

基于Alpha-Beta剪枝的三维空间四子棋系统

陈昊天¹, 刘晓彤^{1,2}, 陈冰洋¹

(1. 北京信息科技大学 计算机学院; 2. 北京信息科技大学 北京市材料基因工程高精尖创新中心, 北京 100101)

摘要: 棋类游戏研究已经从传统的二维空间扩展到三维空间, 三维空间的引入增加了游戏的复杂性和策略深度。针对三维空间四子棋的立体结构和规则, 应用Alpha-Beta剪枝算法可以开发一个高效的三维空间四子棋系统, 使用遗传算法等方法优化棋形得分进一步提高系统智能性, 并通过KANs分析结果。该系统设计了具有良好交互性和游戏性的3D界面, 使用户能够从不同角度观察棋盘。在系统构建过程中, 设计了适应三维棋盘的数据结构和局面评估函数, 以确保AI能够高效地进行决策和对战。该系统支持多种游戏模式, 包括不同难度的AI对战和本地双人对战。此外, 系统提供对局历史记录回放, 用户可以通过该功能进行棋局复盘和数据分析。对局结果数据显示, AI在不同难度下的表现与计客智能四子棋系统的难度相对应, 能够有效适应各类用户的对局需求。

关键词: 三维空间四子棋; Alpha-Beta剪枝; 遗传算法; KANs

DOI: 10.11907/rjtk.241575

开放科学(资源服务)标识码(OSID):

中图分类号: TP18

文献标识码: A

文章编号: 1672-7800(2025)008-0137-08



Three-dimensional Connect Four System Based on Alpha-Beta Pruning

CHEN Haotian¹, LIU Xiaotong^{1,2}, CHEN Bingyang¹

(1. School of Computer, Beijing Information Science and Technology University; 2. Beijing Advanced Innovation Center for Materials Genome Engineering, Beijing Information Science and Technology University, Beijing 100101, China)

Abstract: The study of board games has expanded from the traditional two-dimensional space to three-dimensional space, and the introduction of three-dimensional space increases the complexity and strategic depth of the game. For the three-dimensional structure and rules, an efficient three-dimensional space connect four system can be developed by applying the Alpha-Beta pruning algorithm, and the system intelligence can be further improved by optimizing the scores of the chess shapes using genetic algorithms and other methods, and by using KANs to analyze the results. The system is designed with a 3D interface with good interactivity and gameplay, enabling users to observe the board from different angles. During the construction of the system, data structures and position evaluation functions adapted to the 3D chess board are designed to ensure that the AI can make decisions and play against each other efficiently. The system supports a variety of game modes, including AI matches of different difficulties and local two-player matches. In addition, the system provides game history playback, which allows users to review the game and analyze the data. The game result data shows that the performance of AI in different difficulty levels corresponds to the difficulty level of the GiiKER® Smart Four, which can effectively adapt to the game needs of various types of users.

Key Words: three-dimensional space connect four; Alpha-Beta pruning; genetic algorithm; KANs

0 引言

传统的二维棋类游戏已得到广泛研究, 随着对游戏体

验的不断追求, 人们开始探索将棋类游戏拓展到更高维度的空间。三维空间四子棋是将经典四子棋游戏拓展到三维空间的创新之作^[1]。在传统的二维四子棋游戏中, 玩家只能在平面上进行对战, 而三维空间四子棋则将战场扩展

收稿日期: 2024-08-13

扫描二维码阅读全文:



基金项目: 北京信息科技大学科学研究基金项目(2022XJJ21)

作者简介: 陈昊天(2000-), 男, 北京信息科技大学计算机学院硕士研究生, 研究方向为材料化学领域机器学习与理论计算; 刘晓彤(1987-), 男, 博士, 北京信息科技大学计算机学院副教授、硕士生导师, 研究方向为材料化学领域机器学习与理论计算; 陈冰洋(2002-), 男, 北京信息科技大学计算机学院学生, 研究方向为材料化学领域机器学习与理论计算。本文通讯作者: 刘晓彤。

到了立体空间,为游戏带来了更多的棋局变化状态和可能性。经过调研,市面上只有计客推出的实体版三维空间四子棋,并没有 PC 端资源,虽然移动端有类似软件,但当进行人工智能(Artificial Intelligence, AI)对战时,必须使用相应品牌配套的实体四子棋设备。为了弥补在线三维空间四子棋缺少的资源,让更多人参与到四子棋对战中,本文设计了三维空间四子棋系统。

三维空间四子棋是一种较为新颖的棋类游戏,相比传统的二维棋类游戏,如五子棋和国际象棋,其流行程度相对较低。该游戏的棋盘为一个 $5\times 5\times 5$ 的立方体,每个维度各有 5 个格子,总共包含 125 个位置。玩家轮流在棋盘上放置棋子,每次只能在一个空位上放置一个棋子。游戏的目标是率先在任意直线(包括水平、垂直、对角线等方向)上连接 4 个连续的棋子。由于棋盘在三维空间内展开,棋子的排列组合方式比二维棋类游戏更为复杂,需要考虑更多的进攻和防守线路。与五子棋和国际象棋等传统二维棋类游戏相比,其最大的区别在于状态空间的不同,状态空间的增加使得搜索空间变得更加庞大。二维棋类常用的搜索算法同样适用于三维空间四子棋,但由于状态空间更大,三维四子棋的搜索深度通常会受到一定限制。

