

# Minimax 算法在中国象棋的应用

武自厚 20336014 保密管理

2022 年 4 月 20 日

## 1 原理分析

### 1.1 Minimax 算法

Minimax 算法, 是一种应对博弈游戏 (尤其是零和博弈) 时考虑步骤所需要的博弈树搜索算法. 其基本原理为: 每一个叶子结点都具有一个值, 而奇数层 Max(代表一名玩家的决策) 结点的值是其子结点的最大值; 偶数层 Mini(代表另一名玩家的决策) 结点的值是其子结点的最小值.

算法步骤可以用自然语言归纳为如下几步.

1. 获取当前状态的行动列表.
2. 遍历行动列表, 并获取下一步的权值 (递归执行).
3. 选取最大 (小) 的权值返回.

如此得出当前状态下一步状态的全部权值, 选择其中具有最大权值的行动执行即可.

### 1.2 $\alpha - \beta$ 剪枝

Minimax 算法虽然可行, 但是它过大的时空复杂度限制了算法解决问题的规模. 对此, 一个有效的优化就是引入  $\alpha - \beta$  剪枝.

此时, 一个结点拥有的权值不再是一个, 而是两个, 它们称为  $\alpha$  和  $\beta$ , 代表了这个结点的递归算法中出现过的最小值和最大值. 如果某一时刻一个结点出现了  $\alpha \geq \beta$  的情况, 这就说明它最后采纳的子结点已经被遍历, 不再需要考虑之后的子结点 (即 “剪枝”). 容易知道, Max 结点的  $\alpha$  值就是其权值, 而 Mini 结点的  $\beta$  值就是其权值.

### 1.3 深度受限与评估函数

对于中国象棋而言, 一味向下搜索企图寻找到终止状态 (即一方获胜) 所造成的时空代价会非常高, 所以对于搜索的深度进行限制很有必要. 但是到达最大深度时并不能直接获取状态的权值, 这时就需要评估函数来对最大深度结点进行启发式评估以获取权值.

目前已经提出了一些根据棋子位置、棋子个数、棋子灵活性和特殊局面进行棋局评估的函数<sup>[1]</sup>, 对此本文不再赘述.

### 1.4 历史表启发 (创新点)

$\alpha - \beta$  剪枝在一定程度上可以降低计算次数, 但是其效果取决于行动的排序, 如果被采纳的行动是最后被遍历的, 那么  $\alpha - \beta$  剪枝将不起作用. Jonathan 在 1989 年提出一种称为“历史表启发”的优化方式会启发式地调整算法遍历顺序<sup>[2]</sup>, 让  $\alpha - \beta$  剪枝发挥最大的作用.

历史表记录了一个行动最优性的权值, 它的操作分为访问和更新两种. 访问操作用于获取行动的最优性权值, 而更新操作用于增加权值. 在获取行动列表时, 算法将会根据各个行动在历史表中的权值进行排序. 在返回时, 算法将会对造成最大 (小) 值的行动进行历史表更新.

## 2 伪代码实现

### 2.1 $\alpha - \beta$ 剪枝的 Minimax 算法实现

---

**Algorithm 1:** alphabeta( $s, \alpha, \beta, \text{depth}$ )

---

输入: 当前状态  $s$ , 继承权值  $\alpha, \beta$

输出: 最大  $\alpha$  值

```

1 if depth ≤ 0 then
2   return evaluate( $s$ )
3  $S := s$  的后继状态
4 for  $s' \in S$  do
5    $v := \text{alphabeta}(s', -\beta, -\alpha, \text{depth} - 1)$ 
6   if  $v \geq \alpha$  then
7      $\alpha := v$ 
8   if  $\alpha \geq \beta$  then
9     break
10 return  $\alpha$ 

```

---

## 2.2 具有历史表启发以及 $\alpha - \beta$ 剪枝的 Minimax 算法实现

---

**Algorithm 2:** alphabeta( $s, \alpha, \beta, \text{depth}$ )

---

```

    输入: 当前状态  $s$ , 继承权值  $\alpha, \beta$ 
    输出: 最大  $\alpha$  值
    全局: 历史启发表  $\text{history}$ 
    1 if  $\text{depth} \leq 0$  then
    2   | return evaluate( $s$ )
    3  $S := s$  的后继状态
    4  $S.\text{sort}()$  ; // 按照  $s \mapsto \text{history}[s]$  排序
    5 for  $s' \in S$  do
    6   |  $v := \text{alphabeta}(s', -\beta, -\alpha, \text{depth} - 1)$ 
    7   | if  $v \geq \alpha$  then
    8   |   |  $\alpha := v$ 
    9   |   |  $s^* := s'$  ; // 记录最优后继状态
    10  |   | if  $\alpha \geq \beta$  then
    11  |   |   | break
    12 history[ $s^*$ ] := history[ $s^*$ ] +  $2^{\text{depth}}$  ; // 更新历史表记录
    13 return  $\alpha$ 

