五子棋人机博弈问题

装 订 线

摘 要

五子棋是起源于我国古代的黑白棋种之一，是一种简单的娱乐性较强的大众游戏，深受广大玩家的喜爱，但同时作为比赛，五子棋游戏还有着深奥的技巧。本次实验通过对博弈树启发式搜索过程的熟悉和掌握，采用了α-β剪枝算法对树进行搜索，并设计了合适的评价函数对五子棋的博弈进行评估，最后应用 python 中的 Tkinter 库设计了五子棋的可视化界面，实现了五子棋的人机博弈，包含悔棋、重来等功能。

关键词：人工智能;α－β剪枝算法;五子棋

目录

装 订 线

1 实验概述

1.1 实验目的

熟悉和掌握博弈树的启发式搜索过程、α-β 剪枝算法和评价函数，并利用 α-β 剪枝算法开发一个五子棋人机博弈游戏。

1.2 实验内容

1.以五子棋人机博弈问题为例，实现 α-β 剪枝算法的求解程序（编程语言不限），要求设计适合五子棋博弈的评估函数。

2.要求初始界面显示 15*15 的空白棋盘，电脑执白棋，人执黑棋，界面置有重新开始、悔棋等操作。

3.设计五子棋程序的数据结构，具有评估棋势、选择落子、判断胜负等功能。

4.撰写实验报告，提交源代码（进行注释）、实验报告、汇报 PPT。

2 实验方案设计

2.1 总体设计思路与总体架构

2.1.1 设计思路

本实验采用了 α-β 剪枝算法对五子棋博弈树树进行搜索，并设计了合适的评价函数对五子棋的博弈进行评估，最后应用 python 中的 Tkinter 库设计了五子棋的可视化界面，实现了五子棋的人机博弈。

2.1.2 总体架构

本程序分为前后端，前端为 python 中的 Tkinter 库实现的 UI 设计和可视化模块，后端为 α-β 剪枝算法配合合适的评级函数作为五子棋的核心驱动模块。

2.2 核心算法及基本原理：详细说明核心算法以及其涉及的理论知识。

2.2.1 α-β 剪枝算法

α-β 剪枝算法一种基于剪枝的深度优先搜索，将走棋方定为 MAX 方，因为它选择着法时总是对其子节点的评估值取极大值，即选择对自己最为有利的着法，将应对方定为 MIN 方，因为它走棋时需要对其子节点的评估值取极小值，即选择对走棋方最为不利的、最有钳制作用的着法。在搜索 MAX 节点时，如果发现一个回合之后评估值变差，即孙节点评估值低于下界 α 值，则便可以剪掉此枝，即为 α 剪枝，β 剪枝算法同理。

2.2.2 评价函数

评价函在启发式搜索中，用于评价节点重要性的函数叫做评价函数。评价函数的主要任务就是估计等搜索结点的重要程度，以确定结点的优先级程度。评价函数的一般形式为 f(x)=g(x)+h(x); 其中 h(x)被称为启发函数。在五子棋的设计中，评级函数根据五子棋的规则与棋型，为每种不同的棋型设计不同的耗散值，最终为整个棋盘进行估值。

2.3 模块设计：本实验的具体模块设计

2.3.1 后端 α-β 剪枝算法及评价函数模块

对于评价函数，对每种棋型设计如下，其中只要已落下的棋子满足棋型说明中的情况，就给评估值加上对应的耗散值。

棋型代号	棋型说明	权重（耗散值）	棋型代号	棋型说明	权重（耗散值）
WIN	白方获胜	sys.maxsize / 2	LOSE	白方败北	-sys.maxsize / 2
wflex4	白活 4	50000	bflex4	黑活 4	-100000

wrush4	白冲 4	400	brush4	黑冲 4	-100000
wflex3	白活 3	400	bflex3	黑活 3	-8000
wblock3	白眠 3	20	bblock3	黑眠 3	-50
wflex2	白活 2	20	bflex2	黑活 2	-50
wblock2	白眠 2	1	bblock2	黑眠 2	-3
wflex1	白活 1	1	bflex1	黑活 1	-5
wdead4	白死 4	-2	bdead4	黑死 4	5
wdead3	白死 3	-2	bdead3	黑死 3	5
wdead2	白死 2	-2	bdead2	黑死 2	5

图 2.1

在设计评价函数的权重时应当注意：

敌我双方的棋型相同时的权重应当不同。因为考虑到先后手的缘故，在棋型相同的情况下，后手的估值的绝对值应当小些，因为下一步棋是先手下。例如在敌方是活三，我方是活二的情况下，去阻止对方形成活四应当比让我方形成活三重要。

