Applied Mathematics, Modeling and Computer Simulation
C. Chen (Ed.)

doi:10.3233/ATDE220084

Research and Improvement of Alpha-Beta Search Algorithm in Gobang

Yuan XIE[a], Wenliang GAO[a,b,1], Zuxu DAI[a,b] and Yuanyuan LI[a,b]
[a] *School of Mathematics and Phisics, Wuhan Institute of Technology, China*
[b] *Hubei Key Laboratory of Optical Information and Pattern Recognition, China*

Absrtact. In allusion to the Alpha-Beta search algorithm which is widely used in Gobang intelligent algorithm, this paper presents an improved method, that is, the heuristic search method using static table, which reduces the search time. Then the differential evolution algorithm is used to optimize the chess shape parameters and the static table parameters. According to the simulation data, both methods improve the efficiency of Alpha-Beta search algorithm.

Keywords. Gobang, machine game, Alpha-Beta search, Differential evolution algorithm

1. Introduction

In the field of artificial intelligence, game is a very important branch. Computer game is the experimental field of artificial intelligence, known as the fruit fly of artificial intelligence. Computer games have attracted worldwide attention because of two major events: Deep Blue's victory over chess champion Garry Kasparov in 1997 and AlphaGo's victory over Go champion Lee Sedol in 2016. The valuable algorithms left behind by Deep Blue and AlphaGo are breakthroughs in computer games and bring new ideas to the field of artificial intelligence.

As a classic two-player game, Gobang has simple rules and the game process is easy to be simulated by computer, so it has been concerned earlier in the field of computer games. In 1994 and 2001, people used computer programs to prove that the first hand will win in the case of the original non-forbidden hand and the original forbidden hand[1-2]. However, compared with computer chess and go, the development of computer Gobang is slow, and many Gobang experts believe that current Gobang programs are still unable to surpass the best human players[3]. Therefore, the improvement and innovation of various Gobang algorithms are still widely concerned.

In recent years, the algorithm as well as software and hardware environment of Gobang have been optimized, and the playing strength of Gobang programs has been constantly improved. In terms of the minimax search and Alpha-Beta pruning tree algorithm based on game theory, Sun Shiwen used the minimax algorithm to optimize the whole search process of the game tree in 2018, and optimized the intelligent algorithm of Gobang[4]. The evaluation function of Gobang was optimized by using a

[1] Corresponding Author, Wenliang GAO, School of Mathematics and Phisics, Wuhan Institute of Technology, China, Hubei Key Laboratory of optical information and pattern recognition, China; E-mail: wenlianggao@wit.edu.cn.

relatively dynamic evaluation method[5]. Zheng Jianlei et al. proposed an improved Gobang type evaluation method and used multithreading technology to improve CPU utilization[3]. Deep learning is another important algorithm for Gobang. A self-learning intelligent Gobang algorithm was implemented by using Monte Carlo tree search and convolutional neural network with 32 convolutional layers[6]. Based on deep reinforcement learning, a composite vision field network was designed to enhance the network's comprehensive perception of global and local chess shapes[7]. Furthermore, deep learning has been introduced into strategy generation and situation evaluation of game decision-making mechanism to improve the fitting quality of evaluation function[8]. By the way, some scholars have designed complete sets of Gobang applications[9]. An Android Gobang application based on intelligent algorithm was designed, developed by using Alpha-Beta pruning tree algorithm, with a high degree of intelligence and can defeat most amateur players[9], while Cao Fengyun et al[10] has developed an integrated general Gobang game platform.

