Key Technologies Analysis of Three Sides Chinese Chess Computer Game

Zedan Chen, Miao Liu*, Zhiqiang Li, Xiaofeng Lian
School of Computer Science and Educational Software,
Guangzhou University, Guangzhou, Guangdong, 510006, China
liumiao@gzhu.edu.cn

Abstract

The three sides Chinese chess is a novel game which is developed from the traditional Chinese chess. There are not any research results on the three sides Chinese chess computer game. In this paper, the three sides Chinese game is firstly compared with the traditional Chinese chess. Then an algorithm of three sides game searching with pruning is put forward, which is based on a vector evaluation function. In the end, a machine learning algorithm is addressed in detail, which learns from history game processes. Some experiment results also demonstrate the pruning and learning effect.

1. Introduction

Researchers start to study the traditional Chinese chess computer game in the 1970s. From then on, the traditional Chinese chess software emerges in large numbers. So far, there is not any the three sides Chinese chess software. The three sides Chinese chess game is developed from the traditional Chinese chess, which involves just two players. In the three sides Chinese chess game, the possible state space faces combinatorial explosion problem. When each side moves 50 steps, there are possible 55 choices for one step, and then the number of nodes in the game tree of the three sides Chinese chess game is $10^{111}$ times of that in the traditional Chinese chess. In the three sides Chinese chess game, there is not only opponent relationship but also cooperative relationship among three players. So, the pruning algorithm of the traditional Chinese chess game does not suit for the three sides Chinese chess game. References 1 and 2 put forward the concept of Max’n tree, but neither of them deals with evaluation of situations, and so, the searching algorithms in the papers have something to improve.

The rest of the paper is organized as follows: Section 2 introduces the three sides Chinese chess. Section 3 puts forward an algorithm of three sides game searching, and a vector based pruning method. Section 4 gives an algorithm of machine learning, and some experiment results. Section 5 concludes the paper.

2. The three sides Chinese chess

The board of the three sides Chinese chess consists of three rectangles and one hexagon. There are 171 cross points in the board, and the pieces locate in these cross points. Figure 1 illustrates all pieces in their starting positions.

![Figure 1 All pieces in their starting positions](image)

There is a little difference on moves of pieces between the three sides Chinese chess and the traditional Chinese chess. When rooks or canons arrive at or leave from the hexagon, they should move to the edge cross points of the hexagon first. During gaming, if A side captures the king of B side, then A side possesses of all remaining pieces of B side. When one side captures the kings of the other two sides, then the side wins.

3. The three sides game searching algorithm

3.1. The Data structure

If adopting the adjacency matrix, there will be too many elements up to 29241, which is got by 171 multiply by 171. The number of pieces is less than 48, and so it will waste much memory space. If adopting linked list, there will add much more complexity of operations despite of compressed memory space. So, the paper adopts a three dimensions array, and the size of the array...
is $14 \times 14 \times 14 = 2744$. From the view of red side, the three dimensions pare: the first is from the bottom to the up, the second is from the left corner to the down right corner, and the third is from the top right to the down left corner. Every dimension includes 14 broken lines. The elements which represent cross points in the chess board are assigned some natural numbers, and those elements which represent cross points out of the chess board are assigned some negative number.

### 3.2. The vector based evaluation function

During gaming among three sides, how to define the evaluation function is really a key problem, and the function should reflect the relationship among three sides. The opponent or cooperative relationship changes dynamically according to every situation. The paper presents a vector based evaluation function which will discussed soon.

As illustrated in Figure 2, there is an equilateral triangle, in which three vertex stand for the winning state of three sides respectively. The coordinate values of point $P'$ stand for the current situation.

![Figure 2 The form of the evaluation function](image)

In the figure 2, the point $P$ is the center point of the triangle. In the opening situation, the point $P'$ and $P$ locate in the same place. When the $P'$ moves to the location of $A$ (B or C), then $A$ (B or C) side wins. So, the shorter distance between $P'$ and some vertex, the current situation is more favorable for the side which the vertex represents.

