Connect-4 AI Solver

Artificial Intelligence: Final Project

Sam Sliman, Parker Smith, Tim Schachner, Josh Scherer

Background

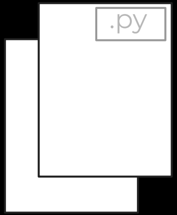
Connect 4 is a 2 player strategy game where players take turns dropping pieces into a 6x7 grid. The objective is to be the first to form a horizontal, vertical, or diagonal line of one's own pieces. Players must anticipate opponent's moves to block, while simultaneously working towards their own winning line.

Contribution

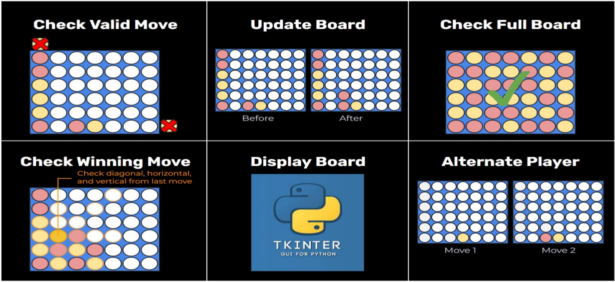
Goal: Create an AI Solver using the MiniMax Algorithm and a reward function that will perform well against itself or a human player.

General Outline

Run Program



Game Structure


Check Valid Move

Update Board

Check Full Board

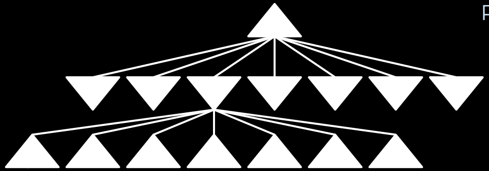
Check Winning Move

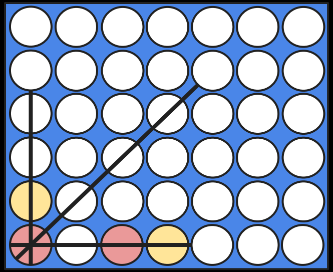
Display Board

Alternate Player

MiniMax Algorithm ⟷ Reward Calculation

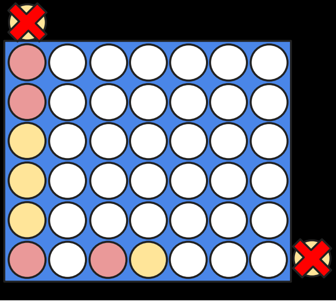
Performed Simultaneously



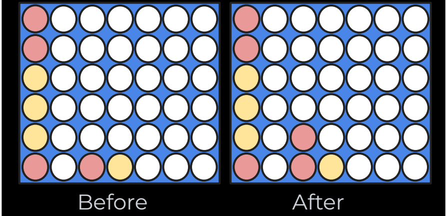


Game Structure

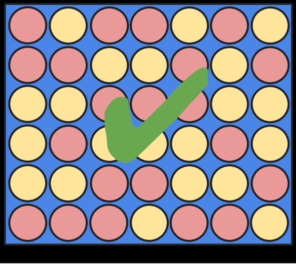
Check Valid Move



Update Board

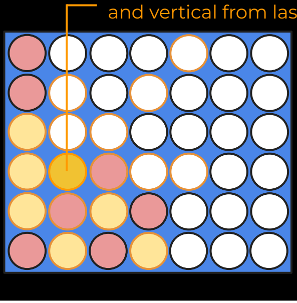


Check Full Board



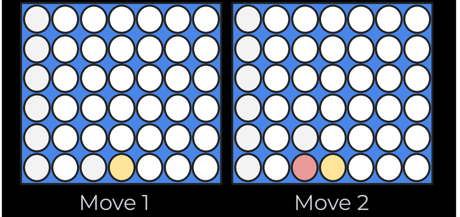
Check Winning Move

Check diagonal, horizontal, and vertical from last move



Display Board

Alternate Player



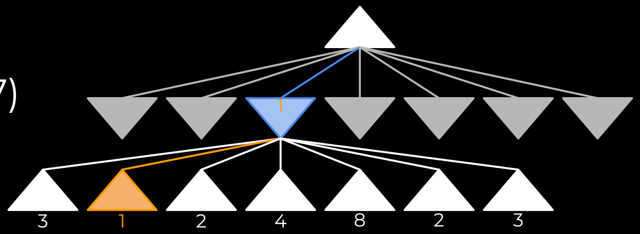
MiniMax Algorithm

Each node is a board state from a subsequent move (branching factor 7)

