2024 年春季学期
数据结构课程设计赛道 B 实验报告

谢子扬 肖阳杰 苏泰可 苏雨乐

软件学院 2022 级 9 班

1 分工与合作

谢子扬（占比 35%）：项目架构师，主导整个系统的设计和算法框架的构建。负责 MCTS 算法的实现与优化以及项目调试，确保算法在各种棋局情况下的准确性和鲁棒性。

肖阳杰（占比 30%）：负责 Alpha-Beta 剪枝算法的实现和优化、JSON 数据处理以及项目调试，致力于确保算法的稳定性和可靠性。

苏泰可（占比 17.5%）：负责 State 类等类方法的实现和对局测试、对局数据的分析，为算法的迭代和优化提供了重要依据。

苏雨乐（占比 17.5%）：负责 Stone 类等基础类方法的实现和棋型分析、棋型数据的整理与反馈，为决策质量的持续提升提供了重要支持。

2 算法思想

2.1 总体思路

本项目旨在设计一个能够高效且准确地进行六子棋游戏决策的人工智能（bot bot）。

为了应对不同的游戏阶段和复杂性，我们根据棋局进程的先后采用了两种算法：Alpha-Beta 剪枝和蒙特卡洛树搜索（MCTS）。

以下是这两种算法的详细思路：

2.1.1 Alpha-Beta 剪枝算法（作用于棋局前期）：

目的：在游戏初期，棋盘上的空位较多，可能性较大，此时需要一个能够快速排除较差走法的算法。

基本原理：通过两个参数 Alpha 和 Beta 来界定可能的走法价值范围，其中 Alpha 代表最大化玩家（bot）的当前最优估计，Beta 代表最小化玩家（对手）的当前最优估计。在搜索树中，每个节点代表一个可能的棋局状态，算法递归地评估每个状态，通过比较当前节点的值和 Alpha/Beta 的值来决定是否继续搜索该节点的子节点。

剪枝：在搜索过程中，如果某个节点的值已经不可能比已知的更好（对于最大化玩家是 Alpha，对于最小化玩家是 Beta），则该节点的搜索会被提前终止，即“剪枝”，以此减少计算量。

2.1.2 蒙特卡洛树（MCTS）搜索（作用于棋局中后期）：

目的：随着游戏的进行，棋盘上的空位减少，局面变得更加复杂和不确定。MCTS 适用于这种复杂局面，通过模拟可能的走法来评估棋局状态。

基本原理：MCTS 是一种启发式搜索算法，它不需要完全搜索游戏树，而是通过随机抽样来获取信息。算法分为四个主要步骤：选择（Selection）、扩展（Expansion）、模拟（Simulation）、反向传播（Backpropagation）。

选择（Selection）：从根节点开始，按照胜率和访问次数选择最有前途的子节点。

扩展（Expansion）：在被选择的节点上添加一个新的子节点，代表一个未探索的走法。

模拟（Simulation）：从新添加的节点开始，进行随机游戏模拟，直到游戏结束或达到预定的步数，以获得一个胜率估计。

反向传播（Backpropagation）：将模拟的结果（胜负情况）从模拟的终点节点反向传播回选择的节点，更新节点的胜率和访问次数。

2.1.3 两种算法的结合

我们决定在前 5 个回合（turnID<=5 时）使用 Alpha-Beta 剪枝算法，因为此时棋盘状态相对简单，Alpha-Beta 可以更容易打开局面。超过 5 个回合后，转向 MCTS 算法，因为此时剩余空位已经不多，足以得到优秀的决策。Alpha-Beta 剪枝提供了一种在有限时间内寻找较优解的高效方法，而 MCTS 则在面对复杂局面时提供更准确的评估。

2.2 所用方法的特别、新颖或创新之处

2.2.1 Alpha-Beta 剪枝和 MCTS 算法的融合策略

在本项目中，我们采取了一种创新的算法融合策略，以适应棋类游戏在不同阶段的决策需求。具体来说，我们根据棋局的动态变化，巧妙地结合了 Alpha-Beta 剪枝算法和蒙特卡洛树搜索（MCTS）算法。

