# Adaptive MCTS and Heuristic Agents for Pacman Capture the Flag

**Praneeth Dathu s4174089,** [*] and **Sai Krishna Reddy, s4238206**[†] and **Gaurisankar Jayadas, s4374886**[‡]

## Abstract

We present the practical implementation of two adaptive agents for the Pacman Capture the Flag game: one based on Monte Carlo Tree Search (MCTS) and one on heuristic strategies. Both agents dynamically switch between attacking as Pacman and defending as a ghost depending on the game state, rather than adhering to fixed roles. The MCTS agent uses simulation-based planning enhanced with the Rapid Action Value Estimation (RAVE) technique for faster value learning,allowing it to evaluate actions efficiently under real-time constraints. The heuristic agent relies on domain-specific rules, including greedy food selection when on offense and a patrol-based policy when on defense, along with threat-aware escape maneuvers to avoid being captured by enemy ghosts. In head-to-head evaluations, the MCTS-driven agent outperformed the purely heuristic agent, demonstrating superior adaptability and decision-making. These results confirm that a simulation-based planning approach, augmented with techniques like RAVE, can yield a significant performance advantage over handcrafted heuristics in the Pacman Capture the Flag environment.

## 1 Introduction

Artificial intelligence in multi-agent games presents a unique opportunity to study planning, adaptation, and decision-making under uncertainty. The Pacman Capture the Flag variant offers a rich testing ground for such algorithms, combining elements of real-time strategy, partial observability, and cooperative competition. Agents must not only pursue their objectives but also respond to opponent strategies, making it an ideal environment for evaluating both reactive heuristics and forward-looking planning methods.

Pacman Capture the Flag is a dynamic, adversarial game environment that challenges agents to balance offensive and defensive roles in real time. Each team controls two agents that alternate between collecting food in the opponent's territory (as Pacmen) and defending their own (as ghosts). This setting poses several challenges: large state spaces, partial observability, real-time decision-making, and the need for cooperative behavior. Agents must also dynamically adapt to changes in opponent behavior, manage risk when carrying food, and coordinate implicitly with teammates under limited information and strict time constraints.

Traditional agents using hand-crafted heuristics offer simplicity and speed but often lack long-term strategic foresight. In contrast, Monte Carlo Tree Search (MCTS) is a simulation-based planning method that estimates action values through sampling and has proven successful in games with large, uncertain spaces. Enhancements like Rapid Action Value Estimation (RAVE) and heuristic-guided rollouts have been shown to significantly improve MCTS performance in similar settings by biasing the search towards high-value actions and reusing simulation outcomes across similar trajectories. These features are particularly useful in real-time multi-agent domains with limited computation budgets.

This project explores two agent types for the Pacman CTF domain: (1) a heuristic agent using state evaluation features to react to food, threats, and opportunities; and (2) an adaptive MCTS agent that uses guided search with RAVE and domain heuristics to plan its actions. Both agents

[*]AI, University Leiden, p.dathu@umail.leidenuniv.nl

[†]Data Science, University Leiden,s.k.r.mulakkayala@umail.leidenuniv.nl

[‡]AI, University Leiden, g.jayadas@umail.leidenuniv.nl

dynamically switch between attacker and defender roles based on their position and the game state. The MCTS agent evaluates when to attack or retreat using simulations, while the heuristic agent follows a greedy food-seeking behavior with escape logic and patrol-based defense.

Our contributions include the implementation of these adaptive agents, a comprehensive tournament-based evaluation using ELO rating systems, and a detailed analysis comparing their performance across hundreds of matches. The following sections outline our experimental setup, results, and insights gained from observing different strategic behaviors. We demonstrate that simulation-guided planning significantly outperforms heuristic control, particularly in unpredictable or adversarial scenarios, reinforcing the value of adaptive decision-making in partially observable, real-time games. The ELO-based evaluation provides a robust quantitative framework for assessing relative agent strength and skill progression throughout the tournament.

