University of Nottingham
UK | CHINA | MALAYSIA

e

Designing Intelligent Agents

COMP3071 - Final Report :

A Comparative Study of Connect 4 AI Agents: Developing and Evaluating Intelligent Agents for Optimal Gameplay Strategies

---

Date of Completion: 17/05/2023

Student ID	Name
20218675	Muhammad Hanz Affiq Bin Zuki

Table of Content:

Research Title:

A Comparative Study of Connect 4 AI Agents: Developing and Evaluating Intelligent Agents for Optimal Gameplay Strategies

Abstract:

This research will use Connect 4 to emphasize the efficiency of machine learning-based agents and the significance of important elements in achieving the best possible gameplay strategies. The study advances AI in gaming and lays the groundwork for subsequent studies on creating intelligent agents for various strategic games.

Chapter 1 : Introduction

The Connect 4 game is a very popular 2-player strategy game. This game involves 2 players taking turns placing tokens into a 7x6 grid. The goal of the game is to have 4 tokens line up in a row, be it horizontally, vertically or diagonally. The first player to do so will be declared the winner. This game is a solved game which means that the player will be able to win the game by using the optimal amount of moves. Although the games may seem simple, the strategies that are involved are quite complex.

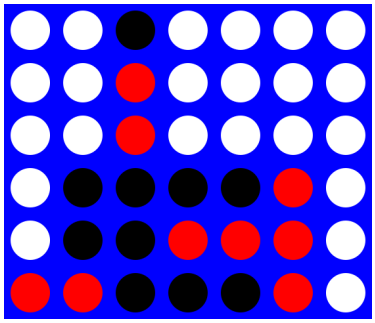

Figure 1: Standard Connect 4 Game Board

The game of "Connect 4" requires players to figure out the best course of action in order to win. It takes a lot of computation power and strategic thinking to solve the problem of determining the best move at every stage of the game. Using brute force search algorithms is currently the state-of-the-art for solving Connect 4 games. These algorithms, however, are computationally costly and cannot be scaled to larger Connect 4 board sizes. In order to solve Connect 4 games in a reasonable amount of time, a more clever and effective solution is required.

The research questions that this research will focus on whether reinforcement learning can be used to solve the Connect 4 games in an efficient manner. We will also focus on how we can improve the accuracy of the Connect 4 game moves by utilizing different search algorithms. Lastly, we will focus on determining how reinforcement learning and different search algorithms will be able to solve Connect 4 games with much higher accuracy and efficiency.

The main objective of this paper is to investigate, develop and compare different artificial agents and their ability to solve Connect 4. Our focus will be to develop 2 separate algorithms and compare their effectiveness in performance of solving the game. One of the artificial intelligent agent algorithms that will be used to solve the game will be the Mini-Max algorithm. The other that will be utilized will be the Monte Carlo Search Tree algorithm.

Overall, this paper will focus on the contribution of development of intelligent agents for solving the Connect 4 game and highlight the strengths and limitations of different artificial intelligent agent approaches.

Chapter 2: Literature Review

a) Mini-Max Algorithm

The minimax algorithm can be explained as a decision making algorithm that is commonly used in decision theory, game theory as well as in artificial intelligences. This algorithm will be best utilized in a two player zero-sum game. The way this algorithm will function is that it will recursively consider all the possible moves along with their corresponding outcomes. It will then choose the moves that will either maximize the minimum gains or minimize the maximum loss of the players. “The aim of the minimax search in the game tree is to find the optimal strategy as a sequence of best possible moves of a given player taking into account possible moves of the other player up to a given depth” (Borovska & Milena, 2007). The minimax algorithm will also be used with alpha-beta pruning in an effort to increase the efficiency by reducing the number of evaluations that is needed. A study by Tommy et al (2017) showed that they implemented the minimax algorithm with AB pruning. This experimental design that is implemented is the study focus on giving a value to the current state of the game board so that the computer will be able to choose the best hole from the remaining holes on the game board to win the game. Different holes are weighted differently and the algorithm will consider the holes with the highest weight in order to win the game. This design overall will account for the possible winnable segments and the weight of the holes based on the possible segments that are able to be created from them as this will provide a systematic method to evaluate the current state of the game board and determine the best move to win the game.

b) Monte Carlo Tree Search Algorithm

The monte carlo tree search algorithm is a “decision-making algorithm that consists in searching combinatorial spaces represented by trees” (Świechowski et al, 2022). The way the monte carlo tree search algorithm works is that it will build a search tree with the possible moves and their corresponding outcomes. It will firstly start off with the current state of the game and then will begin to generate a high amount of random game simulations from that specific state. The simulations that are made will contain a series of random moves that are made by each player until the game will reach a terminal state which means a loss, win and or draw result. Once the simulations are created, the results are then back propagated up the search tree, while

updating the statistics of each move that has been made in the simulation. These include moves such as the number of times that move has been made and or the success rate of the specific move. These statistics will then be used to make the best move in that current state. Exploration and exploitation will then be traded off to determine the next move that will be simulated. This will allow the algorithm to be able to balance the new unexplored moves while also exploiting moves that have a high success rate. The monte carlo tree search is more useful in games that require a high number of possible moves such as Connect 4. A research conducted by Dabas et al (2021) was able to utilize MCTS along with other agents to solve Connect-4.

