Utilizarea strategiei minimax în realizarea unui joc de X și O

Bogdan BUDESCU   
*Universitatea Politehnica Timișoara SIAPS, Anul 2*   
*Timișoara, România*  
E-mail: bogdanbudescu96@gmail.com

*Rezumat — În prezenta lucrare este prezentată o implementarea a strategiei minimax în realizarea jocului de x-o pentru o platforma web, dar și mobile având la bază același cod sursă pentru ambele platforme.*

Cuvinte cheie— MiniMax, AI, TicTacToe, Angular, Ionic

# INTRODUCERE

În teoria jocurilor, minimax este o regulă de decizie utilizată pentru a minimiza pierderea potențială în cel mai rău caz; cu alte cuvinte, un jucător ia în considerare toate cele mai bune răspunsuri ale adversarului la strategiile sale și selectează strategia astfel încât cea mai bună strategie a adversarului să ofere o plată cât mai mare posibil.

Numele „minimax” vine de la minimizarea pierderii implicate atunci când adversarul selectează strategia care dă pierderea maximă și este util în analiza deciziilor primului jucător atât atunci când jucătorii se mișcă secvențial, cât și când jucătorii se mișcă simultan.

Minimax este, de asemenea, util în jocurile combinatorice, în care fiecărei poziții i se atribuie o plată. Cel mai simplu exemplu este atribuirea unui „1” unei poziții câștigătoare și „-1” unei poziții pierzătoare, dar, deoarece acest lucru este dificil de calculat pentru toate jocurile, cu excepția celor mai simple jocuri, evaluările intermediare (alese în mod specific pentru jocul în cauză) sunt în general necesare. În acest context, scopul primului jucător este de a maximiza evaluarea poziției, iar scopul celui de-al doilea jucător este de a minimiza evaluarea poziției, astfel încât se aplică regula minimax. Acesta este, în esență, modul în care calculatoarele abordează jocuri precum șahul și Go, deși sunt posibile diverse îmbunătățiri computaționale pentru implementarea minimax.

# Pseudocod MiniMax

Funcția minimax returnează o valoare euristică pentru nodurile frunze (noduri terminale și noduri la adâncimea maximă de căutare). Nodurile care nu sunt finale își moștenesc valoarea de la un nod descendent. Valoarea euristică este un scor care măsoară cat de favorabil este nodul pentru jucătorul care maximizează. Prin urmare, nodurile care se află într-un rezultat favorabil, cum ar fi o victorie, pentru jucătorul care maximizează au scoruri mai mari decât nodurile mai favorabile pentru jucătorul care minimizează. Valoarea euristică pentru nodurile de terminale sunt scoruri corespunzătoare câștigului, pierderii sau remizelor, pentru jucătorul maximizator. Pentru nodurile non-terminale la adâncimea maximă de căutare, o funcție de evaluare estimează o valoare euristică pentru nod. Calitatea acestei estimări și profunzimea căutării determină calitatea și precizia rezultatului minimax final.

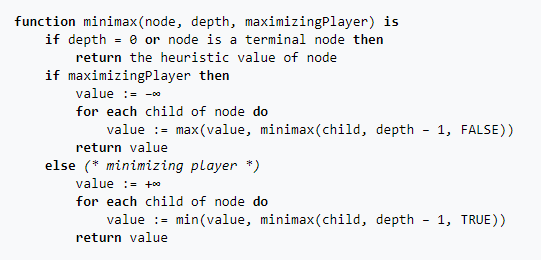


Fig. 1 – Pseudocod algoritm minimax

# Implementare

Implementarea aplicației este disponibilă la acest link: https://github.com/bogdanbudescu/nx-web-mobile. Aplicația a fost realizată utilizând frameworkul Angular și Ionic aceasta având la bază același cod sursă. Angular este folosit pentru a realiza aplicația web, iar Ionic pentru aplicația mobilă.

Interfața cu utilizatorul este prezentată în următoare figura, aceasta este foarte asemănătoare între cele două platforme.

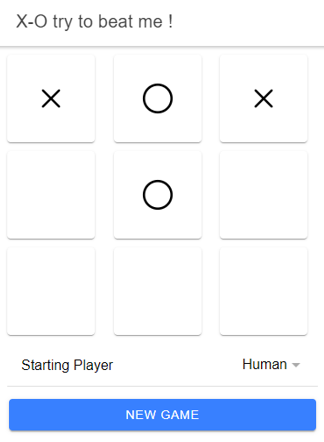


Fig. – Interfața cu utilizatorul

Aplicația realizată oferă posibilitatea de a juca împotriva calculatorului și de a alege cine să înceapă primul. Apăsând pe butonul ”New Game” un nou joc va fi început.

Algoritmul implementat urmărește pseudocodul din figura 1, acesta rulează atunci când este rolul calculatorului să facă mutarea. După ce jucătorul a ales poziția unde să plaseze pe x sau o în funcție de ce a ales jucătorul, algoritmul va urma să caute următoarea cea mai bună poziție de pe tablă. Astfel parametrii funcției pentru minimax sunt următorii: board (care reprezintă starea actuală a tablei), maxizing (dacă se maximizează este true, altfel false) și depth (care reprezintă adâncimea pană la care s-a ajuns în arbore).

Mai întâi se verifică dacă există o soluție apelând metoda isTerminal din cadrul clasei Board, care verifică toate cazurile de câștig si egalitate.

În continuare daca condiția de mai sus nu este îndeplinită pentru toate pozițiile disponibile de pe tabla actuală, se va crea o noua tablă în care se va amplasa pe poziția curentă, pentru maximizare „x”, iar pentru minimizare „o”, mai departe se va reapela metoda getBestMove. După execuția recursivă dacă suntem pe nivelul 0 al arborelui se vor adăuga într-un map valorile obținute pentru fiecare poziție disponibilă(valori obținute în urma apelului metodei getAvailablesMoves), dacă pentru o poziție există mai multe valori se va alege dintre acestea una random.

# Concluzie

În concluzie, utilizarea algoritmului minmax în crearea unui joc de x și o imbatabil s-a dovedit a fi de succes, aceasta putând fi îmbunătățită cu ajutorul alpha-beta pruning.

În același context, trebuie avut în vedere că durata de execuție a algoritmului poate fi variabilă în funcție de dimensiunea datelor de intrare.

# Referințe

|  |  |
| --- | --- |
| [1] | J. L. ANTON LUNDQVIST, „DoS Mitigation Using Proof Of Work”. |
| [2] | https://en.wikipedia.org/wiki/Proof\_of\_work, „Proof of work,” [Interactiv]. [Accesat 17 5 2020]. |
| [3] | A. Stubblefield, „Using Client Puzzles to Protect TLS”. |
| [4] | T. B. B. S. Bruno Rodrigues, „Multi-domain DDoS Mitigation Based on Blockchains,” 2017. |
| [5] | cryptowiki.net, „Proof-of-work system,” [Interactiv]. Available: http://cryptowiki.net/index.php?title=Proof-of-work\_system. [Accesat 5 17 2020]. |
| [6] | https://en.wikipedia.org, „Denial-of-service attack,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Denial-of-service\_attack. |

* https://alialaa.com/blog/tic-tac-toe-js-minimax
* https://www.neverstopbuilding.com/blog/minimax
* https://brilliant.org/wiki/minimax/
* https://brilliant.org/wiki/combinatorial-games-definition/
* https://en.wikipedia.org/wiki/Minimax