在同一方的棋型中，应当满足连 5>活 4>冲 4=活 3>眠 3=活 2>眠 2=活 1 的规则，且差距设置在 20 倍左右比较合理。因为在实现对整个棋局的评估时，会对有些棋型进行重复计算，但是棋型往往是按照质量，即棋型的好坏程度来决定胜负而不是靠数量，比如有 10 个冲四和 1 个活四，按照五子棋的规则，形成活四的权值应当比 10 个冲四的权值要大，故要满足上述规则。

对于 α-β 剪枝模块，除了实现基础的 α-β 剪枝算法之外，因为整个棋盘有 15*15 的大小，因此每次计算遍历所有可能性是不现实的。因此考虑已有棋子周围 1-2 格宽度作为可能节点，舍去其它节点显然基本可以覆盖基本所有的最优解而不遗漏，同时大幅降低了节点数，极大的提高了运算速度。

α-β 剪枝模块维护一个 15*15 的数组和下次可能节点数组。每次下一个棋子，则在数组上进行修改，同时对新下的点进行可能节点的增加，保证每次都能对包括可能节点的所有棋子的周围都纳入考虑范围，实现了算法。

2.3.2 前端 GUI 模块

GUI 方面本次我采用了 Tkinter 库来设计本次五子棋的 GUI。整个设计分为了这样几步。初始化时分为了棋盘参数部分和 GUI 控件部分。为了在后续修改和绘制方便，在这里对前期棋盘的各个参数例如网格大小、比例、落点数、棋子半径、开始及黑方标志等等一系列参数进行了

设置。在 GUI 控件方面则采用了画布控件、按钮控件、标签控件等等来组织好整个棋盘以及其他功能。

2.3.2.1 棋盘棋子的绘制以及四个按钮功能的设计

设计五子棋的 GUI 最关键部分就是在各个元素的组织上。Tkinter 库则给了我们许多方便的的工具。而在画棋盘表格时涉及到的麻烦比较多。由于要考虑好后续悔棋重来等操作对于棋盘的改变，这一部分的修改是最多的，对于画棋子擦除棋子的改变为了做到更自然，在这方面遇到的阻碍最多。棋盘整个的框架搭好之后，后续要解决的便是棋盘功能的设计。四个按钮对应的开始、重来、悔棋、认输四个函数分别承担了这几项功能。对于棋盘处理的功能，我是获取了整个棋盘的数据来进行处理。这四个功能之中，开始与重来都相当于初始化整个棋盘数据，认输后续也是回到初始化阶段，这三个功能相对来说比较好设计。

相对比较难设计的便是悔棋。在这里每次需要记录双方下的上一步棋，然后根据坐标来对应绘制 x 行 y 列处的表格。这也是最开始绘制表格时我采用每处表格单独绘制的原因，不必每次悔棋都重写画整个表格与棋子，相对来说效率上要更高。悔棋这里我禁止了在 ai 思考时进行悔棋，因为 ai 思考时进行悔棋设计难度上比较大，对于 ai 思考，这是后端剪枝和评估函数计算出来的，这里的中断对于前端不好处理，故我禁止了在 ai 思考时打断，所以禁止了在 ai 思考时悔棋，在按钮设计上也有体现。

2.3.2.2 点击事件的设计

之后最关键的设计部分便是点击事件造成的落子设计，这里通过计算将鼠标坐标转换成棋盘坐标。在这里还要判断无效点击。我在这里设计的是距离在规定的棋子这个圆内都算作有效点击，其他的范围属于无效点击。绘制棋子则由双方分别绘制，每次绘制完成后都检查是否达到胜利要求。每一方绘制完后进行交换。整个棋盘交换事件频繁，主要通过变量 is_black 来控制判断。Tkinter 也给出了用来进行消息循环的函数 mainloop()，便于在写好点击事件后反复去执行消息循环。不必再考虑宏观上的循环设计。在点击事件里 ai 下棋时我执行了按钮锁定，避免按钮所涉及的事件去破坏中断 ai 思考导致不可预见的 bug。在游戏进行时就通过 mainloop() 函数的消息循环来持续执行，直到游戏结束

