本科生毕业设计（论文）

基于虚拟遗憾最小化的两人有限注德州扑克

院　　系　人工智能与自动化学院

专业班级　自卓 1601

姓　　名　薛博阳

学　　号　U201614481

指导教师　罗云峰

2020 年 5 月

学位论文原创性声明

本人郑重声明 :所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　年　月　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密口，在　　年解密后适用本授权书

2、不保密口。

（请在以上相应方框内打“√”）

作者签名：　　年　月　日

导师签名：　　年　月　日

摘要

德州扑克是一项风靡全球的扑克竞技游戏，是玩家之间智慧勇气与心理素质的综合对抗，也是人工智能和博弈论研究者的热点课题，在引入虚拟遗憾最小化算法后取得了突破性进展，并且多次战胜了人类选手。两人有限注德州扑克作为大规模非完备信息机器博弈的典型代表，对于解决经济政治等很多现实博弈问题也有重要的参考价值。而两人有限注德州扑克的游戏决策树信息节点数量也多达 3.19×10^{17} ，游戏状态随着对局呈指数级增长，规模庞大的游戏状态以及以上特性使得研究德州扑克算法极具挑战性。

本课题采用虚拟遗憾最小化算法（以下简称 CFR），通过遍历整个扩展式博弈决策树，计算并存储每个节点采取不同策略的遗憾值，多次迭代更新并生成策略，同时使用抽象方法降低复杂度，结合手牌牌力进行决策。对原有的 9-bucketing 卡牌抽象进行改进，将每个节点的手牌与公共牌组合牌力分区到 9 个桶里，在不引起空间复杂度增加的前提下根据公共牌的变化动态调整策略，进一步提高博弈性能；最后租用网上云服务器进行验证分析。

在编写了两人德州扑克的应用程序后引入 CFR 算法不断调整训练最优策略，最终实现一个具有较高智能水平的德州扑克博弈系统，证明 CFR 算法能够有效处理大规模的非完备信息扩展式博弈。而改进的 9-bucketing 策略，在小规模纯策略的测试中也表明性能较之前有所改善，符合预期结论。

关键词：机器博弈；德州扑克；虚拟遗憾最小化；卡牌抽象；改进的 9-bucketing 策略

Abstract

Texas Hold'em Poker is a popular and prevalent game all over the world, which is a also competition for wisedom, courage and mental quality among all players. It is aslo a hot spot for researchers and scholars in Artificial Intelligence and Game Theory, and researchers make great progress after using counterfactual regret minimization(CFR). As a typical representative of imperfect-information extensive game, Hold'em has great research value to slove economic or political problems. Even if the simplest Heads-up limit Hold'em, the number of information nodes in decision tree is over 3.19×10^{17}, which grows exponentially with game states. The large-scale game states also make it challengable to solve this problem.

This project utilizes Counterfactual Regret Minimization algorithm, which searches the entire decision tree to calculate and store the average regret and strategy for each node, and then update the regret in iterations to generate the strategies. In Hold'em, some abstractions is necessary to adopt to reduce the complexity of time and space, so I need to make decisions based on our current cards. I have proposed an improved 9-bucketing strategy, which judges the strength of maximum rank of all the hand and community cards, and also divides all situations into 9 buckets to adjust strategies dynamically without increasing the complexity. Finally, I rent an online cloud server to verify and analyse.

After completing the Heads-up Hold’em application at first, I utilize CFR to adjust optimal strategies, and a highly intelligent Texas Hold'em system is achieved, which verify that CFR is an effective algorithm to deal with large-sclae imperfect-information extensive games. This improved 9-bucketing method performs better in small-scale pure strategy tests than before, and the expected conclusion is obtained.

Keywords: imperfect-information extensive game; Texas Hold'em; counterfactual regret minimization; card abstractions; improved 9-buckting strategy

目录

1. 绪论

1.1. 课题来源，意义

1.1.1. 课题来源

自人类文明诞生之日起，就有了各种各样的博弈游戏。游戏是人类早期集益智与娱乐为一体的活动，传说四千年前就有了围棋。几个世纪以来，人们创造出不计其数的各类游戏，比如象棋、国际象棋、跳棋、扑克、麻将、桌游等。半个多世纪前，电子计算机诞生，从此各种博弈游戏焕发了新貌，人类进入机器博弈的时代[1]。

随着信息技术的发展，近几年来人工智能成为全世界最热门的学科之一，在诸多领域成就斐然。而博弈论（Game Theory）作为人工智能重要的子学科之一，活跃在博弈论方面的学者，一直致力于研发具有高智能的机器博弈体。随着计算机学三位里程碑式代表人物，计算机之父冯·诺依曼，人工智能之父图灵，信息论创始人香农在各自的研究领域所取得的成就，借助机器代替人类进行博弈计算的想法获得了硬件基础和软件理论支持[1]，同时也是早期计算机科学与人工智能学术观点的代表。近乎一个世纪以来，机器博弈一直是新算法思想的试验平台，也是人工智能发展的重要里程碑。比如，1994 年计算机程序 Chinook 在跳棋上击败人类顶级玩家，成为第一个赢得人类冠军的机器算法[2]；1997 年 IBM 开发的深蓝（Deep Blue）在国际象棋上击败 Kasparov[3]，Watson 在 Jeopardy（一种博弈游戏）上击败 Jennings 和 Rutter[4]。但是，击败顶尖的人类玩家并不等于“解决”游戏，而是计算出一种理论上可行的最优方案，该方案在所任何情况下都可以保持

不败。

迄今为止，人类解决的每一个游戏都是完备信息游戏。在完备信息游戏中，所有玩家在做出决策之前都会知晓当前场上的一切信息。大部分棋类游戏，比如象棋，围棋，五子棋等都是完备信息游戏。在非完备信息的游戏中，玩家并不总是知道场上已发生的全部信息（例如，在桥牌游戏中玩家并不知道其他玩家的手牌，再比如拍卖时卖方并不能预测拍卖品价值）。这些游戏更具挑战性，其理论知识，计算方法和已解决的实例均落后于完备信息游戏。尽管大部分室内游戏都是完备信息，但完备信息在现实世界的决策环境中却很少见。在与博罗诺夫斯基叙述的一次谈话中，现代博弈论的创始人冯·诺伊曼（von Neumann）得出了相同的看法：“现实生活中往往充斥着诈术、欺骗和诡计，我无法预测对手下一步行动。而博弈论正是帮助我寻找生活的内在规律的最合适的理论[5]。”

然而，牌类游戏与棋类游戏不同。大部分棋类游戏中所有玩家在做决策之前知晓游戏过程中已发生的情况[6]。这些游戏对于人工智能应用程序的难点在于游戏对局中的信息集，游戏状态，决策节点等数量，因此首先需要高级计算机，同时，扑克牌是一种非完备信息的博弈游戏（或不完美信息，imperfect-information game），其中非完备信息指的是玩家对当前游戏对局中场上所有信息并不完全知晓，比如对手的手牌。这些情况对计算机的理论算法提出更高的要求，迄今为止，非完备信息机器博弈并没有像完备信息机器博弈那样得到完美的解决。同时，人类的非完备信息博弈往往更加复杂，甚至会应用到心理学的知识，使用“诈唬”等方式虚张声势，所以非完备信息博弈向来被看作是对智商和情商的双重考验。

机器博弈是人工智能的重要分支，目前随着 AlphaGo 解决了最复杂的完备

信息博弈问题——围棋，机器博弈的研究中心已转到研究成果并不突出的非完备信息上[7]。2007 年 Albert 大学的 Martin 等人提出了基于虚拟遗憾最小化（CFR）的概念来解决大型博弈游戏[8]。2015 年 Albert 大学 Bowling 等人在《Science》上发表文章，对 CFR 进行改进，使得两人有限注德州扑克取得了突破性的进展，鼓舞了无数学者[9]。这已成为解决非完备信息的一种思路，本课题通过实现 CFR 算法，为以后的非完备信息机器博弈研究做铺垫。

1.1.2. 课题目的和意义

德州扑克是一项标志情商、智慧和勇气的牌类游戏，自 20 世纪初诞生便风靡全球，深受各个不同职业群体玩家的青睐[7]。由于德州扑克非完备信息特性，游戏玩家都有未知的场上信息，同时还有诈唬等心理战术。这些特点使得德州扑克更贴近生活中真实的拍卖、谈判等金融政治博弈场景，所以德州扑克也许是测试人工智能是否能够应付这些情景的一个平台。

前几年，AlphaGo 兴起了机器学习的热潮。在围棋这种完全信息的零和博弈中，作为算法需要解决的仅仅只是如何搜索大规模博弈树的问题，但是在德州扑克这种非完备信息博弈的问题中往往还藏有其他手段。而且对比围棋每次下棋有限的策略(19*19 个交叉点)德州扑克的策略几乎是无限的，专业选手往往押注在 1000 到十几万甚至百万美元，中间押注 10000$和 10500$也存在差距。所以由于德州扑克的不确定性，非完备性和扩展式等特性，在 AlphaGo 4:1 战胜李世石的时候[6]，同期的德州扑克还只能勉强解决两人有限注的问题。

解决此类非完备信息博弈，如果解决思路是从寻找纳什均衡出发，那么解决手段的基础就是虚拟遗憾最小化（Counterfactual regret minimization，CFR）[8]。

现代的博弈论快速与人工智能进行结合，形成了以数据驱动的博弈论新的框架。对于非完备信息博弈，其情况与完备信息博弈（如围棋）有很大的区别。博弈者不仅需要推理出他们对手到底知道什么，并且还要推理出他们什么样的行动会暴露自己的私人信息。解决非完备信息博弈的经典方法是计算或逼近所谓的纳什均衡。直到 2007 年，阿尔伯塔大学计算机扑克研究小组的成员在 Michael Bowling 的领导下实现了一个突破，他们研究的虚拟遗憾最小化便是这一次突破的产物[8]。

扑克作为一类更为复杂和典型的非完备信息机器博弈问题，其信息不确定不完备、快速决策、重复博弈等特性集中了人工智能领域许多核心问题，这类游戏的研究对于整个人工智能领域有着极其关键的影响[10]。而虚拟遗憾最小化算法通过从博弈的过程中吸取以往做决策的经验教训，多次迭代训练出高效的策略，以使智能体未来所做的动作最小遗憾。从该算法的直观含义上可以看出，这一算法模型的研究对解决现实生活中的金融股票、拍卖等经济问题十分有效，通过使每一轮的决策做到最小遗憾，尽可能的减少了自身的损失，从而使收益最大化。

自 2006 年起，世界年度扑克机器博弈大赛每年举办一次，并由人工智能国际协会（AAAI）举行，成立之初就成为数十个研究小组和业余爱好者的关注焦点[11]。 2015 年 Albert 大学 Bowling 等人对 CFR 进行改进，在两人有限注德州扑克取得的突破性的进展更是鼓舞了无数学者[9]。因此研究 CFR 算法对于大规模的非完备信息扩展式机器博弈有着重要的研究价值和现实意义。