三维空间带来的棋局复杂性对算法设计和优化提出了更多挑战。MiniMax 算法是一种经典的博弈树搜索算法,用于确定最优决策策略。该算法递归地搜索整个博弈树,交替地考虑玩家和对手的行动,假设对手采取最优行动,选择能够最大化自己收益的行动。但随着搜索深度的增加,该算法的时间和空间复杂度呈指数增加,这对 AI 对战的用户体验并不友好。与传统的 MiniMax 算法相比,Alpha-Beta 算法通过剪枝策略以减少搜索树的分支,从而提高搜索效率^[2]。在三维空间四子棋系统中,Alpha-Beta 算法的应用可以显著降低计算复杂度,使得系统能够更快地找到最优解^[3]。算法采用启发式评估函数,针对不同难度设计了相应的评估算法^[4]。为进一步优化评估函数中的参数值,在原本基础上使用遗传算法和神经网络两种方法优化,对不同棋形的得分进一步细分调整,以优化 AI 的落子选择,并分析不同棋形得分对结果的影响。

本文基于 Vue、Three.js^[5]、Spring Boot 框架^[6],实现了一个应用 Alpha-Beta 剪枝算法的三维空间四子棋系统,同时设计了合适的用户交互界面和交互方式。该系统能够根据当前棋局使用 Alpha-Beta 剪枝算法自动搜索选择最优落子,并根据当前棋局情况自动判定胜负。经过测试,该系统能够战胜市面上对应强度的实体游戏 AI,具有较高的智能性和一定的挑战性。

1 系统设计

1.1 需求分析

三维空间四子棋系统的服务对象为棋类爱好者,包括

希望提升棋类水平以及与 AI 或朋友对局的玩家。低成本快捷的线上体验,能吸引更多用户,并解决实体棋盘的不便,如实体棋盘的智能 AI 棋手只能提示落子位置,并不能直接落子。此外,对于想观看历史棋局以提高水平的玩家,实体棋盘缺少对棋局的合理记录,为用户体验带来诸多不便,故系统在此基础上实现可以观看棋手对弈的棋局过程,包括 AI 棋手和真人棋手。其他线上棋类游戏多为二维空间展示,缺少三维立体空间的直观操作,三维空间四子棋系统旨在设计并实现一个为棋类爱好者提供多种难度博弈和复盘学习的便捷服务,实现三维空间中的游戏规则及博弈算法^[7],在直观的三维界面中从不同的角度观察棋局。因此,需设计不同难度的棋局并完成 AI 棋手的落子,以及自动判定在三维空间中的胜利条件,尽可能提供良好的交互方式以完成对弈及复盘记录。

1.2 架构设计

三维空间四子棋系统架构设计如图 1 所示,分为展示层、业务层、数据层。前端主要采用 JavaScript 技术,使用基于 Vue 的 Element Plus 组件库实现页面,利用 Three.js 进行游戏界面渲染,可直接运行 GPU 驱动游戏与图形驱动并应用于浏览器^[8]。WebSocket 用于实现前后端游戏双向交互,在单个 TCP 连接上进行全双工通信,连接建立后便保持当前状态,数据可在任意时间进行客户端与服务端的数据推送^[9]。RESTful API 用于前后端业务功能通信。后端基于 Spring Boot 框架实现,主要负责游戏规则验证、管理玩家操作、处理 AI 操作、记录对局信息、游戏状态更新等功能开发。使用 Mybatis-Plus 进行数据访问,并使用 MySQL 作为数据存储,用于重现历史棋局。

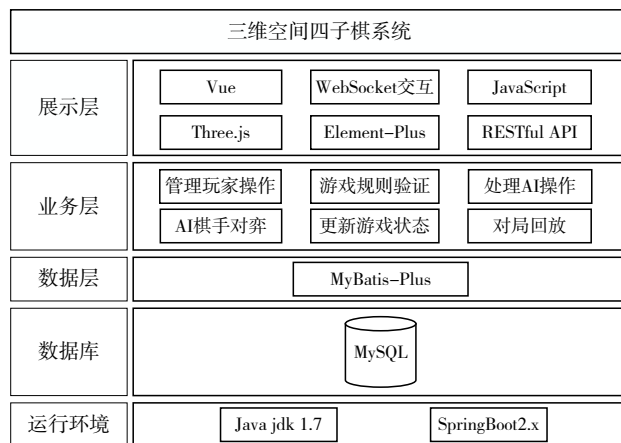


Fig. 1 System architecture design

图1 系统架构设计

2 关键技术

2.1 MiniMax 搜索算法

MiniMax 算法是一种常用于博弈游戏的决策算法,适用于零和博弈中^[10]。在这种游戏中,玩家会选择在每一步

选择对自己利益最大的动作,并同时考虑对手会选择对自己利益最小的动作。将双方的决策过程视作一棵决策树,其中己方选择对己方收益最大的路径,该层称为 MAX 层;对手选择对己方收益最小的路径,该层称之为 MIN 层。

算法通过构建决策树对当前状态进行建模,如图 2 所示。在搜索状态到达叶子节点时,使用评估函数对当前局面进行评估,以确定该状态的得分。算法自底向上计算 MiniMax 值:奇数层为 MAX 层,即己方的选择动作,会选择对己方利益最大的子节点;偶数层为 MIN 层,模拟对手选择对己方利益最小的动作,即选择对己方利益最小的节点。叶子节点的得分是根据评估函数评估当前局面的结果。根节点根据计算得到的 MiniMax 值选择最大的分支作为最终行动策略。这一过程确保在整棵决策树中,每一步选择都既考虑到己方的最大利益,又预见到对手的最佳反应。