c) Differentiation

Our suggested approach for solving Connect 4 games combines reinforcement learning and various search algorithms. By making the game's computations simpler and the moves more precise, our method will counteract the drawbacks of existing solutions. The goal is to implement reinforcement learning to learn from its mistakes and develop its decision-making skills over time. Reinforcement learning is a subset of machine learning that is concerned with teaching agents how to perform particular tasks using a system of rewards and penalties. Agents that learn through reinforcement receive both positive and negative reinforcement for their actions. They develop behaviors that lead to the greatest accumulation of positive rewards in this way. (Buro et al, 2021). Additionally, our model will be able to explore the game tree more effectively and identify the best moves more quickly by utilizing various search algorithms.

Chapter 3: Experiment

a) Environment

For this research, there will be 2 different environments that will be utilized to test the effectiveness of the Connect 4 Solver. The first will be a Connect 4 solver that will utilize the Minimax Algorithm. The second will be a Connect 4 solver that will utilize the Monte Carlo Tree Search Algorithm.

i) Minimax Algorithm Agent:

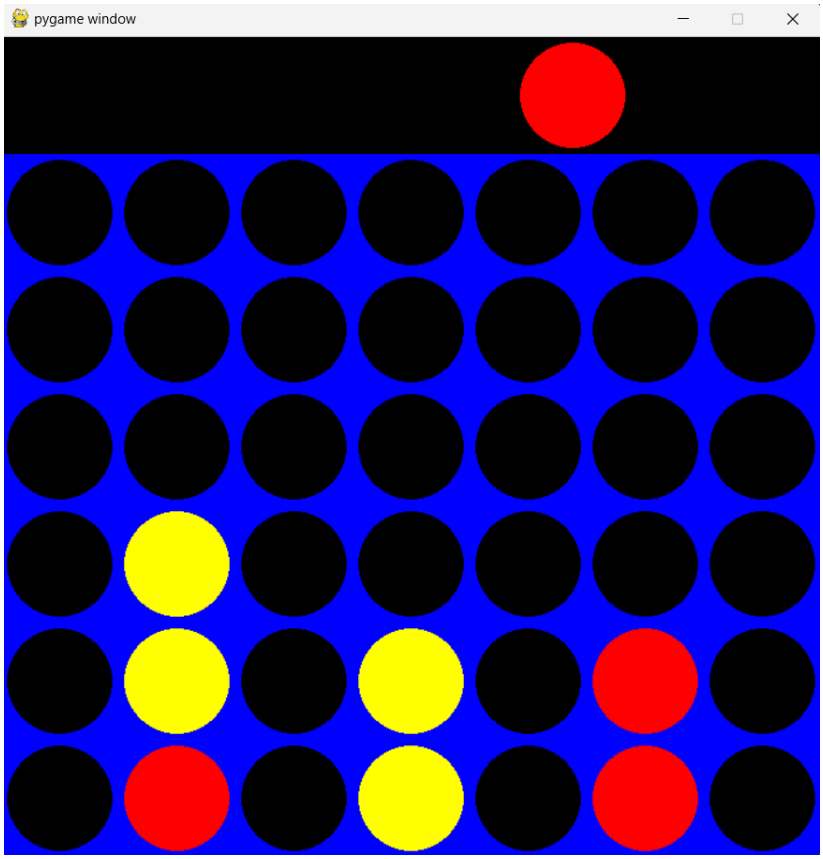

Figure 2: Connect 4 Minimax Algorithm Agent

ii) Monte Carlo Tree Search Agent:

