Optimal Move Selection Strategies for MCTS: Maximizing Performance using Rewards and Visits Analysis

Final Project - Decision Making by Methods of Search and Deep Learning 2023, Dr. Shahaf S. Shperberg

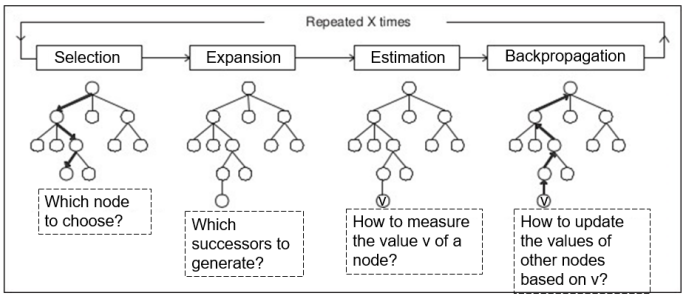
Submitted by Daniel Duenias – 307983130

1. Abstract

In the field of artificial intelligence and decision-making, Monte Carlo Tree Search (MCTS) has emerged as a powerful algorithm for solving complex problems. MCTS explores the search space by sampling potential moves and constructing a search tree. After an MCTS simulation, the rewards and number of visits for each neighbor provide valuable information about the quality and popularity of different moves. Here we focus on investigating optimal move selection strategies based on different strategies of utilizations of these statistics. We conducted empirical evaluations on various domains measuring the effectiveness of each proposed strategy. This work highlights the importance of effectively utilizing rewards and visit statistics during the move selection phase of MCTS. It provides a thorough analysis of diverse optimal move selection strategies, offering valuable insights into their significance and implications.

2. Introduction

In game-trees search, the base diagram for many algorithms is the following one:



After repeating the process (x number of times or limiting the time budget), the algorithm chooses the best move according to the observations. MCTS algorithm is built according to those properties:

1. Selection: UCT
2. Expansion: generate all successors
3. Back-propagation: weighted average
4. Final move: the child of the root with the highest value (or the most visited one)

Choosing the highest value follows an exploitation paradigm. By selecting the move with the highest value, the algorithm prioritizes moves that are expected to have better outcomes. This can lead to faster convergence towards optimal or near-optimal solutions, as the focus is on exploiting the most promising actions.

On the other hand, choosing the most visited move emphasizes exploration. It favors moves that have been extensively sampled during the MCTS simulations, allowing for better exploration of the search space. By selecting the most visited move, the algorithm can rely on more accurate and robust estimations of the move's quality. This approach is useful when the algorithm wants to continue exploring unexplored or underexplored moves to discover potentially better options.

It's important to note that the choice between selecting the highest value or the most visited move depends on the specific problem, the exploration-exploitation trade-off desired, and the characteristics of the search space. In this work, we aim to see how different strategies of reward-visits utilization can benefit different domains.

3. Method

In selecting our proposed methods, we were guided by two main principles. Firstly, we wanted to leverage more statistical information in the decision-making stage. By analyzing and incorporating relevant statistics, we aimed to achieve better results. Secondly, we recognized the importance of striking a balance between exploration and exploitation when choosing the next move. In the subsequent sections, we introduce and discuss our proposed methods in detail, highlighting how they address these guiding principles.

Our tested and proposed utilization methods:

1. Max reward – select the node with the highest average reward.
$$selected\ move = \underset{child \in root.children}{argmax} \ average(child.rewards)$$
The benefits of this method are mentioned at the end of the introduction session.

2. Max visits – select the most visited node.
$$selected\ move = \underset{child \in root.children}{argmax} \ child.visits$$
The benefits of this method are mentioned at the end of the introduction session.