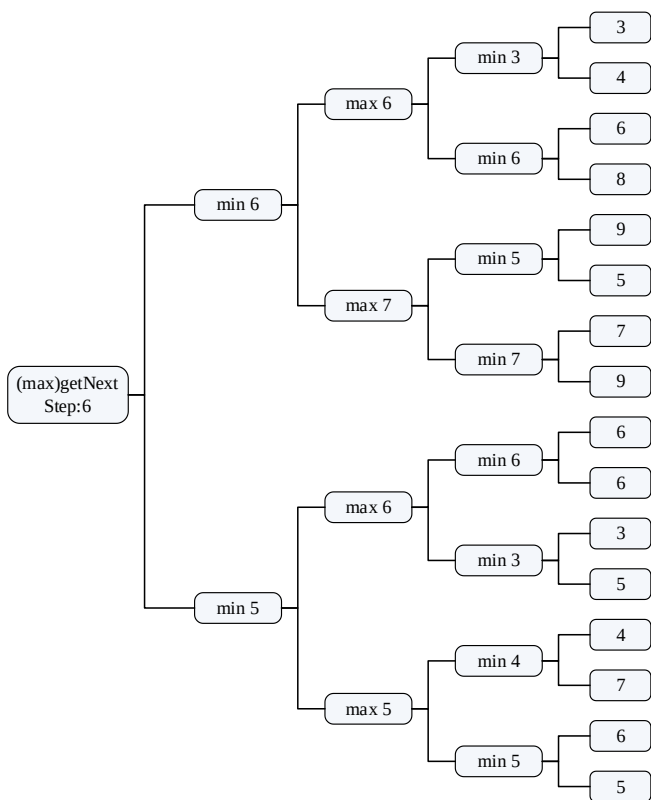


Fig. 2 Decision tree modeling

图 2 决策树建模

2.2 Alpha-Beta 剪枝算法

Alpha-Beta 剪枝算法是一种用于优化固定深度的 MiniMax 算法的技术^[1],通过剪除对决策结果无用的路径从而提高搜索效率^[11-12]。MiniMax 算法在搜索过程中会产生大量无效搜索,其中存在两种明显冗余:极大值冗余(见图 3)和极小值冗余。

Alpha-Beta 剪枝算法首先维护两个值 alpha 和 beta,分别表示当前节点值的下界和上界,初始时 alpha 为 $-\infty$,beta 为 $+\infty$ 。在搜索过程中,MAX 节点(己方)尝试找到所有子节点中的最大值并更新 alpha,MIN 节点(敌方)尝试找到所

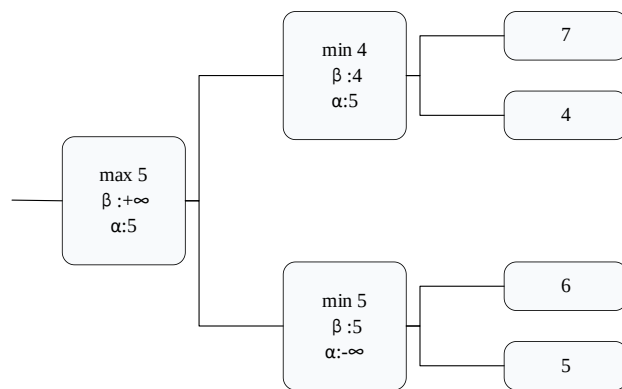


Fig. 3 Maximal value redundancy

图 3 极大值冗余

有子节点中的最小值并更新 beta。对于 MAX 层节点,当搜索到某个节点时,如果当前值大于等于 beta,可以停止搜索该节点的子树,因为上层 MIN 节点不会选择它;对于 MIN 层节点,当搜索到某个节点时,如果当前值小于等于 alpha,可以停止搜索该节点的子树,因为上层 MAX 节点不会选择它。算法通过递归搜索子节点,根据节点的值更新 alpha 和 beta,在搜索到叶子节点时使用评估函数计算当前局面的值。通过这种方式,Alpha-Beta 剪枝显著提高了 MiniMax 算法的搜索效率。

2.3 遗传算法

遗传算法(Genetic Algorithm, GA)是一种基于自然选择和遗传机制的搜索优化算法^[13]。它模拟了达尔文的生物进化过程,通过选择、交叉、变异等操作不断优化问题的解。遗传算法在全局优化问题上表现出色,适用于复杂、多峰值、不规则的搜索空间。具有较高适应度的解在下一代中存活并产生后代的概率较大,有效探索不同参数组合,从而寻找合适的参数组合。

遗传算法基本步骤:根据设定的种群大小 N 随机生成一组初始解,每个解由若干个参数组成。用适应度函数计算每个个体的适应度值,适应度值反映个体解的优劣,适应度高的个体具有更好的解。根据适应度值,选择出用于生成下一代的两个个体,个体 x_i 被选择的概率 P_i 和适应度值 $f(x_i)$ 的关系如式(1)所示。

$$P_i = \frac{f(x_i)}{\sum_{j=1}^N f(x_j)} \quad (1)$$

生成新的子代个体 x_{new} 时,进行交叉操作,将选择的两个个体 x_i 和 x_j 的交叉点 k 进行交换以增加解的多样性,后代 x_{new} 可表示如式(2)所示。

$$x_{new} = (x_{i1}, x_{i2}, \dots, x_{ik}, x_{j(k+1)}, \dots, x_{jn}) \quad (2)$$

通过变异操作引入随机性,以一个小的概率将子代个体的某些参数进行正态分布变化量 δ 的变异,以防止算法陷入局部最优。对于基因 X_{nk} ,突变操作可表示如式(3)所示。

$$X'_{nk} = X_{nk} + \delta \quad (3)$$

在达到终止条件前,不断进化种群中的个体以逼近最优解。

2.4 KANs神经网络架构

Kolmogorov-Arnold Networks (KANs)^[14]是一种新型神经网络架构,其灵感来源于Kolmogorov-Arnold表示定理。根据这一定理,任意多变量连续函数 $f(x)$ 可以表示为一系列单变量函数的组合,可表示如式(4)所示。

$$f(x) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right) \quad (4)$$

其中, x_p 代表向量 x 的第 p 个元素,即 p 的范围为 $1 \sim n$ (n 是输入向量的维度)。 Φ_q 和 $\phi_{q,p}$ 均为单变量函数。