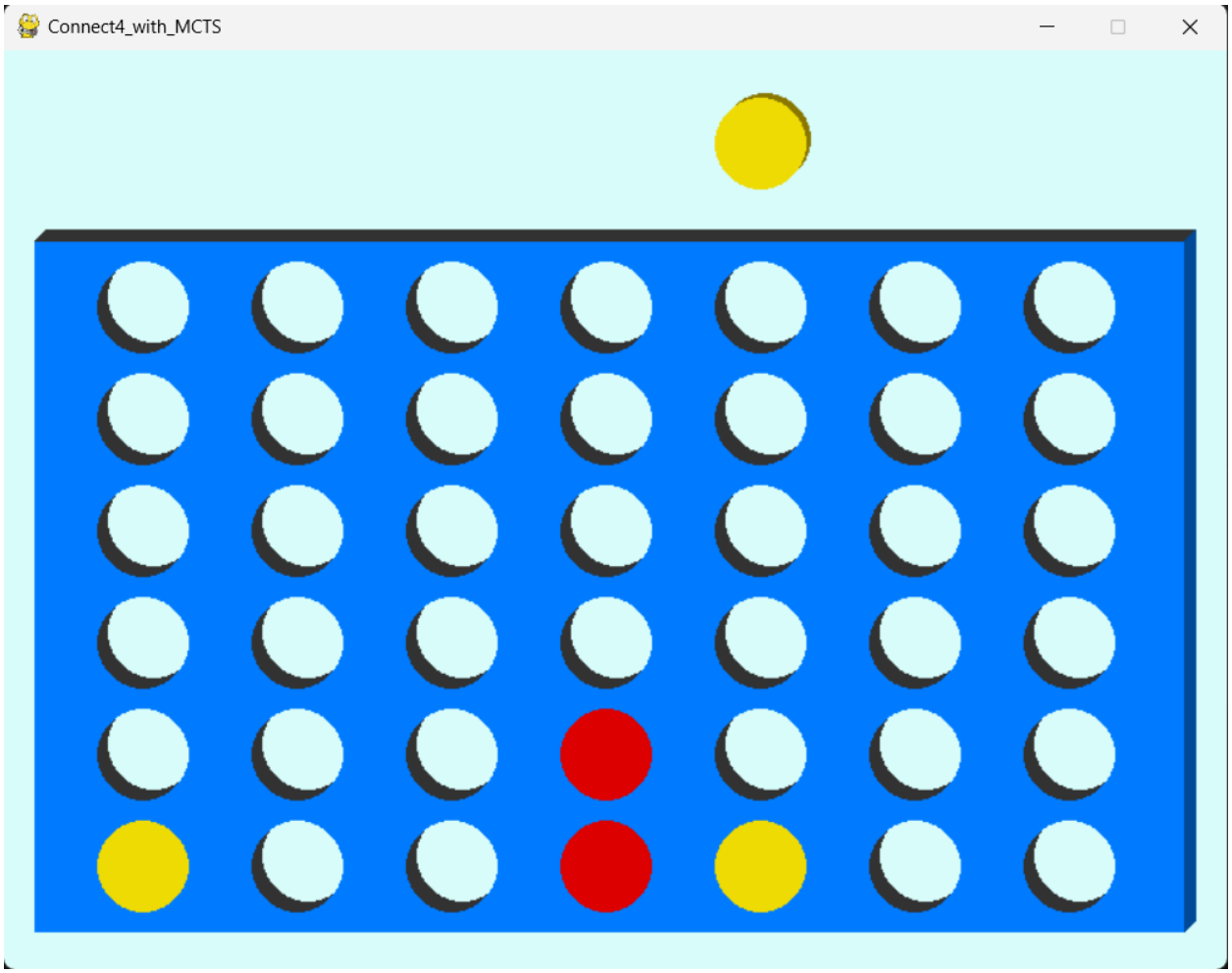

Figure 3: Connect 4 MCTS Algorithm Agent

b) Methodology

The methodology behind the Minimax Algorithm Agent will utilize the minimax algorithm which will begin by simulating thousands of game states to determine the path that will be taken by the 2 players with perfect strategy. This algorithm makes use of game states which will be represented as nodes in a game tree and will then be analyzed by a scoring function. The algorithm will then only determine a score once the terminal node is achieved. The board will be able to be determined as positive if the maximiser is able to succeed and will be determined as negative if the minimiser is able to succeed.

