北京大学

硕士研究生学位论文

题目：一种定量评估斗地主游戏 AI 智能水平的方法——蒙特卡洛当量

姓　　名：	王政飞
学　　号：	1701214014
院　　系：	信息科学技术学院
专　　业：	计算机软件与理论
研究方向：	模式识别与生物特征识别
导师姓名：	李文新 教授

二〇二〇 年 六 月

版权声明

任何收存和保管本论文各种版本的单位和个人，未经本论文作者同意，不得将本论文转借他人，亦不得随意复制、抄录、拍照或以任何方式传播。否则，引起有碍作者著作权之问题，将可能承担法律责任。

摘要

游戏是人工智能研究领域的重要组成部分，许多现实世界的问题都可以映射到游戏中寻求解决方法。目前游戏 AI 领域的研究热点之一是非完全信息的多智能体博弈问题，而斗地主游戏就是一种典型的非完全信息的多智能体博弈问题。另一方面从游戏 AI 的算法上看，蒙特卡洛树搜索算法是一种不依赖于特定游戏的通用搜索方法，目前被广泛用于包括 AlphaGo 在内的很多游戏 AI 中。本文作者围绕蒙特卡洛树搜索算法在斗地主游戏中的应用展开相关研究，主要关注的问题有两个：第一个是使用蒙特卡洛方法对斗地主游戏的难度进行评估，第二个是使用蒙特卡洛树搜索算法对斗地主游戏 AI 的智能水平进行评估。论文的主要工作包括：

1）提出了蒙特卡洛复杂度的概念并给出了相应的计算方法，用于刻画一个游戏问题使用蒙特卡洛树搜索算法进行求解的难易程度；计算分析了三组随机抽取的斗地主游戏初始手牌下的蒙特卡洛复杂度，并将之与黑白棋和西非播棋进行了比较。蒙特卡洛复杂度通过分支数、不同分支的收敛速度、价值分布、子树深度等属性较为直观地绘出了斗地主游戏的搜索树画像；

2）针对游戏 AI 的水平评估，扩展了前人通过蒙特卡洛树搜索算法对确定性完全信息博弈游戏 AI 水平评估的工作，将该方法应用到了斗地主这一随机性非完全信息博弈游戏的 AI 智能水平评估上，并命名为蒙特卡洛当量。在此基础上，提出了对于人类经验价值的蒙特卡洛当量评估方法；

3）扩展了 Botzone 在线游戏 AI 对战平台，使其可以在本地环境下进行模拟运行，为 AI 的本地训练和评估提供了便利。

蒙特卡洛复杂度的提出，为游戏的难度评估增加了新的维度，使得能够从更加全面的角度去刻画游戏问题的难度。蒙特卡洛当量的研究使得游戏 AI 的评估可以独立进行、不再受到其他因素的影响；同时也形成了对人类经验价值的评估方法。Botzone 平台的扩展工作为使用者提供了更多的便利，使得平台开放的资源能够更好地为研究人员所用。

关键词：人工智能，游戏 AI，蒙特卡洛树搜索

A Quantitative Evaluation Method for DouDiZhu AI's Intelligence – Monte Carlo Equivalent

Zhengfei Wang (Computer Software and Theory)
Directed by Professor Wenxin Li

ABSTRACT

Game plays an important role in AI. Many real-world problems can be mapped into games to seek solutions. Currently, one of the research interests is multi-agent in imperfect information games. On the other hand, from the point of view of game AI algorithms, Monte Carlo Tree Search (MCTS) is a general search method without dependencies on specific games and it is widely used in many game AI, including AlphaGo. In this paper, the author conduct research on applications of MCTS in DouDiZhu. There are two major concerns: the first is to evaluate the difficulty of DouDiZhu using Monte Carlo method, the second is to evaluate DouDiZhu AI's intelligence level using MCTS. The main work of this paper includes:

1) Propose the concept of Monte Carlo Complexity and its calculation method, this can describe the difficulty of solving a game problem via MCTS. The experiment computes three DouDiZhu games' Monte Carlo Complexity with randomly selected initial states and compares the results with Reversi and Awari. Monte Carlo Complexity can directly illustrate game tree of DouDiZhu through the properties of the branching factors, convergence speed, value distribution and depth of subtrees.
2) Extend the previous work on the evaluation of deterministic perfect information game AI using MCTS and apply it to random imperfect information game DouDiZhu. Name this method as Monte Carlo Equivalent and propose evaluation method on human knowledge base on it.
3) Extend Botzone to make it can be simulated in the local environment, which provides convenience for AI's training and evaluation.

The proposed Monte Carlo Complexity adds a new dimension to the difficulty evaluation of games, making it possible to depict the difficulty of game problems from a more comprehensive perspective. Monte Carlo Equivalent allows game AI to be evaluated independently of other factors and it also forms an evaluation method for the value of human knowledge. The extension of Botzone has made it easier for users to access the platform's open resources.

KEY WORDS: Artificial Intelligence, Game AI, Monte Carlo Tree Search

目录

第一章　引言 1
1.1　研究背景 1
1.1.1　人工智能的发展 1
1.1.2　游戏领域的发展 2
1.1.3　游戏 AI 水平的发展 2
1.1.4　人工智能研究中游戏问题的发展 2
1.2　研究问题的提出 4
1.2.1　游戏难度的评估 4
1.2.2　游戏 AI 智能体的水平评估 4
1.2.3　斗地主游戏的研究意义 4
1.3　本文的主要工作和创新点 5
1.4　本章小结及后续章节的安排 5
第二章　工作基础及相关工作的国内外研究现状 7
2.1　游戏问题的复杂度分析方法 7
2.1.1　状态复杂度分析与博弈树复杂度 7
2.1.2　存在的问题 8
2.2　游戏 AI 智能水平的评估方法 8
2.2.1　人机对战 8
2.2.2　集中式比赛 8
2.2.3　在线平台 9
2.2.4　存在的问题 9
2.3　蒙特卡洛树搜索及其相关研究进展 9
2.3.1　蒙特卡洛树搜索的基本原理 9
2.3.2　蒙特卡洛树搜索在游戏 AI 领域的应用 11
2.3.3　蒙特卡洛树搜索在游戏 AI 评估的研究工作 11
2.3.4　相关研究对本文工作的启发 12
2.4　斗地主游戏及其相关研究进展 12
2.4.1　斗地主游戏规则 12
2.4.2　斗地主游戏的性质分析 13
2.4.3　斗地主游戏 AI 的研究进展 13
2.5　Botzone 在线游戏 AI 对战平台 13

2.5.1 整体情况介绍 13
2.5.2 游戏 AI 的天梯排行榜评估方式 16
2.5.3 Botzone 平台的斗地主游戏及其评估方式 16
2.5.4 局限性和斗地主游戏 AI 评估方式的不足 17
第三章 Botzone 平台的扩展：本地模拟环境 21
3.1 设计背景 21
3.2 设计与实现 22
3.2.1 本地模拟环境的设计 22
3.2.2 与 Botzone 平台兼容的设计 25
3.2.3 其他设计与实现的重难点 25
3.3 工作流程 27
3.3.1 本地对局的运行流程 27
3.3.2 对局日志数据集的处理流程 28
3.4 与 Botzone 平台的关联性 29
3.5 为斗地主游戏 AI 研究提供的便利 29
3.6 本章小结 30
第四章 斗地主游戏问题的蒙特卡洛复杂度分析 31
4.1 斗地主游戏问题的传统复杂度分析 31
4.1.1 状态复杂度 31
4.1.2 博弈树复杂度 31
4.1.3 结果分析 32
4.2 初始手牌对评估的影响 32
4.2.1 初始手牌倾向性的定义 32
4.2.2 初始手牌倾向性的计算方法 33
4.2.3 实验设计、结果与分析 35
4.3 蒙特卡洛复杂度分析方法 37
4.3.1 定义蒙特卡洛复杂度的原因 37
4.3.2 蒙特卡洛复杂度的定义 37
4.3.3 对蒙特卡洛复杂度数值含义的解释 39
4.3.4 蒙特卡洛复杂度的计算方法 39
4.4 斗地主游戏的蒙特卡洛复杂度计算实验 41
4.4.1 实验设计 41
4.4.2 实验结果总览 41

4.4.3 实验结果分析：子节点个数和总迭代次数 42
4.4.4 实验结果分析：子节点迭代次数 43
4.4.5 实验结果分析：子节点价值估计 43
4.4.6 实验结果分析：子节点深度 44
4.4.7 实验结果分析：时间开销 45
4.4.8 实验结果分析总结 45
4.5 与黑白棋和西非播棋的对比实验 46
4.5.1 实验设计 46
4.5.2 实验结果及分析 46
4.6 本章小结 48
第五章 斗地主游戏 AI 智能水平的蒙特卡洛当量评估 49
5.1 AI 智能水平的蒙特卡洛当量评估方法 49
5.1.1 蒙特卡洛当量的定义 49
5.1.2 蒙特卡洛当量的计算方法 50
5.1.3 验证实验的设计、结果与分析 53
5.2 斗地主游戏 AI 评估前的准备工作 55
5.2.1 评估对象 55
5.2.2 迷你斗地主 56
5.3 斗地主游戏 AI 的蒙特卡洛当量评估 57
5.3.1 实验设计 57
5.3.2 实验结果及分析 57
5.4 斗地主游戏 AI 中人类经验价值的蒙特卡洛当量评估 58
5.4.1 实验设计 58
5.4.2 实验结果及分析 58
5.5 蒙特卡洛当量评估在不同游戏间横向比较的可行性分析 59
5.6 本章小结 59
第六章 总结与展望 61
6.1 本文工作总结 61
6.2 研究工作展望 61
参考文献 63
个人简历、在学期间的研究成果 66
致谢 67

北京大学学位论文原创性声明和使用授权说明 68

第一章　引言

1.1　研究背景

人工智能（Artificial Intelligence, AI）自 1956 年达特茅斯会议提出以来，经过 60 余年的演进，已经在语音识别、视觉识别、机器博弈、自动程序设计等诸多领域取得了无数的突破。各国也在积极制订各自的人工智能发展战略，中国国务院在 2016 年发布的《“十三五”国家科技创新规划》中明确将人工智能作为发展新一代信息技术的主要方向，并在 2017 年印发《新一代人工智能发展规划》明确发展的战略目标。

游戏作为人工智能研究领域的重要组成部分，相比于机器人等现实世界的其他领域，游戏的所有事件、变量和结果都是人为可控的，相关的数据也可以通过不断的模拟游戏运行来产生。这些在现实世界中不可能实现的条件，使得研究者可以更加关注于问题本身而不受其他因素的干扰，让游戏成为了得天独厚的人工智能研究场。除此之外，游戏本身的趣味性也使得它更具有吸引力。Yannakakis 等人在《Artificial Intelligence and Games》一书中也提到，游戏是人工智能最合适的实验台（Yannakakis et al., 2018）。

1.1.1　人工智能的发展

随着核心算法的不断突破、计算机计算能力的迅速提高，以及互联网海量数据的支撑，人工智能在近年来不断发展，成为全球瞩目的科技焦点。随着人工智能技术的不断成熟，其应用领域不断增大、应用产品不断丰富。目前人工智能的应用技术覆盖了语音识别、语音合成的语音类技术，生物识别、图像识别、视频识别的视觉类技术，以及机器翻译、文本挖掘、情感分析的自然语言处理类技术。而在自动驾驶、无人机、虚拟现实与增强现实、智能机器人等相关领域，人工智能的应用也是百花齐放。

在与日常生活最为贴近的产品应用层次，以智能音箱、智能机器人和无人机为代表的人工智能终端产品已经开始出现在人们生活的周围：小米的小爱同学、亚马逊的 Echo 智能音箱已经是国内外用户十分熟悉的智能音箱产品；码头的搬运机器人、家里扫地机器人随处可见；大疆的无人机无论在商用无人机领域还是民用无人机领域都取得了十分优秀的成绩。在行业应用上，以 IBM Watson 为代表的智能医疗应用、海康威视的安防应用、旷视科技的人脸识别技术等也纷纷落地，在各自的领域都开创了新的技术纪元。

与此同时，社会也开始关注人工智能对教育和就业的影响，在隐私与安全方面的隐患以及对社会公平的影响等问题。这些都足以证明人工智能已然发展到了一个全新的阶段。

1.1.2 游戏领域的发展

游戏能够为玩家提供娱乐和放松，同时也是一种功能强大的工具。儿童可以通过游戏发展某些生活技能，教育工作者可以通过游戏补充课堂教学的方法。而随着先进的图形引擎和信息技术的出现，游戏领域也得到了巨大的发展。

随着智能手机、平板电脑等设备的游戏技术不断成熟，加上原有的 PC 端和游戏机端，游戏设备呈现出多元化的趋势。越来越精美的游戏画面、愈发吸引人的游戏故事、更富有挑战性的游戏设置，这些都吸引着全世界的游戏爱好者成为游戏的消费者。在游戏种类上，动作类、角色扮演类、射击类、即时战略类、模拟策略类等经典类型仍旧坚挺，具有更高自由度的开放世界类游戏近年来也收获了许多游戏爱好者的追捧。根据国外媒体的统计，截至 2019 年底，全球的游戏玩家数量已超过 25 亿、游戏收入突破 1500 亿美元。

1.1.3 游戏 AI 水平的发展

自从 1997 年深蓝战胜国际象棋世界冠军加里·卡斯帕罗夫之后（Campbell et al., 2002），人类开始接受人工智能是可以在游戏领域与人类相媲美的，游戏 AI 的研究也开始集中到围棋上。在 2016 年 AlphaGo 战胜围棋世界冠军李世石之后（Silver et al., 2016），人类开始接受人工智能是可以在游戏领域超越人类的。游戏 AI 背后的算法，也从传统的 Alpha-Beta 剪枝、专家系统，逐渐演变到如今的神经网络、强化学习。游戏 AI 水平也随着算法的不断革新，来到了全新的高度，同时也为游戏本身的发展起到了极大的推动作用。

基于 AlphaGo 的围棋教学工具 AlphaGo Teach（图 1.1）通过比较 AlphaGo 和不同水平棋手的下法，帮助使用者学习围棋的布局手法。腾讯 AI Lab 基于 AlphaGo 论文自主研发的围棋程序“绝艺”已经作为中国国家围棋队专用训练 AI 服役两年之久，为国手们近两年横扫世界棋坛立下汗马功劳。除了围棋，游戏 AI 水平的提升也推动着其他游戏的发展，例如腾讯推出的王者荣耀 AI“绝悟”，既能够帮助玩家熟悉游戏规则、练习游戏技巧，还可以在必要时顶上、保证玩家的游戏体验（Ye et al., 2019）。

1.1.4 人工智能研究中游戏问题的发展

在攻克围棋这一难题之后，人工智能的研究开始面向更加复杂的游戏问题。在当下研究的热门游戏中，德州扑克、麻将、桥牌等牌类游戏属于不完全信息博弈游戏，由于玩家不能观察到游戏的完整局面信息，在决策时需要使用更复杂的算法来选择动作；星际争霸、DotA、王者荣耀等游戏属于多智能体游戏，需要玩家之间通过有效的协作配合来争取胜利。除此之外，以 Minecraft 为代表的开放性游戏由于游戏本身没有明确的目标、玩家可以在其中做任何想做的事情，更加贴近现实生活中人类的活动状态，也

得到了学业、业界的广泛研究。总的来说，随着人工智能技术的不断进步、硬件资源计算能力的不断提升，人工智能研究的游戏问题在变得越来越复杂、难度越来越高，也越来越贴近现实生活。

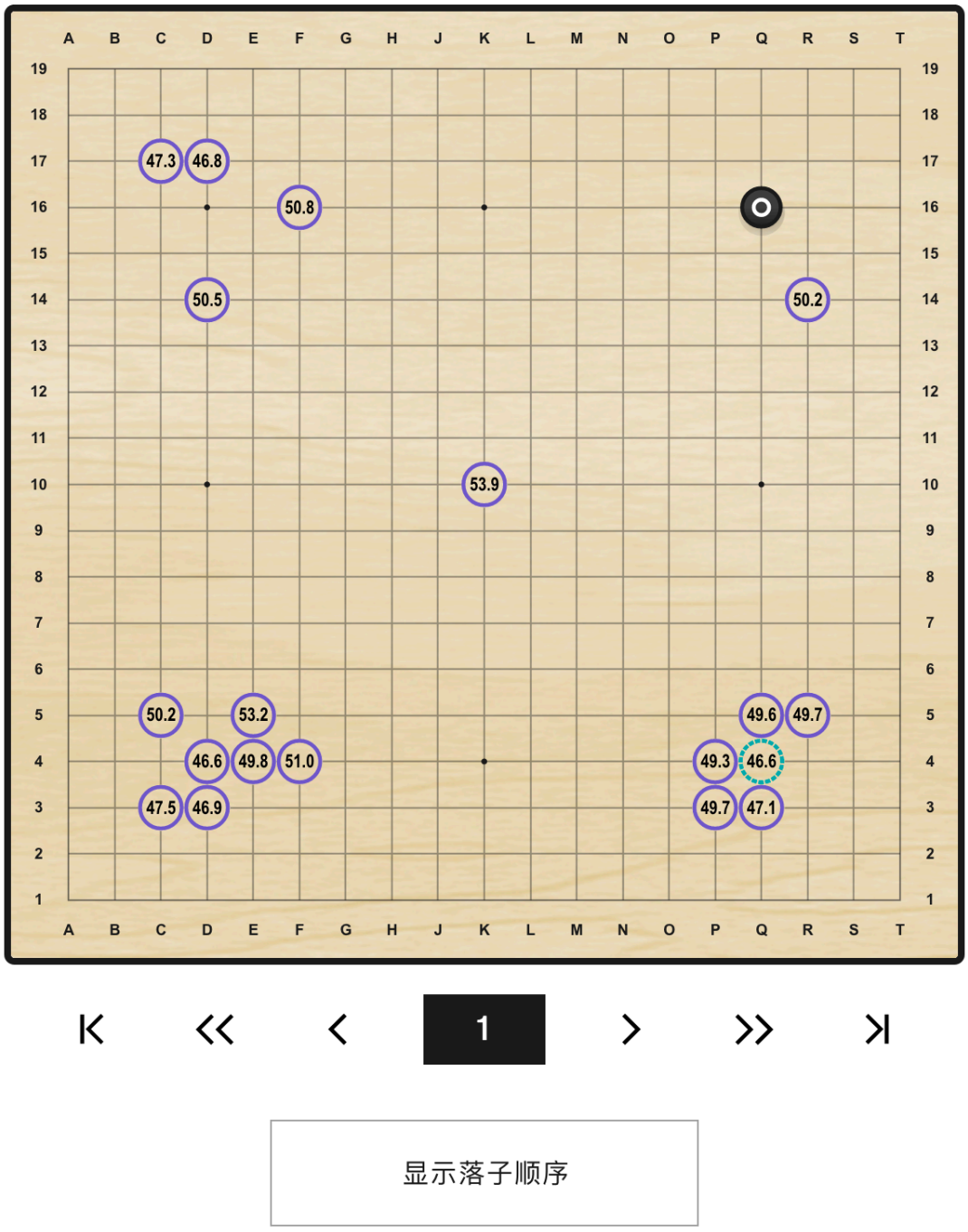

图 1.1　围棋教学工具 AlphaGo Teach

1.2 研究问题的提出

1.2.1 游戏难度的评估

游戏 AI 的研究核心是游戏 AI 的算法，即如何在游戏中根据观察到的状态进行决策以获得更高的收益。不同的游戏问题难度自然不同，不同算法在不同游戏问题上的效果也存在区别。因此如何有效、全面、准确的对游戏难度进行评估是十分关键的。

目前描述游戏难度的方法主要是通过状态复杂度、博弈树复杂度等与游戏问题搜索空间大小直接相关的属性来衡量游戏的难度。由于先前研究中最常使用的算法是搜索算法，所以搜索空间大小是非常合适的衡量工具。而随着神经网络这类基于采样统计的算法研究逐渐成熟，对于游戏难度的描述也需要从更多的维度进行刻画。

1.2.2 游戏 AI 智能体的水平评估

通常情况下，游戏 AI 智能体的水平评估是通过待评估的 AI 程序间直接进行对局来评估的，游戏的结果可以直接反映其相对水平。这种方式要求待评估的 AI 程序之间能够通过某种方式进行游戏对局。如果游戏本身存在随机性，需要确保评估时游戏对局的随机性不会影响到游戏 AI 的水平，必要时可能需要进行多场对局以消除随机性的干扰。

游戏 AI 的评估往往是以相对名次、相对分数的形式给出评估结果，即某个 AI 在参与评估的 AI 中排名多少、分数是多少。这样的结果可以方便的比对任意两个或多个参与评估的 AI 之间的水平；但是如果希望与一个没有参与评估的 AI 进行比较，则必须再进行评估才可以获得结果。

另外，虽然可以通过控制评估时的硬件条件、决策时间等因素对游戏 AI 进行限制、确保一个相对公平的评估环境，但是不同的游戏 AI 在实现时所依赖的硬件条件和消耗的时间是可能存在巨大差别的。一个基于深度神经网络的游戏 AI 程序在训练时所需的硬件条件和时间通常是远大于一个基于专家系统实现的游戏 AI 程序的，而仅仅通过控制评估时的决策时间对于后者显然是不公平的。一种很容易想到的方法是让 AI 的训练也在相同的硬件资源和时间条件下完成，但是这样虽然保证了评估的公平性，但是对于具有充足资源的研究者来说是一种限制，同时也可能限制游戏 AI 的发展。

1.2.3 斗地主游戏的研究意义

游戏 AI 目前已经很好地解决了围棋这一非常复杂的完全信息博弈游戏，开始向非完全信息博弈游戏领域研究。

斗地主游戏作为一种在中国十分流行的纸牌游戏，具有良好的群众基础。由于玩家不能观察到其他玩家的手牌情况，同时由于初始手牌的情况具有较强的随机性，斗

地主游戏属于随机非完全信息博弈游戏。相比于国际象棋、围棋等游戏 AI 已经很好解决的完全信息博弈游戏，不能完全观测到游戏全部局面信息带来的不确定性使得其决策难度大大增加。相比于麻将、桥牌等非完全信息博弈游戏，斗地主游戏的规则十分简单、上手难度更低；同时由于斗地主的牌型组合较多、问题的动作空间非常大，使得其具有一定的难度。总的来说，斗地主非常适合作为游戏 AI 在非完全信息博弈游戏领域的问题对象开展相关的研究。

