Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 8, No. 2, 2024

Application of Multi-threading Mechanisms to the Connect6 Gaming System

Yang Cao [a], Meina Zhang *, Xianjun Wang [b], Yaqin Shang [c], Simiao Jia [d]

University of Science and Technology Liaoning, Anshan Liaoning, 114000, China

*** Corresponding author:** Meina Zhang (Email: zhangmeina@163.com), [a] 2557436813@qq.com, [b] 476690826@qq.com, [c] 16642290595@163.com, [d] 1718949950@qq.com

Abstract: The field of computer games is currently a popular research direction in the field of artificial intelligence, and has a pivotal position in the development and research of artificial intelligence. In the field of computer games, Connect6 is an emerging chess game, whose position complexity and research value are comparable to those of Go and Xiangqi. In the Connect6 game system, there are a large number of branching positions, and the traditional single-threaded search mechanism is time-consuming and laborious, which reduces the search efficiency of the game to a certain extent, so how to find the best positional moves quickly becomes an important factor affecting the strength of the Connect6 game. In this paper, by introducing the multi-threading mechanism in Connect6 intelligent confrontation system, assigning a thread to each branch position, utilizing the computer's multi-threaded parallel computing mechanism combined with a reasonable evaluation method will be able to quickly get the best move for the next move. Through experimental comparison, it can be obtained that in the field of Connect6 game, the multithreaded search mechanism is more efficient for the position search, and also has better game prediction, which can improve the search efficiency and reduce the search time of Connect6 game to a certain extent.

Keywords: Connect6; Multithreading; Gaming; Evaluation Function.

1. Introduction

The history of the development of Artificial Intelligence technology can be traced back to the 1950's. For decades, Artificial Intelligence, AI, as an emerging cross-discipline, has been researched with the aim of exploring how to enable computers to exhibit a way of thinking comparable to that of human beings, and to simulate the way human beings think by using computers' powerful computational power combined with appropriate computational strategies, so that the computers have the ability to surpass humans in some fields. With the continuous development of computer science and technology and the continuous improvement of artificial intelligence theory, in the rapid development of science and technology at the same time, extending a number of technical research directions, such as machine gaming, deep learning, computer vision, natural language processing and so on. Among them, machine game is an important research branch in the field of artificial intelligence in the current frontier field of computer research, through the research on the field of machine game can help human beings to solve many practical problems in reality, and can make the machine intelligence to the real artificial intelligence step forward.

In the field of machine games, there is a very classic term called game strategy, which can be understood as a methodology for computers to make decisions and choose the optimal decision in the game process, which has a lot of similarities with real-life decision-making in politics, economy, and daily life. Game strategy theory is widely used in the design of many board games, such as Connect6, Backgammon, Xiangqi, Weiqi and so on. Different game strategies determine the strength of the game, and in an intricate situation, there are a large number of feasible solutions, so how to quickly find a suitable optimal solution becomes an important criterion for evaluating the strength of a machine gaming program. In this paper, we take Connect6 as an example, and design a search method with multi-threading mechanism search strategy in Connect6 gaming system to discuss the influence of multi-threading mechanism on improving the efficiency of gaming search, and apply PVS search algorithm and TSS forced search algorithm to find the optimal position of discovery in the realization, which further confirms that the assisted search of multi-threading mechanism is important for improving the efficiency of gaming search and enhancing the strength of Connect6. and improve the chess power of Connect6.

2. Connect6 Gaming

2.1. Connect6 Background and Rules

Connect6 was first proposed by Prof. Cheng-Yi Wu of the National Chiao Tung University in Taiwan on the basis of backgammon as one of the board games in the K-board series, and it is a variant of backgammon, formally known as Connect6. [7] Connect6 introduces some new rules and variations to make up for the disadvantage of fairness of backgammon, where the player who is the first to go has an advantage in the situation. The rules of Connect6 are similar to those of traditional backgammon, with the victory condition being that the first player to get six tiles in any direction on the board wins the game. In the opening game, both players hold black and white pieces. Black first places a piece on the board, and then White places two pieces on the board, after which both players take turns placing two pieces on the board, and if the board is full and one player has not yet won, the game is called a draw. The 19 way board of Go is still commonly used in current Connect6 games.