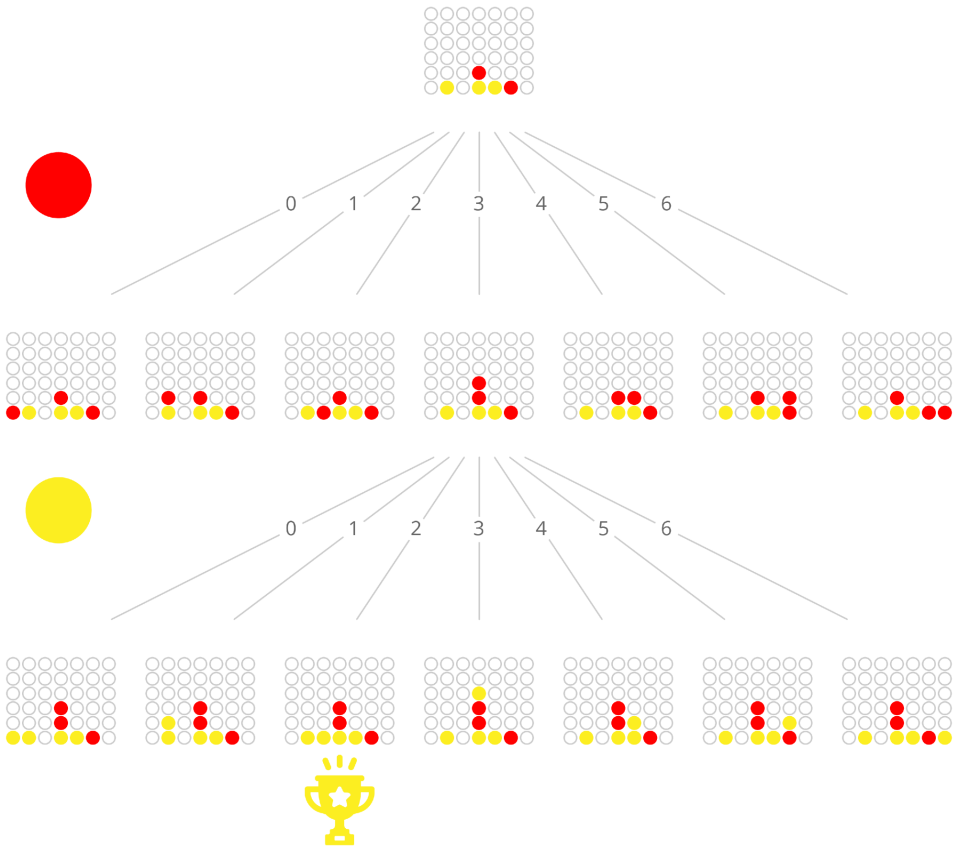

Figure 4: Minimax Algorithm Game Tree

However, if we only implemented the minimax algorithm into the agent, the sheer number of simulations will take too much time to solve the board. In order to assist with this issue, we will also include the use of alpha beta pruning in this algorithm. The alpha beta pruning will function by adding two additional parameters which are alpha and beta. These new parameters will then be monitored during any computation throughout the process. The alpha-beta introduces a score window [alpha;beta] where we are able to look up a position's true score. When the real score is outside the search windows, the restriction on computing the precise score is relaxed. The alpha-beta function should return the precise score if the position's real score falls inside the range. If the position's actual score is less than alpha, the alpha-beta function may return any upper bound of the position's actual score that is less than or equal to alpha.On the other hand, if the position's actual score exceeds beta, the alpha-beta function is permitted to return any lower bound of the position's actual score that is larger than or equal to

beta.This function will then allow us to significantly reduce the number of explored nodes which in turn allow the solver to perform more efficiently.

On the other hand, the methodology of the Monte Carlo Tree Search Agent will utilize the algorithm to firstly build a search tree with n nodes. The nodes will then be defined with a win count and visit count. The tree will initially start with a single root node and will continually perform iterations until it can no longer do so. The algorithm will begin the initial setup by "starting it with a single parent root node and assign a large random UCB value to each non-visited child node" (Agarwal, 2020). The next four phases will be the main structure of the Monte Carlo Tree Search algorithm.

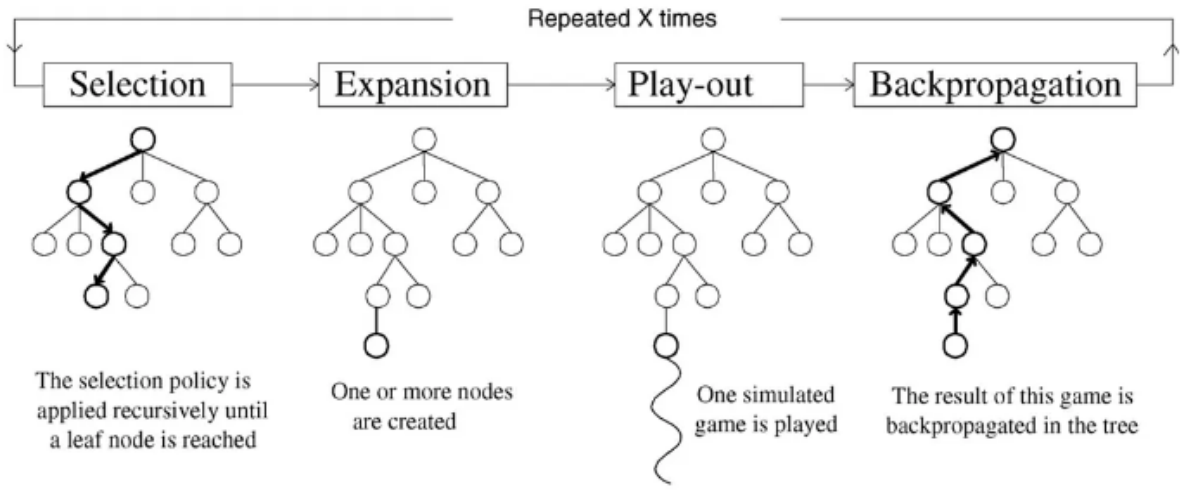

Figure 5: Monte Carlo Tree Search Phases

The first phase will be the selection phase. During this phase, the agent will begin in the root nodes and choose the most urgent node. Once the most urgent node has been selected, the algorithm will then apply the actions and this process will continue till the terminal state has been achieved. The Upper Confidence Bound for tree will be used to decide the most urgent nodes. The goal of the UCB is to "approximate the (true) game-theoretic value of the actions that may be taken from the current state" (Browne et al ,2012). This will allow us to overcome the issue of exploration and exploitation.

