Tic Tac Toe AI Project Documentation Report

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | | Level |
| 20220095 | Aya Farag Mohamed | | 3 |
| 20220579 | Youssef wael Attallah | | 3 |
| 20220431 | Mohamed Mostafa Mahfouz | | 3 |
| 20220297 | Ali Mohamed Alsaid | | 3 |
| 20220155 | Rahma Tayseer Mansour | | 3 |
| 20220124 | | Toqa Sameer Ahmed | 3 |

# Code and Documentation Link

**https://github.com/youssef2004wael/Tic-Tac-Toe-AI-Player.git**

## **Introduction and Overview**

## **Project Idea and Overview**

**An Intelligent Tic-Tac-Toe Player**

* Tic-Tac-Toe is a two-player game where each player takes turns marking spaces on a 3x3 grid. The objective is to get three of their marks in a horizontal, vertical, or diagonal row.
* This project aims to develop an intelligent Tic-Tac-Toe player that can compete against human players and other AI opponents.
* The AI will use advanced algorithms to ensure optimal gameplay.

### **Similar Applications**

* Health Diagnosis assistants:

AI-driven diagnostic tools help doctors by analyzing symptoms, medical history, and lab results.

These tools might recommend further tests or treatments based on the analysis, similar to how an AI in Tic-Tac-Toe evaluates the best possible move.

* Chatbots:

Chatbots interact with customers to solve problems, answer questions, or guide them through processes, similar to how AI in Tic-Tac-Toe game responds to player moves.

* Intelligent Scheduling Tools:

AI-powered scheduling tools like Calendly or Microsoft’s Cortana Scheduler automatically find the best meeting times based on participants' availability and preferences.

The system suggests optimal times, similar to how an AI player in

Tic-Tac-Toe suggests or makes the best possible move.

### **Literature Review**

1. <https://www.geeksforgeeks.org/finding-optimal-move-in-tic-tac-toe-using-minimax-algorithm-in-game-theory/>
2. <https://www.neverstopbuilding.com/blog/minimax>
3. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/>
4. <https://uyennguyen16900.medium.com/minimax-with-alpha-beta-pruning-7e2091ae7d95>

A diagram of a game

Description automatically generated

## **Proposed Solution & Dataset**

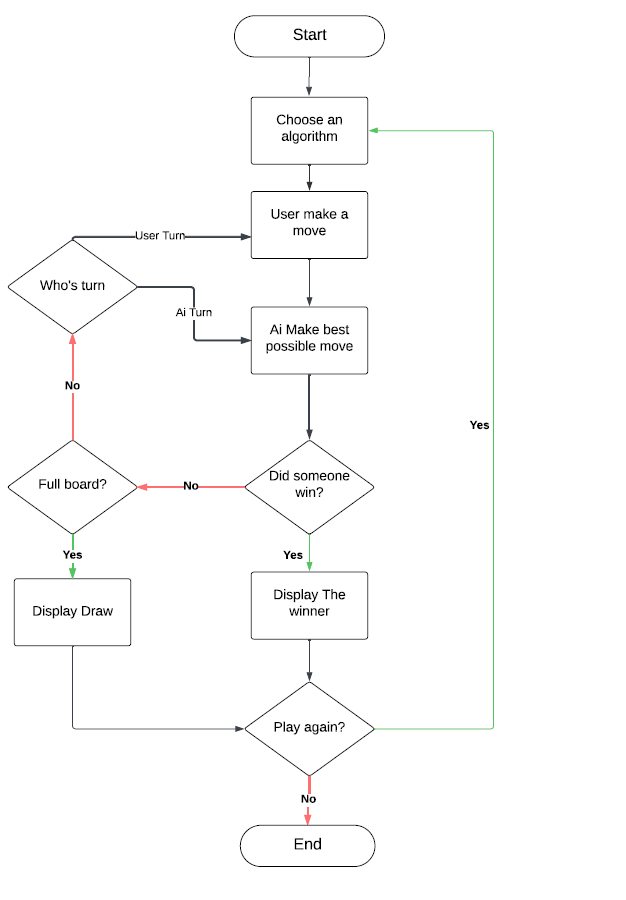
### **Main Functionalities**

User Case DiagramA diagram of a game

Description automatically generated

**Applied Algorithms**

**Algorithm Flow**



* The AI generates all possible moves.
* It evaluates each move using the Minimax algorithm.
* It selects the move with the highest score (for Player X) or the lowest score (for Player O).

### **AI Approaches**

A diagram of a diagram

Description automatically generated

A diagram of a game

Description automatically generated

1. **Minimax Algorithm**:

* A recursive algorithm that evaluates all possible moves to determine the optimal play. It assumes that both players play optimally.
* Utilizes depth-first search to explore game states.

1. **Alpha-Beta Pruning**:

* An optimization technique for the Minimax algorithm that reduces the number of nodes evaluated in the search tree.
* **Alpha (α):** The best value that the maximizer currently can guarantee.
* **Beta (β):** The best value that the minimizer currently can guarantee.
* **Pruning:** If during the tree exploration it is found that a certain branch will not provide a better outcome than previously evaluated branches it is pruned.

A screenshot of a computer

Description automatically generated

Heuristic Functions:

Minimax First heuristic function

Blocking Heuristic:

* The Blocking Heuristic is a modification to the standard Minimax algorithm. It improves the AI’s strategy by prioritizing moves that block the opponent from achieving a winning condition. The heuristic assigns higher scores to such blocking moves, even if they do not lead directly to a win for the AI.
* The algorithm checks if the opponent is one move away from winning. If so, it prioritizes blocking that move.
* If a move blocks the opponent from winning, the algorithm adjusts the score for that move to reflect its importance.

