International Core Journal of Engineering Volume 9 Issue 7, 2023
ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

Research on Search Algorithm in Surakarta Chess Game System

Tao Zhang, Yefeng Jiang, Bowen Li

School of Computer and Software Engineering, University of Science and Technology Liaoning, Anshan 114051, China

Abstract

This paper introduces Surakarta, an important branch in the field of computer games, and discusses two search algorithms used in this game: Alpha-Beta pruning algorithm and UCT algorithm. The search algorithm plays a central role in the game system and affects the performance and efficiency of the system. Alpha-Beta pruning algorithm is an optimized and improved maxima-minimum algorithm, which reduces unnecessary search branches and improves search efficiency through pruning principles. The UCT algorithm is a game tree search algorithm based on the Monte Carlo method, which estimates the true value of the action through random simulation and tree building and expansion, and adjusts the strategy to the best priority strategy. The application of these two algorithms in the Surakarta chess game system can improve the performance and accuracy of computer games. Through the research and application of Surakarta chess, it can not only promote the development of artificial intelligence in chess games, but also provide enlightenment for problem solving in other fields.

Keywords

Surakarta Chess; Alpha-Beta Algorithm; UCT Algorithm; Monte Carlo Algorithm.

1. Introduction

Computer games are an important branch of the field of artificial intelligence, focusing on research strategies and intellectual challenges. It uses problem-solving and search techniques to apply artificial intelligence, and is also a key field for evaluating the state of artificial intelligence development. Because of its intelligent characteristics, board games became the focus of early artificial intelligence research, and are considered to be the "fruit flies" in the field of artificial intelligence. The study of computer games has brought many important methods and theories to the field of artificial intelligence [2]. From the computer program "Deep Blue" that defeated the world chess champion Garry Kasparov to the Go game program "AlphaGo" that defeated the world champion Lee Sedol, the breakthrough of computers in the game field has once again attracted widespread attention and discussion. As one of them, Surakarta chess has been enthusiastically sought after by many machine game enthusiasts in recent years.

The board of Surakarta is different from other board games, and its rules are very interesting, so it attracts many fans of machine games. Through the study of Surakarta, we can explore new algorithms and strategies, promote the development of artificial intelligence in chess games, and provide inspiration for problem solving in other fields.

2. Introduction to Surakarta

Surakarta is a two-player game originating from the island of Java, Indonesia. The chessboard is a square composed of 6 horizontal lines and 6 vertical lines, with a total of 36 intersections as chess pieces.The positions on the chessboard are connected by 8 arcs, usually represented by two pieces of different colors.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

The basic rules of Surakarta are as follows:

(1) At the beginning, decide which side starts first by tossing a coin. Only one piece can be moved at a time, and the two players take turns playing chess.

(2) If there are no other pieces in the target direction, you can choose any piece to move one grid in eight directions (up, down, left, right, upper left, lower left, upper right, and lower right).

(3) If you want to capture the opponent's pieces, the path of movement must be horizontal or vertical, and must pass through an arc tangent to the path. The movement path cannot be blocked by your own pieces.

(4) The black pieces can capture the red pieces, and the red pieces can also capture the black pieces on the same path in the opposite direction.

(5) When all pieces of one side are taken by the opponent, the game is over, and the side with more pieces remaining wins.

(6) In computer games, if each game lasts more than 30 minutes, both sides will usually stop, and the side with more remaining pieces will win.

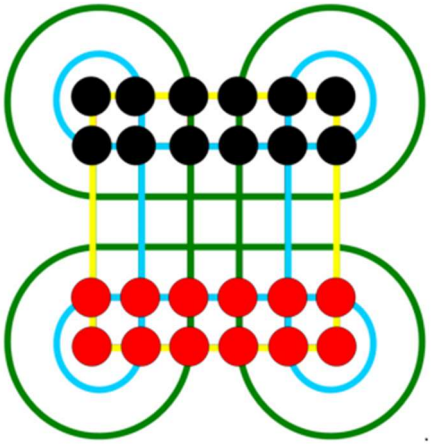

Figure 1. Surakarta pieces, board and initial layout[3]

3. Analysis of Algorithms

In the Surakarta game system, the search algorithm is one of the core components. Together with rules and evaluation functions it forms a complete system. The quality of the search algorithm directly affects the performance and efficiency of the Surakarta game system. The purpose of search is to explore the available knowledge during the reasoning process according to the actual problem, and find a path to solve the problem with less cost. In the field of computer games, Alpha-Beta algorithm and UCT algorithm are algorithms with high search efficiency. Applying these two algorithms to the game system of Surakarta, a search algorithm more suitable for Surakarta can be obtained.

