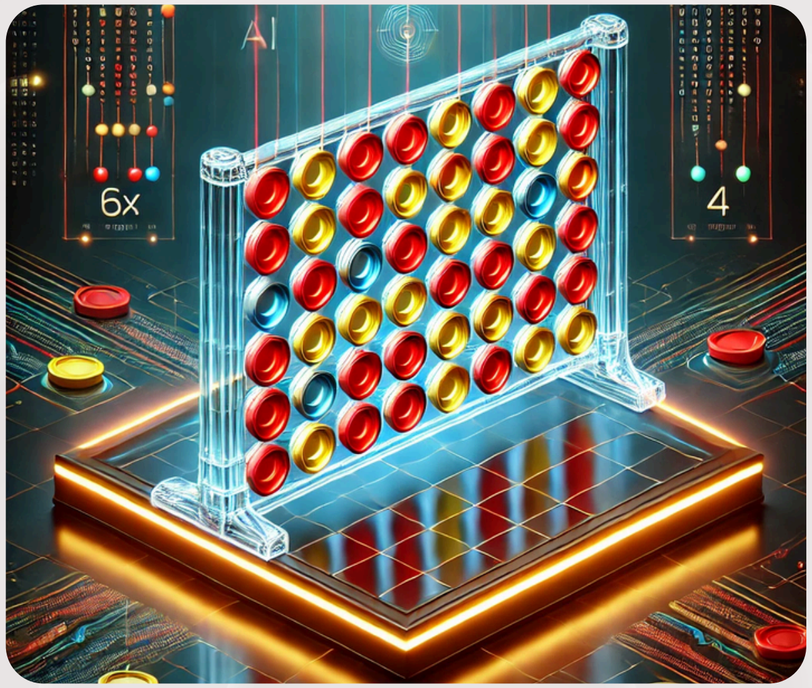
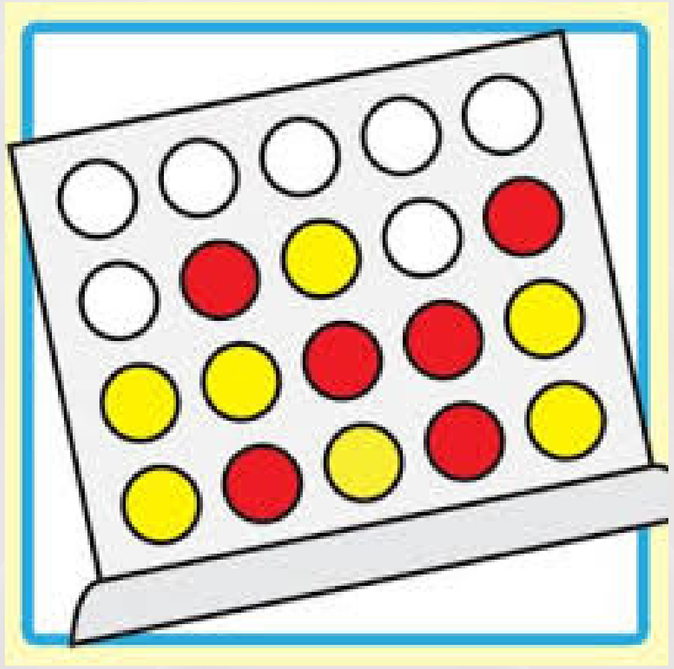
CS246 Artificial Intelligence
Group Project
Fall 2024

