# AI五子棋实验报告

**选型：**

由于对深度学习等人工智能算法接触十分少，对其改进又相对困难，因此决定实现alpha-beta剪枝的五子棋程序。

**平台** :

Ubuntu 16.04+g++

**实现思路 :**

1. **我们先介绍五子棋的基本棋形**

|  |  |  |
| --- | --- | --- |
| 1. **序** | **解释** | **图示** |
| **①** | **连五**：顾名思义，五颗同色棋子连在一起 | https://imgsa.baidu.com/forum/w%3D580/sign=ca8f95c4d439b6004dce0fbfd9503526/9230fdfaaf51f3de6bce2f4395eef01f3a2979bb.jpg |
| **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。 | https://imgsa.baidu.com/forum/w%3D580/sign=544199e43b87e9504217f3642038531b/fcf8b48f8c5494ee125e993e2cf5e0fe99257ebb.jpg |
| **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。 | https://imgsa.baidu.com/forum/w%3D580/sign=d36b654b113853438ccf8729a313b01f/ff393a292df5e0fe7f641f265d6034a85edf72bb.jpg https://imgsa.baidu.com/forum/w%3D580/sign=a21079a34bed2e73fce98624b701a16d/0560f21fbe096b63624ec8770d338744ebf8ac6c.jpg https://imgsa.baidu.com/forum/w%3D580/sign=334be987f703738dde4a0c2a831bb073/406c20a4462309f71a183b35730e0cf3d7cad64c.jpg |
| **④** | **活三**：可以形成活四的三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。中间跳着一格的活三，也可以叫做跳活三。 | https://imgsa.baidu.com/forum/w%3D580/sign=59503004b17eca80120539efa1239712/b3175c6034a85edf66f0680448540923dd5475bb.jpg https://imgsa.baidu.com/forum/w%3D580/sign=1bf30a164d086e066aa83f4332087b5a/db0c2834349b033bf8df6afd14ce36d3d539bd6c.jpg |
| **⑤** | **眠三：**只能够形成冲四的三 | https://imgsa.baidu.com/forum/w%3D580/sign=63acab3343a7d933bfa8e47b9d4bd194/b7bfa9014c086e06718aae6c03087bf40ad1cb99.jpg https://imgsa.baidu.com/forum/w%3D580/sign=27e3afe330adcbef01347e0e9caf2e0e/bcf89c82d158ccbf9bc0231d18d8bc3eb13541bb.jpg https://imgsa.baidu.com/forum/w%3D580/sign=d09c00db7dd98d1076d40c39113fb807/b318b13533fa828b585f0bc4fc1f4134970a5abb.jpg https://imgsa.baidu.com/forum/w%3D580/sign=e942cd111c950a7b75354ecc3ad1625c/3bbd033b5bb5c9eab08495c4d439b6003af3b34c.jpg https://imgsa.baidu.com/forum/w%3D580/sign=1e03059c86d6277fe912323018381f63/03158744ebf81a4cc80d7deed62a6059252da684.jpg https://imgsa.baidu.com/forum/w%3D580/sign=81c4d19c4610b912bfc1f6f6f3fcfcb5/bc3febc4b74543a9cef36ceb1f178a82b80114d6.jpg |
| **⑥** | **活二**：能够形成活三的二 | https://imgsa.baidu.com/forum/w%3D580/sign=032da31f29381f309e198da199014c67/5e29d9f9d72a6059b82f703d2934349b033bba84.jpg https://imgsa.baidu.com/forum/w%3D580/sign=afd63152d788d43ff0a991fa4d1fd2aa/85a4b9014a90f60320580a603812b31bb151edd7.jpg https://imgsa.baidu.com/forum/w%3D580/sign=a20cd7e783025aafd3327ec3cbedab8d/74d20ad162d9f2d32433b5c8a8ec8a136327cc99.jpg |
| **⑦** | **眠二**：能够形成眠三的二 | https://imgsa.baidu.com/forum/w%3D580/sign=e80aca111c950a7b75354ecc3ad1625c/3bbd033b5bb5c9eab1cc92c4d439b6003af3b384.jpg https://imgsa.baidu.com/forum/w%3D580/sign=bb97ae6c03087bf47dec57e1c2d3575e/5493c9ea15ce36d3733b12173bf33a87e950b184.jpg https://imgsa.baidu.com/forum/w%3D580/sign=f0bb7ed350da81cb4ee683c56267d0a4/893e367adab44aed73a628c3b21c8701a08bfbd7.jpg https://imgsa.baidu.com/forum/w%3D580/sign=580e15ea5366d0167e199e20a72ad498/21558bd4b31c8701c05e24c1267f9e2f0608ffd7.jpg |