4. Alpha-Bate Pruning Algorithms

4.1 Minimax Algorithm

Alpha-Beta pruning algorithm is a search algorithm optimized and improved on the basis of the maximum and minimum algorithm [4]. The theoretical basis of the algorithm can be traced back to 1950, when Professor Claude Shannon first proposed the "maximum-minimum algorithm", thus laying the foundation for computer game theory. According to this algorithm, the two sides of the game can be regarded as a zero-sum game, in which one party tries to choose the decision that can maximize its own positional advantage among the available nodes, while the other party tries to minimize the opponent's positional advantage.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

The game process can be regarded as a game tree that grows backwards, starting from the beginning, going through the middle game, and finally reaching the end of the game. The root node of the game tree starts from the first move, and it is a tree with a height of N that grows backwards. In the game tree, the nodes of each layer are assigned the roles of Max and Min alternately, and the leaf nodes are evaluated using the preset static evaluation function. The selection of each step is based on the larger valuation, and each node of each layer is assigned a value in turn. For the parent node of the Max node, it will assign the maximum value in the child nodes to itself; and for the parent node of the Min node, it will assign the minimum value in the child nodes to itself. Through continuous iteration in this way, it is finally possible to determine which child node can maximize its own situational advantage, or minimize the opponent's situational advantage.

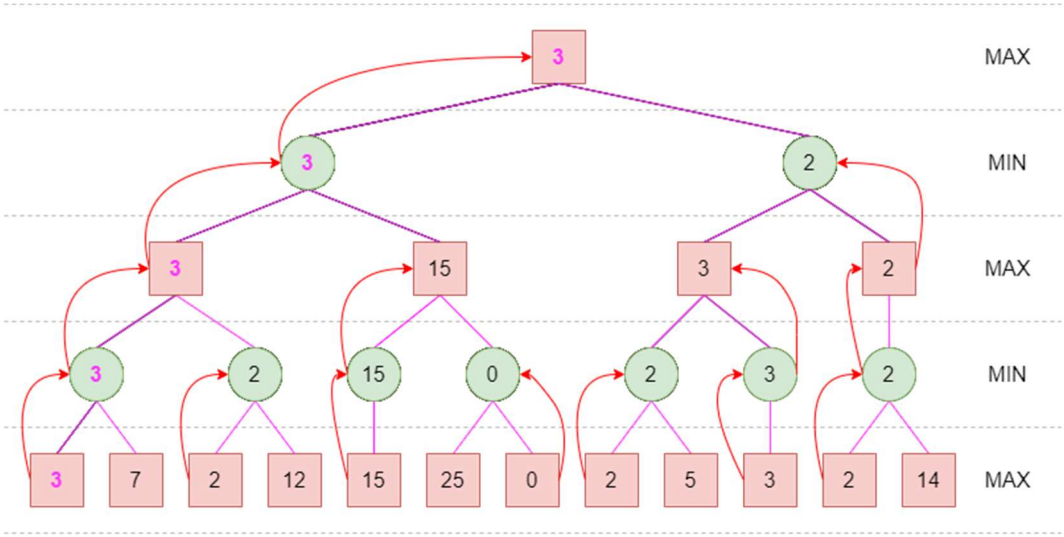

Figure 2. Game Tree of Maximin Algorithm

However, due to the exponential growth of the game tree, it is not feasible to completely generate the entire game tree for searching. Therefore, a static evaluation function is usually defined to evaluate the momentum of the current game state for searching. In the process of the game, each player moves alternately on the chessboard, which creates a search demand for this kind of chess game. In the minimax search strategy, the main consideration is that after several steps, the two sides choose the move with a greater relative advantage from the possible moves. Due to the huge space of the game tree, it is impossible to search the leaf nodes of the entire game tree within the specified game time, so it needs to be pruned.