Each node is assigned a reward

Since MiniMax assumes the opponent makes the best possible move, we can use the same reward calculation, but reverse the sign for subsequent layers

A state is terminal (does not recurse) if either player wins or the board is full



Player 2 - MAX

Player 1 - MIN

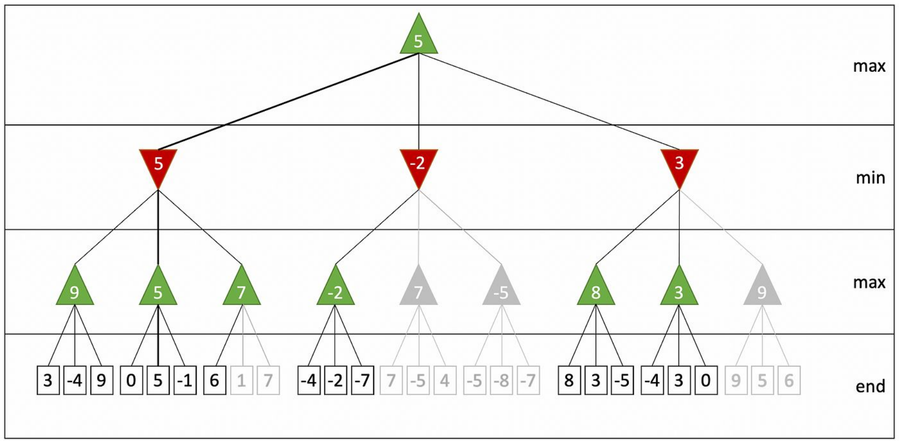
Player 2 - MAX

Shown with fully calculated board rewards

Alpha-Beta Pruning

- If a maximizing node can achieve a value of x via its first branch, and the remaining minimizers beneath it would achieve values less than x, the maximizer ignores those branches and does not expand them any further than necessary.

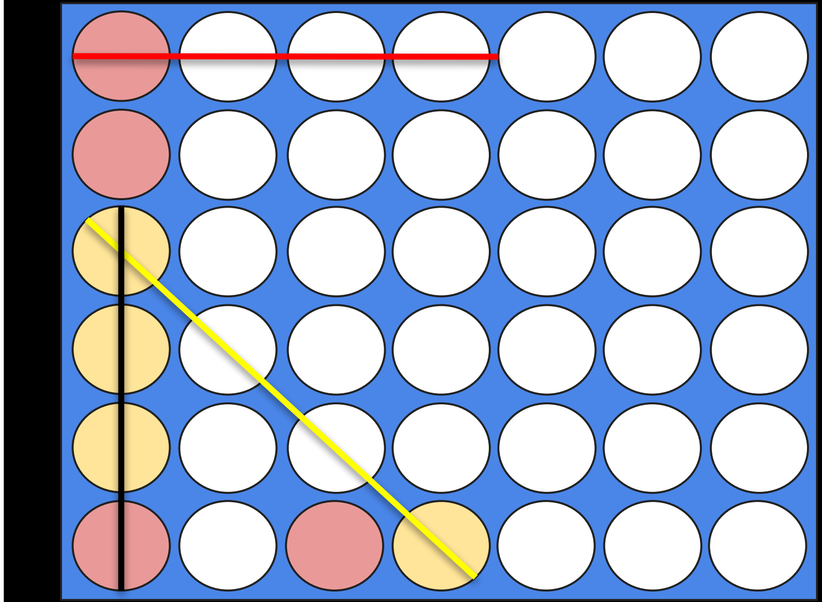
- Similarly, if a minimizing node can achieve a value of x via its first branch, and the remaining maximizers beneath it would achieve values greater than x, the minimizer ignores those branches and does not expand them any further than necessary.



Reward Calculation

- Positive value => Player 1 is winning
- Negative value => Player 2 is winning
- Given a multiplier, for each Connect-4 possibility, if only one player is present, they get a reward of the multiplier raised to the power of the number of pieces they have
- If a player goes next, their multiplier is doubled to more heavily weight sooner outcomes
- If a Player 1 has Connect-4, we add 1,000,000; if Player 2 has Connect-4, we subtract 1,000,000
- If both players are present in a specific Connect-4 possibility, we multiply by 0 to ignore that possibility (it is no longer valid)
- Add this number to a total for each sequence of 4 on the board to assess overall position

Reward Walkthrough



Suppose multiplier = 7.