在这里为了编程方便起见，我们不讨论禁手。

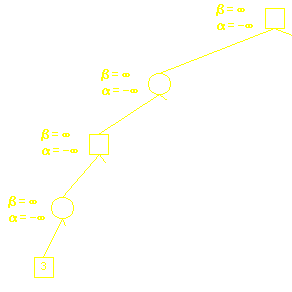
每一个棋形拥有不同的权值，从上到下的威胁度开始降低，权值，自然也随之降低

这里我们让它的分数设为每个棋子乘以100分，即1个棋子100分，两个10000分以此类推。

**2， Alpha-Beta剪枝算法**

这个剪枝算法比较基础，假设α为下界，β为上界，对于α ≤ N ≤ β:

若 α ≤ β 则N有解。若 α > β 则N无解。



这里的正方形表示自己，圆形表示别人，我们要做到让自己的得分最大化，而让对手的得分最小化，因此圆那一层是最小层，会取下一层中的最小值，而正方形的那一层是最大层，会取下一层的最大得分。

当圆圈在最小层时的值小于它上一层（最大层）的最大值，则说明这条路径中对手会选择比自己不利的地方落子。而现实生活中，每个人都会尽力不让自己输，因此这一步出现毫无意义，因此我们对其进行剪枝。

**3．贪心算法**

有的时候，由于对手下一些比较巧妙的棋，而Alpha-Beta剪枝法的层次又不深，则我们最后得出的值其实并不好，在这种情况下，我们决定使用贪心算法来弥补，贪心算法只追求当前最优步，与剪枝法类似，贪心算法也趋向于使落子分数最高，但是有所不同的是贪心算法是对每个落子的点进行评估，但是Alpha-Beta算法是对整个棋盘进行评分。

**4 . HashMap历史表**

因为我们的Alpha-Beta树中很多的节点和他们的评估信息是重复使用的，因此我们可以对他们进行记录，以便于下次有相同的情况时可以重新利用，比如<7,7><7,6>的下法和<7,6><7,7>的下法顺序不同，但是他们最终的局面是相同的，因此我们历史表就会起作用，网上提到使用zobrist哈希，称其对整个棋盘的编码速度极其的快，因为只需要做亦或操作即可，尤其是在悔棋的时候，只需再次做亦或即可恢复原来的状态。我用了网上的C语言库实现这个哈希，但是可能是没有悔棋操作，以及棋局并不大，测试的次数也不是很多，速度提升不是很明显。

**5. 其他未实现的针对alpha-beta剪枝的优化**

因为本人的编程能力不是很好，因此虽然调查到了许多对博弈树进行优化的方法，但是最终十分遗憾，未将他们加入到我们的五子棋当中。

**5.1迭代加深**

迭代加深的意思是指，逐步地进行迭代，因为有的时候局面十分胶着，我们设置的深度限制搜索出来的结果就会很肤浅，因此需要通过再次深入迭代来进行最优查找，前几次迭代可以为后面的几次迭代提供方向，而且通过历史表等方法保存一些局面的信息，使得迭代加深并没有增加特别多的时间消耗，而得出的结果却十分有效。

**5.2 启发式算法**

Alpha Beta剪枝的效率是取决于每一层节点的顺序的，倘若每一层的评分都能够按顺序排列，则剪枝的效率能够大大地增加，主要的原理是通过这个位置是否能成五，活四，活三等来进行打分，然后按照打分的高低进行排序。