1.2. 国内外研究现状

1.2.1. 国外研究现状

随着机器博弈不断发展，一度取得了显著成绩。在相继解决了西洋跳棋、奥赛罗等完备信息机器博弈之后，国外的研究者们逐渐将目光转向非完备信息博弈[4][5]。非完备信息机器博弈自上世纪 90 年代开始成为人工智能领域研究的一个热点，特别是 1995 年 Billing 对扑克游戏的研究掀起了研究者们对扑克类机器博弈研究的热潮[12]。Chinook 对抗世界跳棋冠军 Tinsley 赢得第一场比赛过去 17 年后[13]，电脑程序北极星赢得了与职业扑克玩家的第一场比赛。尽管两人有限注德州扑克 的信息量小于跳棋，但它的非完备特性使它成为对于计算机而言更具挑战性的游戏。

1997 年，Billing 等人成立了加拿大 Albert 大学计算机扑克研究组(University of Alberta Computer Poker Research Group, UACPRG)，并一度成为世界上最先进的扑克机器博弈问题的研究组织，产生了该领域的一大批学术成果，并发表了很多具有重要意义的文章[12]。这些文章大部分针对德州扑克的研究，并进一步开发出了对手建模和神经网络建模等技术。1999 年，第一个基于规则的德州扑克程序诞生，其智能水平已经达到人类玩家平均水平[14]。2003 年，Darse 等人针对德州扑克游戏状态巨大的问题，首次提出了对游戏状态进行抽象的思路，根据此策略创建了具有较高水平的机器博弈程序，主要用于解决二人限注德州扑克问题[15]。2006 年年，世界年度扑克机器博弈大赛(Annual Computer Poker Competition, ACPC）开始由 AAAI 和 IJCAI 会议联合举办，成为非完备信息机器博弈领域的最权威评测机构[11]。同时段，研究者们将纳什均衡逐渐引入德州扑克的研究。2007

年，Alberta 大学的 Martin Zinkevich 等人提出了基于遗憾最小化的虚拟遗憾的概念来解决大型博弈游戏，提出的 CFR 算法是将整体最小遗憾值分解到各个独立的信息集中计算局部最小遗憾值[8]。实践证明，在大型的两人零和游戏中使用该方法近似纳什均衡。2008 年，Polaris 在相关扑克游戏上首次击败了人类玩家，具有里程碑式的意义[16]。2015 年，来自 CMU 的 Sandholm 宣布最小规模的扑克游戏二人有限注德州扑克问题已经被解决[9]。同刊中，Bowling 等人证明了即使由于随机性的存在，在少量的对局中仍然会有所亏损，但利用统计方法从大数据的角度看，最终是可以取胜的[9][17]。

两人有限注德州扑克问题缓慢的进展并非缺乏这方面的探索研究。扑克一直是人工智能，运筹学和心理学三大领域的难题，其研究历程可以追溯到 40 年前[18]。17 年前，Koller 和 Pfeffer 就宣称：“我们离解决大型扑克之类的游戏还差得很远，而且我们目前也不太可能做到这一点[19]。”

1.2.2. 国内研究现状

国内的计算机博弈起步晚发展快，尤其近些年随着人工智能热度的上升，机器博弈也一度成为热门研究学科，诸多高校取得不错成就。东北大学是国内最早进行象棋机器博弈领域研究的院校之一，逐步整理出一套完整的棋盘表示、走法生成、基于自适应遗传算法的估值函数、开局库与残局库设计、参数优化等象棋博弈智能体体系，标志着象棋机器博弈已发展至较高的水平[20][21]。

哈尔滨工业大学很早从事非完备信息机器博弈的研究[22]，在理论和实践上都取得了一定成果。2013 年哈工大课题组利用手牌评估和对手建模方法构建的德州扑克博弈程序，在 ACPC 大赛中的两人限制性德州扑克项目中荣获第四名。

主要研究成果有非完备信息机器博弈系统的信息表示、启发函数优化算法、大规模博弈树搜索、对手建模、风险模型分析等[22][23]。

1.3. 课题要求

1.3.1. 预计达到的目标

⑴ 了解德州扑克的规则，掌握德州扑克的一般玩法，达到入门水准，以作娱乐消遣。

⑵ 理解 CFR 算法思路，并且能够编写 CFR 的主要代码，并运用到如 Kuhn 扑克或德州扑克这类游戏中并取得一定成果。

⑶ 更加熟悉 Python 语言编程和面向对象编程，增强模块化，构思好主要框架，写出整洁美观实用性强可移植性强 bug 少的代码。

⑷ 在两人有限注德州扑克中使用虚拟遗憾最小化的工程中利用已编写的德扑平台和 CFR 算法训练出强力 AI，能达到较好效果，并利用远程服务器进行测试训练。

1.3.2. 关键理论和技术

⑴ 在两人有限注德州扑克中使用虚拟遗憾最小化的工程中利用已编写的德扑平台和 CFR 算法训练出强力 AI，能达到较好效果，并利用远程服务器进行测试训练。

⑵ 德州扑克的规则思路。这是一切的基础，只有非常熟悉规则流程，才能编写出 bug 较少的程序，并结合一些自己的理解适当加入抽象策略和方法，减少计算量并且提高智能程度。

⑶ Python 语言，尤其是面向对象的编程，Python 比 C 语言及 C++算法更高级，

适当使用一些工具包可以极大地提高简洁程度；但最重要的还是要有清晰的思路框架。

⑷ 虚拟遗憾最小化（CFR）。虚拟遗憾最小化算法是目前最具有竞争性应用最广泛的算法。CFR 是一种通过两种遗憾最小化算法之间反复训练的过程来逼近扩展式博弈纳什均衡的迭代方法。通过多次迭代，将整体遗憾分解到各个独立 的信息集中计算局部最小后悔值，后悔值是算法没有选择最佳策略而损失的游戏收益，只有在执行策略后才知道。遗憾最小化算法可以使后悔值随着游戏进行呈线性增长，最终实现与采用最优策略相同的效果。CFR 的关键点在于不会存储呈指数增长的决策数量的后悔值，而是存储和最小化在每个信息集和相应后续决策中的不断修改调整的后悔值，这样就可得出任何决策下的后悔值。通过在每一次迭代中对每个玩家的策略取平均值，可以近似求出纳什均衡，并且随着迭代次数的增加，近似程度也会提高。

1.3.3. 主要研究内容

以德州扑克作为实验的研究对象，利用虚拟遗憾最小化算法不断调整学习最优策略，最终实现一个具有较高智能水平的德州扑克博弈系统，并针对德州扑克的特点对算法进行改进创新，具体如下：

⑴ 根据现有的博弈理论基础知识，利用 2-Kuhn 扑克游戏进行简单上手并归纳其和德州扑克游戏的博弈特点，根据当前非完备信息机器博弈研究的背景及现状，分析并实现当前针对德州扑克研究的 CFR 方法。

⑵ CFR 算法是根据迭代计算游戏中每个信息集的遗憾值和平均策略值来决策下一步所做的动作的。简单的两人限注德州扑克玩法的博弈树都有多达10^{14}的信息集，因此要实现 CFR 算法在德州扑克上的应用必然要结合合适的归

类和抽样方法来降低德州扑克的状态空间复杂度。

1.3.4. 完成课题的方案和主要措施

用 python 语言实现基于虚拟遗憾最小化（CFR）算法的两人有限注德州扑克完整代码，并训练出具有高智能体的 AI。

前期通过阅读大量文献理解 CFR 算法的思路和发展历程，并通过练习基于 CFR 的两人 Kuhn 扑克这一只有 12 个游戏状态的简单扑克游戏快速上手，进一步加深理解，并构思出整体框架；同时，通过资料以及观看职业比赛视频掌握德州扑克的游戏规则和流程。

中期代码实现阶段，采用 Python 语言以及 PyCharm 和 Jupyter 编辑器。由于 Python 的面向对象特性因此特别适合编写博弈游戏的流程。在确定整体框架后就开始编写。首先确定类，游戏平台，玩家，牌桌，扑克牌，荷官，动作，策略，牌级（摊牌后手牌与公共牌的最大牌力组合），牌力（两张手牌）等。

在未加入 CFR 策略时用玩家-玩家对战测试程序，无误后再引入 CFR 算法计算游戏策略并训练高智能体。由于是两人对局，因此对局过程相对简单。首先创建一局游戏，玩家个数默认为 2，给每人分配一定的筹码，并分别下大小盲注；创建荷官和牌桌并生成 52 张扑克牌，随机洗牌后，荷官给两名玩家发两张手牌并出示三张公共牌，根据玩家不同的对局状态（翻牌前，翻牌，转牌，河牌）预先设置好可选择的动作集合（看牌，下注，跟注，加注，弃牌），玩家在不同阶段选择。如果有一人选择弃牌，则另一名玩家获胜。如果没有人弃牌，则河牌后比较两人牌力大小，牌力大者获胜。

采用一定的抽象方法结合 CFR 算法迭代更新策略，然后训练出具备较高水平的智能机器。

1.4. 章节安排

具体的思路见下，第一章介绍了机器博弈的背景和德州扑克的研究意义，国内外机器博弈的发展情况，以及本课题的要求，理论和实现方法。第二章介绍了正则博弈和扩展式博弈的知识，纳什均衡以及德州扑克的规则，使得读者有一个初步的认识，然后全面介绍了虚拟遗憾最小化的理论知识以及在德州扑克中的具体应用，以及辅助 CFR 算法的抽象策略。第三章介绍了关于课题应用程序的实现方法，测试过程，并得出了预期结论。第四章结合本次课题浅谈对德州扑克机器博弈的总结与展望。

2. 基础理论

2.1. 正则博弈和扩展式博弈

2.1.1. 正则博弈

正则（Norm Form Game）形式是博弈论中描述博弈的一种方式，一般使用矩阵来描述博弈过程。这种表示方法往往更加直观，并且相对简单，可以让读者一目了然识别出较优势的策略以及纳什均衡解，而且这些策略往往是同时决策的，比如剪刀石头布游戏，囚徒问题等，都可以简洁明了的表示。缺点是会有信息的损失，同时对于大规模的博弈问题往往显得无能为力。但我们仍然可以通过正则形式的博弈看出每个玩家的可能决策以及相应的收益[9]。

正则形式的博弈可用 (N, A, u) 形式表示:

⑴ 有限集 $N = \{1, 2, \ldots, n\}$ 表示参与博弈的玩家集合。

⑵ $A = A_1 \times A_2 \times \ldots \times A_n$，$A_i$ 表示玩家 i 可选取的的动作集合，A 表示某一时刻所有参与的玩家的策略集合，把 A 称为策略集合；

⑶ 对于每个 $i \in N$，u_i 表示收益函数: $A \to R$，每个参与玩家从相应的动作集合中选取执行一个动作，u_i 表示玩家 i 在策略集合 A 下的可能收益。

如果 $n = 2$ 并且 $u_1 = -u_2$，则我们称这个博弈是零和博弈，否则就是非零和博弈。但如果是对于 $n = 3$ 的博弈游戏并且 $u_1 + u_2 + u_3 = 0$ ，并不是零和博弈。