However, the current Alpha-Beta search algorithm still has some problems, such as the slow speed of deep search, the inaccurate evaluation of chess shape, and the low level of play. To solve the problem of slow speed of deep search, based on alpha beta search algorithm, this paper designs a heuristic search algorithm using static table. This algorithm calculates the heuristic value of static table for all blank points on the chessboard, and reorders the branch nodes with this value as a reference, so that the better nodes can be calculated in advance. The pruning proportion is increased, and the chess playing speed is improved. In order to solve the problem of inaccurate chess shape estimation, this paper uses differential evolution algorithm to randomly generate 8 arrays initially, each array contains 10 chess shape parameters and 8 static table parameters. Then differential evolution iterative operations such as mutation, crossover and selection are carried out on the initial array to optimize each parameter. With the increase of the number of iterations, the optimized parameters make the game level of the program improve continuously, and reach the peak at the 500th generation. The simulation results show that when the search depth is 1 to 6 layers, the number of search nodes can be reduced by 34% -96% by using the **ST** heuristic search. In the simulation game, the winning rate of the player using the optimized parameter group is 15% higher than that of the opponent using the experience parameter group.

2. Basic Algorithm

2.1. Minimax Search

The minimax search algorithm was proposed for the first time by Shannon in 1950[11], which laid the theoretical foundation of computer game[12][13]. According to the characteristics of Gobang, the method of minimax search and evaluation function can be used to design Gobang game program[14]. We use s to denote the current situation, and the situation evaluation function $v = f(s)$. The greater the value of v, the more favorable the current situation is to the player[5].

On the chessboard, the live four, live three, sleep three and other chess shapes of both sides pose a threat to the opponent. We assign a score to each chess shape , which is mainly used for minimax search and Alpha-Beta pruning, so we call it the basic algorithm chess shape parameters, or chess shape parameters for short.

It is assumed that the player whose turn it is to play will establish a search tree from many blank grid points. Each end node (leaf) of the search tree corresponds to a path from the root node to this end node, which corresponds to a situation s. then the chess shape parameters of all chess shapes in the situation can be accumulated to obtain the situation evaluation function value $v = f(s)$ [15].

Because the state space of Gobang is too large for the computer to exhaust, an upper limit of depth to the algorithm tree is generally set, and the value of v is the evaluation score[16] of this layer when outcome is not decided at the last layer.

From the root node to the end node, the algorithm tree sets the Max layer and the Min layer in turn. The minimax search assumes that the opponent is as smart as the player[17] and try its best to make the situation develop in the direction that is most advantageous to the player and most disadvantageous to the opponent. When it comes to the own layer, because the player will choose the move with the greatest value, the value of each node is the maximum value of its son nodes, so is also called the Max layer. When it comes to the opponent layer, it is assumed that the opponent will choose the path with the least value.Therefore, the value of each node is equal to the minimum value of its son nodes, and this layer is called the Min layer. In this way, through logical reasoning from the bottom to the top, we can get the optimal choice of the top.

2.2. Alpha-Beta Pruning Algorithm

Simple minimax search has a lot of redundancy[18] . In 1956, McCarthy proposed the Alpha-Beta pruning search algorithm[13]. In the search process, the maximum value of the Max layer selection is reserved as Alpha, and the minimum value of the min layer selection is reserved as Beta, both of which form an interval $[\alpha, \beta]$, which is called a window. The size of the window represents the value range of the child nodes of the current node value. The process of downward search is the process of narrowing the window, and the final optimal value will fall in this window. If the return value of a child node of a node in the Max layer is greater than the Beta value, pruning occurs; Pruning also occurs if the return value of a child of a node at the Min layer is less than the Alpha value.

Take the case as shown in figure 1 below as an example, suppose that the values of each branch are marked on points d, e, f and g. Give the initial windows $[-\infty,+\infty]$ to nodes a, b and c. after accessing node d, update the window of b as $[-\infty,3]$. After accessing node e, because $-1 < \beta = 3$, update the window of b as $[-\infty,-1]$, and then update the windows of a and c as $[-1,+\infty]$. Then access node f, because - 4 < + ∞, update the window as $[-1,-4]$, because $-4 < -1$, all branches rooted at node c can be cut off.