2.2. Connect6 Game Algorithm

In Connect6 gaming system, search algorithm, valuation function and move generation are the key technologies of the six pieces gaming system[8] , which determines the power of

the gaming system, and borrow other computer technologies, such as GPU-assisted computing, threading mechanism, and opening libraries to optimize the search efficiency in the game searching process, so as to improve the computer's gaming level and reduce the computation time in the opening, which together constitute the Connect6 gaming system. The Connect6 game algorithms include Alpha-beta pruning search algorithm, UCT search algorithm, MTD search algorithm, etc.[5] . Among them, the UCT algorithm is less accurate in evaluating the score of the game, and the alpha-beta algorithm has a large number of nodes after pruning. The Connect6 algorithm used in this paper is an algorithm that improves the pruning efficiency by narrowing the search scope on the basis of the Alpha-beta pruning search algorithm, which is also known as Principal Variation Search, PVS algorithm, also known as the Very Small Window Search. The PVS algorithm assumes that the score of the first node is the optimal one in the search process, and the score of the first node is the optimal one. of the first node in the search process is the optimal score, as the main variable of the search, full-window search, while the other nodes are zero-window search, to determine whether there is a value of $\alpha \geq \beta$. If the subsequent node is better than the current best node, the search continues with that node as the optimal node. By using a very small window to build a game tree with a small range as a way to achieve efficient search, the PVS algorithm relies on the guessing of small changes in the successor node relative to the current predecessor node. In addition, in the process of game search, hash table and replacement table are used to save the value scores that have been calculated by the nodes, by checking the table to determine whether the node currently searched for exists or not, if it exists directly in the replacement table to obtain the value scores, this method speeds up the search process to a certain extent [2] . The main pseudo-code of the PVS algorithm is as follows:

```
Algorithm PVS Algorithm
Require: α (Minimum limit), β (Maximum limit), depth (Search depth)
Ensure: Best evaluation score
Define function PVS(α, β, depth)
if Winning condition met then
    return Score multiplied by (depth + 1)
end if
if Reached search depth then
    return Current evaluation score
end if
Call FINDEMPTYPOSITON function to search empty positions
if First recommended position invalid then
    return 0
end if
for each recommended move do
    for each valid move do
        Make move MAKEMOVE()
        for each valid second move do
            Make move MAKEMOVE()
            if First pair of recommended positions then
                Perform full-window search, decrease depth by one
            else
                Perform zero-window search, decrease depth by one
            end if
            if Score after move lies between α and β then
                Perform full-window search again, decrease depth by one, obtain
                precise score
            end if
            Update α or β
            Undo second move UNMAKEMOVE()
            if Pruning condition met then
                Undo first move, return α or β
            end if
        end for
        Undo first move
    end for
end for
return α or β (depending on whether it is a maximizer or minimizer node)
```

Figure 1. The main pseudo-code of the PVS algorithm

2.3. Game Tree

The game tree is an extremely important concept in the field of computer games. A game tree is a graphical structure used to represent the process of state transfer during a game. A game tree consists of edges and nodes, where each node in the tree represents a state of the game and each edge represents an action or transfer by a player [3]. The game tree starts from an initial state and describes the evolution of the game by continuously adding nodes and edges. In Connect6, the generation of game moves by both players corresponds to the unfolding and advancement of the game tree, and the game tree search structure is shown in Figure 2.