与传统的多层感知机不同,KANs在网络的边缘采用可学习的激活函数,这种设计可显著增强模型的可解释性。在KANs中,权重参数被一种B样条函数取代,从而函数可以直接连接两个神经元,代替了多层感知机中的线性权重,使每条连接的激活函数是可学习的单变量函数。KANs减少了参数数量,使得模型更具可解释性,故可通过观察这些可学习函数的形式和变化,分析输入特征对输出结果的影响。

3 系统实现

3.1 三维空间四子棋功能模块实现

3.1.1 棋盘实现

棋盘由一个二维棋盘格子数组组成,每个棋格包含一个整形数组,用于表示放置棋子的空间。棋盘属性包括棋盘长度、宽度、高度、当前棋子颜色以及启发式评估函数计算方法。对局的难易程度(AI的智能水平)决定了评估函数的不同定义方法,棋格属性用于表示当前棋子高度。默认情况下,棋盘大小为 $5 \times 5 \times 5$,棋盘对象维护一个保存棋子位置的栈,栈帧记录棋局的落子顺序。棋局记录由序列化棋盘的栈进行存储。棋盘落子时,判断落子坐标是否合法并记录。棋局结束时,通过当前棋子颜色属性确定游戏结束时的获胜玩家。

3.1.2 落子流程实现

当尝试落子时,对落子后的局面进行评估,并根据评估值选择最有利的落子位置。评估函数 $f(x)$ 根据局势优劣特征进行定义,对AI方有利的局势 $f(x)$ 返回值为正,对玩家方有利的局势 $f(x)$ 返回值为负。当 $f(x)$ 为 $+\infty$ 表示AI方获胜,当 $f(x)$ 为 $-\infty$ 表示玩家方获胜。评估函数由用户所选的难易程度决定。

根据游戏规则,棋子越靠近中心,赢的机会相对较大,且先手一般落子于中心位置。因此,评估函数对棋盘每个格子设置了不同的初始权重,并设计了两种不同的评估方法。第一种是简单难度下的评估函数,三维空间四子棋和传统的五子棋一样,同一方向连子越多表示该棋形得分越高^[15]。相较于三维空间四子棋的局面情况,五子棋的局面

较为复杂,可以细分为长连、五连、活四、冲四、活三、眠三、活二、眠二这8种情况^[16]。当棋子连成4子时,游戏结束。因此,当判断棋盘中出现四子相连,直接返回 $+\infty$ 或 $-\infty$;当棋子连成3子时,相当于五子棋中的连成四子,若不堵截,下回合即可连成四子,游戏结束。因此,将三连子棋形价值设置为一个较高的值,使得程序会尽可能地“连三”且避免对方“连三”。但这种评估方法会导致在无法决出胜负时盲目追求“连三”,导致损失局面。第二种是困难难度下的评估函数,在基于连子的情况下,将各种情况下的棋形分类,并给出更加细化的分数,将同一行中的空位也考虑进去,不同的棋形所代表的分数不同,从而避免AI盲目追求“连三”。

落子的选择基于Alpha-Beta剪枝,使用启发式评估函数,维护最大搜索深度为4。该方法遍历棋盘每个格子并尝试落子,落子成功后递归进行MiniMax搜索并计算最好得分,递归结束后撤销落子,还原棋局,更新最好得分及落子点位。进行MiniMax搜索前,首先判断当前游戏是否已经结束,若游戏结束则返回胜者信息,再判断当前搜索深度是否等于最大搜索深度。若达到搜索叶子节点时,返回当前局面评估结果;若游戏未结束且搜索未超过最大搜索深度,判断当前搜索层是否为MAX层。若是,则遍历棋盘进行模拟落子,递归进行MiniMax搜索并计算最好得分,递归结束后撤销落子并更新alpha为最高得分下界;若出现 $\beta \leq \alpha$,则表示出现了极大值冗余,需进行剪枝,不再搜索后续子节点。同理,若当前层为MIN层,递归结束后更新beta为最高得分上界;若出现 $\beta \leq \alpha$,则表示出现了极小值冗余,需进行剪枝。

3.1.3 评估函数设计

局面评估分为棋子位置评估和棋形评估。棋子位置评估是根据棋盘不同位置决定,不同位置权重不同,越靠近棋盘中心得分越高,反之越低。棋形评估是计算未被遍历过的棋子连线得分,不同棋局难度得分的计算方法不同,分为连子数得分和局面控制得分。计算时需要对待评估棋子的水平方向、竖直方向、垂直方向、主对角线方向、次对角线方向、水平上升梯度方向、水平下降梯度方向、竖直上升梯度方向、竖直下降梯度方向、主对角线上升梯度方向、主对角线下降梯度方向、次对角线上升梯度方向、次对角线下降梯度方向分别计算连子数得分总和与局面控制得分总和。

连子数计算时,先初始连子数为1,分别向变化量方向正向(加变化量)和负向(减变化量)进行访问。当访问位置合法且访问棋子与评估棋子颜色相同时,连子数+1;当返回连子数为2、3、大于等于4时,得分分别为400、100 000、10 000 000。