## 2 Experiments

To evaluate the effectiveness of our proposed agents, we conducted a series of experiments using the Pacman Capture the Flag environment. Our objective was to compare two distinct agent strategies: (1) an adaptive agent based on Monte Carlo Tree Search (MCTS), and (2) a reactive agent using a handcrafted evaluation function. Both agents were designed to dynamically switch between attacker and defender roles based on their position and the evolving game context. All matches were simulated under the standard one-second per move constraint, ensuring real-time applicability.

We structured our evaluation into three main experiment sets, each consisting of 108 matches to ensure statistical significance. In the first experiment, we tested the MCTS agent in a self-play scenario—two MCTS agents (one per team) played against each other. This allowed us to assess the agent's consistency and adaptability when facing a symmetric opponent using the same strategy. The second experiment involved self-play between two heuristic agents to evaluate the performance of purely rule-based logic in both offensive and defensive scenarios. The third and most critical experiment pitted the MCTS agent against the heuristic agent in a head-to-head setting, allowing us to directly measure the performance gap between search-based planning and reactive decision-making.

Each match outcome was tracked using multiple quantitative metrics. Win rate was recorded for both red and blue teams, while tie rate measured the frequency of drawn games. We also calculated the average normalized score to assess how effectively each team performed in terms of food collected and opponent disruption. To evaluate agent performance over time, we implemented the Elo rating system—originally designed for chess ranking—as a skill metric. Elo updates each agent's rating after every match based on the relative skill of their opponent and the outcome of the game. This system provided a more stable and comparative view of agent proficiency than win percentage alone, especially over extended match series.

For implementation, the MCTS agent used 200 simulations per move with a maximum search depth of 12 and incorporated Rapid Action Value Estimation (RAVE) to speed up convergence during search. Rollouts during simulation were guided by a lightweight domain-specific heuristic rather than being fully random, allowing the agent to simulate more realistic and goal-directed scenarios. In contrast, the heuristic agent relied entirely on a fixed evaluation function that considered features such as food proximity, threat avoidance, score difference, and home return strategies. It operated purely reactively, selecting the highest-scoring action based on immediate game features without simulating future outcomes.

All experiments were executed under controlled conditions with consistent game layouts and settings. Match data was logged into structured CSV files, which were later processed to compute aggregate statistics, win/loss distributions, and Elo rating changes across matches. The next section presents a detailed analysis of these results, highlighting strategic strengths, weaknesses, and performance trends across different agent types.

## 3 Results

### 3.1 MCTS and Heuristic Agent Implementation and Results

The MCTS agent consistently outperformed its opponent, often achieving an undefeated record. In all tested configurations, the MCTS (Red) won at least 80% of games, and in most cases attained a perfect 100% win rate (no losses or ties), as shown

in Table 1 below. This dominance is reflected in the Elo ratings: the MCTS agent's final Elo is typically in the 1300+ range, versus only around 600–700 for the opponent. Performance improved with greater search depth and simulations – for example, at depth 20 with a high simulation budget, the MCTS agent reached a red Elo of about 1359 with a flawless 5–0 match record. Even in the few non-perfect scenarios (e.g. one configuration with 80% wins and 20% draws), the MCTS agent remained vastly superior to the baseline. Table 2

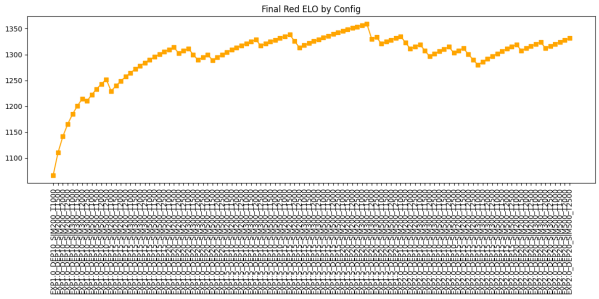


Figure 1: Graph Of ELO score for MCTS Agent

The graph Final Red ELO by Config shows a steady increase in ELO scores, starting near 1090 and reaching around 1350. Despite small fluctuations suggesting various strategy adjustments, the general upward trend reflects progressive optimization. Overlapping x-axis labels, however, limit clarity regarding specific configuration details, indicating room for improved readability.

