全规则五子棋 AI 报告

liuchang codingnoooob ycf

摘 要

本文主要讲述了全规则五子棋 AI 的实现思路以及过程。该 AI 主要使用了 Alpha-beta 剪枝以及与五子棋规则相结合的启发式搜索，最终实现了较为理想的效果。

关键词: 五子棋、AI、Alpha-beta 剪枝、启发式搜索。

1 全规则五子棋

五子棋完整规则除了一般为人熟知的“五子连珠”取胜之外，还有三手开局、三手交换、五手二打以及禁手。这些规则主要是为了限制黑方的先手优势。在完整规则下，黑白双方胜率相当。
市面上五子棋软件很多，但往往是仅有“五子连珠”规则的简单游戏，带有完整规则的五子棋游戏极其稀少。

1.1 三手开局

执黑一方需要在开局时在天元周围连下“黑白黑”三颗子，完成开局。26 种开局中大部分为黑方优势，少部分为平衡，极少部分白方优势。

1.2 三手交换

三手开局完成后轮到执白一方选择是否交换。由于不同开局黑白双方优劣不同，白方可以选择交换双方颜色，以避免过大的黑方开局优势。

1.3 五手二打

执黑方在下第五步时需要给出两个点位，交由白方选择哪个留下。此规则也是为了限制黑方的先手优势。

1.4 禁手

黑方不得一子落下同时形成两个及以上的活三，或是两个及以上的活四，或是六个及以上连珠的长连，否则判负。白方可以逼迫黑方形成这样的落子来取胜，这称之为“抓禁手”。此规则也是为了限制黑方的先手优势。

2 AI 主要方法

本全规则五子棋 AI 主要使用了 Alpha-beta 剪枝和启发式搜索两种方法。Alpha-beta 剪枝能有效减少搜索节点数量，从而提升计算效率。启发式搜索则是对 Alpha-beta 剪枝的辅助，良好的启发式搜索能对节点进行有效的排序，从而使得更多的节点被剪掉。

2.1 Alpha-beta 剪枝

Alpha-beta 剪枝是对 Minimax 算法的变种, 能在保证搜索结果正确的前提下，剪掉无用的节点，提升计算效率。
由于五子棋一般只有绝对的胜负，对手落子期望的计算意义不大，这适用于对抗搜索中的 Minimax 算法，故可以使用 Alpha-beta 剪枝，以提升搜索深度，使得 AI 拥有更高的棋力。

2.2 启发式搜索

启发式搜索利用评估函数来引导搜索。在五子棋 AI 中, 启发式搜索的目的是将更有希望的节点拍到搜索树的前面，从而进一步降低搜索量。
由于以上作用，启发式搜索的评估函数极为重要，有效的评估函数不仅能使搜索量下降, 还能改变五子棋 AI 的棋风。

2.3 评估函数

评估函数主要考虑以下方面:

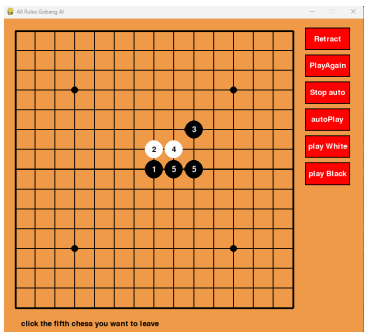
- 连接
  五子棋要保持先手或取得优势需要更多的连接。因此，连接的数量应作为评估函数的重要部分，即应以活四、活三、活二等连接纳入评估函数。
- 开局
  上文提到，不同的开局黑白双方优劣不同，故应对不同的开局有不同的思考方式，即评估函数对不同的开局应有不同的参数。具体方法有：执白方面对黑必胜开局应直接交换，并采用更具侵略性的棋路；面对优势不大的开局，应采用更稳健的棋路。
- 特殊局部
  菱形、叉形、小八卦以及大八卦等特殊局部，往往有利于后期的发展，评估函数应将这些特殊局部纳入考虑。
- 距离
  一般而言，下一个落子会贴近已有的落子，故应先考虑已落子点周围的空位。

3 全规则五子棋 AI 的实现

市面上全规则五子棋极其稀少，且难以获取源代码，故五子棋由独立实现。
AI 框架参考了课内所学和网络上其他五子棋 AI，具体内容及参数设置由独立实现。

3.1 全规则五子棋

利用 pygame 库，较为全面地实现了全规则五子棋。可进行独自摆谱、与 AI 对抗、悔棋、重来等功能，五子棋完整规则亦全部完成。
以下为实际效果



3.2 AI

AI 部分在参考课内所学以及其他五子棋 AI 的基础上，加入了上文“评估函数”所述部分的调整，并根据实际对弈情况进行了微调。

- Alpha-beta 剪枝
  Alpha-beta 剪枝基本与课内所学一致，仅对子节点搜索部分略有调整。
- 启发式搜索的评估函数
  启发式搜索在对局势进行评估后给出评分，活四、活三等不同的局势得分不同，也会对于特殊局部、距离等有所调整。
  评估函数方程为:
  $h(s) = 10 \times O + 50 \times T + 100 \times S + 150 \times F + 9999999 \times W + 10 \times BT + 50 \times BS + 100 \times BF + 200 \times SP$
  其中符号意义如下:

符号	含义
O	活一
T	活二
S	活三
F	活四
W	连五
BT	眠二
BS	眠三
BF	眠四
SP	特殊局部

  在必胜开局后，活二与活三的系数将翻倍。

4 测试结果

- 时间
  以下为在作者使用的 i7-1065G7 上测得的时间

搜索步数	时间 (s)
4	0.05808
6	0.84831
8	5.28176
10	72.99105

市面常见五子棋 AI 搜索深度普遍在 4

至 6 步，以上所示时间并不逊于市面常见五子棋 AI 所需的 1 秒。
可见 8 步搜索深度是一个比较合理的设置。该 AI 在采用 8 步搜索深度时，所需时间短于市面常见五子棋 AI 的 10 秒左右，且棋力较强。
尽管搜索 10 步需要 73 秒左右，考虑到 10 步节点数量非常大，且正规比赛一共有一小时的思考时间，这是一个较为理想的反应时间。

- 与其他五子棋 AI 对弈表现
  该全规则五子棋 AI 在计算在与市面上四款五子棋 AI 小游戏分别对弈 5 盘后，共取得 14 胜 6 负的成绩，较为理想。
- 与人对弈表现
  与人对弈表现的评价较为主观。
  该全规则五子棋 AI 在面对提前设置的开局时表现良好，可以顺利地交换必胜局，并采取进攻；对局时能有效地寻找连接并截断对方的连接，从而不落入下风；能有效地避免胡乱冲四的情况；但是，该五子棋 AI 对大局把握较差，有时会出现在优势局面仍继续防守的情况。

5 外部代码

本次实现的全规则五子棋 AI，主要内容均为独立实现，由能力限制以及方便性的考虑，采用了一些基础库。

- alphabeta,search
  以 github.com/TDK1969/AI_experiment 为主要参考，实现了对活二、活三等的搜索，以及评估函数的框架。重写了其中的评估函数，亦实现了对完整规则的补充。
- pygame
  受到能力限制，为方便获取鼠标操作以及展示五子棋界面，调用了 pygame 库。
- numpy
  将五子棋作为 15×15 的二维数组较为方便，numpy 库可以提高计算速度，并提供了一些有效的函数。
- button
  根据 github.com/Mekire/pygame-button 的代码修改，实现了简易的按钮。
- os, sys
  演示画面需要。
- time
  作为随机数种子。