

Algoritmul Minimax cu retezare alfa-beta aplicat jocului Connect 4

Nume studenți: Ciobanu Andrei & Paladi Andrei

Facultatea de Automatică și Calculatoare Iași

Anul IV, Grupa 1406B

Cuprins

1 Descrierea problemei	1
2 Aspecte teoretice	1
2.1 Algoritmul Minimax	1
2.2 Retezarea alfa-beta	1
3 Modalitatea de rezolvare	2
4 Funcția de evaluare	2
5 Fragmente relevante de cod	2
6 Arhitectura aplicației	4
7 Interfața aplicației	4
7.1 Meniul principal	4
7.2 Setarea adâncimii de căutare	5
7.3 Interfața jocului propriu-zis	5
7.4 Afisarea rezultatului final	6
8 Rezultate obținute	6
9 Concluzii	7
10 Bibliografie	8
11 Contribuția membrilor echipei	8

1 Descrierea problemei

Acest proiect explorează crearea unui adversar controlat de calculator pentru jocul Connect 4, folosind algoritmul **Minimax cu retezare alfa-beta**. Deși regulile sunt intuitive (alinierea a patru piese), numărul mare de mutări posibile transformă fiecare partidă într-o provocare de logică și anticipare.

Programul simulează un mod de gândire strategic: analizează mutările utilizatorului și încearcă să prevadă cele mai bune răspunsuri. Dificultatea jocului poate fi ajustată prin modificarea **adâncimii de căutare**, oferind un echilibru între rapiditatea răspunsului și „inteligenta” acestuia. Proiectul are scopul de a demonstra cum pot fi aplicate algoritmii de decizie într-un mod interactiv și ușor de înțeles.

2 Aspecte teoretice

2.1 Algoritmul Minimax

Algoritmul Minimax presupune existența a doi jucători care au obiective opuse: MAX (calculatorul) și MIN (utilizatorul). Jucătorul MAX încearcă să obțină un scor cât mai mare, alegând mutările care îi oferă cel mai mare avantaj, în timp ce jucătorul MIN încearcă să reducă acest avantaj, alegând mutări care duc la un scor cât mai mic pentru calculator.

Pentru a lua o decizie, algoritmul construiește un arbore de joc care conține toate mutările posibile pornind de la starea curentă. Fiecare nivel al arborelui corespunde unei mutări realizate de unul dintre jucători. Explorarea arborelui se face până la o anumită adâncime stabilită, iar la acest nivel stările sunt evaluate cu ajutorul unei funcții de evaluare. Pe baza acestor evaluări, algoritmul poate determina mutarea care oferă cea mai bună strategie pentru calculator.

2.2 Retezarea alfa-beta

Retezarea alfa-beta reprezintă o optimizare a algoritmului Minimax, având ca scop reducerea numărului de stări analizate în arborele de joc. Această tehnică face posibilă obținerea acelorași rezultate ca Minimax-ul clasic, dar într-un timp de execuție mai mic.

Parametrul alfa reprezintă cea mai bună valoare obținută până la un anumit moment pentru jucătorul MAX, în timp ce parametrul beta reprezintă cea mai bună valoare obținută pentru jucătorul MIN. Pe măsură ce arborele de joc este parcurs, aceste valori sunt actualizate în funcție de mutările analizate.

În situația în care valoarea alfa devine mai mare sau egală cu valoarea beta, explorarea ramurii curente poate fi întreruptă. Acest lucru se întâmplă deoarece jucătorul advers nu ar alege niciodată o mutare care să conducă la un rezultat mai nefavorabil pentru el. Astfel, ramura respectivă este eliminată din căutare, fără a afecta decizia finală a algoritmului.

3 Modalitatea de rezolvare

Implementarea soluției a fost realizată în limbajul C#, urmând pașii:

- definirea tablei de joc sub forma unei matrice;
- implementarea regulilor jocului Connect 4;
- detectarea stărilor terminale;
- definirea funcției de evaluare;
- implementarea algoritmului Minimax cu retezare alfa-beta.