1.3　本文的主要工作和创新点

本文选取斗地主这一随机非完全信息博弈游戏作为研究对象，通过对其游戏难度评估方式以及游戏 AI 智能体的水平评估方式进行探究，尝试解决上一节提出的研究问题：针对 1.2.1 小节提出的问题，本文提出了蒙特卡洛复杂度的概念并给出了相应的计算方法，用于刻画一个游戏问题使用蒙特卡洛树搜索方法进行求解的难易程度。计算分析了三组随机抽取的斗地主游戏初始手牌下的蒙特卡洛复杂度，并将之与黑白棋和西非播棋进行了比较。蒙特卡洛复杂度通过分支数、不同分支的收敛速度、价值分布、子树深度等属性较为直观地绘出了斗地主游戏的搜索树画像。针对 1.2.2 小节提出的问题，本文扩展了前人通过蒙特卡洛树搜索算法对确定性完全信息博弈游戏 AI 水平评估的工作，将该方法应用到了斗地主这一随机性非完全信息博弈游戏的 AI 智能水平评估上，并命名为蒙特卡洛当量。在此基础上，提出了对于人类经验价值的蒙特卡洛当量评估方法。

除此之外，本文在 Botzone 在线游戏 AI 对战平台的现有工作基础上，对其提供的资源进行整合，实现了本地模拟环境的平台功能扩展，提供了统一的游戏逻辑实现方式和使用方法，并对其他资源的使用进行了不同程度的优化，使得使用者可以更加方便的使用 Botzone 平台；也为本文进行的相关实验奠定了良好的实验环境基础。

1.4　本章小结及后续章节的安排

本章首先介绍了本文的研究背景，按照人工智能、游戏领域、游戏 AI 水平和人工智能研究中游戏问题四个角度的发展历程进行了介绍，然后引出当下存在的一些问题，提出了研究问题。最后对本文的主要工作和创新点进行了说明。

本文对后续章节的安排如下：第二章介绍游戏问题的复杂度分析方法、游戏 AI 智能体水平的评估方法、蒙特卡洛树搜索算法、斗地主游戏的相关研究，以及本文依托的工作基础——Botzone 在线游戏 AI 对战平台。第三章展示针对 Botzone 平台现有的局限性与不足设计并实现的扩展——Botzone 平台的本地模拟环境，使得研究者可以更好地使用 Botzone 平台提供的 Bot、对局数据等资源。第四章从斗地主游戏问题的复杂度

入手，提出蒙特卡洛复杂度这一定义，从游戏问题的价值分布角度研究如何评估游戏难度。第五章提出蒙特卡洛当量以及其计算方法，通过实验展示其对 Botzone 平台的斗地主游戏 Bot、人类经验价值的评估结果，并探讨了其在不同游戏间进行评估的可行性。最后，第六章将总结本文的工作，并对未来工作的努力方向进行展望。

第二章　工作基础及相关工作的国内外研究现状

本章介绍本文的工作基础以及相关工作的研究现状。首先对游戏问题的复杂度分析方法进行介绍，并指出现有方法存在的问题；然后对现有的游戏 AI 智能水平评估方法进行展示；接着阐述蒙特卡洛树搜索和斗地主游戏的相关研究工作；最后介绍 Botzone 在线游戏 AI 对战平台的相关情况。

2.1　游戏问题的复杂度分析方法

游戏难度作为游戏的重要属性，一直以来缺少一种方法对其进行明确的度量。状态复杂度和博弈树复杂度是两种常用的描述游戏难度的方法，它们从不同角度对游戏问题进行空间大小的衡量，进而得到不同算法的表现估计、达到辅助算法选择的目的。

2.1.1　状态复杂度分析与博弈树复杂度

状态复杂度是指从游戏的初始局面出发，所有可能产生的合法且不同的局面总数，通常使用枚举和组合数学进行精确估计；在无法精确估计时，使用可能的局面总数上限作为粗略估计。

博弈树复杂度是指从游戏的初始局面出发，得到最优策略所需要展开的博弈树的最少叶子节点个数；精确估计博弈树复杂度只能通过枚举实现，而该方法在面对复杂问题时是不可行的，因此在实际估计时通常采用平均分支因子方法粗略估计。

宽度优先搜索算法求解时的复杂度理论下界就是问题所有状态的个数，也就是游戏的状态复杂度；而博弈树复杂度对应的是深度优先搜索算法的求解复杂度理论下界。由于先前在游戏领域使用的大都是基于宽度优先搜索或深度优先搜索的方法，因此状态复杂度、博弈树复杂度对于估计游戏问题复杂度以及算法的选择具有较大的意义。

对于黑白棋、围棋等确定性完全信息游戏，由于每一步动作的执行结果都是确定的，且所有游戏局面信息都是完全可被观测到的，问题的博弈树复杂度是可以直接通过规则进行简单推导得到的。以围棋为例，每局围棋游戏的平均步数约为 150，平均分支因子约为 250，因此其博弈树复杂度使用平均分支因子法计算公式为 250^{150}、约为 10^{360} （Allis, 1994）。当游戏引入随机事件（随机游戏），或由于每个玩家不能观测到全部局面信息而产生不确定性（非完全信息游戏）时，博弈树上的每个节点由于无法确定对方的合法动作，状态转移也是不确定的，无法直接继续构建博弈树、也无法计算博弈树复杂度。

对于该问题，可以通过机会节点来对问题进行建模（Russell et al., 2003）。对于随机游戏，将每个可能产生随机性的节点都作为机会节点展开，每个分支下放置对应随

机事件的节点、并赋予对应的概率以及相应的价值；对于非完全信息游戏，由于每个节点的局面信息都是不完全的，每个节点下面都会存在机会节点，每个分支下面需要枚举其他玩家所有可能的局面，各个分支的概率相等且和为1。显然通过机会节点的方法，游戏问题中的随机性通过完全展开而消除，博弈树复杂度的计算不再需要关注游戏是否存在随机性以及信息是否完全可观测；同时也使得随机游戏、不完全信息游戏的复杂度大大提升。

2.1.2 存在的问题

游戏问题变复杂时，对应的状态复杂度、博弈树复杂度也会相应变大；反过来，当状态复杂度、博弈树复杂度变得很大时，游戏问题也会变得更难、不容易被求解。当问题足够复杂时，计算机无法求得其最优解，只能计算其近似最优解来代替；同样对于传统搜索算法来说，遍历整个搜索空间已经变得不可能，必须依赖于启发式函数等方法减小空间、完成状态的估值。由于启发函数多种多样、对于游戏问题的影响是难以估计的，状态复杂度、博弈树复杂度这些方法都不能进行很好的估计，十分接近的状态复杂度有可能对应着大相径庭的结果。除此之外，像蒙特卡洛树搜索、神经网络这些基于采样的统计方法不仅需要关注状态复杂度、博弈树复杂度，还需要对价值本身的分布有所掌握，而这些信息是现有方法无法提供的。

2.2 游戏 AI 智能水平的评估方法

本节对游戏 AI 智能水平的评估方法进行介绍。现在游戏 AI 的智能水平主要通过对局的方式来评估，获胜者便是智能水平更强的一方。根据对局对手的类型，可分为人机对战和计算机程序之间的对战，后者还可细分为集中式比赛和在线平台两种。

2.2.1 人机对战

顾名思义，人机对战就是通过游戏 AI 程序与人类玩家进行对局来对游戏 AI 程序的智能水平进行评估。对于许多游戏来说，人类都代表着极高的智能水平，能够战胜人类玩家对于对应的游戏 AI 领域都是十分重要的里程碑。目前比较著名的人机对战有 1997 年国际象棋程序深蓝战胜加里·卡斯帕罗夫和 2016 年 AlphaGo 战胜李世石。

2.2.2 集中式比赛

集中式比赛的评估方法就是由某个组织发起，将某个游戏的 AI 程序集中起来，通过某种比赛赛制进行比赛，最终的胜利者被认为是智能水平最强的游戏 AI 程序。这样的比赛通常作为人工智能类会议的竞赛单元举办，例如每年 IEEE CoG（Conference on Games，原 Conference on Computational Intelligence and Games）会议的星际争霸 AI 竞

赛，自 2010 年以来每年都会安排星际争霸的 1V1 游戏 AI 对抗赛；NeurIPS（Neural Information Processing System）会议的 ViZDoom AI 竞赛、IJCAI（International Joint Conferences on Artificial Intelligence）会议的麻将比赛等等。这些比赛通常会安排优胜者在会议相关议程期间报告使用方法，并邀请撰写论文。

除此之外，还有一些单独举办的游戏 AI 比赛。例如 2018 年腾讯世界人工智能围棋大赛，来自中国、美国、日本等国家和地区的 11 支围棋 AI 队伍参与了本次比赛，最终腾讯 AI Lab 团队自主研发的围城 AI 程序“绝艺”取得了比赛的冠军。

2.2.3　在线平台

在线平台的评估方法一般是通过排行榜的方式对参与评估的游戏 AI 程序进行排名，用户将自己的代码提交至在线平台并由平台进行实时调度。经过一段时间的评估之后，得到一个排名作为评估结果。国外比较有名的在线平台有 The AI Games（Unal et al., 2012）、Google Research Football（Kurach et al., 2019）等；国内比较成熟的是 Botzone 在线游戏 AI 对战平台（Zhang et al., 2012; Zhou et al., 2017，2018），本文将在 2.5 节对该平台进行更加详细的介绍。

2.2.4　存在的问题

本节介绍的几种现有的游戏 AI 评估方法，或依赖人类玩家、或依赖其他游戏 AI 程序，很难做到独立评估。具体到每种类型，人机对战虽然能够很好地展示游戏 AI 的智能水平，但是由于人类玩家通常具备较强的实力，该方法只能用于顶级游戏 AI 的评估；集中式比赛对于资源的消耗过大，同时只能在一定时间内举行、不能随时进行评估；在线平台可以解决实时性的问题，但是这种方式十分依赖平台的建设与维护，同时往往需要与平台具有比较强的耦合性，对于游戏 AI 的评估存在着一定的不便之处。

2.3　蒙特卡洛树搜索及其相关研究进展

本节介绍蒙特卡洛树搜索算法的基本原理及发展历程，然后介绍其在游戏 AI 领域的一些应用以及其在游戏 AI 评估方面的研究工作，最后对这些研究对于本文工作的启发进行简单阐述。

2.3.1　蒙特卡洛树搜索的基本原理

蒙特卡洛树搜索（Monte Carlo Tree Search，MCTS）是一种启发式搜索算法，最早由 Coulom 在 2006 年提出。他在传统树搜索算法的基础上使用蒙特卡洛方法进行价值评估，不断从当前状态进行随机模拟，然后计算平均收益作为当前状态的估计值。搜索时根据估计值有选择的迭代加深搜索层数。他在围棋程序 CrazyStone 中实现了这一算

法（Coulom, 2006），并在第 10 届 KGS（Kiseido Go Server）计算机围棋比赛中赢得了 9×9 规模的冠军。同年，Kocsis 等人对蒙特卡洛树搜索算法进行了改进、将多臂老虎机问题中常用的 UCB1 算法（Auer et al., 2002）应用到节点的选择，得到了 UCT（Upper Confidence bound extended for Tree）算法（Kocsis et al., 2006）。他们通过实验证明 UCT 算法能够发现具有更高收益的动作，同时在无限次采样的情况下，收敛到最优动作的速度相比于 Alpha-Beta 等传统树搜索算法更快。目前蒙特卡洛树搜索算法在无特殊情况下所指的都是 UCT 算法，本文在接下来的叙述中也将遵照这一规则。

如图 2.1 所示，蒙特卡洛树搜索主要包含选择、扩展、模拟和更新四个步骤。

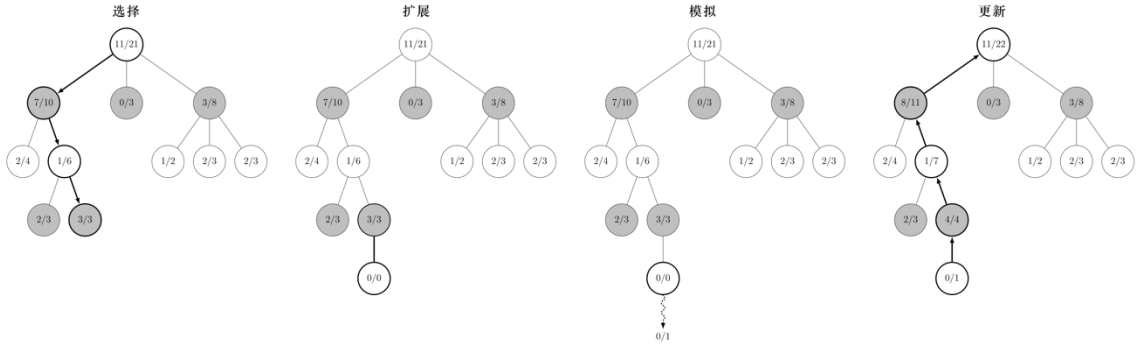

图 2.1 蒙特卡洛树搜索算法流程

选择（Selection）：从根节点出发，递归地调用子节点选择策略向搜索树的下方延伸，直到访问到一个终止节点或从未访问过的子节点截止。子节点选择策略也被称为树策略（Tree Policy），通常使用表达式（2-1）作为选择依据。

$$UCT(v') = \frac{Q(v')}{N(v')} + c\sqrt{2\frac{\ln N(v)}{N(v')}} \qquad (2\text{-}1)$$

表达式（2-1）中，v表示当前节点、v'是v的某个子节点，$N(v)$和$N(v')$表示访问节点v和v'的次数，$Q(v')$表示基于若干次蒙特卡洛模拟对节点v'的估值，c是一个大于 0 的参数、用于控制算法过程中探索未知过程的比例。

扩展（Expansion）：如果当前节点不是终止节点，则根据当前节点的合法动作添加一个或多个子节点；

模拟（Simulation）：从扩展出的新节点出发，基于某种策略（默认策略，Default Policy，通常为随机动作）进行模拟、直到游戏（或问题）结束，得到反馈价值；

更新（Backpropagation）：将模拟阶段得到的反馈价值，沿着路径自下而上的向上传播，更新沿途每个节点的估值信息。

通过不断执行上述的四步操作，蒙特卡洛树搜索便可以迭代地构建出一棵局部增长的不对称搜素树。在进行决策时，只需要根据需要从根节点选择最符合条件的节点对应的动作即可。迭代次数越大，根据大数定律、算法对于每个节点的估值便越准确，进而基于该流程得到的动作越趋近于最优动作。

2.3.2　蒙特卡洛树搜索在游戏 AI 领域的应用

相比于传统树搜索算法，蒙特卡洛树搜索最大的优点在于其不需要对游戏（或问题）有任何先验知识，只需要通过不断的模拟尝试便可以完成不错的决策；当人类经验主导的启发式函数出现偏差或不存在有效的启发式函数时，蒙特卡洛树搜索的优势便更加明显。另外，在解决复杂度较高、特别是分支因子较大的问题时，由于展开节点带来的计算量是呈指数级增长的，传统树搜索算法的时间开销也对应的指数级增加；而由于蒙特卡洛树搜索是选择性迭代加深的，并不会受到该问题的影响。

围棋是一款复杂度极高的游戏，19×19 规模的围棋游戏的状态复杂度约为 10^{170}；虽然拥有近四千年的历史，流传着各种各样的棋谱，但是计算机科学家们一直没有找到对于围棋有效的估值函数，基于传统树搜索算法的围棋 AI 程序也很难有所突破。而随着蒙特卡洛树搜索的提出，相关的研究进展也随之出现。Gelly 等人在围棋程序 MuGo 中结合简单的围棋规则对蒙特卡洛树搜索算法中的选择阶段进行了剪枝优化，并通过对模拟阶段的并行化提升了算法执行效率（Gelly et al., 2008）。MuGo 在 9×9、13×13 规模的围棋上都具有极强的棋力，蝉联了多次 CGOS (Computer GO Server) 的冠军；阿尔伯塔大学计算机围棋小组开发的基于蒙特卡洛树搜索的围棋程序 Fuego 可以在 9×9 规模的围棋上击败低段位的业余选手（Enzenberger et al. 2010）。除了围棋之外，通过结合少量游戏特定的领域知识，蒙特卡洛树搜索在军旗、亚马逊棋等游戏上也有不错的表现（Ciancarini et al., 2010; Lorentz, 2008）。

随着深度学习研究的不断进步，蒙特卡洛树搜索开始借助神经网络的表达能力对状态进行泛化。在某个游戏状态下，通过神经网络代替模拟阶段的默认策略、直接输出每个合法动作的期望收益值，结合大规模的采样数据进行训练对该期望估计值进行调整以得到准确的收益估值；Google DeepMind 便是秉承这一思路训练了 AlphaGo 和 AlphaGo Zero（Silver et al., 2016; Silver et al., 2017）。随着 2016 年 AlphaGo 以 4：1 的比分击败 “围棋第一人” 、韩国职业围棋手李世石，宣告了继国际象棋之后，围棋也被人工智能攻克。

基于 AlphaGo 和 AlphaGo Zero 的框架，Google DeepMind 在 2018 年提出了完全信息博弈棋盘游戏通用智能体 AlphaZero（Silver et al., 2018），通过自对弈训练的方式战胜了国际象棋、将棋、围棋在内的先前最强程序；2019 年又提出了能够战胜职业玩家的星际争霸 2 游戏智能体 AlphaStar（Vinyals et al., 2019），进一步探索蒙特卡洛树搜索+深度神经网络在非完全信息博弈、多智能体等领域的表现。

2.3.3　蒙特卡洛树搜索在游戏 AI 评估的研究工作

Lanzi 在 2019 年提出了一种基于蒙特卡洛树搜索的完全信息博弈游戏 AI 的评估方法（Lanzi, 2019），他定义蒙特卡洛树搜索算法训练游戏 AI 达到一定水平时所需的迭代

次数为蒙特卡洛树搜索复杂度（MCTS complexity）。以不同算法实现的 AI 作为目标进行训练，得到它的蒙特卡洛树搜索复杂度并以此为其水平的衡量。在衡量蒙特卡洛树搜索智能体与目标智能体是否达到同一水平时，Lanzi 引入了对局胜率和动作信息熵（Herbrich et al., 2007）两种方法进行判断。借助蒙特卡洛树搜索复杂度，他对屏风式四子棋、西非播棋（Awari）以及 6×6 规模黑白棋的几种算法进行了水平评估。

2.3.4 相关研究对本文工作的启发

蒙特卡洛树搜索在传统树搜索的基础上，使用蒙特卡洛方法进行模拟采样、进而得到动作的价值评估。因此算法的表现不仅仅与问题的搜索空间相关，其博弈树每个节点的价值分布都可能造成影响；而这种影响现有的复杂度分析方法是无法描述的。

由于不需要任何领域知识作为先验条件，蒙特卡洛树搜索理论上可以被应用到任何一个游戏问题；AlphaZero（Silver et al., 2018）的提出使得完全信息博弈游戏已经被人工智能解决。Lanzi 的工作提出了一种可以摆脱平台、竞赛甚至其他游戏 AI 智能体的游戏 AI 评估方法，但是其工作也只关注了完全信息博弈游戏。

综上所述，本文作者希望借助蒙特卡洛树搜索算法，在非完全信息游戏进行一些尝试；同时，尝试通过蒙特卡洛方法，从价值分布的角度对现有的游戏问题难度分析方法进行一定的补充。

2.4 斗地主游戏及其相关研究进展

本节对斗地主游戏的游戏规则进行介绍，并对其性质进行简单的分析，最后展示游戏 AI 在斗地主游戏的研究进展。

2.4.1 斗地主游戏规则

斗地主游戏使用一副 54 张的扑克牌，每人 17 张牌，留 3 张做底牌。按照出牌顺序进行叫牌，每人只能叫 1 次，叫牌时可以叫“1 分”、“2 分”、“3 分”和“不叫”。后叫牌者只能叫更高的分数或者不叫，叫牌结束后所叫分值最大的玩家为地主；如果所有玩家都不叫，则重新发牌并重新叫牌。将三张底牌交给地主，并亮出底牌让所有人看到。地主首先出牌，然后按照顺序依次出牌。轮到某人跟牌时，可以选择“不出”或出比上一个玩家更大的牌。当某人出完牌时本局游戏结束。如果是地主先出完牌则地主胜，否则另外两家（农民）胜。

斗地主游戏的牌型包括火箭、炸弹、单张、一对、单顺、双顺、三带、四带二、飞机、航天飞机。其中火箭最大，可以打任意其他的牌；炸弹比火箭小，比其他牌大，都是炸弹时按牌的大小比较。除火箭和炸弹外，其他牌型必须要牌型相同且总张数相同

才能比大小。牌的大小顺序是大王＞小王＞2＞A＞K＞Q＞J＞10＞9＞8＞7＞6＞5＞4＞3，不区分花色。一对、三带按牌的大小比较，顺牌按最大的一张牌的大小比较，飞机、航天飞机、四带二按照三顺或四张部分牌的大小比较、从牌不进行比较。

2.4.2　斗地主游戏的性质分析

斗地主游戏中，每个玩家都只能看到自己的手牌、牌桌上已打出的牌以及底牌情况、不能看到其他玩家的手牌；同时由于初始手牌的情况存在随机性，因此属于随机非完全信息博弈游戏。

斗地主游戏的状态空间较大，同时动作空间也很大、在不考虑花色的情况下共有30,723 种动作（Zha et al., 2019）。因此试图通过遍历整个状态空间进行搜索的方法是不可能的，必须依靠合理的剪枝、基于人类经验的专家系统或其他方法才有可能解决斗地主游戏。