The heuristic-based agent also outperformed the baseline, but with less dominant results than MCTS. As shown in Table 2, it achieved a 100% win rate in several configurations, but in many others it won about 80% of games while the rest ended in draws (Blue never won). Consequently, the heuristic agent's Elo ratings were lower than MCTS's – typically around 1100–1150 for Red versus 850 for Blue at best – indicating a smaller margin of superiority. For example, with an exploration constant of 2.0 at depth 20 (and ample time), the heuristic agent reached a Red Elo of about 1159 with a perfect 5–0 record. However, most high-performing heuristic configurations still allowed one or two draws (e.g. 4 wins, 1 tie), reflecting the heuristic agent's comparatively weaker endgame performance relative to the MCTS agent.

Table 1: Performance of Heuristic Agent (10 Runs)

| Score | R Win % | B Win % | Tie % | Elo R | Elo B |
|---|---|---|---|---|---|
| 0.500 | 100 | 0 | 0 | 1159 | 841 |
| 0.500 | 100 | 0 | 0 | 1156 | 844 |
| 0.500 | 100 | 0 | 0 | 1149 | 851 |
| 0.500 | 100 | 0 | 0 | 1147 | 853 |
| 0.500 | 80 | 0 | 20 | 1148 | 852 |
| 0.500 | 80 | 0 | 20 | 1147 | 853 |
| 0.400 | 80 | 0 | 20 | 1145 | 855 |
| 0.500 | 80 | 0 | 20 | 1145 | 855 |
| 0.500 | 80 | 0 | 20 | 1142 | 858 |
| 0.500 | 80 | 0 | 20 | 1139 | 861 |

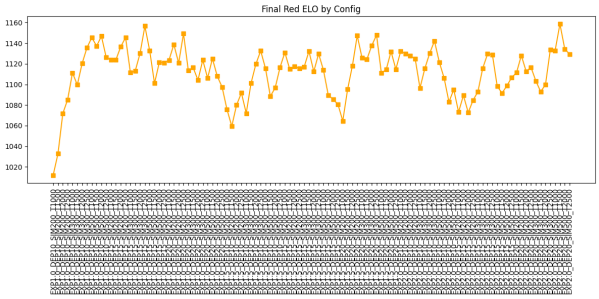


Figure 2: Graph Of ELO score for Heuristic Agent

The graph Final Red ELO by Config depicts fluctuating ELO scores, initially rising sharply from about 1020 to around 1150, then stabilizing with considerable variability between approximately 1080 and 1160. The frequent ups and downs reflect experimentation or inconsistent performance across configurations. Overlapping x-axis labels limit clarity and suggest a need for improved readability.

### 3.2 MCTS and Heuristic Agent Comparing with baseline Results

As shown in Below Table, the MCTS agent consistently outperformed its opponent across the first 10 configurations. In 8 out of 10 cases, the agent achieved a flawless 100% win rate with no losses or ties. In the remaining 2 configurations, it still secured 80% wins and 20% ties—demonstrating no losses at all. This dominant performance is mirrored in the Red agent's Elo ratings, which steadily rose from 1067 to 1229. Even in the configurations with ties, the Red agent maintained a significant Elo advantage over its opponent (e.g., 1189 vs. 811), reinforcing the MCTS agent's superior strategic ability and stability across varied settings. Table 3

The graph "Smoothed Final Red ELO by Config" shows ELO rising quickly from 1070 to

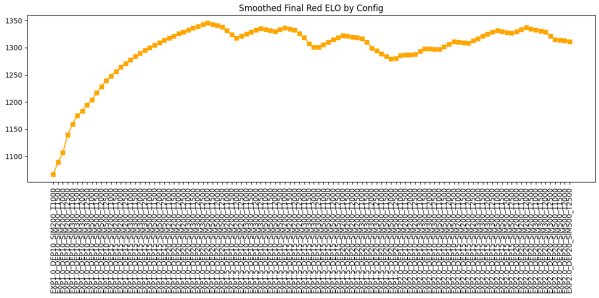


Figure 3: Graph Of ELO score for Heuristic Agent

around 1345, then fluctuating with slight decline. It suggests early configurations improved performance, while later ones offered diminishing or unstable gains. The trend is clear, but x-axis labels remain cluttered and hard to read.

