

# Determinism in Connect 4 using Monte Carlo Tree Search

Ebenezer Lobe Boyom

*Departments of Mathematics, Computer Science Department of Mechanical Engineering Department of Computer Science*  
Stanford University  
Stanford, CA  
twiggy20@stanford.edu

Romeo Garcia Jr.

*Stanford University*  
Stanford, CA  
rgj0@stanford.edu

Esteban Barrero-Hernandez

*Stanford University*  
Stanford, CA  
ejbh24@stanford.edu

**Abstract**—The strategic depth of board games has long challenged the development of artificial intelligence, culminating in groundbreaking applications like Google DeepMind’s AlphaGo, which utilized Monte Carlo Tree Search (MCTS) to achieve unprecedented success in Go. This study extends the exploration of MCTS to the realm of Connect 4, a game known for its strategic complexity despite simpler rules compared to Go. We implemented MCTS in a simulated Connect 4 environment, examining its effectiveness against various opponents, including an algorithm making random moves, the algorithm competing against itself, and human players. The results underscore MCTS’s robustness in game decision-making, reflecting a performance that not only surpasses random strategies but also offers significant challenge to human strategies. This research not only demonstrates the versatility of MCTS in different strategic contexts but also echoes its potential for broader applications in complex decision-making scenarios, reminiscent of its historic success in AlphaGo.

**Index Terms**—Monte Carlo Tree Search, Connect 4, Game Strategy, Artificial Intelligence, Heuristic Algorithms, Board Game Analysis, Computational Game Theory

## I. INTRODUCTION

Connect 4 is a two-player connection game in which players alternately drop colored discs into a seven-column, six-row vertically suspended grid. The objective is to be the first to form a horizontal, vertical, or diagonal line of four of one’s own discs. Connect 4 is a solved game, with a perfect play from both players leading to a draw, assuming the first player starts in the center column. The simplicity of the rules of Connect 4, coupled with the complexity of its strategic depth, makes it an ideal subject for AI and game theory research. It presents a clear case of a zero-sum game with finite possibilities, which can be fully mapped out to determine optimal strategies. This mapping can be represented as a decision tree that AI algorithms can traverse. The study of such decision-making processes in games can be extended to complex real-world problems where similar strategic interaction is required.

AI’s role in strategic decision-making is pivotal in an era where data-driven and automated systems are prevalent. In games, AI must evaluate the current state, predict potential outcomes, and make decisions that maximize chances of success. Similarly, in business, military, and other domains, AI systems analyze vast amounts of information to identify patterns, predict future trends, and make decisions that align with strategic

goals. The adaptability of AI in uncertain environments and its ability to process information faster than humans make it a valuable asset in strategic decision-making. AI can consider more variables and potential outcomes than a human, often leading to more informed and objective decisions. MCTS is a heuristic search algorithm for some decision processes, most notably in game play. MCTS combines the classical tree search with random sampling of the search space. It builds a search tree incrementally and evaluates the moves based on random simulations of entire games. The significance of MCTS in AI research lies in its flexibility and efficacy. Unlike traditional search algorithms that require extensive domain knowledge to prune the search tree effectively, MCTS adapts to the available information dynamically, allowing it to be applied to a wide range of problems without substantial customization. This aspect is particularly useful in games with large state spaces or when the decision model is not fully known. MCTS has been successfully applied to complex games such as Go, where its use in programs like AlphaGo has defeated world champions. The application of MCTS goes beyond games, extending to real-world problems such as planning, and optimization. Its importance in AI research is underscored by its balance between exploration (trying new possibilities) and exploitation (using known information), a fundamental concept in AI and machine learning.