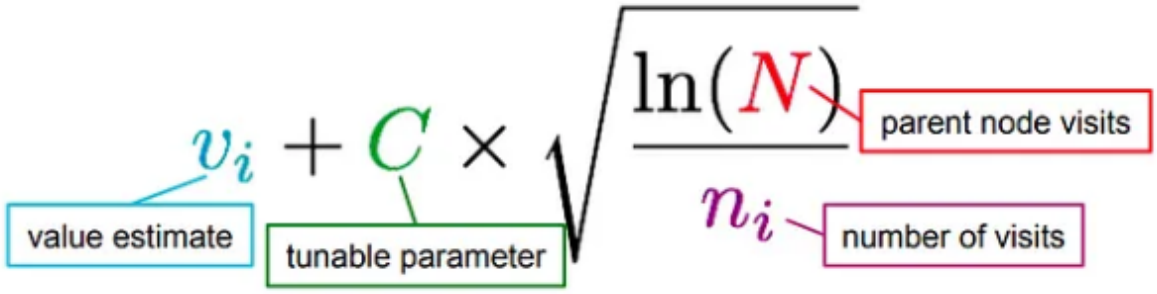

Figure 6: Upper Confidence Bound formula

Next phase will be the expansion phase. This phase is when the UCB cannot be used to find the next node, the game tree will increase even further to include unexplored child nodes by appending all possible nodes from the leaf nodes. Once that is done, the Simulation phase will begin. This phase will select the child node randomly or with a specific criteria until the final stage of the game is achieved. The fourth phase will be the backpropagation phase in which the algorithm agent will reach the final state of the game board with a winner. The traversal nodes are then updated along with the win and visit score of each of the nodes. Finally, the root node's child with the most visits is chosen as the next action because the more visits, the higher the UCB.

```
# Main function for MCTS move computation
def compute_move(self, root):
    time0 = time.time()
    while(time.time() - time0) < self.t:
        # Selection and Expansion Phase
        leaf = self.select(root)
        # Simulation Phase
        simulation_result = self.rollout(leaf)
        # Backpropagation Phase
        self.backpropagate(leaf, simulation_result)
    # from next best state get move coordinates
    selected = self.best_child(root)
    for j in range(6):
        for i in range(7):
            if selected.board[j][i] != root.board[j][i]:
                return (j, i)
```

Figure 7: Implementation of MCTS phases

c) Experiment

The experiment will focus on utilizing 2 different artificial intelligent agents of the Connect 4 solver. As the game of Connect 4 requires 2 different players to successfully play a game, these agents will require another player. To fulfill this criteria, we have decided to use the artificial agents to compete against human participants to identify and compare the results. The experiment will consist of 20 participants to play against the Connect 4 solvers algorithms. The moves taken by the user along with the time taken for each participant to complete a game on each algorithm will be recorded. The outcome of the game will also be recorded to determine the efficiency of the algorithm against humans.

The experiment will proceed as follows:

1. The participants will play a Connect 4 game against the agent with the Minimax algorithm.
2. The time taken, moves taken and game outcome will be recorded.
3. The participants will then play a Connect 4 game against the agent with the Monte Carlo Tree Search algorithm.
4. The time taken, moves taken and game outcome will be recorded.
5. This process will be repeated for the subsequent participants.
6. The results will then be compared between each other.

