

****

**基于极大极小搜索的五子棋**

2017年6月

# 一、功能说明：

五子棋是一种两人对弈的纯策略型棋类游戏，通常双方分别使用黑白两色的棋子，下在棋盘直线与横线的交叉点上，先形成5子连线者获胜。其基本规则为:(1)对局双方各执一色棋子；(2)空棋盘开局；(3)黑先、白后，交替下子，每次只能下一子；(4)棋子下在棋盘的空白点上，棋子下定后，不得向其它点移动，不得从棋盘上拿掉或拿起另落别处；(5)黑方的第一枚棋子可下在棋盘任意交叉点上）。

本程序实现了基于这一系列规则进行人机对弈的智能五子棋游戏。

以玩家执黑先手，电脑执白，并且具有悔棋功能。

# 二、整体设计思路

①确定游戏规则：为了实现简单，该五子棋没有采用国际比赛的有禁手规则，采用的是普通游戏玩法无禁手规则：即可以在棋盘任何没有落子的位置进行落子。

②实现游戏界面：UI设计，实现悔棋功能

③游戏逻辑设计：如何判定获胜，如何判定最佳落子位置

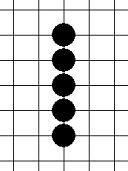
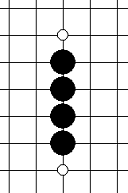
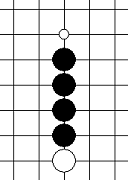
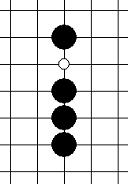
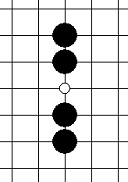
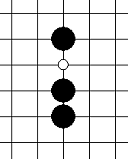
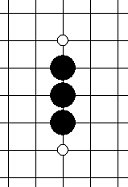
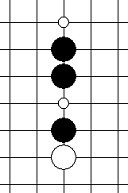
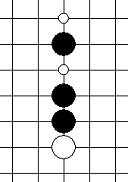
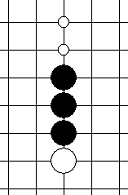
④构造棋型估分函数，用于进行决策

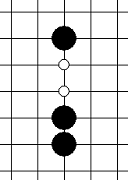
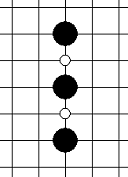
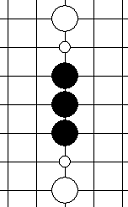
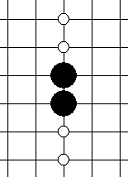
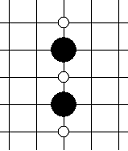
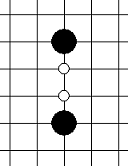
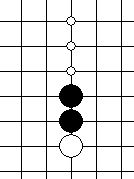
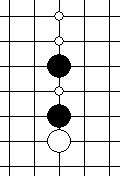
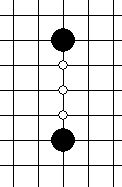
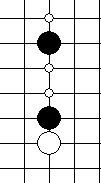
⑤性能优化：1.应用博弈树 2. α—β剪枝 3.限定下棋范围。

# 三、关键算法与实现

## （一）评估函数

1.五子棋基本棋型：

最常见的基本棋型大体有以下几种：**连五，活四，冲四，活三，眠三，活二，眠二**。  
**①连五**：顾名思义，五颗同色棋子连在一起，不需要多讲。  
  
  
**②活四**：有两个连五点（即有两个点可以形成五），图中白点即为连五点。  
稍微思考一下就能发现活四出现的时候，如果对方单纯过来防守的话，是已经无法阻止自己连五了。  
   
**③冲四**：有一个连五点，如下面三图，均为冲四棋型。图中白点为连五点。  
相对比活四来说，冲四的威胁性就小了很多，因为这个时候，对方只要跟着防守在那个唯一的连五点上，冲四就没法形成连五。  
     
**④活三**：可以形成活四的三，如下图，代表两种最基本的活三棋型。图中白点为活四点。  
活三棋型是我们进攻中最常见的一种，因为活三之后，如果对方不以理会，将可以下一手将活三变成活四，而我们知道活四是已经无法单纯防守住了。所以，当我们面对活三的时候，需要非常谨慎对待。在自己没有更好的进攻手段的情况下，需要对其进行防守，以防止其形成可怕的活四棋型。其中下图中间跳着一格的活三，也可以叫做跳活三。  
   
**⑤眠三：**只能够形成冲四的三，如下各图，分别代表最基础的六种眠三形状。图中白点代表冲四点。眠三的棋型与活三的棋型相比，危险系数下降不少，因为眠三棋型即使不去防守，下一手它也只能形成冲四，而对于单纯的冲四棋型，我们知道，是可以防守住的。  


     
  
如上所示，眠三的形状是很丰富的。  
  
  
**⑥活二**：能够形成活三的二，如下图，是三种基本的活二棋型。图中白点为活三点。  
活二棋型看起来似乎很无害，因为他下一手棋才能形成活三，等形成活三，我们再防守也不迟。但其实活二棋型是非常重要的，尤其是在开局阶段，我们形成较多的活二棋型的话，当我们将活二变成活三时，才能够令自己的活三绵绵不绝微风里，让对手防不胜防。  
     
