Transactions on Computer Science and Intelligent Systems Research
ISSN: 2960-1800, eISSN: 2960-2238 | Volume 9
AIDML 2025

Design and Optimization of a Chinese Chess engine Based on MCTS

Jiashu Chang *

United International College, Beijing Normal-Hong Kong Baptist University, Zhuhai, China

* Corresponding Authors: 230034004@mail.uic.edu.cn

Abstract. Chinese chess, as part of Chinese traditional culture, presents unique challenges and value in the field of artificial intelligence research. This study focuses on innovatively applying the Monte Carlo Tree Search (MCTS) algorithm to the design of a Chinese chess engine, breaking through the limitations of traditional engines that rely on Alpha-Beta pruning and manual evaluation functions. By implementing a mixed strategy generation framework, the engine achieves decision-making and optimization in complex situations. Experiments show that the MCTS-based engine demonstrates significant tactical combination discovery capabilities during the middle game, with win rates for red and black pieces reaching 62.5% and 50%, respectively (against intermediate traditional engines), with critical decision response times controlled within 500ms. Its innovation lies in dynamically allocating computational resources, integrating opening library rules with MCTS random search, ensuring flexibility and the ability to handle complex rule requirements. The study validates the transfer potential of MCTS in non-complete information games, providing a reusable framework for AI development in traditional board games such as Chinese chess and Korean chess. Future work can further enhance the precision of handling complex situations by introducing reinforcement learning, promoting deep integration and innovation between artificial intelligence and traditional chess skills.

Keywords: Chinese chess; Monte Carlo tree search; engine design; optimization.

1. Introduction

Chinese chess, a brilliant gem of Chinese civilization, can be traced back to the Warring States period. Over more than two thousand years of evolution and inheritance, it has developed unique cultural connotations and an intellectual competition system [1]. From its early form in Northern Zhou as a game called "Xiangxi" to its modern form established during the Song Dynasty with "32 pieces arranged in 10 rows and 9 columns," Chinese chess has always embodied the wisdom of traditional Chinese culture. It is not only a widely popular recreational activity in the streets and alleys but also became an official national sport in 1956. This led to the emergence of professional events such as the National Individual Championship and the First Division League, as well as international stages like the Asian Games and the World Mind Sports Games, showcasing its enduring charm across time and space.

In the context of the thriving development of artificial intelligence technology, the advancement of Chinese chess engines lags behind projects like Go. Early Chinese chess software primarily relied on the combination of extreme value algorithms and Alpha-Beta pruning, using manually tuned evaluation functions to analyze game positions. This method often falls short when dealing with complex middlegames and endgames due to limitations in the accuracy of evaluation functions and search depth, making it difficult to fully explore potential variations in the game [2]. In contrast, Go engines have leveraged breakthroughs in Monte Carlo Tree Search (MCTS) technology, which involves simulating numerous random moves and combining them with strategic evaluations, achieving significant advantages in handling complex situations. This technological disparity highlights the urgent need for Chinese chess engines to introduce new algorithmic frameworks to break through the bottlenecks of traditional approaches.

This research paper focuses on innovatively applying the Monte Carlo Tree Search algorithm to the field of Chinese chess, aiming to construct an optimized MCTS framework to enhance engine intelligence. Monte Carlo Tree Search, through iterative simulation and backtracking optimization, can efficiently explore legal moves in game scenarios with large branching factors, making it particularly suitable for chess, a game with high complexity and multiple decision paths [3]. Compared to traditional algorithms, MCTS does not rely on manually designed evaluation functions but instead optimizes strategies through self-play and data-driven methods, thus providing more flexible handling of complex games. This technological innovation not only offers a new technical path for chess engines but also provides important insights into the adaptability of artificial intelligence in traditional board games.

The optimization of the chess engine has multiple significances. Technically, it promotes the cross-project application of AI algorithms in game theory, validates the effectiveness of MCTS under different rule systems, and provides theoretical support for subsequent research [4]. From the perspective of cultural inheritance, empowering chess with AI technology helps enhance its modern appeal and promotes its spread and development worldwide [1]. Additionally, the optimized chess engine can serve as an intelligent auxiliary tool for professional players' training, aiding them in breaking through mental constraints through deep game analysis and strategy simulation, thereby boosting overall competitive levels. The research objective is to achieve an optimized Chinese chess engine model based on MCTS and improve upon the existing traditional engine.