局面控制计算方法为先初始局面控制棋形值为1.0,即只有一个棋子,分别向变化量方向正向(加变化量)和负向(减变化量)进行访问。当高度变化量不为0时,表示当

前计算行为梯度计算行, 棋形值为 1.0; 当访问位置合法且访问位置棋子颜色与评估棋子颜色相同, 并且访问位置高度与初始位置高度相同时, 棋形值增大 1。若访问位置为空即棋子颜色为 0, 则棋形值加 0.1, 正向和负向只计算一次空位, 即棋形值最多加 0.2。当棋的某一侧存在一个空位时, 棋形加 0.1, 即为棋形 2.1, 如图 4(a) 所示; 当存在两个空位时为棋形 2.2, 如图 4(b) 所示, 只考虑第一个遇到的空位, 若棋形边界第一个格子不是空位, 则不考虑。由此可得出棋形 2、3、2.1、3.1、2.2、3.2、4, 对应棋形得分分别为 100、300、310、600、800、10 000、10 000 000。最终的局面评估分为己方棋子位置得分和棋形得分与对方得分之差。

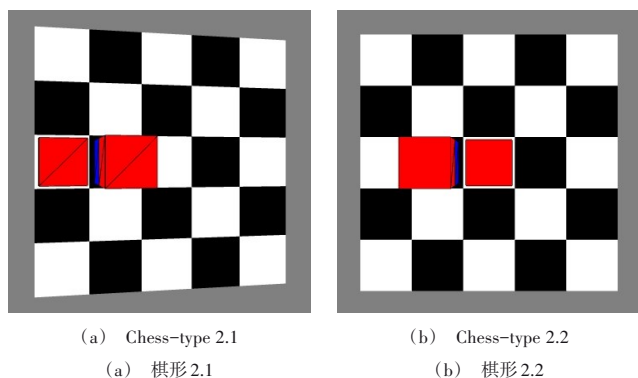


Fig. 4 Chess-type classification

图 4 棋形分类示例

3.1.4 基于遗传算法的棋形得分优化

困难模式下, 不同棋形得分初始值还有进一步优化空间。为进一步提高 AI 棋手智能性, 需对棋形得分作更进一步优化。通过在种群内让两个 AI 棋手之间互相对弈的方法^[17], 计算每个个体的胜利次数并将其当作适应度值, 胜场次数高的个体适应度高, 使用正态分布的方式进行变异。基于棋形得分的数值选择范围较广, 测试两种不同的优化策略, 在父母选择策略、交叉方式上有所不同。第一种策略使用轮盘赌选择的策略选择父母, 采用单点交叉的方式; 第二种策略, 在父母选择策略上采用了锦标赛选择法, 随机选择若干个体进行小规模竞争, 选择适应度最高的个体作为父母, 交叉方式为双点交叉。此外, 采用动态调整变异率的机制, 根据适应度停滞情况调整变异率。第二种策略通过与原始数据进行对弈测试的效果较第一种好, 在种群大小为 20, 最大迭代次数为 6, 初始变异率为 0.5 下进行优化, 对应棋形为 2、2.1、2.2、3、3.1、3.2、4, 结果为 161 773.54、73 514.74、222 892.68、787 114.43、10 513.23、452 609.91、549 304.48。受迭代时间影响, 迭代次数和种群数量只能以较小的值进行优化, 故让初始种群中包含初始值以保证优化质量, 但子代的多样性依然不足。为了在一定程度上解决多样性不足的问题, 使用神经网络加以优化。

3.1.5 基于神经网络的棋形得分优化

鉴于 AI 棋手对弈时间上的考虑, 无法在适当时间内

对更高的迭代数进行优化, 故采用神经网络的方式拟合在种群内的胜场次数, 同时最大化模型输出值, 进而找到对应的棋形得分。训练集为种群大小为 50 进行初始化的数据, 个体 AI 棋手的胜场次数为模型真实值。由于随机初始的棋形得分范围较广, 为避免训练数据个体特征差异过大而影响模型收敛效果, 对训练数据进行数据标准化预处理, 并将训练集的 10% 划分为验证集, 使用早停的方法以解决数据量较少容易过拟合的问题, 模型以 MSE 作为损失函数进行优化。以 batch size=8, max epoch=100 进行训练, 当验证损失在至少 20 个 epoch 内没有改善时, 训练会提前停止。网络结构如图 5 所示, 模型将输入数据通过全连接层将原始特征映射到高维空间, 通过门控循环单元 (GRU) 层, 并使用多头注意力机制进一步处理 GRU 的输出, 通过前馈神经网络层生成模型预测结果^[18-19]。

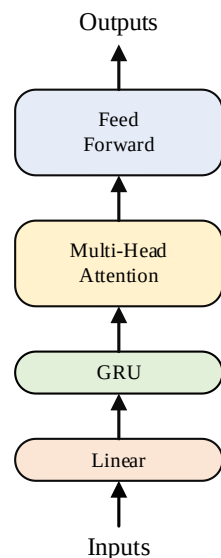


Fig. 5 Model architecture

图 5 模型结构

为了使 7 种棋形对应得分带来的胜利场数尽可能多, 需要优化输入数据, 以使神经网络模型的输出最大化。参数初始为形状为 (1, 7) 的张量, 通过 Adam 优化器在学习率为 $1e-3$, 优化步骤为 200 次时调整输入数据, 并在最后逆转换到原始数据的尺度。整体优化流程如图 6 所示, 优化得出对应棋形为 2、2.1、2.2、3、3.1、3.2、4 的得分为 170 134.23、200 505.81、368 450.7、247 526.4、335 117.06、352 046.38、4 294 718.5, 通过 AI 对弈的方式, 验证结果得出优化后的结果优于原始初始棋形得分。

3.1.6 棋形得分分析

本文利用 KANs 的可解释性, 解释三维空间四子棋棋形得分与胜利次数之间的关系, 以期后续三维空间四子棋策略优化提供一定的理论依据。将棋形的 7 种不同类型的得分作为 KANs 的输入, 以拟合胜利次数, 从而建立棋形得分与胜利次数之间的映射关系。经过训练的模型, 可在一定程度上揭示出各棋形得分在最终胜利次数中的相