## PROBLEM STATEMENT

Deterministic, perfect information games, such as Connect 4, present a unique set of challenges in the field of AI and computational game theory. In these games, the entire state of the game is known to all players, and there is no element of chance involved in determining the outcome of any given play. This total transparency requires strategies that are both comprehensive and exhaustive in nature. The challenge in finding efficient strategies lies in the vastness of the decision tree that represents all possible moves and countermoves. For Connect 4, although the game is theoretically solvable, the number of potential game states is enormous—on the order of trillions [3]. This makes it computationally infeasible to explore every possible state to find an optimal strategy using brute force methods. Moreover, as players alternate turns, each decision must not only consider the current move’s immediate

impact but also its longer-term implications. This foresight requires deep lookahead capabilities and sophisticated evaluation functions to assess non-terminal game states. Given these complexities, AI algorithms must be both smart in pruning the decision tree and efficient in evaluating game states. This is where heuristic approaches, like MCTS, become valuable as they offer a more practical solution compared to exhaustive search methods.

## II. LITERATURE REVIEW

MCTS has been a game-changer in the realm of AI, particularly for games. It gained significant attention after its successful application in computer Go, where traditional algorithms like minimax and alpha-beta pruning were less effective due to the vast complexity and high branching factor of the game [6]. MCTS was pivotal in the development of AlphaGo, the program that famously defeated world champion Go player Lee Sedol.

Since its inception, MCTS has been applied to a variety of games, showcasing its versatility and effectiveness. Some notable examples include:

### A. Board Games:

Apart from Go, MCTS has been employed in other classic board games such as Chess and Shogi, where it has been used to enhance the play of existing AI or create entirely new AI players [8].

### B. Real-time Strategy Games:

Games like 'Starcraft' pose a significant challenge due to their real-time nature and the requirement to manage resources, build units, and conduct combat simultaneously [5]. MCTS helps in managing the overwhelming number of decisions that need to be made in a short amount of time.

### C. Card Games:

In games with hidden information and significant chance elements, like Poker, researchers have adapted MCTS to work in the presence of uncertainty [10]. It has also been applied to more complex card games like "Magic: The Gathering," demonstrating its ability to handle incomplete information and make probabilistic inferences [9].

Over time, MCTS has been optimized for different games through various enhancements. Techniques like Rapid Action Value Estimation (RAVE), Progressive Bias, and domain-specific heuristics are used to improve the performance of MCTS [1]. Additionally, researchers have explored parallelization to allow MCTS to perform more simulations within the same time frame, thereby increasing its decision-making quality [6]. The adaptability and success of MCTS in diverse gaming environments underscore its significance in AI research, highlighting its potential to solve complex decision-making problems in other domains as well.

## III. METHODOLOGY

The Connect4 game is simulated using a class. The grid is represented by a 7-column, 6-row matrix, initialized to an empty state of all 0s. Players, identified as 1 and 2, take turns placing their discs in the grid. There is a set of validPlays, which represents the columns that have yet to be filled. The player who makes the first play is chosen randomly, and currPlayer keeps track of which player's turn it is. The prevPlay tuple also records the most previous play made. Given a column, the function makePlay makes a play and updates the grid. makeHumanPlay allows a human user to input a column (1-7) and make a play, and makeRandomPlay allows a random agent to make a play. After each turn, the function swapPlayers, updates whose turn it is. To check if a player has won after taken a turn, the function fourInARow is called to check if there are 4 consecutive chips placed by the same player anywhere in the grid. prevPlay tells us which player made said winning play. If no one has won, the gridFilled function checks if the game has ended is a tie, a situation in which all the columns have been filled before a player wins. Finally, drawBoard visually represents the game board. A detailed implementation of the MCTS algorithm used in this study, along with the simulation environment for Connect 4, can be found in our GitHub repository at Connect4-MCTS .

### A. Mathematical Foundation of MCTS

MCTS balances exploration and exploitation through four key phases: Selection, Expansion, Simulation, and Backpropagation.

The selection phase involves navigating through the tree from the root node to a leaf node using a policy that balances exploration and exploitation [1]. This is typically achieved using the Upper Confidence Bound applied to Trees (UCT) algorithm. The UCT value of a node is calculated using the formula:

$$UCT = \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}} \quad (1)$$

where  $w_i$  is the number of wins after the  $i$ -th move,  $n_i$  is the number of simulations after the  $i$ -th move,  $N_i$  is the total number of simulations after the parent node, and  $c$  is the exploration parameter [1].

