Optimization of the Connect6 Classical Evaluation Function Based on Threat Theory and Game Strategy

Ruofan Wu
Engineering Training Center, University of Science & Technology Beijing
Beijing, China
e-mail: demonsimon@gmail.com

Ke Zhou
Engineering Training Center, University of Science & Technology Beijing
Beijing, China
e-mail: cocofay126@126.com

Abstract: **Connect6 is simple in rules but quite complex while playing. Owing to the great game-tree complexity, the game situation can be changed easily after even one stone put on the board, the players have to change strategies according to different board situations. To make the program more clever, different strategies must be taken into concern. This paper covers about Connect6 and its rules, the design of classic evalution function and also brings out a opimization method to the evalution function based on threat theory and game strategy.**

Keywords: Connect6; Evaluation Function; Game Strategy; Threat Theory; Computer Game

I. INTRODUCTION

Connect(m,n,k,p,q) denotes a family games of k-in-a-row[1][2]. Connect6 can be described as Connect(19,19,6,2,1), two players, named Black and White, alternately place two stones of their color respectively on a 19*19 board, except the Black puts only one stone the first step of the game. The one who gets six continuous stones in a line first wins the game, the line can be horizontal, vertical and diagonal. Unlike Chess, during the k-in-a-row game playing, no stone capture is permitted.

In game theory, a strategy[3] refers to one of the options that a player can choose. That is, every player in a non-cooperative game has a set of possible strategies, and must choose one of the choices. Similar to football games, the strategy of Connect6 can be attack, defense or something like counter-attack. To a certain extent, VCF is considered as a kind of application of attack strategy.

II. THREAT THEORY AND GAME STRATEGY

Before looking into what is Threat theory, some terms [1][2][4]of Connect6 should be known.

Conn-6, six stones in the same color connect continuously.

Live-5, just need to put one stone to form a Conn-6, but the opponent need to put two stones to prevent it forming a Conn-6.

Sleep-5, similar to Live-5, but only one stone can prevent from Conn-6.

Live-4, need to put two stones to form a Conn-6, also two stones is needed to defend.

Sleep-4, similar to Live-4, but only one stone is enough to prevent it.

Live-3, another one stone can make it a Live-4.

Sleep-3, even two stones can only make it a Sleep-5.

Live-2, another two stones can make it a Live-4.

Sleep-2, even two stones can only make it a Sleep-4.

As I-Chen Wu defined, for Connect6, assume that one player, say W, cannot connect six. B is said to have t threats, if and only if W needs to place t stones to prevent B from winning in B's next move. So, we can find out that:

A Live-4 or Live-5 owns TWO threats, while a Sleep-5 or Sleep-4 get ONE.

A Live-3 or Live-2 can own TWO threats in the NEXT turn, while a Sleep-3 or Sleep-2 can own only ONE. But a Live-3 may gain more threats in the Next turn than a Live-2.

And we need at least THREE threats to confirm to win.

Assume that it's our turn to make the move, the attack strategy of Connect6 can be interpreted as following:

1. If any of our Live or Sleep 4/5 do exists, do with it and make it a Conn-6.
2. If no Live or Sleep 4/5 exists for both sides, make our Live-2/3 to Live-4/5, the more the better. The so-called VCF algorithm is just to make Live-4s continuously, to make the opponent to use two stones to defend. Always in the process of making continuous Live-4, additional crosswise fours can be made, which always leads to win.
3. If Live-4/5 cannot be made, make our Sleep-2/3 to Sleep-4/5, the more the better. Always, Sleep-4 is more efficient.
4. If no Live-2/3 exists, make it, but makeing a Live-2 is usually more efficient and useful than Live-3. And crosswise Live-2s is best choice.

Also, the defense strategy can be interpreted as:

1. If any of opponent's Live or Sleep 4/5 exists, stop it and make it dead.
2. If any of opponent's Live-2/3 exists, make it sleep, the more sleeper the better.
3. If more than one Sleep-3 are existed, make them dead.

Supposing that neither has a Live 4/5 and we get less Live-2/3s, Synthesize the above two strategies, a

978-0-7695-4304-8/10 $26.00 © 2010 IEEE
DOI 10.1109/GCIS.2010.192

compromise is occurred, I call it counter-attack strategy, which name comes from a football tactic that used in a adverse situation during a game:

1. If any of our Sleep-4/5 exists, make it a Conn-6.
2. If any of opponent's Sleep-4/5 exists, make it dead, the more sleeper the better. If the opponent only has one Sleep-4/5, use one stone to make it dead, then there can be several choices for the other stone according to different board situations, to make our Live or Sleep 2/3 to 3/4, or to make opponent's Live-2/3 to sleep are the optional.
3. If any of opponent's Live-2/3 exists, make it sleep, the more sleeper the better. If the opponent only has one Live-2/3, use one stone to make it sleep, and use the other to make our Live or Sleep 2/3 to 3/4, or to make new Live-2s of ourselves own. Usually, making Live or Sleep 3s to 4s is more threatening now, but making new Live-2s may lead to a great advantage in the next three or four steps.
4. Make our Live-2/3 to Live-4/5, or make our Sleep-3s to Sleep-4s.the more the better. Always, 4s is better.
5. Make Live-2s, the more the better.