2. Main Methods of Chess Engine

2.1. Chess Rules Representation

First of all, to realize a chess engine, we need to start with the rules of chess. The introduction of chess rules includes the representation of the board and pieces, the generation of the rules of piece movement and capture, and the judgment of the game's victory or defeat.

2.1.1. Chessboard and Pieces Representation.

The Chinese chessboard is composed of nine vertical lines and ten horizontal lines intersecting, which can be represented as a 10x9 two-dimensional array; each side has sixteen pieces, including the general (Shuai), advisor, elephant, horse, rook, cannon, and pawn (soldier), each with its specific movement rules [2]. It is worth noting that due to the different colors of the two sides, the same piece type needs to be coded twice, while different pieces of the same type use the same letter code corresponding to their initial positions on the board, with empty spaces filled with 0. Clarifying these rules forms the foundation for building a chess engine, where the program uses logical code to determine the legality of piece movements.

2.1.2. Legal Generation.

The generation of moves in a game refers to calculating or deducing the next feasible action based on specific rules and the current board position. The complexity of move generation varies across different board games due to their distinct rules [2]. Given the unique game mechanics of Chinese chess, move generation must consider both movement and capture. In terms of movement, each piece has its own code [5]. For example, the rook can move vertically or horizontally without restriction; the knight moves in an "L" shape and has the knight's check rule, among others. Regarding captures, most pieces can be captured according to their movement rules, except for the cannon, which requires a special definition due to its "over-the-hill" rule.

2.1.3. Winning and Losing Judgment.

The conclusion of a game of chess depends on the definition of winning or losing. According to the rules, one side wins by checkmating or trapping the opponent's king (general), while the other loses. In simpler terms, one side wins if they prevent all possible moves from the opponent to save their king (general); conversely, the other wins. If neither side can force the opponent to checkmate, the

game ends in a draw. During the process of playing, special rules such as long check or long capture must be set; otherwise, it would contradict the chess proverb "long check does not lead to a draw."

2.2. State Transition

The game state is defined by the positions of pieces on the board and which player is currently in turn. A state transition occurs when a piece moves, leading to a change in the game's state. When a player (or engine) makes a move, the program first checks whether the move is legal. If it is, the position information of the pieces on the board is updated, and the current player is switched, thus completing the state transition. In code implementation, this is reflected through modifications to the data structure.

2.3. Monte Carlo Tree Search Principle

The core part of the system, Monte Carlo Tree Search (MCTS) is a heuristic search algorithm used to find the optimal action in a game tree. Its key steps include: selection, expansion, simulation, and backpropagation [6]. As shown in Figure 1, selection starts from the root node and selects child nodes based on specific strategies (here referring to UCB values) until an unexpanded node is found. Expansion generates a new legal action corresponding to a child node, which means selecting an unprecedented point from a node. Simulation begins at the expanded node and involves a random simulation game until it ends. During the simulation, both players randomly choose legal actions, ultimately resulting in a game outcome. Finally, the simulation results are backpropagated along the search path, updating the statistical information of all nodes on the path, such as visit counts and win rates. Through multiple iterations of this process, the MCTS algorithm gradually focuses on more promising branches in the game tree, thereby finding the optimal solution.

$$\mathrm{UCB1}(S_i) = V_i + c\sqrt{\frac{log^N}{N_i}} \tag{1}$$

Before diving into a reference example, it is essential to have a framework for UCB values and node values. As shown in the formula, the UCB value is the sum of the node value and the exploration value. The node value can be approximated as the score given to a state by a general engine. Suppose the node values for root node S0 and leaf node S1 are 20, while S2 is an unexplored node with a n_x value approaching 0 and a UCB value tending towards infinity. In this case, the node selected for simulation would certainly be S2. After one exploration, if the number of nodes at S2 is 10, when backtracking to S0, the node with the higher value, S1, will be chosen. If S1 is a leaf node, it will generate two new nodes through expansion and then perform random simulations.