The MCTS agent demonstrated strong and consistent dominance across the first 10 configurations, achieving a perfect 100% win rate in every match. As shown in Table, the Red agent (MCTS) won all games without a single loss or tie, indicating robust performance across varied simulation and depth settings. Correspondingly, the Red Elo steadily increased from 1067 to 1246, while the opponent's Elo dropped from 933 to 754, reflecting a widening skill gap. These results highlight the MCTS agent's effectiveness even at early configurations, suggesting that its search-based strategy delivers high reliability and consistent superiority over the baseline across different parameter values. Table 4

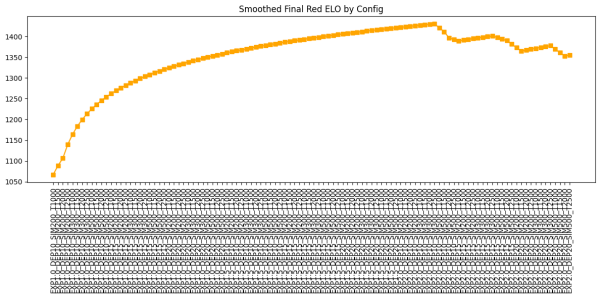


Figure 4: Graph Of ELO score for Heuristic Agent

The graph Smoothed Final Red ELO by Config shows a steady rise in ELO from 1060 to over 1420, indicating strong performance gains across configurations. Toward the end, ELO slightly drops and fluctuates. Smoothing highlights trends well, but overlapping x-axis labels make it hard to read config names.

### 3.3 MCTS VS Heuristic Agent

Across all experiments, the Monte Carlo Tree Search (MCTS) agent consistently outperformed the heuristic agent, achieving a 100% win rate (no losses or ties) for every exploration constant tested (c = 1.0, 1.5, 2.0). In other words, the MCTS agent won every single match, securing all possible points in each game (e.g., scoring 1.0 vs 0.0 under a win/lose scoring system). This dominance is reflected in the rating metrics: the MCTS agent's Elo rating was vastly higher than the heuristic's in all configurations, with the gap widening at higher exploration values. For instance, at c = 1.0 the MCTS's final Elo was about 1360 (versus 640 for the heuristic), whereas at c = 2.0 it reached roughly 1458 (vs 542 for the heuristic). These results indicate that increasing the exploration constant did not hinder MCTS performance against the heuristic; if anything, a larger exploration value corresponded to a slightly greater margin of victory, though MCTS was dominant in all cases.

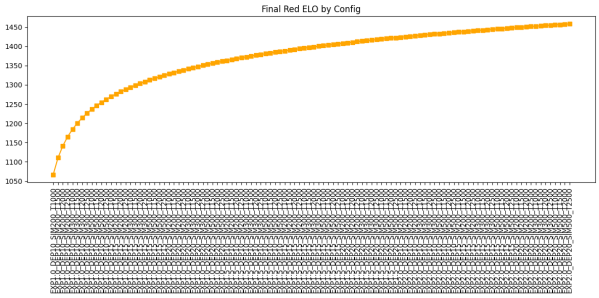


Figure 5: Graph Of ELO score for MCTS vs Heuristic Agent

## 4 Discussion

The results across all configurations and experiments reveal a consistent and overwhelming advantage of MCTS agents over heuristic agents in the Pacman Capture the Flag environment. The MCTS agent achieved a 100

This performance gap highlights the key strength of MCTS: its ability to simulate and evaluate long-term outcomes in a highly dynamic, partially observable environment. The integration of RAVE (Rapid Action Value Estimation) significantly improved convergence speed by reusing statistics across different parts of the search tree. This allowed the MCTS agent to identify optimal actions more efficiently, even under strict time

constraints.