III. Classical Evaluation Function Design

Nowadays, how to construct a appropriate evalution function is still a major issue on computer games like Go, Chess, etc. Since Connect6 is quite similar to Renju, which the Connect6 evalution function draws lessons from. The classic evalution function is usually related to several aspects, such as the static value of the board, the number and the evalution of different stone types, etc. Consider simply, the evalution of the entire board can be summarized as a formula[5]:

$$E_s = E_0 + \sum_{i=1}^{n} k_i * E_i .$$

E_s stands for the sum of the evalution, E_0 means the static value of the board which different from the stone position on the current board, E_i means different value for different type of stone vector, k_i means different weight coefficient.

The difficulty lies at the point how to identify different stone vectors correctly, and how to assign appropriate value to different stone type.

The algorithm to identify different stone vectors[4] in one line is as follows:

```
if (RightEdge - LeftEdge == 4)
{     //Five stones connect consecutively
      if (RightEdge < GRIDNUM && Line[RightEdge+1]
== NOSTONE)
      {
            if (LeftEdge && Line[LeftEdge-1] ==
NOSTONE)
                  LineRecord[AnalysePosition] =
FIVE;       //Live-5
            else
                  LineRecord[AnalysePosition] =
SFIVE;      //Sleep-5
      }
      else if (LeftEdge && Line[LeftEdge-1] ==
NOSTONE)
            LineRecord[AnalysePosition] = SFIVE;
            //Sleep-5

      return LineRecord[AnalysePosition];
      //return the stone types in this line at the
analysing position
}
```

In the code sections above, array Line store the very line to be analysed, varible AnalysePosition is the analyse starting position, array LineRecord store the result of analyse. Since the board is consist of many lines in different directions, the code will be called for many times to analyse the whole board.

A appropriate assignment of different values to different stone types is always closely related to the programmer's knowledge of Connect6 game and the adjustment to the evalution parameters need lots of game expriments which will not be mainly discussed here.

IV. Optimization to the Evaluation Function

Conventionally, the usual practice is to change the weight coefficient ki while different game strategies is taken, which is far from enough. If attack strategy is taken, the weight of Live-2/3/4 may get a higher value, while Sleep-2/3 may be in low. So, different strategy is in use, the weight of the same stone types may differ a lot.

But in this method, strategies take little effect on the evalution function. Another method should be taken to expand the influence.

We can notice that in the above code sections, there is an array called LineRecord which only stores the stone type at the analysing position. We can do something with this array and make it more useful.

While we are analysing the stone vectors in a line, we can not only get the stone types, but also the candidate positions for the next moves according to different strategies.

For example, the third code sections can be modified as following:

```
if (RightEdge - LeftEdge == 4)
{
      if (RightEdge < GRIDNUM && Line[RightEdge+1]
== NOSTONE)
      {
            if (LeftEdge && Line[LeftEdge-1] ==
NOSTONE)
            {
                  LineRecord[AnalysePosition] =
FIVE;       //Live-5
                  LineRecord[LeftEdge-1] =
IMPORTANT;
            }
            else
                  LineRecord[AnalysePosition] =
SFIVE;      //Sleep-5
            LineRecord[RightEdge+1] = IMPORTANT;
      }
      else if (LeftEdge && Line[LeftEdge-1] ==
NOSTONE)
      {
            LineRecord[AnalysePosition] = SFIVE;
            //Sleep-5
            LineRecord[LeftEdge-1] = IMPORTANT;
      }

      return LineRecord[AnalysePosition];
      //return the stone types in this line at the
analysing position
}
```

The code section above is just an instance, just to show how to make use of the LineRecord array. In the process of scanning and analysing the board, we can get the candidate positions which can form threats in the next move passingly.

According to the threat theory and basic Connect6 game knowledges, for different types of stone vectors, the importance of vacant positions in a line differs in threats it may form in the next turn. So, mark the positions in the line which can lead to Conn6 or Live-4/5(threats) as IMPORTANT. Then, filter the less importance positions out, and only add the important positions as the candidate positions of the next move. This will lessen the number of the branches of the game-tree, decrease the width of the game-tree, it also will improve the speed of seaching and make the program faster.

Further, the game strategy should be taken into concerned, too. As mentioned above, we mark the positions which can lead to win or more threats as tag IMPORTANT, actually, corresponding to different kinds of game strategies, the tags can be classified as ATTACK, DEFENSE and COUNTER. Mark the vacant positions in the line with different tags, sort them out, and store them in different stacks or arrays. While different game strategy is triggered off, different candidate positions of the next move are in use. This will shorten the time, and also make the evaluation function well targeted. But the difficulty lies in the condition of using different strategies.

V. Experiments and Results

To examine whether this method can take a positive effect on the evaluation function, contrast experiment need to be done.