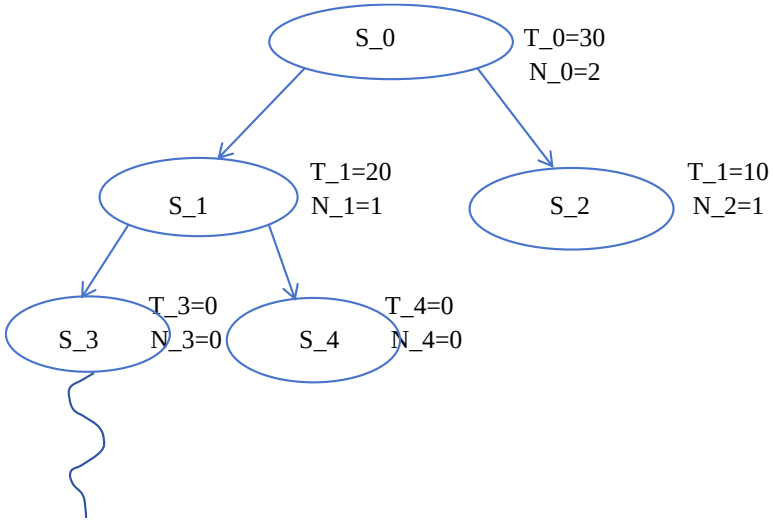

Figure 1. UCB demonstrates an example (Picture credit: Original)

2.4. Evaluation Function

The evaluation function is used to quickly estimate the quality of the current game state during the MCTS simulation process. For Chinese chess, its influencing factors not only include the number of pieces but also depend on their positions, the level of coordination between pieces, and the safety of the king and general. As shown in table 1, this is the piece value at the opening stage. Generally speaking, the value of a rook can be equivalent to 1200 points, while the cannon has a higher value at the opening stage due to its larger number of pieces and more support structures, whereas the knight has fewer supporting pieces compared to the rook, so the cannon's value is slightly higher than that of the knight [2].

Table 1. Initial score of subforce

rook	knight	cannon	Troops (soldiers)	Advisor/Bishop
1200	560	570	100	240

The position of the pieces is also a crucial factor in evaluating the board. Generally, large pieces (specifically chariots and horses) placed on the front line are far superior to those staying within their own camp; the unique rules of Chinese chess further confirm that pieces located in the center are better than those in the corners or sides. Each large piece has its specific advantageous positions: for example, a horse jumping up to occupy the river mouth is a good position for controlling the game; the horse's underpass and corner positions are excellent for threatening the king (or general). Similarly, a cannon's direct shot can be used in conjunction with various checkmate techniques to threaten the king (or general); a cannon's side line, when paired with a chariot, poses a threat to the king (or general). Additionally, the coordination and flexibility between pieces are also important factors to consider. If the pieces are concentrated in the center but crowded together, it may not be an ideal position. Lastly, the safety of the king (or general) cannot be overlooked. A game should not be judged solely by the number and position of the pieces; the safety of the king (or general) significantly influences the direction of the game. In summary, the evaluation function provides guidance and prerequisites for Monte Carlo tree search, while fine-tuning each influencing factor leads to the engine's playing style.

2.5. UI Debugging Tools

The UI intuitively presents the board layout and piece status, facilitating user interaction with the engine. During debugging, the UI can display the engine's moves in real-time on the terminal for user record-keeping; meanwhile, based on the UCCI engine interface, the UI has added a new MCTS engine interface, allowing users to set up game positions and switch engines [5].

2.6. Opening Library and Endgame Library

Importing opening and endgame databases into the engine is also crucial. The first 10-15 moves of an opening are known as the opening stage, during which many opening patterns and official moves have been summarized from previous research [2]. When the number of pieces is less than or equal to three, the game transitions from the middle game to the endgame, a phase where many winning strategies have been established by predecessors. By importing these opening pattern databases and endgame rules, it can significantly reduce unnecessary searches and ease the training load.

3. Research Results and Innovations

3.1. Research Results and Analysis

Research on the Chinese chess engine based on Monte Carlo Tree Search (MCTS) has made breakthrough progress, with its core achievements reflected in two major dimensions: game win rate

and decision efficiency. In 100 matches against traditional engines like TianTian Chess, the MCTS engine playing red can achieve a perfect win against beginner software engines, with a 62.5% win rate (excluding draws) against intermediate software engines; the MCTS engine playing black only loses one game out of ten against beginner software engines, while the outcome is evenly split against mid-level software engines. Notably, due to the initial parameter settings, the decision response time is up to 500ms, so the Monte Carlo Tree Search engine cannot always calculate the best move strategy. Since the engine is still in the early stages of practical play without extensive exposure to real-game scenarios, some brilliant moves were not fully explored. However, practical experience has shown that setting a longer decision time increases the probability of finding optimal moves among numerous mid-game tactics. This indicates that after a period of searching, incorrect or soft moves can be eliminated during the decision-making process, thus enabling the engine to achieve the optimal solution for the game situation.