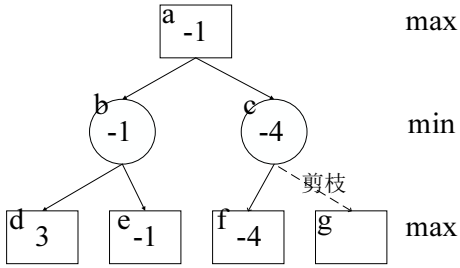

Figure 1. Alpha-Beta Pruning process

In 1975, Knuth et al[18] proved that the number of nodes generated by using Alpha-Beta pruning was about twice as large as the square root of the number of nodes of the minimax search algorithm in the case of optimal node arrangement, and it was concluded that the search depth of Alpha-Beta pruning can be twice as large as that without Alpha-beta pruning in the case of optimal node arrangement.

3. Algorithm Improvement

3.1. Static Table Heuristic Search

Five pieces of the same color are connected along the horizontal, vertical, left or right diagonal， which forms a chess shape called the winning combination. The goal of both players is to get the winning combination first. For a chess board of size 15×15 , there are 572 winning combinations, including 11×15 horizontal or vertical, 11×11 left or right oblique[10]. As shown in figure 2, seven different winning combinations and their corresponding numbers are listed:

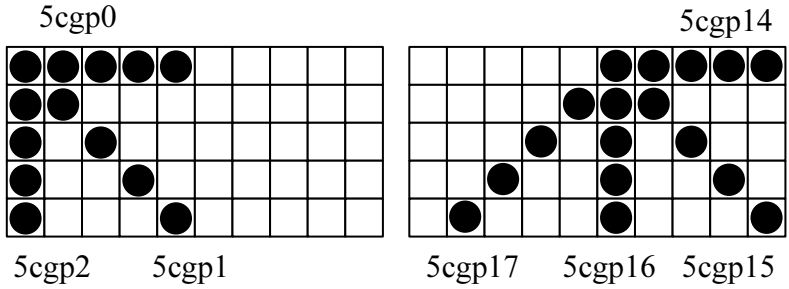

Figure 2. Examples of 7 different winning combinations

The five grid points occupied by each winning combination on the chessboard are five adjacent grid points in the four directions of horizontal, vertical, left oblique or right oblique diagonal, which are called five connected grid points (**5cgp**). Each **5cgp** on the chessboard corresponds to a winning combination, and they are given the same number. The number of chess pieces of both sides in each **5cgp** corresponds to a certain score referred to as static table parameters, which is used to generate a static table.

Table 1 below is the static table parameters given by experience. The more pieces a player has in a **5cgp**, the greater the threat it poses, and therefore the higher the score. In addition, OWN0-OWN4 or OPP0-OPP4 indicate that there are 0-4 own pieces or

opponent pieces in the **5cgp** respectively. In the actual competition process, players pay more attention to the threat of the opponent, so in general, under the same conditions, the opponent's score is higher, except that there are four own pieces in a **5cgp**, giving the player an absolute high score 20000. These parameters will be optimized later in this article.

Table 1. Empirical static table parameters

name	OWN0	OWN 1	OWN 2	OWN 3	OWN 4
value	1	200	400	2000	20000
name	OPP0	OPP1	OPP2	OPP3	OPP4
value	1	220	420	2100	10000

In the Alpha-Beta pruning search, we want to sort the child nodes of any node. For this reason, we score all the empty points on the chessboard, and the correspondence between all the empty points and their scores is called a static table.

The scoring method for a certain empty grid point is as follows: first, traverse all the **5cgp** in the four directions containing the empty grid point, then each **5cgp** corresponds to the number of pieces of both sides. Secondly, respectively sum the static table parameters according to these number of pieces of both sides. Finally, the static table values of both sides (or called static table heuristic values) are obtained.