2.4.3　斗地主游戏 AI 的研究进展

相比于德州扑克（Brown et al., 2018）、麻将（Li et al., 2020）等非完全信息博弈游戏，斗地主相关的研究工作并不是很多。Whitehouse 等人在 2011 年提出了一种简化的两人 18 张牌的斗地主游戏，并通过实验证明通过“作弊”获取完全信息的情况下，蒙特卡洛树搜索算法在该问题可以得到不错水平的模型（Whitehouse et al., 2011）。在这之后很长一段时间没有有关斗地主游戏的研究工作。

Liu 等人在 2018 年提出了一种基于大规模数据集训练的监督学习模型 DeepRocket（Liu et al., 2018）。他们划分出叫牌模块、打牌模块和跟牌模块。其中叫牌模块基于人类经验编码，打牌模块和跟牌模块是基于 800 万场高质量斗地主对局数据训练得到的卷积神经网络。DeepRocket 能够准确的预测出牌情况、实现与队友的合作，并达到了顶级人类业余玩家的水平。

Jiang 等人在 2019 年提出了一种基于 AlphaZero 框架（Silver et al., 2018）训练的模型 DeltaDou（Jiang et al., 2019）。他们首先使用基于人类经验的启发式算法产生近 20 万场对局，基于这些对局训练一个初始的策略网络，然后进行自对弈训练。当任意玩家手牌数量少于 15 张牌时通过贝叶斯模型推测每个玩家的手牌情况。DeltaDou 能够在与人类顶级玩家的对局中获胜。

2.5　Botzone 在线游戏 AI 对战平台

2.5.1　整体情况介绍

Botzone 在线游戏 AI 对战平台（Zhang et al., 2012; Zhou et al., 2017，2018，下文简

称为 Botzone 平台）是由北京大学人工智能实验室开发的在线程序对战平台，包含基于 RATE 生物特征识别评测平台（Xian et al., 2015）的 1.0 版本和基于 BotZoneBenchmark 在线动态基准测试模型实现的 2.0 版本，被广泛的应用于各类人工智能类竞赛和课程实践。在无特殊说明的情况下，本文中所指的 Botzone 平台均指基于 BotZoneBenchmark 在线动态基准测试模型实现的 Botzone 2.0 在线游戏 AI 对战平台。

Botzone 平台目前拥有 85 个小组、超过 8000 名用户，举办了包括 NOI 团体赛、北京大学游戏对抗赛、北京大学游戏对抗邀请赛、IJCAI 麻将人工智能比赛（正在进行中）在内的十余场比赛赛事，并被北京大学、北京邮电大学、北京交通大学等高校的相关课程用作教学工具使用。

Botzone 平台的游戏是与 AI 程序进行交互的环境，包括游戏规则逻辑和展示机制；Bot 是 Botzone 平台上用户提交的游戏 AI 程序，通常只适用于特定的游戏。Botzone 平台上目前有 27 款游戏、覆盖了常见的游戏类型，并在游戏人数、决策时机、随机性、对称性、信息完全性等方面各有不同。每个游戏都维护了一个该游戏所有 Bot 的排行榜（称为天梯排行榜），通过平台不断调度对局（称为天梯对局）并根据对局结果调整参与对局 Bot 的排名，使得每个 Bot 在排行榜中的位置能够相对客观地反映其水平。

Botzone 平台上的每个游戏都包含一个裁判程序，包含游戏的规则逻辑，并负责和平台完成交互。每场对局中，裁判程序并不会实时维护当前游戏的状态，而是通过初始状态和每轮各个 Bot 的决策输出还原当前状态。在每轮与 Bot 进行交互时，裁判程序也只会将以往的交互内容（初始状态和每轮各个 Bot 的决策输出）发送给 Bot，在收到决策输出后执行该决策并推进游戏发展。整个过程如图 2.2 所示。

自 2014 年上线以来，Botzone 平台每个游戏的每一场对局日志数据都会以 JSON 格式存储下来，并由系统定期进行打包、并对用户开放下载使用。图 2.3 展示的是 Botzone 平台斗地主的对局日志数据样例，用户可以从对局日志数据中提取出对局初始状态（publiccard，地主底牌）、每一轮某方 Bot 决策结果（response）等信息。截至本文撰写时，Botzone 平台已有超过 5.4 万个 Bot、进行了接近 200 万场对局。用户可以方便地下载这些对局数据，用于游戏 AI 相关的研究和实验工作。

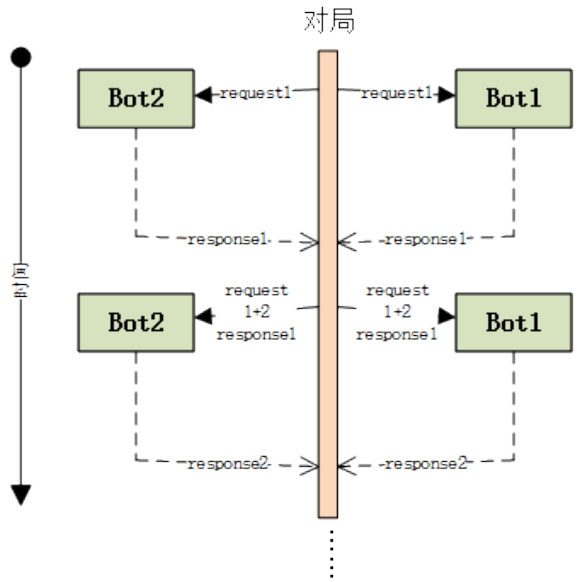

图 2.2 Botzone 平台 Bot 与裁判程序交互方式示意图


```
[
  {
    "keep_running":false,
    "memory":399,
    "output":{
      "command":"request",
      "content":{
        "0":{
          "publiccard":[
            7,
            16,
            0
          ],
          "own":[ ],
          "history":[ ]
        }
      },
      "display":{
        "allocation":[ ],
        "publiccard":[
          7,
          16,
          0
        ]
      }
    },
    "time":62,
    "verdict":"OK"
  },
  {
    "0":{
      "keep_running":false,
      "memory":394,
      "time":5,
      "verdict":"OK",
      "response":[
        0,
        2
      ]
    }
  },
```


图 2.3 Botzone 平台斗地主对局日志数据样例

不同游戏的 Bot 使用的算法往往不同，Botzone 平台上的游戏的常用算法包括专家系统、Alpha-Beta 剪枝、蒙特卡洛树搜索、遗传算法，部分游戏还会涉及到纳什均衡（俄罗斯方块、坦克大战）、强化学习（围棋、斗地主、麻将）等领域。

2.5.2 游戏 AI 的天梯排行榜评估方式

Botzone 平台的 Bot（即游戏 AI 程序）通过天梯排行榜进行排名与评估，即根据所有 Bot 的天梯分数和对局参与度，不断调度新的评测对局并根据结果实时更新 Bot 的分数，使得天梯排行榜能够逐渐收敛、从而通过排名反映出 Bot 的水平。

对于天梯分数，新加入天梯排行榜的 Bot 的分数为 1000 分。对局之后，参与的两个 Bot 会根据天梯等级分公式对分数进行更新。天梯的等级分公式以 Arpad Elo 提出的 ELO 为基础，在参考对局结果的同时还考虑了对战双方的水平差距。对于对局参与度，主要是 Bot 参与的天梯对局的数目。在此基础上，每次 Bot 的版本更新且长时间未参与新的对局会对该数据进行衰减。新加入天梯排行榜的 Bot 的对局参与度为 0。

除此之外，Botzone 平台还会根据每个游戏天梯排行榜的 Bot 数以及游戏的性质对实际的评估方式进行微调，以尽可能确保评估的公平性。

2.5.3 Botzone 平台的斗地主游戏及其评估方式

Botzone 平台的斗地主游戏与 2.4.1 小节介绍的游戏规则存在一些区别：

- 取消叫牌环节，Bot 位置直接决定游戏身份；发牌时，每个玩家得到 17 张牌，剩余 3 张牌公示后直接交给地主位置的 Bot；
- 一次评测参与 Bot 数为 2 个，包含两场对局，使用同一牌堆。两场对局地主和农民互换位置，每场对局控制 2 个农民的是同一个 Bot，但两个 Bot 独立运行、无法通信。

除此之外，在计算分数时 Botzone 平台的斗地主游戏除了考虑胜负关系，还引入了奖励分系统。具体的分数计算公式如表达式（2-2），表达式中的牌型权重见表 2.1

$$\text{总分} = \text{胜}\ ?\ 2 : 0 + \frac{\sum \text{牌型数目} \times \text{牌型权重}}{100} \qquad (2\text{-}2)$$

表 2.1 斗地主游戏牌型权重表（根据人类经验确定）

牌型	权重
单张	1
一对	2
单顺	6
双顺	6
三带	4
四带二	8
飞机	8
航天飞机（二连）	10
航天飞机（三连及以上）	20
炸弹	10
火箭	16

在评估方式上，基于上一小节介绍的 Botzone 天梯排行榜评估方式，Botzone 平台的斗地主游戏做了一些改进。对局双方的 Bot 会使用同样的一副牌堆进行两场对局，两场对局互换地主和农民的身份。每一场对局中，农民使用相同的 Bot、但相互之间不能交流。

截至本文撰写时，Botzone 平台的斗地主游戏天梯排行榜包括 341 个 Bot，排名分数区间为[-180, 1629]。图 2.4 给出了 341 个 Bot 实现语言的情况，其中 309 个为基于 C++语言实现的、35 个为基于 Python 语言实现的。

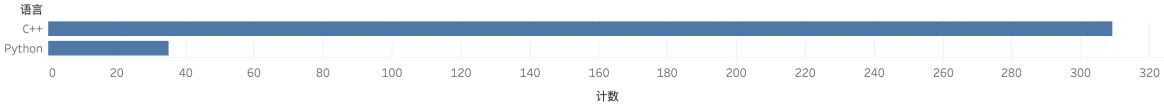

图 2.4 Botzone 平台斗地主游戏 Bot 实现语言情况

2.5.4 局限性和斗地主游戏 AI 评估方式的不足

作为一个中心化、在线动态的基准测试平台，Botzone 平台具有一些天生的局限性。Botzone 平台的对局主要分为小组内安排的比赛对局、用于维护天梯排行榜的天梯对局以及自定义生成的对局。前两类属于系统生成的对局，最后一类目前需要通过用户在前端界面执行一系列的操作来生成、相对来说比较麻烦。同时受限于有限的评测服务器资源，对局的评测是存在一定程度时延的。也就是说如果希望在现有的 Botzone 平台上，根据自己的需要产生一定条件的对局，是相对复杂和繁琐的，同时还可能花费一定的时间等待结果的产生。

Botzone 平台上每个游戏的裁判程序都是公开的，但是在用户实际使用中仍有一些

不足：如本文在上一小节中所描述的，裁判程序不实时维护当前游戏状态、每次都需要通过初始状态和每轮各个 Bot 的决策输出还原；裁判程序中游戏规则逻辑的代码是和与 Botzone 平台进行交互的代码耦合在一起的。例如斗地主游戏的裁判程序共有 573 行，其中涉及到与 Botzone 平台进行交互的代码占 175 行。除此之外，裁判程序的实现语言也并不统一、包括了 C++、Python 和 JavaScript。这些都会增加用户理解裁判程序和游戏规则逻辑的负担。另外部分裁判程序存在逻辑错误的情况。例如斗地主游戏的裁判程序会将正确的“四条加两只”牌型识别为违规牌型，将违规牌型识别为“飞机”牌型。

开放下载的对局日志数据中，只包含每次对局的初始状态（如存在）以及各个 Bot 在每轮的决策输出，因此在实际使用时仍需要借助规则逻辑相关的程序还原决策时的游戏状态、无法做到真正与 Botzone 平台解耦。

在斗地主游戏 AI 评估方式上，Botzone 平台需要依赖其他 Bot 的共同参与才可以完成天梯排行榜的评估；同时受限于 Botzone 平台的限制，无法对基于大型模型的 Bot 进行评测。除此之外，Botzone 平台斗地主游戏的天梯排行榜难以收敛。图 2.5 展示了 2020 年 4 月 21 日、5 月 1 日和 5 月 18 日的斗地主游戏天梯排行榜前 10 名的情况，在天梯排行榜 Bot 没有更新的情况下，前 10 名的 Bot 排名一直没有收敛。对于排名没有收敛的 Bot，只能知道其大概的水平、不能得到一个确定的排名。该现象可能与斗地主游戏本身存在的随机性有关，使得评估时难以获得 Bot 的真实水平。

排名	Bot 名	排名分
1	这是真的sample	1606.09
2	FTL_V1_0	1570.77
3	反冲机	1569.17
4	纯c斗地主	1568.00
5	写bug到凌晨	1559.79
6	训练机	1550.47
7	知世就是力量	1548.79
8	下个Bot见	1535.76
9	date201858	1531.07
10	人工zz	1527.34

(a)

排名	Bot 名	排名分
1	知世就是力量	1639.15
2	这是真的sample	1600.58
3	下个Bot见	1598.49
4	纯c斗地主	1563.65
5	训练机	1554.69
6	test	1550.69
7	反冲机	1546.63
8	FTL_V1_0	1531.43
9	人工zz	1531.35
10	nobody_knows_why	1522.91

(b)

排名	Bot 名	排名分
1	知世就是力量	1599.21
2	这是真的sample	1584.30
3	下个Bot见	1582.65
4	反冲机	1582.56
5	test	1571.72
6	FTL_V1_0	1568.47
7	纯c斗地主	1562.09
8	nobody_knows_why	1523.46
9	date201858	1521.19
10	训练机	1516.27

(c)

图 2.5 Botzone 平台斗地主游戏天梯排行榜前 10 名情况。(a) 4 月 21 日；(b) 5 月 1 日；(c) 5 月 18 日

2.6　本章小结

本章从以下几个方面对本文的相关工作进行了介绍：游戏问题的复杂度分析方法、游戏 AI 智能水平的评估方法、蒙特卡洛树搜索算法、斗地主游戏以及 Botzone 平台。基于本章的介绍，可以发现在游戏问题的难度评估、游戏 AI 智能体水平的评估上，现有的方法还存在一定的不足和提升的空间。基于蒙特卡洛树搜索的算法在完全信息博弈游戏上已经取得了极好的效果，但在包括斗地主游戏在内的非完全信息博弈游戏中还没有比较显著的成绩。除此之外，作为国内成熟的在线游戏 AI 对战平台，Botzone 平台仍存在一些问题值得改进。本文接下来将针对这些问题展开讨论。

第三章　Botzone 平台的扩展：本地模拟环境

本章介绍本文的实验环境、Botzone 在线游戏 AI 对战平台的扩展——本地模拟环境 PyBotzone。首先结合 Botzone 平台的现有局限性说明 PyBotzone 的设计背景，然后阐述其设计以及实现中的重点与难点，接着介绍 PyBotzone 的运行方式与工作流程，最后对其与 Botzone 平台的关联性进行介绍，并分析其为斗地主游戏 AI 的研究提供了怎样的便利。

3.1　设计背景

Botzone 在线 AI 对战平台提供了一种动态在线的游戏 AI 评测方法：用户提交自己的游戏 AI 程序，借助 Botzone 平台的天梯对局机制得到对应的排名情况，达到评估游戏 AI 的目的。Botzone 平台的所有游戏裁判程序和对局数据集都面向所有人开放下载和使用，部分 Bot（即游戏 AI 程序）也是开源的。这些资源的开放使得用户可以在本地进行游戏AI的研究、开发和评估工作，对游戏领域人工智能的发展有着极大的帮助。

但是如本文 2.5.4 小节所描述，受限于平台的运行机制及交互方式，Botzone 平台游戏的裁判程序不仅仅包含游戏的规则逻辑，还有与平台交互及游戏状态还原的代码，这使得裁判程序与 Botzone 平台有着极强的耦合性；另外裁判程序的实现语言比较多元，包括了 C++、Python 和 JavaScript。在易用性上与 OpenAI Gym （Brockman et al., 2016）、ALE （Bellemare et al., 2013）等规则逻辑代码简洁、对外 API 接口统一封装的常用框架有着一定的差距。同样受限于与 Botzone 平台的耦合性，用户开源的游戏 AI 程序同样存在着摆脱 Botzone 平台便无法运行的问题。

另外虽然 Botzone 平台开放了全部对局数据集的下载，但是与 MuJoCo （Todorov et al., 2012） 等同样提供数据集的工具相比，Botzone 平台存储的对局数据仅包括每场对局的初始参数和所有动作决策结果。用户在实际使用时如果需要每次决策时的游戏状态，需要根据初始参数和所有动作决策结果逐一还原才可以。

Botzone 平台本身是为了进行游戏 AI 的在线动态评测而设计与开发的，在设计时有所取舍、使得本地使用时不太方便也是正常的。但是随着近年来深度学习（LeCun et al., 2016）、强化学习（Sutton et al., 2018） 研究的兴起，对于大规模高质量数据集、数据高效采样的需求愈发强烈。Botzone 平台作为一个拥有近 30 种游戏、200 万场游戏对局数据的平台，也应当顺应潮流、在不影响本身功能的情况下增加对本地功能的支持。

本文作者基于以上考虑设计了 Botzone 在线游戏 AI 对战平台的本地辅助工具，使用目前人工智能研究领域使用最广泛的 Python 语言进行实现，并命名为 PyBotzone（Python Botzone）。其特点包括：

- 完全使用 Python 语言实现，游戏间提供完全统一的 API 接口进行交互；
- 可本地对 Botzone 平台游戏进行模拟，游戏规则逻辑与 Botzone 平台完全相同，同时可快速调整游戏规则逻辑以适应不同的使用需求；
- 可模拟 Botzone 平台在线对局方式在本地进行对局和评测，支持指定对局的双方或多方 Bot；
- 可基于 Botzone 平台开放的对局日志数据集直接生成状态-动作对数据集，用于对手建模（Laviers et al., 2009）、模仿学习（Ho et al., 2016）等相关研究。

3.2 设计与实现

3.2.1 本地模拟环境的设计

PyBotzone 作为 Botzone 平台的本地辅助工具，在本地模拟功能上既要保证交互方式的简单性和统一性，还需要尽可能维护一个简洁的游戏规则逻辑。如图 3.1 所示，Botzone 平台裁判程序每次的输入都是对局初始参数、历史动作决策和当前动作决策的集合，需要先恢复当前游戏状态后再执行动作合法性校验、推进游戏发展、判断游戏是否终止等后续逻辑。再加上与平台交互的逻辑代码，裁判程序作为游戏规则逻辑就显得十分“臃肿”。

造成这个问题的主要原因是因为裁判程序是在每次需要判断 Bot 的动作决策是否合法时才被 Botzone 平台调度执行，执行完成后立即退出、无法保存任何信息。在 PyBotzone 的设计中，游戏规则逻辑相关的程序会一直运行，游戏每个时刻的状态都存储在内存中，因此无需恢复操作。PyBotzone 的游戏运行机制如图 3.2 所示，除非明确的选择退出，否则游戏会一直处于运行状态。因此在游戏运行的任意时刻，都可以直接通过接口获得当前游戏的状态。同时由于可以直接与游戏进行交互，类似于裁判程序中的交互逻辑也被省去，游戏规则逻辑部分显得比较简洁、清晰。