3.2. Comparison with Traditional Engines

The Monte Carlo Tree Search algorithm engine demonstrates differences from traditional engines in both performance and principles. MCTS adopts a parallel simulation architecture, pruning low-value branches and dynamically adjusting the expansion factor, which allows it to find the best moves faster under the same conditions; traditional engines, due to their need to traverse fixed-depth search trees, tend to encounter computational bottlenecks in complex scenarios such as mixed battles involving rooks, bishops, and cannons. Additionally, in grasping "rare moves," the MCTS engine has an advantage, as it explores asymmetric strategy spaces through simulation.

Compared with the traditional engine, this study makes the following innovations:

(1) Dynamic resource allocation. MCTS can focus on the key moves of a certain step, and this flexible method of allocating computing resources can efficiently analyze the chaotic situation, thus saving time and effort [7].

(2) The hybrid strategy generation framework calls the opening library and the endgame library to make sure that the experienced moves do not go wrong or soft, while the complex middle game can show novel moves that may be better than the previous strategy, which is also an innovation point.

(3) Lightweight architecture design. Through GPU-accelerated parallel simulation technology, the engine can both comply with the rules and be flexible [8].

4. Conclusion

Through multiple rounds of experiments, the Chinese chess engine based on MCTS has demonstrated unique performance. In the brief decision-making time, the win rate can reach a basic engine level. During the middle game phase, optimal moves can typically be obtained in shorter and more efficient time frames, thanks to the simulation mechanism of the MCTS principle and the dynamic balancing characteristics of the UCT formula. This study successfully introduces the MCTS algorithm into the design of a Chinese chess engine, achieving an intelligent chess system. Additionally, it innovatively designs a dynamic expansion factor adjustment mechanism, allowing the search depth to vary with the complexity of the position, contributing to the development and innovation of chess engines. More importantly, this study validates the transfer potential of MCTS in non-complete information games, providing a reusable algorithm framework for AI development in other board games such as Shogi and Korean Chess.

For the limitations of MCTS algorithm in computational power and accuracy under complex scenarios, adjustable simulation strategies can be employed to enhance accuracy over short periods. In the future, residual convolutional neural networks can be introduced to build scenario value assessment models, while advanced algorithms such as reinforcement learning can be integrated. A reward function with 12-dimensional features, including sub-king value, formation structure, and threat level, can be designed to enable the engine to learn and optimize itself. On the application side,

human-machine collaboration modes can be explored, developing intelligent training systems with personalized chess style adaptation capabilities, promoting deep integration and innovation between artificial intelligence and traditional chess skills.

References

[1] C. Zhu. Research on the Popularization of Chinese Chess Culture in Chinese Language Reading Teaching. Secondary School Curriculum Guidance (Teacher Education), 2020, (17): 30.

[2] X. H. Pei. Research on Chinese Chess Search Engine Based on Pruning Strategy (Master's Thesis, Hebei University). (2009).

[3] Z. Y. An, H. Baier, D. Abhishek, M. Ayan. Enabling MCTS Explainability for Sequential Planning Through Computation Tree Logic. 4068 - 4075. (2024).

[4] B. Ronit. Structure and Reduction of MCTS for Explainable-AI. arXiv:2408.05488 1246 - 1253. (2024).

[5] H. J. Zhang. Design and implementation of Chinese chess interface and search engine in computer. Xi'an University of Technology, 2009.

[6] H. Li, X. Y. Pang, B. X. Sun, et al. A concise review of intelligent game agent. Entertainment Computing, 2025, 52:100894 - 100894.

[7] S. R. Yan, N. X. Liu, Computational design of residential units' floor layout: A heuristic algorithm. Journal of Building Engineering, 2024, 96:110546 - 110546.

[8] A. Papadopoulos, I. Kyriakou, Y. Matsuya, et al. Analytic and Monte Carlo calculations of dose-mean lineal energy for 1 MeV-1 GeV protons with application to radiation protection quality factor. Radiation and environmental biophysics, 2025: 1 - 19.