For example, consider the situation shown in figure 3:

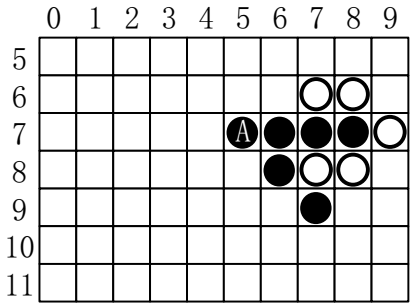

Figure 3. A situation in actual combat

The point $A(7,5)$ in figure 3 are located in 20 5cgps numbered 148, 162, 179, etc.

Assume it is black's turn to play. The 5cgp on the horizontal line containing point A and the corresponding static table parameters of the own side are shown in figure 4.

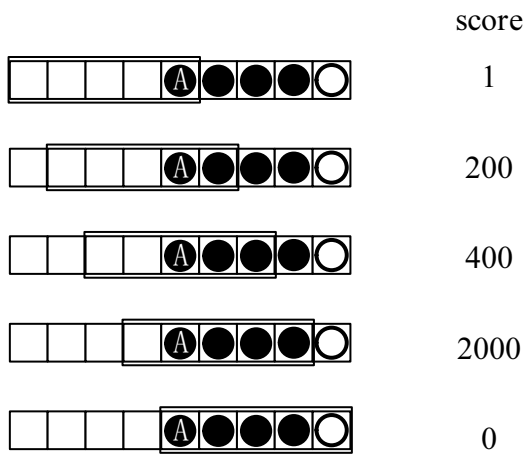

Figure 4. 5cgps and scoring on the horizontal line

Continue to calculate **5cgps** of other direction according to the method, the static table parameters of the own side at point *A* can be obtained:

$$own(A) = (h)1 + 200 + 400 + 2000 + (v)1 + 1 + 1 + 1 + 1 + 1 + 1$$
$$+ (lo)1 + 200 + 400 + 400 + 400 + (ro)1 + 1 + 1 + 1 + 1 = 4012$$

Similarly the static table parameters of the opponent side at point *A* can be calculated as:

$$opp(A) = (h)1 + (v)1 + 1 + 1 + 1 + 1 + 1$$
$$+ (lo)1 + (ro)1 + 1 + 1 + 1 + 1 = 12$$

Thus, the final static table value of point A is

$$score(A) = \max(own(A), opp(A)) = 4102.$$

This operation is performed on all empty grid points to obtain the static table value for each point. Figure 5 below shows the static table values for some of the empty points on the board.

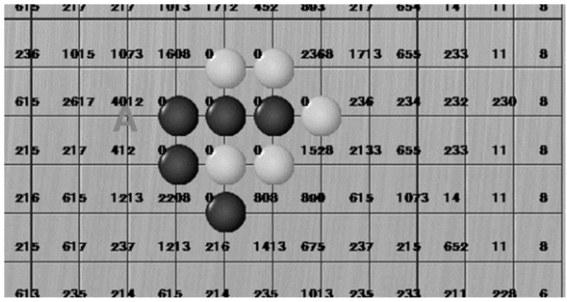

Figure 5. Static performance in self-made software

The nodes of the search tree are sorted according to the values of the static table, and the nodes with more potential can be preferentially searched. In the case of figure 3, point *A* is searched first, so that the search speed is increased.

The number of nodes searched before and after using the static table heuristic algorithm is compared by simulation experiments. The results show that when the search

depth is 1-6 layers, the number of search nodes can be reduced by 34% -96% by using the static table heuristic algorithm. The experimental results are shown in table 2.

Table 2. Comparison of Average Number of Search Nodes for Different Search Algorithm

Depth	Average of search nodes		Reduce proportion
	Before using ST	After using ST	
1	32	21	34%
2	90	54	40%
3	1524	540	65%
4	4513	1277	72%
5	37146	3138	81%
6	164210	6824	96%

It can be seen from table 2 that the static table heuristic method has a great effect on reducing redundant search, and with the increase of search depth, the effect becomes more and more obvious. This also proves that the pruning efficiency of Alpha-Beta algorithm is highly related to the search order of child nodes.