4 Funcția de evaluare

Funcția de evaluare are rolul de a atribui un scor numeric fiecărei stări intermediare a jocului, atunci când algoritmul Minimax nu ajunge la o stare finală. Acest scor reflectă cât de avantajoasă este poziția curentă pentru calculator în comparație cu adversarul.

În cadrul acestui proiect, evaluarea se face ținând cont de mai mulți factori, precum pozițiile favorabile ale pieselor calculatorului, potențialele amenințări create de adversar și controlul zonelor centrale ale tablei de joc. Pozițiile care oferă mai multe posibilități de extindere a alinierilor sunt evaluate pozitiv, în timp ce pozițiile favorabile adversarului sunt penalizate.

Prin utilizarea acestei funcții de evaluare, algoritmul poate face diferență între stări aparent similare și poate alege mutări care oferă un avantaj strategic pe termen mediu, chiar și în situațiile în care nu este posibilă determinarea imediată a unui câștig sau a unei pierderi.

5 Fragmente relevante de cod

Funcția MiniMax implementează algoritmul MiniMax cu retezare alfa-beta și este responsabilă de alegerea mutării optime pentru calculator. Algoritmul explorează recursiv toate mutările valide până la o adâncime prestabilită sau până la atingerea unei stări terminale a jocului.

În cazul în care este rândul calculatorului, funcția încearcă să maximizeze scorul obținut, iar în cazul utilizatorului încearcă să minimizeze acest scor. Pentru fiecare mutare posibilă, se simulează evoluția jocului, se evaluează poziția rezultată și se păstrează cea mai bună valoare.

Retezarea alfa-beta este utilizată pentru a elimina ramurile care nu pot influența decizia finală, reducând astfel semnificativ timpul de execuție fără a afecta corectitudinea rezultatului.

```

1  public int MiniMax(int depth, int alpha, int beta, bool maximizing,
2    int[,] board)
3  {
4    if (depth == 0 || CheckWin(2, board) || CheckWin(1, board) ||
5      GetValidMoves(board).Count == 0)
6      return EvaluateBoard(board);
7
8    if (maximizing)
9    {
10      int maxEval = int.MinValue;
11      foreach (int col in GetValidMoves(board))
12      {
13        MakeMove(col, 2, board);
14        int eval = MiniMax(depth - 1, alpha, beta, false, board);
15        UndoMove(col, board);
16
17        maxEval = Math.Max(maxEval, eval);
18        alpha = Math.Max(alpha, eval);
19        if (beta <= alpha)
20          break; // retezare alpha-beta
21      }
22      return maxEval;
23    }
24    else
25    {
26      int minEval = int.MaxValue;
27      foreach (int col in GetValidMoves(board))
28      {
29        MakeMove(col, 1, board);
30        int eval = MiniMax(depth - 1, alpha, beta, true, board);
31        UndoMove(col, board);
32
33        minEval = Math.Min(minEval, eval);
34        beta = Math.Min(beta, eval);
35        if (beta <= alpha)
36          break;
37      }
38      return minEval;
39    }
40  }

```

6 Arhitectura aplicației

Pentru o mai bună înțelegere a modului de funcționare a aplicației, în figura de mai jos este prezentată o schemă logică (organigramă) a principalelor componente și a interacțiunii dintre acestea.