但如果有 $u_1 + u_2 = C$，我们可以称之为常和博弈，这样的游戏很容易转变成零和博弈且不需要改变原有的策略，仅需要在其中一个玩家的收益中减去常数 C 即可实现到零和博弈的转换，零和博弈对于接下来的两人有限注德州扑克有着很

重要的意义。

2.1.2. 扩展式博弈

不同于正则形式的矩阵表示形式，扩展式博弈一般使用决策树的结构描述博弈过程，每个决策节点表示博弈游戏中可能存在的一个游戏状态，叶子节点代表游戏结束，这种表示形式往往用于参与玩家不是同一时刻做出决策，而是轮流进行，因此棋牌类游戏都比较适合采用博弈树的形式来表示。扩展式博弈决策树从唯一的初始节点开始，通过游戏参与者的决策，沿着某一路径到达叶子节点，游戏结束的同时，在叶子节点处需要计算每个游戏参与者的收益值。每个非叶子节点只由一个玩家决策，参与者往往需要在非叶子节点选择一个动作，通过一个动作到达另一个节点。同时每一个游戏参与者可以多次做出决策，并且在不同的游戏状态时可以选择不同的策略，但凡涉及非完备信息与随机事件同时存在的游戏，扩展式博弈均能更有效地建立相关的数学模型[23][24]。

在一个扩展式博弈的决策树结构中，非叶子节点表示玩家的决策，在每个节点处场上一名玩家做出决策，叶子节点计算游戏结束时各人的收益值。信息集是同一类游戏状态的集合，在实际表示中对于某一个玩家而言，可见的游戏状态划分为一类，扩展式博弈游戏形式化构成要素 (N, H, Z, I, p, δ)定义如下：

⑶ 有限集 $N = \{1, 2, \dots, n\}$ 表示参与博弈的玩家集合。

⑷ 序列的有限集 H 表示已发生的历史动作序列， H 中的每个元素 h 的前缀也位于 H 中，每个不同的动作序列均代表一个决策节点。

⑸ Z 是终点状态集合， Z 中每个元素均代表叶子节点。

⑹ $A(h) = \{a: (h, a) \in H\}$ 代表序列 H 后可采取的动作集合。

⑺ $P(h)$为每个非叶子节点 h 后的可以行动的玩家。如果是$P(h) = c$，即确定行

动玩家 c 可以采取的策略。

⑻ 函数 f_c 是在 $P(h)=c$ 时当前节点 $f_c(\cdot|h)$ 关于 $A(h)$ 的一种概率度量，$f_c(a|h)$就是出现 h 是动作 a 出现的概率，其中 $a \in A(h)$，$f_c(a|h)$均为独立同分布。

⑼ 对于每个玩家 $i \in N$，存在这样一个信息分区：$\mathcal{I}_i = \{\mathrm{h} \in \mathrm{H}: \mathrm{P(h)} = \mathrm{i}\}$，$\mathcal{I}_i$ 具有以下属性：若 h 和 h' 在均属于 $\mathcal{I}_i$，则 $A(h)=A(h')$。建立一个信息集 $I_i \in \mathcal{I}_i$，对于任意 $h \in I_i$，同时包含可采取动作 $A(h)$ 以及下一名玩家 $P(h)$ 等信息。本质上 $\mathcal{I}_i$ 是同一名玩家具有相同后续可选取动作的信息集合，并存储可采取动作即下一名玩家等信息，用于在实际训练中存储更新策略。

⑽ 对于每个玩家 $i \in N$，函数u_i表示从叶子节点集合 Z 映射到实数集 R 上的收益值。定义 $\Delta_{u,i}=\max_Z u_i(z)-\min_Z u_i(z)$ 为玩家 i 的收益值范围。

⑾ 给定一个策略 δ，$\pi^{\delta}(h,h')$ 为在信息节点 h 处依据 δ 策略执行动作 a 后到达 h' 节点的概率，$\pi_{-i}^{\delta}(h)$表示除了玩家 i 其他玩家信息节点 h 处依据 δ 策略执行动作 a 后到达 h' 节点的概率。

2.2. 纳什均衡

2.2.1. 囚徒问题的纳什均衡

以最经典的囚徒困境为例，决策者可以利用下面的收益矩阵剔除劣势策略[26][27]。囚徒困境的问题背景是，有两名罪犯合伙作案，落网后分别审讯，警方采取以下方式。两个囚徒被告知：如果两人都不承认罪行，那么各判一年；如果两人都招供，那么各判八年；如果一个招供而另一个不承认，招供的放出去，抵赖的判十年。对于一个囚徒而言，选择招供貌似好一些，因为最多判八年，甚至直

接释放，于是他们都更倾向于招供，所以都被判处八年，很显然没有达到纳什均衡解。问题本身是一个非零和博弈，如果从对方的角度考虑，很显然抵赖可以使两人共同被判处的刑期更少。对竖排决策者来说，比较每列第一个数字可以发现，由于$10 > 8$、$1 > 0$，因此无论横排决策者怎样选择，抵赖都是最优策略；同理，对于横排决策者来说，比较每列的第二个数字表明无论竖排决策者如何选择抵赖都能够获得更多的收益。因此，此博弈存在且唯一的纳什均衡是（抵赖，抵赖）。如表 2.1 所示：

	抵赖	招供
抵赖	$(1，1)$	$(0，10)$
招供	$(10，0)$	$(8，8)$

表 2.1 囚徒问题纳什均衡解

2.2.2. 纳什均衡

不完全信息的博弈是一种玩家对正在玩的游戏没有共同知识的博弈，也就是说，不能只通过子游戏的信息解决子游戏，这样通常达不到全局最优。在相关研究中，通常的做法是逼近游戏中的纳什均衡策略，也就是每个玩家都无法通过只改变自己的策略来提高自己的收益。在零和游戏中，如果每个玩家都选择相对对手而言的最优策略，那么游戏整体就趋近于纳什均衡，每个玩家的策略就是纳什均衡策略[28][29]。

对纳什均衡更为直观的解释是：对于任意的 $\epsilon \geq 0$，任意一名玩家单方面的违背策略 δ 导致获得的期望收益不高于ϵ，就说策略δ是 $\epsilon -$纳什均衡的。

严格定义如下：对于任意的 $\epsilon \geq 0$，策略 δ 是 $\epsilon -$纳什均衡的，则

$$\max_{\acute{\delta}_i \in \sum_i} u_i\left(\acute{\delta}_i, \delta_{-i}\right) \leq u_i(\delta) + \epsilon, i \in N \tag{2.1}$$

纳什均衡至少可以避免犯严格劣策略错误：不论对方采取什么策略，已方采取的这个策略所获得的收益都不如其他策略。虽然在一两局的游戏中由于非完备信息游戏的随机特性纳什均衡策略未必最优，但从长远的统计学角度来看，纳什均衡策略一定是最优策略。

2.3. 德州扑克

2.3.1. 游戏规则

德州扑克是一种非常受欢迎扑克牌游戏，也是赌场上很常见的游戏，现在已经成为一种职业比赛。一般情况下一局德州扑克由 2-10 人参与，根据课题要求，本文以两人游戏为背景介绍德州扑克的规则。

德州扑克一共有 52 张牌，四种花色的 A-K 各一张。每名玩家有两张仅自己可见的手牌，还有五张由荷官在不同游戏阶段陆续发出场上所有玩家都可见的公共牌。玩家根据自己手牌与公共牌组合的牌级进行下注或弃牌，经过四个阶段的押注后，若仍有两个以上的玩家没有弃牌，那么所有对局中的玩家会进行摊牌，亮出各自的手牌以较与公共牌的最大组合，牌级较大的玩家获胜。游戏顺序是沿桌顺时进行，庄家的左手边的两名玩家分别下大小盲注，其他玩家依次轮流下注。两人德州扑克第一名玩家先下小盲注，称为玩家一（下同）；第二名玩家下大盲注，类似于庄家身份，称为玩家二，后手选取策略。

有限注区别于无限注，指的是玩家不能一次性将筹码全下（All-in），也就是赌博电影里的“梭哈”，因此不会在未到最后一轮下注前摊牌。有限注保证玩家总是有足够的筹码进行跟注和加注直到有人弃牌或进行摊牌。而在无限注型玩法中，

玩家可以选择将自己所有的剩余筹码投到底池中，不能再进行跟注和加注了，这种玩法虽然更加受玩家的欢迎，但是其研究价值却不如有限注的德州扑克了，选择全下的玩家在后续的摊牌阶段仍然参与牌型的比较并决胜负[30]，本文仅讨论与有限注相关的问题。

2.3.2. 游戏阶段

游戏阶段包括：翻牌前（Pre-flop），翻牌（Flop，即前三张牌公共牌），转牌（Turn，第四张公共牌）和河牌（River,第五张公共牌）。

翻牌前不发放底牌，两位玩家依次下大、小盲注，荷官为各玩家发出两张底牌。从玩家一开始下注，可选择下注（Call）、加注（Raise）、弃牌（Fold），玩家二基本相同，只是将下注改为跟注，但注额相同。跟注或加注都是由上一个玩家下注的的筹码而决定，在没有人加注的情况下，大盲注位置的玩家行动结束后，本轮押注行动结束。若有人加注的情况下，直到最后加注玩家右则的第一个玩家行动结束后，本轮押注行动结束。翻牌阶段荷官发出 3 张公共牌，所有人可见。依旧从玩家一开始选择弃牌、看牌或下注。后面的玩家选择同第一轮。转牌阶段荷官发出第 4 张公共牌到牌桌中央，所有人可见，本轮押注行动与翻牌一致。河牌阶段荷官发出第 5 张公共牌到牌桌中央。所有公共牌都发出后，各玩家的牌面都固定下来，需再次判断形势作出合适的行动。河牌结束后，没有弃牌的玩家摊牌，每位玩家用两张手牌与五张公共牌组合最大牌型，牌型最大的玩家赢得其他玩家下注的筹码。若最终大小相同，两名玩家只能平分底池。中途若有玩家弃牌，则牌局提前结束，奖池归另一名玩家。

2.3.3. 牌级比较

五张牌牌型组合由大到小依次为：

⑴ 皇家同花顺（Royal Flush）：同一花色最大的顺子。

⑵ 同花顺（Straight Flush）：同一花色的顺子，牌面较大者胜，下同。

⑶ 四条（Four of a kind）：四张相同和一张单张。

⑷ 葫芦（Full House）：三张相同和一对。

⑸ 同花（Flush）：五张牌花色相同。

⑹ 顺子（Straight）：花色不同的顺子。

⑺ 三条（Three of a kind）：三张相同和两张单牌。

⑻ 两对（Two Pairs）：两对对子和一张单牌。

⑼ 一对（Pair）：一对对子和三张单牌。

⑽ 单张（Nothing）：五张单牌。

如下为各牌级组合示意图：

图 2.1 牌级组合示意图

2.4. 虚拟遗憾最小化算法

2.4.1. 发展背景

博弈游戏达到纳什均衡时，每个参与者的均衡策略都可以帮助该玩家获得预期最大收益，如果其他玩家也遵循这样的原则，那么任何玩家都无法通过单方面违背该策略来增加其预期收益。所有有限的扩展式博弈都至少有一个纳什均衡，在零和博弈游戏中，玩家在所有均衡状态下的期望收益值都相同。一个ϵ-纳什均衡对每个玩家而言，是一种可以使得任何玩家都不能通过选择其他策略来增加自己的收益值是的策略。按照 Allis 的分类，如果能计算出零和博弈的收益值，那么这种情况属于极弱拆解；而如果能计算出纳什均衡策略，则是弱拆解。如果一个游戏计算出的纳什均衡值极小，甚至一个人终其一生玩过的无数次后都从统计上无法区分纳什均衡解的话，我们称该解为究极弱拆解。对于完备信息博弈游戏，求解纳什均衡通常涉及对游戏决策树的（部分）遍历。但是，相同的处理方法不能用于非完备信息博弈。我们简述了解决非完备信息博弈的发展现状，并通过在解决复杂度越来越大的扑克游戏方面所取得的进步来衡量算法，如图 2.2 所示。