3. Max mix reward-visits – select the node with the highest value of a linear combination between the average reward and the number of visits. To avoid biases due to the different scales of those parameters, they are normalized across all the root's children before the linear combination. The normalization compresses the values of the visits to be between 0 and 1 and does the same to the rewards. We note $min_{rewards}, min_{visits}, max_{rewards}, max_{visits}$, the minimum or maximum values of all the children visits and average rewards. Therefore, the linear combination is as follows,
$$linear\ comb(child) = \alpha \cdot \frac{mean(child.rewards) - min_{rewards}}{max_{rewards} - min_{rewards}} + \beta \cdot \frac{child.visits - min_{visits}}{max_{visits} - min_{visits}}$$
Because of the normalization, for convenience, to address the mixture as percentages of each component, we force $\alpha + \beta = 1, \ \alpha, \beta \geq 0$. Thus, the node for the next move is selected by the formula,
$$selected\ move = \underset{child \in root.children}{argmax} \ linear\ comb(child)$$
In order to get manage to grasp the influence but still not swipe over to many combination options, we used three different combinations of (α, β) - $(0.25, 0.75)$, $(0.5, 0.5)$ and $(0.75, 0.25)$.
This method tries to find the right balance between taking the move with the highest reward and the most explored move (exploration-exploitation balance-based decision).

4. Max reward median – select the node with the highest value of its median rewards.

$$selected\ move = \underset{child \in root.children}{argmax} \ meidan(child.rewards)$$

This method is an exploitation-based method. The use of median and not mean should be more robust to noised rewards (very high or very low rewards outliers).

5. Max reward optimistic – select the node with the highest average reward plus standard deviation.

$$selected\ move = \underset{child \in root.children}{argmax} \ \mu(child.rewards) + \sigma(child.rewards)$$

While μ, σ are the mean and std operations respectively. It is the optimistic because choice because over children with the same average reward it prefers the one with highest standard deviation - optimistic about the distribution of the rewards given a certain move. It may also prefer a lower average reward with high standard deviation (option to be very high or very low) over higher average reward move. The average reward plus the standard deviation can be interpreted as "the best possible outcome". This method leverages more statistical information about the rewards distribution given a specific move.

6. Max reward pessimistic – select the node with the highest average reward minus standard deviation.

$$selected\ move = \underset{child \in root.children}{argmax} \ \mu(child.rewards) - \sigma(child.rewards)$$

While μ, σ are the mean and std operations respectively. It called pessimistic because it is symmetric to the optimistic method. Here the computation can be interpreted as finding all the "worst case reward" for the children and chooses the best option between those. Here as well, more statistical information about the distribution of the reward is very beneficial. Std helps us assess the uncertainty of our estimations.

To test the proposed method, we adapted the MCTS algorithm for our need:

1. Selection: UCT / epsilon greedy
2. Expansion: generate all successors
3. Back-propagation: add to rewards list
4. Final move: proposed methods X

To update the estimate the value of a node we are using rollout with random probabilities and depth budget (different in each experiment).

We know that UCT have some disadvantages because it aims to minimize the cumulative regret. In the games case, the cumulative regret in the selection part has no relevance because the final action is the only thing that counts. As we saw in class, we can use epsilon greedy algorithms for the bounding the simple regret (the regret of the last move alone). The epsilon greedy selection algorithm selects the node with the highest average reward with probability $\epsilon = 0.5$, and randomly chooses a different node otherwise.

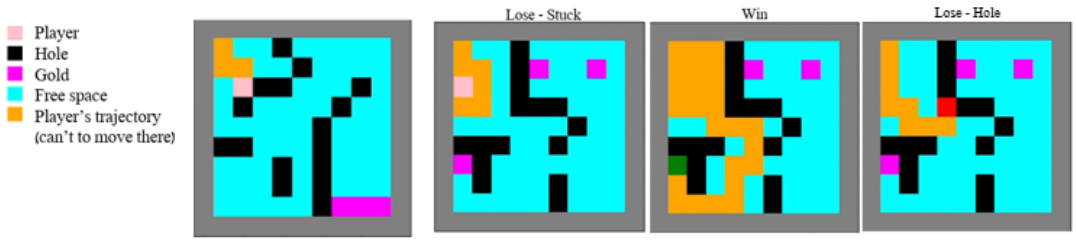
We chose to try another selection algorithm to compare our proposed methods with one more known selection algorithm. Our goal here is to compare between our proposed methods so we didn't try more advance selection methods.