2.3.2.3 胜利判断以及提取出的交换操作

胜利判断在这里理解也十分简单，在每落下的一颗棋子坐标下去查看该坐标衍生的四个方向，满不满足五子连珠的条件。这个胜利判断的搜索要比搜寻整个棋盘效率要高，只是棋盘的一部分，同时也要获取每次落棋的坐标。整个五子棋游戏进行时双方交换的频率十分高，大部分的功能都需要这样一个判断，于是我在这里将这个操作提取出来单独设计，主要就是判断当前 is_black 的状态来进行操作。同时 is_black 的状态也随着每次落子后函数 trans_identify()的操作来改变。这两个函数相互配合就解决了其他各个功能所需要的频繁交换。

2.4 其他创新内容或优化算法

评价函数的优化算法方面，原本是对整个棋盘进行评价，遍历的次数较大、耗时较长，后改进为只对四个方向，即上至下、左至右、右上至左下、左上至右下来进行评估，大大减少了遍历的次数，并将原本整个棋局的值用某点及其四个方向上形成的值进行替代。且原本评价函数中需要对传进来的数组进行复制（为了设置边缘方便评估棋局的方便），后也增添新的函数对位置进行判断后返回数组对应位置的值或者边缘的值，减少了复制整个棋盘所带来的消耗。

在 α-β 剪枝算法方面，为了实现对所有棋子 1-2 格进行考虑，同时又不必每次下一个棋子就遍历寻找可能的节点，程序引入了一个栈。本栈保证每次确定落子时，把周围所有未纳入栈的节点入栈，将他们都当作一个元素。同时在进行递归搜索时，每次向下一层，则在数组上记录，将可能节点入栈，回溯时则直接出栈还原。这样保证了算法的效率。

同时由于每次评价函数只考虑新节点的分数，但是在搜索中可能出现很多新节点，程序采用了动态规划的思想，对每次新下节点的记录，在回溯时则还原。这样在每次新确定一个节点时都调用一次评价函数，而不是只在底层调用，否则相邻节点只要一个子位置不同但全部分数都要重新算，显然浪费时间。使用了动态规划后，显著减少了计算量。