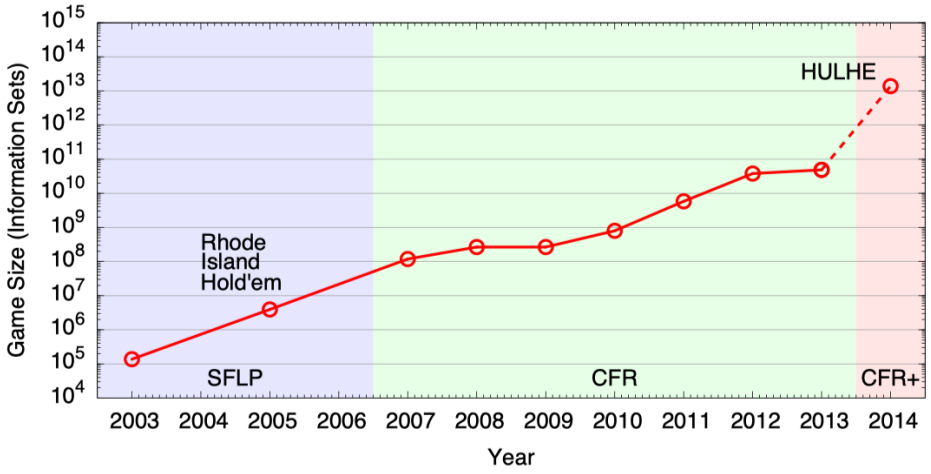

图 2.2：随着时间的推移，以独特信息集（即去除对称性后）衡量的非完备信息游戏的复杂度不断增加。不同颜色的区域代表所用的算法。

正则式线性规划：解决扩展式博弈游戏的最早方法是将其转换为正则博弈，将原始扩展式博弈中每对可能的策略值用矩阵表示，然后使用线性代数求解。然而，可确定的策略数量与游戏的信息集数量成指数关系。因此，尽管线性规划可以用数千种策略来处理正则式博弈，但即使只有几十个决策点也是不可行的。两人 Kuhn 扑克是一种有三张牌，一个下注回合和最大信息集数量为 12 的扑克游戏，这种情况可以用线性规划解决[31]。 但是即使是 Leduc Hold'em，有 6 张牌，2 轮下注回合以及 2 次最大注（总共有 288 个信息集），也具有超过 10^{86} 种确定性策略。

序列式线性规划：Romanovskii 和 Koller 等人建立了解决非完备信息游戏的现代理论，提出一种用序列形式表示策略的方法。通过简单更改变量，他们得出扩展式博弈可以用线性形式表示并直接解决，而无需将指数形式转换为一般形式。序列式线性规划是解决扩展式非完备信息博弈的第一个算法，其计算时间随着表示游戏策略的多项式的增加而增加[32]。2005 年，Gilpin 和 Sandholm 将 SFLP 与一种寻找游戏对称性的自动化方法一起解决了 Rhode Island Hold'em，这是一款合成的扑克游戏，在去除对称性后有 3.1×10^6个信息集。

虚拟遗憾最小化：2006 年，第一届年度计算机扑克大赛举办后，比赛中算法在解决复杂度较大的游戏方面不断取得进步，提出了多种改进方式。虚拟遗憾最小化（CFR）是目前最具有竞争性应用最广泛的算法[33]。CFR 是一种通过两种遗憾最小化算法之间反复训练的过程来逼近扩展式博弈纳什均衡的迭代方法。通过多次迭代，将整体遗憾分解到各个独立的信息集中计算局部最小后悔值，后悔值是算法没有选择唯一的最佳策略而损失的游戏收益，只有在执行策略后才知道。遗憾最小化算法可以使后悔值随着游戏进行呈亚线性增长，最终实现与采用最优

策略相同的效果。CFR 的关键点在于不会存储和最小化呈指数增长的决策数量的后悔值，而是存储和最小化在每个信息集和相应后续决策中的不断修改调整的后悔值，这样就可得出任何决策下的后悔值。通过在每一次迭代中对每个玩家的策略取平均值，可以近似求出纳什均衡，并且随着迭代次数的增加，近似程度也会提高。该算法所需的内存与信息集的数量是线性关系，而不是二次关系，高效的 LP 方法就应该是这样[34][35]。由于之前解决大型博弈游戏通常会受到可用内存的限制，CFR 的出现使得可解决的游戏复杂度较之前大幅提升，比如 Koller 等人近期取得的成就。自 2007 年被提出后，CFR 已用于解决更为复杂的两人有限注德州扑克简化问题，在 2012 年可以处理具有 3.8×10^{10} 个信息集的博弈游戏。

2.4.2. 遗憾匹配

博弈游戏中在同一个游戏状态历史存储的遗憾值可以对接下来相同状态的策略进行定向的纠正，这个纠正过程使用的算法是遗憾匹配算法[8][36][37]。遗憾值的含义是在当前状态下，选择行为 A 而不是行为 B 的后悔程度。比如，在石头剪刀布中，对方出了剪刀，自己出了布，这个时候，自己会更后悔没有出石头，相比之下不那么后悔没出剪刀。

一个遗憾值的定义是后悔没选的动作 a 达到的状态的遗憾值与当前状态的遗憾值的差值。比如在上面的例子中，在对方出剪刀的情况下，自己出布的收益值是-1；出剪刀的收益值是 0；出石头的收益值是 1。因此己方选择布而不是剪刀的后悔值 R（剪刀->布）= 1；而不选择石头的后悔值 R(石头->布）= 2。

定义了后悔值之后，我们可以应用遗憾值的方法来进行决策，其思路是倾向于选择后悔值更大的行为。比如，在石头剪刀布中，如果我们进行了 n 次游戏，然后把每次游戏所有动作的后悔值加起来，然后做一个归一化，就可以得到一个

概率分布作为我们的策略：

$$R(a) = \sum_{i=1}^{n} R_i(a_i) \tag{2.2}$$

$$P(a) = \frac{R(a)}{R\left(\text{石头}\right) + R\left(\text{布}\right) + R(\text{剪刀})} \tag{2.3}$$

而在每一步的后悔值由对手的行为所决定，所以遗憾匹配算法其实是在根据对手的策略改变我们自己的策略。不难看出，该算法通过计算整体的平均遗憾值可以优化策略选择，在扩展式博弈游戏中，由于游戏状态数量过于庞大，还需要引入虚拟遗憾最小化策略得到下一动作的可选策略。

2.4.3. 遗憾最小化

遗憾值是一种在线学习概念，由此产生了一系列计算能力强大的学习算法。为了定义这个概念，首先假设重复进行一个扩展式博弈游戏。 令 δ_i^t 为第 t 轮玩家 i 所使用的策略。在时间 T ，玩家 i 的平均遗憾值为：

$$R_i^T = \frac{1}{T} \max_{\delta_i^\star \in \sum_i} \sum_{t=1}^{T} \left(u_i(\delta_i^\star, \delta_{-i}^t) - u_i(\delta^t) \right) \tag{2.4}$$

公式含义：重复进行 T 次博弈后的平均遗憾值，为每一次使 $u_i(\delta_i^\star, \delta_{-i}^t)$ 最大的 $\delta_i^\star$ 的收益值与选择 δ_i^t的收益值之差，作为第 t 次的遗憾值，累积 T 内所有遗憾值，并除以 T。

此外，定义 $\bar{\delta}_i^t$ 是玩家 i 从时间 1 到 T 的的平均策略，对于每个信息集 $I \in \mathcal{I}_i$ 中可采取动作 $a \in A(I)$ ，有：

$$\bar{\delta}_i^t(I)(a) = \frac{\sum_{t=1}^{T} \pi_i^{\delta^t}(I)\delta^t(I)(a)}{\sum_{t=1}^{T} \pi_i^{\delta^t}(I)} \tag{2.5}$$

是一种加权计算平均策略的方式，并且这个公式完美建立了遗憾值与纳什均衡之间的关系。注：具体符号定义见扩展式博弈部分介绍。

在迭代次数 T 的零和博弈中，如果两个玩家的平均总体遗憾小于 ϵ，则在这段时间内采取的平均策略 $\bar{\delta}_i^t$ 是 2ϵ 的纳什平衡。

自我博弈的遗憾最小化算法可以当做一种逼近纳什均衡解的方法，此外，算法平均遗憾程值的限制也影响了纳什均衡近似解的收敛速度。

传统方法的遗憾最小化一般用于解决类似囚徒困境之类的问题。尽管从理论上讲可以将任何有限的扩展式博弈转换为等效的正则博弈，但随之而来的是指数增长的游戏状态，使得遗憾最小化算法并不可行。所以，我们需要一种更轻松实现扩展式博弈算法。

2.4.4. CFR 算法分析

CFR 算法基本思想是将整体博弈决策树的遗憾值分解到一组子博弈树上，并且能够独立实现遗憾最小化。特别是，为扩展式博弈引入了一个新的概念，称为虚拟遗憾，它是根据单个信息集定义的。算法表明总体遗憾值受虚拟遗憾之和的限制，并且还表明如何在每个独立的信息集上将虚拟遗憾最小化[8]。

首先考虑一个特定的信息集 $I \in \mathcal{I}_i$ 和玩家 i 在该信息集中做出的决策。定义虚拟收益 $u_i(\delta, h)$ 为预期收益，h 为历史动作集合序列，δ 为当前玩家使用的策略。此处虚拟的含义是玩家 i 到达信息集 I 时假设采取的策略。最后，对于所有所有 $a \in A(I)$，定义$\delta|_{I\to a}$ 是一组在信息集 I 中除了选择动作 a 之外相同的策略配置文件。定义当前的虚拟遗憾为：

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\delta^t}(I)\left(u_i(\delta^t|_{I\to a}, T) - u_i(\delta^t, T)\right) \tag{2.6}$$