3.2. Optimization of Chess Shape Parameters

In 1995, Storn R and Price K proposed the Differential Evolution algorithm[20]. Its basic algorithm includes three operations: mutation, crossover and selection. The intermediate population is obtained by mutation and crossover operations, and the new generation population is obtained by the selection operation by using the competition between the offspring and the parents[21]. Compared with other swarm intelligence algorithms, such as particle swarm optimization and genetic algorithm, this algorithm has the advantages of less undetermined parameters, not easy to fall into local optimum and fast convergence[22].

We can use the differential evolution algorithm to optimize the above chess shape parameters, which is divided into the following four steps:

1. Randomly generate initial population

As shown in table 3, among all 22 parameters, OWN4 and OPP4 in the static table chessboard parameters are specified as maximum values and have no optimization significance. The own Sleep Four[23] chess shape parameter and the opponent Sleep Four chess shape parameter are set to a fixed maximum value. The remaining 18 chessboard parameters are taken as optimization targets, each individual in the initial population is an array of these 18 chessboard parameters as components.

Assuming that the initial population contains 8 individuals, the initial population is expressed as:

$$\{X_i(0) \mid x_{ij}(0) = x_{ij}^L + rand(0,1)(x_{ij}^L - x_{ij}^L), i = 1,2,\cdots,8, j = i = 1,2,\cdots,18\}$$

where $x_{ij}(0)$ represents the jth component of ith individual in the initial population, x_{ij}^U, x_{ij}^L represents the upper and lower bounds of x_{ij} respectively, $rand(0,1)$ is a random number uniformly distributed in $[0,1]$.

2. Optimizate initial population by differential evolution algorithm

Set scaling factor $F = 0.5$, crossover probability $CR = 0.4$, and number of iterations $T = 600$. The initial population is iterated for T times, and each iteration is divided into the following three steps of A), B) and C):

A) Mutation: Scale the vector difference of two different individuals of the gth generation (that is, multiply it by a scaling factor), and then combine it with the individual to be mutated to produce a (g+1)th generation variant:

$$v_i(g+1) = x_{r_1}(g) + F \cdot (x_{r_2}(g) - x_{r_3}(g)), i = 1,2,\cdots,8$$

where $r_1, r_2, r_3 \in \{1,2,\cdots,8\}$ are random numbers that are not equal to each other, $x_{r_1}, x_{r_2}, x_{r_3}$ represent different individuals of the gth generation.

B) Crossover: Crossover operation performs discrete crossover operation between the mutation individual $v_i(g+1)$ generated by mutation operation and the parent individual $x_i(g)$ to obtain candidate individuals $u_i(g+1)$:

$$u_{ij}(g+1) = \begin{cases} v_{ij}(g+1), & if\ rand(0,1) \le CR\ or\ j = j_{rand} \\ x_{ij}(g), & otherwise \end{cases}, i = 1,2,\cdots,8, j = 1,2,\cdots 18$$

where j_{rand} is the random integer in $\{1,2,\cdots,18\}$. selecting $j = j_{rand}$ is to ensure that at least one component of each mutated intermediate is passed on to the next generation, so that the candidate is not identical to the parent.

C) Selection: Candidate individuals $u_i(g+1)$ obtained in B) compete with parent individual $x_i(g)$ according to the rules of random start and exchange of first and second hands. Win 1 point, draw 0.5 point, lose 0 point, play five games, and the one with the highest score wins. For simplicity, the parent individual wins if the scores of the five games are the same.

$$x_i(g+1) = \begin{cases} u_i(g+1), & if\ u_i(g+1)\ \ win \\ x_i(g), & otherwise \end{cases}, i = 1,2,\cdots,8$$

3. Select the excellent population from the iterative population

Through the above algorithm, it is observed that the evaluation parameters are basically convergent when the evolution reaches 600 generations. Then the first individual x_1 of the 0th, 100th, 200th, 300th, 400th, 500th and 600th generations is used to compare with the individuals formed by the empirical parameters. The two sides of the game respectively adopt the first individuals selected above x_1 and empirical parameter group, according to the rule of of random start and exchange of first and second hands. 100 games are played, and the results shown in figure 6 are obtained.