Minimax Second Heuristic Function

* This function helps in determining the best move by assigning scores based on the board state and potential future outcomes.
* If there are any opponent marks in the line, the weight is 0
* If the player has 2 marks and 1 empty space, it returns 100 (indicating a strong winning opportunity).
* If the player has 1 mark and 2 empty spaces, it returns 10 (indicating a potential blocking opportunity).
* If the player has no marks but 3 empty spaces, it returns 1.

## **Experiments & Results**

### **Experiments Conducted**

* **AI Vs. Human**: Multiple games were played against the AI at different difficulty levels to assess performance.
* **Win Rates:** The AI's win/loss ratio was calculated to evaluate the effectiveness of each algorithm.

### **Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Trial 6 | Trial 7 | Trial 8 | Average without symmetric Reduction |
| Minimax | 0.8780 | 0.0250 | 0.0029 | 0.0010 | 1.1731 | 0.0230 | 0.0020 | 0.0010 | 0.26325 seconds |
| Minimax Alpha Beta | 0.2980 | 0.0060 | 0.0010 | 0.0000 | 0.0980 | 0.0048 | 0.0010 | 0.0000 | 0.0511 seconds |
| Minimax First Heuristic | 0.0006 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0009 | 0.0000 | 0.0000 |  |
| Minimax Second Heuristic | 0.0010 | 0.0020 | 0.0030 | 0.0010 | 0.0019 | 0.0020 | 0.0010 | 0.0010 |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Trial 1 | Trial 2 | Trial 3 | Trial 4 | Trial 5 | Trial 6 | Trial 7 | Trial 8 | Average with symmetric Reduction |
| Minimax | 1.3520 | 0.0027 | 0.0010 | 0.0010 | 1.0540 | 0.0020 | 0.0029 | 0.0014 | 0.3019625 seconds |
| Minimax Alpha Beta | 0.7569 | 0.0030 | 0.0010 | 0.0020 | 0.5240 | 0.0020 | 0.0010 | 0.0010 | 0.1613625 seconds |
| Minimax First Heuristic | 0.0017 | 0.0086 | 0.0020 | 0.0010 | 0.0030 | 0.0071 | 0.0010 | 0.0018 |  |
| Minimax Second Heuristic | 0.0090 | 0.0029 | 0.0050 | 0.0011 | 0.0173 | 0.0124 | 0.0076 | 0.0124 |  |

**Conclusion**

* Minimax vs. Minimax Alpha Beta: Minimax Alpha Beta Pruning performs better with both symmetric reduction and without it. The addition of alpha-beta pruning significantly cuts down on computation time by pruning unnecessary branches in the search tree.
* Heuristics vs. Minimax Algorithms: The heuristic-based approaches are much faster compared to Minimax and Minimax Alpha Beta. This is expected since heuristics are designed to evaluate board states more efficiently rather than exhaustively exploring all possibilities. Both heuristics show significant improvements with symmetric reduction.
* Effectiveness of Symmetric Reduction: Symmetric reduction generally improves performance for all algorithms, but the degree of improvement varies. It is especially effective for Minimax Alpha Beta and the heuristic-based algorithms. However, the overall speed-up with Minimax is relatively modest, suggesting that while symmetric reduction helps, the reduction in unique board states might be limited by the inherent structure of the Minimax algorithm.

## **Analysis, Discussion, and Future Work**

### **Analysis of Results**

The results indicate that the AI effectively utilizes Minimax and heuristic evaluations to make informed decisions. This leads to a high win rate against average human players.

### **Advantages and Disadvantages**

* **Minimax**

Advantages:

* Guarantees optimal play
* Simple to implement
* Exhaustively explores all possible moves
* Provides deterministic outcomes

Disadvantages:

* Computationally expensive for larger games
* Requires significant memory
* Assumes optimal play by the opponent: In real-world scenarios, players may not always act rationally or optimally, which Minimax does not account for.
* **Alpha-Beta Pruning in minimax**

Advantages:

* Reduces computational load by pruning unnecessary branches
* Speeds up decision making
* Makes deeper searches feasible
* Retains the optimal play of the Minimax algorithm.

Disadvantages:

* More complex to implement than basic Minimax
* Efficiency depends on move ordering
* Requires additional memory
* Alpha-beta pruning does not change the fundamental complexity of the game itself. It only improves the search process
* **First Heuristic function**

Advantages:

* Prioritizes critical moves (blocking and winning)
* Simple and intuitive
* Provides immediate feedback
* Effective for the small state space of Tic-Tac-Toe

Disadvantages:

* Narrow focus on blocking and winning, missing other strategic factors
* Short-term evaluation without long-term planning
* Assumes opponent’s play is not considered
* Potential imbalance in scoring between blocking and winning
* Dependent on the accuracy of utility functions

* **Second Heuristic function**

Advantages:

* Evaluates strategic value based on line configurations
* Provides high reward for immediate wins
* Adapts to various board states with line weighting

Disadvantages:

* Single line focus rather than multiple line aggregation
* Assumes specific player symbols which may need adaptation

### **Future work could involve:**

* Implementing more advanced heuristics to improve decision-making.
* Allowing users to customize certain parameters, such as the aggressiveness of the AI or the depth of the search, could enhance user engagement and satisfaction. This personalization would cater to different player preferences and skill levels.
* Exploring machine learning techniques to allow the AI to learn from past games and adapt its strategy.
* Expanding the project to include a multiplayer mode where two humans can compete against each other could enhance the social aspect of the game. This would involve adding functionality to manage multiple players and their turns.
* Enhancing the user interface for better accessibility and engagement.