4. Experiments

In this section, we present three experiments we conducted to evaluate the effectiveness of our proposed methods. Each experiment focused on testing the methods across different domains, represented by three distinct games. Each game possesses unique characteristics and success metrics. Employing the MCTS algorithm, we use a time budget for each move. Once the time limit is reached, a move is selected based on each of our proposed "best move selection" methods. We provide detailed descriptions of each game, along with the results obtained and a comprehensive discussion on the performance of our proposed methods within each game's context.

4.1 Walking-Game

Description: The game takes place in a 2d grid world. In the game, there are gold, holes, and the player. The goal is for the player to find the gold without falling to a hole. The player starts from the top left of the board and chooses each step. After being in a certain location, the player can't visit it again. If the player gets to the gold he wins, if he falls to a hole or get stuck, he loses.

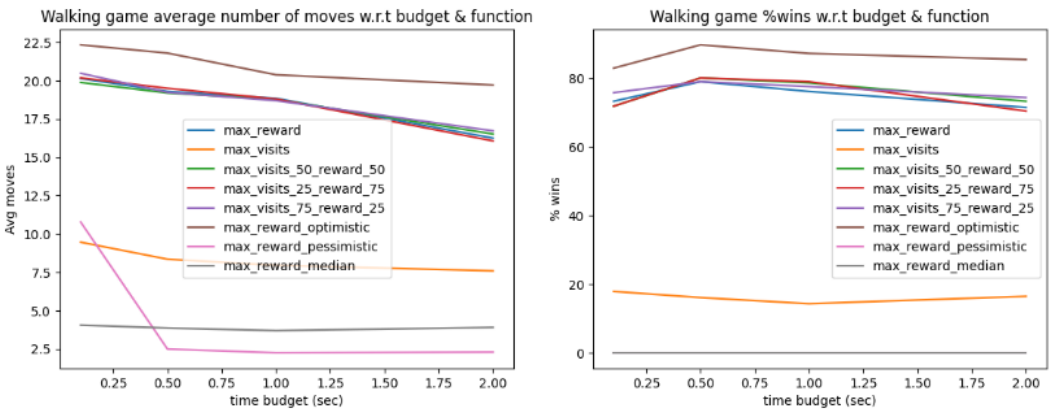


We created the game, as well as 28 levels of it with different difficulties (see appendix). The reward is 1 if the player wins and -1 for losing. To help the player with narrow spaces we added a small living reward, that is what helped it not choosing to terminate the game by falling to a hole in those situations.

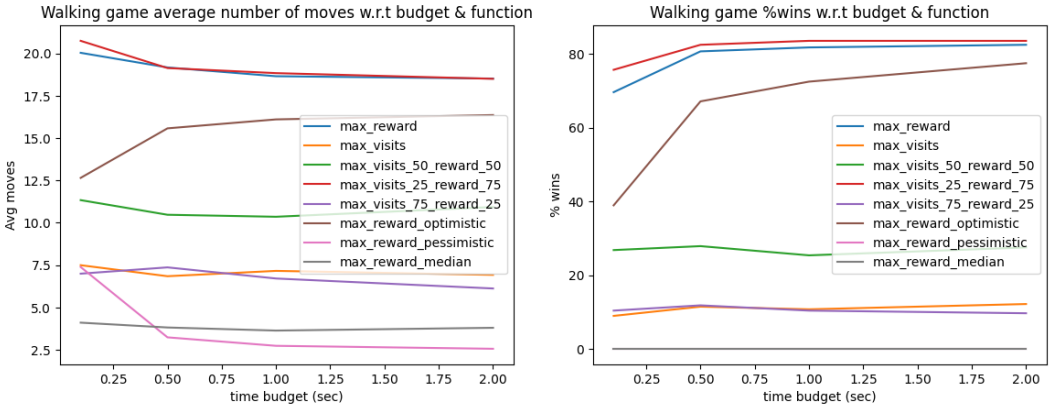
Results:

According to the rules, we created graphs that will help us evaluate each of our proposed methods. We played using each method and swiped over several time budgets ("thinking time" per each move), 10 games per level in each combination for reliable results. The winning percentages is the most relevant one while the average number of moves can tell us about the efficiency of each method (assuming that there is high win percentage). Low win percentage combined with low number of moves points on poor moves selection that ends the game fast without winning.

