1951505 姜星宇

人工智能原理第 3 次作业

一、作业内容

运用极大极小算法以及 α - β 剪枝算法分别设计一个基于对抗搜索的小游戏或者应用于具体项目中解决某个小问题.

二、实验原理

对抗搜索主要运用在棋类博弈中, 而极大极小算法是解决对抗搜索问题的主要途径. 极大极小搜索主要就是假设将某一局面每一步分支展开成一棵树, 叶子结点即为最终的效用值, 回溯时假设交替选择最大值与最小值, 这样可以有效地模拟双方博弈的过程. 但是这种方法往往分支因子很大, 并且到终点的深度也十分深, 象棋的分支因子大约为三十多, 而步数大约是几十步, 需要遍历大于 30^{50} 个结点, 这样几乎是不可能用当前的计算水平完成的. 而博弈只是需要当前的最优解, 那么只需要给一个大概的估计值也可以, 这样就可以在任意的局面给一个分数(imperfect real-time decisions), 而不一定要到叶子结点, 不过根据当前结点很难完美地评价局面, 因此一个较好的程序仍然需要不断地加深层数.

加深层数可以用剪枝来实现. 基于最大最小算法的特点, 对于 MAX 层结点, 最大最小算法是要选择其子结点的最大值, 以和同一层的结点对应的最大值比较, 选择一个最小值出来回溯提供给父结点. 根据这一特点, 可以在这一层维护一个由父结点传入的作为已经遍历过的兄弟结点提供给父结点的最小值 β, 如果在这一个结点向下搜索最大值(对应的子节点中最大值 α)的过程中, 发现 $\alpha \geq \beta$, 也就是说这个结点所提供的最大值不会比 β 小, 那么一定不会被父结点所选中, 那么就可以直接回溯, 不再在这个结点继续搜索了, 也就是发生了 β 截断.

对于 α - β 剪枝, 仍然可以继续优化, 这些优化主要体现在对于搜索顺序的判定上, 能够尽早地发生截断. 这个优先顺序的判断不影响最终的结果, 之影响时间, 因此不一定要十分准确的, 只要能够在概率上提早截断的时间就是一种优化. 一般来说会用历史表排序(history heuristic), 杀手表(killer heuristic)等等. 为进一步提高搜索的效率, 对于某些在搜索中可能出现重复局面的博弈中, 还可以用置换表(transposition table)将已经搜索过的局面记录下来为后面的搜索提供信息, 一旦发现之前搜索过同样的局面并且深度不小于当前需要搜索的深度, 即可

以直接返回这个值. 对于五子棋这种不会出现重复局面的情况则不需要考虑.

除搜索和其优化之外, 另一个十分重要的就是如何评价当前局面的好坏. 不像效用值函数, 评价函数不直接与规则相关. 对于五子棋, 可以根据连子的情况来判断, 而象棋可以用子力之和与棋子的位置来判断. 可以说评价函数是对于搜索层数限制的一个补偿, 一个好的评价函数可以缓解搜索层数带来的时间压力. 比如如果一个程序能够识别两边空的三子连珠, 那么就可以立刻决定应该下在哪里, 而搜到最终局面需要继续搜索四层, 而对于井字棋, 甚至很容易直接根据当前局面而不继续扩展子结点判断出最佳的位置. 但是一味追求好的评价函数也会让其变得十分复杂, 因此在搜索层数与评价函数的选取上要找到一个平衡.

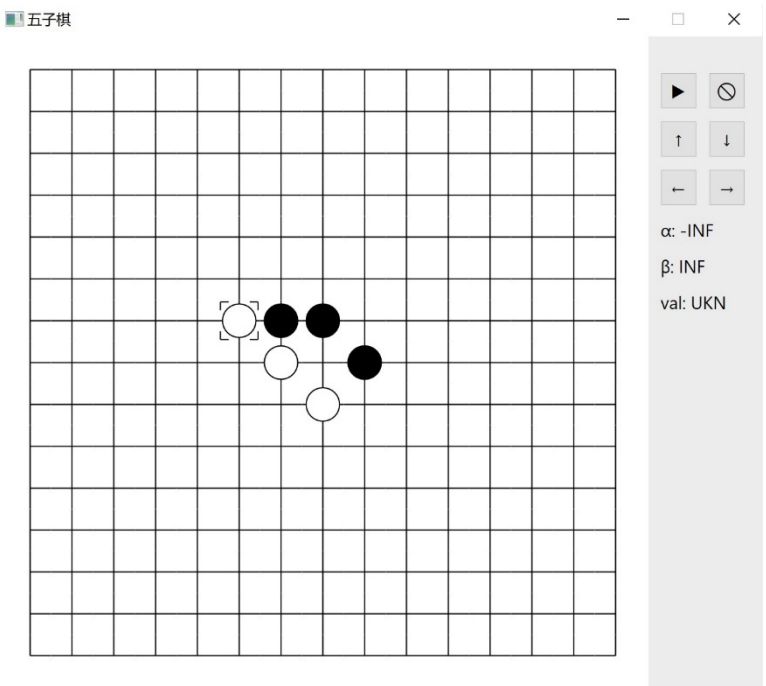
参考资料:

