Connect Four AI Game Using MiniMax Alpha-Beta Pruning

Authors: Ari Gleich, Mihir Joshi, Andrea Kuhn

Problem Description:

Connect Four is a two player game in which players take turns dropping pieces into a six row and seven column vertical grid. When a piece is played, it will drop to the lowest open row in the selected column. The goal of the game is to be the first player to create a horizontal, vertical, or diagonal line out of four of one's own pieces.

We sought to create a playable Connect Four game using Python. Once the game was created, we planned to make one opponent a rational artificial intelligence agent utilizing minimax search with alpha-beta pruning for the user to play against. This AI agent will assume both players act optimally. The AI was intended to have five levels of difficulty corresponding to the depth of the search (one being the lowest depth and five being the deepest). It is intended for the user to select the difficulty before playing. To implement minimax search, a heuristic has to be created to score the board so the AI can select its best move. Lastly, we intended for the game to not only work with the traditional 6x7 grid and four in a row winning, but to be compatible with multiple grid sizes and various win lengths. Thus, we needed to create a heuristic that worked with various board sizes and win lengths.

The performance of the AI agent will be measured by its ability to win the game and select legal moves. The environment the agent will be operating in is the Connect Four game as previously described in the background section and it will assume that its opponent is acting optimally. Its actuators are the ability to place a piece in an empty slot on the grip. The agent sensors consist of "seeing" where the pieces are currently on the grid and which pieces are the agents versus its opponents.

Solution Plan:

Using Python and particularly PyGame, we first wrote code to build the Connect Four environment and allow for game play. We then converted one of the players into an artificial intelligence agent that utilizes MiniMax search with Alpha-Beta pruning to select its moves. The heuristic gave points for having tokens in a vertical, horizontal, or diagonal row. The more tokens of the AI in a row, the higher its score. The fewer moves it took to get these tokens in a row also led to a higher score. The more tokens the opponent could have in a row with a theoretical move lead to a lower score.

We also created various “difficulties” ranging from 1 to 5 corresponding to the depth of the search (how many moves ahead the AI could check). We played at these various difficulties to ensure there was a difference and checked the computation times while coding it.

We also created a second version of the gameplay file to allow for AI vs AI gameplay. We then created graphs showing time vs move number for simple minimax vs minimax with alpha beta pruning.

Current Results:

After following the steps mentioned above, a playable Connect Four game was created with a graphical user interface.

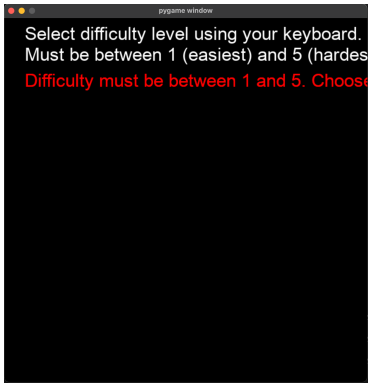

Fig. 1 Initial Game Screen
The user is prompted to select their difficulty from 1 to 5.

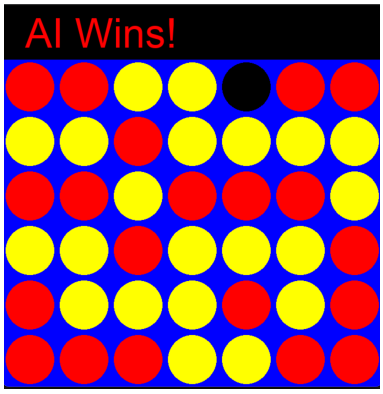

Fig. 2 Gameplay
The Connect Four board that alternates the player and AI agent’s turn (depth = 3).

In Figure 1, the user selects their difficulty which corresponds to the depth of the search (i.e. higher depth means higher difficulty). Since higher depth techniques can take increased amounts of time, Alpha-Beta pruning was implemented as a way to optimize the Minimax search algorithm to find the best move.

The user can also run the AI vs AI gameplay file and select the depth of search for two AI agents. It can then watch the agents play each other and a message will be displayed stating which AI won.

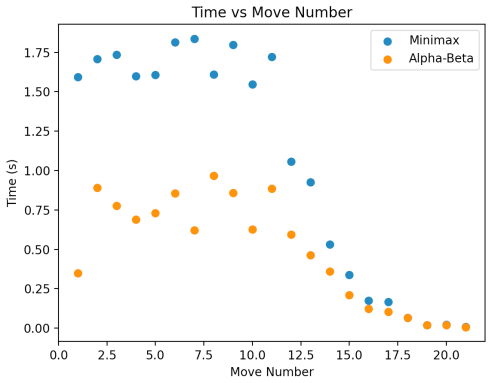

Fig. 3 Time vs. Move Number
Scatter plot showing time taken for each move from Figure 2 gameplay (depth = 3).

Search Technique	Mean	Standard Deviation
Minimax	1.0417	0.7445
α-β Pruning	0.4868	0.3391

Fig. 4 Statistics
Table showing the means and standard deviations of times taken (in seconds) per move for Minimax and α-β Pruning.

In order to compare the time taken for each search technique, the code was modified to first separate out the Minimax and the Alpha-Beta pruning techniques. In the gameplay portion of the code, both functions were called and evaluated for each move to minimize potential variability in move selection time between different games. The times for each move were collected in an array to allow for plotting and statistical analysis. Based on the timings above, the Alpha-Beta pruning technique was significantly faster on average than the minimax technique since unnecessary nodes were not being opened.

Problems faced along the way:

We noticed that the AI would occasionally not play a winning move when it was available. Instead, it would play a move that would lead to an eventual win down the line, but maximized the heuristic which gave points for three in a row. Therefore, we added in a metric for the number of moves taken to the heuristic. The higher the number of moves, the lower the score. This caused the AI to play the move leading to the immediate win rather than the eventual win and thus solved our problem.

Future Works:

In the future, we could create an AI agent to play other logical games like checkers or chess. We could also work on improving our algorithm by employing other techniques to make the search time faster since the time the AI takes to play significantly increases as depth increases. Lastly, with the two AI agents playing each other, it could be interesting to see the win percentages of agents with different search depths playing each other. This could be demonstrated graphically.