Selection with UCT



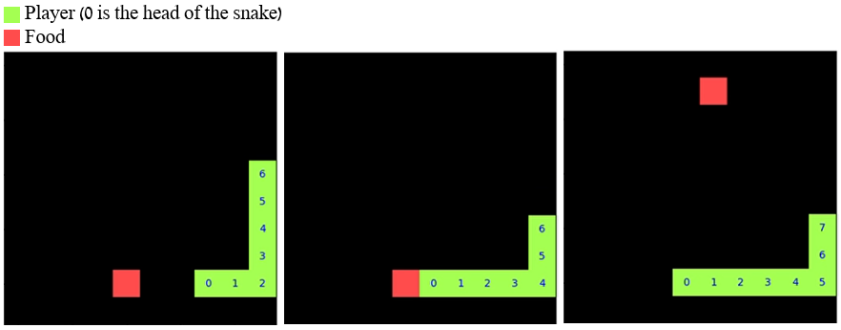
Selection with epsilon greedy



In most cases, as the time budget increases, the number of moves drops (more efficient). Looking at the "epsilon greedy selection" graphs, there are 3 main leading methods while the rest are far below – "max_reward", "max_reward_optimistic" and linear combination of visits-reward with ratio of 25%-75%. Looking at the UCT graph in the other hand, most of the methods that combines reward and visits are the same while "max_reward_optimistic" leads and the visits only, reward_pessimistic and reward_median are far below. Comparing the UCT and the "epsilon greedy" selection, we can see that in UCT most of the methods have higher winning percentage.

4.2 Snake

Description: The game takes place in a 2d grid world. In the game, there are the player (a snake) and randomly generated food. The player controls the head of the snake, and its goal is to eat as much food as he can. While eating, the snake grows and each time it eats, a new food is randomly spawn in the world. If the snake touches itself the player loses. In this version, it can go through the wall and appear on the opposite one.



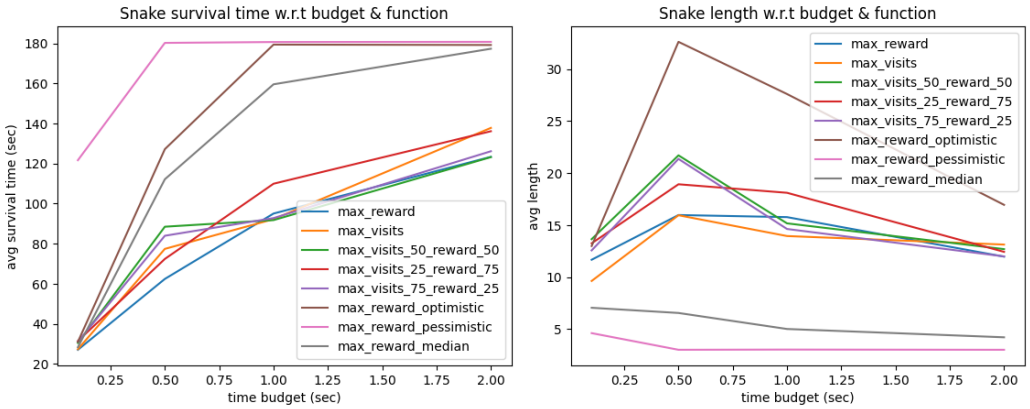
Here 0 marks the head and the snake start with length of 3. The reward is the length of the snake, there is immediate reward for eating in the current move to encourage selecting the move that eats (10 pts). There is a living cost of -0.1 points and dying cost of -5. The only terminal state is when the snake is dead. Because it may never happen, so we added a total game time limit of 3 minutes (180 seconds).