Russell, S. J., & Norvig, P. (2010). Artificial Intelligence-A Modern Approach, Third Edition.

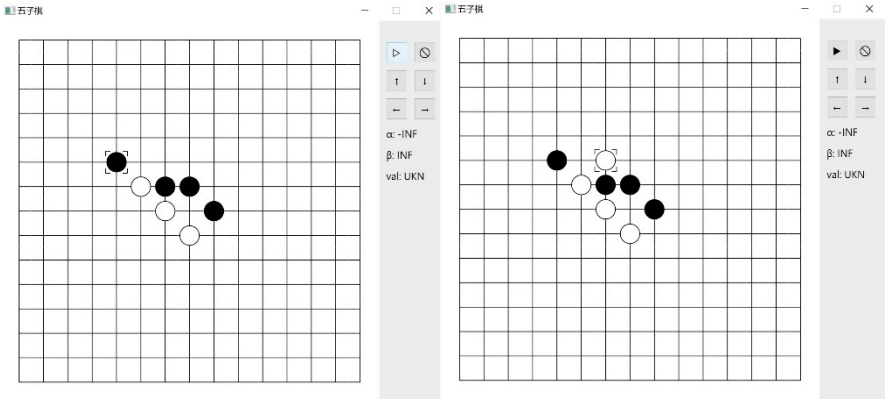
www.wikipedia.org, www.chessprogramming.org, www.xqbase.com.

三、实验内容

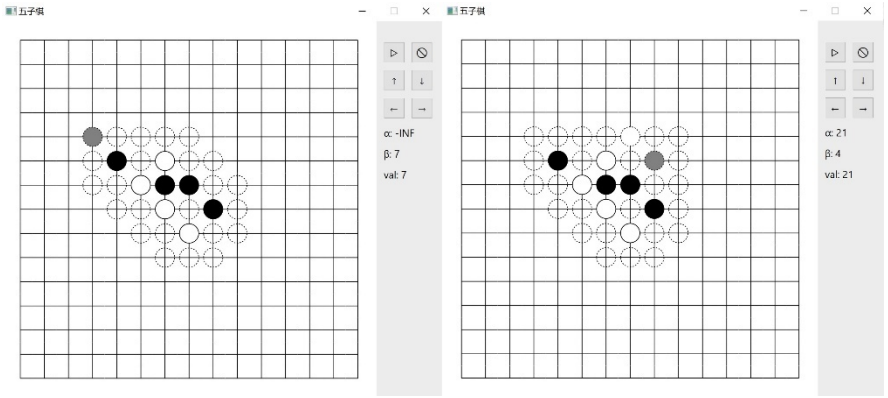
1. 源文件构成: (编译器 G++, MinGW-w64)
   - main.cpp: 所有函数包含在此文件中.
2. 实现功能:
   - 五子棋基本功能实现: 点击棋盘以下子. 黑棋先, 黑白交替. 点击左箭头与右箭头撤销与重复. 点击圆圈符号初始化界面.



- 机器下棋：点击三角形，三角形开始闪烁，等待搜索线程返回最优解. 默认迭代加深搜索 7 层，可再次点击以终止搜索直接返回较浅结果.



- 手动分析：点击下箭头，开始手动模拟 MIN-MAX 搜索与 α - β 剪枝过程. 虚线表示待搜索的位置，此时左右箭头控制手动选择的位置. 上箭头表示回溯一层. (不过好像回溯时继续搜索忘记更新了，不过能体现一个结点下的 α - β 应该差不多了)



- 注：箭头可由键盘方向键代替，自动下棋与停止可以用回车键代替.

3. 主要数据结构:
   - `board` 结构体：其中 15×15 的存储棋盘矩阵.
   - `position` 结构体：两个成员 `i`, `j` 表示棋盘位置.
   - `history` 变量: `vector<position>`类型，用于存储历史位置.
   - `dashed` 变量: `vector<position>`类型，分析时待扩展的位置，在棋盘上以虚线表示.
   - `history_analyse` 变量: `vector<position>`类型，分析时的记录.
   - `st_dashed` 变量: `stack<vector<position>>`类型，用来存储待扩展位置的历史记录.