```

---

## 3 关键代码

### 3.1 棋局评估函数

```

1 class Evaluate(object):
2     # 棋子棋力得分
3     single_chess_point = {
4         'c': 989,    # 车
5         'm': 439,    # 马
6         'p': 442,    # 炮
7         's': 226,    # 士
8         'x': 210,    # 象
9         'z': 55,     # 卒
10        'j': 65536   # 将

```

```
11     }
12     # 红兵（卒）位置得分
13     red_bin_pos_point = [
14         [1, 3, 9, 10, 12, 10, 9, 3, 1],
15         [18, 36, 56, 95, 118, 95, 56, 36, 18],
16         [15, 28, 42, 73, 80, 73, 42, 28, 15],
17         [13, 22, 30, 42, 52, 42, 30, 22, 13],
18         [8, 17, 18, 21, 26, 21, 18, 17, 8],
19         [3, 0, 7, 0, 8, 0, 7, 0, 3],
20         [-1, 0, -3, 0, 3, 0, -3, 0, -1],
21         [0, 0, 0, 0, 0, 0, 0, 0, 0],
22         [0, 0, 0, 0, 0, 0, 0, 0, 0],
23         [0, 0, 0, 0, 0, 0, 0, 0, 0],
24     ]
25     # 红车位置得分
26     red_che_pos_point = [
27         [185, 195, 190, 210, 220, 210, 190, 195, 185],
28         [185, 203, 198, 230, 245, 230, 198, 203, 185],
29         [180, 198, 190, 215, 225, 215, 190, 198, 180],
30         [180, 200, 195, 220, 230, 220, 195, 200, 180],
31         [180, 190, 180, 205, 225, 205, 180, 190, 180],
32         [155, 185, 172, 215, 215, 215, 172, 185, 155],
33         [110, 148, 135, 185, 190, 185, 135, 148, 110],
34         [100, 115, 105, 140, 135, 140, 105, 115, 110],
35         [115, 95, 100, 155, 115, 155, 100, 95, 115],
36         [20, 120, 105, 140, 115, 150, 105, 120, 20]
37     ]
38     # 红马位置得分
39     red_ma_pos_point = [
40         [80, 105, 135, 120, 80, 120, 135, 105, 80],
41         [80, 115, 200, 135, 105, 135, 200, 115, 80],
42         [120, 125, 135, 150, 145, 150, 135, 125, 120],
43         [105, 175, 145, 175, 150, 175, 145, 175, 105],
44         [90, 135, 125, 145, 135, 145, 125, 135, 90],
45         [80, 120, 135, 125, 120, 125, 135, 120, 80],
46         [45, 90, 105, 190, 110, 90, 105, 90, 45],
```

```

47         [80, 45, 105, 105, 80, 105, 105, 45, 80],
48         [20, 45, 80, 80, -10, 80, 80, 45, 20],
49         [20, -20, 20, 20, 20, 20, 20, -20, 20]
50     ]
51     # 红炮位置得分
52     red_pao_pos_point = [
53         [190, 180, 190, 70, 10, 70, 190, 180, 190],
54         [70, 120, 100, 90, 150, 90, 100, 120, 70],
55         [70, 90, 80, 90, 200, 90, 80, 90, 70],
56         [60, 80, 60, 50, 210, 50, 60, 80, 60],
57         [90, 50, 90, 70, 220, 70, 90, 50, 90],
58         [120, 70, 100, 60, 230, 60, 100, 70, 120],
59         [10, 30, 10, 30, 120, 30, 10, 30, 10],
60         [30, -20, 30, 20, 200, 20, 30, -20, 30],
61         [30, 10, 30, 30, -10, 30, 30, 10, 30],
62         [20, 20, 20, 20, -10, 20, 20, 20, 20]
63     ]
64     # 红将位置得分
65     red_jiang_pos_point = [
66         [0, 0, 0, 0, 0, 0, 0, 0, 0],
67         [0, 0, 0, 0, 0, 0, 0, 0, 0],
68         [0, 0, 0, 0, 0, 0, 0, 0, 0],
69         [0, 0, 0, 0, 0, 0, 0, 0, 0],
70         [0, 0, 0, 0, 0, 0, 0, 0, 0],
71         [0, 0, 0, 0, 0, 0, 0, 0, 0],
72         [0, 0, 0, 0, 0, 0, 0, 0, 0],
73         [0, 0, 0, 9750, 9800, 9750, 0, 0, 0],
74         [0, 0, 0, 9900, 9900, 9900, 0, 0, 0],
75         [0, 0, 0, 10000, 10000, 10000, 0, 0, 0],
76     ]
77     # 红相或士位置得分
78     red_xiang_shi_pos_point = [
79         [0, 0, 0, 0, 0, 0, 0, 0, 0],
80         [0, 0, 0, 0, 0, 0, 0, 0, 0],
81         [0, 0, 0, 0, 0, 0, 0, 0, 0],
82         [0, 0, 0, 0, 0, 0, 0, 0, 0],

