENȚĂ ARTIFICIALĂ**

**"Connect 4"   
  
Utilizarea algoritmului minimax cu retezarea alfa-beta  
în dezvoltarea unui joc om vs calculator**

**Coordonator,**

**Asist. drd. ing. Codruţ-Georgian Artene**

**Cîrjă Ioan, 1409B**

**Studenți,**

**Cazamir Andrei, 1409B**

**An universitar 2024-2025**

**Cuprins**

**Descrierea problemei considerate** **3**

**Aspecte teoretice privind algoritmul 3**

**Algoritmul minimax 3**

**Retezarea alfa-beta 3**

**Modalitatea de rezolvare 4**

**Exemple semnificative din implementare 6**

**Funcția de evaluare 6**

**Calcularea următoarei poziții a calculatorului 7**

**Algoritmul minimax cu retezare alfa-beta 8**

**Rezultatele obținute prin rularea programului 9**

**Concluzii 13**

**Bibliografie 13**

**Rolul fiecărui membru al echipei 14**

1. **Descrierea problemei considerate**

Connect Four este un joc strategic pentru doi jucători, în care fiecare participant selectează o culoare și plasează alternativ discuri într-o grilă verticală de o dimensiune specificată. Obiectivul este de a forma o linie de patru discuri de aceeași culoare, dispuse pe verticală, orizontală sau diagonală, înaintea adversarului.  
  
 În cadrul acestui proiect, dorim să implementăm o versiune om vs calculator, în care inteligența deciziilor calculatorului va fi bazată pe algoritmul minimax optimizat prin **retezarea alfa-beta.**

1. **Aspecte teoretice privind algoritmul**
   1. **Algoritmul minimax**

Algoritmului minimax este un algoritm recursiv pentru a găsi cea mai bună mișcare într-o situație dată. Algoritmul minimax constă dintr-o [funcție de](https://koaha.org/wiki/Funzione_di_valutazione) [evaluare pozițională](https://koaha.org/wiki/Funzione_di_valutazione) care măsoară bunătatea unei poziții (sau a stării de joc) și indică cât de dorit este ca jucătorul dat să atingă acea poziție; jucătorul face apoi mișcarea care minimizează valoarea celei mai bune poziții atinse de celălalt jucător.

Cei doi jucători sunt numiți maximizant și minimizant. Maximizantul încearcă să obțină cel mai mare scor posibil, în timp ce minimizantul încearcă să obțină cel mai mic scor posibil. Dacă asociem fiecărei table de joc un scor de evaluare, atunci unul din jucători încearcă să aleagă o mutare care să îi maximizeze scorul, iar celălalt alege o mutare care are un scor minim, încercând să contra-atace.

Vom considera jocul ca fiind de sumă nulă, ceea ce înseamnă că aceeași funcție de evaluare poate fi aplicată ambilor jucători.

* Dacă *f(n)>0*, poziția *n* este avantajoasă pentru calculator și nefavorabilă pentru om.
* Dacă *f(n)<0*, poziția *n* este dezavantajoasă pentru calculator și favorabilă pentru om.

Astfel, se calculează funcția de evaluare pentru frunze și se propagă evaluarea în sus, selectând minimele pe nivelul minimizant (decizia omului) și maximele pe nivelul maximizant (decizia calculatorului).

* 1. **Retezarea alfa-beta**

Algoritmul de tăiere alfa-beta îmbunătățește Minimax eliminând din analiză subarborii inutili. Dacă o mutare *m* este mai slabă decât cea mai bună mutare curentă, restul variantelor sale nu mai sunt evaluate. Parametrul alfa reprezintă cea mai bună valoare garantată pentru maximizator, iar beta pentru minimizator. Pe măsură ce arborele este parcurs, alfa și beta se actualizează, iar ramurile care nu pot îmbunătăți rezultatul sunt ignorate, economisind timp și resurse

1. **Modalitatea de rezolvare**

Acest proiect implementează algoritmul minimax pentru un joc între un jucător uman și un computer. Jucătorul uman maximizează scorul, iar computerul îl minimizează. Scorurile fiecărei poziții sunt ajustate în funcție de starea curentă a jocului pentru ca algoritmul să funcționeze eficient. Se va prezenta o porțiune din logica algoritmului în cele ce urmează.