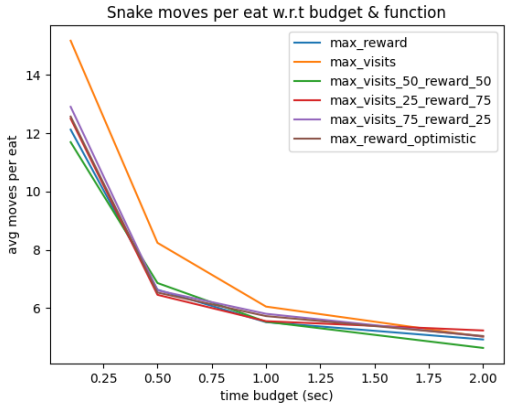
Results:

According to the rules, we created graphs that will help us evaluate each of our proposed methods. We played using each method and swiped over several time budgets ("thinking time" per each move), 50 games each combination for reliable results. Survival time is how long the snake lived (maximum of 3 minutes). Moves per eat is how many moves the snake did per one eat. In the "moves per eat graph" we

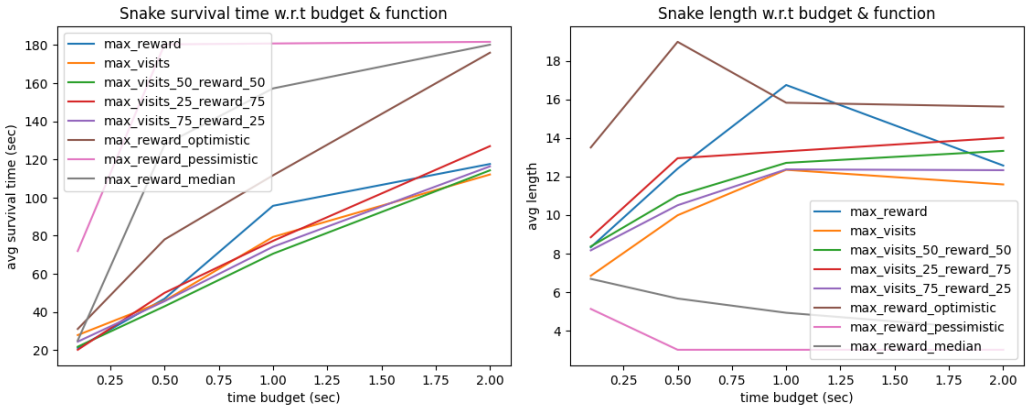
discarded the "max_reward_pessimistic" and the "max_reward_median" methods because their results where to far from the rest, so it was hard to observe the rest.