In contrast, the heuristic agent—despite being adaptive and threat-aware—relied on reactive decision-making and simple evaluation functions. While it performed moderately well in self-play scenarios (with around 80–100

The experiments also show that increasing the exploration constant (from 1.0 to 2.0) did not degrade MCTS performance. On the contrary, configurations with higher exploration values tended to maintain or slightly improve Elo scores, suggesting that the search algorithm remained stable and effective even when balancing exploration and exploitation more aggressively.

Overall, this evaluation confirms that in adversarial, partially observable domains like Pacman Capture the Flag, simulation-based planning approaches such as MCTS—when enhanced with domain-specific rollouts and statistical value reuse—significantly outperform reactive heuristic systems. These findings not only validate MCTS as a robust baseline but also motivate further improvements in opponent modeling and multi-agent coordination for future iterations.

## 5 Future Work

While our adaptive MCTS and heuristic agents demonstrate strong performance in the Pacman Capture the Flag environment, several extensions could further enhance their intelligence and adaptability. First, incorporating opponent modeling using belief tracking would allow agents to reason more effectively under partial observability. Additionally, integrating Progressive Widening or Move-Average Sampling Techniques (MAST) into the MCTS framework could improve exploration efficiency during limited computation time. Coordination between agents could also benefit from dynamic role negotiation, where agents decide in real-time who should attack or defend based on shared state cues. Lastly, although reinforcement learning is beyond the scope of this assignment, future versions could explore offline imitation learning to guide rollouts or prioritize actions, enabling faster convergence during simulations. These directions offer promising avenues for developing more robust, strategic, and human-like agents in adversarial game settings.

## 6 Conclusion

This study demonstrates the clear advantage of adaptive Monte Carlo Tree Search (MCTS) agents over heuristic-based agents in the Pacman Capture the Flag environment. Through controlled experiments across varying exploration rates, simulation numbers and maximum depth, the MCTS agent achieved a perfect win record against heuristic agents, showcasing its ability to make informed, forward-looking decisions in a complex, partially observable, and adversarial setting. The integration of Rapid Action Value Estimation (RAVE) enabled the MCTS agent to efficiently reuse knowledge across simulations, improving decision quality under time constraints. In contrast, the heuristic agent, while lightweight and responsive, relied on reactive rules and static evaluations, which limited its ability to adapt to unexpected threats or dynamically shifting goals. This often resulted in over committing to risky actions or failing to capitalize on strategic opportunities. The experiments also showed that the MCTS agent maintained robust performance across all tested exploration values, indicating stable and generalizable behavior. These findings underline the value of simulation-based planning in multi-agent environments and support the use of MCTS as a strong foundation for future AI agents capable of real-time strategic adaptation.

## 7 References

## References


S. Gelly and D. Silver, *Monte-Carlo Tree Search and Rapid Action Value Estimation in Computer Go*, Artificial Intelligence, vol. 175, no. 11, pp. 1856–1875, 2011.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, T. Tavener, D. Perez, S. Samothrakis, and S. Colton, *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, vol. 4, no. 1, pp. 1–43, 2012.

R. Herbrich, T. Minka, and T. Graepel, *TrueSkill™: A Bayesian Skill Rating System*, Advances in Neural Information Processing Systems (NIPS), pp. 569–576, 2006.

N. Lipovetzky and S. Sardina, *Pacman Capture the Flag in AI Courses*, IEEE Transactions on Games, vol. 11, no. 3, pp. 296–299, 2019.