**Monte Carlo Tree Nodes:** The ActionNode class represents a node in the Monte Carlo tree. Each node is associated with a specific action taken in the game and points to its parent node, reflecting the game's sequential nature. For each actionNode, the class records the number of visits, Nsa, and the total reward value accumulated from simulations, Q. The getNodeValue function calculates the node's value using the UCB1 heuristic, balancing exploration of less-visited nodes and exploitation of nodes with high rewards. AddChildNodes allows multiple child actions nodes, representing possible actions to be taken after the current node, to be added simultaneously for exploration.

**Monte Carlo Search Algorithm:** The MonteCarlo class encapsulates the MCTS algorithm. It selects nodes to explore,

simulates game outcomes from those nodes, and updates the tree with the results of these simulations. The `selectActionNode` function guides the tree’s expansion and selection process, choosing the next action based on the UCB1 value. Nodes with higher UCB1 values are prioritized for exploration. This step ensures a balance between exploring new moves and exploiting known advantageous moves. In the `performMonteCarloSearch` function, numerous game scenarios are played out randomly, with the outcomes informing the update of node values in the tree. Each simulated game’s results are back-propagated, refining the visit counts and reward values of the nodes. This process continues until the time limit is reached. The `killerMove` function determines the best move to make from the current game state by selecting the action node with the highest visit count, indicating the most promising move according to the algorithm’s simulations. Finally, the `play` function updates the tree accordingly when a play is performed.

#### IV. EXPERIMENT SETUP

In our experiments, we tested the performance of the MCTS algorithm in the game of Connect4. We devised three distinct experimental scenarios. First, we, human players, played against an AI 5 times, where the AI’s decision-making process was powered by MCTS with a specified search time. In this setup, the human player interacted through a function allowing them to input their chosen column, while the AI utilized MCTS to determine its moves. The search time started low in the first game and increased each game, increasing the difficulty for the human players. Second, we created a scenario where the AI, again using MCTS, competed against a random algorithm, automated agent making non-strategic, random moves, contrasting strategic AI behavior. We also ran this scenario 5 times such that the the search time is increased each game, making the AI agent better. Finally, we set up a duel between two AIs, both using MCTS but with differing search time constraints, to explore how varying the depth and breadth of the search impacted the AI’s gameplay strategy and effectiveness. We ran 5 games, experimenting with different search times for both AI agents.

#### V. RESULTS

The data from Table 1 details the outcomes of Connect 4 games between a human and an AI using MCTS. The AI’s decision-making time varied, with the human winning when the AI had less time (0.01 to 2 seconds) and the AI winning as its time increased (5 to 15 seconds). This suggests that the AI’s performance improves with longer search times, allowing for a more thorough exploration of potential moves. The final board visuals provide insight into the strategic plays, with the AI’s victories in longer searches indicating its enhanced capability to devise winning strategies. Essentially, the results highlight the effectiveness of MCTS in scaling the AI’s performance with increased search time, offering valuable insights into the AI’s potential to adapt and overcome human strategy in strategic board games.

Table 1: Human vs AI Games Results

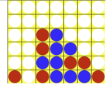
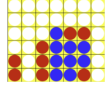
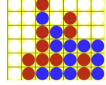
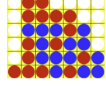
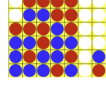
Trial	AI Search Time	Winner	Final Board
1	0.01 seconds	Human	
2	0.2 seconds	Human	
3	2 seconds	Human	
4	5 seconds	AI	
5	15 seconds	AI	

Fig. 1. Results of Human v. AI Trials

In Table 2, we observe the outcomes of Connect 4 games between an AI and a randomly operating robot. The AI’s search time, denoting the length of its decision-making process, varied from an extremely rapid 0.0001 seconds to a slower 5 seconds across five trials. Interestingly, the random robot won the first game with the AI’s quickest decision time. However, as the AI was afforded more time—from 0.05 to 5 seconds—it consistently won the remaining games. The increase in AI search time correlates with a transition from a single loss to consecutive victories, suggesting that even minimal additional processing time can significantly bolster the AI’s performance against a non-strategic, random opponent. The final boards illustrate the culmination of each game, with the AI’s winning sequences becoming apparent as the search time lengthened, demonstrating the effectiveness of a more calculated approach compared to random moves. These results highlight the importance of search time in the AI’s ability to execute winning strategies and the comparative advantage of algorithmic play over random moves in strategic games.