**5.3 空步剪裁**

有的时候棋局上没有冲4，活3这种明显优势的棋局，这个时候进行剪枝会很不明晰，得出的结果不是很靠谱（这里我们用了贪心算法来弥补，但是后来找到方法说是可以通过空步剪裁的方式），其实就是轮到自己下的时候让对方再下一子，了解对方走子的可行性，以及对方再下一子对自己的威胁，然后再决定自己该如何下子。

**5.4 棋盘剪裁**

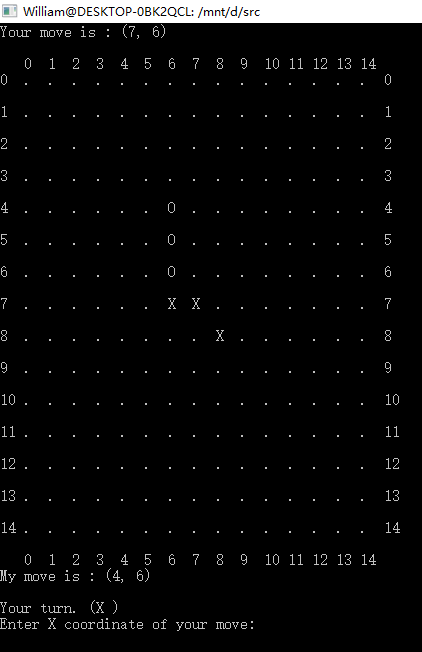
某些时候，我们由于下错，或者失误，将自己的子下的离主力太远，也就是说落子附近没有其他棋子，所以在搜索的时候我们可以直接将其暂时忽略，以达到介绍搜索的效果

**5.5 并行alpha beta算法**

我们可以通过并行算法来搜索树，以加快速度。

这些是比较常见的优化算法，其他的还有一些方法比如冲棋延伸、 吸出搜索、驳斥表，杀手启发，历史启发，MTD(f)算法，等等优化方法，由于能力问题。本次都没有利用上去。

**实验演示**



**参考资料和代码**

1. AlphaZero实战：从零学下五子棋（附代码）<https://zhuanlan.zhihu.com/p/32089487>
2. 五子棋AI博弈树之带Alpha-Beta剪枝的极大极小过程函数<https://blog.csdn.net/xyh_adolph/article/details/26730071>
3. alpha-beta剪枝搜索<http://www.cnblogs.com/IThaitian/p/3616550.html>
4. 极小极大搜索方法、负值最大算法和Alpha-Beta搜索方法<http://www.cnblogs.com/pangxiaodong/archive/2011/05/26/2058864.html>
5. 言川的博客<https://blog.csdn.net/lihongxun945/article/details/50668253>
6. Alpha-Beta剪枝(Alpha Beta Pruning)

<https://blog.csdn.net/tangchenyi/article/details/22925957>

1. CS 161 Recitation Notes - Minimax with Alpha Beta Pruning

<http://web.cs.ucla.edu/~rosen/161/notes/alphabeta.html>

1. 五子棋AI循序渐进【4】接近人类的思考方式

<https://www.cnblogs.com/zcsor/archive/2012/07/14/2591007.html>

1. 迭代加深搜索算法 <https://blog.csdn.net/myc0_0/article/details/79275435>
2. 空着向前裁剪<https://www.xqbase.com/computer/advanced_nullmove.htm>

11.并行博弈树搜索算法-第4篇 更上一层楼：Alpha-Beta算法的改进<https://blog.csdn.net/fsdev/article/details/7294902>

12. 五子棋AI循序渐进【6】置换表

<https://www.cnblogs.com/zcsor/archive/2012/07/23/2605211.html>

13. 十四步实现拥有强大AI的五子棋游戏

<http://www.cnblogs.com/Blog_SivenZhang/archive/2010/06/13/1757677.html>

14．QT五子棋详解之九：置换表，zobrist哈希，生成32位和64位随机数<https://blog.csdn.net/weixin_39788534/article/details/79539329>

15．<https://github.com/lemire/zobristhashing>

16. <https://github.com/AngelaViVi/Gomoku.git>

17. <https://github.com/tdang33/Gomoku-Five-in-a-row->

18. https://github.com/lihongxun945/gobang