4.2 Optimization and Improvement of Minimax Algorithm

In the max-min search algorithm, there is indeed a certain amount of redundant data. For example, in Surakarta, when black finds a win at the first node, it stands to reason that there is no need to continue searching the remaining nodes. However, the minimax search algorithm still expands the branches of the game tree and evaluates each leaf node, and then backtracks to the root node for comparison. Therefore, there are redundant data in the max-min search algorithm, which seriously wastes computing resources.

Alpha-Beta pruning algorithm is an effective solution to this data redundancy problem. This algorithm uses the principle of depth-first traversal and pruning, so that it is not necessary to fully expand the entire game tree, so it is more efficient than the max-min search algorithm [5].

In the Alpha-Beta pruning algorithm, two parameters alpha and beta are introduced and passed to the recursive minimax function. Initially, alpha represents the worst case of Max, and beta represents the worst case of Min, so their initial values are negative infinity and positive infinity respectively. In the process of depth-first traversal, in the round of Max, if the value of the node is greater than alpha, the

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

value of alpha is updated and gradually increased; in the round of Min, if the value of the node is less than beta, the value of beta is updated , gradually decreasing. During the recursive call, alpha and beta are constantly changing. As the search deepens, their scope gradually narrows, that is, more and more content falls outside the scope, and the efficiency of pruning becomes higher and higher. When alpha and beta are close (alpha>=beta), it indicates that all child nodes of this node will not have better results for both Max and Min, so the search for this child node can be skipped.

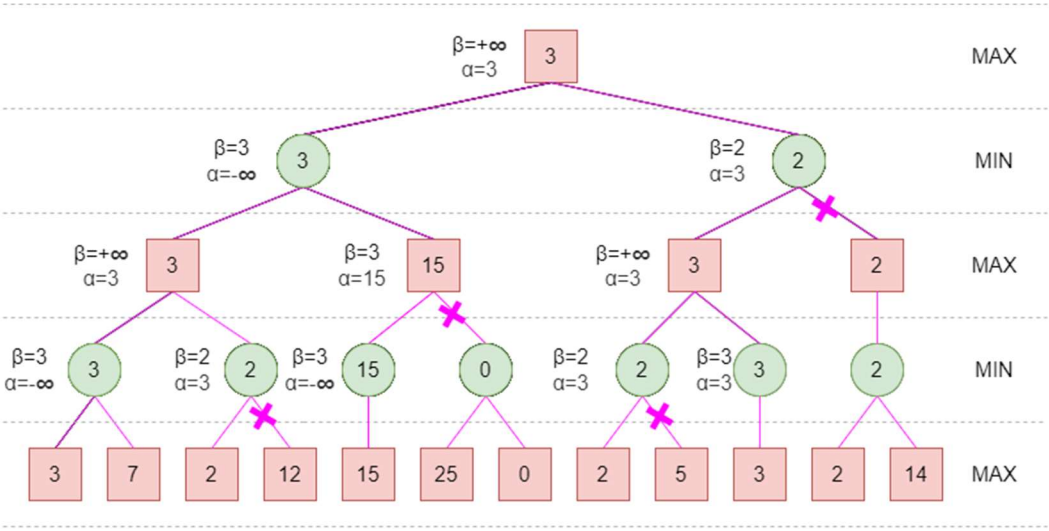

Figure 3. Alpha-Beta Pruning Algorithm Game Tree

The Alpha-Beta pruning algorithm can effectively reduce the number of search branches and improve search efficiency. Cutting out those branches that do not need to be further searched enables the computer game system to find the best move faster and saves the use of computing resources.

5. UCT Algorithm

5.1 Monte Carlo Method

Monte Carlo methods are a powerful tool for making optimal decisions in artificial intelligence problems. Its core idea is to approximate solve the problem through multiple simulations or sampling by associating with a probabilistic model [6]. The Monte Carlo method is widely used in numerical calculations and solutions to optimization problems, especially for those problems with high complexity and cannot obtain accurate solutions within a limited time. With the continuous improvement of computer performance and parallel computing technology, Monte Carlo method has received more and more attention and utilization in the field of artificial intelligence.

