Project Report: 5-Player Ludo with AI

Submitted By: Haiqa Khan
Course: Artificial Intelligence
Instructor: Abdullah Yaqoob
Submission Date: May 11, 2025

1. Executive Summary

Project Overview:
This project develops a 5-player version of the classic Ludo game, enhanced with AI opponents using the Minimax algorithm with alpha-beta pruning. The game introduces a pentagon-shaped board, power-up tiles (*Double Roll* and *Safe Zone*), and supports one human player against four AI players. The primary objectives were to implement a strategic AI, create an engaging user interface using Pygame, and innovate on traditional Ludo rules to support five players with unique gameplay mechanics.

2. Introduction

Background:
Ludo is a traditional board game for 2–4 players, where the objective is to move tokens around a square board to a central home area. This project was chosen to explore multi-player AI strategies in a modified game environment. A 5-player version was developed with a pentagon-shaped board, three tokens per player, and power-up tiles to add strategic depth and visual appeal.

Objectives of the Project:

- Develop a Minimax-based AI with alpha-beta pruning to play the modified 5-player Ludo game.
- Design a pentagon-shaped board with innovative power-up mechanics.
- Implement an interactive user interface using Pygame.
- Test the AI's performance against a human player in a multi-player setting.

3. Game Description

Original Game Rules:
In traditional Ludo, each of 2–4 players has four tokens. Players roll a die to move tokens clockwise around a square board, aiming to reach the central home area. Landing on an opponent's token sends it back to the start, and safe zones protect tokens from capture. The first player to move all tokens to the home area wins.

Innovations and Modifications:

- Increased to 5 players, each with 3 tokens, to support more players and faster gameplay.
- Designed a pentagon-shaped board with 40 tiles per player path, divided into five 8-tile segments.
- Introduced power-up tiles: *Double Roll* (grants an extra turn) and *Safe Zone* (protects tokens).
- Modified win condition: the first player to move at least two tokens to the home area wins.

4. AI Approach and Methodology

AI Techniques Used:
The AI employs the Minimax algorithm with alpha-beta pruning to evaluate game states and select optimal moves. This approach ensures strategic decision-making in a competitive 5-player environment by anticipating opponents' moves.

Algorithm and Heuristic Design:
The Minimax algorithm explores possible moves up to a depth of 2, using alpha-beta pruning to reduce computation time. The evaluation function calculates the AI player's score (10 points per token in the home area plus remaining board positions) minus the average opponent score divided by 4. This heuristic prioritizes advancing tokens while considering opponents' progress.

AI Performance Evaluation:
The AI's performance was assessed based on its win rate against the human player, decision-making time, and ability to utilize power-ups. Informal testing showed the AI winning approximately 60% of games, with an average decision time of 1–2 seconds per move, depending on game state complexity.

5. Game Mechanics and Rules

Modified Game Rules:

- Each of 5 players has 3 tokens, starting in their home base.
- A roll of 6 is required to move a token from the home base to the starting tile.
- Tokens move clockwise based on the dice roll, up to 40 tiles to reach the home area.
- Landing on an opponent's token on a non-safe tile sends it back to the home base.
- Power-up tiles:
    - *Double Roll*: Allows another turn.
    - *Safe Zone*: Prevents capture.

Turn-based Mechanics:
Players take turns rolling a die. The human player (Player 1) uses the spacebar to roll, move, or use power-ups. AI players (Players 2–5) automatically make decisions. After a move, the turn passes to the next player unless a *Double Roll* power-up is activated.

Winning Conditions:
The first player to move at least two of their three tokens to the central home area (tile 40) wins the game.

6. Implementation and Development

Development Process:
The game was developed using Python and Pygame. The process involved designing the pentagon board, implementing game logic, coding the AI, and creating the user interface. The `LudoGame` class manages game state, while the `AIPlayer` class handles AI decisions. The UI was iteratively refined to display player status and available moves clearly.

Programming Languages and Tools:

- Programming Language: Python
- Libraries: Pygame

Challenges Encountered:

- AI Performance: The Minimax algorithm was initially slow due to the large number of possible moves. Alpha-beta pruning significantly reduced computation time.

- Board Design: Calculating tile positions on a pentagon board required complex parametric interpolation, addressed by the `get_tile_position` function.

- UI Clarity: Displaying five players' statuses required careful layout and design considerations.

7. Team Contributions

Team Members and Responsibilities:

- [Haiqa Khan]: Responsible for AI algorithm development (Minimax, Alpha-Beta Pruning).
- [Nashmia Mirza]: Handled game rule modifications and board design.
- [Warsiha Siddique]: Focused on implementing the user interface and integrating AI with gameplay.

8. Results and Discussion

AI Performance:
The AI achieved a win rate of approximately 60% against the human player in informal testing, with decision-making times averaging 1–2 seconds per move. The AI effectively utilized power-ups, particularly prioritizing *Double Roll* to gain extra turns. The alpha-beta pruning optimization ensured reasonable performance despite the 5-player complexity.

9. References

- "Minimax Algorithm in Game Playing," GeeksforGeeks, https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory/, accessed April 2025.