4. 主要函数:
   - `int Eval(board &brd)`: 评估函数, 在本次实验中, 仅以各方向上的可活动连子个数加权之和评估局面.
   - `int MaxValue(board &brd, int beta, int depth, int total, position &pos_ret)`: 求 MAX 结点下的最大值, 如果是根节点(`depth==0`)即将结果记录在 `pos_ret` 中.
   - `int MinValue(board &brd,int alpha, int depth, int total, position &pos_ret)`: 与上类似, 求 MIN 结点下的最小值.
5. 可视化实现方法: 调用 WinAPI 与 Direct2D.

四、结果分析

1. 五子棋结果分析:

   由于很大程度上与课程设计的大作业是重复的, 因此没有过于深入开发五子棋的功能, 只是把象棋的 GUI 搬过来改一改, 于是就没能通过引擎之间的对局来体现算法的优劣. 不过可以确定的是, 这个最简单的判断连子个数的评估函数和最简单的 6 到 7 层的 α-β 算法已经可以看上去不那么随便乱下了, 大部分情况下我是下不过它的(估计还是我太弱了). 通过分析的功能, 虽然 α 和 β 的显示还不是十分完善, 不过确实也可以比较艰难地看出来在某一结点的延伸过程中发生了截断.

2. 象棋结果分析:

   由于大作业象棋也是一样的对抗搜索, 我就把那边得到的一些结论搬过来. 由于象棋有一个统一的接口, 因此评测起来要方便得多. 它们的分支因子差不多, 在中局大约都是二三十, 三四十的样子. 现在主要来看一下不同的搜索方法对于评估的结点数量与层数的影响.

   下面是各个不同的方法得到的在 30 秒计算的深度即评估结点数量.

```
                                    info depth: 3   time: 13
info depth: 3   time: 39  min-max   info depth: 4   time: 146   α-β
info depth: 4   time: 1695          info depth: 5   time: 1225
info count nodes total: 56650004    info count nodes total: 39719586

info depth: 3   time: 6             info depth: 3   time: 14
info depth: 4   time: 73            info depth: 4   time: 129   置换表
info depth: 5   time: 767  历史表    info depth: 5   time: 917
info depth: 6   time: 5830          info depth: 6   time: 21317
info count nodes total: 7446889     info count nodes total: 43862153
```

```
info depth: 3   time: 8
info depth: 4   time: 60
info depth: 5   time: 448
info depth: 6   time: 1763
info depth: 7   time: 22955
info count nodes total: 8781412
```

置换表+历史表

```
info depth: 3   time: 2
info depth: 4   time: 8
info depth: 5   time: 21
info depth: 6   time: 53
info depth: 7   time: 128
info depth: 8   time: 265
info depth: 9   time: 435
info depth: 10  time: 1194
info depth: 11  time: 5947
info depth: 12  time: 22945
info count nodes total: 13525057
```

杀手启发

总的来说, α - β 剪枝对于评估的结点数量没有什么贡献, 只是说能够大大缩小分支因子, 让搜索层数增加. 同样对于历史表排序也是如此, 因为排序的时间也是比较多的, 结点数量下降了不止一点, 但是搜索同样层数的时间却能够大大缩小. 最后的杀手启发更能是增加搜索的层数, 因为在生成与排序之前就已经被截断了(不过现在不太确定这个结果是否准确, 因为对局的结果并不是增加很多甚至还有些下降, 可能是哪里的小 bug 吧, 但是它的作用应该会是很显著的).

3. 总结:

计算机博弈也是这几年人工智能进入大众视野的主要原因之一, 不过它的历史也是有一些的了. 这些模拟博弈的搜索算法以及对于它的优化真的是让人感到人类的智慧与计算机的强大. 同时在每一种的算法上进行进一步的改进也越来越难, 也许就应当是一种循序渐进的过程吧. 智能在这里不仅体现于评价函数上, 也在各种剪枝与截断上, 毕竟在人类下棋看来, 某些棋显然就是不好的, 当然应该直接舍去. 博弈虽然说是蕴含了不少智能与智慧, 但是毕竟还是有规则与标准的, 这方面机器超过人类是很正常的事情, 人工智能看上去还有很长的路要走.

五、文件清单

```
1951505-姜星宇-第三次作业
  |--实验报告.pdf
  |--实验报告.pptx
  |--pic(结果图片)
  |   |--(…)
  |--main.cpp
  |--main.exe
```