●Alpha-Beta 剪枝算法的应用：在游戏的初期阶段，即前 5 个回合（turnID <= 5），棋盘上的空位较多，棋局状态相对简单明了。在这个阶段，Alpha-Beta 剪枝算法能够有效地减少搜索空间，快速排除不利的走法，从而在短时间内找到较为合理的落子点。这种方法充分利用了剪枝算法在广度搜索中的优势，为 bot 在早期游戏中抢占先机提供了有力支持。

●MCTS 算法的引入：随着游戏的进行，棋盘上的空位逐渐减少，局面变得愈发复杂。在超过 5 个回合后，我们转向使用 MCTS 算法。MCTS 算法通过模拟随机游戏过程来评估棋局，能够适应更加复杂的决策场景，提供更为深入和准确的走法评估。在这个阶段，MCTS 的引入显著提升了 bot 的决策质量，尤其是在中后期棋局中，能够有效地探索和利用棋盘上剩余的空位，以获得更优的决策。

●算法融合的优势：Alpha-Beta 剪枝算法与 MCTS 算法的结合，为 bot 提供了一种既高效又准确的决策机制。Alpha-Beta 剪枝在时间复杂度较低的早期游戏中发挥优势，而 MCTS 则在局面复杂、需要深入探索的中后期游戏中展现其强大的评估能力。这种算法的动态融合，确保了 bot 在不同游戏阶段都能做出最优或近似最优的决策，从而在整体上提升了 bot 的游戏表现。

通过这种精心设计的算法融合策略，我们的 bot 能够灵活应对棋类游戏中的各种局面，实现在不同阶段的平滑过渡和高效决策。

2.2.2 动态调整探索参数 C 在 MCTS 中的 UCB 评估

在蒙特卡洛树搜索（MCTS）算法中，探索与利用的平衡是决定搜索效率和决策质量的关键因素。探索参数 C，通常用于上置信界限（Upper Confidence Bound，UCB）的计算中，控制着算法在未知领域探索的倾向。在本项目中，我们实现了探索参数 C 的动态调整机制，以适应不同深度和不同游戏阶段的搜索需求。

●UCB 评估的基本概念：UCB 公式结合了节点的胜率和访问频率，以及探索参数 C，来评估每个节点的价值。一个较高的 C 值鼓励算法探索那些尚未充分评估的节点，而一个较低的 C 值则使得算法倾向于选择当前最佳估计的节点。

●动态调整机制：我们的设计中，探索参数 C 不是固定的，而是随着游戏树的深度和游戏进程的推进而动态变化。这种自适应调整允许算法在游戏初期更积极地探索新的走法，在游戏后期则更注重利用已知的有效策略。

●深度依赖性：随着搜索深度的增加，我们通过减少 C 值来降低探索的强度，使得搜索更加聚焦于那些已经显示出较高胜率的路径。这种深度依赖的调整策略，有助于算法在面临时间压力时，快速收敛到最优解或者近似最优解。

●游戏阶段的适应性：在游戏的不同阶段，根据棋盘上的局势和剩余的空位数量，动态调整 C 值以适应当前的搜索需求。例如，在棋盘上剩余空位较多时，增加探索以发现潜在的战略机会；而在棋盘上剩余空位较少时，减少探索以利用已知的有利走法。

●性能优化：通过动态调整 C 值，我们的 MCTS 算法能够更高效地利用计算资源，减少不必要的搜索，同时保持对有前景的新走法的探索。这种优化策略显著提升了算法的总体性能，使得 bot 可以在有限的时间内做出更加精准的决策。

通过这种动态调整探索参数 C 的策略，我们的 MCTS 算法不仅提升了在各种棋局情况下的适应性，而且增强了在复杂决策场景下的鲁棒性和有效性。

2.2.3 关键“路”的识别机制

在本项目的决策系统中，关键“路”的识别机制是一个核心创新点，它赋予了 bot 在复杂棋局中识别和优先处理胜负关键因素的能力。以下是该机制的几个关键特点：

●多维度评估：我们的关键“路”识别机制不仅仅考虑单一的棋子数量或连续性，而是采用多维度的评估标准包括棋子的连续性、潜在的棋型形成，以及它们对棋局控制力的影响。

●实时动态分析：该机制能够实时监控棋盘上的变动，动态地分析每条“路”的潜在价值和紧迫性。这种实时性使得 bot 可以在对手落子后迅速重新评估棋局，及时调整策略。

●优先级排序算法：我们开发了一种先进的优先级排序算法，该算法可以根据“路”的价值分数和战略重要性对其进行排序。这确保了 bot 在决策过程中始终能够关注那些最有可能影响游戏结果的关键路径。

