

Gomoku

Ver 3.0(Final)

09016401

顾婷瑄

Botzone id: Gutingxuan

1. Old versions review

- i. Gomoku1.0 is based on a static evaluation function which simply evaluates the value of each available position. It is easy to understand and realize and it seems to be effective at first. But with the time going, its ability is not strong enough to beat my classmates' programs, so I decided to evolve it with a more complicated algorithm next time.
- ii. Gomoku2.0 is simply based on a MaxMin algorithm and although it can foresee future moves, it spends a lot of time on doing calculations so the deepest searching level it can reach is a mere three. So I was thinking about making some improvements to cut down the searching time to enhance the depth of searching.
- iii. Gomoku2.1 is based on a MaxMin algorithm together with the alpha-beta pruning. It can make tentative moves and foresee future moves of both sides in a shorter time and the depth of the MaxMin algorithm was enhanced due to the help of alpha-beta pruning. This one is basically my final version of Gomoku.

Last but not least, I made some changes to my evaluation functions every time when working on my new version of Gomoku. I guess this helps too.

2. Literatures I read

[1] 王志水. 基于搜索算法的人工智能在五子棋博弈中的应用研究[D].山东：中国石油大学, 2006

[2] Blog: 五子棋基本模型@我是老邱

五子棋估值算法@maxuewei2

3. Current idea of design

i. Essence

A function based on the MaxMin algorithm and the alpha-beta pruning.

ii. Origin of the idea

The paper I've read.

iii. Expectation

The function can make tentative moves, evaluate the value of each state on both sides and then make the best choice. Furthermore, the time it

spends to do the calculations will be decreased.

iv. Implementation

A MaxMin algorithm, alpha-beta pruning, new evaluation method.

v. Advantages of the design

With the alpha-beta pruning, the time spent on calculating can be decreased. In the last version, I sorted the values of the available positions before searching and only visited the first 12 positions and the depth it could go was 6. But in this version, I tried to visit the first 45 possible positions and go with the depth of 4 and the result turns out to be better than the last version on Botzone. However if there is not a time limit, the depth of 6 will definitely beat the 4 by visiting same amount of positions. And I fixed a possible bug of my evaluation function which evaluates the value of the board, I guess this helps a lot.

vi. Weakness

The alpha-beta pruning was not strong enough so the number of positions the program can visit in a pleasant time is still restricted. And the evaluation of the board value was not that accurate compared with the actual value of the board.

vii.Improvement directions

I will try to optimize the evaluation method and try to improve my alpha-beta pruning. But I guess this will only happen if I got some spare time since I' m pretty satisfied with my current version of Gomoku.

4. Evaluation of the performance


Former ranking on Botzone: 20/37

(2018/5/3 20:00 Ver2.1)

18	刘小江Max		刘小江Max	1248.07	刘小江Max2.0	0	 .cpp17  人机  ID
19	hhh		Dengyiyong	1220.84	1	9	 .cpp17  人机  ID
20	test1		Gutingxuan	1184.44	6 12	8	 .cpp17  人机  ID
21	pro		Dengxiayang	1174.22	尝试	28	 .cpp17  人机  ID
22	test		sigvol	1153.20	1.0	14	 .cpp17  人机  ID

Current ranking on Botzone: 15/54

(2018/6/1 15:00 Ver3.0)

13	OldSchool	 Vigilans	1522.69	跨越99%BUG的障壁，抵达最终的...	17	 .cpp17  人机
						 ID 
14	wzq	 syiml	1480.00	aaa	17	 .cpp  人机 
						
15	test2	 Gutingxuan	1434.96	?	26	 .cpp17   人机
						 ID 
16	凭感觉下	 YYT	1403.64	缘分到了自然会防守	19	 .cpp17  人机
						 ID 
17	yayaho	 Jahoo	1274.70	ya~~~	13	 .cpp  人机 
						

5. Code structure and detailed implementations

Since the MaxMin algorithm and alpha-beta pruning were explained in former reports and they were not changed in this version, this time I will only introduce my enhanced evaluation function.

To evaluate the value of the board, I only collect the alive-threes, dead-fours, alive-fours and five-in-a-rows because these can best represent the value of the current board and these are the moves that have to be dealt with, otherwise you will definitely lose the game. This kind of simplification can save a lot of time spent on the searching of board types.

But there's a problem with this evaluation function which is that the same set value may be calculated for a few times since there might be more than one position can detect this set. But since there are huge gaps between different levels of sets, this error won't make a big mistake.

```
1. int Evaluate3(int x, int y, int choice)
2. {
3.     int temp=0;
4.     int count=0;
5.     for (int i = 1; i <= 8; i++)
6.     {
7.         //连五
8.         if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice &
            & getLine(x, y, i, 3) == choice
```

```

9.          && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == choic
e)
10.      {
11.          temp += 5500000;
12.      }
13.
14.      //活四 01111*
15.      if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice &
& getLine(x, y, i, 3) == choice
16.          && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == -1)
17.          temp += 500000;
18.
19.      //冲四 A 21111*
20.      if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == choice &
& getLine(x, y, i, 3) == choice
21.          && getLine(x, y, i, 4) == choice && (getLine(x, y, i, 5) == 1-ch
oice||getLine(x,y,i,5)==-2))
22.          temp += 450000;
23.
24.      //冲四 B 1*111
25.      if (getLine(x, y, i, -1) == choice && getLine(x, y, i, 1) == choice
&& getLine(x, y, i, 2) == choice
26.          && getLine(x, y, i, 3) == choice)
27.          temp += 400000;
28.
29.      //冲四 C 11*11
30.      if (getLine(x, y, i, -2) == choice && getLine(x, y, i, -1) == choice
&& getLine(x, y, i, 1) == choice
31.          && getLine(x, y, i, 2) == choice)
32.          temp += 250000;
33.
34.      //活三 A-1 0111*0
35.      if (getLine(x, y, i, 1) == -1 && getLine(x, y, i, -1) == cho
ice && getLine(x, y, i, -2) == choice
36.          && getLine(x, y, i, -3) == choice && getLine(x, y, i, -4) == -1)
37.      {
38.          temp += 35000;
39.          count++;
40.      }
41.
42.      //活三 B-1 01*110
43.      if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == -1 &&
getLine(x, y, i, 1) == choice

```

```

44.         && getLine(x, y, i, 2) == choice && getLine(x, y, i, 3) == -1)
45.     {
46.         temp += 35000;
47.         count++;
48.     }
49.
50.     //活三 B-2  *10110
51.     if (getLine(x, y, i, 1) == choice && getLine(x, y, i, 2) == -1 && ge
        tLine(x, y, i, 3) == choice
52.         && getLine(x, y, i, 4) == choice && getLine(x, y, i, 5) == -1)
53.     {
54.         temp += 35000;
55.         count++;
56.     }
57.
58.     //活三 B-3  01011*
59.     if (getLine(x, y, i, -1) == choice && getLine(x, y, i, -2) == choice
        && getLine(x, y, i, -3) == -1
60.         && getLine(x, y, i, -4) == choice && getLine(x, y, i, -5) == -1)
61.     {
62.         temp += 35000;
63.         count++;
64.     }
65. }
66.
67. if(count>=2) temp+=400000;
68. return temp;
69. }

```