Monte Carlo Tree Search (MCTS) is an advanced algorithm that applies the Monte Carlo method to game tree search [7]. It explores the possible optimal moves by gradually building and expanding the game tree to improve the accuracy of the search. MCTS combines the characteristics of stochastic simulation and game tree search, so that those branches that may become optimal moves have more opportunities to explore. The algorithm is based on two core concepts: using stochastic simulations to estimate the true value of actions, and adjusting the policy to a best-priority policy by making efficient use of these estimates. MCTS gradually constructs partial game trees during the search process based on the previous exploration results of the game trees, and uses these trees to estimate the value of the move. As the tree is built, these estimates will become more accurate, allowing MCTS to find better decisions.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

5.2 UCT Algorithm

UCT (Upper Confidence Bound applied to Trees) algorithm is an algorithm that combines Monte Carlo Tree Search and UCB (Upper Confidence Bound) formula, also known as upper confidence interval algorithm. The algorithm obtains the optimal solution by simulating a large number of nodes.

During the search process of the UCT algorithm, the search depths of different branches can be different, which means that some nodes can obtain deeper search depths. The purpose of this is to balance the trade-off between exploration and utilization, both to ensure the exploration of unknown areas and to make full use of known information. By applying the UCB formula, the UCT algorithm considers the balance between exploration and exploitation when choosing the next action.

The calculation formula of UCT algorithm is:

$$UCB = v_i + c \times \sqrt{\frac{2\ln\left(\sum_i T_i\right)}{T_i}} \tag{1}$$

In the UCT algorithm, v_i represents the average value of all simulation results where the node n_i (non-leaf node) is the root node, that is, the winning rate of the node n_i. By simulating the node multiple times, recording the number of victories and calculating the average win rate, the pros and cons of the node can be evaluated. T_i represents the number of visits to node n_i, that is, the number of times the node is selected by the selection strategy. This value is used to balance the trade-off between exploration and exploitation. The more times a node is selected, it means that the node is considered as a more potential choice. In addition, a constant c is used in the algorithm to adjust the balance between depth-first search and breadth-first search. This constant controls the ratio of exploration to exploitation in the UCT algorithm. A larger value of c will prompt the algorithm to explore more unknown areas, while a smaller value of c will pay more attention to utilizing known information.

By calculating the UCB value of the node, the UCT algorithm selects the node with a higher UCB value for expansion and simulation, thereby gradually building and updating the game tree. By continuously simulating and updating the number of visits and the winning rate of nodes, the optimal action strategy can be finally found.

The execution process of the UCT algorithm is shown in the following figure 4:

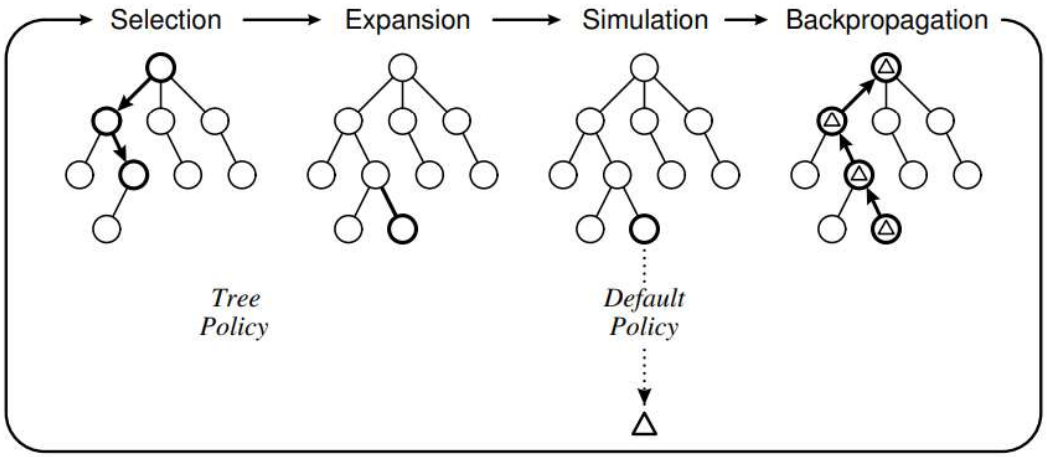

Figure 4. Basic Flow of UCT Algorithm

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

(1) Selection: Starting from the root node, gradually select the optimal child node until reaching the leaf node. The basis for selection is usually to use the UCB formula to calculate the UCB value of the node, and select the child node with the highest UCB value for expansion.

(2) Expansion: When a leaf node is reached, all legal child nodes of the node are added to the search tree, and the corresponding evaluation value and visit count are initialized for each new node.