通过这一创新机制，我们的 bot 在处理棋局时能够更加精准地识别出那些对胜负具有决定性影响的“路”，并据此制定出更为高效和有针对性的策略。这种对关键因素的深刻理解和优先处理能力，显著提升了 bot 在激烈棋局中的竞争力。

2.2.4 分化为 Self 和 Opponent 的权重分配策略

在构建项目时，我们引入了一种新颖的权重分配策略，该策略将棋局分析分为两个独立的视角：Self（己方）和 Opponent（对手）。这种分化方法提高了 bot 对棋局复杂性的理解能力，并增强了其决策的深度和准确性。

●视角分离：我们将棋局分析分为两个独立的评估过程，分别从己方和对手的角度出发。这种分离允许 bot 更细致地评估每一步棋对双方的影响，从而在决策时能够更全面地考虑局势。

●自适应权重调整：bot 会根据当前棋局的发展动态调整 Self 和 Opponent 的权重。例如，在攻击时增加己方权重以强化攻击线路的构建，在防守时增加对手权重以更好地预测和阻止对手的威胁。

●风险与机遇的平衡：通过区分 Self 和 Opponent 的权重，bot 能够更好地平衡风险与机遇，识别出那些高风险高回报的棋路，同时规避可能导致失败的走法。

●动态游戏理论应用：此策略体现了动态游戏理论的实际应用，其中 bot 不仅要最大化自身的胜率，还要最小化对手的胜率，这种双向优化推动了 bot 在对抗性游戏中的表现。

通过这种创新的权重分配机制，我们的 bot 在处理棋局时能够更加精准地识别和评估关键线路，无论是在构建攻势还是布置防守时都能够做出更加合理的决策，显著提升了其在棋类游戏中的竞争力和适应性。

2.2.5 Alpha-Beta 剪枝算法实现过程中的边界限制与点位价值排序

在 Alpha-Beta 剪枝算法实现过程中，我们采用了创新策略来提高算法的效率，减少不必要的性能开支：

●边界限制策略：在六子棋对弈过程中，双方的棋子往往一开始只占据棋盘的小部分区域，在该区域内的点位是最具有价值的，而远离战场的点位则往往没有必要进行分析，可以排除在外。所以我们会通过输入的数据确定当前战场的范围，在进行合理的拓宽后，把范围内的点位作为 Alpha-Beta 剪枝算法的待处理点位，而不是一开始便将所有的点位进行评估分析。

●点位价值排序算法：即使缩小了范围，剩下的棋步数仍然无法在限制的时间内搜索完毕。为了能够尽可能下出最好的棋步，我们需要从所有的合法棋步中选出较为重要的。在确定战场边界后，会对边界内的每个可选点位进行评分，将评分高，即最具有战略价值的点位排在前面，从而使得剪枝算法更有机会挑选出最好的棋步。

通过这些策略，我们的 Alpha-Beta 剪枝算法在保证决策质量的同时，显著提高了运行效率，使得 bot 能够在激烈的棋局对抗中快速做出精准的决策。这些优化措施的综合应用，体现了我们在算法实现上的深度思考和精心设计。

3 总结

3.1 遇到问题

在构建项目的准备和实现阶段，我们都遇到了很多不同方面的问题：

●数据结构选择：我们在棋盘数据结构的选择上进行了深入考量。起初，我们面临二维数组和位棋盘两种方案的抉择。尽管位棋盘在处理大规模棋盘时内存效率更优，考虑到六子棋棋盘相对较小且状态简单，位棋盘带来的性能优化微乎其微，且二维数组简洁可见，所以我们最终决定采用二维数组（在项目中由 Board 类实现）来表示棋盘，每个数组元素（在项目中由 Stone 类实现）直接映射棋盘上的一个或多个交叉点状态（在项目中由 State 类实现）。这一决策基于对实现简便性与性能需求的综合考量，简化了实现过程，并提高了代码的可读性和维护性，确保了开发效率和程序的直观性。未来若面临更复杂的棋类游戏，我们将重新评估位棋盘等其他数据结构的适用性。