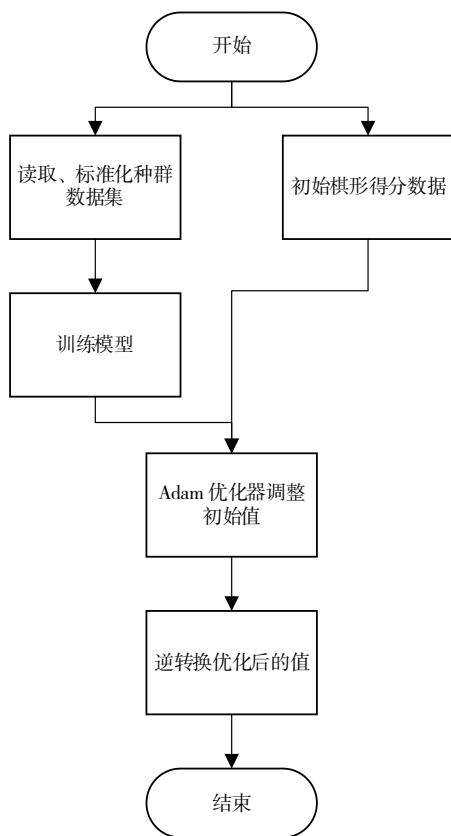


Fig. 6 Optimizing chess-type scores flow

图6 优化棋形得分流程

对重要性和影响。训练后,当前种群中的棋形得分与胜利次数的映射关系如式(5)所示。

$$(0.8092x_1 + 1.5420x_2 + 1.3905x_3 + 2.4042x_4 + 1.2258x_5 + 1.2307x_6 + 0.1542x_7) \times 10^{-5} - 2.0231 + 2.4886 \times e^{(-0.5776 \times (0.0001x_1 - 1)^2)} \quad (5)$$

其中, $x_1 \sim x_7$ 分别代表7种不同棋形的对应得分,常数项用于确保得分的合适范围。在线性部分,根据系数关系可知,第4种棋形在当前种群中对胜利次数的影响较大。第7种棋形得分在样本中已经较高,故系数最小。指数修正项只与第一种棋形得分相关,当第一种棋形得分接近10 000时,指数项接近1,修正项达到最大值,反之修正项趋近于0,但第一种棋形得分对整体胜利次数的影响依旧较小。可以发现,在当前实验中如果更多地关注参数 x_4 ,则更可能提升胜利次数,从而使系统智能性达到更好的效果。

3.2 展示模块实现

系统为三维空间四子棋,需要玩家能够从各角度观察棋盘,能够看到各棋子的位置,故选用Three.js库创建和渲染棋盘、棋子的3D模型。系统页面分为主页、登录、游戏页面、历史记录页面和游戏回放页面。

3.2.1 游戏页面设计

游戏页面中心为一个立体棋盘,如图7(a)所示(彩图扫OSID码可见)。棋盘大小为5×5的黑白棋盘,棋子为红

蓝棋子,先手为红色棋子,后手为蓝色棋子。可拖动以旋转棋盘进行观察,点击棋盘格子进行落子,如图7(b)所示。

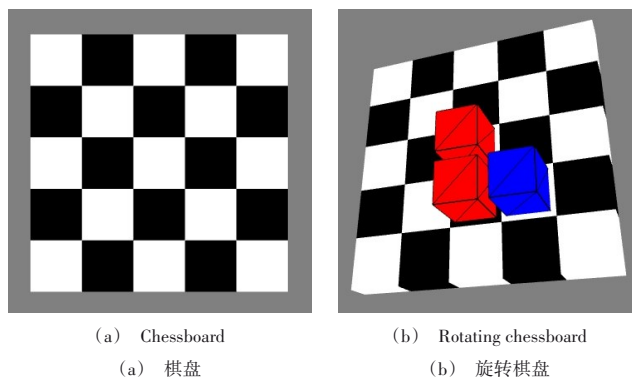


Fig. 7 Chessboard display

图7 棋盘展示

3.2.2 对战界面实现

当选择在线对战时,用户登录后进行难度选择,并将用户信息和难度存入localStorage,进入对应难度的在线对战游戏界面。在mounted生命周期进行WebSocket连接,定义游戏场景scene,游戏相机camera并初始化相机位置,将相机加入场景中,创建控制器并加入场景中。创建游戏棋盘,初始化棋盘样式、大小并加入场景中。创建渲染器renderer,进行场景渲染。在3D棋盘,监听鼠标点击事件,根据点击位置及摄像机创建一条射线,获取射线与棋盘相交的格子。判断当前格子中棋子数量是否超过上限,若小于上限,根据当前落子方创建对应的棋子对象,设置棋子将棋子落入对应格子。玩家点击落子后,使用WebSocket将落子信息发送给后端,并等待后端传回AI落子信息后继续游戏。当某一方连成四子后游戏结束,无法再进行落子操作。落子逻辑是由AI先手,玩家后手。当进入游戏页面时,进行WebSocket连接,当建立连接时,后端发送第一步落子。每次落子后切换当前玩家属性,只有当玩家属性为当前玩家时,棋盘的点击操作才视为有效操作。同理,本地双人对战,区别在于不需要选择难度,本地对战双方都是玩家主机进行点击操作,每次落子后切换玩家进行落子。

3.2.3 对局回放实现

历史记录页面可查看用户所有的历史对局(见图8),需要用户登录后方可查看。用户进入历史记录页面时,在mounted周期使用axios调用/moves/gameMoves接口将返回的数据绑定到el-table表单上。同时,定义dateFormat方法,将日期数据从ISO 8601格式转换为YYYY-MM-DD hh:mm:ss格式进行显示。可查看历史对局信息,包括游戏开始结束时间、游戏难度、游戏获胜方、游戏是否结束等待。查看详情后会将对应局的游戏落子数据发送给对局回放页面进行游戏复盘。