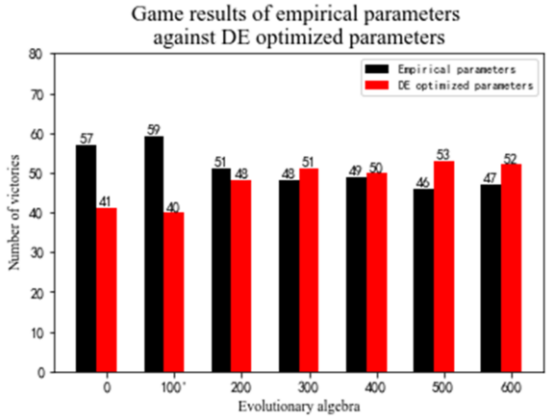

Figure 6. Comparison of the number of wins

Figure 6 shows that in the process of simulated game, with the increase of evolutionary generations, the game level of the side using optimized parameters is constantly improving, and it surpasses the opponent using empirical parameters at the 300th generation. The difference in winning rate between the two reached a peak at the 500th generation, and the winning rate of the side using the optimized parameter group was 15% higher than that of the opponent.

4. Select excellent individuals from selected population

According to the ELO scoring standard commonly used in international competitions[24], the eight individuals of the 500th generation were finally selected, and the optimized chess shape parameters are obtained.

Set the initial ELO score of 8 individuals in the 500th generation as 1500. According to the rules of random start and exchange of first and second hands, the game is played in pairs. After each game, the level score of the corresponding individual is updated according to the score update formula. Every two individuals test 10 situations and play 20 games. A total of 560 games are played. The grades of 8 individuals are as follows:

1507.8636937446697, 1561.9833298241174,
1516.4055330939761, 1470.8607582419932,
1469.6827744353977, 1492.073877951811,
1511.9882067412634, 1481.5242528311849

Because the second individual has the highest grade, the parameter values of the second individual is selected as the final result of this experiment, which are shown in table 3 below.

Table 3. Optimized chess shape parameters

Shape	Empir-ical	Optim-ized	Shape	Emp-irical	Optim-ized
OWN0	1	19.9	Sleep Two	4	96.6
OWN1	200	177.6	Live Two	4	178.9
OWN2	400	1151.3	Sleep Three	10	175.3
OWN3	2000	1240.2	Live Three	100	336.7
OWN4	20000	20000	Double Live Three	500	1264.3
OPP 0	1	58.7	Sleep Four	2000	2000
OPP 1	220	297.7	Opp's Sleep Two	4	8.9
OPP 2	420	1163.5	Opp's Live Two	4	130.8

OPP 3	2100	1200.2	Opp's Sleep Three	10	147.2
OPP 4	10000	10000	Opp's Live Three	400	686.7
			Opp's Double Live Three	2000	2390.8
			Opp's Sleep Four	2000	2000

4. Conclusion

According to the rules of Gobang, this paper puts the static table with higher priority in the front. From the experimental data, it can be seen that this method will reduce the number of nodes of the 1-6 layers of the search tree by 34% -96%, and improve the search efficiency. Then, the chessboard parameters of the basic algorithm and the static table are optimized through the optimization method of differential evolution algorithm. Finally, the playing strength of the program is further improved.

Acknowledgements

The work was supported by the National Natural Science Foundation of China (No.61771353); National Natural Science Foundation of China (No.12001408); Scientific Research Fund of Wuhan Institute of Technology (17QD53); Graduate Education and Teaching Reform Research Fund of Wuhan Institute of Technology (2020JYXM08).

