

Tactical Precision versus Statistical Exploration: An Empirical Study of Minimax and Monte Carlo Tree Search in Connect Four

Siyang Tao

College of Science and Engineering, University of Minnesota Twin Cities, Minneapolis, MN, USA

tao00114@umn.edu

Abstract. This study compares Minimax and Monte Carlo Tree Search (MCTS) in Connect Four through systematic AI-versus-AI experiments. We investigate how search budget, heuristic evaluation, and rollout policies affect algorithmic performance across game stages. Results show a positive correlation between search budget and win rate for both algorithms, with MCTS benefiting more from increased simulations. Incorporating domain-specific heuristics yields substantial gains, particularly for Minimax. Stage-based analysis reveals that Minimax with heuristics consistently outperforms MCTS with enhanced rollout policies, while MCTS fails to secure any wins in the opening stage. These findings elucidate the distinct behavioral characteristics of the two algorithms and highlight the critical roles of search depth, heuristic guidance, and game phase in adversarial game-playing.

Keywords: Game AI; Minimax; Monte Carlo Tree Search; Search Budget Optimization

1. Introduction

Connect Four is a two-player connection game played on a 7 x 6 vertical grid, where players alternately drop colored discs into columns with the objective of forming a line of four. Despite simple rules, its state space of approximately 4.5 trillion positions makes it a valuable benchmark for AI algorithm evaluation ^[1]. The game was independently solved in 1988 by Allis, who proved that the first player wins with perfect play from the center column.

This paper compares two foundational game-tree search algorithms: Minimax with Alpha-Beta pruning ^[2], a deterministic approach that explores the game tree to a fixed depth, and Monte Carlo Tree Search (MCTS) ^[3], which uses randomized simulations and the UCT formula to balance exploration and exploitation. These algorithms represent contrasting paradigms: symbolic versus statistical search. We investigate three questions: (1) how search budget affects performance; (2) how domain-specific heuristics improve each approach; and (3) how performance varies across game stages.

2. Methodology

2.1 Game Environment

The experimental framework was implemented in Python, using a Connect Four environment adapted from an open-source repository ^[4], representing the board as a 7 x 6 column-major array with values $\{+1, -1, 0\}$ for RED, YELLOW, and empty cells respectively. Win detection examines eight directions from the last move; terminal states occur on four-in-a-row or board filling.

2.2 Minimax with Alpha-Beta Pruning

The Minimax implementation uses recursive depth-limited search with Alpha-Beta pruning. The plain version evaluates terminal states as $\{+1, -1, 0\}$ for win/loss/draw. The heuristic version adds: (1) center-column preference; (2) immediate win detection (100-point bonus); and (3) opponent threat blocking (120-point penalty).

2.3 Monte Carlo Tree Search

The MCTS implementation uses standard UCT [5]. The plain version employs random rollouts; the heuristic version uses policy-guided rollouts prioritizing center-column moves, immediate wins, and opponent blocking. Backpropagation awards +1 for wins, +0.5 for draws, and +0 for losses.

3. Experiments and Results

3.1 Experiment 1: Search Budget Analysis

Minimax depths {2, 3, 4, 5} were paired against MCTS simulations {100, 300, 500, 1000}, creating 16 configurations (100 games x 3 trials each). The Minimax and MCTS Scores are calculated as followings:

$$\text{Minimax Score} = \frac{\# \text{Minimax Wins} + 0.5 \times \# \text{Draws}}{100}$$

$$\text{MCTS Score} = \frac{\# \text{MCTS Wins} + 0.5 \times \# \text{Draws}}{100}$$

Table 1.: Average performance scores under varying search budgets (3 trials x 100 games each)