对战回放页面和在线对战页面类似如图9所示,由一



Fig. 8 History interface
图 8 历史记录页面

个 3D 棋盘组成, 用户无法点击棋盘进行落子。进入本页面后会将本次对局的对局数据进行解析, 通过控制可以重新加载对应步数的落子结果, 以及撤销落子。通过将接收的字符串数据解析为游戏动作数组 moveData, 并维护棋子数组和 moveData, 进行落子、撤销落子操作。

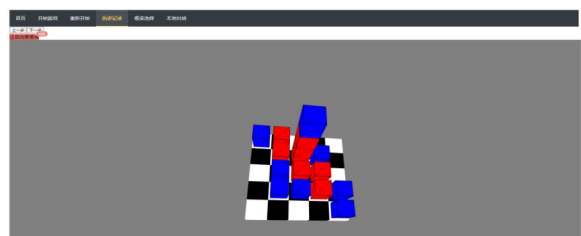


Fig. 9 Example of historical chess games
图 9 历史棋局示例

4 系统测试

4.1 基础功能测试

基础功能测试主要包括如下两方面:

(1) 三维空间四子棋规则判定测试。对系统中落子位置、胜负判定等功能进行测试, 表 1 对规则判定的测试用例进行了详细说明。经过测试得出, 系统规则判定功能良好, 可自动判定落子合法性和游戏结果。

Table 1 Test cases for determining the rules of connect four in three-dimensional space

表 1 三维空间四子棋规则判定测试用例

序号	测试内容	预期结果	运行结果
1	棋子限高测试	同一个棋盘格放置五颗棋子后再次点击棋盘格无反应	与预期结果一致
2	水平四连判断	水平方向出现相同颜色四子连珠, 游戏结束	与预期结果一致
3	竖直四连判断	竖直方向出现相同颜色四子连珠, 游戏结束	与预期结果一致
4	对角线四连判断	对角线出现相同颜色四子连珠, 游戏结束	与预期结果一致
5	梯度四连测试	梯度方向出现相同颜色四子连珠, 游戏结束	与预期结果一致

(2) 历史棋局测试。对加载历史棋局、落子及撤销落子功能进行测试, 表 2 对历史棋局的测试用例进行了详细说明。经过测试得出, 系统历史棋局功能良好, 可正常进行历史棋局回放。

Table 2 Test cases for historical chess game
表 2 历史棋局测试用例

序号	测试内容	预期结果	运行结果
1	历史棋局加载	点击加载详情后, 进入对应历史棋局界面	与预期结果一致
2	下一步(落子)	点击下一步后, 回放棋局中的一次落子	与预期结果一致
3	上一步(撤销落子)	点击上一步后, 撤销上一次落子	与预期结果一致

4.2 AI能力测试

市面上三维空间四子棋的相关算法较少, 故通过和计客智能四子棋的不同难度进行对战, 测试了本文系统 AI 的智能水平。计客智能四子棋是一款成熟的智能棋类游戏, 其 AI 算法具有一定的代表性, 因此选用它作为对比对象能够有效地展示本文系统 AI 的性能。在实验中, 计客智能四子棋的 AI 默认为后手落子, 本文系统 AI 进行先手落子(红色), 实体棋盘对应为白色棋子。实验分别在简单难度和困难难度下进行测试, 并将结果与计客智能四子棋的不同级别 AI 对战结果进行比较。经测试, 在简单难度的测试中, 该系统 AI 成功战胜了计客智能四子棋的一级难度 AI, 对战结果如图 10 所示; 在困难难度的测试中, 该系统 AI 成功战胜了计客智能四子棋的二级难度 AI, 对战结果如图 11 所示。

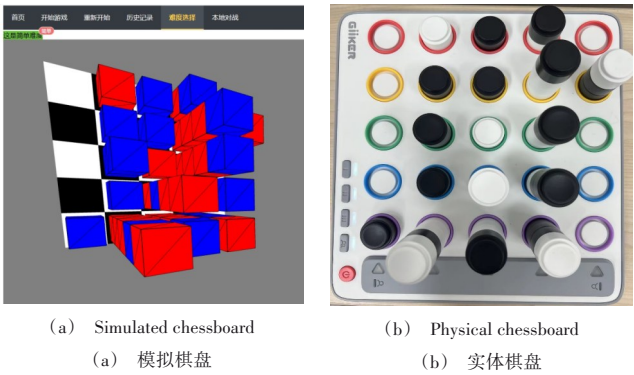


Fig. 10 Easy difficulty AI match results
图 10 简单难度 AI 对局结果

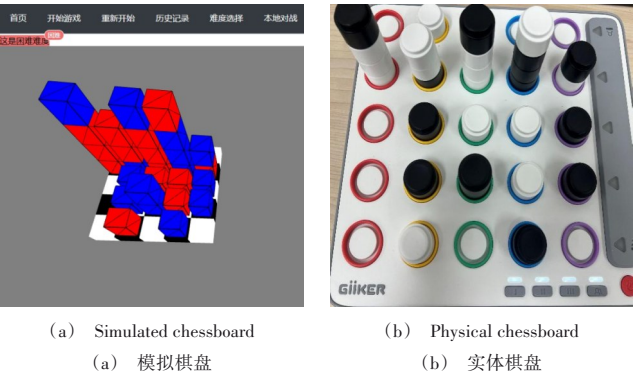


Fig. 11 Hard difficulty AI match results
图 11 困难难度 AI 对局结果

5 结语

本文从理论基础、系统设计、技术实现等方面阐述了三维空间四子棋系统,该系统使用 Alpha-Beta 剪枝算法并通过优化棋形得分实现不同智能水平的 AI。在此基础上,前端使用 Vue 和 Three.js,同时考虑到系统的扩展性,选用了 WebSocket 进行前后端双向通信,后端使用 Spring Boot 实现相关算法。由此,完成人机 AI 对战、本地双人对战、棋局记录及复盘功能的三维空间四子棋系统开发与设计。并且,测试完成与市面上计客智能四子棋实体棋盘的比较,难度划分和实体棋盘的 AI 等级对应,将其设计为一个具有较高水平的人机对战三维空间四子棋系统。