3 实验过程

3.1 环境说明：操作系统、开发语言、开发环境及具体版本、核心使用库等

操作系统：Windows10

开发语言：Python

开发环境及其版本：PyCharm 2022

核心使用库：Numpy、Tkinter

3.2 源代码文件清单，主要函数清单

evaluate.py	评价函数的设计与实现
gameAIcore.py	α-β 剪枝算法的设计与实现
main.py	前端 GUI 包括对内核的调用的实现

3.3 实验结果展示

开始界面如图 3.1 所示，点击左上角开始键开始下棋。

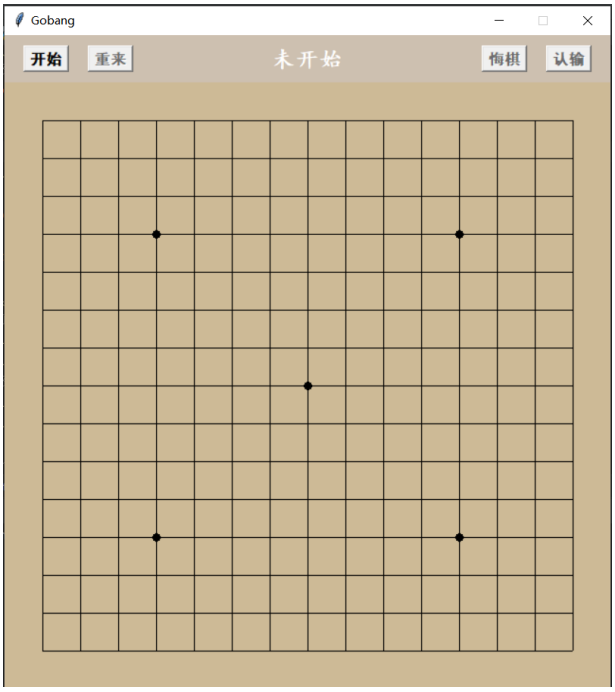

图 3.1

接着我方作为黑棋开始下棋，可以点击右上角悔棋键进行悔棋或认输键选择认输，也可以点击左上角重来键清空棋盘重开一局，如图 3.2 所示。

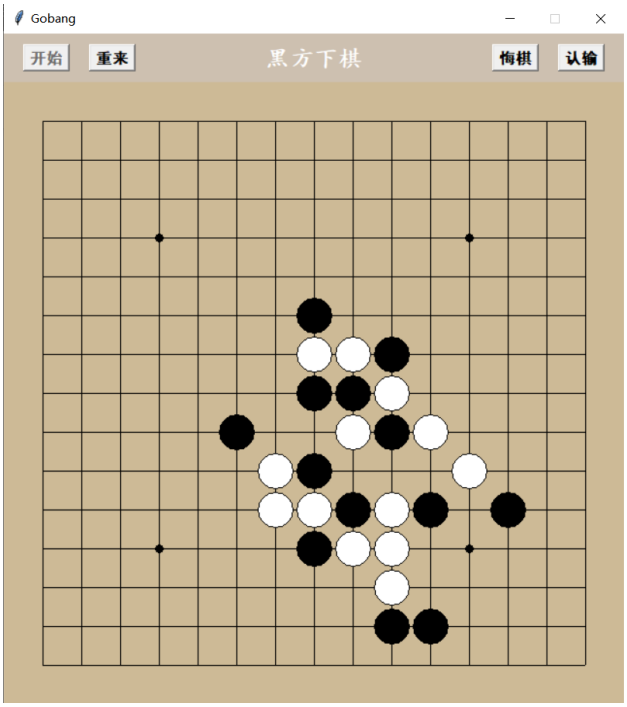

图 3.2

下棋至最后直到某方获胜，图 3.3 展示了白方获胜（电脑），可以选择左上角开始键重新来一局。

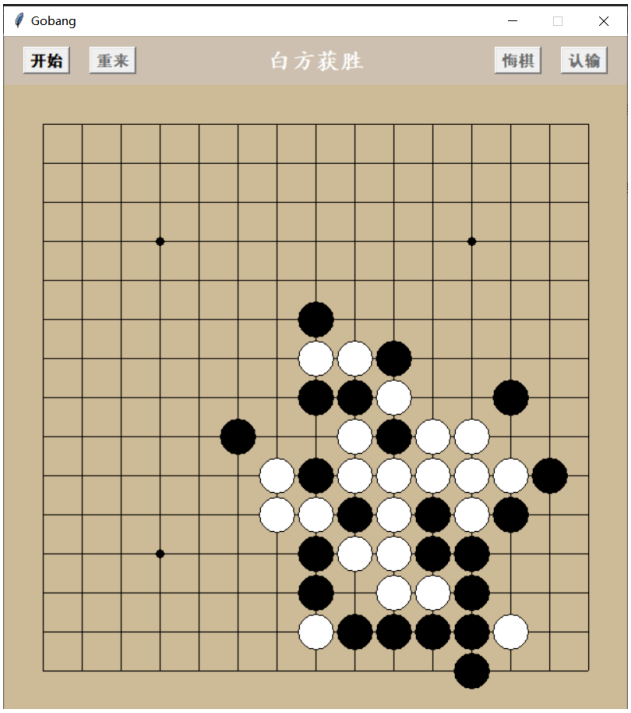

图 3.3

4 总结

4.1 实验中存在的问题及解决方案

4.1.1 实验中存在的问题：

1、在棋子棋盘格绘制时所出现的痕迹问题以及计算问题

2、功能组件的设计与布置

3、点击事件人类和电脑双方落子的组织

4、设计评价函数时对于权值的考虑

5、在设计 α-β 剪枝算法时如何不遍历寻找每次的可能点

4.1.2 解决的方法：

1、为了后续在悔棋时方便撤销棋子，在棋盘绘制时我是采用每个棋子坐标处单独绘制，初始化时直接整个循环绘制。这样单独绘制时可以再次利用到悔棋上面。但是在我设计好后发现，每次悔棋撤销棋子时总会留下一点点痕迹，有点棋子“撤销不干净”的感觉。查询资料后发现这里应该是倍率问题，按照原样 1 的倍率总会出现这种情况，将倍率稍微增加一点就能够避免这种情况。再就是坐标计算问题是一个很大的麻烦。尤其是在画网格线时，在这里我尝试了多次，添加了不同的系数以及计算方式，最后才做到画出来比较自然，每个单独画的网格看起来不生硬。

2、功能组件的设计在制作时也遇到了一些麻烦。四个按钮功能组件问题最大的就是悔棋功能。一是涉及到棋子擦除以及网格重画的问题，二是在棋盘数据的存储上也遇到了些麻烦。第一个问题在上面也提到了，倍率问题的影响在我查询资料后得以解决。第二个问题便是数据存储问题。在这里每次下棋时都单独记录了双方上一步棋子的坐标，通过这个数据传入做到擦除棋子同时在棋盘存储中进行修改。由于只记录了最近一步的棋子的坐标，故每次悔棋只能悔棋一步。悔完一步后再次点击悔棋按钮会提示现在不能悔棋。同时防止中断 ai 思考所出现不可预见的 bug，在这里我禁止了在 ai 思考时进行悔棋。