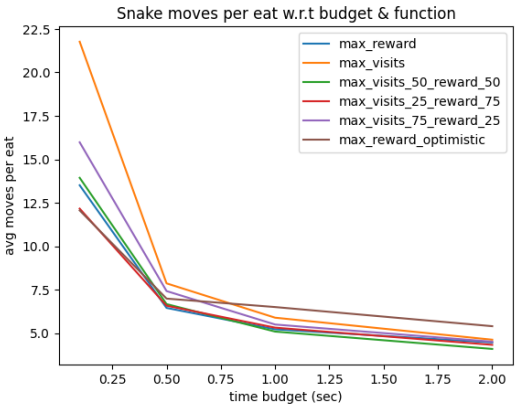
Selection with UCT





Selection with epsilon greedy





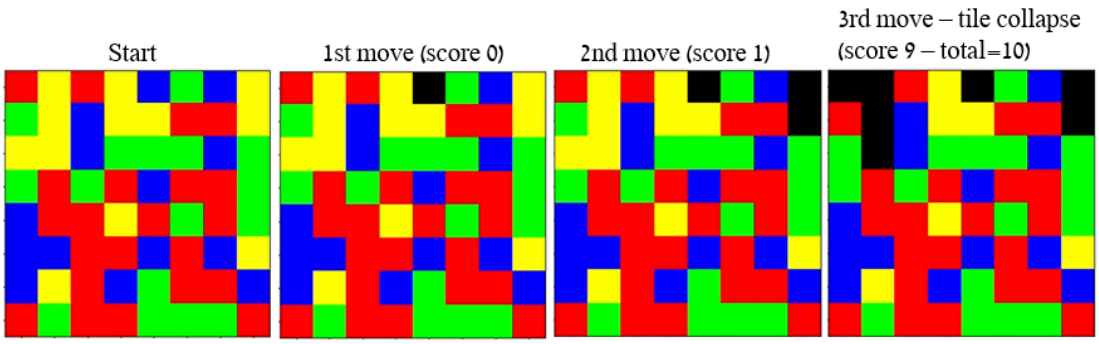
We can see that although the good survival time of "max_reward_pessimistic" and the "max_reward_median" methods, the length of the snake stays rather small. They both miss the main goal. As for the rest, as expected, more time budget means better efficiency of eating (lower moves per eat ratio). The moves per eat ratio with a long budget is around the same for all, however, the "max_visits" method seems to get slightly wors results than the rest. Looking both at the survival time and the length, one can observe that the better methods are the ones that prefer the reward (only reward, reward optimistic and combination of 25%visits + 75% reward). With the most successful one over most of the time budgets is the "max_reard_optimistic" method. This method provides both the maximum length and surviving time which are somehow opposite to each other (larger snake is harder to keep alive). A small comparison between the UCT and the epsilon greedy results shows us that in this case, the UCT achieves better results with longer snakes that survives longer (relevant for most move selection methods).

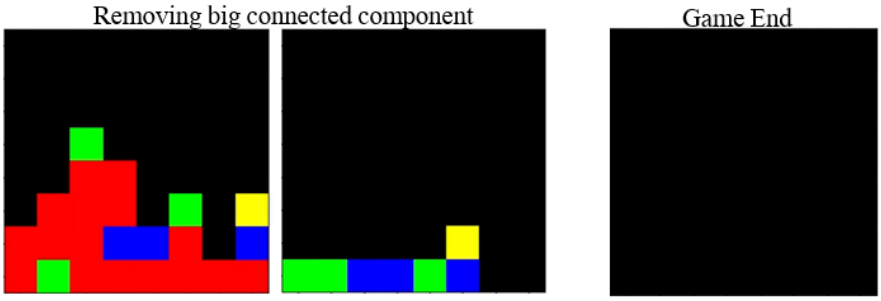
4.3 Same-Game

Description: A version of the game known also as Clickomania. This is a captivating tile-matching puzzle game where the goal is to clear the board by removing groups of adjacent tiles of the same color. Players strategically select and remove these groups, causing tiles above to collapse. The challenge is to maximize the scores while clearing the entire board (game ends). The scores of each move are relative to the size of the connected component of the same color that is removed and is given by the formula,

$$scores = (n-1)^2$$

While n is the number of tiles that where in the removed component. One can see a clear advantage in removing larger components. The game's score is the sum of the moves scores of the whole game.