(3) Simulation: For each new node obtained by the expansion, the process of randomly simulating the chess game is performed until the final state is reached. During the simulation process, random moves are selected according to the rules of the game.

(4) Backpropagation: Once the simulation is complete, propagate the simulation result (win or fail) back to all nodes on the search path. Update the visit count and evaluation value of each node for reference during the next selection process.

The UCT algorithm gradually builds and updates the game tree through the process of continuous selection, expansion, simulation and backpropagation to obtain the optimal action strategy. Through multiple iterations, a child node with the highest winning rate is finally found as the best action choice. In each step of the selection process, the UCB formula is used to balance the trade-off between exploration and utilization to ensure that better nodes have more opportunities to be selected. The simulation process is performed by randomly simulating the game to obtain an approximate evaluation of the winning rate. Through backpropagation, the simulation results will be propagated back to the root node, and the visit times and evaluation values of the nodes will be updated to provide a more accurate basis for selection.

Through the iterative execution of the above four processes, the UCT algorithm can gradually optimize the search path and find the optimal action strategy, which is especially suitable for solving complex game problems. UCT algorithm has a wide application value in solving complex decision-making problems and games with huge search space. It can find a strategy close to the optimal solution in a limited time and find a balance between exploration and exploitation.

6. Analyse and Compare

6.1 System Realization

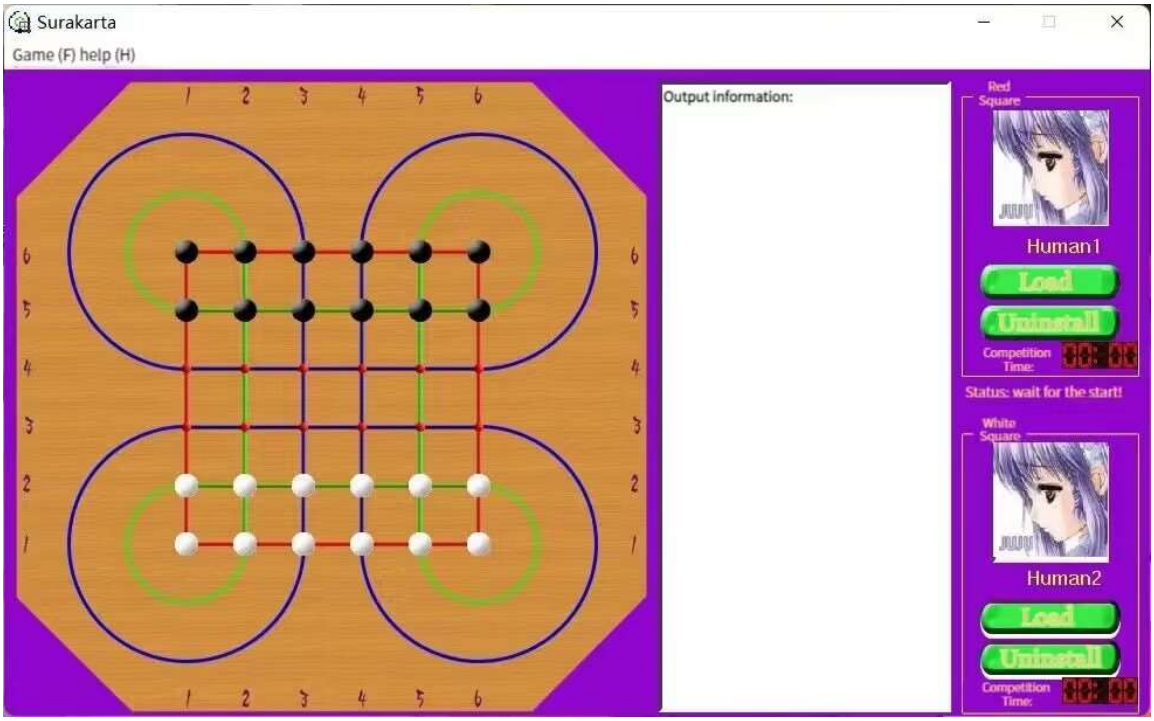

Figure 5. Surakarta Chess Battle Platform

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

Based on the research on the above algorithms, this paper uses C++ language to implement the Surakarta game system based on the Alpha-Beta pruning algorithm and the Surakarta game system based on the UCT algorithm in the Visual Studio 2022 environment. In order to facilitate the statistical comparison after the game, we use the Surakarta Chess Battle Platform of the Chinese College Student Computer Game Competition for testing. After each game is over, the system will automatically generate game records and move information, which will help us to carry out subsequent analysis and evaluation.