●搜索算法选择：在项目的准备阶段，我们面临着多个算法的选择，包括但不仅限于：贪心、纯估值、Alpha-Beta 剪枝、蒙特卡罗树（MCTS）搜索、MCTS 并行化和多线程、MCTS 与 Alpha-Beta 剪枝的结合、神经网络模型、根据棋局进程分别使用 Alpha-Beta 剪枝和 MCTS。根据项目的时间限制和 Botzone 平台虚拟机限制，我们选择放弃深度学习和并行化，转而分工完成 Alpha-Beta 剪枝和蒙特卡罗树（MCTS）搜索，最终选择根据棋局进程结合两种算法。这种算法的动态融合，确保了 bot 在不同游戏阶段都能做出最优或近似最优的决策，从而在整体上提升了 bot 的游戏表现。

●状态管理实现：状态管理的实现是一个关键的技术挑战。我们通过广泛的文献回顾，最终采用了“路”的概念，将每条“路”作为独立的评估对象，以直接影响搜索算法的决策过程。在项目中，State 类负责表示和管理每条“路”的状态，它不仅跟踪棋子的连续性，还综合评估了它们的分布模式和潜在棋型形成，如活五、眠四等关键棋型。我们设计了一个复杂的状态影响分析系统，能够评估每条“路”上的棋子如何影响周围区域，以及如何与其他“路”相互作用，形成对整个棋局的全面理解。此外，每条“路”的价值量化考虑了其对棋局的直接影响和潜在发展，包括对棋型威胁的评估和对棋局长远影响的预测。我们的框架能够实时更新“路”的状态，以响应对手的每一步棋，确保 bot 在动态棋局中的适应性。最终，状态管理框架为搜索算法提供了精确的评估数据，并支持了算法在不同“路”之间进行选择和权衡，从而做出最优的决策。这一实现显著提升了 bot 的决策质量和游戏表现。

●参数调优：参数调优是项目中关键而复杂的环节，涉及 Alpha-Beta 剪枝和 MCTS 算法中的多个参数。找到最优的参数设置不仅是一个技术挑战，也是一个耗时的过程。由于参数的敏感性，即便是微小的调整也可能对 bot 的决策质量产生显著影响。此外，参数搜索空间的庞大使得穷尽所有可能性变得不切实际。我们采用了自动化的调参技术，如网格搜索和随机搜索，结合统计学方法来缩小搜索范围，并通过大量的对局模拟来评估不同参数组合的性能。这个过程要求我们不断迭代和测试，以确保找到的参数能够在实际对局中稳定地发挥不错的作用。尽管参数调优是一个繁琐的工作，但它对于提升 bot 的整体性能至关重要。

●头文件循环引用问题：在项目制作中期，因为我们把每一个类都设立在一个单独的头文件，这导致了头文件循环引用的问题，导致了编译错误和效率低下。我们通过重新设计类的接口和依赖关系，使用#pragma once 和将类放在同一个文件解决掉了这个问题。

3.2 排除问题

在构建项目的准备和实现阶段，我们采取了大量的措施以解决程序的 bug 和构建问题：

●代码审查和性能调试：我们实施了严格的代码审查流程，并且使用了 tscancode 等代码分析工具，确保每段代码在合并前都经过了彻底的检查，以此来减少 bug 的引入和热点和瓶颈区域的识别，然后针对性地进行优化。

●面向对象编程：我们采用了面向对象编程（OOP）的原则来构建项目，这使得代码更加模块化和可重用。通过将数据和处理数据的方法封装在类中，我们提高了代码的可维护性和可读性。此外，面向对象的设计还促进了团队成员间的协作，因为每个成员可以专注于特定的类或模块，而不会影响到其他部分。