Connect – 4



Anna Asatryan
Ruzanna Karamyan
Nane Hayrapetyan
Armine Babajanyan

Introduction

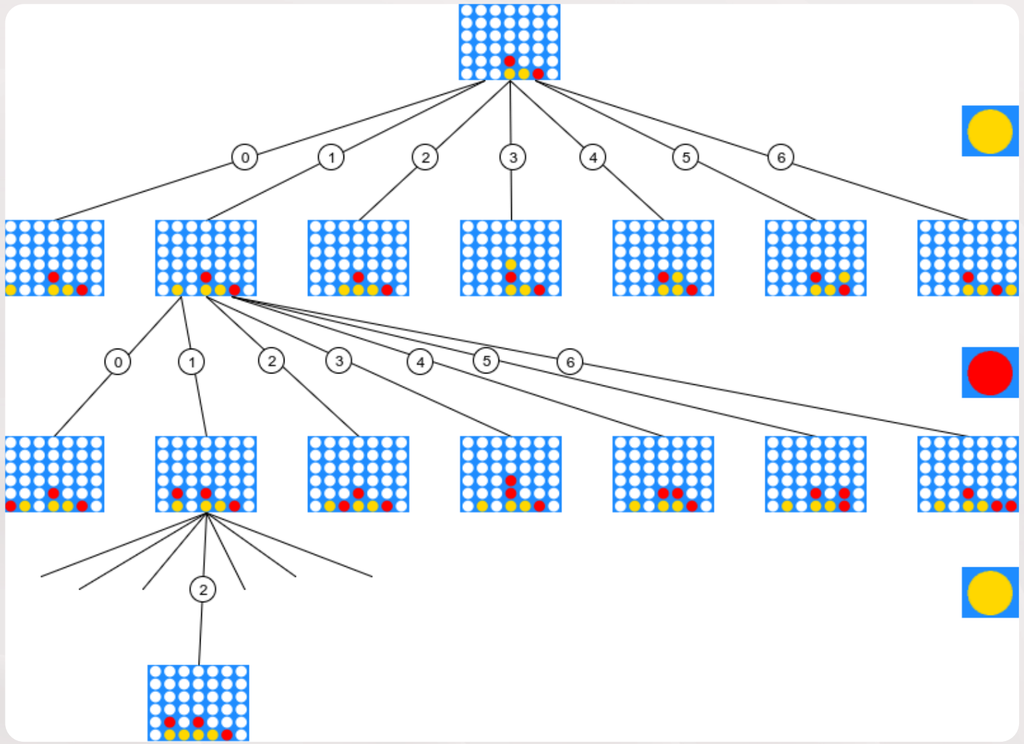


The Game

- **Players**: 2 players - one using yellow discs, one using red discs
- **Board**: 6 rows, 7 columns
- **Game**: players take turns dropping discs into the columns
- **Goal**: A player wins by connecting four discs vertically, horizontally, or diagonally.

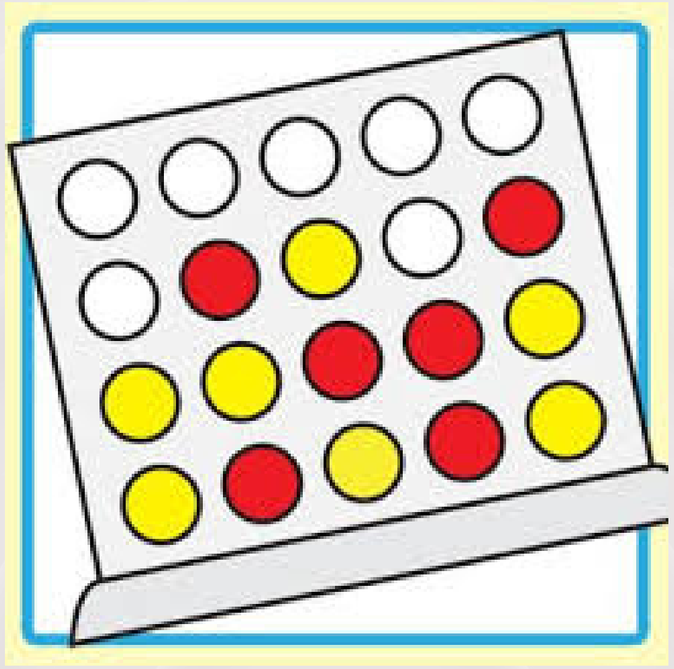
Problem Setting

- **State Space**: 42 cells each can be empty or occupied by red or yellow => 3^42
- **Initial State**: empty board
- **Goal State**: straight line of 4 consecutive discs of the same color
- **Actions**: player can choose one of 7 columns (actions decrease as the columns fill up
- **Transition model**: after each move, the chosen column of the board is filled with one disc, which takes place at the lowest possible position
- **Game tree**: the root node of the game tree represents the empty board. Each branch of the tree represents a state as a result of a move made by the player
- **Branching Factor**: 7
- **Maximum Depth**: 42