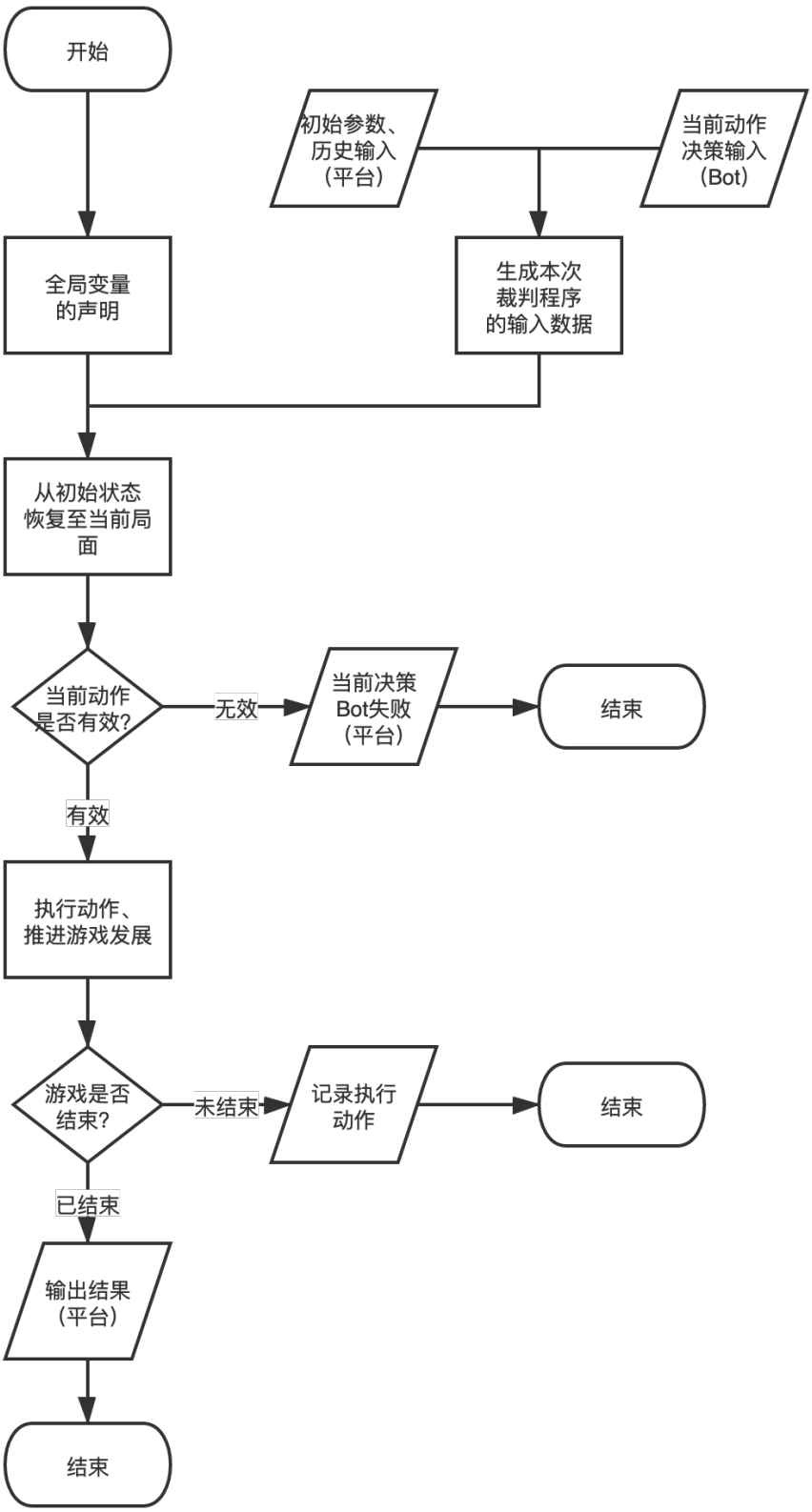

图 3.1 Botzone 平台裁判程序的运行流程

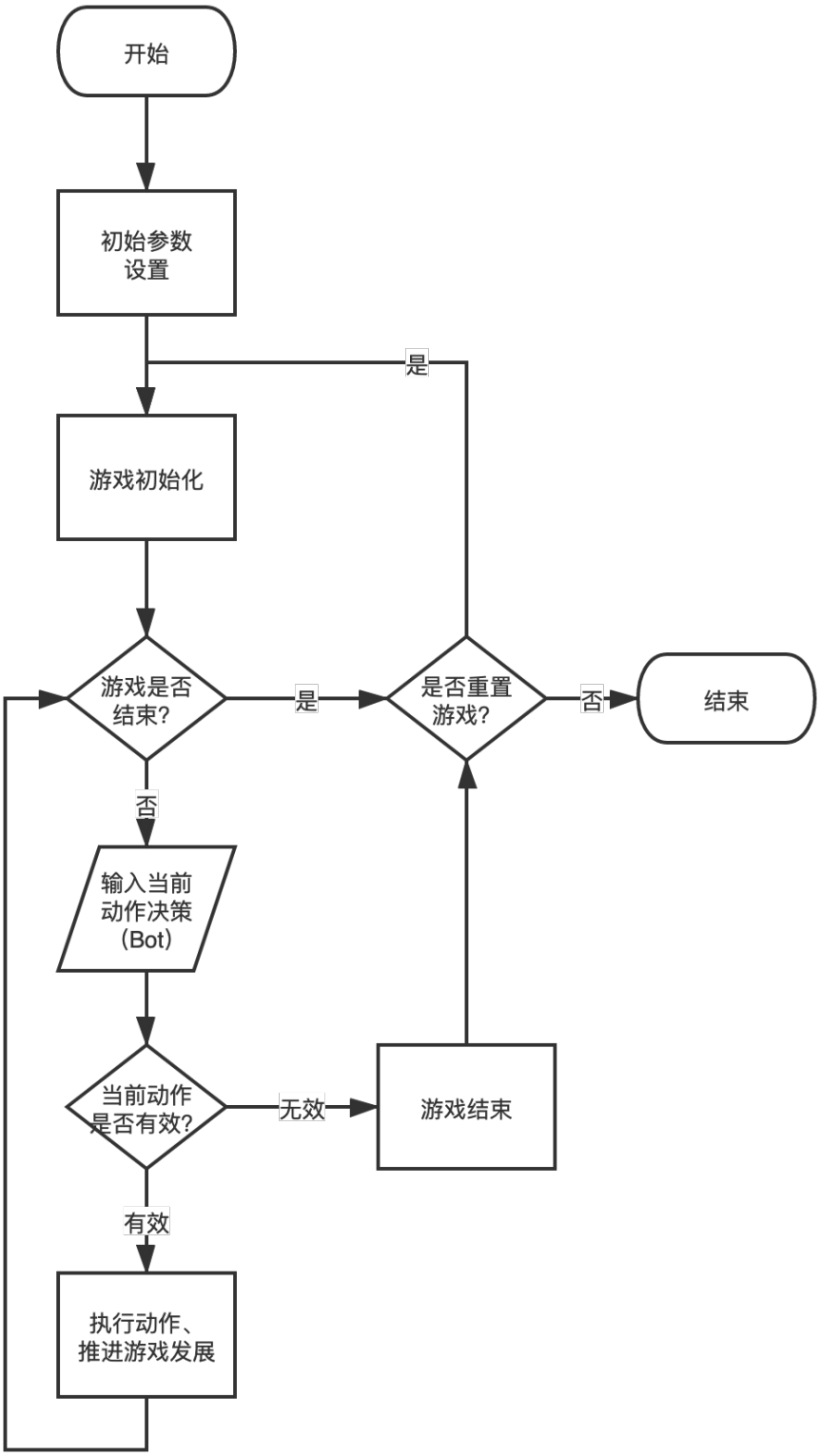

图 3.2 PyBotzone 游戏的运行机制

3.2.2　与 Botzone 平台兼容的设计

如本文先前所述，Botzone 平台公开的对局数据集和开源的 Bot 与平台有着极强的耦合性：对局数据集的本质是每场对局的平台日志，仅记录了每场对局的初始参数和每轮的动作决策；Bot 是提交至平台的游戏 AI 程序，虽然本身是无状态的，但是也需要仿照裁判程序完成游戏状态恢复的操作。为了实现在简化用户交互复杂度的同时，使得对局数据集和 Bot 能在本地使用，PyBotzone 必须在内部封装对 Botzone 平台的兼容。

Botzone 平台与 PyBotzone 游戏的最大区别就是是否保存状态：Botzone 平台的游戏是无状态的，通过输入的初始参数和历史动作决策还原到即将决策的状态；PyBotzone 的游戏是有状态的，可以实时获取当前的状态。因此只需要在现有 PyBotzone 本地模拟环境的基础上，记录下初始参数和每次决策的动作、每次向 Bot “询问”动作决策时将对应的历史数据一起传递即可。

对于对局数据集的处理也采用类似的方法：对于每场对局数据，首先将初始参数传递给 PyBotzone 的游戏，然后在每轮的动作决策执行前获取当前的游戏状态、与即将执行的动作形成状态-动作对进行存储，然后游戏继续推进即可。这样便实现了对局日志数据集向对局状态-动作对数据集的转换。

3.2.3　其他设计与实现的重难点

在 PyBotzone 的实现中，很大的一部分工作是处理 Botzone 平台游戏的裁判代码，即去除与平台进行 JSON 交互部分和恢复游戏状态的代码。同时将执行动作统一为一个接口，对于五子棋、黑白棋等需要 Bot 依次进行决策的游戏，传递的便是选择的动作决策；对于贪吃蛇、吃豆人等所有 Bot 同时进行决策的游戏，需要在收集齐所有动作决策之后将其一起传入，游戏将按照 Bot 顺序依次执行这些动作。除此之外，PyBotzone 在实现中还考虑了并行性能与对局数据记录两方面：

并行性能：随着近年来深度强化学习的兴起（Silver et al., 2018; Vinyals et al., 2019），对于数据采样效率的要求越来越高。同时计算机全面进入多核时代，模拟环境的并行性便成为了自然而然的要求。为了实现在单机时利用本机的多核计算资源、集群时利用分布式系统的资源，PyBotzone 选择了 Ray （Moritz et al., 2017）实现并行。

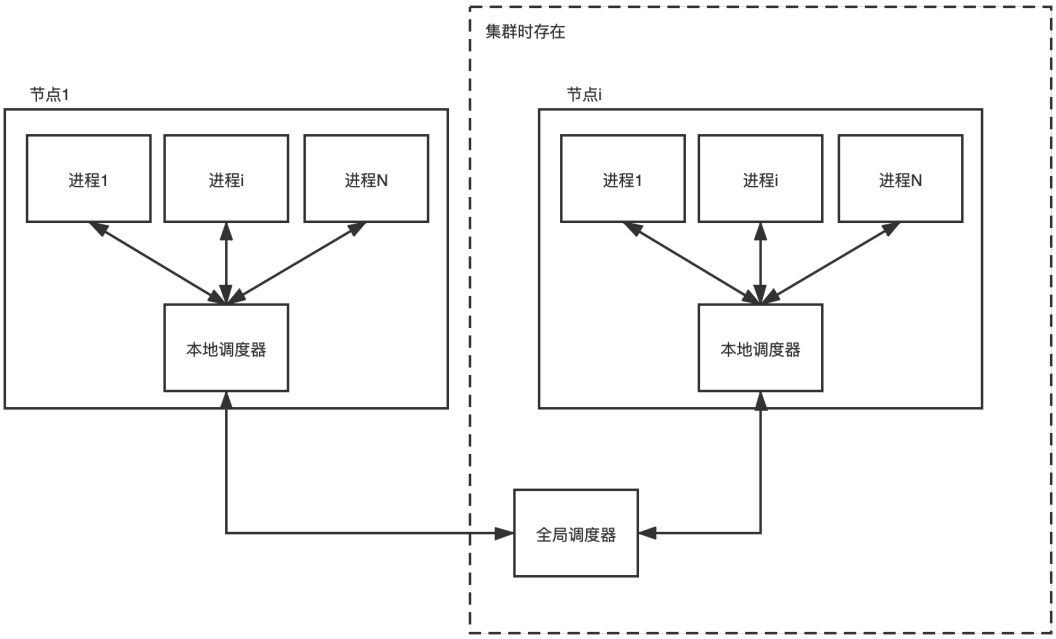

图 3.3 PyBotzone 实现的基于 Ray 的并行机制

Ray 本身是基于远程过程调用协议（Remote Procedure Call Protocol, RPC）实现的分布式框架，具备可伸缩性和动态负载平衡，同时对于原本代码的侵入度极低，非常适用于人工智能领域的并行化。如图 3.3 所示，当 PyBotzone 需要并行模拟时，会优先基于本地调度器在本地的多进程下执行；如果需要并行的任务较多且处于集群状态时，会将多出来的任务发送至集群的全局调度器、由其分配至集群中的其他节点用于执行并最终返回结果。

基于以上的结构，只要（本机或集群）存在可用的计算资源，PyBotzone 便会将需要执行的任务发送并执行，达到了充分利用计算资源的目的。PyBotzone 使用串行、进程池并行和 Ray 并行三种方式在 16 核单机上模拟 10000 次五子棋对局的耗时实验结果见表 3.1。

表 3.1 16 核单机模拟 10000 次五子棋对局的耗时（时间单位：秒）

实验方法	**串行**	**进程池并行（16 进程）**	**Ray 并行**
实验次数	10	10	10
平均耗时	136.38	25.81	29.95

虽然 Ray 并行方法的平均耗时略高于进程池并行方法，但是远低于简单的串行方式。同时考虑到 Ray 对于分布式集群的可扩展性支持以及代码极低的侵入度，其并行性能是可接受的。

对局数据记录：由于 PyBotzone 的游戏执行时记录着游戏状态，因此在每次执行 Bot 的动作决策时都可以得到其状态-动作对。随着模拟的不断进行、对局数据实时记录至内存，并在达到一定大小的情况下存储至硬盘。得到的数据集经过简单的预处理即可应用到其他研究中。

3.3　工作流程

3.3.1　本地对局的运行流程

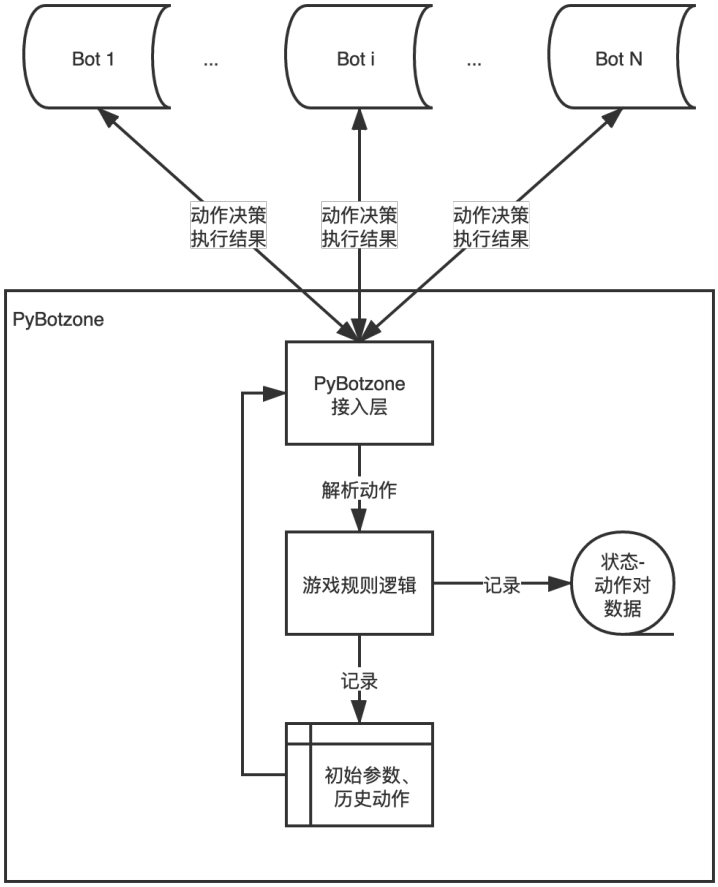

图 3.4 本地对局运行流程

本地对局是指借助 PyBotzone 在本地执行某个（或某些）Botzone 平台开源的 Bot 之间的对局。首先需要将待执行对局的 Bot 进行编译、生成可执行的二进制文件供 PyBotzone 调用。对局初始化后，PyBotzone 的接入层会将内存中存储的初始参数、历史动作决策数据组成请求，通过管道发送给对应的 Bot，并通过管道接收 Bot 的回应、即决策结果。游戏规则逻辑执行动作、记录该动作和状态-动作对并推动游戏发展，循环该流程直至游戏结束。最终输出对局结果，即获胜的 Bot。

3.3.2 对局日志数据集的处理流程

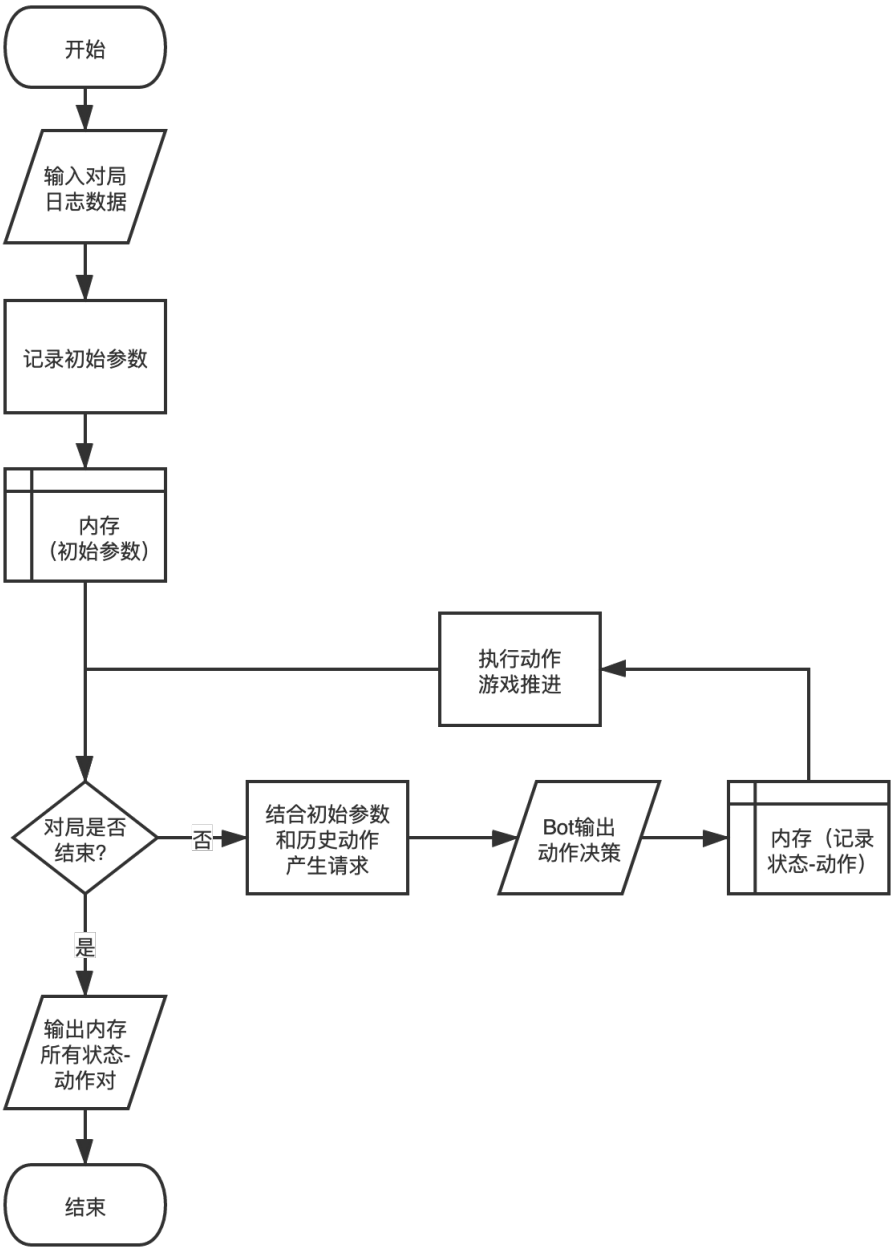

图 3.5 单场对局日志数据的处理流程

Botzone 平台开放的对局数据是以平台日志的形式存储的，以月为单位、每 100 场对局为一个文件。处理时首先记录初始参数，然后在对局未结束时（日志数据未处理完或没有异常结果），不断地组合初始参数及历史动作决策信息形成请求、发送给即将执行决策的 Bot，得到其结果后记录动作和状态-动作对。由于是真实的对局数据，不需要校验动作的合法性、直接执行动作推进游戏发展即可。在对局结束后，将内存中的所有状态-动作逐个输出即可完成单场对局日志数据的处理流程。

3.4　与 Botzone 平台的关联性

PyBotzone 实现的游戏逻辑规则来自于 Botzone 平台对应游戏的裁判程序，仅去除了对于本地模拟来说冗余的平台交互及状态恢复逻辑，核心逻辑没有任何改动，可以认为 PyBotzone 的游戏与 Botzone 平台的游戏是完全一致的，因此如果完全基于 PyBotzone 本地训练得到某个游戏的 AI 程序，在补充平台交互及状态恢复部分逻辑后即可直接提交至 Botzone 平台用于在线评估。这使得基于与环境交互的强化学习算法（Lillicrap et al., 2015; Schulman et al., 2017）的训练变得更加便捷。而 PyBotzone 提供的将对局数据集转化为状态-动作对数据集的功能，使得行为克隆（Bojarski et al., 2016）为代表的深度学习方法可以将更多精力投入到方法本身。

反过来，结合 Botzone 平台游戏的天梯排行榜和开源的 Bot，PyBotzone 可以在本地快速、集中的执行指定 Bot 间的对局以达到评测的目的。例如希望快速评估自己的 Bot 与天梯排行榜前 10 名已开源的 Bot 水平，就可以下载并编译这些 Bot 的代码，在本地快速的执行若干场与它们的对局来得到一个结果。这样的执行效率比将 Bot 提交至天梯等待天梯对局评测更高，比申请建立小组单独安排比赛更容易。

3.5　为斗地主游戏 AI 研究提供的便利

Botzone 平台本地模拟环境 PyBotzone 的实现为斗地主游戏 AI 研究提供的便利主要包括以下几点：

- 游戏规则逻辑与 Botzone 平台解耦：原先的斗地主游戏裁判程序是针对 Botzone 平台编写的，包含大量与平台进行交互的逻辑以及输出相关信息的代码，整体可读性较弱。PyBotzone 的实现中，以游戏规则逻辑为核心、去掉了所有无关的代码逻辑，解除了原先裁判程序与 Botzone 平台的强耦合关系，同时大大提升了代码的可读性；
- 数据模拟采样效率提升：斗地主游戏在计算当前状态合法动作时需要枚举所有手牌组合并判断其有效性，该过程不可避免且十分消耗时间，导致游戏的模拟速度较慢、无法有效的进行数据采样。PyBotzone 提供的并行功能可以有效改善这一问题、提升模拟速度和采样效率；
- 天梯排行榜 Bot 可本地使用：由于 Bot 与 Botzone 平台强耦合，斗地主游戏天梯排行榜上的几百个 Bot 无法脱离平台进行使用。通过 PyBotzone 对于 Botzone 平台的兼容性设计，使得这些 Bot 无需任何改动便可以在本地运行、为斗地主游戏 AI 的研究提供了大量的基准程序。

3.6 本章小结

本章介绍了Botzone在线AI对战平台的扩展——本地模拟环境PyBotzone。Botzone平台作为一个游戏AI的在线动态评测平台，在本地使用时存在与线上耦合性强、易用性较差等问题。为了解决这些问题，本文作者设计并实现了PyBotzone，使得使用者可以在本地进行Botzone平台游戏的模拟、指定Bot间的本地快速对局以及对局数据集的处理，令一些游戏领域人工智能研究的最新方法的应用成为可能。通过一些专门的处理与优化，PyBotzone在并行性能和对局数据记录方面有着较为突出的表现。本章的工作为游戏AI的研究提供了诸多便利，也为本文后续实验的顺利开展奠定了良好的基础。

第四章　斗地主游戏问题的蒙特卡洛复杂度分析

本章对斗地主游戏的难度进行分析。首先使用传统的复杂度分析方法，从状态复杂度、博弈树复杂度的角度描述斗地主游戏问题搜索空间的相关性质；然后探讨初始手牌带来的随机性对于斗地主游戏的影响；接着提出蒙特卡洛复杂度分析方法，从价值分布的角度对斗地主游戏问题进行刻画；最后，结合实验结果对斗地主游戏的难度进行分析。

4.1　斗地主游戏问题的传统复杂度分析

首先使用传统的游戏复杂度分析方法对斗地主游戏进行分析，通过状态复杂度和博弈树复杂度对其搜索空间的性质进行探究。

4.1.1　状态复杂度

斗地主游戏中共有 54 张牌，发牌、叫牌结束后地主玩家手牌数为 20 张、其中 3 张为需要像农民明示的底牌，农民甲和农民乙两名玩家手牌数各为 17 张，因此初始状态就有$C_{54}^{20} \cdot C_{20}^{3} \cdot C_{34}^{17} \cdot C_{17}^{17}$种。每张牌之后有可能在玩家的手牌中，也可能被玩家打出、在牌桌上，因此状态数上限的计算公式为：$C_{54}^{20} \cdot C_{20}^{3} \cdot C_{34}^{17} \cdot C_{17}^{17} \cdot 2^{54} \approx 10^{43}$，对应的斗地主游戏的状态复杂度估值也就是$10^{43}$。