References

[1] Allis L V. Searching for solutions in games and artificial intelligence[M]. Wageningen: Ponsen & Looijen, 1994.
[2] Wágner J, Virág I. Solving renju[J]. ICGA journal, 2001, 24(1): 30-35.
[3] Zheng Jianlei, Kuang Fangjun. Research and implementation of intelligent Gobang game algorithm based on minimax search and Alpha Beta pruning algorithm [J]. Journal of Wenzhou University (Natural Science Edition), 2019, 40 (03): 53-62.
[4] Sun Shiwen. Research on the realization of Gobang artificial intelligence algorithm. China New Communication, 2018, 20 (23): 143.
[5] Zhang Mingliang, Wu Jun, Li Fanchang. Design of evaluation function for Gobang machine game system. Computer Applications, 2012, 32 (07): 1969-1972 + 1990.
[6] Li Dazhou, Shen Xueyan, Gao Wei, Zhang Xiaoming, Meng Zhihui. Design and implementation of a self-learning intelligent Gobang algorithm [J]. Small Microcomputer Systems, 2020, 41 (06): 1169-1175.
[7] Wang Qin. Research on Gobang Algorithm Based on Deep Reinforcement Learning [D]. Chongqing University, 2019.
[8] Wurui. Research and Implementation of Game Decision Mechanism Based on Deep Reinforcement Learning [D]. Beijing University of Posts and Telecommunications, 2020.
[9] Song Wanyang. Design and Implementation of Android Gobang Program Based on α-β Pruning Tree Algorithm [J]. Modern Information Technology, 2019, 3 (11): 92-93 + 97.
[10] Cao Fengyun, Zhao Weihua. Research on Gobang Game Platform Based on Java [J]. Journal of Chongqing Technology and Business University (Natural Science Edition), 2021, 38 (02): 10-15.

[11] Shannon C E. XXII. Programming a computer for playing chess[J]. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1950, 41(314): 256-275.
[12] Niu J, Tang W, Xu F, et al. Global research on artificial intelligence from 1990–2014: Spatially-explicit bibliometric analysis[J]. ISPRS International Journal of Geo-Information, 2016, 5(5): 66.
[13] Zhang Xiaojian. Research and Design of Gobang Computer Game System. Anhui University, 2017.
[14] Liu Rui. Design and Implementation of Artificial Intelligence Algorithm for Gobang [D]. South China University of Technology, 2012.
[15] Li Hao. Research and Implementation on Optimization of Gobang Man-machine Game Algorithm [D]. Dalian Maritime University, 2020.
[16] Takeuchi S, Kaneko T, Yamaguchi K. Evaluation of game tree search methods by game records[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2010, 2(4): 288-302.
[17] Yue Jinpeng, Feng Su. Research and Improvement of Alpha-Beta Search Algorithm for Chinese Chess [J]. Journal of Beijing Normal University (Natural Science Edition), 2009,45 (02): 156-160.
[18] Knuth D E, Moore R W. An analysis of alpha-beta pruning[J]. Artificial intelligence, 1975, 6(4): 293-326.
[19] Chen Xuerong, Yue Shudan. Design and Implementation of Gobang Game Based on Java. Information Systems Engineering, 2020 (08): 104-105.
[20] Liu Bo, Wang Ling, Jin Yihui. Research progress of differential evolution algorithm. Control and Decision, 2007 (07): 721-729.
[21] Xu Bin. Research and application of multi-objective optimization method based on differential evolution algorithm. East China University of Science and Technology, 2013.
[22] Yang Qiwen, Cai Liang, Xue Yuncan. A survey of differential evolution algorithms. Pattern Recognition and Artificial Intelligence, 2008, 21 (04): 506-513.
[23] Chen Liang. Improved adaptive differential evolution algorithm and its application [D]. Donghua University, 2012.
[24] Cao X, Lin Y. UCT-ADP Progressive Bias Algorithm for Solving Gomoku[C]//2019 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2019: 50-56.