Horizontal from (0,0): 1 red, 3 blank => (1) * $(7)^1$ = 7

- If red moves next, multiplier = 2 * 7 = 14, so result = 14

Diagonal from (2, 0): 2 yellow, 2 blank => (-1) * $(7)^2$ = -49

- If yellow moves next, multiplier = 14, so result = -196

Vertical from (6, 0): 1 red, 3 yellow => 0

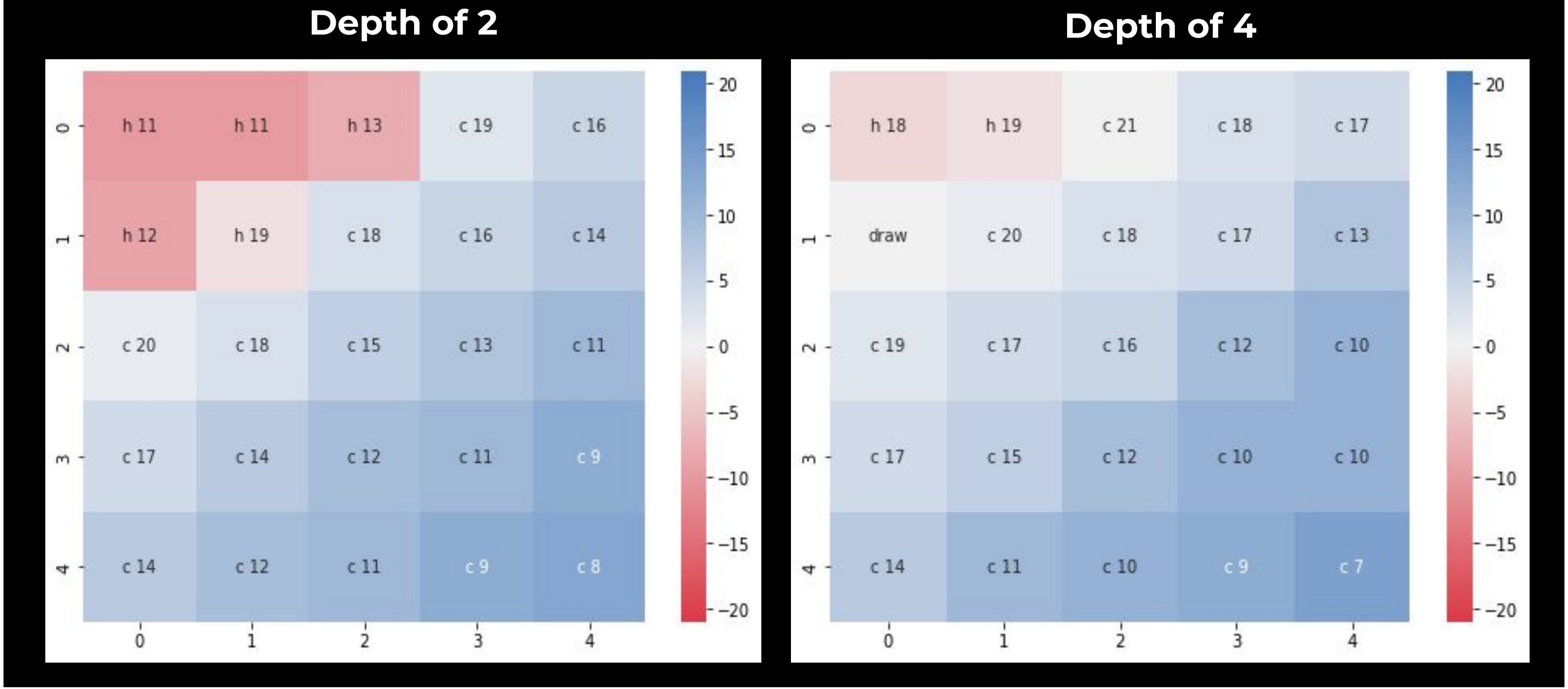
Challenges

- Note: We were doing this without any foundational code; we built the entire thing from scratch
- Minimax Struggle
  - Our minimax tree depth was initially 3 (odd number) and should have been 4 (or an even number).
  - The “depth” of our tree needed to be even to allow both the minimizer and the maximizer to have an even number of moves.
- Heuristic Struggle
  - Our multiplier was too low (initially at 3).
  - As a result, the number of possible winning paths during the heuristic analysis would exceed our multiplier. For instance, if there were 4 potential winning paths, it would be better than if we had one potential winning path with 3.

Accuracy Analysis

How Does Our Solver Perform?

Solver Performance



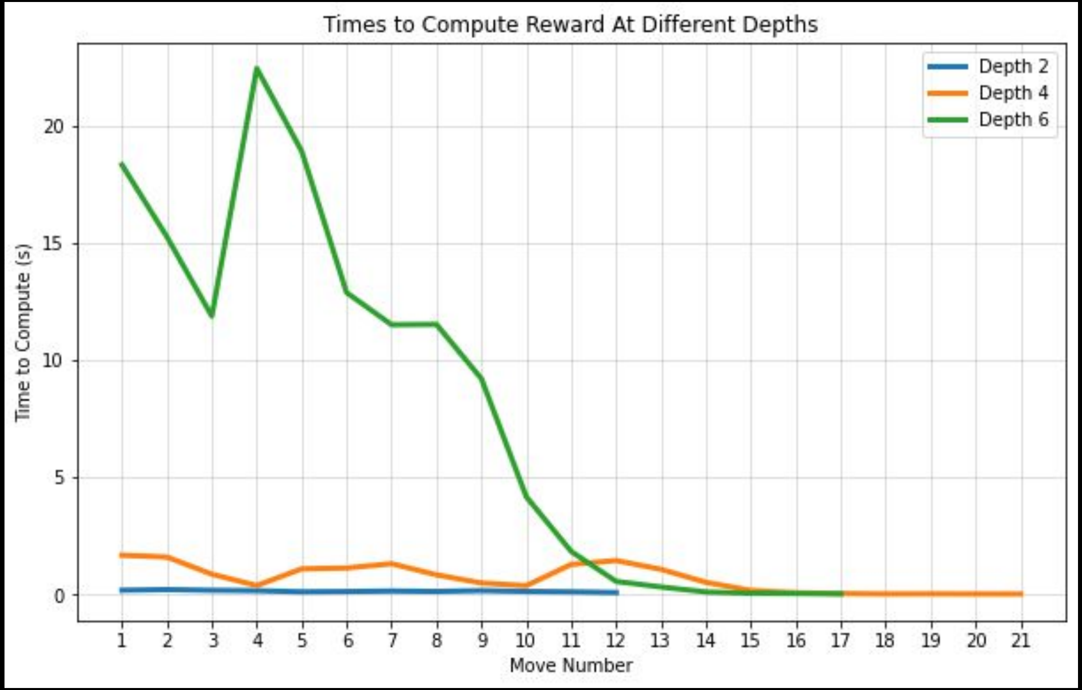
Depth of 2

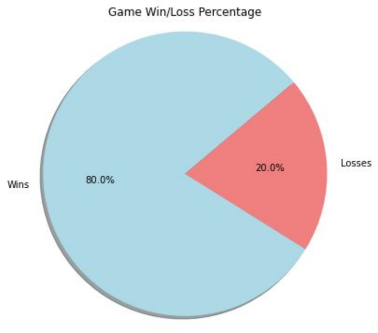
	0	1	2	3	4
0	h 11	h 11	h 13	c 19	c 16
1	h 12	h 19	c 18	c 16	c 14
2	c 20	c 18	c 15	c 13	c 11
3	c 17	c 14	c 12	c 11	c 9
4	c 14	c 12	c 11	c 9	c 8

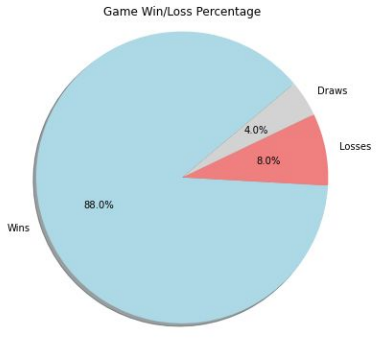
Depth of 4

	0	1	2	3	4
0	h 18	h 19	c 21	c 18	c 17
1	draw	c 20	c 18	c 17	c 13
2	c 19	c 17	c 16	c 12	c 10
3	c 17	c 15	c 12	c 10	c 10
4	c 14	c 11	c 10	c 9	c 7

Computational Time vs. Accuracy







Computer vs. Self (Depth 6)

Thank You!