4.1.2　博弈树复杂度

博弈树复杂度的计算通常使用平均分支因子法进行估计，即平均总步数与平均分支因子的乘积。对于斗地主游戏，这两项数据目前还没有论文、数据集等公开发表的工作可供参考。此处借助第三章介绍的本地模拟环境中的斗地主游戏进行估计：通过不断随机选择动作的方式进行斗地主游戏，记录每场游戏的决策次数以及每次决策的合法动作数；由此可以得到斗地主游戏的平均对局长度，并通过每次决策合法动作数之和与总决策次数得到平均分支因子数。经统计得到斗地主游戏的平均对局长度为 71.6，平均分支因子数为 14.6。

由于每个玩家无法观察到其他玩家的手牌情况，在构建博弈树时为了计算其他玩家的合法动作，需要对他们的手牌情况进行枚举，并使用机会节点（Russell et al., 2003）的方法构建博弈树。以地主玩家为例，除去手牌中已知的 20 张牌，需要枚举剩余 34 张牌的分布，因此有$C_{34}^{17} \cdot C_{17}^{17}$种可能；对应的机会节点下就有$C_{34}^{17} \cdot C_{17}^{17}$个分支。斗地主游戏初始状态下每个玩家的手牌情况是随机的，且每种手牌情况对应一棵博弈树的根节点。上一小节已经分析得到斗地主游戏的初始状态数是$C_{54}^{20} \cdot C_{20}^{3} \cdot C_{34}^{17} \cdot C_{17}^{17}$。因此可以得

到对博弈树复杂度的一个粗略估计：$(C_{54}^{20} \cdot C_{20}^{3} \cdot C_{34}^{17} \cdot C_{17}^{17}) \cdot (C_{34}^{17} \cdot C_{17}^{17}) \cdot 15^{72} \approx 10^{125}$。

4.1.3 结果分析

根据前面两个小节的计算，斗地主游戏问题的状态复杂度为10^{43}、博弈树复杂度的估计为10^{125}；状态复杂度略低于国际象棋（10^{46}），而博弈树复杂度略高于国际象棋（10^{123}）（Allis, 1994）。由此可以认为斗地主游戏是一个非常复杂且难度较高的游戏问题。

斗地主游戏的难度还体现在它是三人博弈问题，两个农民是合作关系而农民和地主之间是对抗关系。初始的叫牌环节也增加了它的难度。不过这些不是本文研究的重点。

4.2 初始手牌对评估的影响

斗地主游戏具有很强的随机性，其随机性主要体现在初始手牌的分配上。初始手牌会很大程度影响对局的结果，从而对难度以及游戏 AI 的评估造成影响。因此在对斗地主游戏的难度做进一步分析前，需要对其随机性对评估可能带来的影响进行讨论。本节提出一种对于斗地主游戏初始手牌倾向性的定义，并给出其对斗地主游戏影响的分析。然后基于 Botzone 平台的斗地主游戏 Bot 提出一种评估初始手牌倾向性的方法，并以此计算斗地主游戏中初始手牌倾向性的分布情况，并结合相关实验结果进行分析。

4.2.1 初始手牌倾向性的定义

对于国际象棋、围棋等不存在随机性的游戏，游戏 AI 的评估方法通常是将训练得到的程序与该游戏先前的最强程序（Silver et al., 2018）或人类顶级玩家（Silver et al., 2016）对局进行评估。而斗地主游戏中存在的随机性会极大程度地影响对局结果，导致对局结果不能客观反映对局双方真实水平的情况，进而影响评估效果。以斗地主游戏中的极端情况为例：地主玩家拿到的初始手牌是“航天飞机（三连）”牌型加“火箭”牌型，例如“333344445555678910J 小王大王”。那么只要地主玩家的 Bot 能够识别出“航天飞机”和“火箭”牌型，不论是先“火箭”后“航天飞机”还是先“航天飞机”后“火箭”，地主玩家都肯定能够获得本次对局的胜利，但这与农民玩家 Bot 的水平完全没有任何关系。同样的，这样极端的初始手牌情况也会极大的影响对游戏难度的评估。虽然这样的必胜牌型出现概率极低（约为亿分之一），但是地主或农民存在非常容易获胜的初始手牌的情况也是客观存在的。这样的初始手牌的出现完全符合斗地主的游戏规则，但是它使得对局的胜负与玩家（或 Bot）本身的水平没有太大关系，对于评估和区分游戏 AI 智能体会造成很大影响。

为了更加有效地评估斗地主游戏的难度和斗地主游戏 AI 智能体的水平，避免前文所描述的类似于“航天飞机（三连）+火箭”的牌型干扰，这里尝试通过定义斗地主游戏初始手牌的倾向性以描述上述的问题。

称斗地主游戏完成 54 张牌的分配后、地主出牌前三个玩家的手牌的集合为初始手牌。对于一副初始手牌，如果其存在某种当前技术下能够发现的动作决策序列使得地主能够确保取得胜利，称这幅初始手牌是倾向于地主的；反之称其为倾向于农民的。如果一副初始手牌既不倾向于地主也不倾向于农民，则称其为没有倾向性的初始手牌。

对于任意一副初始手牌，通过极大极小搜索遍历整个状态空间一定能够获得一个地主必胜动作序列或农民必胜动作序列，即一副初始手牌一定是倾向于地主或倾向于农民的。但是这里需要强调的是当前技术能够发现的，即通过一定深度的搜索便可以找到的必胜动作序列；如果某个必胜动作序列需要极其复杂的推理或极深的搜索才可以获取到，此处并不认为其具有倾向性。

4.2.2　初始手牌倾向性的计算方法

Botzone 平台的斗地主游戏规则与传统规则存在着一定差别，主要体现在无叫牌环节和分数计算上引入牌型权重两方面。虽然 Botzone 平台在评测时只有两个 Bot，且农民位置是由同一个 Bot 控制的，但是通过农民间不可通信的规则设定使得评测效果与传统规则没有区别。

在 Botzone 平台规则下，Bot 的位置顺序直接决定了他的角色以及初始手牌情况，这使得 Bot 缺少了叫牌功能。在传统斗地主规则下除去叫牌相关的规则逻辑，初始手牌的情况与 Botzone 平台规则并没有区别：2 个农民各持 17 张牌，地主持 20 张牌且其中 3 张是明牌。虽然斗地主游戏具有很强的随机性、初始手牌会很大程度地影响单场对局结果，但 Botzone 平台在评测时通过交换手牌进行两场对局的方式消除随机性对评估的影响。因此无叫牌环节的规则除了令 Bot 没有叫牌能力之外，并不会影响其打牌的水平、更不会更擅长打地主牌或农民牌。

分数计算引入牌型权重可能会催生一批激进型 Bot：面对无望取胜的局面，通过拆散合理牌型、拼凑高权重牌型以获得更多的奖励分。但是根据表达式（2-2），单场对局的分数仍旧以胜负关系为主。牌型权重带来的奖励分只会在一次评测双方 Bot 均为一胜一负的情况下对双方进行区分。因此 Botzone 平台规则下的 Bot 仍旧是以胜利为首要目标而设计的，在传统规则下对局时也不会影响其水平。

综上所述，除去叫牌部分，Botzone 平台斗地主游戏 Bot 是可以迁移至传统规则下进行游戏和评估的。天梯排行榜上 Bot 的排名也可以映射到传统规则下，并认为在传统规则下相对名次也是和天梯排行榜相同的。所以可以通过天梯排行榜上的 Bot 对传统规则的斗地主游戏的性质进行研究工作。

基于上述的分析和结论，此处提出初始手牌倾向性的计算方法。令两个 Botzone 平台天梯排行榜排名差距很大的 Bot 使用一副初始手牌进行对局，然后交换地主农民身份、使用相同的初始手牌再次进行对局，如果两次对局都是地主获胜，则认为该初始手牌倾向于地主；反之认为倾向于农民。如果两次对局都是某一个 Bot 获胜，则认为该初始手牌没有倾向性。

即对于一副初始手牌$InitCard$，通过两组 Botzone 平台斗地主游戏 Bot 集合$Bots_{top}$和$Bots_{bottom}$来评估其倾向性$Tendency$。其中$Bots_{top}$和$Bots_{bottom}$分别表示斗地主游戏天梯排行榜上排名靠前的 Bot 集合和排名靠后的 Bot 集合。算法 1 给出了该评估方法的伪代码。

1 基于 Botzone 平台斗地主游戏 Bot 的初始手牌倾向性计算方法

函数：$EvaluateTendencyOfInitCards$

输入：$InitCard$，$Bots_{top}$，$Bots_{bottom}$

输出：$Tendency$

从两组集合中各随机选取一个 Bot $\forall b_{top} \in Bots_{top}, \forall b_{bottom} \in Bots_{bottom}$

以b_{top}为地主、b_{bottom}为农民，$InitCard$为初始手牌进行一次对局，得到胜者w_1

以b_{bottom}为地主、b_{top}为农民，$InitCard$为初始手牌进行一次对局，得到胜者w_2

if $w_1 \neq w_2$ **then**

  if $w_1 = b_{top}$ and $w_2 = b_{bottom}$ **then**

    $Tendency =$ 倾向地主

  else

    $Tendency =$ 倾向农民

  end if

else

  $Tendency =$ 无倾向性

end if

return $Tendency$

显然无倾向性的初始手牌对于斗地主游戏难度以及斗地主游戏 AI 智能体水平的评估是有效的，而斗地主游戏初始手牌的可能性有$C_{54}^{20} \cdot C_{20}^{3} \cdot C_{34}^{17} \cdot C_{17}^{17} \approx 8 \times 10^{26}$种，如果其中有很大一部分情况都具有倾向性，那么基于对局的游戏 AI 智能体评估方法效率就会很低、需要手工筛选一些无倾向性的初始手牌进行评估；如果倾向地主与倾向农民的概率是不同的，完全无筛选初始手牌的评估甚至会产生错误的结果。因此需要对倾向性的分布有所计算。

基于算法 1 提出的$EvaluateTendencyOfInitCards$初始手牌倾向性计算方法，可

以得知某副确定初始手牌的倾向性。那么通过不断从8×10^{26}种可能里枚举初始手牌的情况并进行倾向性的评估，便可以对其倾向性的分布有所估计。算法 2 给出了该方法的伪代码。

2 斗地主游戏初始手牌倾向性分布的评估方法

函数：$EvaluateTendencyDistribution$

输入：$Bots_{top}$，$Bots_{bottom}$，δ

输出：p_{none}，$p_{landlord}$，$p_{peasants}$

$n_{none} \leftarrow 0$，$n_{landlord} \leftarrow 0$，$n_{peasants} \leftarrow 0$

$p_{none} \leftarrow 0$，$p_{landlord} \leftarrow 0$，$p_{peasants} \leftarrow 0$

repeat

随机生成一副初始手牌$initCard$

$tendency = EvaluateTendencyOfInitCards(initCard, Bots_{top}, Bots_{bottom})$

if $tendency =$ 无倾向性 **then**

$n_{none} \leftarrow n_{none} + 1$

else if $tendency =$ 地主 **then**

$n_{landlord} \leftarrow n_{landlord} + 1$

else

$n_{peasants} \leftarrow n_{peasants} + 1$

end if

$\Delta p_{none} \leftarrow |p_{none} - n_{none}/sum(n_{none} + n_{landlord} + n_{peasants})|$

$p_{none} \leftarrow n_{none}/sum(n_{none} + n_{landlord} + n_{peasants})$

$\Delta p_{landlord} \leftarrow |p_{landlord} - n_{landlord}/sum(n_{none} + n_{landlord} + n_{peasants})|$

$p_{landlord} \leftarrow n_{landlord}/sum(n_{none} + n_{landlord} + n_{peasants})$

$\Delta p_{peasants} \leftarrow |p_{peasants} - n_{peasants}/sum(n_{none} + n_{landlord} + n_{peasants})|$

$p_{peasants} \leftarrow n_{peasants}/sum(n_{none} + n_{landlord} + n_{peasants})$

until $max(\Delta p_{none}, \Delta p_{landlord}, \Delta p_{peasants}) < \delta$

return p_{none}，$p_{landlord}$，$p_{peasants}$

4.2.3　实验设计、结果与分析

实验基于本文第三章介绍的 PyBotzone 环境进行。前文提出的评估方法需要从 Botzone 平台斗地主游戏天梯排行榜选取两组排名差距较大的 Bot 进行对局。实验随机选取了天梯排行榜排名前 10 名中的 5 个 Bot 作为高排名 Bot，排名 200 名之后的 3 个 Bot 作为低排名 Bot，根据算法 2 给出的伪代码进行实验。

将实验结果以模拟对局次数作为横轴，初始手牌无倾向性、倾向地主和倾向农民的概率分别作为纵轴绘制曲线得到图 4.1；倾向性概率稳定后（任意一种倾向性相邻两次概率差均小于10^{-4}）具体的数值如表 4.1 所示。

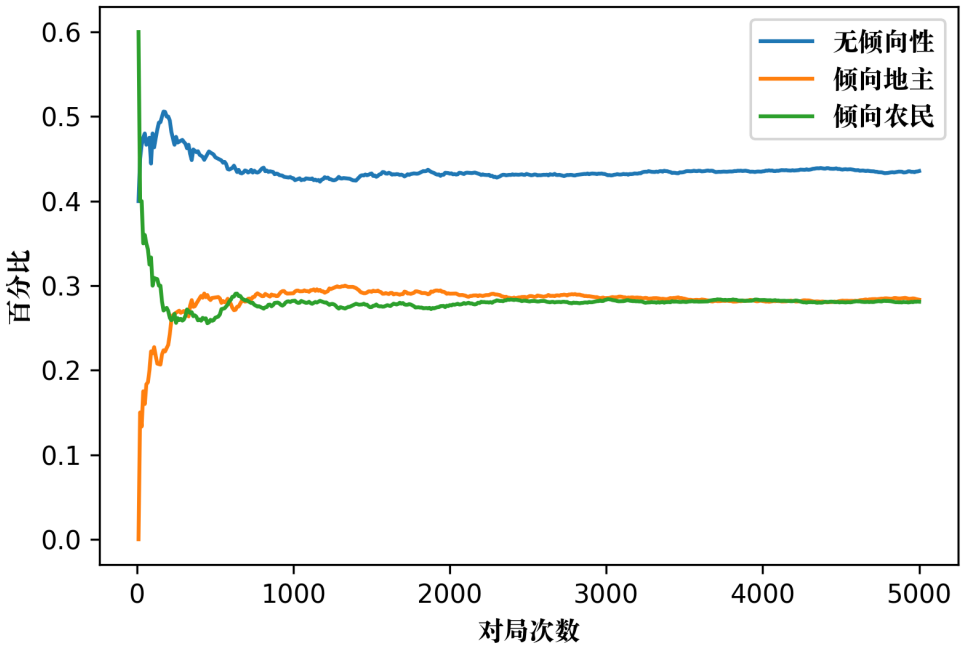

图 4.1 斗地主游戏初始手牌倾向性分布实验结果图

表 4.1 斗地主游戏初始手牌倾向性分布实验结果表

实验结果	无倾向性	倾向地主	倾向农民
出现概率	43.56%	28.34%	28.10%

由图 4.1 可以看出，初始手牌倾向性的分布在模拟对局次数达到 3000 次左右时便基本收敛不再浮动；结合表 4.1 可以得知，斗地主游戏初始手牌无倾向性的比例占到了 43.56%，且倾向地主与农民的比例接近。因此可以认为如果对初始手牌不进行预先筛选，那么通过对局进行斗地主游戏 AI 智能体的评估方法有超过一半的对局会因为存在偏向性而导致评估结果不够客观有效；同理游戏难度的评估也会存在偏差。而由于倾向性在地主和农民的分布是接近的，当评估对局数足够多时，不筛选并不会产生错误的评估结果；但是当评估对局数较小时，评估结果可能出现偏差、需要进行筛选。而对于初始手牌的筛选方法，除了结合人类经验进行精心筛选之外，还可以参考本节提出的$EvaluateTendencyOfInitCards$初始手牌倾向性计算方法进行筛选。

4.3　蒙特卡洛复杂度分析方法

本节介绍游戏问题的蒙特卡洛复杂度的定义以及其计算方法。在给出具体定义前，首先对为什么需要蒙特卡洛复杂度进行解释，然后给出蒙特卡洛复杂度的定义以及数值的含义，最后给出其计算方法。

4.3.1　定义蒙特卡洛复杂度的原因

对于复杂的游戏，在有限的时间内，使用极大极小搜索无法搜索整个搜索空间时，通常会使用启发式估值函数对分支节点进行排序，从而优先搜索可能更有价值的分支，或者直接得到节点的估值。启发式函数一般需要借助问题相关的人类经验，所以并不能够在不同游戏之间通用；同时，有些游戏没有或者尚未发现足够有效的启发式函数。相比之下，基于蒙特卡洛树搜索的方法，通过在不同分支反复模拟采样的方法，来得到对分支节点价值的估计。这种方法不依赖于特定的游戏，是一种可以在不同的游戏之间通用的方法。然而，不同游戏的搜索树的大小与形状是不同的，对应的使用模拟采样进行节点价值估计的复杂度也是不同的。

影响模拟采样估值复杂度的原因有四点。其一是子节点估值收敛的速度，也就是需要进行的模拟采样次数。如果一个游戏只需要很少次数的模拟采样就可以得到准确的价值估计，那么可以很快得到决策结果；反之就需要花费更多的计算来进行价值的估计。其二是节点子树上叶子节点的价值分布情况，一些游戏的节点子树叶子节点的价值分布比较分散，需要很多次迭代才能对其价值进行准确估计；而另一些游戏的节点子树叶子节点的价值分布比较集中，只需要较少的模拟次数即可以得到比较准确的估计。其三是子节点估值的差异情况，由于对子节点估值的目的是选出最好的动作，而有些游戏不同子节点的估值差别不大或者彼此有很大的重叠，导致即使进行再多的模拟采样也不容易决策哪个子节点更优，也就无法确定当前应该选择哪个动作。其四是有些游戏可能存在搜索深度较浅的浅层陷阱（Ramanujan et al., 2010），如果使用极大极小搜索进行完全搜索，是可以很容易发现这样的点的；但是使用随机模拟采样的方法未必能够找到这样的点。这使得需要关注一个节点子树下叶子节点的深度分布情况。

关于上述价值分布、浅层陷阱等因素对于模拟采样估值的影响，前人已有一些研究工作进行了初步的探究（Ramanujan et al., 2010; Ramanujan et al., 2012）。接下来将通过蒙特卡洛复杂度，对这些因素进行更加细致的研究与刻画。

4.3.2　蒙特卡洛复杂度的定义

在给出蒙特卡洛复杂度具体的定义前，还需要对一些设定进行明确。首先，游戏问题的搜索空间应当是内有限时间不可穷尽的。当某个游戏问题的搜索空间是在有限时间内可穷尽时，直接使用传统树搜索算法进行求解即可，无需使用基于模拟采样估值

的算法进行求解。其次，游戏问题的参与者也应当都是基于模拟采样估值算法进行决策的。游戏问题的最优解有时可能存在于统计结果并不占优势的搜索空间中，因此会被算法忽略；而如果游戏问题的参与者存在基于搜索算法的决策者的话，势必会放大这种缺点带来的影响。总的来说，蒙特卡洛复杂度面向的是问题本身足够复杂，且参与者都是基于模拟采样估值的算法决策者的游戏问题。

对于某个符合这一设定的游戏问题，在不借助任何先验知识的情况下，称使用蒙特卡洛方法对其进行动作价值估计的难度为蒙特卡洛复杂度，使用一个 11 维向量进行表示，表 4.2 给出了具体的符号表示以及对应含义。

表 4.2 蒙特卡洛复杂度的定义

序号	符号表示	含义
1	m	游戏问题对应博弈树根节点下子节点个数
2	n	所有子节点收敛时模拟采样迭代次数之和
3	$Iteration_{avg}$	子节点收敛所需迭代次数的平均值
4	$Iteration_{min}$	子节点收敛所需迭代次数的最小值
5	$Iteration_{max}$	子节点收敛所需迭代次数的最大值
6	$Value_{avg}$	子节点估值的平均值
7	$Value_{min}$	子节点估值的最小值
8	$Value_{max}$	子节点估值的最大值
9	$Depth_{avg}$	子节点下叶子节点深度的平均值
10	$Depth_{min}$	子节点下叶子节点深度的最小值
11	$Depth_{max}$	子节点下叶子节点深度的最大值

在符合设定的游戏过程中，每个决策者需要做的就是根据某种策略进行不断采样，以得到对于所有合法动作尽可能准确的价值估计，从而在决策时选择更优的动作。显然，最朴素的采样策略就是随机采样，最简单的选择方法就是选择平均价值更高的。通过随机采样进行价值估计，如果希望价值估计的更加准确，就需要保证一定的样本空间大小，也就是随机采样的次数。除此之外，基于上一小节的介绍，还需要对价值分布和深度进行刻画，由此得到表 4.2 所列出的这些信息，用于表示蒙特卡洛复杂度。

4.3.3　对蒙特卡洛复杂度数值含义的解释

当根节点下子节点的个数m较大时，博弈树越大、蒙特卡洛复杂度也越高，即使用蒙特卡洛树搜索等基于模拟采样估值的方法的计算代价越大，游戏的难度也就越大。当与迭代次数有关的n、$Iteration_{avg}$、$Iteration_{min}$和$Iteration_{max}$越大时，表示估值所需的模拟采样次数越多，蒙特卡洛复杂度越高。

子节点的价值估计平均值$Value_{avg}$反映了该游戏开局对哪一方更有利，正则对己方有利、负则对对方有利。最小值$Value_{min}$与最大值$Value_{max}$描述了估值的值域，它们的差值越小，越不容易进行动作的抉择；反之，差值越大则越容易抉择。

