Project Proposal: 5x5 Tic Tac Toe with Advanced AI

**Submitted By: Faizan Ali (22I-2496), Zaid Vohra (22K-4195), Hamail Rehman (22K-4443)**

**Course: Artificial Intelligence**

**Instructor: Mr. Khalid Khan**

**Submission Date: 11th May 2025**

# 1. Project Overview

## ● Project Topic:

The project explores an innovative variation of Tic Tac Toe played on a 5x5 grid where the goal is to achieve 4 in a row instead of the traditional 3. The challenge is enhanced by a Minimax AI agent that uses advanced techniques such as Alpha-Beta Pruning, Move Ordering, Heuristic Evaluation, and Caching for an optimal game strategy. The AI agent competes against human players and other AI agents, making strategic decisions based on the current state of the board.

## ● Objective:

The primary objective of this project is to create a strategic AI agent for an unconventional 5x5 Tic Tac Toe game. The AI will use the Minimax algorithm with Alpha-Beta Pruning to efficiently explore the game tree, while optimizing the decision-making process through techniques such as move ordering and caching. This will allow the AI to make the best possible moves and provide a challenging opponent for human players.

# 2. Game Description

## ● Original Game Background:

Tic Tac Toe is a popular game traditionally played on a 3x3 grid. Two players take turns placing their marks (X or O) in the empty cells of the grid. The goal is to align three of one’s marks in a row, column, or diagonal to win. If all cells are filled without a winner, the game ends in a draw.

In this project, the game will be played on a 5x5 grid, and the goal is to achieve 4 in a row instead of 3. The game will still be turn-based, with two players (human vs. AI or AI vs. AI) alternating their moves.

## ● Innovations Introduced:

5x5 Grid: The grid size is increased from the traditional 3x3 to a 5x5, providing more complexity and larger state space.

Winning Condition of 4 in a Row: The winning condition is modified to require 4 consecutive marks in a row (horizontally, vertically, or diagonally) rather than the traditional 3 in a row.

Strategic AI: The AI agent will use advanced search techniques such as Minimax with Alpha-Beta Pruning for optimal gameplay, making the game more challenging and interesting.

Move Ordering and Caching: The AI will incorporate move ordering (evaluating and prioritizing moves based on their potential impact) and caching to avoid redundant calculations, thus improving its performance.

These innovations introduce more strategic depth to the game, requiring players to think several moves ahead to win or block their opponents.

# 3. AI Approach and Methodology

## ● AI Techniques to be Used:

Minimax Algorithm: The AI agent will use the Minimax algorithm to recursively evaluate game states and select the optimal move. The algorithm will explore the tree of possible game states, maximizing its chances to win and minimizing the opponent's chances.

Alpha-Beta Pruning: Alpha-Beta pruning will be implemented to optimize the Minimax search by cutting off branches of the game tree that do not need to be explored, reducing the number of evaluated states.

Move Ordering: Moves will be ordered by evaluating their impact with a shallow heuristic evaluation, helping the AI prioritize more promising moves.

Caching: To speed up the search, previously evaluated game states will be cached and reused to avoid redundant calculations.

Reinforcement Learning (Optional): In future versions, we may implement reinforcement learning to allow the AI to learn from gameplay and improve over time through self-play.

## ● Heuristic Design:

The heuristic evaluation function will score game states based on the following factors:

4 in a row (Winning move): 10,000 points.

3 in a row: 100 points.

2 in a row: 10 points.

1 in a row: 1 point.

Blocking opponent’s winning move: This will be weighted by a factor (BLOCK\_WEIGHT) to penalize the AI for blocking the opponent.

This heuristic will allow the AI to assess the relative strength of different game states and make informed decisions.

## ● Complexity Analysis:

The complexity of the Minimax algorithm grows exponentially with the number of possible moves at each level (O(b^d), where b is the branching factor and d is the depth of the tree). By implementing Alpha-Beta pruning and move ordering, we reduce the number of explored states significantly, making the search process more efficient.

Since the grid size is 5x5, the game’s state space is significantly larger than the traditional 3x3 grid, leading to increased computational complexity. Caching and pruning help mitigate this challenge.

# 4. Game Rules and Mechanics

## ● Modified Rules:

Grid Size: The game is played on a 5x5 grid.

Winning Condition: A player wins by aligning 4 marks in a row either horizontally, vertically, or diagonally.

AI Move: The AI uses the Minimax algorithm to select the best move based on the current game state.

## ● Winning Conditions:

A player wins the game by aligning 4 of their marks in a row either horizontally, vertically, or diagonally. If the board is filled without a winner, the game ends in a draw.

## ● Turn Sequence:

The game alternates turns between the player and the AI.

The player will choose a move by selecting an empty cell on the board.

The AI will automatically calculate and make its move after the player.

# 5. Implementation Plan

## ● Programming Language:

Python

## ● Libraries and Tools:

Pygame: For the graphical user interface (GUI), if applicable.

NumPy: For efficient board state representation and data handling.

Scikit-learn (optional): For advanced machine learning techniques if implementing reinforcement learning.

## ● Milestones and Timeline:

Week 1-2: Design the 5x5 grid, finalize the game rules, and define the modified winning conditions.

Week 3-4: Develop the AI using the Minimax algorithm and implement basic move evaluation heuristics.

Week 5-6: Integrate Alpha-Beta pruning and implement move ordering for optimal AI performance.

Week 7: Implement caching for storing evaluated game states and optimize the overall performance.

Week 8: Test the AI and finalize the project with performance metrics, including state exploration count and move decision time.

# 6. References

Minimax Algorithm: https://en.wikipedia.org/wiki/Minimax

Alpha-Beta Pruning: https://en.wikipedia.org/wiki/Alpha%E2%80%93beta\_pruning

Tic Tac Toe Game Theory: https://en.wikipedia.org/wiki/Tic-tac-toe

Pygame Documentation: https://www.pygame.org/docs/