Optimizing Game Tree Search In Tic-Tac-Toe AI Using Minimax Algorithm And Alpha Beta Pruning Heuristics

Miwan Sariana Saqib
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
miwan.saqib@northsouth.edu

Md. Mubtasim Fuad
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
mubtasim.fuad01@northsouth.edu

Redwan Hossain
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
redwan.hossain21@northsouth.edu

MD Naimul Hasan Munna
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
naimul.munna@northsouth.edu

***Abstract*— This project focuses on building an intelligent Tic-Tac-Toe AI using the Minimax algorithm with Alpha-Beta Pruning to ensure optimal decision-making and fast performance. Implemented in Python, this AI is designed to be unbeatable if and when played correctly by the user. We've completed the core Minimax logic and are currently optimizing the pruning process and integrating heuristics for faster evaluations under limited depth. Key challenges include managing game tree complexity and maximizing accuracy with speed. Future work includes introducing alpha beta pruning and extending this system to larger game boards or introduce a graphical interface to improve user usability.**

Keywords— minimax, alpha-beta pruning, heuristics, adversarial games, search optimization.

I. Introduction

Tic-Tac-Toe is a simple yet strategic 3×3 grid game often used in AI to study decision-making and game tree algorithms due to its manageable complexity. This project develops an AI agent that plays Tic-Tac-Toe optimally using the Minimax algorithm [1], which evaluates all possible game outcomes to choose the best move considering a rational opponent. To improve the efficiency, Alpha-Beta Pruning [2] is integrated to eliminate unnecessary branches during the search. The AI supports human vs. AI and is further enhanced with heuristic evaluations for faster decisions under limited depth thus ensuring optimal play always resulting in a win or draw.

II. Methodology

A. Board and Game Rules

The game board is modeled as a 3×3 matrix, where each cell contains one of three symbols: 'X' for the AI's move, 'O' for the human player's move, or a blank space for an unoccupied cell. A win is declared when a player aligns three of their marks horizontally, vertically, or diagonally. A draw occurs if all positions are filled without a winner. Players alternate turns until one of these conditions is met.

B. Minimax Algorithm

The Minimax algorithm serves as the decision-making core of the AI. It recursively simulates all possible game states, assuming both players make optimal moves. Each terminal state is scored as follows: +1 for a win, –1 for a loss, and 0 for a draw as we see in fig. 1. below. The AI selects the move with the highest possible payoff while anticipating the opponent's best response. This ensures the AI never makes suboptimal decisions.

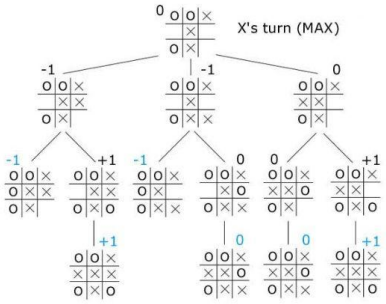

Fig. 1. A breakdown of the minimax algorithm in AI [3]

C. Alpha-Beta Pruning

To enhance performance, Alpha-Beta Pruning is applied to the Minimax search [4]. This method eliminates branches of the game tree that cannot influence the final decision by maintaining two parameters: Alpha (α), which represents the best score achievable by the maximizing player (AI), and Beta (β), the best score achievable by the minimizing player (opponent). When the condition $\alpha \geqslant \beta$ is met, further exploration of that branch is terminated all of which is visualized in the fig. 2. below. This significantly reduces the number of evaluated states, potentially cutting computation time in half while preserving decision accuracy.

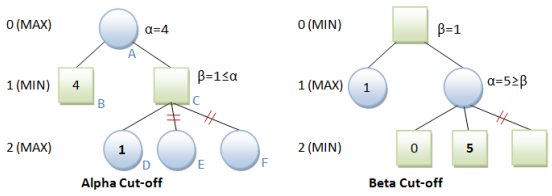

Fig. 2. A visual of Alpha-Beta Pruning [5]

III. CURRENT PROGRESS AND RESULTS

The Minimax algorithm has been fully implemented and tested. The AI carefully avoids losing and performs blocking

where necessary and winning moves. The AI analyzes game states effectively and selects the best actions based on the current board configuration. The integration of Alpha-Beta pruning will be started within the following week, with implementation allowing the AI to discard irrelevant branches for faster evaluation and quicker output.

IV. Conclusions

The project successfully demonstrates a Tic-Tac-Toe AI that uses the Minimax algorithm for optimal decision-making and further incorporates Alpha-Beta pruning to improve computational efficiency. The AI consistently guarantees a win or a draw against any opponent, showcasing its effectiveness. For future, we plan on fully optimizing the Alpha-Beta pruning technique integrating heuristic evaluations to support limited-depth searches and expanding the system to accommodate larger board configurations possibly. Additionally, the development of a graphical user interface (GUI) is planned to improve user interaction and experience. Overall, this work lays a strong foundation for building intelligent agents for classical turn-based games using classical search methods.

References

[1] Wikipedia Contributors, "Minimax," *Wikipedia*, Sep. 16, 2019. https://en.wikipedia.org/wiki/Minimax

[2] Wikipedia Contributors, "Alpha–beta pruning," *Wikipedia*, Jun. 17, 2025.

[3] "Min Max Algorithm in AI: Components, Properties, Advantages & Limitations," *upGrad blog*, Dec. 22, 2020. https://www.upgrad.com/blog/min-max-algorithm-in-ai/

[4] A. madi, "Tic-Tac-Toe agent using Alpha Beta pruning - Alaa madi - Medium," Medium, Jun. 02, 2023. https://medium.com/@amadi8/tic-tac-toe-agent-using-alpha-beta-pruning-18e8691b61d4

[5] "Tic-tac-toe AI - Java Game Programming Case Study," https://www3.ntu.edu.sg/home/ehchua/programming/java/javagame_tictactoe_ai.html.