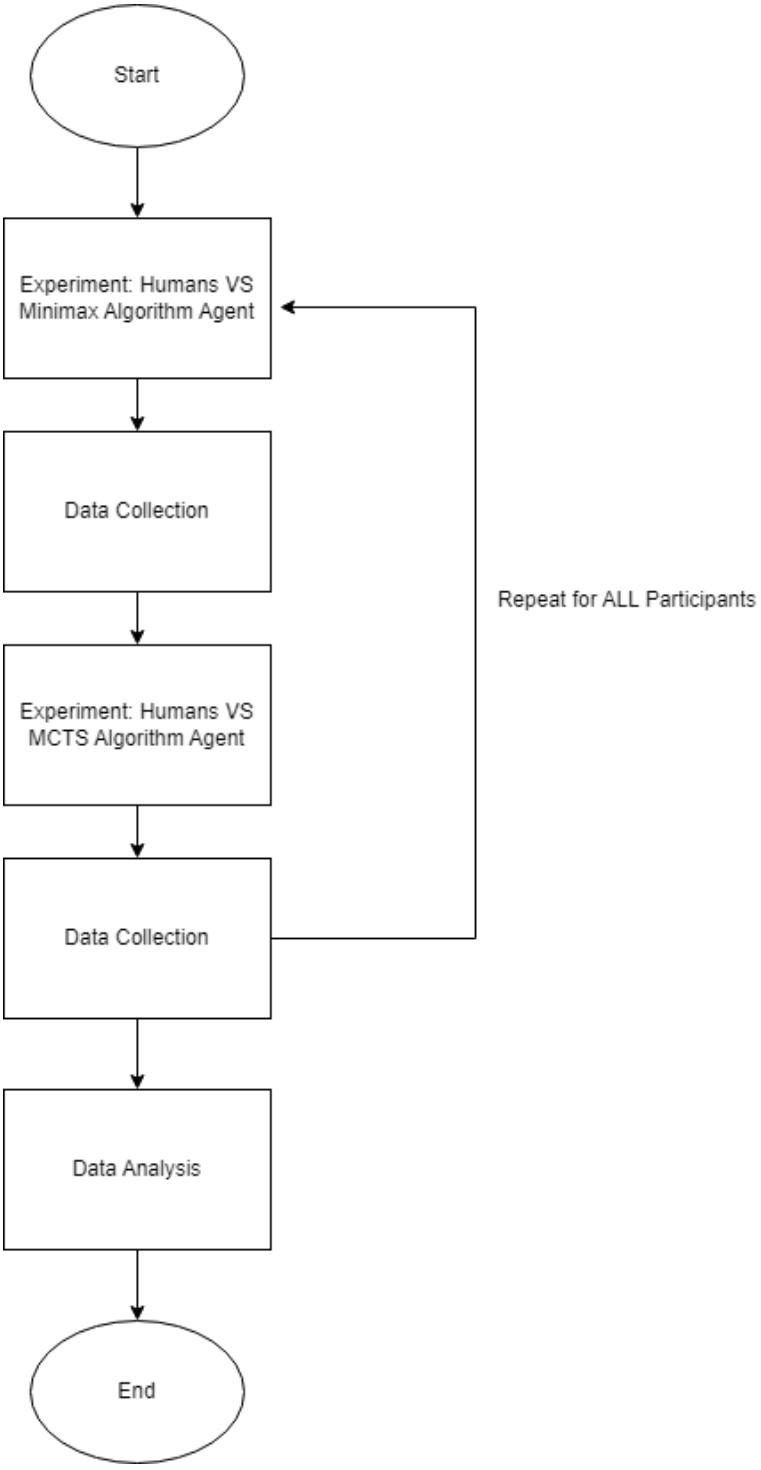

Figure 8: Flowchart for Experiment Process

Chapter 4: Result and Discussions

Experiment: Human Competition Experiment

Environment: Minimax Algorithm Agent

Player No.	Time Taken (m:s.ms)	Moves Taken (Pieces Placed)	Win/ Lose
1.	00:16.21	6	Lose
2.	00:46.82	9	Lose
3.	00:34.40	10	Lose
4.	00:32.59	7	Lose
5.	00:29.30	4	Lose
6.	00:11.65	6	Lose
7.	00:11.98	6	Lose
8.	00:46.78	9	Lose
9.	00:23.61	7	Lose
10.	00:49.48	7	Lose
11.	00:30.51	9	Lose
12.	00:20.65	7	Lose
13.	00:55.90	11	Lose
14.	01:20.33	14	Lose
15.	00:52.76	9	Lose

16.	**01:27.10**	**10**	**Lose**
17.	**00:33.09**	**8**	**Lose**
18.	**02:58.92**	**16**	**Lose**
19.	**00:45.52**	**8**	**Lose**
20.	**02:11.52**	**14**	**Lose**