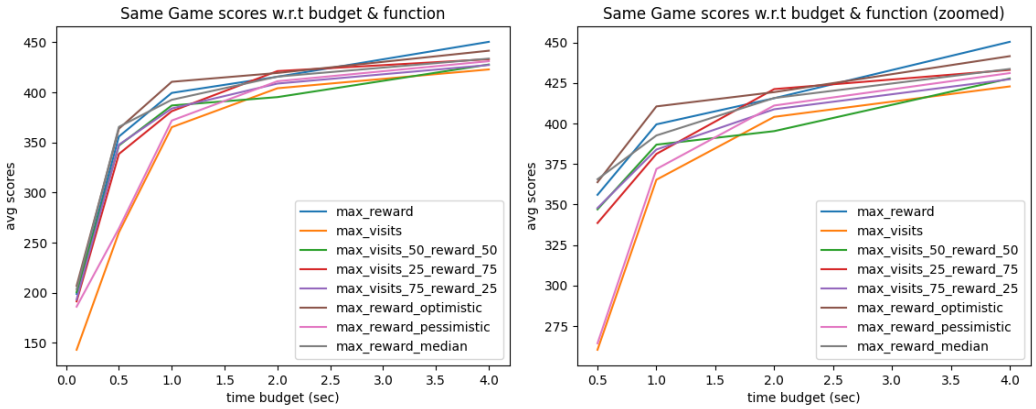


Results:

According to the rules, we created graphs that will help us evaluate each of our proposed methods. We played using each method and swiped over several time budgets ("thinking time" per each move), 100 games each combination for reliable results. The two graphs are the same but the left one is "zoomed in" for better separation between the methods.

Selection with UCT

Selection with epsilon greedy



As expected, the higher the budget per move, the higher the scores. In general, using UCT results in slightly higher scores for most methods. That said, the best performing methods are similar in both the UCT and the epsilon greedy selection. From the graphs, the best performing methods are the ones that favors rewards in general. The top 2 from the epsilon greedy selection are the "max_reward" and the "max_reward_optimistic". From the UCT selection all the following methods have roughly similar performance: linear combinations between rewards and visits, "max_reward", "max_reward_optimistic".

4. Results and Discussion

The results of the experiments clearly demonstrate that relying solely on the number of visits is not sufficient for achieving optimal performance. The leading methods in our study are those that gives a substantial weight to the reward information. Among the tested methods, "max_reward_optimistic" stands out as the most effective in all of the domains. This method effectively leveraging the standard deviation to guide the search towards potentially more rewarding paths. As mentioned before, the average reward plus the standard deviation can be interpreted as choosing the best possible belief. Leveraging more statistical information about the rewards distribution given a specific move is indeed beneficial.

The least effective methods were "max_reward_median", "max_reward_pessimistic" and "max_visits". In a reward-based game, it does makes sense that one will need to consider the rewards and can't rely on the number of visits alone because the visits is based solely on the selection method, UCT or epsilon greedy. As for the median method, this method may be more reliable if the rewards estimation is better than random rollouts. We can also see that when the branching factor was bigger and the range of the rewards was larger (Same Game), the median method did perform better than other games. Its strength comes when there are more values to compare to. The pessimistic approach may not have any statistical justification to be more successful apart from the intuition – choosing the "least worse possible outcome" gives in far worse results than the "best possible outcome". In a more general note, it is probably better being optimistic.

The exploration of move selection strategies in our experiments prompts us to inquire: Is there a superior approach, or does the optimal method hinge solely on the specific domain and foundational assumptions? The implementation details and reward structures in the games undoubtedly influenced the obtained outcomes. Well-designed rewards that align with the game's objectives led to more favorable results, while inadequately defined rewards may have hindered performance, thus led to slightly different favorable methods, although they were also reward based. To gauge the adaptability of the proposed methods, we need to explore how robust they are to variations in a game's rewards and definitions. Conducting sensitivity analyses on the rewards and definition parameters across diverse game scenarios could shed light on the strategies' resilience and generalizability.

An unexpected finding we observed was that across all domains, MCTS with UCT outperformed the epsilon-greedy selection method, even though it was not part of our initial research focus.

Future work

Based on these findings, future work could involve exploring the combination of "max_reward_optimistic" with visit-based methods. By incorporating both reward's second order statistics and visit information, we can potentially create a more comprehensive and powerful decision-making strategy. Another possible approach could involve using a list of rewards obtained during simulations and giving more weight to rewards obtained later in the simulations. The assumption here is that as the simulations progress, using rollouts, the rewards tend to become more accurate and reliable indicators of the true quality of a move or action.

Appendix

Attached to this PDF there are folders containing the raw results of the experiments.

Walking game levels: