Design of a Special Gomoku Decision Model Based on Monte-Carlo Tree Search

MSDM 5002 Final Project Individual Report: Hu, Xinfeng

Abstract

Monte Carlo Tree Search (MCTS) is a powerful algorithm primarily used for decision-making processes, particularly in game playing. It combines the precision of tree search with the randomness of Monte Carlo methods, allowing it to explore large search spaces efficiently. This report will focus on its application with Gomoku and outline the key components and workings of our decision model based on this methodology.

Basic Rules

Traditional Gomoku is played on a 15x15 square board, however, this project introduces a circular board to alter the gameplay dynamics. The circular board is unwound into a 16x10 rectangular grid for ease of visualization and calculation, with the radial coordinate r ranging from 0 to 9 and the angular coordinate θ ranging from 0 to 15. Any coordinates with r=0 on the rectangular grid correspond to the center of the board, equivalent to the point (0,0) in polar coordinates. A player wins by aligning five of their stones in a line, which can be horizontal, vertical, or diagonal on the board (when checking the winning conditions, diagonal chains are not allowed to contain (0,0). Otherwise, the advantage is too big). The game continues until a player achieves the winning condition or until the board is full of no winning configuration, resulting in a draw. Thus, a draw is treated as a win for the White player, acknowledging the first-move pros of the Black player.

Monte-Carlo Tree Search

MCTS consists of four main steps that are repeated until a termination condition is met, such as a time limit or a maximum number of simulations. Key Components of MCTS are listed as below:

a. Tree Structure:

With MCTS, we always first build a search tree where each node represents a game state, and edges represent possible actions leading to subsequent states. The tree grows dynamically as the algorithm explores more states.

b. Selection:

Starting from the root, MCTS selects nodes down to a leaf node using a strategy that balances exploration (trying new moves) and exploitation (choosing nodes that have previously shown promise). The Upper Confidence Bound for Trees (UCT) is a common method used for this purpose.

c. Expansion:

Once a leaf node is reached, the algorithm expands the tree by adding one or more child nodes representing possible moves from that state.

d. Simulation:

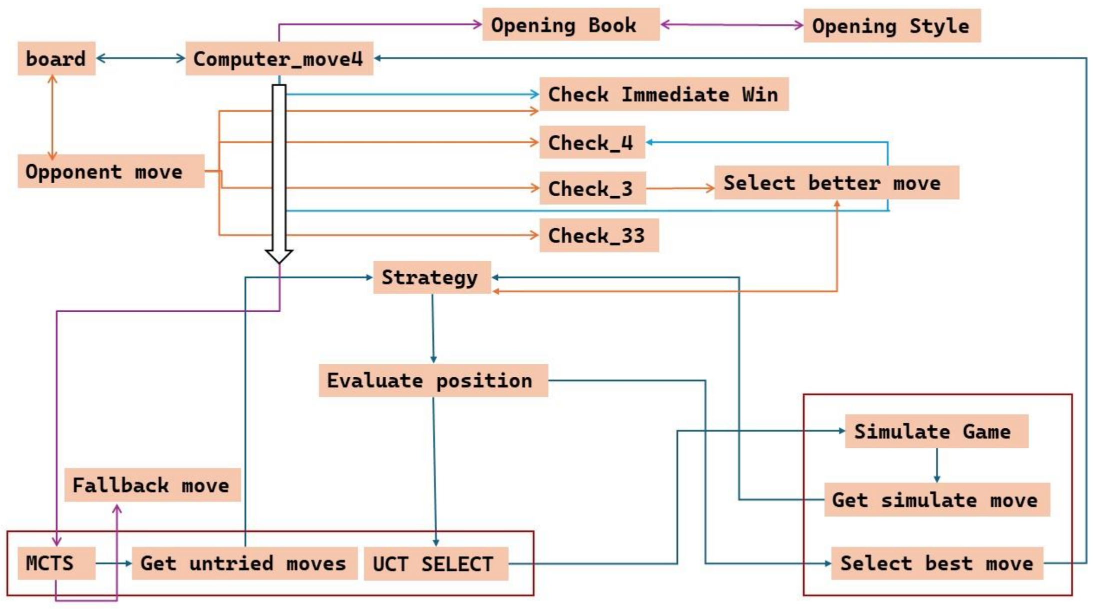
A simulation (or playout) is performed from the new node to a terminal state (win, lose, or draw). This is typically done using random moves, though more sophisticated strategies can be employed.

e. Backpropagation:

The results of the simulation are then propagated back up the tree, updating the statistics of each node along the path. This involves incrementing the visit count and updating the win/loss statistics based on the outcome.

MCTS can be applied to a wide range of problems, including imperfect information games and complex decision-making tasks. It effectively handles large state spaces, scaling well with the complexity of the game, meanwhile, it does not require an evaluation function, making it suitable for games where such evaluations are difficult to define. However, Computational cost could be quite computationally expensive, especially in games with deep search trees where many simulations are required. Besides, the effectiveness of MCTS heavily relies on the quality of simulations. Random simulations can sometimes lead to suboptimal decisions. Another crucial point that should be discussed is the trade-off between exploration and exploitation, and it always indicates that poor tuning can lead to inefficient searches.

Decision Model Design



As you can see, this model design diagram illustrates our group’s final decision-making approach. computer_move4 represents the final strategy we will use in the match, while opponent move indicates information related to the opponent. The white arrows show the decision sequence of our strategy in situations other than the opening move: first, we consider whether we can win in this move (i.e., there are four connected live stones). If not, we then consider whether the opponent has four connected stones. Subsequently, we consider the opponent’s potential four-in-a-row, potential three-in-a-row, our potential four-in-a-row, and the opponent’s potential live three-three strategy. Regarding the opponent’s potential ‘live three’ positions, there may be many locations to consider. We have implemented a selection function based on the core strategy to execute the response steps. The core module of this strategy in our plan includes numerous parameters and corresponding position evaluators. These are used to comprehensively assess whether a specific position has sufficient relative advantages in terms of pattern formation, territory control, connectivity, and mobility. When these predefined strategies do not need to be executed, we will use the MCTS model to find the most advantageous move. Specifically, our model will first obtain the nodes to be considered based on the strategy module, rather than acquiring all positions from the entire board (which would undoubtedly increase the computational burden). The UCT method will be used to filter nodes into the simulation phase. Through simulations of games within 30 moves and an additional step of predicting the opponent’s move based on our strategy core, we can obtain the optimal move calculated by the MCTS strategy. To ensure the robustness of the program, a heuristic scheme based solely on the position evaluation of the strategy core (Fallback move) will be used in special cases where the

MCTS returns no value.

My Contribution

In this project, my main responsibilities and contributions are as follows. Firstly, I iterated and optimized the MCTS module and gradually debugged it in computer_move2, computer_move3 and computer_move4. For example, identifying that MCTS was not executed due to incorrect parameter passing in the 'simulate_game' function; Improving 'simulate_game' to follow the adversarial approach by predicting an additional move based on our decision core, etc.); Writing the final strategy for the opponent's live four, live three, and our winning live four strategies; Solving the issue of frequent false positives and false negatives in the three-three and live three strategies, which often led to unexpected losses; Due to the complexity of our group's plan, adjusting the execution and invocation order of all our strategies to balance offense and defense (as shown in the decision-making diagram). Additionally, I was responsible for the adversarial testing of the five different generation plans. The current tests have shown satisfactory results, and I am very excited about its potential to win the championship!

Peer Evaluation

Team Member	Score (0 ~5)
HU, Xinfeng	5
HUANG, Jiayu	5
YIN, Yin	5
CHEN, Siyan	5

I sincerely appreciate my teammates for their efforts. We all fulfilled our responsibilities in this project, put in tremendous effort, and managed to create such a complex and intelligent Gomoku player. I am proud of our work. In our team, Huang excelled at integrating functional functions, strategy programs, and testing modules. In our team, her engineer coding skills are excellent, and she is always highly focused, responsible, and logically clear. Every aspect of our project relied on her contributions. YIN developed our team's first "normal" machine strategy, which was akin to the first step of the moon landing, bringing new hope to our entire work. Additionally, her framework for the entire plan and hyperparameter setting techniques were crucial to our success. CHEN is the genius player in our team. She discovered many fundamental patterns and practical techniques under this special rule, playing a leading role in the design of predefined strategies and iteration tests.