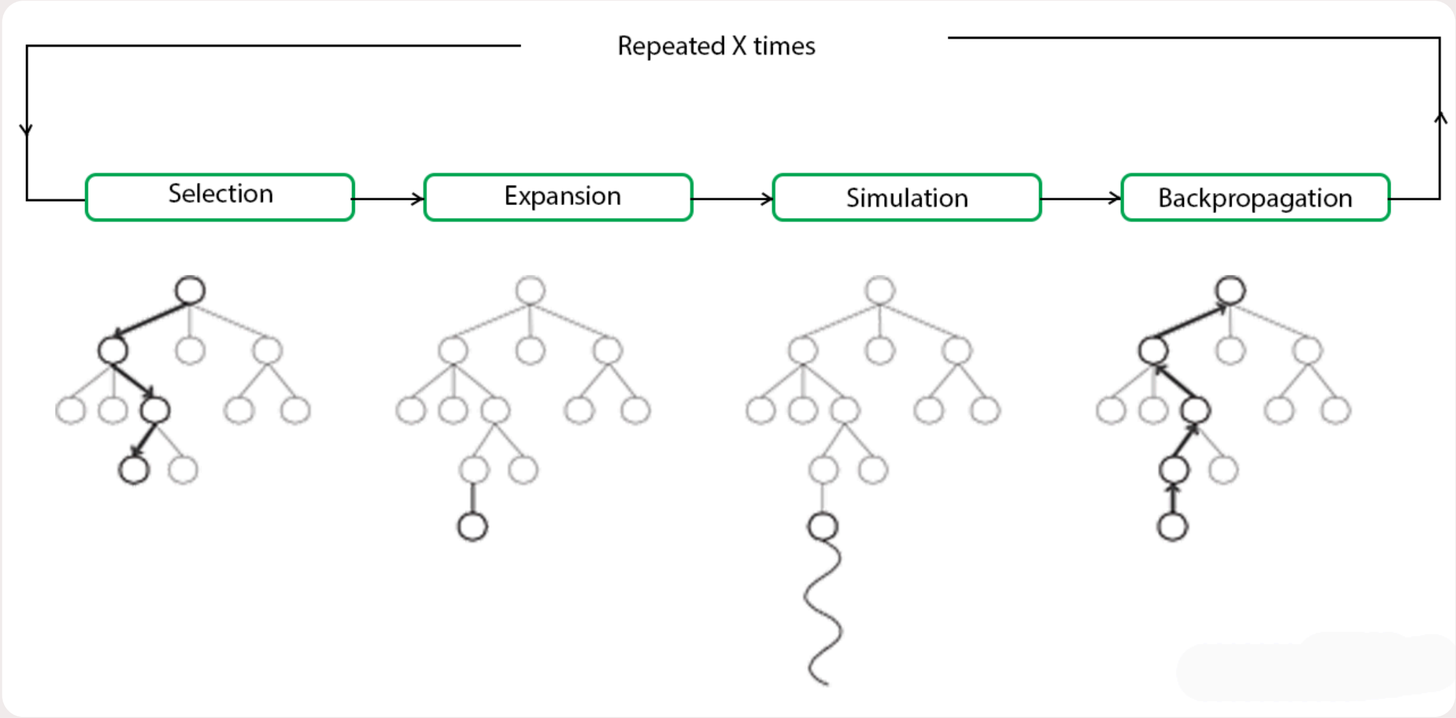
Literature Review

Existing Algorithms for Connect-4



Monte-Carlo Approach

- Large sample of random moves of players is generated
- Probability of winning with each move is estimated.
- The idea is to simulate a large number of board configurations from a given state which allows us to know which states will lead to a better outcome.



Monte-Carlo Approach

Selection

- Aim is to select a leaf node for simulating the outcome
- Upper Confidence Interval policy
- The higher the UCT value of a node, the more likely we will select it

$$UCT(n_i) = \frac{Q}{N} + c_i\sqrt{\frac{logNp}{N}}$$

exploitation exploration

Expansion

- After we select a node, we add all possible actions as children of that node
- This will result in creating new nodes for all possible actions