Table 3 presents the results of Connect 4 games where two different AIs competed against each other, with each AI having varying search times to decide on moves. The trials show that neither consistent speed nor slower deliberation guarantees victory. The final board states depicted in the table suggest that the winning AI in each trial may have employed a more effective strategy or been better at adapting to its opponent’s moves. These results could imply that while search time is a factor in an AI’s performance, the efficiency of the search algorithm and the quality of the heuristic evaluation function might play more decisive roles in determining the winner in AI vs. AI games.

Table 2: Random Robot vs AI Games Results






Trial	AI Search Time	Winner	Final Board
1	0.0001 seconds	Random Robot	
2	0.05 seconds	AI	
3	0.2 seconds	AI	
4	2 seconds	AI	
5	5 seconds	AI	

Fig. 2. Results of Randomized Bot v. AI Trials

Table 3: AI vs AI Games Results

Trial	AI 1 Search Time	AI 2 Search Time	Winner	Final Board
1	0.01 seconds	0.03 seconds	AI 2	
2	0.2 seconds	0.8 seconds	AI 2	
3	3 seconds	2 seconds	AI 1	
4	9 seconds	8 seconds	AI 1	
5	15 seconds	16 seconds	AI 1	

Fig. 3. Results of AI v. AI Trials

## VI. DISCUSSION

Table 3 presents the results of Connect 4 games where two different AIs competed against each other, with each AI having varying search times to decide on moves. The trials show that neither consistent speed nor slower deliberation guarantees victory. For instance, in the first trial, AI 2 with a slightly longer search time of 0.03 seconds won over AI 1 with 0.01 seconds. However, as the search times increased for both AIs, the outcomes did not follow a consistent pattern favoring the AI with the longer search time; AI 1 won trials 3 and 4 with one more second of search time compared to AI 2. By the fifth trial, with the longest search times of 15 seconds for AI 1 and 16 seconds for AI 2, AI 1 emerged victorious. The final board states depicted in the table suggest that the winning AI in each trial may have employed a more effective strategy or been better at adapting to its opponent’s moves. These results could imply that while search time is a factor in an AI’s performance, the efficiency of the search algorithm and the quality of the heuristic evaluation function might play more decisive roles in determining the winner in AI vs. AI games.

Moreover, MCTS has emerged as a formidable algorithm in the realm of decision-making processes, particularly in games such as Go, Chess, and Connect 4 [4]. Its application in Connect 4 reveals a myriad of strengths and weaknesses, offering valuable insights into its broader applicability in strategy games and real-world scenarios.

The primary strengths of MCTS in Connect 4 include its adaptive approach to search space exploration, which negates the need for exhaustive examination of all possible moves. This adaptive nature, coupled with its independence from domain-specific heuristics, enables MCTS to effectively learn and apply strategies in Connect 4 without extensive game-specific programming. The algorithm’s capability to balance exploration with exploitation is crucial for adapting strategies in response to an opponent’s moves, while its scalability and flexibility allow for improved decision-making with increased computational resources.

However, MCTS is not without its drawbacks. The algorithm can be computationally intensive, necessitating numerous simulations for optimal decision-making, which may result in slower response times, especially on limited hardware. Additionally, MCTS may struggle in endgame scenarios of Connect 4, where long-term strategic planning is essential, due to its focus on short-term probabilistic decision-making.

The implications of MCTS in the field of strategy games are significant. Its adaptability makes it an ideal candidate for developing AI in various strategy games, providing human-like decision-making capabilities. Beyond gaming, MCTS holds potential in real-world applications such as robotic motion planning, financial forecasting, and optimization problems, where its strengths in handling uncertain scenarios and adaptive learning could be particularly beneficial.

