HW9: Games – Playing Teeko

1 Introduction

In this assignment, you will develop an AI game player for the game Teeko. Teeko is a solved game, meaning an optimal move can be determined from any given position with enough computing power.

2 How to Play Teeko

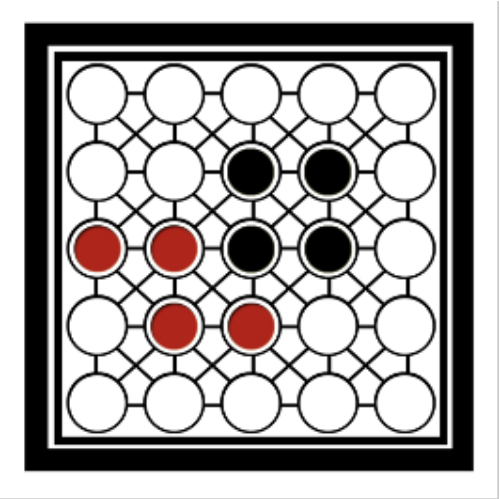

Figure 1: A typical configuration in Teeko. In this case, the black marker wins as it forms a 2×2 square.

Teeko is a simple game between two players on a 5×5 board. Each player has four markers (**red** or **black**). Starting with black, players take turns placing markers during the **drop phase**. The goal is to arrange four markers in a row (horizontal, vertical, diagonal) or form a 2×2 square. If neither player has won after the drop phase, players continue by moving one marker per turn to an adjacent empty space (**including diagonals**). Note that, there is absolutely no wrap-around allowed. This means a player cannot move off the edge of the board, nor can they use pieces from the opposite edge of the board to achieve a winning condition.

Win Conditions

- Four same-colored markers in a horizontal, vertical, or diagonal row.
- Four same-colored markers forming a 2×2 square.

You can also refer to this link as an interactive demo.

3 Program Specification

You're provided with a Python class skeleton in `game.py`. Your task is to implement an intelligent AI player using the minimax algorithm.

3.1 `make_move` method

Implement `make_move(self, state)` by generating a minimax tree up to a certain depth, scoring nodes using a heuristic function, and selecting the best move.

3.2 Helper Functions

Implement the following helper functions:

1. **Generate Successors:** Define `succ(self, state)` that returns a list of legal successor states (i.e., states that are one marker-adjacent to the current states). During the drop phase, this simply means adding a new piece of the current player's type to the board; during continued gameplay, this means moving any one of the current player's pieces to an unoccupied location on the board, adjacent to that piece.

   Note: Wrapping around the edge is **NOT** allowed when determining "adjacent" positions.

2. **Evaluate Successors:**

   - Finish `game_value(self, state)` for diagonal and 2×2 win conditions (1 if this Teeko-Player wins, -1 if the opponent wins, 0 if no winner).
   - Create `heuristic_game_value(self, state)` to evaluate non-terminal states (use floating-point scores between -1 and 1). For some hints, check out the lecture slides on Game Theory, Minimax and Alpha-Beta Pruning (you should call the `game_value` method from this function to determine whether the state is a terminal state before you start evaluating it heuristically.) This function should return some floating-point value between 1 and -1.

3. **Implement Minimax:** Follow the pseudocode from class slides, implement `max_value(self, state, depth)` with a depth cutoff (up to 3 should be good enough) ensuring moves are chosen within 5 seconds.

   Note: You can use Python's time library to measure how long it takes to select the next move.

4 Evaluating Your Code

Your implementation will be graded on:

- Correctly following Teeko rules.
- Returning moves without modifying the current state.
- Selecting moves within 5 seconds.

Your AI will be tested against three opponents of varying difficulty levels:

- **Easy:** Your AI must be able to beat the easy opponent 75% of the time for full credit.
- **Medium:** Your AI must be able to beat the medium opponent 50% of the time for full credit.
- **Hard:** Your AI must be able to beat the hard opponent 25% of the time for full credit.

5 Submission

Submit your `game.py` file. Ensure no extra code or prints outside provided main statements and functions.

6 Extra Credit

You can earn up to 15 extra points by tying or beating the Challenge Opponent. Activate this test by setting `run_challenge_test` flag to `True` in `game.py`.

7 Grading Note

This assignment totals 115 points, with extra credit allowing you to compensate for lost points (up to 15 points) from previous assignments. Your final assignments tab grade will be calculated as: $\min(\texttt{assignment\_score}, 800)$. Good luck!