Suppose in one situation, $a$, $b$, $c$ are scores for $A$, $B$ and $C$ respectively, then coordinate values of point $P'$ is determined by the following formula.

$$ OP' = OP' + a(BA+CA) + b(AB+CB) + c(AC+BC) $$

The score of one side is calculated according to the following formula.

One side score = all pieces basic strength values in the side + all pieces flexibility values in the side + all pieces protection values in the side - all pieces threaten values in the side.

The flexibility value is determined by the following formula.

Flexibility = the move value * the possible move number.

The threaten values and protection values are calculated according to the flow chart in the figure 3.

![Figure 3 The flow chart of the calculating threaten values and protection values](image)

In the figure 3, $Pos$ is the location of pieces, $Ascore$ is the strength of attack, $nH$ equals basic strength value divided by 16, and ret represents the return value. When ret is 18888, it means the side is captured.

### 3.3. The searching algorithm with pruning

In the three sides Chinese chess game, the searching space is very huge. Each side takes turns to move his pieces, and a game tree is constructed accordingly. The root of the game tree is the opening situation. The child nodes are situations created according to A sides various possible moves, the nodes in the next layer are situations created according to B sides various possible moves for every parent node, and for every B sides situation, there create many child nodes according to C side possible moves. The process will continue to repeat until every leave node represents some end game. The three sides Chinese chess may have 40 to 80 possible moves for one step. Take 50 possible moves for one step on average, the three sides have 16 steps each, and then the game tree has about $10^{160}$ nodes. If there is only $10^{-8}$ seconds for creating one node, then there will take about $10^{140}$ years to create the game tree. So, it is very important to find an effective pruning method.

Suppose in one situation, the gaming tree is as depicted in the figure 4, $M$, $M'$ and $M''$ represent the calculated possible situations of $A$, $B$, and $C$ respectively. $M''_0$ and $M''_1$ represent the possible two situations which $C$ side moves next.
The above game tree is transformed to the vector based evaluation function in the figure 5.

When the calculated situation point moves to some vertex, then the side which the vertex represents wins. So, the strategy of every side is choosing the situation point which is the closest to its vertex. When A side gets a possible situation M, the searching scope becomes a circular orbit which center is A, and radius is AM. The circular orbit is called A fan. In the same way such as illustrated in the figure 5, we get B fan and C fan.

In the figure 5, if C side gets the best situation M" equals M"_0 in level 3, then B fan does not intersect with C fan. So, B will not choose the parent situation of M" in level 2, and the parent situation is pruned. If C side gets the best situation M" equals M"_1 in level 3, then B fan intersects with C fan. B fan’s radius becomes BM"_1 now. But, A fan does not intersect with the B fan. So A will not choose the parent situation M' in level 2, B will not chose the parent situation of M" in level 3, and the parent situation is pruned. A side will search other situations right now.

Some main code snippet of the searching algorithm with pruning is listed in the following.

```c
void Search(depth, isTurn, Pos0, Pos1, Pos2, retPos, RetLayer) {
    //initialization
    BestPos.Ar = BestPos.Br = BestPos.Cr = 100000;
    i = IsGameOver();
    if (i!= NoDead) {
        switch(i) {
            //When two of three sides are captured, there is no need to search, just return the situation evaluation values.
            case OverRedAndBlack: case OverBlackAndYellow: case OverYellowAndRed:
                return Evaluate(); break;
            //When some one side is captured, then the side’s window is initialized, and skips the side during searching.
            case IsRed: case IsBlack: case IsYellow:
                if(i == isTurn) {
                    Search(depth, (isTurn+1)%PlayNum, Pos1, Pos2, BestPos, next_return_Pos, RetLayer);
                    retPos = next_return_Pos; //return to the upper layer
                    if(RetLayer > 0) RetLayer--;
                    return;
                }
            //arrives at the predefined depth
            if (m_nMaxDepth - depth <= 0) return Evaluate();
            CreatePossibleMove;
            for(i=0; i<MoveCount; i++) {
                MakeMove(i); //execute the i’th move
                Search(depth + 1, (isTurn+1)%PlayNum, Pos1, Pos2, BestPos, next_return_Pos, RetLayer);
                if(RetLayer > 0) break;
                if(isTurn%PlayNum == IsRedTurn) //Red side
                    if(next_return_Pos.A_r < BestPos.A_r) {
                        BestPos = next_return_Pos;
                        if(BestPos.A_r + Pos2.C_r <= triangle_side) {
                            if(BestPos.C_r < Pos2.C_r)
                                if( BestPos.C_r + Pos1.B_r < triangle_side )
                                    if( BestPos.A_r + Pos1.B_r < triangle_side )
                                        if(RetLayer = 1) break; //Pruning 1
                                else
                                    if(RetLayer > 0) RetLayer--;
                            else
                                if(RetLayer > 0) RetLayer--;
                        }
                    }
            }
            retPos = BestPos;
            if(RetLayer > 0) RetLayer--;
        }
    }
}
```