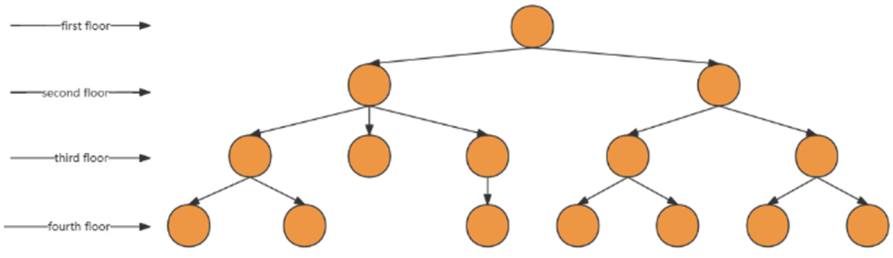

Figure 2. The game tree search structure

The expanded leaf nodes in the graph represent a possible state obtained by searching a certain branch of the game process, representing a positional state of the game, which may be the final positional state of the winner or the state of the game at a certain stage of the game between the two players. If the current node has won or lost, then it returns an extremely large value or an extremely small value. If there is no winner, then a backtracking[6] is required. Each edge of the game tree represents a drop, from the current situation to the next game state. For the PVS algorithm, it assumes the optimal value in the first search, and then builds an initial window, which is the valuation range of the situation, and recursively searches each node of the game tree from the heel node, and judges each node with a very small window, and updates the window according to the size of the return value of the subsequent node and judges whether it is necessary to carry out a more in-depth search, and judges whether it is necessary to carry out more in-depth search based on the value of the node in the process of searching. Alpha-beta pruning is performed to minimize the branches of the search.

However, despite the use of efficient search algorithms for pruning, the scale of a complete game is also very large, with a large number of game states, for Connect6, its game changes are very complex, the higher the complexity, the more nodes in the game tree, the more possible game states there are, the more difficult the search process and backtracking of the game tree is, which directly affects the search efficiency of the game tree. Search efficiency[9] , for Connect6, the complexity of its game tree can reach 10^{188} , it can be seen that the improvement of search efficiency is an important factor to improve the game level of Connect6.

3. Multi-threaded Search

3.1. Multi-threading Mechanism

A thread is the smallest unit of a program execution flow, which can run independently and share the same memory space. Multi-threaded search is a technique that accelerates the game tree search process by utilizing multiple threads at the same time, and is the main implementation of parallel computing. The use of parallel computing is to fully utilize the performance of the computer, the branches of the game tree are dynamically assigned to the threads to search, giving full play to the powerful parallel processing ability of the computer multi-core, which does not cut and reduce the size of the game tree[4] , by improving the speed of the game search process, and then improve the level of the game.

Since the search for a situation needs to be assigned a thread to complete, it is inevitable that parallel computing is needed, and the methods of parallel computing are: OpenMP, MPI and Windows Multithreading. OpenMp is a parallel programming technique based on shared memory multicore computing, which provides a set of instructions for parallelizing programs, and identifies the parallel region by using annotations, thus achieving the parallelization of code. parallelization of code. MPI is a parallel programming model widely used in distributed memory systems, a standard specification that provides a way to efficiently utilize multicore computing resources, initially originating in the MPI forum, with a large number of function development interfaces, commonly used for large-scale data processing tasks [1]. Windows Multithreading is an API for multithreaded development on the Windows platform, and the MFC is the main interface for multithreaded development on this platform. The multithreading technology used in Connect6 system in this paper is implemented in Windows platform using the thread class library in C++17 standard library, which is compatible with Window multithreading programming, and the use of multithreading mechanism becomes easier and more intuitive to use.

3.2. Design and Implementation of Multi-Threading Mechanism

In the Connect6 game, we chose to assign a thread to each position for search. First, at the beginning of the game, we use the opening bank to make the first moves, and when the game reaches a certain position, the program starts to jump out of the opening bank and use the algorithm to analyze the next moves. Due to the large number and depth of branches in the game tree, we limit the depth and breadth of the game in Connect6, assuming that the breadth of the search is 16. Before we conduct the next multi-threaded search, we first traverse the entire board, search for the top 16 highest-scoring drop locations, and then arrange them in descending order, so as to get the best drop locations of the game, and then combine the 16 drop locations into a combination of the 16 drop locations. After that, we combine these 16 moves, assign each move to a separate thread, and then replicate the current state of the board so that each thread can search on a separate copy. Each thread here corresponds to a possible positional situation at the same level of the game tree. After that, the main thread waits for all the sub-threads to finish executing, and then compares the optimal drop positions obtained by each thread. The optimal drop point position is selected and returned as the final result. Each thread corresponds to a situation, and the game process executed in this thread will simulate the drop and find the local optimal drop location, and finally return the best drop location found. The pseudo-code of the algorithm for the multi-threaded design implementation of the search is as in Figure 3:

