Artificial Intelligence Project

Mini-Checkers Game

By Sneha Munden (sm7352)

Description

This project is the implementation of a Mini Checkers game in Java using AI concepts where a human can play against the computer. The game has a reduced board size of 6 x 6 squares compared to the traditional 8 x 8 board and also few rules have been modified. The game uses the minimax algorithm along with alpha-beta search algorithm for pruning to determine the next best move.

Instructions to Run

- To execute the project, open the project folder in a java IDE such as IntelliJ.
- Select the Java SDK path for the project in project options.
- Select the project output path to where the class files should be created. This should be <project folder path>/out.
- Go to Edit Run Configurations for the project and select the main class to be as MiniCheckers.java (com.project.MiniCheckers).
- Click on run and select the difficulty level to start the game.

Game Features

- A graphical user interface (GUI) developed using JavaFX which allows users to move pieces by mouse drag and drop.
- The game supports three levels of difficulty: Easy, Normal and Hard. The user is prompted to select the difficulty level at the start of the game.
- It involves a 6 x 6 board with 6 pieces of white and black each. Black pieces belong to the user and white to the computer. All pieces are placed on dark tiles.
- The user can choose to move first or second at the start of the game.
- There are two types of moves possible, A regular move where a piece can move forward diagonally to an adjacent square that is empty. Or a capture move where a piece can jump over and capture opponent's piece in diagonal direction and land on an empty square.
- If there are no legal moves possible for the user, then their turn is skipped.
- The game ends when one player captures all pieces of the other player or if there are no more possible moves left for either of the players. In that case, the player with more number of pieces left wins the game. If both have same number of pieces left, then it will be a draw.

Implementation

- This project is implemented using Java and developed in IntelliJ IDE. It uses JavaFX for the GUI.
- To determine the next best move, the Alpha Beta Search algorithm is used.
- Whenever the user's turn is over, the Alpha beta search (ABS) function is called, which based on the current board state considers all possible valid moves and determines the next best move.
- Based on the next best move using the ABS function, the board state changes and the user is allowed to make the next move.
- This cycle repeats till the game ends.
- There are three terminal states with assigned utility values:
  Black wins (+1000), White wins (-1000), Draw (0)
- The ABS function uses an evaluation function which uses the difference between number of black or white pieces as a heuristic.
- It also uses the Cutoff function by setting the max depth limit. As soon as the search reaches the max depth limit, it cuts the search and returns the best move.
- The higher the max depth limit is set, the difficulty of the game increases.
- For implementing three different levels of difficulty, different max depth limits are used:
  Easy (3), Normal (9), Hard (15).