6.2 Result Analysis

According to the data in Table 1-Table 3, it can be seen that the Alpha-Beta pruning algorithm has certain advantages in terms of chess strength compared to the UCT algorithm. Regardless of the length of the search time, the Alpha-Beta pruning algorithm has a winning rate of more than 80%. When the search time is longer, the advantages of the Alpha-Beta pruning algorithm are more obvious. Therefore, from the point of view of chess strength, the Alpha-Beta pruning algorithm is more suitable for Surakarta chess.

Table 1. The game results of different algorithms under the 30s time limit

Algorithm	Time Limit (s)	Number of Victories	Number of Failures	Winning Rate
Alpha-Beta Pruning Algorithm	30	16	4	80%
UCT Algorithm	30	4	16	20%

Table 2. The game results of different algorithms under the 60s time limit

Algorithm	Time Limit (s)	Number of Victories	Number of Failures	Winning Rate
Alpha-Beta Pruning Algorithm	60	17	3	85%
UCT Algorithm	60	3	17	15%

Table 3. The game results of different algorithms under the 120s time limit

Algorithm	Time Limit (s)	Number of Victories	Number of Failures	Winning Rate
Alpha-Beta Pruning Algorithm	120	19	1	95%
UCT Algorithm	120	1	19	5%

7. Conclusion

This paper compares the performance of Alpha-Beta pruning algorithm and UCT algorithm in computer games, and draws some important conclusions. The research results show that under fair conditions, the Alpha-Beta pruning algorithm has higher search efficiency than the UCT algorithm, and shows greater advantages in time-limited game competitions. In addition, by introducing techniques such as endgame optimization and iterative deepening, the search efficiency of the Alpha-Beta pruning algorithm is further improved. The Surakarta game program developed using this algorithm won the second national prize in the Surakarta game in the Chinese College Students Computer Game Competition, which further verifies the effectiveness of the algorithm.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202307_9(7).0039

Acknowledgments

Fund support: 2023 University of Science and Technology Liaoning College Students Innovation and Entrepreneurship Training Program Project.

References

[1] Li Hao. Research and Implementation of Algorithm Optimization of Gomoku Human-Machine Game [D]. Dalian Maritime University, 2020. DOI: 10.26989/d.cnki.gdlhu.2020.000523.

[2] Li Xiali, Chen Yandong, Yang Ziyi, Zhang Yanyin, Wu Licheng. A Two-Stage Computer Game Algorithm for Tibetan Jiuqi [J]. Journal of Chongqing University of Technology (Natural Science), 2022, 36(12):110-120.

[3] Zhang,T.,&Jiang,Y.(2023).Design and Realization of Surakarta Chess Game Program.Frontiers in Computing and Intelligent Systems,3(1), 130–133.https://doi.org/10.54097/fcis.v3i1.6349.

[4] Li Dongxuan, Hu Wei, Wang Jingwen. Research on Surakarta Game System Based on Alpha-Beta Algorithm [J]. Intelligent Computer and Application, 2022,12(2):123-125.

[5] Yue Jinpeng, Feng Su. Research and Improvement of Alpha-Beta Search Algorithm in Chinese Chess[J]. Journal of Beijing Normal University: Natural Science Edition, 2009(2):156-160.

[6] Guan Yanxia, Liu Xunyun, Liu Yuntao, Xie Min, Xu Xinhai.Research on Parallel Monte Carlo Tree Search Algorithm for Multi-Agent Games[J].Computer Engineering and Science,2022,44(12):2128-2133.

[7] Ji Hui, Ding Zejun. Improvement of Monte Carlo Tree Search Algorithm in Two-Player Game Problems[J]. Computer Science, 2018,45(1):140-143.

[8] Zhang Xiaochuan, Li Qin, Nan Hai, Peng Lirong. Application of Improved UCT Algorithm in Einstein Chess[J]. Computer Science, 2018,45(12):196-200.

[9] Zhou Mingming, Gao Hang, Zhao Guoan. Application and Improvement of UCT Algorithm in Computer Go[J]. Data Acquisition and Processing, 2012(S2):330-335.