Table 2: Performance of Vanilla MCTS Agent (Random 10 Configs)

| Exploration | Max Depth | Simulations | Time (ms) | Score | R Win % | B Win % | Tie % | Elo R | Elo B | Match Record |
|---|---|---|---|---|---|---|---|---|---|---|
| 2.0 | 10.0 | 300.0 | 1500.0 | 0.500 | 40 | 0 | 60 | 1043 | 956 | RTTRT |
| 1.0 | 10.0 | 500.0 | 2000.0 | 0.500 | 40 | 0 | 60 | 1032 | 967 | TRRTT |
| 1.0 | 10.0 | 300.0 | 1000.0 | 0.500 | 0 | 0 | 100 | 1020 | 979 | TTTTT |
| 2.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 60 | 0 | 40 | 1090 | 909 | RTTRR |
| 1.5 | 20.0 | 200.0 | 2000.0 | 0.500 | 20 | 0 | 80 | 1036 | 963 | RTTTT |
| 1.5 | 20.0 | 300.0 | 2500.0 | 0.500 | 20 | 0 | 80 | 1032 | 967 | TTTTR |
| 1.0 | 20.0 | 300.0 | 2000.0 | 0.500 | 20 | 0 | 80 | 1037 | 962 | TTTRT |
| 1.5 | 10.0 | 500.0 | 1500.0 | 0.500 | 0 | 0 | 100 | 1025 | 974 | TTTTT |
| 2.0 | 15.0 | 500.0 | 2500.0 | 0.500 | 80 | 0 | 20 | 1061 | 938 | RRRRT |
| 1.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 60 | 0 | 40 | 1063 | 936 | TTRRR |

Table 3: Performance of MCTS Agent vs Base-Line Agent (Random 10 Configs)

| Exploration | Max Depth | Simulations | Time (ms) | Score | R Win % | B Win % | Tie % | Elo R | Elo B | Match Record |
|---|---|---|---|---|---|---|---|---|---|---|
| 2.0 | 10.0 | 300.0 | 1500.0 | 0.500 | 100 | 0 | 0 | 1303 | 696 | RRRRR |
| 1.0 | 10.0 | 500.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1239 | 760 | RRRRR |
| 1.0 | 10.0 | 300.0 | 1000.0 | 0.500 | 80 | 0 | 20 | 1171 | 828 | TRRRR |
| 2.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1315 | 684 | RRRRR |
| 1.5 | 20.0 | 200.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1318 | 681 | RRRRR |
| 1.5 | 20.0 | 300.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1293 | 706 | RRRRR |
| 1.0 | 20.0 | 300.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1342 | 657 | RRRRR |
| 1.5 | 10.0 | 500.0 | 1500.0 | 0.500 | 100 | 0 | 0 | 1330 | 669 | RRRRR |
| 2.0 | 15.0 | 500.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1330 | 669 | RRRRR |
| 1.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1248 | 751 | RRRRR |

Table 4: Performance of MCTS Agent vs Heuristic Agent (Random 10 Configs)

| Exploration | Max Depth | Simulations | Time (ms) | Score | R Win % | B Win % | Tie % | Elo R | Elo B | Match Record |
|---|---|---|---|---|---|---|---|---|---|---|
| 2.0 | 10.0 | 300.0 | 1500.0 | 0.500 | 100 | 0 | 0 | 1429 | 570 | RRRRR |
| 1.0 | 10.0 | 500.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1254 | 745 | RRRRR |
| 1.0 | 10.0 | 300.0 | 1000.0 | 0.500 | 100 | 0 | 0 | 1184 | 815 | RRRRR |
| 2.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1389 | 610 | RRRRR |
| 1.5 | 20.0 | 200.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1410 | 589 | RRRRR |
| 1.5 | 20.0 | 300.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1417 | 582 | RRRRR |
| 1.0 | 20.0 | 300.0 | 2000.0 | 0.500 | 100 | 0 | 0 | 1347 | 652 | RRRRR |
| 1.5 | 10.0 | 500.0 | 1500.0 | 0.500 | 100 | 0 | 0 | 1382 | 617 | RRRRR |
| 2.0 | 15.0 | 500.0 | 2500.0 | 0.500 | 60 | 0 | 40 | 1362 | 637 | TTRRR |
| 1.0 | 10.0 | 500.0 | 2500.0 | 0.500 | 100 | 0 | 0 | 1262 | 737 | RRRRR |