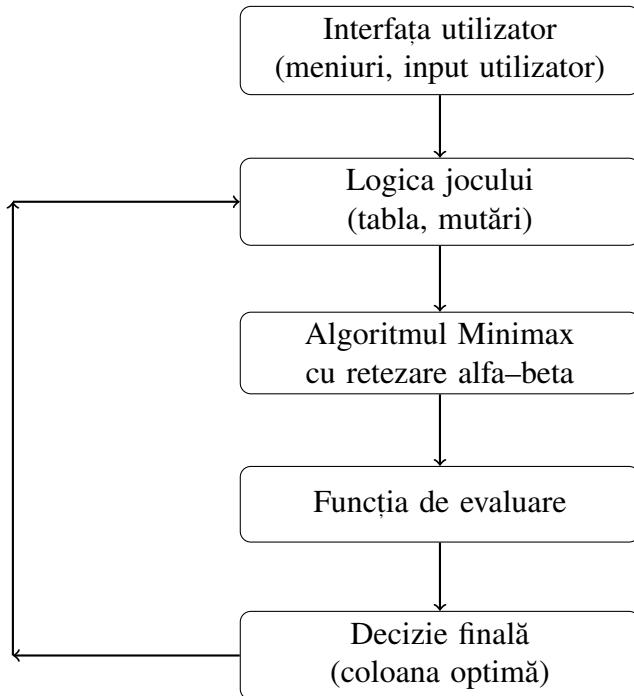


Figura 1: Organigramă simplificată a aplicației Connect 4

7 Interfața aplicației

Interfața aplicației a fost concepută pentru a fi simplă și intuitivă, astfel încât utilizatorul să poată interacționa ușor cu jocul fără a necesita cunoștințe tehnice avansate. Aplicația oferă posibilitatea configurării jocului și afișează în timp real starea partidei și rezultatul final.

7.1 Meniul principal

La pornirea aplicației, utilizatorul este întâmpinat de meniul principal, din care poate începe o partidă nouă, poate configura parametrii jocului sau poate închide aplicația. Acest meniu reprezintă punctul de acces către toate funcționalitățile aplicației.

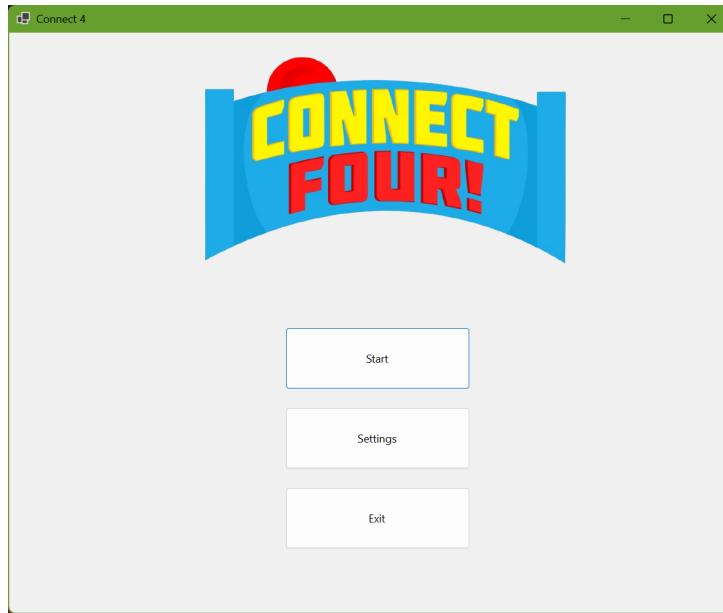


Figura 2: Meniu principal al aplicației

7.2 Setarea adâncimii de căutare

Înainte de începerea jocului, utilizatorul poate selecta adâncimea de căutare a algoritmului Minimax. Această opțiune permite ajustarea nivelului de dificultate al jocului, o adâncime mai mare conducând la un comportament mai intelligent al adversarului controlat de calculator.