由于最小化当前的虚拟遗憾值会最小化整体遗憾值，因此，如果我们只能最小化当前的虚拟遗憾值，就能使我们找到近似的纳什均衡。则对于信息集 I_i，通

过遗憾匹配得到当前的策略集合：

$$\delta_i^{T+1}(I_i, a) = \begin{cases} \dfrac{R_i^{T,+}(I_i, a)}{\sum_{a \in A(I_i)} R_i^{T,+}(I_i, a)}, & if \sum_{a \in A(I_i)} R_i^{T,+}(I_i, a) > 0 \\ \dfrac{1}{|A(I_i)|}, & otherwise \end{cases} \tag{2.7}$$

CFR 算法在一次迭代过程中会在博弈树的每一个节点上计算虚拟遗憾值，当下一次迭代开始时，将会带入上次迭代的结果通过遗憾匹配计算新的博弈策略。CFR 算法不断的迭代执行，递归的回溯博弈树进行计算并更新策略，从而最小化玩家的平均遗憾值。

CFR 算法一开始对于每个策略都是从随机选取，并且计算出每个动作的遗憾值，并反复迭代优化策略。每次优化用的是遗憾值最小化的办法，把到目前为止的累计遗憾值分别对比，看选择哪一个动作的累计遗憾值低，以后多走这一步。为防止被对手建模，一般采用轮盘赌转法随机选择策略。

2.4.5. 德州扑克 CFR+算法

为了解决计算方面的困难，来自 UACPRG 团队的 Michael Bowling 等人发明了一个叫做 CFR+的算法，这种算法是 CFR 的一个变体[9][38]。CFR 算法每次迭代执行时分别对子博弈树进行更新，对每个非叶子节点执行后悔匹配算法。CFR+算法主要在迭代过程中约束遗憾值为非负数，如果遗憾值为负可以直接舍弃掉，这一点比较容易实现，同时也极大的减少了计算量，逐渐逼近纳什均衡，同时也非常适合大型博弈游戏的并行化处理。Allis 对于游戏解决的程度给出了三个不同的定义。首先明确定义，我们称这种解决游戏的方式为“拆解”（solve）[17]。对于二人（回合制、完全信息、不存在运气成分、有限）游戏，拆解分为以下几个层次：极弱拆解（Ultra-weakly Solved）：证明在初始局面下，

双方的最大收益值是否可以确定存在 d，只需证明存在性即。弱拆解（Weakly Solved)：给出一个算法，能从初始局面开始保证先行方（或者后行方）取得最大收益，无论其对手采取何种策略。强拆解（Strongly Solved)：给出一个算法，能从任何局面开始给出双方的最强应对，即当前状态的任意一方都能获得最大收益。在非完备信息的游戏中，从一开始策略就不是不是唯一时，Allis 的“强拆解”方法就无法很好的实现。此外，由于玩家策略或游戏本身具有随机性，非完备信息游戏通常具有不同的实际收益值，而不是单一的结果（例如在国际象棋和跳棋中“胜局”，“败局”和“和局”）。最后，游戏的理论收益值通常是近似估计的，因此解决这类问题的一个关键因素是衡量解决方案的近似程度。Bowling 等人证明了利用其改进的 CFR+算法已能够在两人有限注德州扑克中取得弱拆解。即使由于随机性的存在，在少量的对局中仍然会有所亏损，但利用统计方法从大数据的角度看，大量的对局中只要我们不单方面违背纳什均衡，那么最终是可以取胜的。

CFR+算法同样通过反复迭代逼近纳什均衡。近似程度通过其可用性衡量：与最坏情况下对手所采取的策略时相比，收益值小了多少。计算策略可用性需要计算最坏情况的收益值，通常需要遍历整个博弈树，因此长期以来，这对于 HULHE 这样复杂度很高的游戏来说都是不现实的。近期研究表明，可通过利用博弈游戏的非完备信息结构和收益规则来加速计算。这就是衡量纳什均衡策略值的近似程度的方法，已在小型博弈游戏中进行了验证，而且针对 HULHE 中简单策略的可用性进行了独立计算，改进的 CFR+也证明了游戏中庄家具有优势这一常识。

2.4.6. CFR 新变体：Lazy-CFR

虚拟遗憾最小化（CFR）是解决具有非完备信息的两人零和博弈的最有效算法，并且在实践中达到了预期效果。然而，由于这种算法过于依赖经验，对遗憾值的界限要求不严格，因此我们对 CFR 的性能还没有完全了解。而且，CFR 必须在每个回合中遍历整个游戏树，这在大型游戏中非常耗时。[44]中提出了一种新颖的技术，即延迟更新（Lazy-update），可以避免遍历每一次迭代的整个游戏树。对于延迟更新的 CFR，产生的遗憾值界限更为严格。将这种新型的 CFR 变体称为 Lazy-CFR。延迟更新是一种思想，指更新步数是动态的，根据我们所需将所有的子步数划分在不同的片段中，从而达到对某些值进行约束的目的。与在普通 CFR 中遍历 $O(|I|)$ 信息集相比，Lazy-CFR 只需要每回合遍历 $O(|\sqrt{I}|)$ 信息集，同时使遗憾约束几乎相同，I 所有信息集，Lazy-CFR 的性能明显优于 CFR。

2.5. 虚拟博弈

虚拟博弈（Fictitious Play）是一种流行的博弈理论学习模型[45]，在此模型中，玩家通过采取对对手平均策略的最佳响应进行博弈。训练出的智能体拥有两个策略集，一个是针对对手的最优策略集，另一个是自己的历史平均策略集[42]。在每一轮博弈的开始，每个智能体根据对手的历史平均策略集，找到一个最优的针对策略，然后根据自己的历史平均策略和本轮最优策略更新自己的历史平均策略。在一些两人零和博弈中，虚拟对局可以收敛到纳什均衡。

2.5.1. 扩展式博弈转换正则博弈

然而，由于做出最佳响应策略需要遍历整个博弈树，并且传统的虚拟博弈

都是用来解决正则博弈很少用于扩展式博弈，因此虚拟博弈直到最近才慢慢被应用到德州扑克的问题上。通过生成弱化虚拟对局放宽对最佳响应策略的求解精度，及产生近似的最佳响应策略。同时引入 Kuhn 定理在正则博弈策略和扩展式博弈策略之间找到一个等价的转换关系，可以将扩展式博弈模型转换为正则博弈。虚拟博弈和虚拟遗憾最小化本质上都是一种强化学习算法，只是虚拟遗憾最小化基于多人对局因此对对手策略不可见，而虚拟遗憾是基于自我对局，因此可以获取历史策略从而进行针对。此处的正则博弈策略是混合策略，扩展式博弈策略是行为策略，虚拟博弈的目的是将最优纯策略加到平均策略上，将扩展式博弈转换为正则博弈的方法是找到混合策略在行为策略上的等价表示。具体转换公式如下：

设 π 和 β 为两种行为策略，Π 和 B 为与 π 和 β 等效的两种混合策略，同时有 $\lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$ 且 $\lambda_1 + \lambda_2 = 1$，则对于任意信息节点 $u \in \mathcal{U}$，有

$$\mu(u) = \pi(u) + \frac{\lambda_2 x_\beta(\sigma_u)}{\lambda_1 x_\pi(\sigma_u) + \lambda_2 x_\beta(\sigma_u)}\big(\beta(u) - \pi(u)\big) \tag{2.8}$$

在 u 处定义行为策略 μ 是混合策略 $M = \lambda_1 \Pi + \lambda_2 B$ 的等价表示。

2.5.2. 虚拟博弈的收敛特性猜想

近几年随着神经网络大热，逐渐被引入博弈论的领域，基于神经网络的虚拟自我对局被提出，使用神经网络来拟合最优策略以及平均策略，在算法收敛性上有了进一步的提升，目标是寻找纳什均衡解，与一般神经网络问题求最优解并无太大差别，在迭代次数相同的情况下，CFR 算法收敛性往往不如虚拟博弈，参考深度学习的概念，CFR 就像随机梯度下降收敛，虚拟博弈就像动量法或者 Adam 优化那样引入某些指数加权移动平均的思想定向收敛所以比较快。参考以上策略

转换公式可以证明这一观点，实际上由于虚拟博弈的目的是将最优纯策略加到平均策略上，然后更新自己的历史平均策略，所以下一次的平均策略可以看做上一次的最优反应策略与历史平均策略的线性组合，Π, B分别为平均策略和最优纯策略，a 为常数，则

$$\Pi_{t+1} = (1 - a_{t+1})\Pi_t + a_{t+1}B_t \tag{2.9}$$

自然地转换到了指数移动加权平均的模型上，同时如果博弈树信息集规模过大则可能导致模型变复杂所以容易陷入局部最优。

2.6. 状态空间抽象方法

前文提到，2 人限注德州扑克游戏游戏状态数量多达10^{17}。应用 CFR 算法难免会有些乏力，因此我们需要适当采取一些抽象方法减少游戏状态的数量。抽象方法是基于我们已知的规则并结合我们的常规经验对游戏的一种简化，目的是为了减少游戏状态空间的复杂度，往往使用一些方法将大致相同的节点归为一类，或者减少一些影响不大的游戏状态[8]，可以将不同的游戏状态近似为一个抽象的信息集，或者对游戏的规则进行限制。抽象分为两个方面，分别是行为抽象（Action Abstraction）和卡牌抽象（Card Abstraction）。

本文采用了两种抽象方法，分别是基于指定可选动作的和行为抽象和基于改进的 9-bucketing 策略。由于行为抽象对信息集有着极大的影响，早期的论文都做了对每一轮限制加注次数的处理，极大地减少了空间和计算的复杂度。

2.6.1. 行为抽象

行为抽象是可以不考虑所有可能的下注，即假设从 100 到 20000 的任意整数，只考虑以 100 为增量的下注，即 100，200，300....，这样抽象可以大大减少

决策树的宽度，本课题使用的行为抽象是将所有下注筹码均设为 1，同时对玩家的下注回合加以限制，而且下 100 和 101 区别也不大。

每人每轮最多允许加注 4 次，按照德州扑克的一般规则，在未达到最大加注次数之前，如果超过一轮所有选手均为选择加注，则也可以判断本轮提前结束，因此规定，如果两人连续选择不加注，则判断本轮结束。而早期的相关研究表明，四轮加注的实际表现比三轮加注要好很多，虽然信息集呈指数级增长，但更加具有实用性。按照本文的处理方法，既不违反了四轮加注的原则，也在一定程度上限制了信息集的增长，同时也符合实际德扑的规则。

2.6.2. 9-bucketing 卡牌抽象

卡牌抽象是指对同类型的卡牌进行聚类，像 KK 和 QQ 区别也不是很大，这里主要指的是手牌[8][39][39]。关于卡牌抽象，一直以来被广泛使用的是 9-bucketing 策略，即根据不同的手牌强度[41]，将手牌强度近似的手牌组合划分进同一个分区，同一个分区的手牌默认是相同的。由排列组合的原理可知，德州扑克中可能有 $C_{52}^{2} = 1326$ 种手牌的组合形式，如果只考虑花色是否相同和牌面值大小，一共有$13 \times 13 = 169$ 种组合。如图 2.3 所示：