Minimax Depth	MCTS Simulations	Minimax Score	MCTS Score
2	100	0.3300	0.6700
2	300	0.1133	0.8867
2	500	0.0900	0.9100
2	1000	0.0533	0.9467
3	100	0.3517	0.6483
3	300	0.1467	0.8533
3	500	0.1583	0.8417
3	1000	0.1150	0.8850
4	100	0.4267	0.5733
4	300	0.2017	0.7983
4	500	0.1117	0.8883
4	1000	0.1083	0.8917
5	100	0.4367	0.5633
5	300	0.1833	0.8167
5	500	0.1700	0.8300
5	1000	0.1517	0.8483

Table 1 shows MCTS consistently dominates plain Minimax (MCTS scores 0.563-0.947 vs. Minimax 0.053-0.437). Increasing Minimax depth improves performance at low MCTS simulations (100-300) but yields marginal gains at higher counts. MCTS shows diminishing returns beyond 500 simulations.

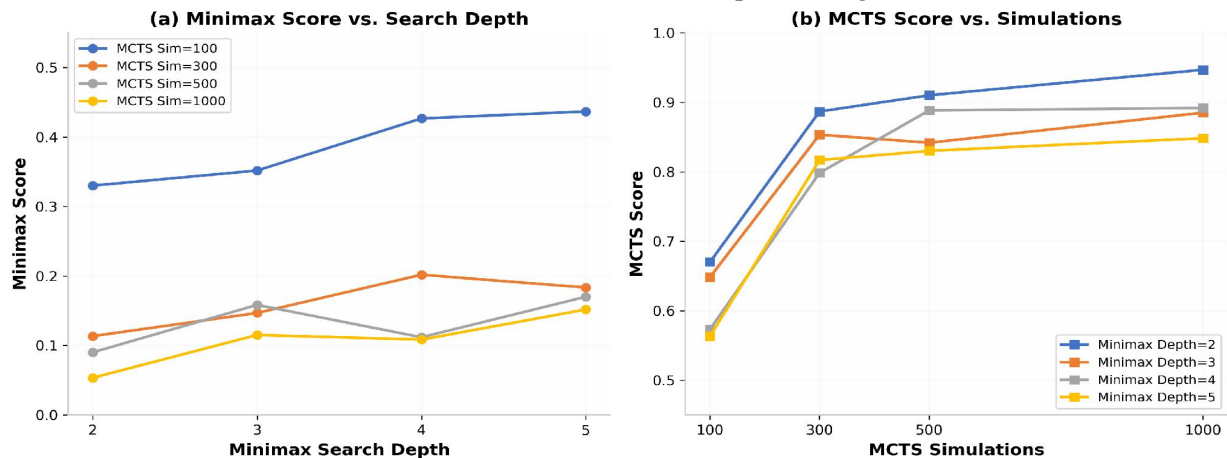


Fig. 1 Figure 1: (a) Minimax score trend under fixed MCTS simulations; (b) MCTS score trend under fixed Minimax depths.

Figure 1(a) shows that Minimax scores generally increase with depth, particularly pronounced at MCTS=100 where the score rises from 0.330 (depth 2) to 0.437 (depth 5). Some fluctuations occur due to the stochastic nature of MCTS random rollouts. Figure 1(b) demonstrates that MCTS scores consistently improve with more simulations across all Minimax depths, with the most dramatic gains occurring between 100 and 300 simulations.

3.2 Experiment 2: Heuristic Enhancement

Using Minimax depth 5 and MCTS 300 simulations, four configurations were tested (100 games x 5 trials each): Plain vs. Plain, Heuristic Minimax vs. Plain MCTS, Plain Minimax vs. Heuristic MCTS, and Heuristic vs. Heuristic. The scores were computed following the same procedure as in Experiment 1.

Table 2: Performance comparison with and without heuristic enhancements (Minimax depth=5, MCTS simulations=300, 5 trials x 100 games).