Simulation

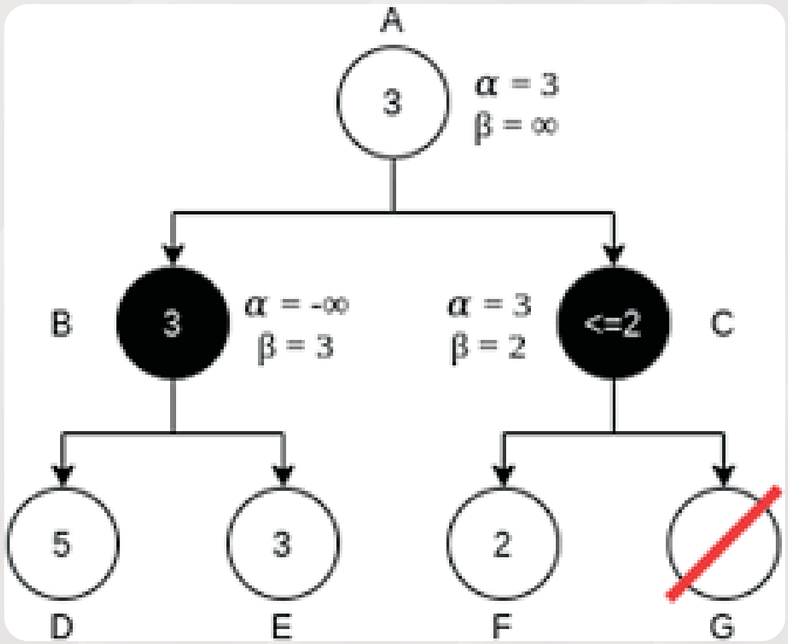
- After expanding all the child nodes, we then select one of them randomly and simulate a game from it where each player plays randomly until we reach a terminal state and the score is calculated.
- We will have either a win (+1), a draw (0), or a loss (-1).

Backpropogation

- We start with a terminal state when the game could be over in a win, loss, or draw.
- Each time we visit a node during a simulation phase, we keep track of how many times that node was visited.
- Win -> increase the node score, loss - decrease, draw - unchanged
- We do these actions on ancestor nodes repeatedly until we reach a root node.

Minimax Algorithm

- Search method that builds a game search tree for the agent in determining the optimal action by predicting future game states up to a specified depth d
- Evaluates the best move in a game by calculating the minimax value



Alpha-Beta Pruning

- Alpha-beta pruning is an optimization technique that enhances the performance of the Minimax search algorithm
- The term pruning refers to the process of cutting off branches and leaves
- A branch or subtree is pruned when the condition alpha ≥ beta is met.

Feature-Based Heuristic

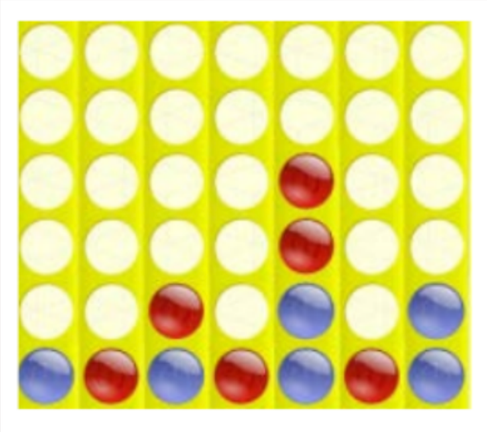
- Specific features the heuristic should find in the states
- Corresponding values to each feature

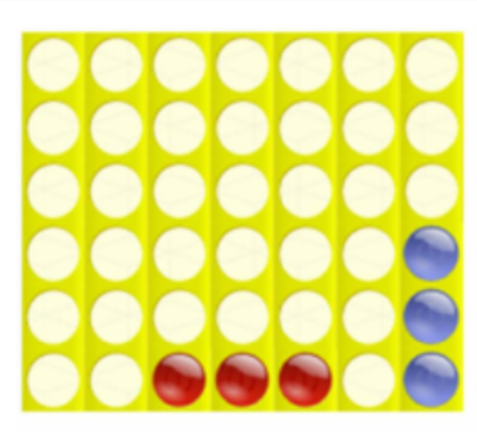
Feature 1: Four discs are connected vertically, horizontally, or diagonally. The feature immediately gets the value of infinity since it already is a win state.

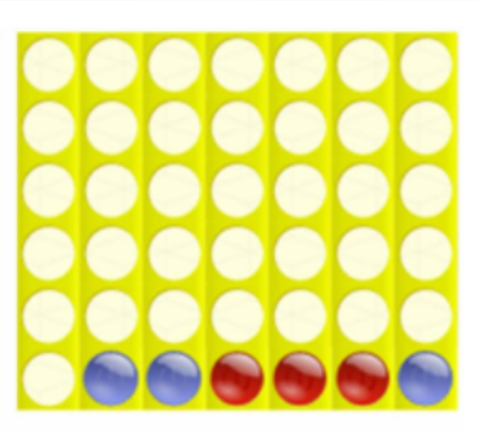
Feature 2: Three discs are connected vertically, horizontally, or diagonally. If a move can be made on either of the immediately adjacent columns, the situation is evaluated as infinity since the players will surely win. If the move can be made only on one adjacent column or the same colored disk can be found a square away from two adjacent discs, the feature gets a score of 900,000 since there is a high chance of winning, but the opponent still might stop it. And if no move can be made on adjacent columns, the feature gets 50,000.

Feature 3: Two discs are connected vertically, horizontally, or diagonally. If a move can be made only on one adjacent column, the feature gets a score of 40,000, 30,000, 20,000, or 10,000 depending on the number of squares available along the direction of the discs correspondingly 5, 4, 3, 2.

Feature 4: A disc is not connected to any other disc in any direction. This feature is evaluated based on the column the disc is on. The highest score of 200 gets the central column. The further the column is fro the center, the lower it gets.





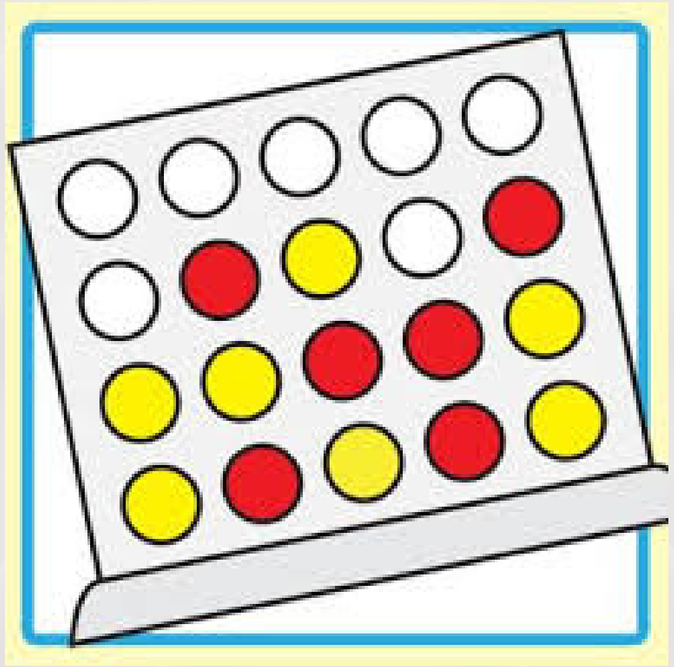


Board-Based Heuristic

- Evaluates each square on the board rather than a specific feature
- The positions closer to the middle column and row are more favorable because they have more options to win

$$\begin{bmatrix} 3 & 4 & 5 & 7 & 5 & 4 & 3 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 5 & 8 & 11 & 13 & 11 & 8 & 5 \\ 4 & 6 & 8 & 10 & 8 & 6 & 4 \\ 3 & 4 & 5 & 7 & 5 & 4 & 3 \end{bmatrix}$$

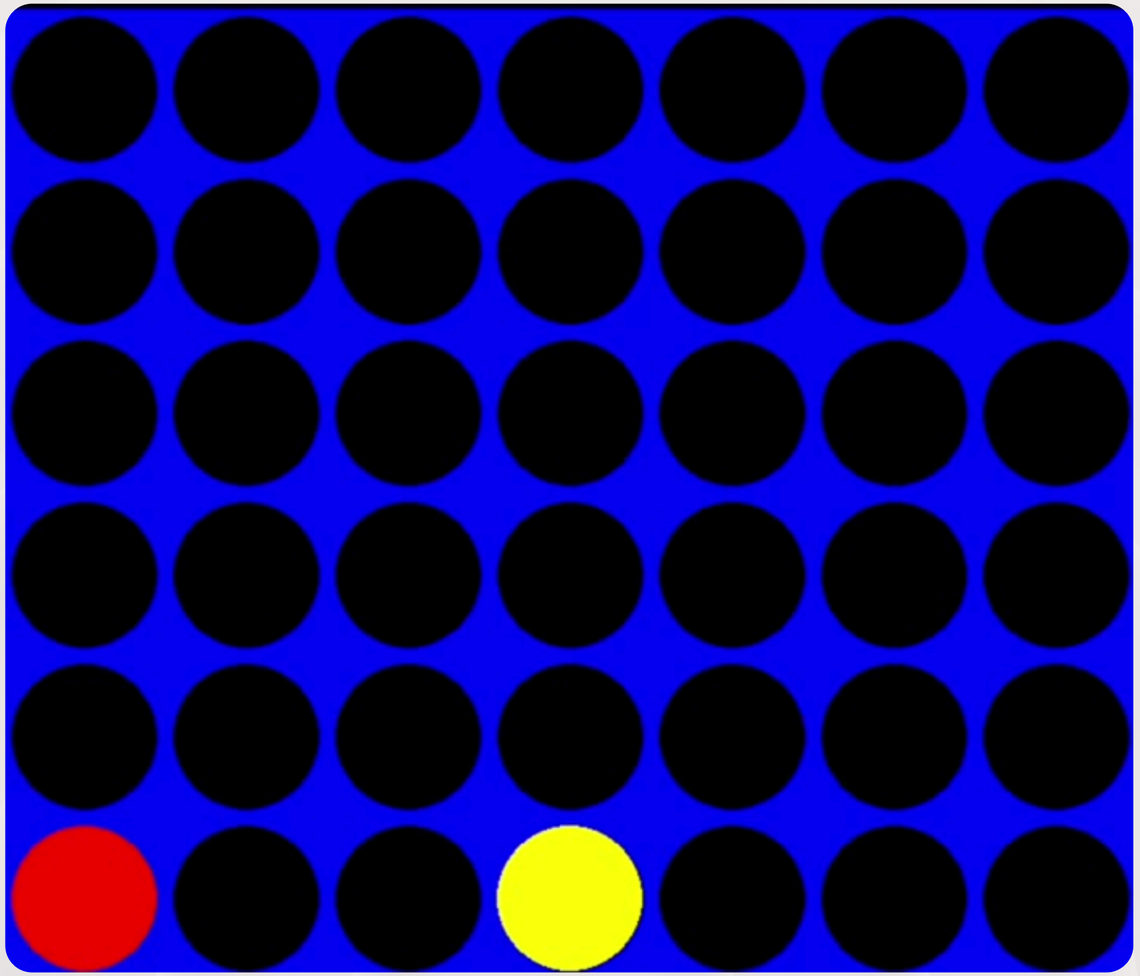
Methodology



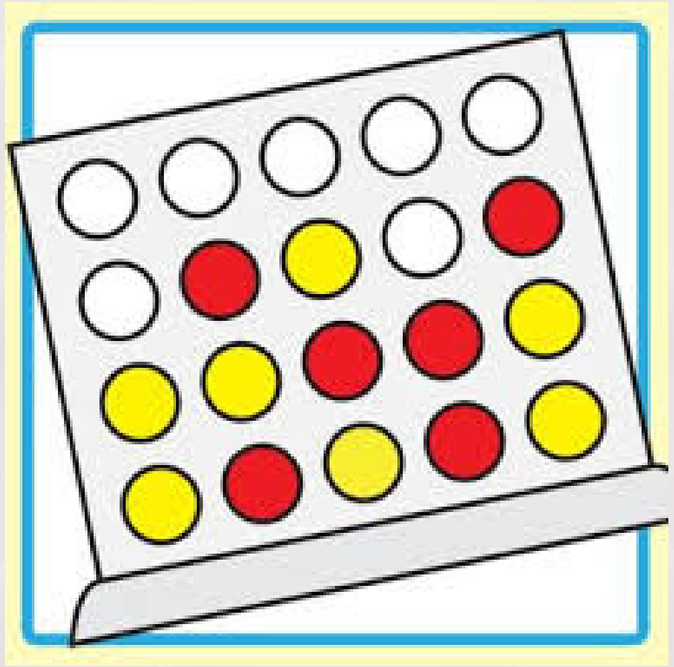
Methodology

Monte-Carlo	Heuristics	Minimax with alpha-beta pruning
• Play against a random agent • Try with 100, 500, 1000, 5000 simulations • Play each 10 times • Compare the proportion of Monte-Carlo Agent winning with different simulations	• Board-based vs Feature-based • Play 10 and 100 times • Compare the proportions of victories • Compare running times and memory usages of 2 heuristics	• Implement Board-based heuristic with minimax algorithm with alpha-beta pruning • Play against different depths

Feature-Based vs Board-Based



Results



Results

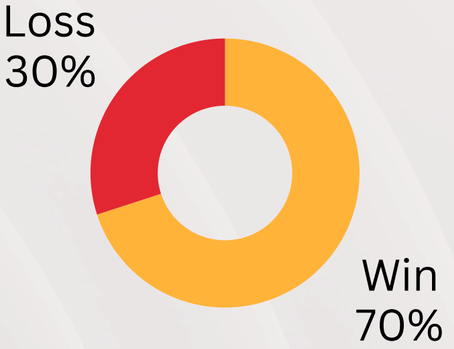
Monte-Carlo

100 simulations

500 simulations

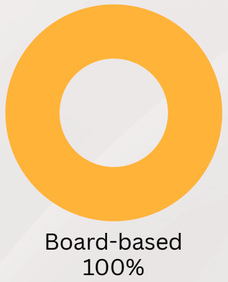
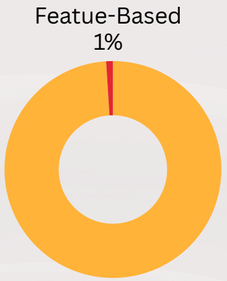
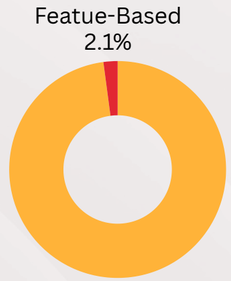
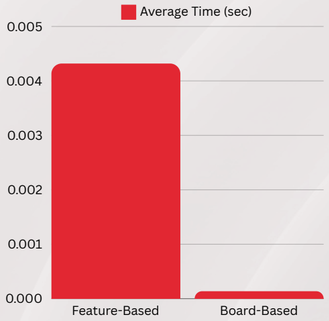
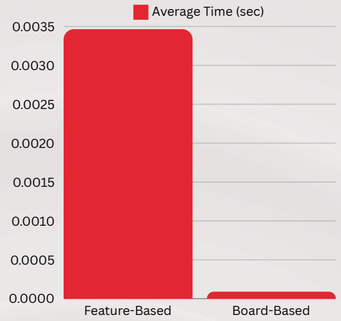
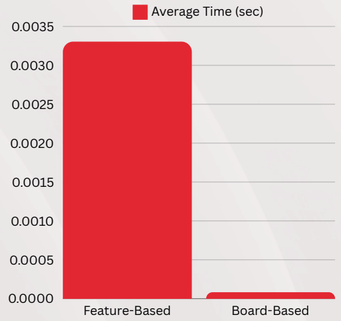
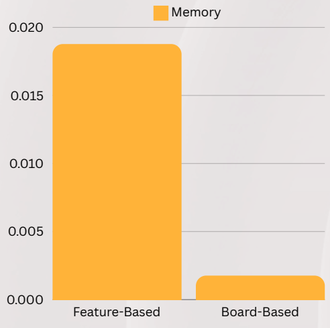
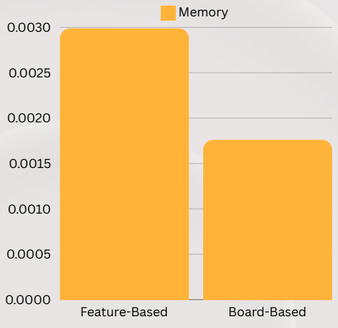
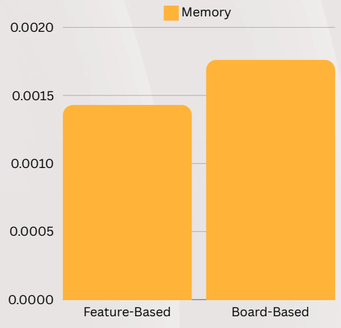
1000 simulations