Table 1: Minimax Agent Results

Average Statistics for Minimax Algorithm Agent

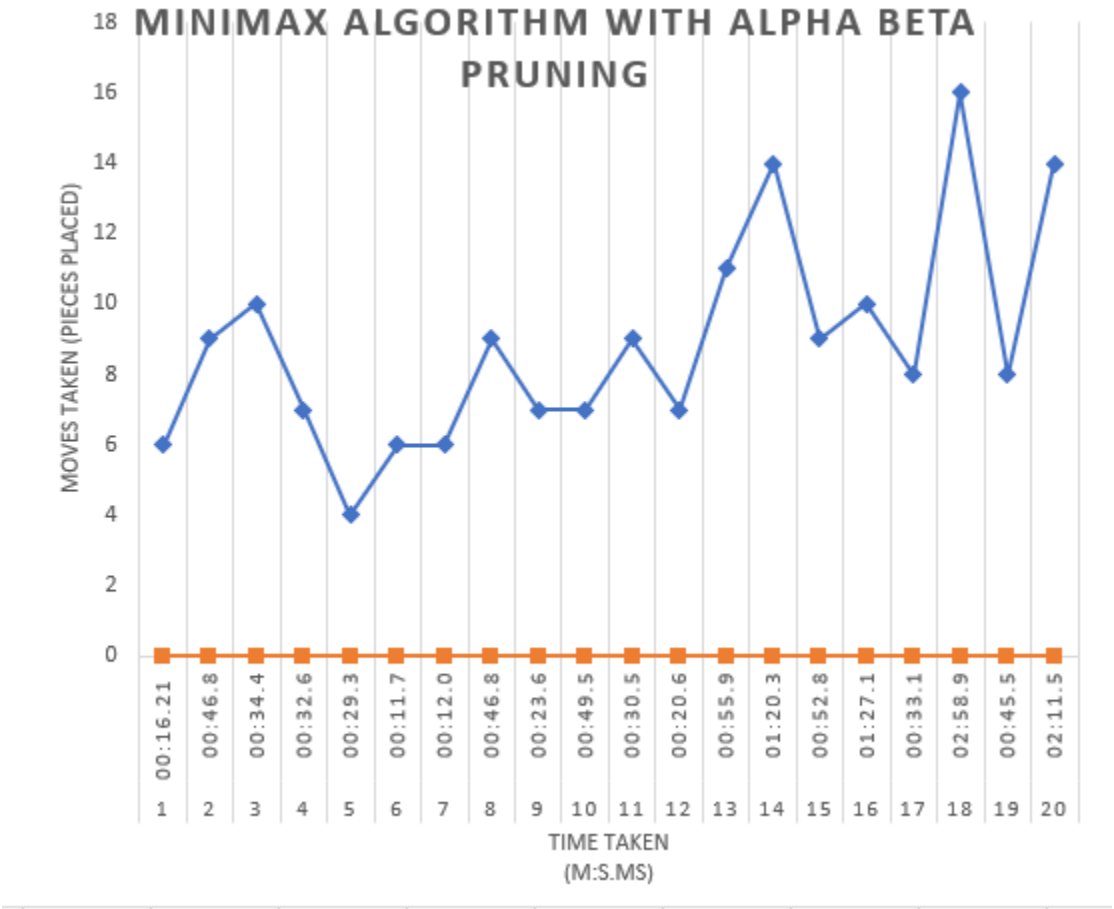

Figure 9: Graph of Results of Minimax Agent

Average Time Taken	Average Moves Taken	Rate of Success
00:50.08	**9.35**	**100%**

Table 2: Average Minimax Agent Results

For the first experiment, the results of the minimax algorithm with alpha beta pruning agent against 20 participants was able to show that the average time taken for the players to lose were about 50.08 seconds. The experiment also showed that the average moves taken by the human participants before losing the game were 9.35. The success rate of the minimax algorithm agent is 100% against every human participant.

Environment: Monte Carlo Tree Search Agent

Player No.	Time Taken (m:s.ms)	Moves Taken (Pieces Placed)	Win/ Lose
1.	**01:36.45**	**10**	**Lose**
2	**02:48.87**	**13**	**Lose**
3.	**00:28.74**	**3**	**Lose**
4.	**00:29.23**	**3**	**Lose**
5.	**01:41.76**	**8**	**Lose**
6.	**02:02.32**	**8**	**Lose**
7.	**02:47.03**	**20**	**Lose**
8.	**01:48.75**	**11**	**Lose**
9.	**03:14.49**	**19**	**Lose**
10.	**01:51.05**	**11**	**Lose**

11.	**00:51.59**	**6**	**Lose**
12.	**01:46.82**	**11**	**Lose**
13.	**01:09.95**	**7**	**Lose**
14.	**04:10.72**	**20**	**Lose**
15.	**02:12.25**	**11**	**Lose**
16.	**02:32.82**	**9**	**Lose**
17.	**02:08.38**	**13**	**Lose**
18.	**01:46.40**	**8**	**Lose**
19.	**00:54.71**	**7**	**Lose**
20.	**01:47.68**	**13**	**Lose**

Table 3: Monte Carlo Tree Search Agent Results

Average Statistics for Monte Carlo Tree Search Algorithm Agent

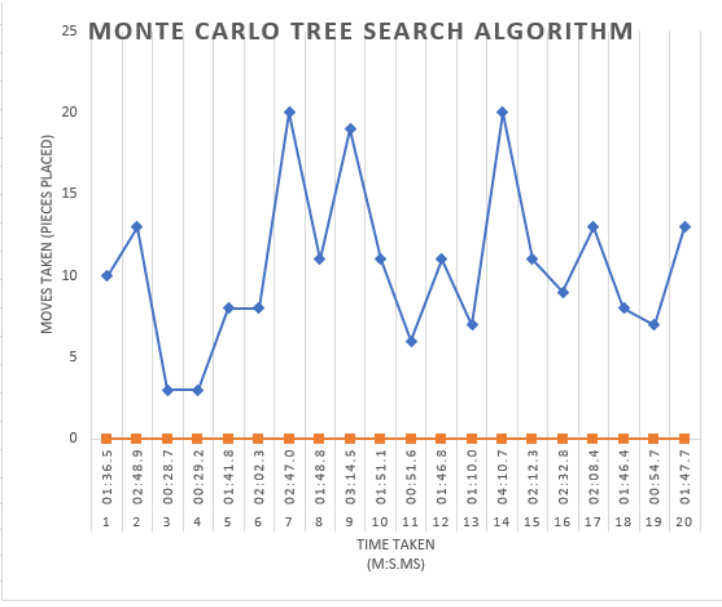

Figure 10: Graph of Results of Monte Carlo Tree Search Agent

Average Time Taken	Average Moves Taken	Rate of Success
01:54.34	**12.35**	**100%**

Table 4: Average Monte Carlo Tree Search Agent Results

For the second experiment, the results of the monte carlo tree search agent against 20 participants was able to show that the average time taken for the players to lose were about 1 minute and 54.34 seconds. The experiment also showed that the average moves taken by the human participants before losing the game were 12.35. The success rate of the MCTS algorithm agent is 100% against every human participant as well.

Comparison and Analysis of Results

This experiment shows that the agents were successful in completing the Connect 4 experiment against human participants. Both experiments showed that the agents were able to consistently succeed against their human counterparts. However, when comparing the data collected between the minimax agent and MCTS agent, there are very big differences in their results. For example, the average time taken among both of the algorithms are quite different. For the minimax algorithm with alpha beta pruning, the average time taken for the agent to win a game was 50.08 seconds. Meanwhile, the MCTS agent had an average time of 1 minute and 54.34 seconds to win a game. The time difference among these two agents is over a minute. This shows that the efficiency of the minimax algorithm agent was higher in comparison to the MCTS algorithm agent for this specific experiment.