子节点下叶子节点深度的平均值$Depth_{avg}$越大，表示子节点下潜在的搜索空间越大，蒙特卡洛复杂度也越大。而最小值$Depth_{min}$与最大值$Depth_{max}$的差值越大则表示越有可能存在浅层陷阱，对应的估值结果也就越可能不准确、给动作的选择造成更大的困难。

4.3.4　蒙特卡洛复杂度的计算方法

算法 3 给出了计算游戏问题蒙特卡洛复杂度的伪代码。

3 游戏问题的蒙特卡洛复杂度计算

函数：$ComputeGameMonteCarloComplexity$

输入：$game$

输出：m， n，$Iterations$，$Values$，$Depths$

基于游戏规则得到游戏问题初始状态的合法动作个数m

初始化大小为m的集合$Iterations$、$Values$和$Depths$

$n \leftarrow 1$

repeat

for i 从 1 到m **do**

if $Values[i]$ 已经收敛 **and** $Depths[i]$ 已经收敛 **then**

$Iterations[i] = Values[i].length$

continue

end if

执行第i个合法动作a_i $ExecuteAction(game, a_i)$

$depth \leftarrow 1$

repeat

随机选择一个合法动作$a \leftarrow RandomAvailableAction(game)$

执行动作$ExecuteAction(game, a)$

$depth \leftarrow depth + 1$

until $IsTerminated(game) = true$

$reward \leftarrow Reward(game)$

记录采样信息 $Values[i].append(reward), Depths[i].append(depth)$

$n \leftarrow n + 1$

end for

until $Values$ 和 $Depths$ 中所有元素的价值均已收敛

return m， n，$Iterations$，$Values$，$Depths$

在给定某个确定的游戏问题后，便可以根据游戏的规则不断使用蒙特卡洛方法进行模拟。从游戏的初始状态开始不断随机选择合法动作执行、推进游戏发展直到游戏结束。除了最后获得的胜负结果，还需要记录得到该结果的动作序列长度。不断重复模拟过程、直到根节点所有子节点的估值和深度都收敛，得到表示游戏问题的蒙特卡洛复杂度所需的全部信息。

4.4　斗地主游戏的蒙特卡洛复杂度计算实验

本节介绍针对斗地主游戏的蒙特卡洛复杂度计算实验的相关内容。在介绍实验的相关设计后，首先给出总的实验结果，然后逐项进行相关分析。

4.4.1　实验设计

斗地主游戏的合法动作与每个玩家的手牌直接相关，所以如果希望使用算法 3 对其蒙特卡洛复杂度进行估计，就需要固定其初始手牌。通过本文 4.2 节提出的$EvaluateTendencyOfInitCards$初始手牌倾向性计算方法筛选出 3 副无倾向性的初始手牌，每次实验固定使用其中的一副进行模拟。实验按照算法 3 给出的伪代码执行，游戏结束时，地主胜利收益记为+1、失败则记为-1；收敛的判断标准是相邻两次计算的均值结果差值在10^{-4}之内。实验的硬件运行环境是 2.3 GHz 八核 Intel Core i9、内存为 16 GB 2400 MHz DDR4 的 macOS 10.15.4。

4.4.2　实验结果总览

表 4.3 给出了 3 次实验得到的蒙特卡洛复杂度计算结果。可以发现斗地主游戏的初始手牌情况不同会极大的影响蒙特卡洛复杂度的计算结果。接下来将分为五小节对实验结果进行更加详细的分析。

表 4.3 斗地主的蒙特卡洛复杂度计算结果

实验序号	**1**	**2**	**3**
m	450	43	229
n	19,869,066	1,739,158	7,610,373
$Iteration_{avg}$	44,153	40,445	33,233
$Iteration_{min}$	37,171	38,132	28,662
$Iteration_{max}$	49,985	43,334	38,768
$Value_{avg}$	0.0487	-0.0189	0.0015
$Value_{min}$	-0.0498	-0.1713	-0.0999
$Value_{max}$	0.150	0.0987	0.0998
$Depth_{avg}$	71.0	74.5	61.2
$Depth_{min}$	54	63	36
$Depth_{max}$	88	88	87

4.4.3 实验结果分析：子节点个数和总迭代次数

由于初始手牌的不同，可以组成的牌型不同，因此初始状态节点下的合法动作数也会不同。本节的 3 次实验中，每次实验间的初始手牌情况都是不同的，对应的合法动作数也是不同的。表 4.4 给出了 3 次实验中初始手牌的具体情况。

表 4.4 实验初始手牌具体情况

实验序号	地主初始手牌	农民甲初始手牌	农民乙初始手牌
1	红桃 3,方块 3,黑桃 3,草花 3,红桃 4,方块 5,草花 8,红桃 9,方块 9,黑桃 9,草花 9,草花 10,方块 J,黑桃 J,草花 Q,红桃 1,黑桃 2,草花 2,小王,大王	方块 6,草花 6,红桃 7,方块 7,黑桃 7,草花 7,红桃 8,黑桃 8,红桃 10,红桃 J,草花 J,红桃 Q,方块 Q,黑桃 Q,红桃 K,黑桃 1,草花 1	方块 4,黑桃 4,草花 4,红桃 5,黑桃 5,草花 5,红桃 6,黑桃 6,方块 8,方块 10,黑桃 10,方块 K,黑桃 K,草花 K,方块 1,红桃 2,方块 2
2	红桃 3,草花 3,红桃 4,方块 4,红桃 5,方块 5,红桃 6,方块 6,黑桃 6,方块 7,黑桃 7,红桃 8,草花 8,草花 9,方块 10,草花 10,方块 K,草花 1,黑桃 2,小王	黑桃 4,草花 4,草花 5,红桃 7,方块 9,红桃 10,黑桃 10,红桃 J,方块 J,黑桃 J,草花 J,方块 Q,草花 Q,草花 K,方块 1,红桃 2,方块 2	方块 3,黑桃 3,黑桃 5,草花 6,草花 7,方块 8,黑桃 8,红桃 9,黑桃 9,红桃 Q,黑桃 Q,红桃 K,黑桃 K,红桃 1,黑桃 1,草花 2,大王
3	草花 3,方块 4,草花 4,红桃 5,方块 5,黑桃 5,草花 5,红桃 7,黑桃 7,红桃 8,方块 8,草花 8,方块 J,草花 J,黑桃 Q,方块 K,方块 1,方块 2,草花 2,大王	红桃 3,黑桃 3,红桃 6,黑桃 6,草花 7,黑桃 8,红桃 9,方块 9,草花 9,红桃 10,方块 10,红桃 J,红桃 K,黑桃 K,草花 K,黑桃 2,小王	方块 3,红桃 4,黑桃 4,方块 6,草花 6,方块 7,黑桃 9,黑桃 10,草花 10,黑桃 J,红桃 Q,方块 Q,草花 Q,红桃 1,黑桃 1,草花 1,红桃 2

结合表 4.3 中m与n的结果，可以发现总迭代次数与子节点个数有着紧密的关系，子节点个数大时总迭代次数很大、子节点个数小时总迭代次数很小。由于每个子节点都需要一定数量的模拟采样迭代，因此每增加一个子节点总迭代次数就会增加一点，从而导致子节点个数越大总迭代次数越大的现象。

总迭代次数越大，游戏难度也就越大。因此当初始状态节点下子节点个数较多时，对应的游戏难度也会较大。根据这一标准，可以认为斗地主游戏的难度是较大的。

4.4.4　实验结果分析：子节点迭代次数

图 4.2 给出了初始状态节点下各个子节点收敛时所执行蒙特卡洛方法次数的分布情况，每个分图对应一次实验。其中横坐标表示收敛时子节点蒙特卡洛方法的迭代次数，纵坐标表示子节点个数占所有子节点个数的百分比。

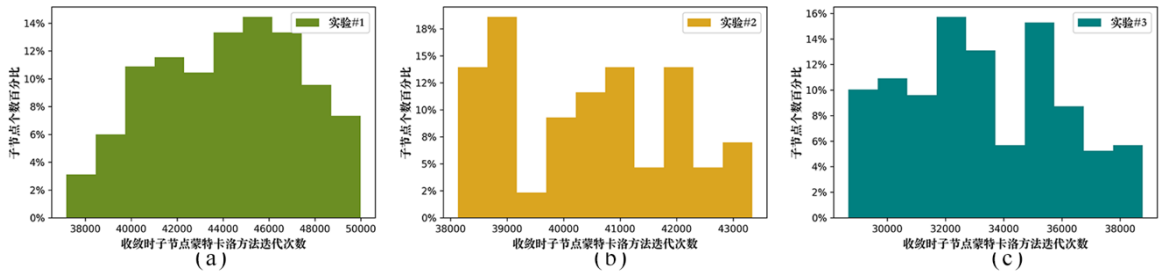

图 4.2 初始状态节点下子节点收敛时蒙特卡洛方法迭代次数分布情况

由图 4.2 可以发现初始状态节点下子节点个数较多的两次实验的迭代次数分布比较接近均匀分布，子节点个数较少的第 2 次实验的迭代次数分布则较为离散。虽然值域范围较大，但整体上各个子节点收敛所需的迭代次数都较大；结合 4.4.3 小节的分析，当合法动作数增加时，总的迭代次数也会相应的增加，对应的蒙特卡洛复杂度也会越高。

4.4.5　实验结果分析：子节点价值估计

图 4.3 给出了初始状态下各个子节点收敛时各节点价值估计的分布情况，每个分图对应一次实验。其中横坐标表示子节点的价值估计，纵坐标表示子节点个数占所有子节点个数的百分比。

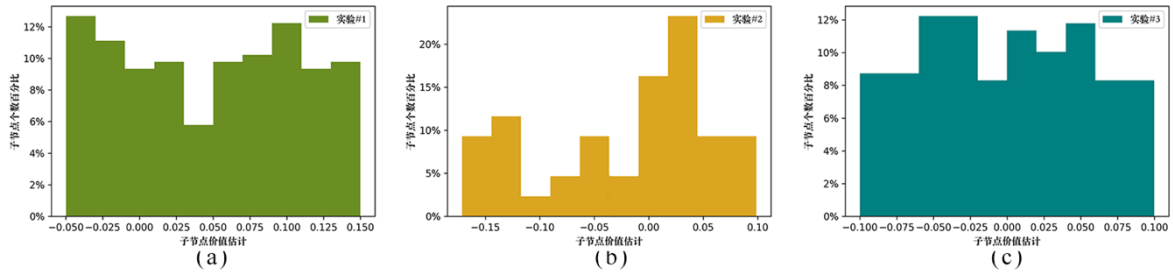

图 4.3 初始状态节点下子节点收敛时各节点价值估计分布情况

由图 4.3 可以发现对于初始状态合法动作较多的第 1 次实验（$m = 450$），大部分动作的估值都是大于 0 的；而合法动作较少的第 2 次实验（$m = 43$），估值小于 0 的动作占比更多一些；第 3 次实验的分布则更加均匀一些。虽然实验使用的初始手牌都是经过筛选、不存在倾向性的，按照随机采样策略进行模拟得到的胜负比例直觉上应该是接近的。不过考虑到当合法动作数较多时，意味着存在牌型更加复杂、出牌数也更多的动作，选择这些动作可以有效的减少手牌数、使自己更加趋向于胜利。因此当合法动

作较多时，获胜的几率会更大一些。

在子节点的价值估计分布上，可以看到都是接近均匀分布的，而且值域上最大能够达到 0.2 左右的差距。这意味着通过一定次数的模拟采样之后，得到的子节点估值存在着一定的差异性，也就说明可以从这些子节点对应的合法动作中选择出相对较优的动作。可以认为斗地主游戏通过蒙特卡洛树搜索算法为代表的模拟采样估值算法是能够在合法动作中进行抉择的。

4.4.6 实验结果分析：子节点深度

图 4.4 给出了 3 次实验蒙特卡洛方法中对局长度、即博弈树深度的分布情况，每个分图对应一次实验。其中横坐标表示博弈树深度，纵坐标表示模拟对局的个数占所有模拟对局个数的百分比。

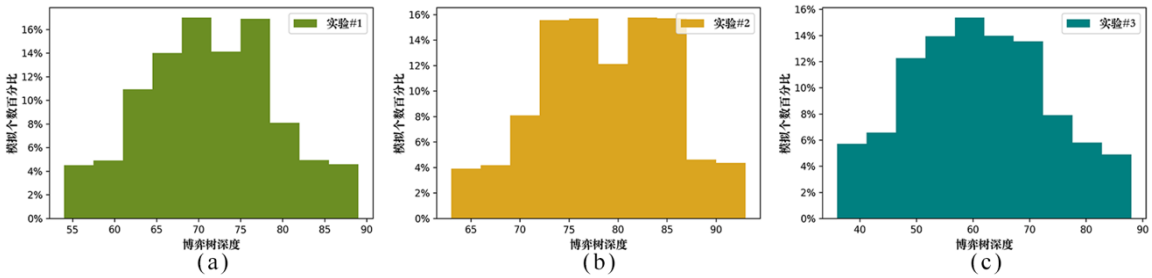

图 4.4 蒙特卡洛方法中所有对局长度的分布情况

图 4.5 给出了初始状态下各个子节点收敛时各节点深度的分布情况，每个分图对应一次实验。其中横坐标表示子节点的深度，纵坐标表示子节点个数占所有子节点个数的百分比。

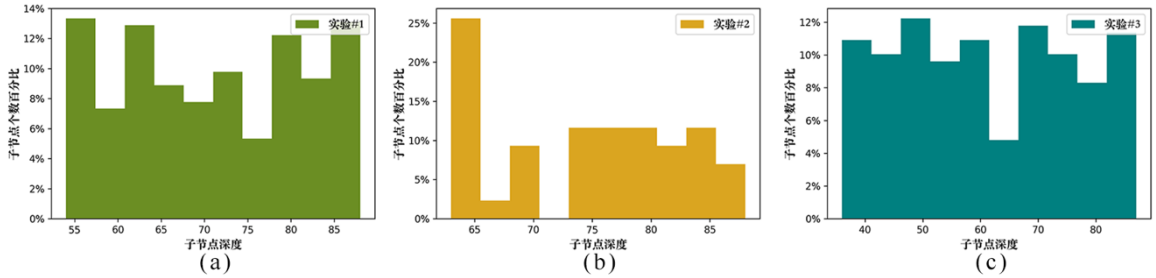

图 4.5 初始状态节点下子节点收敛时各节点深度分布情况

由图 4.4 可以发现在蒙特卡洛方法中，所有模拟对局的对局长度是接近于正态分布的，且不存在明显的异常值。由图 4.5 可以发现每个子节点的平均深度分布是相对均匀的。斗地主游戏中不同动作对应的出牌数有可能不同，而出牌数会直接影响剩余手牌数，进而影响对局长度。因此虽然深度的值域跨度较大，但是考虑到斗地主游戏的规则以及没有明显的异常分布，可以推测浅层陷阱出现的概率是较低的。

除了依靠本节实验得到的结果，Botzone 平台提供的斗地主游戏对局数据也可以用

于统计对局长度、即博弈树深度的统计信息。在剔除掉所有人类玩家参与的比赛以及由于非法操作（程序崩溃、决策超时等）结束的比赛之后，统计得到的平均对局长度为36.8，明显小于实验得到的平均结果。这是由于实验中使用的采样策略是随机采样，合法动作中有许多单张出牌的选择，所以在模拟过程中存在大量的单张出牌情况；选择过牌也会同样导致对局长度的增加。相比之下，Botzone 平台的 Bot 通过各种各样算法的训练，更倾向于打出牌数更多、牌型更复杂的动作，所以对局会更快的结束、对局长度也就更小。

4.4.7　实验结果分析：时间开销

斗地主游戏的一个特点是动作空间非常大，以本节第一次实验为例，初始状态下便存在 450 个合法动作。在计算某个状态下的合法动作时，由于斗地主游戏的合法牌型多种多样，且牌数并不固定，所以只能枚举出所有基于当前手牌的手牌组合，从一张牌的单张到十几张牌的航天飞机，逐一判断该组合是否是合法牌型。假如玩家当前手牌数为n_{hand}，则需要枚举$2^{n_{hand}}$种可能的手牌组合。因此计算合法动作这一步是十分耗时的。而对于蒙特卡洛方法来说，十分重要的一点就是随机模拟采样，即从当前状态下所有的合法动作中随机选取一个进行模拟。因此在实验中，需要花费很多的时间计算每个状态的合法动作，以供模拟时随机选择使用。

基于本节实验的硬件环境，计算一个动作是否是合法动作的平均耗时为 $20\mu s$。斗地主游戏的初始状态是地主出牌，其手牌数为 20。因此需要枚举$2^{20} = 1048576$种手牌组合并计算每个动作是否是合法动作，耗时约为 20s；农民甲、农民乙第一次决策时的手牌数为 17，因此计算合法动作所需耗时约为 3s。因此在使用蒙特卡洛方法进行模拟时，一场对局的耗时可能接近 30s 甚至更多。

4.4.8　实验结果分析总结

综上所述，基于本节实验得到的结果，斗地主游戏初始手牌带来的随机性对于子节点个数、即动作分支数，以及收敛所需的迭代次数有非常大的影响，对于价值分布、搜索深度分布没有非常大的影响。通过一定次数的蒙特卡洛方法模拟，可以从合法动作中筛选出一些平均情况下更理想的选择，因此使用蒙特卡洛方法进行模拟采样估值的方法应该是有一定效果的。

由于随机采样可能会较多的选择过牌或出单张的动作，所以对局长度会更长、对应的博弈树深度就更大，所需要进行搜索的空间也就越多。因此如果能够尽早的通过模拟采样估值筛选掉一些这样的动作，可以有效的减小对局长度并缩减其复杂度，进而更加有效的解决。换言之，斗地主游戏具有较高的蒙特卡洛复杂度，需要进行一定数量的模拟采样来获得较为准确的动作估值；但从估值中也可以筛选出一些更加理想、

预期效果更好的动作作为决策结果。

值得注意的是，本节还给出了基于当前实验条件的一场斗地主游戏对局的蒙特卡洛模拟时间。由于计算合法动作这一过程需要进行枚举，每局游戏的模拟耗时在 30s 左右。这对于离线的问题难度评估、性质分析是可以接受的，但是对于像蒙特卡洛树搜索的在线搜索算法是不可接受的。不过可以发现，当手牌数从 20 减少到 17 时，计算时间便从 20s 缩减到了 3s，由此可见当手牌数较小时，计算合法动作时枚举的情况就会变少，耗时也会相应的大幅降低。因此基于当前的实验条件，如果希望加快实验迭代速度的话，可以考虑对手牌数进行适当的减少。

4.5 与黑白棋和西非播棋的对比实验

本节介绍针对黑白棋和西非播棋的蒙特卡洛复杂度计算实验的相关内容。在介绍实验的相关设计后，给出实验结果并结合 4.4.节的内容进行相关分析。

4.5.1 实验设计

与斗地主游戏不同，黑白棋和西非播棋两个游戏不存在任何随机性，且游戏状态完全可观测，属于确定完全信息博弈游戏。实验按照算法 3 给出的伪代码执行，游戏结束时，第一个进行决策的玩家胜利收益记为+1、失败记为-1，平局记为 0。收敛的判断标准以及实验的硬件运行环境同 4.4 节。

4.5.2 实验结果及分析

表 4.5 给出了黑白棋、西非播棋以及本文 4.4 节中斗地主的蒙特卡洛复杂度计算结果对比。

表 4.5 黑白棋、西非播棋与斗地主的蒙特卡洛复杂度计算结果

游戏	黑白棋	西非播棋	斗地主#1	斗地主#2	斗地主#3
m	4	6	450	43	229
n	225,924	247,926	19,869,066	1,739,158	7,610,373
$Iteration_{avg}$	56,481	41,321	44,153	40,445	33,233
$Iteration_{min}$	56,237	40,935	37,171	38,132	28,662
$Iteration_{max}$	56,711	422,82	49,985	43,334	38,768
$Value_{avg}$	-0.0808	-0.0013	0.0487	-0.0189	0.0015
$Value_{min}$	-0.0827	-0.073	-0.0498	-0.1713	-0.0999
$Value_{max}$	-0.0792	0.006	0.150	0.0987	0.0998
$Depth_{avg}$	60.0	50.0	71.0	74.5	61.2
$Depth_{min}$	59.9	49.8	54.8	63.2	36.7
$Depth_{max}$	60.1	50.1	88.6	88.1	87.4

由表 4.5 可以发现，黑白棋和西非播棋的初始状态合法动作数、即子节点个数m是远小于斗地主的，因此虽然三个游戏每个子节点所需的迭代次数是接近的，但总迭代次数n斗地主是远大于这两个游戏的。这就意味着斗地主游戏蒙特卡洛复杂度是大于黑白棋和西非播棋两个游戏的、其游戏难度也更大。

子节点估值的区分度上，相比于黑白棋的 0.003，西非播棋 0.08 的区分度更高，而斗地主 0.2 左右的区分度更大。这也就表示同样是通过模拟采样估值，黑白棋很难从估值中发现更优的动作，而西非播棋和斗地主则相对更容易发现。

在子节点深度的分布上，黑白棋和西非播棋的分布都比较集中；斗地主的分布则因为游戏规则的原因较为离散，且深度明显大于前两个游戏。这意味着相比之下斗地主存在着更大的搜索空间，对应的游戏难度也更大一些。

基于本节的实验结果，可以发现黑白棋与西非播棋的蒙特卡洛复杂度是比较接近的，但是西非播棋的估值更具有区分性、能够更好的选择动作。根据前人的研究成果，西非播棋能够被蒙特卡洛树搜索很好的解决，而蒙特卡洛树搜索解决黑白棋时效果则不好（Browne et al., 2012）的结论也与先前的分析相吻合。

相比之下，斗地主游戏由于初始状态下子节点个数多导致总迭代次数非常大，且子节点的平均深度也更大，其蒙特卡洛复杂度相较于黑白棋和西非播棋是更大的。对斗地主游戏进行一定数量的模拟采样后，可以对其价值分布得到一定的探知。由于其动作价值的区分度较高，因此可以从中有倾向性的选择出一些更理想的动作进行进一步探索。也就是说，通过一定程度的模拟估值，可以对斗地主游戏的价值分布获得一个大致的判断、进而在动作选择时过滤掉一些预计回报不好的动作，在整体情况更优的动作中做出更好的决策。