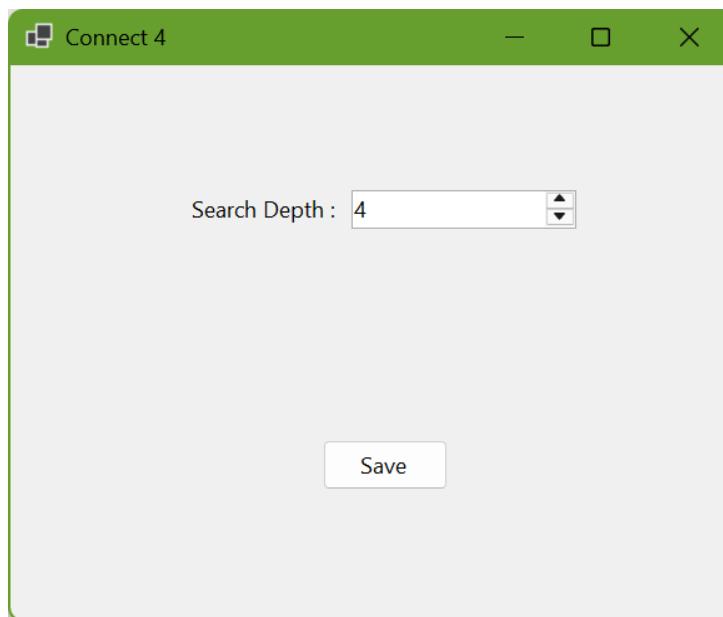


Figura 3: Selectarea adâncimii de căutare

7.3 Interfața jocului propriu-zis

În timpul jocului, tabla de joc este afișată clar, iar utilizatorul poate introduce piese prin selectarea coloanei dorite. Calculatorul răspunde automat cu propria mutare, în funcție de decizia luată de algoritm Minimax. Interfața oferă o vizualizare clară a evoluției partidei.

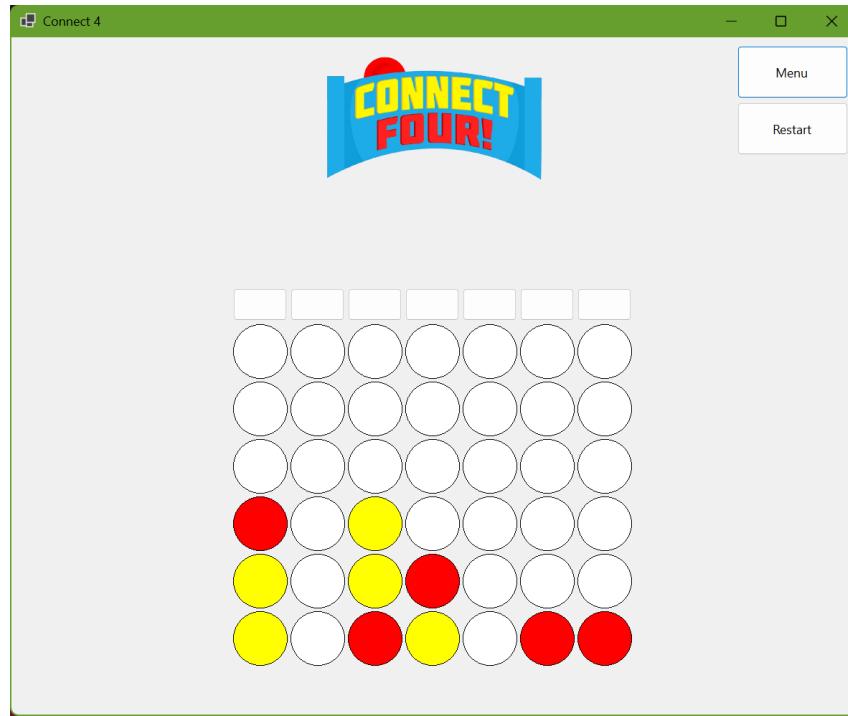


Figura 4: Desfășurarea unei partide de Connect 4

7.4 Afișarea rezultatului final

La finalul jocului, aplicația afișează rezultatul partidei, indicând dacă utilizatorul a câștigat sau a pierdut. Acest feedback vizual permite încheierea clară a jocului și oferă posibilitatea începerii unei noi partide.