The Pos is the evaluation structure, which is composed of the coordinate values and distances among the situation point and three vertexes. The three distance data are initialized to assign huge numbers. The retPos is the
return evaluation value for the next layer. The retLayer is the predefined pruning depth. The triangle side is the side length of the equilateral triangle. The PlayNum is the number of players. The function IsGameOver() judges the state of the game. There are three states, one is no one side is captured, one is one side is captured, and the other one is two sides are captured. The function Evaluate() calculates and returns the current situation evaluation value.

We experiment the searching algorithm with the opening situation twice, one without pruning, and the other with pruning. The results are listed in table 1.

<table>
<thead>
<tr>
<th>Level</th>
<th>No</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>59</td>
<td>1638</td>
<td>84017</td>
<td>120435765</td>
</tr>
<tr>
<td>2</td>
<td>59</td>
<td>741</td>
<td>7385</td>
<td>50733480</td>
</tr>
</tbody>
</table>

4. A machine learning algorithm

Many research [3-7] emphasis on Chinese chess game machine learning is adjusting parameters of evaluation functions. In different situations, the evaluation values of the same piece are not the same. The paper puts forward a machine learning algorithm which learns from history game processes.

When one game is over, the machine evaluates every move during the previous game process. For every move, the machine calculates evaluation value, and the value equals sum of all situations after the move divided by the total number of steps after the move. When get evaluation values of every move, the machine compares them with previous data stored in the database. If there is the same move for the same situation, then the smaller evaluation value overlaps the bigger one. If there are different moves for the same situation, then the move which has higher evaluation value replaces the move which has lower evaluation value, and so are the evaluation values. If there is not the same situation in the database, then store the move, the situation and the evaluation value into the database.

We make experiments for the machine learning algorithm with five games. The machine records one people player moves in the figure 6 for each game. The vertical axis represents the distance of situation point from the vertex which represents the people player. The horizontal axis represents the moves, and here just show 10 moves for the space limit. The curves f1, f2, f3, f4 and f5 represent five games respectively.

As a result of machine learning from the above five games, the moves and the corresponding evaluation values are shown in the figure 8.
The dotted segments represent the moves not used by the machine. According to the figure 8, for the fourth move of the game 1 and game 2, the ninth move of game 3 and game 5, the corresponding situation is the same, but the next move is different. The machine chooses the better move and stores the worse evaluation values.

5. Conclusions

In this paper, a three sides game search with pruning algorithm is discussed, which is based on the three sides Chinese chess game. The idea may be extended to deal with four or more players game. Take four players game for example, a three dimension searching space may set up, and pruning may be according to spheres collision. In order to test the algorithms, a three sides Chinese chess computer software is implemented, the software has already won some ordinary people players, and there is a lot of work to do on improving the software intelligence greatly.

Acknowledgement

It is a project supported by the National Natural Science Foundation of China under Grant No.60373156, 10871222, 10726012.

References