Next, the average moves taken for each of the human participants to lose to both of the Connect-4 solvers are quite contrasting as well. This is because the minimax algorithm with alpha-beta pruning performed with an average of 9.35 moves taken against the agent before the human participants lost. On the other hand, the MCTS agent performed with an average of 12.35 moves taken by the human participants before they lost against this agent. This will also show the efficiency of determining the best game decision of the minimax algorithm agent is stronger in comparison with the game decision making of the MCTS algorithm agent for this specific experiment.

However, among both experiments, there was one similarity among the both of the intelligent agents in which they were able to have a win-rate of 100% against every human participant in the experiment.

To conclude the analysis for these experiments, it is clear that the minimax algorithm agent and the MCTS algorithm agent have successfully performed their respective functions which is to solve the Connect 4 board. However, through experiment, we were able to determine the minimax algorithm agent was able to perform more efficiently in this experiment when compared to the MCTS algorithm agent. The time and moves required to win are significantly lower for the minimax agent than the MCTS agent. This should be due to the fact that the minimax agent includes alpha beta pruning in the algorithm while MCTS agent does not. This shows that the alpha beta pruning plays an extremely significant role in the development of an efficient intelligent agent algorithm.

Chapter 5: Conclusion

In conclusion, both the minimax algorithm with alpha beta pruning and the monte carlo tree search algorithm are great algorithms to use to create an intelligent agent to solve Connect 4. However, the context in which they are used are also incredibly important as both algorithms have their strengths and weaknesses respectively.

For the minimax algorithm with alpha beta pruning, the algorithm will perform very well when utilized in a game board that requires a small search space which allows the algorithm to explore more of the possible moves and search the tree exhaustively to determine the most optimal moves. In a game of Connect 4, the minimax algorithm will be able to accurately evaluate the consequences of each move while making the most optimal decisions to win the game.

On the other hand, the monte carlo search tree also will perform extremely well in games However, the algorithm may not be the most optimal for a game such as Connect 4 as shown by the results. This is due to the fact the the search space for a Connect 4 game board is considerably large which would make searching the tree for the possible game states quite a difficult and inefficient task. However, MCTS would be able to focus on promising branches by simulating and sampling random games which would allow for more efficient exploration.

In the case of Connect 4, both algorithms agents have been successfully applied. Minimax with alpha-beta pruning can be shown to be highly effective if it is implemented in an efficient manner which will take advantage of the game's properties, such as the small board size and limited depth of the game tree. On the other hand, MCTS can also be an excellent choice, particularly when combined with domain-specific knowledge and heuristics. Further improvements to the algorithm can be made to allow the agent to perform in a more efficient manner. In the end, the relative effectiveness of these algorithms can change depending on the implementation details and the constraints of the problem.

References

Agarwal, P. (2020, November 11). *Game AI: Learning to play Connect 4 using Monte Carlo Tree Search*. Medium. https://pranav-agarwal-2109.medium.com/game-ai-learning-to-play-connect-4-using-monte-carlo-tree-search-f083d7da451e

Borovska, P., & Lazarova, M. (2007). Efficiency of parallel minimax algorithm for game tree search. *Proceedings of the 2007 International Conference on Computer Systems and Technologies - CompSysTech '07*. https://doi.org/10.1145/1330598.1330615

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43. https://doi.org/10.1109/tciaig.2012.2186810

Buro, J., Mikkelsen, N. D., Nguyen, A., & Thom, J. (2021). *Implementation of Reinforcement Learning Algorithms for Connect 4*. 1–16. https://www.thejacobthom.com/papers/SENG-474-paper.pdf

Cazenave, T., & Abdallah Saffidine. (2010). *Score Bounded Monte-Carlo Tree Search*. 93–104. https://doi.org/10.1007/978-3-642-17928-0_9

Dabas, M., Dahiya, N., & Pushparaj, P. (2021). Solving Connect 4 Using Artificial Intelligence. *Advances in Intelligent Systems and Computing*, 727–735. https://doi.org/10.1007/978-981-16-2594-7_59

Kang, X., Wang, Y., & Hu, Y. (2019). Research on Different Heuristics for Minimax Algorithm Insight from Connect-4 Game. *Journal of Intelligent Learning Systems and Applications*, *11*(02), 15–31. https://doi.org/10.4236/jilsa.2019.112002

Nasa, R., Didwania, R., Maji, S., & Kumar, V. (2018). *Alpha-Beta Pruning in Mini-Max Algorithm –An Optimized Approach for a Connect-4 Game*. https://www.irjet.net/archives/V5/i4/IRJET-V5I4366.pdf

Świechowski, M., Godlewski, K., Sawicki, B., & Mańdziuk, J. (2022). Monte Carlo Tree Search: A Review of Recent Modifications and Applications. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-022-10228-y

Tommy, L., Hardjianto, M., & Agani, N. (2017). The Analysis of Alpha Beta Pruning and MTD(f) Algorithm to Determine the Best Algorithm to be Implemented at Connect Four

Prototype. *IOP Conference Series: Materials Science and Engineering*, *190*, 012044. https://doi.org/10.1088/1757-899x/190/1/012044