The experiments are taken between two groups. Group 1 uses the normal classic evaluation function, group 2 uses the evaluation function optimized. The other parts of the two programs are the same, using the traditional NegaMax Pruning Algorithm, without using VCF algorithm and any opening book of Connect6 game. The maximum searching depth is three. The opponents of the two groups are the same, an individual program which wins the first place in the inner competetion of University of Science and Technology Beijing. Each group plays to the opponents for 20 times, half in the color of black, while the other half in white. Record the average time costed for a single step and the outcomes of the game. The results are recorded in Figure 1 and Figure 2.

From the Figure 1, group 2 won all of the 20 games, while group 1 only won 7. It illustrates that, the evaluation function combined with both threat theory and game strategy is more likely to win the game. Compared to the formal evaluation function, the stone positions where the new program chooses to place are more reasonable, also are more offensive in attack, more defensive in defense.

Also, in Figure 2, by comparing the average time of a single move, it is obvious that group 1 takes the much longer time, which is almost three times more than group 2. It indicates that the evaluation function combined with game strategy and threat theory is more efficient in game tree seaching, the method mentioned can ignore more less important positions, and make the program only focuses on the candidate positions which is proper to specific strategy the pragram is right now taking.

VI. Conclusion

This paper simply coverd the Connect6 game's rules and threat theory introduced by I-Chen Wu, summarized several game strategies in Connect6 game, then proposed a method to optimize the evaluation function to improve the speed of the seaching. The method emphasizes the importance of different strategies by ignoring the less important positions and only focusing on the specific candidate positions corresponding to specific strategies.

The results of the experiments showed that it outperform the other versions of program which are not combined with the game strategy, not only at the time spending on a single step, but also the stability of winning games.

References

[1] Wu I-C, Huang D-Y, A New Family of k-in-a-row Games, in Proceedings of The 11th Advances in Computer Games Conference, 88-100, 2005.

[2] Wu I-C, Huang D-Y, Chang H-C, Connect6, ICGA Journal, 28, 4, 234-242, 2005.

[3] Webpage of Strategy, http://en.wikipedia.org/wiki/Strategy_(game_theory)

[4] Wang Xiaochun. PC Game Programming(Games between Human and Machine) [M]. Chongqing: Chongqing University Press, 2002.

[5] Francis Dominic Laram. Chess Programming Part I: Getting Started. http://www.gamedev.net/reference/articles/article1014.asp.

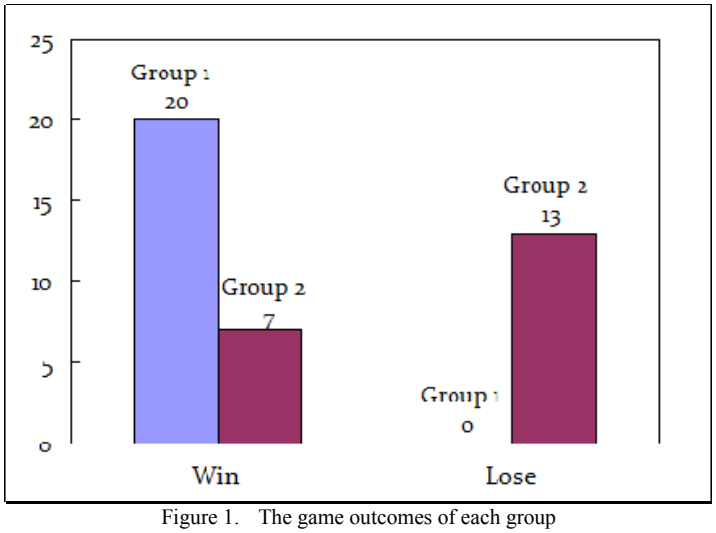

Figure 1. The game outcomes of each group

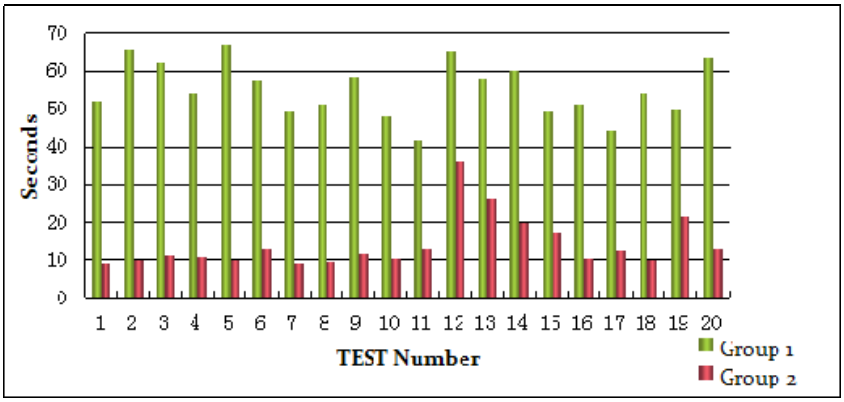

Figure 2. The average time for a single step