```

```

83         [0, 0, 0, 0, 0, 0, 0, 0, 0],
84         [0, 0, 60, 0, 0, 0, 60, 0, 0],
85         [0, 0, 0, 0, 0, 0, 0, 0, 0],
86         [80, 0, 0, 80, 90, 80, 0, 0, 80],
87         [0, 0, 0, 0, 0, 120, 0, 0, 0],
88         [0, 0, 70, 100, 0, 100, 70, 0, 0],
89     ]
90
91     red_pos_point = {
92         'z': red_bin_pos_point,
93         'm': red_ma_pos_point,
94         'c': red_che_pos_point,
95         'j': red_jiang_pos_point,
96         'p': red_pao_pos_point,
97         'x': red_xiang_shi_pos_point,
98         's': red_xiang_shi_pos_point
99     }
100
101     def __init__(self, team):
102         self.team = team
103
104     def get_single_chess_point(self, chessmap, src):
105         chess = chessmap[src[0]][src[1]]
106         if chess[0] == self.team:
107             return self.single_chess_point[chess[1]]
108         else:
109             return -1 * self.single_chess_point[chess[1]]
110
111     def get_chess_pos_point(self, chessmap, src):
112         row, col = src
113         chess = chessmap[row][col]
114         red_pos_point_table = self.red_pos_point[chess[1]]
115         if chess[0] == 'r':
116             pos_point = red_pos_point_table[row][col]
117         else:
118             pos_point = red_pos_point_table[9 - row][col]

```

```

119         if chess[0] != self.team:
120             pos_point *= -1
121         return pos_point
122
123     def evaluate(self, chessmap):
124         point = 0
125         for chess in fetch_any_chess(chessmap):
126             point += self.get_single_chess_point(chessmap, chess)
127             point += self.get_chess_pos_point(chessmap, chess)
128         return point

```

### 3.2 具有历史表启发以及 $\alpha - \beta$ 剪枝的 Minimax 函数

```

1 history_map = {}
2 # 此处初始化history
3 def history_get(x):
4     if not history_map[x]:
5         history_map[x] = 0
6     return history_map[x]
7
8 def history_renew(x, depth):
9     history_map[x] += 2 << depth
10
11 def alpha_beta(state, alpha, beta, depth):
12     if depth == 0:
13         return evaluate(state)
14     camp = 'r' if depth % 2 == 0 else 'b' # 深度决定行动方
15     steps = generate_possible_steps(state, camp)
16     steps.sort(key=history_get)
17     next_states = [make_next_state(state, src, dst) for src, dst in steps]
18     target = None
19     for i, next_state in enumerate(next_states):
20         temp = alpha_beta(next_state, -beta, -alpha, depth - 1)
21         if temp > alpha:
22             alpha = temp
23             target = steps[i]

```

```
24         if alpha >= beta:
25             break
26
27     if target is not None:
28         history_renew(target, depth)
29     return alpha
```

## 4 结果分析

同目录的 `result.mp4` 文件展示了一个模拟棋局的效果. 可以发现该算法具有一定的智能, 至少对于一个象棋菜鸟来说绰绰有余.

## 5 总结

博弈树搜索作为一种特殊的搜索, 让人们发现了特殊的 Minimax 算法进行处理, 而  $\alpha-\beta$  剪枝, 启发式评估函数和历史启发表则一步步地从不同角度优化 Minimax 算法的时空复杂度. 正是这种精益求精的精神让我们的算法效率更高, 让我们处理问题的速度也更快.

## 参考文献

- [1] 徐心和, 王骄. 中国象棋计算机博弈关键技术分析[J]. 小型微型计算机系统, 2006, 27(6): 961-969.
- [2] SCHAEFFER J. The history heuristic and alpha-beta search enhancements in practice [J]. IEEE transactions on pattern analysis and machine intelligence, 1989, 11(11): 1203-1212.