Setting	Minimax Score	MCTS Score
Plain Minimax vs. Plain MCTS	0.191	0.809
Minimax Heuristic vs. Plain MCTS	0.934	0.066
Plain Minimax vs. MCTS Heuristic	0.067	0.933
Minimax Heuristic vs. MCTS Heuristic	0.848	0.152

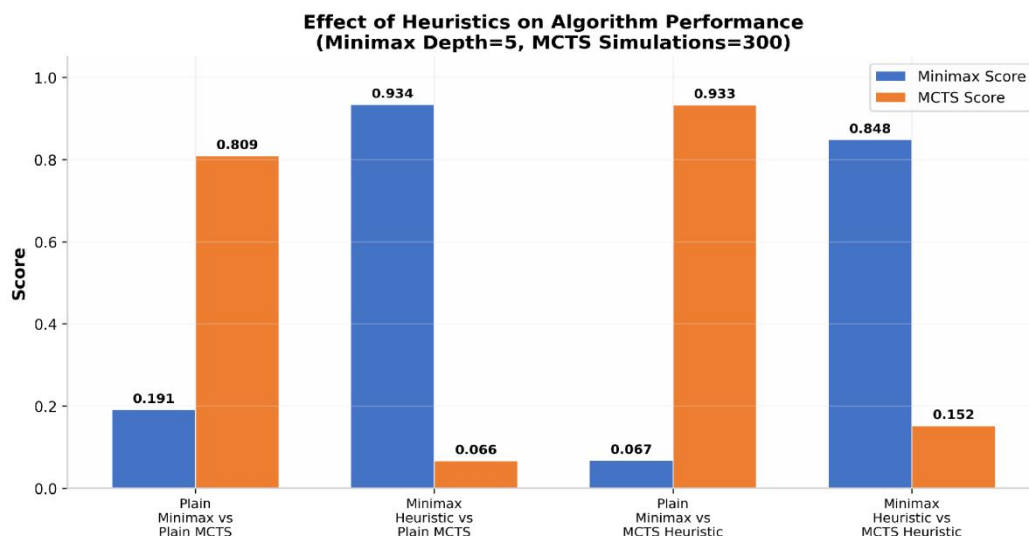


Figure 2: Effect of heuristic enhancements on algorithm performance

Heuristics dramatically improve both algorithms. Minimax with heuristics achieves 4.9x improvement over plain Minimax vs. plain MCTS (0.934 vs. 0.191), and dominates when both use heuristics (0.848 vs. 0.152). This suggests deterministic threat detection provides stronger tactical advantages than statistical sampling in this domain.

3.3 Experiment 3: Stage-Wise Performance

The 42 possible moves were divided into Beginning (1-14), Middle (15-28), and Final (29-42) phases. Using heuristic-enhanced algorithms, 100 games x 3 trials were conducted, recording termination stage and per-stage algorithm performance. The MCTS and Minimax scores are computed as follows:

$$\text{Minimax Score} = \frac{\# \text{Minimax Wins} + 0.5 \times \# \text{Draws}}{\# \text{Total Plays at This Stage}}$$

$$\text{MCTS Score} = \frac{\# \text{MCTS Wins} + 0.5 \times \# \text{Draws}}{\# \text{Total Plays at This Stage}}$$

Table 3: Stage-wise game termination counts and algorithm performance scores (3 trials)

Game Stage	Avg. Games Ended	MCTS Avg. Score	Minimax Avg. Score
Beginning	9.7	0.000	1.000
Middle	60	0.128	0.872
Final	30.3	0.352	0.648