4. Experimental Results and Analysis

In order to verify the effectiveness of the multi-threaded mechanism designed in this paper to assist the search to improve the search speed, the search strategy and single-threaded search strategy under the same search algorithm PVS algorithm for gaming test comparison. The game is played on a 19×19 chessboard, the game rules are the same as the Connect6 game rules described earlier, and the test machine only runs one software, the game program is SAU Game Platform. the local environment of the machine is Intel(R) Core (TM) i5-1135G7 @2.40GHz. the game is played on the same board. The CPU utilization and the average time between discs are shown in Table 1 for the condition that white loads the multi-threaded program engine and black loads the single-threaded serial engine:

Experimental data can be obtained, under the condition of the same search algorithm and other functions of the system remain unchanged, the efficiency and CPU utilization of the strategy using multi-threaded search is higher than that of searching using the single-threaded mechanism, which reduces the interval time between the fall of seeds in the process of the game, and enhances the game level of Connect6 to a certain extent.

Algorithm Multi-threaded Search Algorithm

Initialize the thread counter *threadnum* to zero
Traverse the array of empty positions to generate all possible move combinations
for each pair of valid empty positions **do**
  Create a move combination and allocate it to the move combination array [*threadnum*]
  $threadnum = (threadnum + 1)\%numberofthreads$
end for
Initialize an array of board objects *boards* with size equal to the number of threads
for i from 0 to the number of threads -1 **do**
  Call the *copy* method of the board to copy the board state to *boards*[i]
end for
Initialize an array of thread objects *threads* with size equal to the number of threads
for i from 0 to the number of threads -1 **do**
  Create a new thread to execute the function to find the locally best move combination
  Store the new thread in *threads*[i]
end for
for each thread *threads*[i] **do**
  Call the *join* method to wait for the thread to finish
end for
Summarize and filter the results of multi-threaded search

Figure 3. The pseudo-code of the algorithm for the multi-threaded design implementation of the search

Table 1. Comparison of Single-Range Multi-Range Efficiency

Search mechanism	Average CPU utilization	Average time between drops
Parallel multithreaded search	98.97%	15s
Serial single-threaded search	18.46%	53s

5. Conclusion

Machine gaming is a difficult and challenging research topic, but also full of fun. The Connect6 field of computer gaming is still waiting for people to explore and develop the learning and research on game strategies, and there are many unknown techniques waiting to be discovered. In this paper, based on the Connect6 gaming system, we design and implement a multi-threaded search strategy, combining the mechanism of multi-threading and the search algorithm of the program itself, making full use of the system resources of the computer, to a certain extent, we improve the search efficiency of the Connect6 gaming program, reduce the time consumed by the gaming search, and improve the level of chess power of the Connect6 gaming program. The program has improved the level of chess power of Connect6.

References

[1] HouXL. Research and application of parallel computing in computer gaming [D]. Chongqing University of Technology, 2015. doi:10.7666/d.D699391.

[2] H. F. Wang, J. W. Wang,Y. Li. Research on Connect6 gaming system based on PVS algorithm[J]. Intelligent Computer and Application,2021,11(2):97-100.

[3] Luo Jing, Ye Junmin, Zhao Liang, et al. Research on game tree based backgammon algorithm[C]. //Computer Science. :207-208, 225.

[4] Y. J. Wang, H. K. Qiu, Y. Y. Wu, et al. Research and development of computer gaming[J]. Journal of Intelligent Systems,2016,11(6):788-798. DOI:10.11992/tis.201609006.

[5] JIN Shuxian,GAO Ming,WANG Xiukland. Application of opening libraries in point grid chess computer gaming system[J]. DigitalTechnology and Application,2022,40(1):61-63. DOI: 10.19695/j.cnki.cn12-1369.2022.01.20.

[6] Kunbing Wang. Research and implementation of search technology in Connect6 game[D]. Anhui:Anhui University, 2016. doi:10.7666/d.y3018719.

[7] Min W.-J. Research on key technology of Connect6 computer game [D]. Chongqing:Chongqing Jiaotong University, 2010. DOI: 10.7666/d.Y1694468.

[8] Cuizhu Li. Research and implementation of Connect6 computer gaming system[D]. Chongqing University of Technology, 2010. DOI:10.7666/d. D538993.

[9] Miao Sha. Research and improvement of search algorithms in computer games[J]. China New Communication, 2023,25 (10): 61-63. DOI:10.3969/j.issn.1673-4866.2023.10.022.