Future advancements and optimization of MCTS are poised to further enhance its effectiveness. Integrating MCTS with machine learning techniques, such as neural networks, could improve performance in learning from past games and predicting opponent moves. Adjustments in the algorithm’s exploration-exploitation balance and simulation processes can lead to more efficient decision-making. Moreover, leveraging parallel processing could significantly reduce computational time, making MCTS more viable for real-time applications. While MCTS demonstrates significant promise in strategy games like Connect 4, its potential in diverse real-world applications continues to be an exciting avenue for exploration and innovation.

## VII. CONCLUSION

In conclusion, the series of trials conducted to evaluate the performance of two distinct AIs in the context of the strategic game Connect 4 reveal that the relationship between an AI’s search time and its success is not linear or straightforward. While one might intuitively expect that longer search times would consistently yield better performance, the data indicates that other factors, such as the efficiency of the search algorithm and the quality of the heuristic evaluations, are equally, if not more, influential in determining the outcome of the games. The

varying results underscore the need for a nuanced approach to AI development, where the focus should be placed on the balance between search time and the effectiveness of the decision-making processes. This insight is particularly relevant for advancing AI capabilities in real-world applications, where strategic and timely decisions are paramount. The findings from these trials contribute to our understanding of AI behavior and point towards the importance of algorithmic quality and adaptability in competitive AI systems.

#### REFERENCES

- [1] M. Kochenderfer, T. Wheeler, and K. Wray, "Algorithms for Decision Making", 1st ed. Cambridge, MA: The MIT Press, August 16, 2022.
- [2] G. B. Holt, "ch09.py," GitHub repository, decisionmaking-code-py, src/ch09.py, 2023. [Online]. Available: <https://github.com/griffinbholt/decisionmaking-code-py/blob/main/src/ch09.py>. [Accessed: 12-08-2023].
- [3] S. Edelkamp and P. Kissmann, "Symbolic Classification of General Two-Player Games," in KI 2008: Advances in Artificial Intelligence, A.R. Dengel, K. Berns, T.M. Breuel, F. Bomarius, and T.R. Roth-Berghofer, Eds., vol. 5243, Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008. [https://doi.org/10.1007/978-3-540-85845-4\\_23](https://doi.org/10.1007/978-3-540-85845-4_23)
- [4] S. Takeuchi, "Comparison of Search Behaviors in Chess, Shogi, and the game of Go," 2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Tainan, Taiwan, 2022, pp. 183-188, doi: 10.1109/TAAI57707.2022.00041.
- [5] D. Soemers, "Tactical Planning Using MCTS in the Game of StarCraft," Doctoral dissertation, Department of Knowledge Engineering, Maastricht University, 2014.
- [6] A. Liu, Y. Liang, J. Liu, G. Van den Broeck, and J. Chen, "On Effective Parallelization of Monte Carlo Tree Search," 2020, arXiv:2006.08785 [cs.LG].
- [7] M. Świechowski, K. Godlewski, B. Sawicki, et al., "Monte Carlo Tree Search: a review of recent modifications and applications," Artificial Intelligence Review, vol. 56, pp. 2497–2562, 2023. [Online]. Available: <https://doi.org/10.1007/s10462-022-10228-y>
- [8] S. Takeuchi, "Comparison of Search Behaviors in Chess, Shogi, and the game of Go," in 2022 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Tainan, Taiwan, 2022, pp. 183-188, doi: 10.1109/TAAI57707.2022.00041.
- [9] P. I. Cowling, C. D. Ward, and E. J. Powley, "Ensemble Determinization in Monte Carlo Tree Search for the Imperfect Information Card Game Magic: The Gathering," in IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 4, pp. 241-257, 2012.
- [10] G. Van den Broeck, K. Driessens, and J. Ramon, "Monte-Carlo Tree Search in Poker Using Expected Reward Distributions," in Advances in Machine Learning: First Asian Conference on Machine Learning, ACML 2009, Nanjing, China, November 2-4, 2009. Proceedings 1, Springer Berlin Heidelberg, 2009, pp. 367-381.