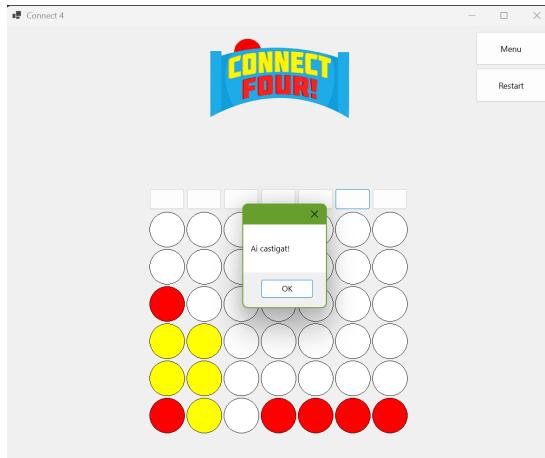


Figura 5: Mesaj afișat la câștigarea partidei

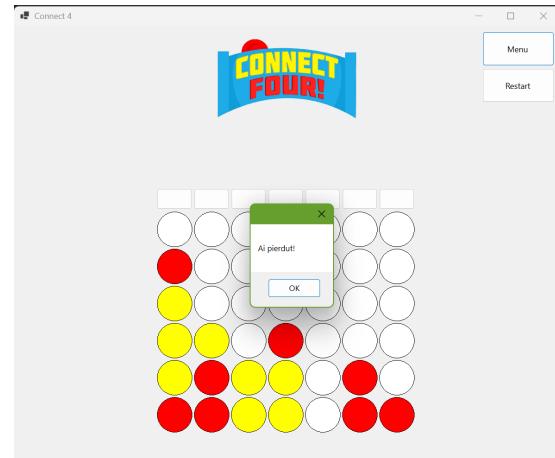


Figura 6: Mesaj afișat la pierderea partidei

8 Rezultate obținute

Aplicația a fost testată pentru mai multe adâncimi de căutare ale algoritmului Minimax. La adâncimi reduse, timpul de răspuns al aplicației este foarte mic, ceea ce oferă o experiență de

joc fluentă, însă nivelul de dificultate este mai scăzut.

Pe măsură ce adâncimea de căutare crește, calculatorul analizează un număr mai mare de mutări posibile și ia decizii mai bune, ceea ce conduce la un joc mai dificil pentru utilizator. Totuși, acest lucru implică un timp de execuție mai mare pentru fiecare mutare.

Chiar și pentru adâncimi de căutare relativ mici, adversarul controlat de calculator se dovedește a fi dificil de învins, datorită funcției de evaluare și a utilizării retezării alfa-beta, care permit identificarea rapidă a mutărilor favorabile.

Figura 7: Animație a desfășurării unei partide de Connect 4

9 Concluzii

Algoritmul Minimax cu retezare alfa-beta permite realizarea unui adversar artificial performant pentru jocul Connect 4, capabil să ia decizii strategice pe baza analizei mutărilor posibile. Prin utilizarea acestui algoritm, calculatorul este capabil să anticipateze acțiunile adversarului și să aleagă mutări avantajoase.

Retezarea alfa-beta contribuie semnificativ la reducerea timpului de calcul, eliminând ramurile din arborele de joc care nu pot influența decizia finală. Astfel, algoritmul poate analiza un număr mai mare de mutări într-un timp rezonabil, fără a afecta corectitudinea deciziilor luate.

Rezultatele obținute arată că, prin combinarea algoritmului Minimax cu o funcție de evaluare adecvată, se poate obține un adversar artificial dificil de învins, chiar și pentru adâncimi de căutare relativ reduse. Aplicația demonstrează eficiența utilizării algoritmilor clasici de inteligență artificială în dezvoltarea jocurilor strategice.

10 Bibliografie

Bibliografie

- [1] Suport de curs Inteligență Artificială, 2025–2026.
- [2] F. Leon, *IA03 – Jocuri și Probleme de Satisfacere a Constrângerilor*, https://florinleon.byethost24.com/Curs_IA/IA03_Jocuri_CSP.pdf, material de curs.
- [3] F. Leon, *Laborator IA 07 – Jocuri*, http://florinleon.byethost24.com/Lab_IA/LaboratorIA07.zip, material de laborator.
- [4] Clipart Library, *Connect Four Cliparts*, <https://clipart-library.com/clipart/connect-four-cliparts-12.htm>, logo utilizat în aplicație.
- [5] Pixabay, *Free Sound Effects*, <https://pixabay.com/>, sunete utilizate în aplicație.

11 Contribuția membrilor echipei

Proiectul a fost realizat în echipă, contribuțiile fiind împărtite astfel:

- **Ciobanu Andrei**: implementarea logicii jocului Connect 4, implementarea interfeței cu utilizatorul, testarea aplicației;
- **Paladi Andrei**: implementarea algoritmului Minimax cu retezare alfa-beta, adăugarea comentariilor și redactarea documentației.