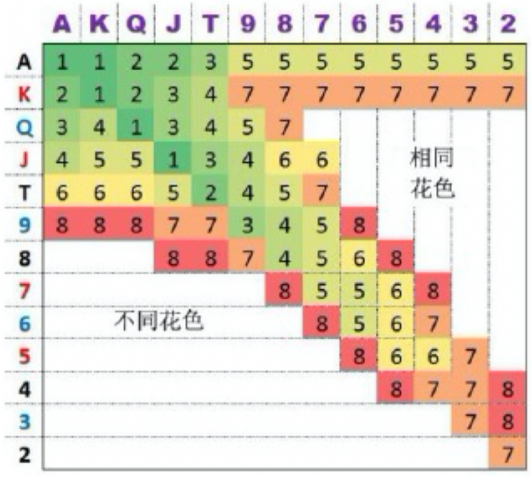

图 2.3 9-bucketing 策略示意图

在图 8-1 中的矩阵的行列代表牌面值，矩阵数组的每个元素代表对应两张手牌组合的牌力，左半区为不同花色，右半区为同一花色。根据常规经验可以知道，牌值相同的手牌更有机会构成四条、三条等，花色相同更有机会组成同花，牌值连续有机会组成顺子，单纯的点数较大也在双方都是单张时有较大的优势，常规经验表明，基于 9-bucketing 的底牌抽象技术，将手牌按照牌力划分到 9 个 bucket，编号越小手牌获胜的概率越大。

9-Bucketing 二维数组如图 8-2 所示。在这个数组的基础上对两张手牌牌力的判断，获取两张手牌所对应的分区，CFR 算法执行时在起始节点时标注手牌牌力，一个玩家的信息集的牌力相同，手牌不同的博弈树如果 9-bucketing 牌力相同，相同游戏状态的节点就可以视作同样的节点，从而降低游戏状态空间，将 1326 种手牌组合变到 9 个状态，提高了 CFR 算法的存储和计算能力。图 2.4 为 9-bucketing 二维数组示意图：

```
hand_strength = [[1, 1, 2, 2, 3, 5, 5, 5, 5, 5, 5, 5, 5],
                 [2, 1, 2, 3, 4, 7, 7, 7, 7, 7, 7, 7, 7],
                 [3, 4, 1, 3, 4, 5, 7, 9, 9, 9, 9, 9, 9],
                 [4, 5, 5, 1, 3, 4, 6, 6, 9, 9, 9, 9, 9],
                 [6, 6, 6, 5, 2, 4, 5, 7, 9, 9, 9, 9, 9],
                 [8, 8, 8, 7, 7, 3, 4, 5, 8, 9, 9, 9, 9],
                 [9, 9, 9, 8, 8, 7, 4, 5, 6, 8, 9, 9, 9],
                 [9, 9, 9, 9, 9, 9, 8, 5, 5, 6, 8, 9, 9],
                 [9, 9, 9, 9, 9, 9, 9, 8, 5, 6, 7, 9, 9],
                 [9, 9, 9, 9, 9, 9, 9, 9, 8, 6, 6, 7, 9],
                 [9, 9, 9, 9, 9, 9, 9, 9, 9, 8, 7, 7, 8],
                 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 7, 8],
                 [9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 7]]
```

图 2.4　9-bucketing 二维数组示意图

2.7. 本章小结

本章对涉及到本课题的相关理论知识做了简单介绍，先是对扩展式博弈，正则博弈与纳什均衡这些基本的博弈论知识做了介绍，接着对德州扑克的规则作了简介，使得读者对于德州扑克有所了解，然后对虚拟遗憾最小化算法的发展，算法分析，几种变体做了介绍，同时介绍了虚拟博弈在德州扑克上面的应用以及与 CFR 算法的对比，最后对抽象方法进行介绍。通过本章基础知识的理解，就能对下面的实验设计思路有更清晰的认识。

3. 实验验证

3.1. 程序设计

在未加入 CFR 策略时用玩家-玩家对战测试程序，无误后再引入 CFR 算法计算游戏策略并训练高智能体。

3.1.1. 两人有限注德州扑克应用程序

本课题采用 Python 语言以及 PyCharm 编辑器。由于 Python 的面向对象特性因此特别适合编写博弈游戏的流程。在确定整体框架后开始编写。首先确定类，游戏平台，玩家，牌桌，扑克牌，荷官，动作，策略，牌级（摊牌后手牌与公共牌的最大牌力组合），牌力（两张手牌）等。

⑴ 游戏(Game)类：在未加入 CFR 策略时用玩家-玩家对战测试程序，无误后再引入 CFR 算法计算游戏策略并训练高智能体。首先创建一局游戏，玩家个数默认为 2，给每人分配一定的筹码，并分别下大小盲注；创建荷官和牌桌并生成 52 张扑克牌，随机洗牌后，荷官给两名玩家发两张手牌并出示三张公共牌，根据玩家不同的对局状态（翻牌前，翻牌，转牌，河牌）预先设置好可选择的动作集合（看牌，下注，跟注，加注，弃牌），玩家在不同阶段选择。如果有一人选择弃牌，则另一名玩家获胜。如果没有人弃牌，则河牌后比较两人牌力大小，牌力大者获胜。

⑵ 玩家(Player)类：包含编号(id)，筹码(stack)，手牌(hands)，动作集合(actions)，是否为人类玩家(is_human)，大小盲注(blind_type)等元素，编号和筹码初始化时设定，手牌，动作采取，盲注类型以及筹码盈亏都由内部函数设定，同时还自带显示信息，动作采取等函数。

⑶ 扑克牌(Card)类：包含数字(number)和花色(suit)两个属性，并带有一些检查的函数，以及快速判断与另一张牌是否为同一花色或同一数字。

⑷ 动作(Action)类：只有一个序号(index)属性，根据序号确定是看牌(check)，下注(bet)，跟注(call)，加注(raise)，弃牌(fold)中的哪一个动作。由于玩家在一轮游戏中可采取的动作与上一个玩家的动作有很大关系，所以定义三个策略集合，如上一个玩家下注，那下一个玩家只能采取跟注，加注或弃牌；如果上一个玩家看牌，拿下一个玩家只能选择看牌，下注或弃牌等等。根据玩家的历史动作集合以及游戏状态：翻牌前(pre-flag)，翻牌(flag)，转牌(turn)，河牌(river)，便可确定唯一的可采取动作集合供玩家选择，这个过程在玩家类的动作采取函数中完成。

⑸ 牌桌(Deck)类：包含 52 张扑克牌，在游戏开始由荷官创建，具有洗牌，发牌等函数。

⑹ 荷官(Dealer)类：包含牌桌，钱罐(pot)，公共牌等属性，对牌桌进行操作，在不同游戏状态执行不同的函数。

⑺ 牌级(Rank)类：摊牌后，两张玩家手牌与五张公共牌组合最大牌级。

以下是本课题编写的德州扑克应用平台一局人人对局的过程，如图 3.1，3.2，3.3 所示：

图 3.1 翻牌前与翻牌

图 3.2 转牌

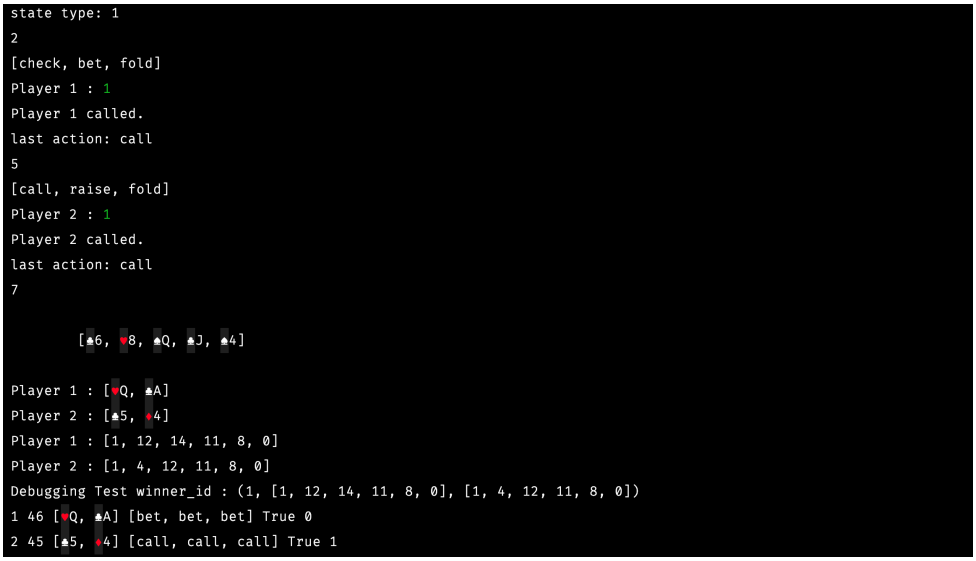

图 3.3 河牌

3.1.2. 德州扑克 CFR 程序设计

两人有限注德州扑克是一种典型的扩展式博弈，并且由于手牌，公共牌，可选取动作策略，以及诈唬等策略的存在，使得该游戏的信息集数量异常庞大，即使采取一些约束条件仍然很难降低其复杂度；设计 CFR 的思路就是基于扩展式博弈的博弈树，根据所选取的策略通过迭代构建新的节点，并且在每一个节点存储其包含的必要信息，到达叶子节点后就不断回溯至上一级节点，在遍历完一个节点的全部子节点后，更新当前节点的收益值和遗憾值以及历史策略等信息并存储，再回溯至上一级节点。本质上是一种深度优先遍历，完整遍历一次博弈树就完成了一次训练，在每次训练后更新全部信息集的遗憾值等信息，完成多次的训练后就得到一个具有一定智能的机器博弈体。以下是 CFR 的设计步骤：

⑴ 创建游戏以及两个玩家，各自获得两张手牌并且不知道对方的手牌，设置大小盲注，记录每个玩家下注的额度以及历史动作序列。每次函数的调用就相当于创建一个节点，选择一个当前的可执行动作加入历史动作序列，更新双方玩家的下注额，传入到下一个节点。