4.6 本章小结

本章主要讨论了斗地主游戏的难度评估问题。使用状态复杂度、博弈树复杂度等传统复杂度分析方法，对斗地主游戏问题的搜索空间性质进行了初步探索。然后结合Botzone 平台的斗地主游戏 Bot，提出了一种对斗地主游戏初始手牌倾向性的评估方法。在使用该方法以确保不受到随机性影响的基础上，按照本章提出的蒙特卡洛复杂度计算方法对斗地主、黑白棋以及西非播棋进行了实验，对其问题的博弈树特征、尤其是价值分布上的一些性质进行了探究与分析。基于本章的分析和实验结果，斗地主游戏问题虽然具有较大的搜索空间，但是通过结合类似于蒙特卡洛方法的采样估值方法可以有效地对合法动作进行筛选，从而减小一部分相对价值较低的搜索空间、进一步提升算法的效果；与此同时，通过分析指出适当减少手牌数可以有效地减少计算合法动作的时间开销、加快斗地主游戏的模拟速度。这些都为第五章的相关工作奠定了基础。

第五章　斗地主游戏 AI 智能水平的蒙特卡洛当量评估

本章提出一种基于蒙特卡洛树搜索的斗地主游戏 AI 评估方法。其评估指标被命名为蒙特卡洛当量，用于衡量游戏智能体的水平；该指标还可用于对人类经验价值的评估。通过对 Botzone 平台斗地主游戏天梯排行榜的部分 Bot 进行评估，验证该评估方法和蒙特卡洛当量的效果。本章首先分析蒙特卡洛树搜索的算法原理，给出一种游戏 AI 智能体的评估指标。接着基于该指标，提出基于蒙特卡洛树搜索的游戏 AI 评估方法。然后针对斗地主这一游戏问题，对 Botzone 平台的 Bot 进行评估，结合天梯排行榜的名次对结果进行分析。最后，介绍如何使用蒙特卡洛当量评估人类经验的价值，以及在不同游戏间使用蒙特卡洛当量的可行性。

5.1　AI 智能水平的蒙特卡洛当量评估方法

本节给出蒙特卡洛当量的定义及其计算方法。首先通过对蒙特卡洛树搜索的算法原理进行详细介绍，得到一种评估游戏 AI 智能体的指标，进而提出蒙特卡洛当量这一概念；然后再介绍如何对其进行计算。最后设计验证实验，并结合实验结果说明这是一种有效的评估方法和计算方法。

5.1.1　蒙特卡洛当量的定义

本文在 2.3.1 小节介绍了蒙特卡洛树搜索的基本原理与算法流程。作为一种只需要知道游戏合法动作和终止条件便可以进行游戏 AI 智能体训练的算法，蒙特卡洛树搜索无需提前知道游戏问题的任何先验知识，完全通过不断的模拟尝试来学习更优的策略，该特点使得它在理论上适用于任何游戏的游戏 AI 智能体训练。

蒙特卡洛树搜索的初始状态对应着游戏问题中的某个局面，算法开始时以该状态建立博弈树的根节点，选择一个合法动作执行、并将执行后的状态作为初始状态的子节点添加至博弈树上，然后使用蒙特卡洛方法模拟到游戏结束，获得本次游戏的胜负结果并将结果沿着本次模拟的路径反向逐步更新、直到初始状态。不断迭代这样的“选择→扩展→模拟→更新”流程，蒙特卡洛树搜索便构建出了一棵给定游戏问题的博弈树，进而可以产生较优的动作决策。

蒙特卡洛树搜索每次迭代过程都会通过蒙特卡洛方法产生一组动作序列以及对应的胜负结果，实现一次对游戏问题的探索。不断循环这样的过程，智能体便获得了一定数量的采样数据，从而可以对游戏问题的状态（或动作）价值进行估计。蒙特卡洛树搜索算法默认情况下都使用 UCT 算法，能够保证在迭代的过程中平衡探索与开发的困境，即在保证一定程度探索的基础上，尽可能地在具有更高价值的空间里进行搜索（Kocsis

et al., 2006）；而根据大数定律，该估计值会随着样本数的增加而变得更加准确。因此当迭代次数增加时，算法对于价值的估计也会更加准确，进而在动作决策时产生更加正确的选择。从最开始对游戏规则一无所知，到后来拥有与顶尖选手相媲美的水平，蒙特卡洛树搜索算法训练得到的游戏 AI 智能体的智能完全来自于算法过程中不断的迭代。迭代的过程本质上就是计算机不断进行蒙特卡洛模拟计算更加准确的估值的过程。

许多游戏 AI 智能体的研发过程中，会加入很多基于人类经验的启发式函数。这些人类经验知识是基于人类长期以来对游戏规则的理解、对局的总结与归纳形成的，对一些特定的模式特征直接进行估值、而不是基于搜索的估值；换句话说，就是利用人类长久以来积累的智慧来代替计算机搜索计算得到的估值。由此得到的智能体水平，直接与这些人类智慧是否准确挂钩。相比之下，基于蒙特卡洛树搜索得到的游戏智能体更像是一个纯粹的计算机程序，用计算机最擅长的计算来弥补人类智慧的缺失。

通常情况下，一个基于人类经验的游戏智能体，随着人类经验数量的增加、质量的提高，其水平也会相应的提升；一个基于蒙特卡洛树搜索的游戏智能体，随着迭代次数的增加，其模拟采样的次数也随之增加，对价值的估计也更加准确，其水平往往也更强。

如果某个不是基于蒙特卡洛树搜索的游戏智能体$target$，在水平上与一个基于蒙特卡洛树搜索、算法迭代次数为m的游戏智能体$mcts$表现相当时，定义游戏智能体$target$的蒙特卡洛当量为m。基于本节的分析，当一个游戏智能体的蒙特卡洛当量越大时，其水平越高、能力越强；反之则水平越差。

5.1.2 蒙特卡洛当量的计算方法

蒙特卡洛当量的提出来自于蒙特卡洛树搜索算法，因此其计算方法也与算法密切相关。计算方法如图 5.1 所示，对于待评估的目标智能体$target$，使用蒙特卡洛树搜索训练一个智能体$mcts$，令目标智能体$target$与蒙特卡洛树搜索智能体$mcts$进行n场对局。当智能体双方在对局中呈现出接近的水平时，便可以用此时蒙特卡洛树搜索智能体$mcts$的迭代次数m作为目标智能体$target$的蒙特卡洛当量。使用二分查找对目标智能体$target$的蒙特卡洛当量进行搜索：首先给定一组最大值max和最小值min，设定迭代次数的初始值为二者的平均值并开始迭代搜索。当$target$与$mcts$进行对局时表现相当，则认为当前的迭代次数m即为解；如果n场对局之后$mcts$的表现更好，则认为当前迭代次数m过大，搜索区间变更为$[min, m)$；反之变更为$(m, max]$。此处假设当迭代次数达到max时，$mcts$强于$target$；否则二分查找输出的结果为$m = max$，并不保证二者的水平相当。本节稍后会提出一种方法用于计算合适的搜索区间以保证目标智能体$target$的蒙特卡洛当量存在于该区间中。

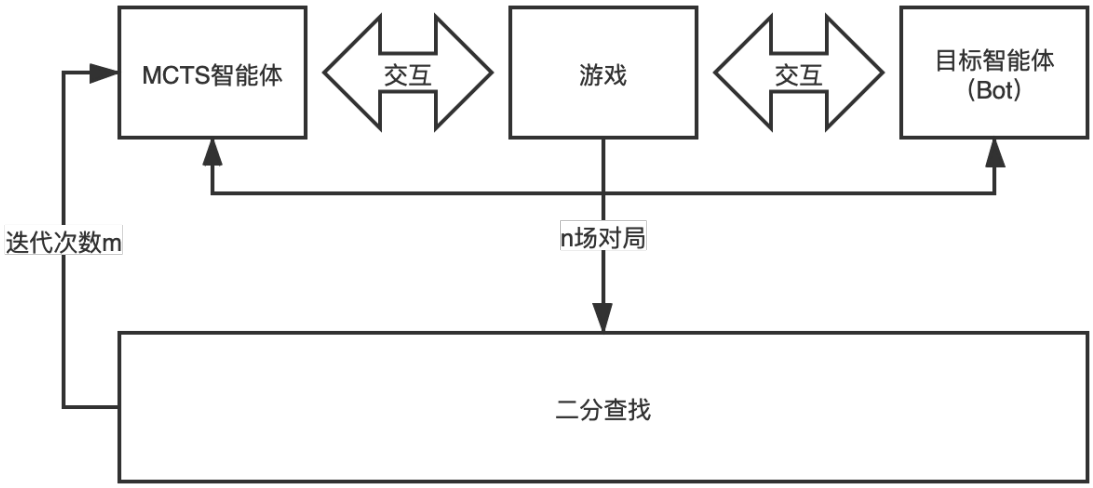

图 5.1 基于蒙特卡洛树搜索的游戏 AI 智能体水平评估方法示意图

通过比较n场对局中$target$的胜场数win_{target}和$mcts$的胜场数win_{mcts}来评估目标智能体$target$和蒙特卡洛树搜索智能体$mcts$的水平。当$win_{target} = win_{mcts}$时认为二者水平相当，$win_{target} > win_{mcts}$时目标智能体$target$更强、需要提升蒙特卡洛树搜索智能体$mcts$的迭代次数；反之蒙特卡洛树搜索智能体$mcts$更强、需要降低它的迭代次数。算法 4 给出了二分查找的伪代码。

4 二分查找搜索合适的迭代次数

函数：$BinarySearchForIterationNumber$

输入：$target$，$mcts$，min，max，n

输出：m

$lower \leftarrow min$，$upper \leftarrow max$

$iteration \leftarrow (lower + upper)/2$

$done \leftarrow false$

repeat

设定$mcts$的迭代次数为$iteration$ $mcts.iteration = iteration$

进行n场对局得到各自胜场数 $win_{target}, win_{mcts} = Matches(target, mcts, n)$

if $win_{target} - win_{mcts} = 0$ **then**

$done \leftarrow true$

else if $win_{target} - win_{mcts} > 0$ **then**

$lower \leftarrow iteration$

else

$upper \leftarrow iteration$

end if

$iteration \leftarrow (lower + upper)/2$

if $lower \geq upper$ **then**

$done \leftarrow true$

end if

until $done$

return $iteration$

二分查找要求待检索元素需要按照关键字有序排列，区间$[min, max]$显然满足这一性质。当目标元素不存在于待检索元素集合中时，二分查找会返回区间的边界值min或max。在算法 4 中，应保证目标迭代次数（蒙特卡洛树搜索复杂度）$m \leq max$，否则输出结果不能代表正确的估计；同时为了减少不必要的计算代价，min的值应在保证$m \geq max$的基础上尽可能大以减小二分查找的迭代次数。参考数组扩容的机制（Cormen et al., 2009），给出计算二分查找初始区间的算法 5。

5 计算二分查找的初始区间

函数：$ComputeInitBound$

输入：$target$，$mcts$，min，max，n

输出：$lower$，$upper$

$lower \leftarrow min$，$upper \leftarrow max$

$done \leftarrow false$

repeat

设定$mcts$的迭代次数为$iteration$ $mcts.iteration = upper$

进行n场对局得到各自胜场数 $win_{target}, win_{mcts} = Matches(target, mcts, n)$

if $win_{target} \leq win_{mcts}$ **then**

$done \leftarrow true$

else

$lower \leftarrow upper - Random(0.1 \times upper)$

$upper \leftarrow 2 \times upper + Random(0.1 \times upper)$

end if

until $done$

return $lower$，$upper$

算法 5 在扩大区间时对区间边界进行了随机调整，这样在实际计算时可以通过多次计算取平均值来获得实际评估时的初始区间。

5.1.3 验证实验的设计、结果与分析

接下来通过实验验证算法 4 和算法 5 是有效的。如本文 4.2 节所描述的，在斗地主游戏中初始手牌存在倾向性、在有限次数的对局评估中会对游戏 AI 智能体的水平评估结果造成影响。因此在目标智能体$target$与蒙特卡洛树搜索智能体$mcts$进行对局时，应对每场对局的初始手牌进行倾向性评估并选择无倾向性的初始手牌进行对局评估、以避免评估结果的误差。本小节验证实验使用的游戏环境是迷你斗地主，将在接下来的 5.2.2 小节做详细的介绍。

算法 4：通过二分查找寻找合适的蒙特卡洛树搜索迭代次数，使得蒙特卡洛树搜索智能体$mcts$能够与目标智能体$target$得到相近的结果，以该迭代次数作为目标智能体$target$的蒙特卡洛当量。验证时选取已知迭代次数m_{eval}的蒙特卡洛树搜索智能体作为目标智能体$target_{eval}$、初始搜索区间$lower = 1$、$upper = 2 \times m_{eval}$、对局数$n = 1000$，$target_{eval}$和$mcts$的蒙特卡洛树搜索算法参数$UCB = 1.0$；同时在每次对局开始前调用

$EvaluateTendencyOfInitCards$方法对初始手牌进行倾向性评估，当当前初始手牌存在倾向性时，重新发牌直到无倾向性为止。之后根据算法 4 计算其蒙特卡洛当量。

表 5.1 算法 4 的验证实验结果，对局数$n = 1000$

m_{eval}	$lower$	$upper$	$iteration$	$ratio$
500	1	1000	485±40	0.97±0.08
1000	1	2000	1020±100	1.02±0.10
2000	1	4000	1980±120	0.99±0.06

表 5.1 给出了算法 4 验证实验的实验结果，每组不同的m_{eval}实验进行了 10 次，$iteration$列的结果是 10 次蒙特卡洛当量的平均值和标准差、$ratio$列的数值是将同行的$iteration$与m_{eval}做比得到的。由此可以看出通过蒙特卡洛当量对游戏 AI 智能体进行评估的结果是较为准确的、可以应用到基于其他算法设计的智能体评估。

算法 5：通过模拟对局计算二分查找的初始区间。验证时选取已知迭代次数m_{eval}的蒙特卡洛树搜索智能体作为目标智能体$target_{eval}$、对局数$n = 1000$，$target_{eval}$和$mcts$的蒙特卡洛树搜索算法参数$UCB = 1.0$，输入参数$min = 1$、$max = 100$；对局开始前调用$EvaluateTendencyOfInitCards$方法进行倾向性校验。根据算法 5 计算得到的初始区间均可以覆盖m_{eval}，因此可以认为通过算法 5 计算二分查找的搜索区间是可以保证蒙特卡洛树当量落在搜索区间内部的。不过在使用算法 5 时应注意，输入的参数min和max差距不宜过小，以避免蒙特卡洛当量接近边界值导致评估不准确。

算法 4+算法 5：最后结合算法 4 与算法 5 进行验证，同样选取已知迭代次数m_{eval}的蒙特卡洛树搜索智能体作为目标智能体$target_{eval}$、对局数$n = 1000$，$target_{eval}$和$mcts$的蒙特卡洛树搜索算法参数$UCB = 1.0$，使用输入参数$min = 1$、$max = 100$计算二分查找的初始区间；对局开始前调用$EvaluateTendencyOfInitCards$方法进行倾向性校验。

表 5.2 算法 4+算法 5 的联合验证实验结果，对局数$n = 1000$

m_{eval}	$lower$	$upper$	$iteration$	$ratio$
500	398±11	861±19	505±30	1.01±0.06
1000	824±29	1743±41	1020±80	1.02±0.08
2000	1667±64	3642±63	1960±100	0.98±0.05

表 5.2 给出了算法 4+算法 5 验证实验的实验结果，$lower$列与$upper$列的值通过算法 5 计算得到，表中其余结果计算方式与表 5.1 完全相同。综上所述，本节提出的蒙特卡洛当量以及其计算方法在斗地主游戏 AI 智能体的水平评估上是有效的。

5.2　斗地主游戏 AI 评估前的准备工作

在进行评估前，需要对待评估对象、即实验选取的 Botzone 平台斗地主游戏的 Bot 情况以及基本算法进行介绍。同时，对实验中使用的游戏环境迷你斗地主进行介绍。

5.2.1　评估对象

本小节对即将进行评估的对象进行介绍。对于游戏 AI 智能体，除了根据游戏规则执行随机动作的智能体之外，最常见的便是基于一些基本算法的样例程序，以及深度学习、强化学习的预训练模型（Lanctot　et al., 2019; Zha et al., 2019）。随机动作智能体的评估价值普遍较低，而后两种智能体形式多样、迁移成本较大。Botzone 平台在提供了 20+款游戏的同时，在用户同意的情况下，游戏 Bot 也是开源给所有使用者的。Botzone 平台的斗地主游戏上线于 2014 年 11 月，截至本文撰写时，天梯排行榜上共有 341 个 Bot，涉及算法主要包括专家系统、Alpha-Beta 剪枝、纳什均衡、强化学习等。结合本文第三章设计并实现的本地辅助工具 PyBotzone，可完成对大部分天梯排行榜 Bot 的评估。

本章实验共选取 6 个评估 Bot，其中 1 个是随机动作智能体，其余 5 个均来自 Botzone 平台。这 5 个 Bot 包含 1 个样例程序、2 个排名靠前、1 个排名较为靠前以及 1 个排名中等偏上的。在后两个 Bot 的选择上基于代码可读性强、逻辑比较清晰易懂的原则进行了选择。

Botzone 平台斗地主游戏提供了两个样例程序，主要目的是熟悉游戏规则、了解平台使用方式，分别用 C++和 Python 实现。其中 Python 版本的样例程序在 C++版本的基础上考虑了农民之间不相互炸的情况，在过往课程同学们的测试中水平更强一些，天梯排行榜上也有不少基于其做小幅度修改的 Bot，颇有 Botzone 平台斗地主游戏基准线（baseline）Bot 的感觉。Python 版本样例程序的基本思路是在跟牌阶段，根据上家牌型从自身手牌中遍历找出所有相同牌型的可行手牌，并加上炸弹和火箭作为可选择动作，在避免农民乙炸农民甲的前提下从可选择动作中随机选取一个动作进行决策；如果当前没有可选择动作则跳过。在出牌阶段，从手牌所有可选择动作中随机选择一个执行。根据对其代码逻辑的梳理可以看出 Python 样例程序是一个不会农民内讧的随机动作智能体，本文的实验选取它作为第二个评估对象。

天梯排行榜目前排名前三名的 Bot 分别是“知世就是力量”（bot_id: 5b13f4e78472ec612b15452d，作者：知识就是力量）、“这是真的 sample”（bot_id: 5b23b7720edf94798cdc74f2，作者：__Y__）和“下个 Bot 见”（bot_id: 5b10c9cc7307073b572b9c51，作者：AreWeCoolYet）。其中排名第二的“这是真的 sample”无法在本地完成编译运行、因此不能使用该方法进行评估。“知世就是力量”使用的是 Alpha-Beta 剪枝结合估值的方法，特别是在选择从牌和决胜阶段的牌型价值上有着十

分细致的划分。“下个 Bot 见”使用的方法是专家系统。主要思路是在能够找出相同牌型可行手牌的基础上，进行更合理的划分使得剩余手牌能够具有更高的牌型大小。该 Bot 会通过记录所有已经打出的牌来计算当前三人手牌中最大的牌和其余两人剩余手牌数量，在关键时刻进行控场（例如对手剩一张牌不出单、剩两张牌不出对）。除此之外，通过代码的注释可以看出设计该 Bot 的小组观看了大量 Botzone 平台斗地主游戏的对局，总结许多特别的出牌技巧，比如尽量不要过牌、尽量留炸弹等等；他们也对牌型进行了重新估值。从 Botzone 平台的排名分趋势图来看，这两个 Bot 的成绩一直稳定在 1500 分以上、排名前 10 之内。

排名较为靠前的 Bot 选择的是排名 22 的“FTLbot3”（bot_id: 5b1cd1310edf94798cd99cb2，作者：cy1700012807），它明确区分了地主和农民两种角色以及主动出牌和被动跟牌的策略，对于每种牌型都增加了一定的从牌选择策略。

最后一个评估对象是天梯排行榜排名 111 的“FTL_group3”（bot_id: 5b2292d30edf94798cdc0662，作者：T_T），使用的方法是贪心。主要思路就是打出估价尽可能高的牌，同时优先出“三带”、“四带”等牌型。具体实现上采用了 01 编码的状态压缩动态规划，在 Botzone 平台指定的计算时间内进行了更多的计算。作为农民处理另一个农民的出牌时，也会采取尽量过的方式让同伴尽可能的出牌。

根据本文作者的观察，Botzone 平台斗地主游戏的天梯排行榜还没有收敛至稳定状态，大部分 Bot 的排名会在某个区间内波动。因此对于本小节介绍的 Bot 排名可能存在一定出入，属正常现象。

5.2.2 迷你斗地主

上一小节中介绍的评估对象具有一个共同点：都会基于当前手牌枚举所有可能的出牌方式；而蒙特卡洛树搜索算法执行过程中，默认策略在执行时也会计算合法动作用于模拟。如本文 4.4 节所描述的，由于斗地主游戏的牌型组合方式多、牌数灵活，每次搜索都需要进行大量的计算。以地主第一次出牌为例，其手牌数为 20、需要判断是否合法的动作就有2^{20}种。而适当减小手牌数可以有效地减小这一数字，从而加快斗地主游戏的模拟速度，因此提出一种对斗地主游戏规则进行简单修改的迷你斗地主游戏。

迷你斗地主游戏与斗地主游戏一样使用一副 54 张的扑克牌，但是在发牌时每张牌都有 0.5 的几率不落入手牌而是被直接发到牌桌，发牌结束时若地主的手牌数小于等于 3 张则重新发牌。即理论上每个玩家的手牌数会是原来情况下的一半，同样对于地主的第一次出牌，手牌数期望为 10，则理论上最多的出牌方式就只有2^{10}种、判断合法动作所需的计算量也随之大幅减少；同时由于手牌数减少，一些比较复杂的牌型也无法组合，出牌方式数会进一步减小。其他规则与斗地主完全保持一致。