为进一步提升本文系统的 AI 智能性,未来可能会依赖 KANs 分析结果修改棋形得分,同时将原本在三维空间计算的基础棋形,改为在多个二维平面内分别计算得分并求和。将原本 5×5×5 的棋盘在不同角度方向上分割为多个二维平面,这些二维平面需要至少满足 4 个棋子存在于同一条直线上,并根据类似五子棋的 8 种棋形,划分出对应四子棋的棋形得分。同时,提供玩家稳定的在线对战,不断优化和扩展该系统,使其成为一个更加完善、功能丰富的三维空间四子棋游戏平台。

参考文献:

- [1] HUANG Z J. Explore a new dimension of toy design in 3D[J]. Toy Industry, 2022(8):54.
黄子婧. 立体化设计提升创意维度[J]. 中外玩具制造, 2022(8):54.
- [2] ZHOU Z L. Implementation and optimization of game search tree algorithm[J]. Scientific and Technological Innovation, 2021(18):108-110.
周子龙. 博弈搜索树算法的实现及其优化[J]. 科学技术创新, 2021(18):108-110.
- [3] KNUTH D E, MOORE R W. An analysis of alpha-beta pruning[J]. Artificial Intelligence, 1975, 6(4): 293-326.
- [4] YAN J M. Research on strategy search algorithm in two-player zero-sum games[D]. Dalian: Dalian University of Technology, 2021.
闫俊名. 双人零和博弈问题中策略搜索算法的研究[D]. 大连:大连理工大学, 2021.
- [5] DANCHILLA B. Beginning WebGL for HTML5[M]. California: Apress, 2012.
- [6] SUN H P. Design and implementation of Youwei communication community based on SpringBoot and Vue [D]. Chongqing: Chongqing University, 2022.
孙洪盼. 基于 SpringBoot 和 Vue 的友为交流社区的设计与实现[D]. 重庆:重庆大学, 2022.
- [7] LI H. Research and implementation of human computer game algorithm optimization of Gobang[D]. Dalian: Dalian Maritime University, 2021.
李昊. 五子棋人机博弈算法优化研究与实现[D]. 大连:大连海事大学, 2021.
- [8] LU W. Development and research of 3D visualization application platform in mining area based on Three.js [D]. Mianyang: Southwest University of Science and Technology, 2024.
鲁巍. 基于 Three.js 的矿区三维可视化应用平台开发与研究[D]. 绵阳:西南科技大学, 2024.
- [9] XUE L B, LIU Z Y. Real-time communication based on WebSocket[J]. Computer and Digital Engineering, 2014, 42(3):478-481.
薛晓彬, 刘钊远. 基于 WebSocket 的网络实时通信[J]. 计算机与数字工程, 2014, 42(3):478-481.
- [10] QIN Y Y. Research on algorithm for solving minimax problems in machine learning[D]. Jinzhong: Taiyuan Normal University, 2024.
覃媛媛. 机器学习中求解极小极大问题的算法研究[D]. 晋中:太原师范学院, 2024.
- [11] MOZAFFARI S, AZIZIAN B, SHADMEH M H. Highly efficient alpha-beta pruning minimax based Loop Trax Solver on FPGA [C]//Tehran: 2015 18th CSI International Symposium on Computer Architecture and Digital Systems(CADS), 2015.
- [12] PLAAT A, SCHAEFFER J, PIJLS W, et al. Best-first fixed-depth minimax algorithms[J]. Artificial Intelligence, 1996, 87(1-2): 255-293.
- [13] HOLLAND J H. Genetic algorithms [J]. Scientific American, 1992, 267(1): 66-73.
- [14] LIU Z, WANG Y, VAIDYA S, et al. Kan: kolmogorov-arnold networks [DB/OL]. <https://arxiv.org/abs/2404.19756>, 2024.
- [15] JIANG Y. Design of man-machine battle system of Gobang[D]. Chengdu: University of Electronic Science and Technology of China, 2012.
姜勇. 五子棋人机对战系统设计[D]. 成都:电子科技大学, 2012.
- [16] CHEN S B, HE Y M, YUAN J M. Research on gobang falling algorithm [J]. Computer and Information Technology, 2021, 29(5):49-51, 94.
陈树彬, 和昱旻, 原菊梅. 五子棋落子算法的研究[J]. 电脑与信息技术, 2021, 29(5):49-51, 94.
- [17] JI B, YOU H B, LU H X, et al. Method for evaluating quality of self-play chess game learning example [J]. Journal of Chinese Computer System, 2021, 42(3):467-471.
姬波, 尤惠彬, 卢红星, 等. 一种自对弈棋局学习样例质量评价方法[J]. 小型微型计算机系统, 2021, 42(3):467-471.
- [18] LI M H, XIE Z B, YAN P Y, et al. Automatic modulation recognition algorithm based on multi-attention residual network and GRU [J]. Radio Engineering, 2025, 55(1):36-44.
李鸣皓, 解志斌, 颜培玉, 等. 基于多注意力残差网络和 GRU 的自动调制识别算法[J]. 无线电工程, 2025, 55(1):36-44.
- [19] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need [DB/OL]. <https://arxiv.org/abs/1706.03762>, 2017.

(责任编辑:孙娟)