⑵ 判断当前回合数以及并执行是否更新公共牌，该处的执行函数只需要在迭代中判断一次，获得底牌即可，不用多次执行。

⑶ 判断当前节点是否为叶子节点，若是，则返回双方的收益值至上一级节点，否则执行第四步。

⑷ 判断当前回合是否结束，若是，则更新历史动作序列，进行下一轮的下注，否则继续当前回合。

⑸ 当前回合根据采取的策略不同建立新的节点并返回第二步，若所有子节点均遍历完，执行第六步。

⑹ 遍历完当前节点的所有子节点后，将不同动作返回的收益值存储在当前节点。

⑺ 为防止被对手建模，根据轮盘赌转法选择策略，计算当前节点的平均收益值以及累积遗憾值。

⑻ 根据遗憾值匹配更新策略，即采取每一个动作的概率，并和本次遗憾值一块存储，然后将当前节点的两名玩家的收益值返回值上一级节点，执行第五步。

德州扑克的 CFR 训练流程图如下：

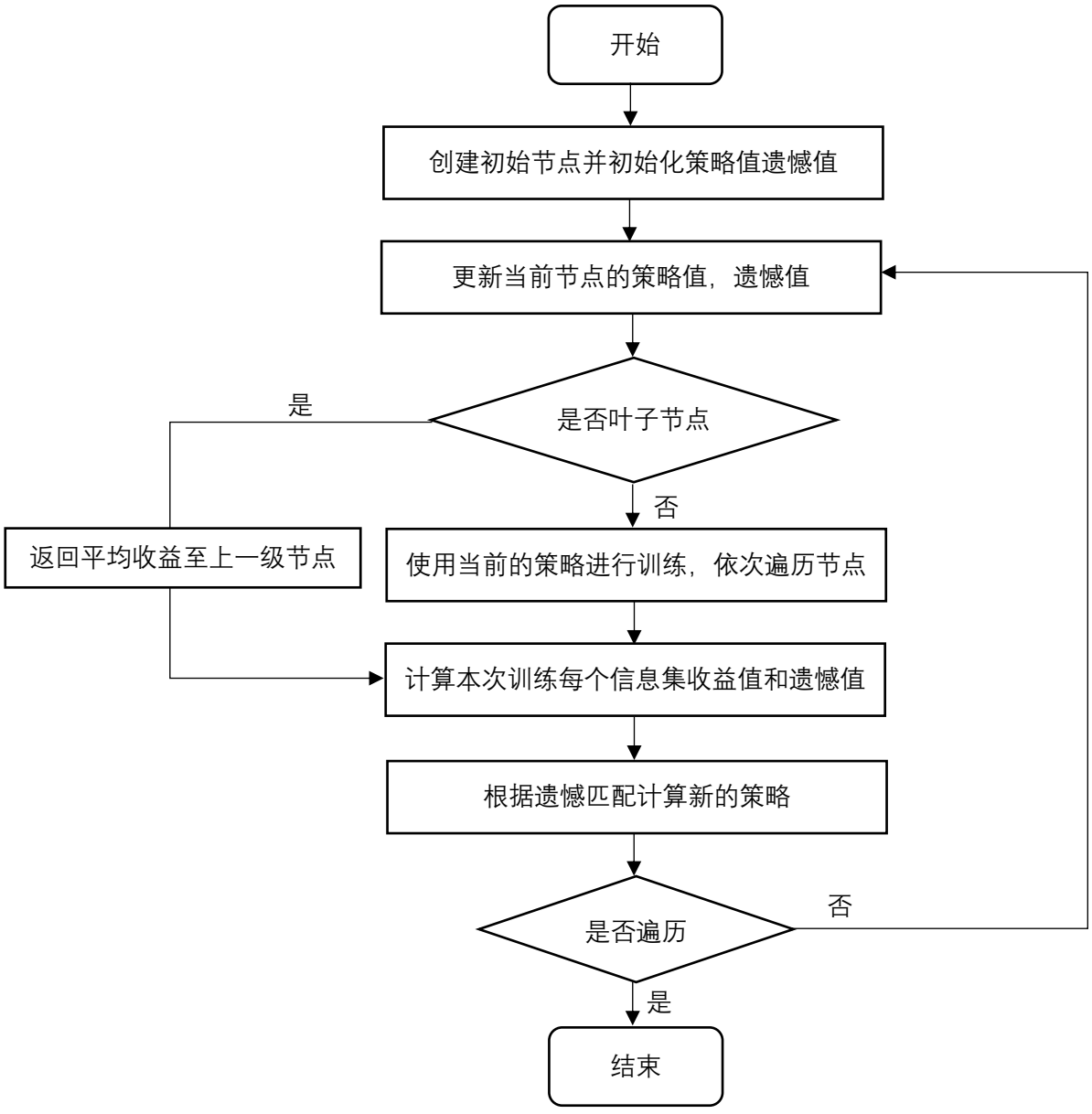

图 3.4 CFR 遍历流程图

3.2. 改进的 9-bucketing 抽象方法

在本课题的研究中发现，传统的 9-bucketing 策略，侧重关注两张手牌，而忽视了公共牌在实际对局中的作用。实际比赛中，只有发牌前这一阶段，由于玩家已知信息只有自己两张手牌，所以两张手牌组成的牌力大小发挥着较大作用，但随着游戏进行，在翻牌，转牌，河牌阶段，公共牌都会出现较大变化，甚至改变场上一张牌也能改变决策。本课题提出的改进抽象方法，是为了是将个人决策与除了手牌和对手动作外，底牌信息这些公共信息相关联，从而做出最优的决策。

3.2.1. 改进 bucketing 策略牌力分区

即使两张手牌牌力较小，但如果公共牌牌面值也很小，这样就有机会和底牌组成对子甚至顺子这些组合，特别是在已知能组成较大牌级的情况下，贸然弃牌则是一种非常不明智的选择，反而对手的大牌在这种情况下毫无用武之地，根据概率论，公共牌的牌面值也是均值分布的。因此己方在这一博弈层面中采取了一种近乎反诈唬的策略，从心理层面战胜对手，对手如果误以为己方在诈唬于是不断加注，这样己方就能获得更大的收益。这种场景类似于当下一种比较流行的千层饼理论：当对面还在第二层的时候，以为我们在第一层，而实际上我们已经在第五层了。

这种改进的 bucketing 策略，与原方法类似，也是根据不同的牌面组合大小将当前最大的组合置入一个 bucket 里。不同的是，在此处需要结合公共牌。为减少计算量，本课题也采用设置 9 个 bucket，由于缺少大数据的验证，本课题采用平均分配的方法设置 bucket，除了在翻牌前采用之前的 bucket 分区，剩余

阶段皆采用五张牌的组合。

由于 5 张牌面共有 $C_{52}^{5} = 2{,}598{,}960$ 种组合，这种方法需要对五张牌的牌面值作分配，因此需要设置好每个牌级可能出现的所有情况后再实现近似均分，每个 bucketing 大约有 288,773 种情况。根据计算所得每种牌级的组合为：

同花顺：$10 \times 4 = 40$ 种；

四条：$13 \times 12 \times 4 = 624$ 种；

葫芦：$C_4^3 \times 13 \times C_4^2 \times 12 = 3{,}744$ 种；

同花：$C_{13}^5 \times 4 - 40 = 5{,}108$ 种；

顺子：$4^5 \times 10 - 40 = 10{,}200$ 种；

三条：$C_4^3 \times 13 \times C_{48}^2 - 3744 = 54{,}912$ 种；

两对：$C_4^2 \times 13 \times C_4^2 \times 12 \times 44 \div 2 = 123{,}552$ 种；

一对：$C_4^2 \times 13 \times 48 \times 44 \times 40 \div A_3^2 = 1{,}098{,}240$ 种；

单张：$2{,}598{,}960 - 1{,}098{,}240 - 123{,}552 - 54{,}912 - 10{,}200 - 5{,}108 - 3{,}744 - 624 - 40 = 1{,}302{,}540$ 种.

可以看出，两对及以上的牌级一共只有 198,180 种组合，于是改进的 9-bucketing 策略可如下分配。

第一 bucket：两对及以上的牌级组合；

第二 bucket：对牌为 A, K, Q 一对；

第三 bucket：对牌为 J, 10, 9 的一对；

第四 bucket：对牌为 8, 7, 6 的一对；

第五 bucket：对牌为 5, 4, 3, 2 的一对；

第六 bucket：最大点数为 A, K 的单张；

第七 bucket：最大点数为 Q, J 的单张；

第八 bucket：最大点数为 10, 9 的单张；

第九 bucket：最大点数为 8, 7 的单张。

改进的 9-bucketing 策略，每个节点包含的不再是当前两张手牌的牌力，而是手牌与公共牌的最大组合，分到 9 个桶里，这样根据场上不同阶段公共牌的变化动态调整策略，也更符合实际情况；为适应这种策略改进了牌级判断方式，可以输入 5-7 张牌，返回最大组合的牌级。与前述方法区别在于，算法执行过程中，处理决策树的节点时，需要对所有的节点标记当前手牌与公共牌最大组合的牌力，而不只是对根节点进行标记，所以按照此方法在同一棵博弈树中，不同节点也会落在不同的 bucket 区间，而牌力相同的手牌决策节点在 CFR 遍历过程中可以当做完全相同的节点，用相同的策略处理，从而降低德州扑克的复杂度，同时也更加符合实际情况，实际上存储空间较之前并没有增加。

3.2.2. 基于改进 bucketing 策略的牌级判断方法

由于使用改进的 9-bucketing 策略，需要对决策树上每一个节点的手牌与公共牌的最大组合进行判断分配，因此需要能对任意输入的 5-7 张牌进行判断，如果按照一般思路，将底牌任意三张牌的组合与手牌组合进行判断，那么在河牌阶段每个节点都需要进行 $C_5^3 = 10$ 次判断，过于麻烦并不适合，因此做了如下改进。先写三个基函数，分别为判断共有 n 个点数相同的手牌，各种花色的手牌有几张，能否组成顺子并返回最大的顺子。然后在九种牌级的判断函数中调用相应的函数，运行效率将会获得极大的提升。

判断后最大组合牌级将以长度为 6 的数组返回，数组中第一个元素为牌级，如图 3.5 所示：

Rank:	$C_{52}^{5} = 2{,}598{,}960$
Royal-flush? (9, 0, 0, 0, 0)	*4*
Straight-flush? (8, highest card, 0, 0, 0, 0)	*36*
4-of-a-kind? (7, card in 4s, kicker, 0, 0, 0)	*624*
Full house? (6, card in 3s, card in 2s, 0, 0, 0)	*3744*
Flush? (5, highest card, 2nd, 3rd, 4th, 5th)	*5108*
Straight? (4, highest card, 0, 0, 0, 0)	*10200*
3-of-a-kind? (3, card in 3s, kicker, 2nd, 0, 0)	*54912*
2 pairs? (2, high pair, low pair, kicker, 0, 0)	*123552*
Pair? (1, pair, kicker, 2nd, 3rd, 0)	*1098240*
Nothing? (0, 1st, 2nd, 3rd, 4th, 5th)	*1302540*

图 3.5 各牌级手牌组合示意图

皇家同花顺，默认最高级别，牌级为 9；同花顺，牌级为 8，第二个元素为同花顺最大的牌；四条，牌级为 7，第二个元素为组成四条的牌值；葫芦，牌级为 6，第二个元素为组成三条的牌值，第三个元素为组成对子的牌值；同花，牌级为 5，剩余五个元素依次为牌值大小排列；顺子，牌级为 4，第二个元素为最大牌值；三条，牌级为 3，第二个元素为组成三条的牌值，第三四个元素为剩余两张牌值由大到小排列；两对，牌级为 2，第二个元素为牌值较大的对子的牌值，第三个元素为牌值较小的对子牌值，第四个元素为剩余的单牌牌值；一对，牌级为 1，第二个元素为对子的牌值，剩余为单张的牌值由大到小排列；单牌，牌级为零，剩余五个元素依次为牌值大小排列。