本节的所有实验结果都在迷你斗地主游戏下得到的。

5.3　斗地主游戏 AI 的蒙特卡洛当量评估

本节介绍针对斗地主游戏 AI 的蒙特卡洛当量评估实验的相关内容，简述实验的设计，然后结合实验结果进行适当的分析。

5.3.1　实验设计

以 5.2.1 小节中介绍的 6 个 Bot 作为评估对象，根据 5.1.2 小节中提出的算法 4 对它们进行实验；每个 Bot 重复实验 5 次。在每场对局进行前，调用针对迷你斗地主的$EvaluateTendencyOfInitCards$方法对初始手牌的倾向性进行评估。

5.3.2　实验结果及分析

表 5.3 使用蒙特卡洛树复杂度对 Botzone 平台斗地主游戏 Bot 评估的结果，对局数$n = 1000$

评估对象	天梯排名	$lower$	$upper$	$iteration$
知世就是力量	1	818±25	1742±53	1425±107
下个 Bot 见	3	811±23	1755±51	1217±81
FTLbot3	22	802±27	1770±47	1047±76
FTL_group3	111	196±5	419±6	245±12
样例程序	303	1	100	57±5
随机动作	-	1	100	1.4±0.3

表 5.3 给出了使用蒙特卡洛当量对 Botzone 平台斗地主游戏 Bot 评估的结果。实验过程中评估对象作为目标智能体$target$、对局数$n = 1000$，二分查找初始区间通过算法 5 计算得到、输入参数$min = 1$、$max = 100$。

表中$iteration$列即为评估对象的蒙特卡洛当量，根据此结果可以看到天梯排行榜排名靠前的 Bot 的蒙特卡洛当量也是更大的；同时样例程序作为农民不内讧版本的随机动作智能体，蒙特卡洛当量也与随机动作智能体的接近。

Botzone 平台天梯排行榜上大多数 Bot 都采取了枚举搜索合法牌型结合人类经验的方式。从实际效果来看，这种方式是一种短期内提升 Bot 水平与排名最快捷的方法。人类经验可能体现在基于专家系统的模式判断和特殊操作，也可能是对于状态和动作的价值估计：本次评估的“下个 Bot 见”程序中包含了大量结合人类经验的专家系统特殊判断，在程序的注释中也可以看到小组成员通过观看 Botzone 平台线上对局总结归纳的启发式方法；“知世就是力量”则是对许多牌型的价值进行了估计，使得在决策时产生了更好的动作。结合他们在天梯上的优异成绩，可以说这些人类经验确实极大的增强了 Bot 的水平。

5.4 斗地主游戏 AI 中人类经验价值的蒙特卡洛当量评估

人类经验是人类对自然现象、事物运行规律等进行充分观察后，经过总结和高度凝练后得到的知识。在游戏 AI 的研究中，将人类经验编码到程序中经常可以快速提升智能体的水平，这一点通过上一节的实验结果以及 Botzone 平台斗地主游戏天梯排行榜的结果也可以看到。蒙特卡洛当量是本章提出的一种对于游戏智能体的评估指标，本质上是通过蒙特卡洛树搜索的算法迭代次数来衡量游戏 AI 的水平。对于某个基于人类经验的游戏 AI 程序，删除某条其已有的人类经验应该会对其水平造成一定的影响，对应的其蒙特卡洛当量也会产生变化。直观上，可以通过蒙特卡洛当量数值上的变化来衡量删除的这条人类经验对于智能体水平的影响。

实验评估的 Bot 中，“知世就是力量”程序中人类经验的体现是对价值的估计，不容易实施上述的删除人类经验方法；“下个 Bot 见”程序中则有许多条件判断式的模式判断，比较容易删除。因此选择“下个 Bot 见”作为本节实验的对象。

5.4.1 实验设计

通过对“下个 Bot 见”程序的分析，其主要包含的人类经验有：不要轻易过牌、尽量留炸弹、关键时刻控场，其中关键时刻控场指的是对手剩一张牌时尽量不出单和剩两张牌时尽量不出对。

根据 5.3.2 小节的实验结果，“下个 Bot 见”的蒙特卡洛当量为 1217，此处记做$iteration_{full}$；上述三个人类经验分别记做$knowledge_1$、$knowledge_2$和$knowledge_3$。采取控制变量法，每次评估时删除某一条人类经验$knowledge_i$得到待评估智能体$evaluate_i$，使用算法 4 计算其蒙特卡洛当量$iteration_i$。则这条人类经验$knowledge_i$的价值$value_i$可通过$iteration_{full} - iteration_i$计算得到。

5.4.2 实验结果及分析

表 5.4 “下个 Bot 见”程序中人类经验价值的蒙特卡洛当量评估

人类经验	$iteration_{full}$	$iteration_i$	$value_i$
不要轻易过牌	1217	1192	+25
尽量留炸弹	1217	1221	-4
关键时刻控场	1217	1066	+151

由表 5.4 可以看到，随着删除“不要轻易过牌”和“关键时刻控场”的人类经验，智能体的蒙特卡洛当量呈现了下降的趋势。可以认为这两条经验对于斗地主游戏 AI 是有益的。而删除“尽量留炸弹”后智能体的蒙特卡洛当量不仅没有下降、反而有所增加，经验价值为负数。

5.5　蒙特卡洛当量评估在不同游戏间横向比较的可行性分析

本章 5.3、5.4 节的实验结果表明，蒙特卡洛当量在评估斗地主游戏 AI 智能体的水平，以及人类经验的价值上都具有一定的效果。其基本原理与计算方式均来自于蒙特卡洛树搜索，而算法本身并不需要任何与游戏问题相关的领域知识作为先验条件提前输入。以本章的实验为例，基于蒙特卡洛树搜索的智能体都是通过不断的蒙特卡洛方法模拟采样来学习如何决策的，除了环境输出的可选择的合法动作，它完全不依赖斗地主游戏的任何相关知识。

同时，蒙特卡洛当量的本质是蒙特卡洛树搜索的算法迭代次数，一次算法的迭代表示计算机通过模拟的方式对游戏进行了一次（或多次）计算。也就是说，蒙特卡洛当量是简洁的通过计算机在解决问题时的计算次数作为评估指标对游戏 AI 智能体的水平、人类经验的价值进行评估的。

综合以上两点，一方面蒙特卡洛树搜索理论上可以应用于任何游戏的 AI 智能体训练，因此也可以得到不同游戏上不同 AI 智能体以及人类经验的蒙特卡洛当量；另一方面，由于蒙特卡洛当量是对计算机计算次数的一种衡量方法，因此不仅相同游戏的蒙特卡洛当量可以进行比较，不同游戏间得到的蒙特卡洛当量间的比较也是有意义的、本质上都是计算机程序在求解游戏问题时所耗费的资源。

5.6　本章小结

本章主要介绍了蒙特卡洛当量，一种基于蒙特卡洛树搜索的游戏 AI 评估指标，可用于衡量智能体的游戏水平。通过对蒙特卡洛树搜索的算法原理介绍，引出蒙特卡洛当量以及其跟智能体水平的关系。然后阐述了其评估方法，并给出了提升计算效率的方法以及验证方案。实验结合本文第三章提出的 Botzone 平台本地辅助工具 PyBotzone、以简化的斗地主游戏为问题，通过第四章提出的方法消除了随机性对评估的影响，对 Botzone 平台斗地主游戏天梯排行榜上的若干 Bot 水平进行了评估。除此之外，还针对其中基于人类经验实现的 Bot 进行了分析，并使用蒙特卡洛当量对其主要的人类经验价值进行了评估。最后，对蒙特卡洛当量在不同游戏间横向比较的可行性及意义进行了分析。

第六章　总结与展望

6.1　本文工作总结

本文主要介绍了蒙特卡洛复杂度，一种基于蒙特卡洛方法的斗地主游戏难度评估方法，以及蒙特卡洛当量，一种基于蒙特卡洛树搜索的斗地主游戏 AI 智能体水平评估方法。

蒙特卡洛复杂度通过不断的模拟采样来探索游戏问题的搜索树，在现有状态复杂度、博弈树复杂度仅关注搜索空间大小的基础上，为游戏难度的评估提供了新的视角。通过分支数、不同分支的收敛速度、价值分布、子树深度等属性较为直观地绘出了斗地主游戏的搜索树画像。

蒙特卡洛当量使用蒙特卡洛树搜索训练一个与目标游戏 AI 水平接近的游戏 AI，然后以蒙特卡洛树搜索的算法迭代次数作为评估指标对目标游戏 AI 的水平进行定量评估。该方法试图解决游戏 AI 评估中现有的若干问题，只需要待评估的游戏 AI 即可进行，不再依赖于其他游戏 AI 程序，使得游戏 AI 的评估可以独立完成。在此基础上，还提出了对于人类经验价值的蒙特卡洛当量评估方法。考虑到蒙特卡洛树搜索无需任何领域知识的特点，该方法理论上可以应用到任何游戏 AI 的评估上，具有极好的普适性。

本文还对于 Botzone 平台提供的各类资源进行了整合，并在许多地方进行了优化提升，形成了一套本地辅助工具。该工具可以有效提升研究者使用 Botzone 平台各类资源的效率，加快相关研究工作的进行。

6.2　研究工作展望

本文提出的蒙特卡洛当量可用于定量评估斗地主游戏 AI 智能体的水平，其目前还有一些可以优化和改进的方面，未来的研究工作可以从以下几个方面对本文的工作进行改进和扩展：

- 提升蒙特卡洛当量的计算效率，本文计算蒙特卡洛当量的方式是通过二分查找在某个区间进行搜索得到的。尽管二分查找的复杂度已经是对数级别的，但是每次缩小区间都需要进行一次蒙特卡洛树搜索智能体的训练，这个过程是极其耗费时间的。结合蒙特卡洛树搜索已有的并行方式对该过程进行加速，是对该方法性能上的优化点之一。
- 将本文提出的方法在传统规则的斗地主游戏上延伸，出于对问题简化的目的，本文第五章的实验是基于迷你斗地主游戏环境进行的。传统规则下的斗地主游

戏具有更大的动作空间，对于蒙特卡洛树搜索智能体的训练是一个不小的挑战；同时更大的动作空间通常意味着更大的搜索空间和迭代次数，这也意味着可能需要更长的时间来获得评估结果。

- 蒙特卡洛当量的评估结果和 Botzone 平台天梯排行榜的进一步联合分析，本文抽取了天梯排行榜上的几个 Bot 进行了实验，并做了简单的分析。接下来可以进一步扩大实验 Bot 规模并尝试通过蒙特卡洛当量给出 Bot 的排名、着重对比两种评估方式下 Bot 的排名是否有差异，或许可以从蒙特卡洛当量的角度解释 Botzone 平台斗地主游戏天梯排行榜长期不收敛的问题。

参考文献

Allis L V. Searching for solutions in games and artificial intelligence[M]. Wageningen: Ponsen & Looijen, 1994.

Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem[J]. Machine learning, 2002, 47(2-3): 235-256.

Baier H, Winands M H M. MCTS-minimax hybrids[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2014, 7(2): 167-179.

Bellemare M G, Naddaf Y, Veness J, et al. The arcade learning environment: An evaluation platform for general agents[J]. Journal of Artificial Intelligence Research, 2013, 47: 253-279.

Bojarski M, Del Testa D, Dworakowski D, et al. End to end learning for self-driving cars[J]. arXiv preprint arXiv:1604.07316, 2016.

Brockman G, Cheung V, Pettersson L, et al. Openai gym[J]. arXiv preprint arXiv:1606.01540, 2016.

Brown N, Sandholm T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals[J]. Science, 2018, 359(6374): 418-424.

Browne C B, Powley E, Whitehouse D, et al. A survey of monte carlo tree search methods[J]. IEEE Transactions on Computational Intelligence and AI in games, 2012, 4(1): 1-43.

Campbell M, Hoane Jr A J, Hsu F. Deep blue[J]. Artificial intelligence, 2002, 134(1-2): 57-83.

Chen T , Murali A , Gupta A . Hardware Conditioned Policies for Multi-Robot Transfer Learning[J]. 2018.

Ciancarini P, Favini G P. Monte Carlo tree search in Kriegspiel[J]. Artificial Intelligence, 2010, 174(11): 670-684.

Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to algorithms[M]. MIT press, 2009.

Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]//International conference on computers and games. Springer, Berlin, Heidelberg, 2006: 72-83.

Enzenberger M, Muller M, Arneson B, et al. Fuego—an open-source framework for board games and Go engine based on Monte Carlo tree search[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2010, 2(4): 259-270.

Gelly S, Silver D. Achieving master level play in 9 x 9 computer go[C]//AAAI. 2008, 8: 1537-1540.

Herbrich R, Minka T, Graepel T. TrueSkill™: a Bayesian skill rating system[C]//Advances in neural information processing systems. 2007: 569-576.

Hingston P, Masek M. Experiments with Monte Carlo Othello[C]//2007 IEEE Congress on Evolutionary Computation. IEEE, 2007: 4059-4064.

Ho J, Ermon S. Generative adversarial imitation learning[C]//Advances in neural information processing systems. 2016: 4565-4573.

Jiang Q, Li K, Du B, et al. DeltaDou: expert-level doudizhu AI through self-play[C]//Proceedings of the 28th International Joint Conference on Artificial Intelligence. AAAI Press, 2019: 1265-1271.

Kocsis L, Szepesvári C, Willemson J. Improved monte-carlo search[J]. Univ. Tartu, Estonia, Tech. Rep, 2006, 1.

Kurach K, Raichuk A, Stańczyk P, et al. Google research football: A novel reinforcement learning

environment[J]. arXiv preprint arXiv:1907.11180, 2019.

Lanctot M, Lockhart E, Lespiau J B, et al. Openspiel: A framework for reinforcement learning in games[J]. arXiv preprint arXiv:1908.09453, 2019.

Lanzi P L. Evaluating the Complexity of Players' Strategies using MCTS Iterations[C]//2019 IEEE Conference on Games (CoG). IEEE, 2019: 1-8.

Laviers K, Sukthankar G, Aha D W, et al. Improving Offensive Performance Through Opponent Modeling[C]//AIIDE. 2009.

LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553): 436-444.

Li J, Koyamada S, Ye Q, et al. Suphx: Mastering Mahjong with Deep Reinforcement Learning[J]. arXiv preprint arXiv:2003.13590, 2020.

Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.

Liu Z, Hu M, Zhang Z. A Solution to China Competitive Poker Using Deep Learning[J]. 2018.

Lorentz R J. Amazons discover monte-carlo[C]//International Conference on Computers and Games. Springer, Berlin, Heidelberg, 2008: 13-24.

Moritz P, Nishihara R, Wang S, et al. Ray: A Distributed Framework for Emerging AI Applications[J]. arXiv preprint arXiv:1712.05889, 2017.

Nichol A , Pfau V , Hesse C , et al. Gotta Learn Fast: A New Benchmark for Generalization in RL[J]. 2018.

Ramanujan R, Sabharwal A, Selman B. On adversarial search spaces and sampling-based planning[C]//Twentieth International Conference on Automated Planning and Scheduling. 2010.

Ramanujan R, Sabharwal A, Selman B. Understanding sampling style adversarial search methods[J]. arXiv preprint arXiv:1203.4011, 2012.

Russell S J , Norvig P . Artificial Intelligence: A Modern Approach[M]. 2003.

Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. nature, 2016, 529(7587): 484.

Silver D, Hubert T, Schrittwieser J, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play[J]. Science, 2018, 362(6419): 1140-1144.

Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.

Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. MIT press, 2018.

Tang X , Qin Z T , Zhang F , et al. A Deep Value-network Based Approach for Multi-Driver Order Dispatching[C]// the 25th ACM SIGKDD International Conference. ACM, 2019.

Todorov E, Erez T, Tassa Y. Mujoco: A physics engine for model-based control[C]//2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012: 5026-5033.

Unal, Ceyhan. The AI Games. [OL] 2012. [2020-05-14] http://theaigames.com.

Vinyals O, Babuschkin I, Czarnecki W M, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning[J]. Nature, 2019, 575(7782): 350-354.

Whitehouse D, Powley E J, Cowling P I. Determinization and information set Monte Carlo tree search for the card game Dou Di Zhu[C]//2011 IEEE Conference on Computational Intelligence and Games

(CIG'11). IEEE, 2011: 87-94.

Xian R, Ni L, Li W. The icb-2015 competition on finger vein recognition[C]//2015 International Conference on Biometrics (ICB). IEEE, 2015: 85-89.

Yannakakis G N, Togelius J. Artificial intelligence and games[M]. New York: Springer, 2018.

Ye D, Liu Z, Sun M, et al. Mastering Complex Control in MOBA Games with Deep Reinforcement Learning[J]. arXiv preprint arXiv:1912.09729, 2019.

Zha D, Lai K H, Cao Y, et al. RLCard: A Toolkit for Reinforcement Learning in Card Games[J]. arXiv preprint arXiv:1910.04376, 2019.

Zhang H, Gao G, Li W, et al. Botzone: A game playing system for artificial intelligence education[C]//Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012: 1.

Zhao X, Xia L, Tang J, et al. Reinforcement learning for online information seeking[J]. arXiv preprint arXiv:1812.07127, 2018.

Zhou H, Zhang H, Zhou Y, et al. Botzone: an online multi-agent competitive platform for AI education[C]//Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education. 2018: 33-38.

Zhou H, Zhou Y, Zhang H, et al. Botzone: a competitive and interactive platform for game ai education[C]//Proceedings of the ACM Turing 50th Celebration Conference-China. 2017: 1-5.

个人简历、在学期间的研究成果

个人简历

1995 年 7 月 30 日出生于北京市海淀区；2013 年 9 月考入北京工业大学信息学部软件学院嵌入式系统方向实验班；2017 年 9 月保送进入北京大学信息科学技术学院计算机科学技术系网络与信息系统研究所计算机软件与理论专业攻读硕士学位。

科研论文

黄红拾, **王政飞**, 许国雄, 李文新, 张思, 张东霞, 敖英芳. 基于步行时足底压力信息的前交叉韧带断裂辅助诊断方法. 北京大学学报:自然科学版, Vol.55, No. 5, pp. 859-864, Sept. 2019.

会议论文

Guoxiong Xu, **Zhengfei Wang**, Hongshi Huang, Wenxin Li, Can Liu, Shilei Liu. A Model for Medical Diagnosis Based on Plantar Pressure. in 2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR), Bangalore, India, pp. 1-6, Dec. 2017.

参与课题

- 国家自然科学基金重点项目，91646202，面向全流程智慧健康管理决策的多源异构大数据融合方法研究
- 在线教育研究基金（全通教育）重点项目，2017ZD102，基于大量历史数据的具有智能学习指引功能的人工智能算法在线学习平台研究
- 北京大学医学-信息科学联合研究种子基金，基于步态触地信息的前交叉韧带断裂疾病的计算机辅助诊断和康复评定

奖励荣誉

- 2017 年“北大天网-秒针创新奖学金”
- 2018 年 NeurIPS AI for Prosthetics Challenge Top 10 Award
- 2018 年“北大天网-唱吧奖学金”
- 2018 年北京大学硕士生专项学业奖学金

致谢

三年研究生生涯，在我写下这一页时，已然奏响了终章的序曲。这一路走来，需要感谢的人实在是太多太多。

首先，我要衷心感谢我的导师李文新教授。感谢李老师在四年前给了我在北大继续深造的机会，以及这四年来对我的指导、帮助和人生道路的指引。从最开始的足底压力、到后来的强化学习，再到最后毕业论文的相关工作，李老师一直都十分支持我的选择；几次实验结果不顺利，李老师也是一直在鼓励我，并与我一起思考解决方法。感谢李老师一直以来对我的信任，我现在都还记得被通知要担任算法小班课助教时的忐忑心情；这几次助教经历，让我在各方面都得到了极大的提升。从李老师身上，我学会了很多很多。在学术之外，李老师经常与我们分享见闻与感悟，并常常鼓励我们要多读书、多思考。感谢李老师，您不仅仅是我学业科研上的导师，也是我人生的导师，像灯塔一样指引着我，您的谆谆教诲我将铭记于心。

感谢人工智能实验室的师兄师姐师弟师妹们，与大家一起学习、科研的日子总是快乐的。感谢张勤健老师对我方方面面的大力支持。感谢张海峰、洪星星师兄一直以来对我的认可，每次与他们的交流都让我收获颇多。感谢周昊宇师兄、周昱杉师妹在 Botzone 方面对我的帮助与支持。感谢许国雄老师、张艺师姐在合作研究足底压力时对我的启发。感谢林舒师兄，李昂、鲁云龙师弟，我在与你们的讨论中学习到了很多。感谢王鑫超同学三年来对我的一贯支持与帮助。希望你们的未来都能前程似锦。

感谢 42 号楼 229 宿舍的孙泽宇、舒斌和王珂，每次回到宿舍都能与你们开心地畅聊一番、甚是惬意。希望孙泽宇、王珂的博士生涯顺利，舒斌的工作一帆风顺。

感谢我的父亲和母亲，为我创造了良好的学习和生活条件，我爱您们。

在此特别感谢我的女朋友宋彭婧，在我几次濒临崩溃的时候，是你在我身边开导我、鼓励我，帮助我走出困境；在我取得成绩的时候，与我共同分享喜悦，并让我不要骄傲、继续努力。感谢你让我感受到家的温暖。

谢谢！

北京大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含任何其他个人或集体已经发表或撰写过的作品或成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律结果由本人承担。

论文作者签名：王政飞　日期：2020 年 5 月 23 日

学位论文使用授权说明

（必须装订在提交学校图书馆的印刷本）

本人完全了解北京大学关于收集、保存、使用学位论文的规定，即：

- 按照学校要求提交学位论文的印刷本和电子版本；
- 学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务，在校园网上提供服务；
- 学校可以采用影印、缩印、数字化或其它复制手段保存论文；
- 因某种特殊原因需要延迟发布学位论文电子版，授权学校□一年/□两年/□三年以后，在校园网上全文发布。

（保密论文在解密后遵守此规定）

论文作者签名：王政飞　导师签名：

日期：2020 年 5 月 23 日