  
**⑦眠二**：能够形成眠三的二。图中四个为最基本的眠二棋型。图中白点为眠三点。  
  

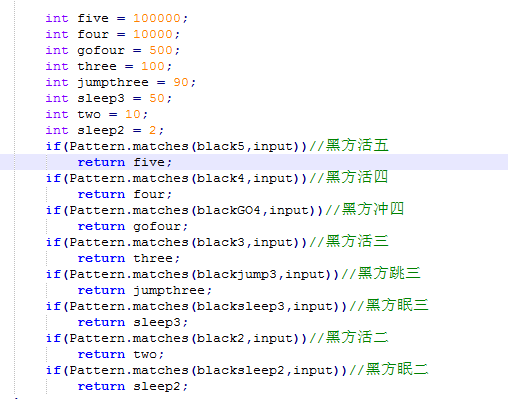
## 2.设置棋型权重；

根据棋型威力决定权重值。

具体来说：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 棋型 | 活五 | 活四 | 冲四 | 活三 | 跳三 | 眠三 | 眠二 | 活二 |
| 分数 | 100000 | 10000 | 500 | 100 | 90 | 50 | 10 | 2 |

在代码中体现如下：



### 3.得出评估策略；

①对每个已落子，其半径为二的米字型范围内是合理的下一步落点，即每个已落子最多有16个可能的合理落点

②对下一步的可能落点进行评估时，评估选取以改点的中心，半径为5的米字型区域进行棋型匹配，求出该点的得分总和，即为改点得分。



③使用正则表达式进行匹配

④得分总和为= 该点对我方利益+该点削弱的对方利益

⑤每次落子选取得分最大的一步

## （二）性能优化

但是，这样的策略对下一步优化带来一定的不便。为了满足之后剪枝的需求，我们在代码实现时做出了一些修改。之前那种搜索只是判断新落点对棋盘的影响，无法判断整个棋盘的局势，也没有办法进行更深一步的极大极小搜索与剪枝。

新的评估函数：新落点对局势的影响-🡪 整个棋盘的局势估计。

##### 整体局势评估函数

棋盘上每一个白色棋子能够形成的连子局势进行全局求和为电脑方分数，棋盘上每一个黑色棋子能够形成的连子局势进行全局求和为玩家分数。将电脑方分数减去玩家分数为该局势对电脑的评估函数值。以此为基础运用基于极大极小搜索的博弈树。

1. 应用博弈树：

对弈中,根据下一步由谁来走,AI对任何一个局面根据前面估分方法给出一个评估函数，并返回分值。据此来选择下一步的走法。

1. 限定下棋范围：

这里的下棋范围和之前的一步判断搜索是一样的。每个已落点附近的2\*2的范围内为有效范围落点，否则我们认为这是不合法的落点不进行搜索。

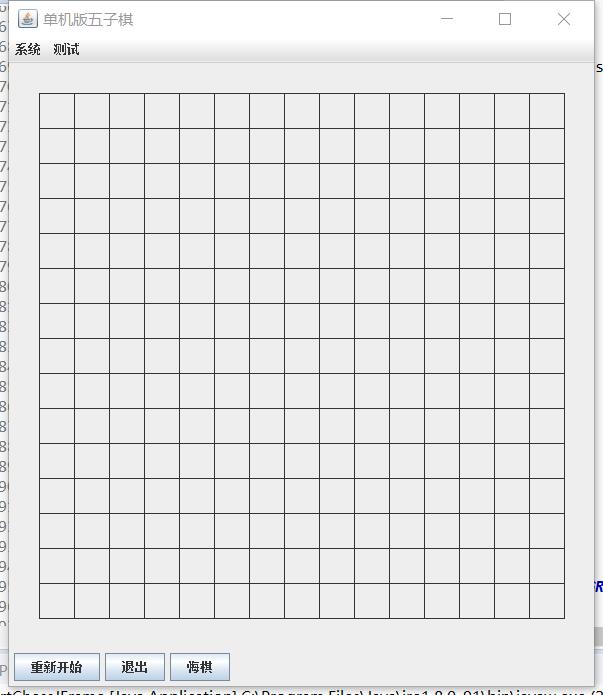
##### 4. Alpha-beta剪枝：

①当前为AI下棋节点：α剪枝：如果当前节点的值不比父节点的前兄弟节点的大值大,则舍弃此节点；β剪枝：如果当前节点子节点的值不比当前节点的前兄弟节点中的最小值小,则舍弃该子节点和该子节点的所有后兄弟节点。

②当前为用户下棋节点：α剪枝：如果当前节点的某子节点的值不比当前节点的前兄弟节点中的最大值大,则舍弃该子节点和该子节点的所有后兄弟节点；β剪枝：如果当前节点的子节点的值不比当前的父节点的前兄弟节点中的最小值小则舍弃此节点。经过α-β剪枝，可以极大的减少搜索的数量。

# 四、效果展示

# UI界面



## 博弈过程

黑子为玩家下棋，白色棋子为电脑落子。经过多次测试，玩家需要经过较强的思考才能够在二十五回合左右击败电脑玩家。而且如果玩家出现破绽，电脑会准确的找到破绽并获胜。