●定期回顾：团队定期举行技术回顾会议，讨论遇到的问题和潜在的改进措施，保持了开发流程的持续改进。

●知识共享：团队成员之间积极分享知识，相互学习和协助，形成了一个互助合作的学习环境。

3.3 经验和教训

良好的团队沟通和协作是项目成功的关键。在这个过程中，我们不仅建立了高效的工作流程，还培养了团队成员之间的深厚友谊。我们感谢每一位组员的辛苦付出，是他们的不懈努力和创造性思维让项目得以顺利推进。每一位成员都在项目内发挥了重要作用，同时也在团队合作中展现了卓越的协同能力。

不仅如此，在项目中我们从零开始学习了博弈论相关的算法，这不仅提升了我们的技术能力，也拓宽了我们的视野。我们较为深入地研究了 Alpha-Beta 剪枝和蒙特卡洛树搜索等高级算法，学习的过程实在令人酣畅淋漓。在实践中，我们复习并巩固了 C++编程语言和面向对象编程（OOP）的语法知识，更加深刻地理解了知识的实际用途。

开发锻炼了我们的综合项目能力，我们学会了如何将理论知识应用到实际问题中，如何解决实际开发过程中遇到的各种技术难题，以及如何在有限的时间和资源下高效地完成项目。

●得意之处：包括 Alpha-Beta 剪枝和 MCTS 算法的融合策略、动态调整探索参数 C 在 MCTS 中的 UCB 评估、关键“路”的识别机制、分化为 Self 和 Opponent 的权重分配策略、Alpha-Beta 剪枝算法实现过程中的边界限制与点位价值排序等这些项目的创新点，还有我们标准的代码规范和清晰易读的代码风格，是我们的得意之处。

●优化方向：考虑将深度学习技术更深入地集成到 AI 中，优化权值，以进一步提升其学习和适应能力。考虑进行多线程处理，以充分利用现代多核处理器的计算能力。探索算法在其他棋类游戏中的应用，提高算法的泛化能力。对更多的棋型进行更深入的研究，提高细化能力。

4 参考文献

[1]汪坤兵. 六子棋博弈中搜索技术的研究与实现. Diss. 安徽大学, 2016.

[2]黄继平, 张栋, and 苗华. "六子棋智能博弈系统的研究与实现." 电脑知识与技术 (2009).

[3]于江明等. "六子棋博弈系统中机器学习算法设计与研究." 韶关学院学报 35.10(2014):6.

[4]周新林等. "六子棋博弈系统设计与实现." 软件导刊 14.3(2015):3.

[5]刘雅靖. "计算机博弈之六子棋的主要技术分析." 电脑知识与技术：学术版 7.4(2011):3.

[6]何轩等. "机器博弈主要技术分析——以六子棋为例." 电脑知识与技术：学术版 15.11X(2019):2.

[7]李一波, 安涌, and 乔志华. "人工神经网络在六子棋机器博弈中的应用." 中国人工智能学会第 12 届全国学术年会 0.

[8]周菁菁. "六子棋——人工智能系统的设计与研究." 湖北广播电视大学学报 (2011).

[9]段浴, and 王宛宛. "基于人工智能的六子棋博弈平台研究与实现." 科技创新与应用 12.19(2022):4.

[10]ML | Monte Carlo Tree Search (MCTS) - GeeksforGeeks

[11]https://download.csdn.net/download/qq_28273781/10484356

[12]https://zhuanlan.zhihu.com/p/341547952

[13]https://zhuanlan.zhihu.com/p/394688802

[14]https://www.bilibili.com/video/BV1sq4y1u7UY/?spm_id_from=333.788&vd_source=543edd3bca1686f83c3ca16ad0eb7486

[15]【详细原理】蒙特卡洛树搜索入门教程! - 简书 (jianshu.com)

[16]Monte Carlo Tree Search - beginners guide int8.io

[17]面向初学者的蒙特卡洛树搜索 MCTS 详解及其实现_mcts 算法-CSDN 博客

[18]python 实现的基于蒙特卡洛树搜索(MCTS)与 UCT RAVE 的五子棋游戏 - xmwd - 博客园 (cnblogs.com)

[19]六子棋的 15 种棋形 - 知乎 (zhihu.com)

[20]【强化学习】多臂老虎机的上置信界算法（全网最通俗） - 知乎 (zhihu.com)

[21]【Algorithm】最容易理解的蒙特卡洛树搜索（Monte Carlo Tree Search，MCTS）算法_monte carlo 搜索模板-CSDN 博客