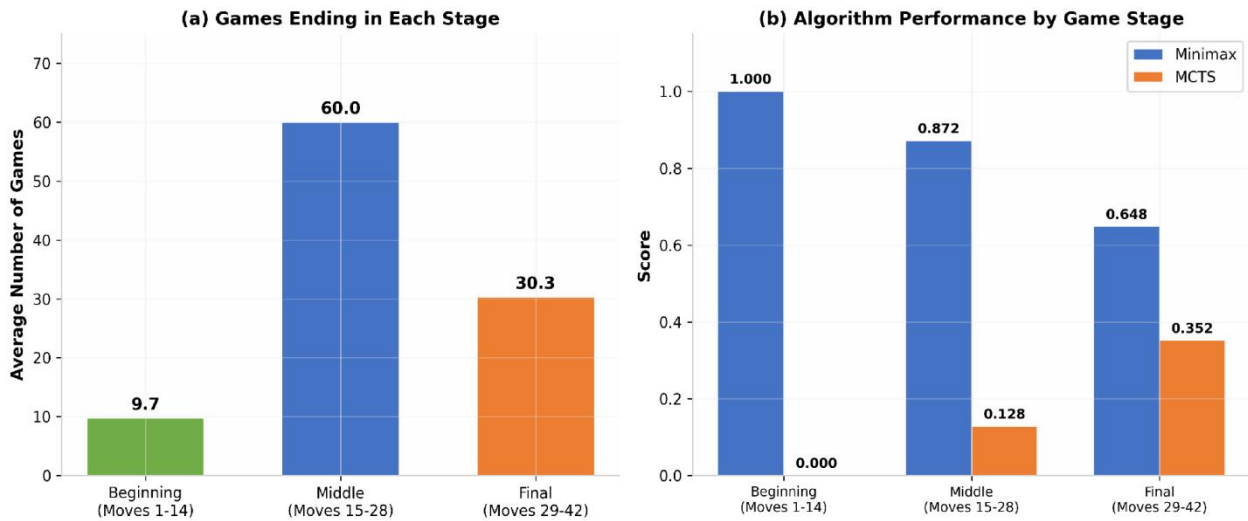


Figure 3: (a) Distribution of games ending in each stage; (b) Algorithm performance scores by game stage.

Most games (60.0 on average) conclude in the middle stage, with only 9.7 ending in the beginning. Minimax dominates all stages with perfect scores (1.000) in the beginning phase, while MCTS improves from 0.000 to 0.352 as board complexity increases.

4. Discussion

MCTS's baseline superiority indicates that randomized rollouts can effectively capture positional patterns under limited search budgets. However, the substantial gains from simple heuristics suggest that Connect Four contains tactical structures that may not be fully captured by pure statistical

sampling. This is consistent with Baier and Winands ^[6], who showed that MCTS can suffer from tactical weaknesses in domains requiring precise threat detection.

Stage-wise results show Minimax's early dominance stems from exact threat detection, while MCTS requires sufficient board complexity for meaningful simulations. This suggests hybrid approaches combining Minimax for tactical detection with MCTS for positional evaluation may be promising ^[6]. Limitations include simple heuristic implementations, untuned MCTS hyperparameters, and evaluation limited to the standard 7 x 6 board.

5. Conclusion

This paper presented an empirical comparison of Minimax with Alpha-Beta pruning and MCTS in Connect Four. Key findings: (1) MCTS consistently outperforms plain Minimax under limited budgets; (2) simple heuristics dramatically improve both algorithms, with Minimax showing particularly large gains; (3) performance varies across game stages, with Minimax dominating early phases. Future work could explore stage-aware hybrid algorithms, sophisticated heuristics, and evaluation on variant board sizes.

References

- [1] Victor Allis. A knowledge-based approach of connect-four: The game is solved: White wins. *ICGA Journal*, 1988, 11(4):165-165.
- [2] Donald E. Knuth, Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 1975, 6(4):293-326.
- [3] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012, 4(1):1-43.
- [4] Shlomi Asaf. ConnectFour[EB/OL]. GitHub, 2024. Available: <https://github.com/shlomias1/ConnectFour>. Accessed: 2026-04-20.
- [5] Levente Kocsis, Csaba Szepesvári. Bandit based Monte-Carlo planning. In: *Proceedings of the European Conference on Machine Learning*. Berlin Heidelberg: Springer, 2006:282-293.
- [6] Hendrik Baier, Mark H. M. Winands. Monte-Carlo tree search and minimax hybrids. In: *2013 IEEE Conference on Computational Intelligence in Games (CIG)*. IEEE, 2013:1-8.