3000 simulations



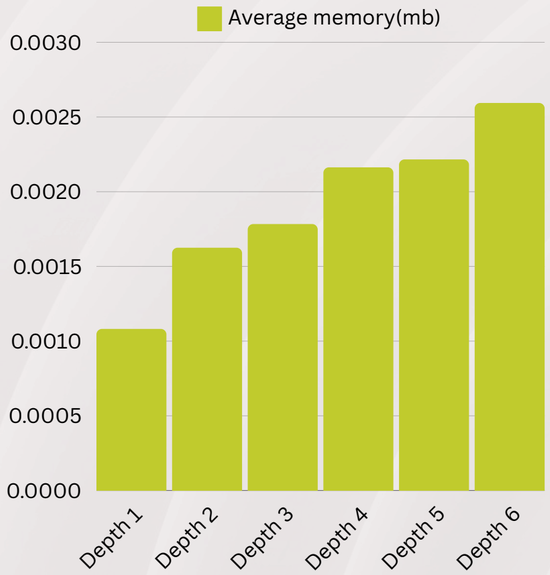
Results

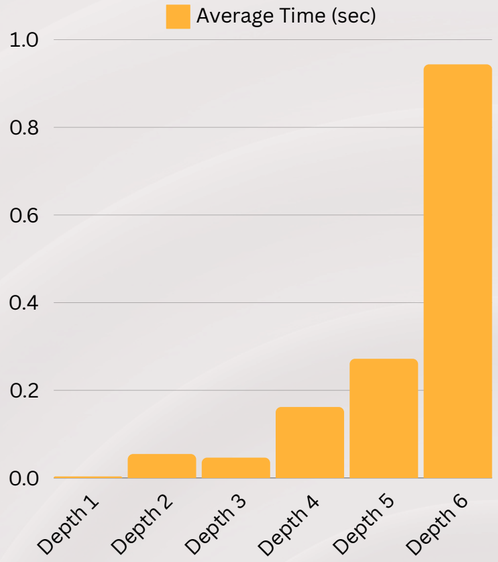
Feature-Based vs Board-Based

	10 plays	100 plays	1000 plays
Winning %			
Time (sec)			
Memory (MB)			

Results

Minimax and Alpha-Beta Pruning Using Board-Based Heuristic

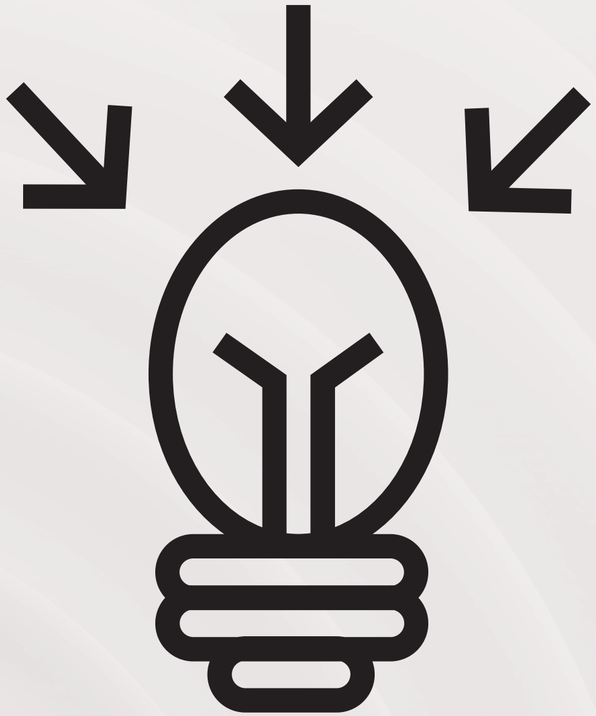




Player 1	Player 2	Player 1 wins	Player 2 wins
Depth 1	Depth 2	5	5
Depth 2	Depth 3	4	6
Depth 3	Depth 4	5	5
Depth 4	Depth 5	0	10
Depth 2	Depth 6	5	5

Conclusion

1. First Player wins most of the time
2. Best choice: Board-Based Heuristic with alpha-beta pruning
3. Depth 5 will be most optimal
4. Blocking Algorithm should be explored
5. Neural Networks should be explored



Thank You!