3、点击事件是五子棋 GUI 设计中的重中之重，毕竟人类一方每次落棋都要进行鼠标点击。这个存在着一个鼠标坐标到棋子坐标的转换，同时还要根据鼠标坐标来判断此次点击是有效点击还是无效点击。毕竟不是每次点击都正好是点击中了位置来落棋。在这里我设计的范围便是棋子圆形范围的大小。对于事件循环 Tkinter 库则提供了比较好用的事件循环函数 mainloop，使得我不必再考虑设计事件循环问题，只需要调用这个封装好了的函数。

4、需要考虑多方面的问题，例如权值大小的设计，不同持方棋型估值的不同，同一方棋局上的不同棋型的倍数，需要在网络上借鉴大量的资料以及对五子棋规则的理解设计出一套可以使用权值。

5、引入一个栈。本栈保证每次确定落子时，把周围所有未纳入栈的节点入栈，将他们都当作一个元素。同时在进行递归搜索时，每次向下一层，则在数组上记录，将可能节点入栈，回溯时则直接出栈还原，每次递归的过程中自动维护栈即可。

4.2 心得体会

评价函数方面对于评估结点权值的方式、对棋局的权值的设计相对比较繁琐，此外在遍历某个位置四个方向这个问题上需要考虑复杂的边界问题，最终对五子棋评价函数或者说评价函数在启发式搜索中的理解进一步加深。

在实现 α-β 剪枝算法时，我学习了 α-β 剪枝算法的实现，对于 α-β 剪枝算法的运行原理和运行方法有了极大的了解，对对抗搜索的内涵有了更为深入的理解。同时我还使用了动态规划思想和回溯思想，对我算法的设计能力也有了很大的帮助。

GUI 方面通过本次的学习我对图形库 Tkinter 有了一定的了解，能够运用 Tkinter 库做一点简单的东西。同时在设计过程中我学到了各个事件的组织方式。能够帮助我在后面的学习中对于该种设计能够更加属性与上手。

4.3 后续改进方向

可以增加选择先后手，标记最新落子的位置，增添游戏音效等功能。可以在之后增加难度设置，加大 AI 考虑的广度和深度。

4.4 总结

经我们小组成员几个星期的实验与努力，终于及时完成了本次的五子棋的设计与实现，成功的实现了游戏的开始，游戏的其他功能等问题。不过由于时间短暂，所以整个设计仍然存在一些问题与不足，如游戏无法选择先后手等，不过仍还是有一些体会的。

在此次设计中，我意识到把理论知识转化成实际操作的重要性，小组成员深刻认识到了博弈树的搜索，熟悉并实现了 α-β 剪枝算法，为五子棋设计了合适的评估函数等等。另外为了完成这次试验，小组成员还需要了解课本外的知识，比如在此次设计中，若想要实现人机对战就必须先实现五子棋的可视化界面，棋子，制定棋局规则，判断输赢等功能。有时候当你遇到你无法解决的困难时，需要打破常规方法，另辟蹊径，多角度尝试，总会有方法实现的。

另外在小组合作方面，我们事先就搭建实验框架与模板，划分了实验任务，对接好函数接口，尽量避免了实验的重复与疏漏，小组成员各司其职，在实验过程中积极地充分地交流，顺利地实现了实验要求的内容。

成员分工与自评

秦天：编写评价函数这部分的程序，寻找五子棋对应的评估方式，填写对应的实验报告，制作汇报 ppt，分工平均，同时也学到了很多，加深理解的同时感觉很有帮助。

张睿：在这次实验中，我负责编写了 α-β 剪枝算法，作为游戏的 AI 核心，提出了每次只考虑单一棋子的分数的思想，极大优化了程序的性能。在完成实验之后，填写了对应的实验报告。我对这次实验非常满意，学习到了很多上课的知识，加深了理解，分工平均。

孙颖斌：在本次实验中，我负责编写五子棋 GUI 界面展示与交互。为了能使五子棋有着更好的呈现，学习 Tkinter 库来完成界面交互，响应事件。并且也学到了如何使用好图形库

Tkinter。在完成实验之后，填写了对应的实验报告。这次实验让我学到了很多，对于整个实验流程加深了理解，分工平均。