Definim următoarele variabile:

* alfa: este cea mai bună valoare pe care maximizantul o poate garanta la nivelul curent sau la nivelul următor;
* beta: este cea mai bună valoare pe care minimizantul o poate garanta la nivelul curent sau la nivelul următor.

A diagram of a tree

Description automatically generated

*Arborele inițial*

În continuare vor fi prezentați o serie de pași pe care algoritmul îi va efectua în calcularea valorilor nodurilor E, D, B:

1. **Apelul începe**: La A, alfa = -INF și beta = +INF.
2. **Maximizatorul alege B**: Maximizatorul alege nodul B.
3. **B alege D**: B explorează nodul D.
4. **Evaluarea lui D**: D verifică copilul drept (valoare 3). Alfa devine 3. Se continuă căutarea (beta > alfa).
5. **Evaluarea altui copil al lui D**: D verifică copilul drept (valoare 5). Alfa devine 5 și returnează 5 la B.
6. **Actualizarea lui beta la B**: Beta devine 5 la B.
7. **B explorează E**: B verifică nodul E (alfa = -INF, beta = 5).
8. **Evaluarea lui E**: E verifică copilul stâng (valoare 6). Alfa devine 6. Condiția beta <= alfa se îndeplinește, căutarea în E este oprită.
9. **Returnarea valorii de la E**: E returnează 6 la B.
10. **Actualizarea valorii lui beta la B**: Beta rămâne 5. Valoarea finală a lui B este 5.

A diagram of a tree

Description automatically generated

*Arborele după calcularea valorilor pentru nodurile E, D și B*

În continuare se respectă aceeași pași pentru calcularea valorilor pentru fiecare nod până se ajunge la următoarea structură arborescentă.

A diagram of a diagram

Description automatically generated

*Arborele final*

1. **Exemple semnificative de implementare**
   1. **Funcția de evaluare**

A screenshot of a computer program

Description automatically generated

* 1. **Calcularea următoarei poziții a calculatorului**

A screenshot of a computer program

Description automatically generated

* 1. A screen shot of a computer program

     Description automatically generated**Algoritmul minimax cu retezare alfa-beta**

1. **Rezultatele obținute prin rularea programului  
     
   A screenshot of a game

   Description automatically generated**

*Începerea unui joc nou (calculatorul – roșu a făcut prima mutare)*

A screenshot of a game

Description automatically generated

*Calculatorul a câștigat*

A screenshot of a game

Description automatically generated

*Player-ul a câștigat*

A screenshot of a phone

Description automatically generated

*Meniu principal*

1. **Concluzii**

Algoritmul minimax este folosit în mod frecvent în jocurile de tip sumă zero pentru a determina strategia optimă, explorând toate posibilele soluții ale problemei. Cu toate acestea, această abordare este ineficientă în ceea ce privește timpul de procesare și utilizarea memoriei, deoarece adesea sunt examinate stări de joc care nu aduc niciun beneficiu.

Pentru a îmbunătăți acest proces, a fost introdus algoritmul de retezare alfa-beta, care adaugă o optimizare semnificativă. În scenariul ideal, atunci când mutările cele mai bune sunt analizate întâi, algoritmul poate elimina rapid opțiunile care nu sunt relevante. Totuși, în cel mai nefavorabil caz, atunci când evaluarea celor mai bune mutări are loc la sfârșit, retezarea alfa-beta nu aduce nicio îmbunătățire față de complexitatea algoritmului minimax standard.

1. **Bibliografie**

[1] <https://www.studocu.com/row/document/gift-university/artificial-intellegence/assignment-3-implementing-the-minimax-algorithm-with-alpha-beta-pruning-tree-visualization/95890524>

[2] <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/>

[3] <https://www.youtube.com/watch?v=xBXHtz4Gbdo>

[4] Inteligență artificială – Laborator 7, Curs 3

[5] <https://www.pygame.org/news>

[6] https://www.youtube.com/watch?v=y9VG3Pztok8

Github Repository : https://github.com/IoanCirja/MinMax-Algorithm

1. **Rolul fiecărui membru al echipei**

Cazamir Andrei

* creare prototip interfață grafică
* adăugare meniu principal interfață
* perfecționare algoritm minimax (bug fix)
* optimizare game loop
* selectare număr de nivele algoritm

Cîrjă Ioan

* creare prototip efectuare mutări de către player/calculator
* evaluarea tablei pentru identificarea câștigătorului
* implementare minimax recursiv (variantă inițială)
* optimizare funcție de evaluare
* documentație