五子棋AI实验报告

傅若山

该五子棋AI规定只能由玩家执黑，没有设置悔棋、禁手、时限。

使用Clion为开发工具，easyx为图形库。配置开发环境方式：下载libeasy.a和easyx.h、graphics.h，将libeasyx.a放入D:\CLion 2022.2.4\bin\mingw\x86\_64-w64-mingw32\lib下，将easyx.h和graphics.h放入D:\CLion 2022.2.4\bin\mingw\x86\_64-w64-mingw32\include下，然后再Clion中include头文件graphics.h，并更改CMakeLists.txt文件：在add\_executable前添加link\_libraries(libeasyx.a)即可。

为了使用easyx图形库工具，我将使用c语言写的五子棋代码放入c++中执行并辅以图形化，下述将只讲述以c语言进行的AI代码设计。

实战中模块主要分为总体执行模块main.c、基础树形结构tree.c、评分函数模块setscore.c、遍历函数模块play.c。

Tree.c模块主要构造由多层结点组成的用于储存遍历结果的树形结构，其中包含结构体node类，以及树的相关操作creatTree（创建头节点）、addChildrenNode（增加子节点）、deleteTree1（彻底删除该根节点）、deleteTree2（保留该节点并彻底删除其所有子节点）、keepTree（在该节点的兄弟节点中只保留该节点，彻底删除其他节点）。

setscore.c模块主要包含对于某步棋的attackScore、defendScore的评判（评判之后可以根据现有的评分进行一定的剪枝）、对其两者关系的论述defend\_attack、将分数加总函数getToBottom、最终比较函数cmpNode以及判定是否已经获胜的函数ifWinner。这是我的思考重点。

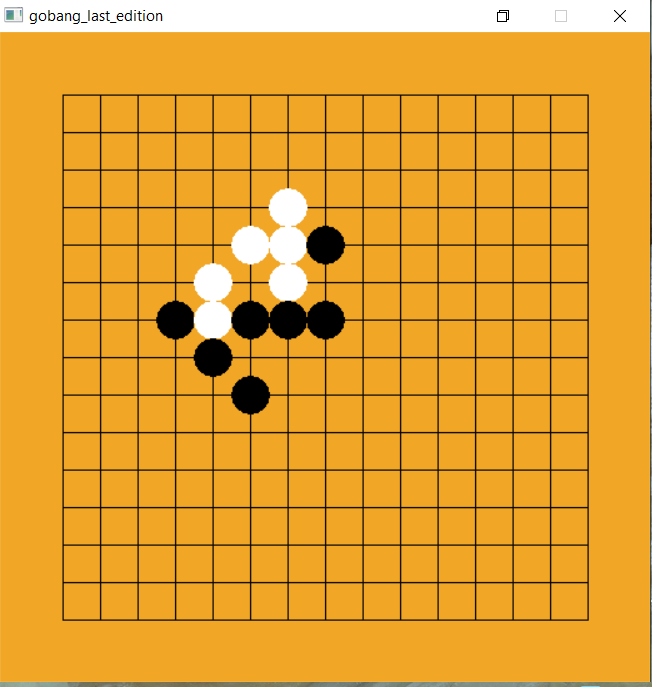
Play.c模块主要包含遍历函数traverse、相关剪枝函数以及对AI下棋的集大成函数playAI函数。

Main.c主要是图形化界面以及下棋的过程。

下面主要讲setScore模块的设计。（0代表无棋，\*代表白棋，+代表黑棋）

1. defendScore:这里设计的比较简单，只defend必输棋型，如0++++\*（下称“有遮挡成四型”）和0+++0（下称“无遮挡成三型”），但可能出现这两种情况同时出现，或同一情况出现多次等复杂情形，由于可能出现反杀（与attackScore的关系），所以应该遵循能多活一步是一步的原则，即先挡成四型，再挡成三型，与attackScore关系较为复杂，第三点会论述。
2. attackScore：分为成四、三、二三种大类，其中每种又分多种情况，如成四分为0++++0、0++++\*、\*++++0\*、\*+++0+\*等，每种分别赋分。
3. defend\_attack：第一，根据defend和attack之间的关系，在defend中我对阻止成五赋80000分、阻止有遮挡成四赋40000，在attack中成五赋100000分、成四60000，即在同一步中优先级成五>阻止对方成五>成四>阻止对方成四（下将上述四种情况分别称为“必胜棋型”或“必输棋型”；第二，若出现两个成四则分数会超过成五，然而实战我们肯定会选择成五的下法，所以这里也需要先进行分数的变换，如两个成四则赋分为70000。同理也有其他的分数需要变换。所以我们在比较分数前是需要先同父节点比较attackScore和defendScore进行剪枝。第三，对于defendScore的思考。遮挡3、4等棋在大局观下本身不是什么好棋，只是在当前棋中显得必要而已，若将defendScore与attackScore加和比较，会出现明显的弊端。如AI方会有因为挡四分数高而不愿挡三，挡二，即使挡二、挡三的位置恰好是绝佳的进攻位置。第五，综上述论述，在正常比较节点分数之前，我们还需确定是否有必胜或必输棋型，若有则优先考虑，不需要进行更细节的评分。且正常比较时只加和attackScore。
4. 分数加总模块：由于担心出现机器随机遍历时对人类的棋遍历过于“笨”，导致出现在多步对比下人类下的差棋造就的机器方面的好机会下出的类似成五之类的好棋而对整个评分分支的分值起决定性作用，我决定对于人类分支在同父母节点下仅选择分数最高的结点继续遍历。分数加总即为将所有分顺着分支加到最底层，注意人类的分应减去机器得分，体现极大极小的思想。
5. 为了弥补在刚开始下棋时缺少成三、四或挡三、四等棋型导致AI乱下，增添了成二、挡二、跳二等棋型的评分。

成效简介：先附图



AI棋力不太理想，可能由于遍历层数较多，AI下出来的棋我感觉很奇怪，但又不知道哪里应该修改。该AI程序大小212kB，最多遍历到第五层，且卡顿10秒左右，遍历范围为上一步棋的周围9\*9方格。

改进方向：

1.分数设定有待改进，可能可以通过合理的设定避免过多的同级结点比较，减少运算。

2.代码不够简洁，有很大的重构空间。

3.模块不够清晰，部分模块（如setScore和playAI）有明显的杂糅现象。