3.3. 数据抽象模型

3.3.1. 数据表示方法

德州扑克程序设计的数据表示是一项值得探究的任务，由于训练虚拟遗憾最小化算法需要多次迭代，如果程序设计不够简洁，那么训练所消耗的时间将呈现

指数级的增长，因此将程序设计的尽可能简洁是很有必要的。由于 CFR 算法本身的特点，多次的迭代并不会带来空间存储上指数级的增长，因此程序设计在一定程度上采用了用空间换时间的思路。

德州扑克共有 52 张牌，在设计程序时扑克牌类包含牌值和花色两个属性，其中牌值用 2-14 表示，A 牌牌面为 14，其余为各自实际牌面值，花色分别用数字 1-4 表示；使用 0-2 共 3 个数字表示游戏中玩家所做的动作：0-叫注(call)、1-弃牌(fold)和 2-加注(raise)。将玩家采取的动作加入历史动作序列，用数字 4 来分隔不同阶段的动作序列。如图 3.6, 3.7 所示：

```
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 0] 31192
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 0] 31193
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 0] 31194
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0, 0] 31195
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0, 2] 31196
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0] 31197
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2, 0] 31198
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2, 2] 31199
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2] 31200
[0, 2, 0, 2, 2, 2, 2, 2, 4, 0, 2, 0, 2, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0, 2] 31201
```

图 3.6 一段历史动作序列

```
in:[0, 2, 0, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0, 4, 0, 2, 0, 2, 0]
id:2 sg:7 st:1 rc:12.327272727272728 sc:0.7865429234338747 rf:0 sf:0.0 rr:3.3454545454545457
in:[0, 2, 0, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0]
id:1 sg:4 st:0 rc:15.5 sc:1.0 rf:0 sf:0.0 rr:0 sr:0
in:[0, 2, 0, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 2, 0]
id:1 sg:4 st:0 rc:0.0 sc:0.5 rf:0.0 sf:0.5 rr:0 sr:0
in:[0, 2, 0, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0, 4, 0, 2, 0, 2, 2, 0, 2, 2]
id:1 sg:4 st:0 rc:0.0 sc:0.5 rf:0.0 sf:0.5 rr:0 sr:0
```

图 3.7 包含遗憾值的信息集片段

每个节点需要存储的数据包含，当前的历史序列，玩家的 id，不同动作的遗憾值和策略值等信息。关于信息集存储文件的片段节选如图 3.8 所示：

```
in:[0, 0, 4, 0, 0, 4, 0, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0, 2] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 0, 2] id:2 rc:5.666666666666666 sc:0.5 rf:0 sf:0.0 rr:5.666666666666666 sr:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2, 2] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2, 2, 2] id:2 rc:6.0 sc:0.5 rf:0 sf:0.0 rr:6.0 sr:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0, 2] id:1 rc:1.5555555555555558 sc:0.41176470588235303 rf:0 sf:0.0 rr:2.222222222222222
sr:0.588235294117647
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 0] id:2 rc:6.7407407407407405 sc:0.9285714285714285 rf:0 sf:0.0 rr:0.5185185185185186
sr:0.07142857142857144
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 0, 0] id:1 rc:8.5 sc:1.0 rf:0 sf:0.0
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 0, 2, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 0, 2, 2] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 0, 2] id:2 rc:6.0 sc:0.5 rf:0 sf:0.0 rr:6.0 sr:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 0] id:1 rc:0.3333333333333335 sc:0.08000000000000003 rf:0 sf:0.0 rr:3.8333333333333335
sr:0.9199999999999999
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 0, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 0, 2] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 0] id:2 rc:6.0 sc:0.5 rf:0 sf:0.0 rr:6.0 sr:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 2, 0] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 2, 2] id:1 rc:0.0 sc:0.5 rf:0.0 sf:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2, 2] id:2 rc:6.333333333333334 sc:0.5 rf:0 sf:0.0 rr:6.333333333333334 sr:0.5
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2, 2] id:1 rc:2.111111111111111 sc:0.43181818181818177 rf:0 sf:0.0 rr:2.777777777777778
sr:0.5681818181818182
in:[0, 0, 4, 0, 0, 4, 0, 2, 0, 2, 2] id:2 rc:2.1481481481481484 sc:0.4280442804428044 rf:0 sf:0.0 rr:2.8703703703703702
sr:0.5719557195571955
```

图 3.8 信息集存储文件片段

实际程序设计中还作了如下处理：

⑴ 9-bucketing 策略，手牌与公共牌最大组合牌力落入哪一区间。

⑵ 每阶段最多四轮加注，叫注和跟注视为同一个动作，每阶段第一个玩家第一个动作只能是跟注或弃牌，如果两个玩家依次选择跟注就进入下一阶段。

⑶ 根据当前游戏阶段改变公共牌，每轮训练只进行一次，公共牌一旦确定不再变动。

⑷ 存储每个节点的遗憾值，收益值，再根据遗憾匹配结合轮盘赌转法更新策略，防止被对手建模。

3.3.2. 数据存储模型

预测试阶段将所有信息集存储在同一个文件，内存消耗较少（22.4MB），但是在迭代更新时需要消耗较多的时间遍历找到对应的节点（第一次生成需要 16s，之后每次迭代一遍大约 35s，此处不能使用 for 循环遍历，正确做法是获取所有节点名，然后直接调用当前节点对应的节点信息），时间消耗较长，因此只好改变存储思路，改为 pickel 的存储方式，并且将涉及到字符的信息全部用数字代替，

然后将一个节点的信息用列表表示，每个信息集单独建立一个文件，最后内存消耗 622MB，但迭代运行时间只有约 20s。

这是一种空间换时间的思路，虽然磁盘的读取速度比内存要慢很多，但如果只是在寻找系统硬盘里的目录节点，使用指针和查表的方式可以很快找到文件名，所以整体上看这种方式花费时间比直接在内存里读取花费的时间反而要少。综合考虑，我们的时间远远比空间要宝贵，而 CFR 算法本身并不会是存储空间呈指数级增加，所以在此采取了节省时间的做法。

3.4. 程序测试

3.4.1. CPU 服务器测试

受疫情影响，今年不能到实验室进行测试，而本课题需要大规模计算，因此决定网上租用云服务器完成运算。这部分对计算机 CPU 型号选择作以比较分析。

一般云服务器配置都是以 CPU 和内存来说明的。不过也要区分不同机型，即使配置相同也有不同性能。常见实例分类有通用型，计算型，内存型，高主频型等，同时这几款也是日常大部分业务的常用机型，可以说一般的项目使用这几款机型都够用了。无论是计算型、通用型还是内存型网络增强云服务器，都是使用 2.5GHz 的处理器，具有超高网络 PPS 收发包能力以及网络延迟低的特性。计算型、通用型和内存型之间的区别主要在于 CPU 和内存之间的配比，顾名思义计算型更倾向于 CPU，通用型是 CPU 和内存之间的平衡，内存型更侧重于内存，计算型 CPU 和内存比为 1:2，通用型为 1:4，内存型为 1:8，三者的适用场景也不同。计算型 CPU 一般适用于前端服务器，需要大型计算的多人在线游戏，大数据分析、并行计算、视频编码等应用[43]；通用型 CPU 一般适用于中小型数据库

系统的缓存、分析和计算[43]；内存型 CPU 适用于高性能数据库，数据分析与挖掘、分布式内存缓存以及其他企业大内存需求应用等[43]。综合比较，本课题更适合计算型 CPU。最终选择 8 核，16GB，2.6GHz 主频，型号为 Intel Xeon Platinum 6271 的计算型 C4 服务器，使用后付费的方式，实例每小时收费 4.5 元，使用多进程运行。

创建服务器实例成功后，还需要建立一个弹性公网 IP 地址和服务器的私网 IP，然后就可以在本地实现远程调试。本课题选择使用专业版的 PyCharm 进行调试，PyCharm 的 Professional 版本具有远程调试功能，在配置界面输入服务器的公网 IP 以及端口后等信息，PyCharm 也带有同步本地代码到服务器的功能，连接成功后，还需要对服务器的环境进行配置，本课题需要的环境并不复杂，也没有大量的数据包，因此可以快速传输文件。在服务器远程连接后其余操作与本地基本相同，设定好训练次数，训练完成后将策略文件同步到本地即可。由于 Python 的缘故，程序运行效率相对较低，最后生成 5.75GB 的策略文件，然后在本地进行测试即可。

3.4.2. 纳什均衡收敛性度量

在讨论最终结果之前，需要考虑一下如何评估近似纳什均衡策略的收敛性。通常使用评估策略的可利用性（Exploitability）。可以理解为在零和游戏中，完美纳什均衡具有零收益，而-纳什均衡具有可利用性。可利用性度量是（mb/h），即每局游戏中小盲注的千分之一的收益。通常，在整个游戏中计算策略的可利用性很困难，需要借助一些已有的博弈程序或者改变策略观察收益的方法才能描述当前策略离均衡的距离，由于硬件条件的限制无法采用中提出的可利用度的方法描述收敛性，于是本文采用了一种简便的思路，按照我们虚拟遗憾最小化的目的，

直接观察子博弈树的平均遗憾值变化率即可判断是否已经实现遗憾最小化即是否达到纳什均衡策略。将平均遗憾值变化率代替可利用度描述本课题德州扑克 CFR 算法的纳什均衡收敛性的示意图如下：

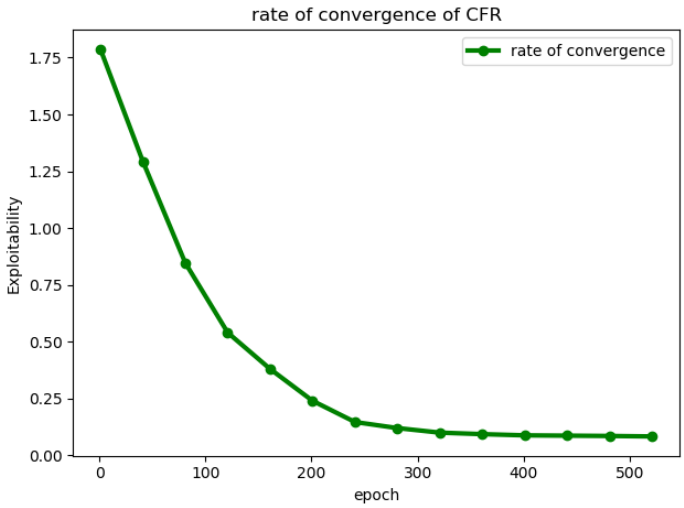

图 3.9　CFR 算法收敛性示意图

3.4.3.　纯策略测试

一定时间内将虚拟遗憾值和平均策略值写回磁盘，再从磁盘读取，4 个线程同时运行。遍历一次博弈树文件大小为 622.9MB，消耗时间为 13.2 s。结合翻牌前和其他阶段不同的 9-bucketing 抽象方法，所有策略信息文件共需要 5.75GB 存储空间，迭代 500 次大约需要 2 小时，对迭代 500 次的策略使用纯策略的算法进行测试。

训练了一个每轮最多三次加注的程序，在与纯策略（全部选择叫注，无加注弃牌）的对局中，表现尚可，策略选取较保守，弃牌次数略多，不弃牌的情况一般都能取胜；而与纯策略为加注的对局中，可能存在被诈唬的情况增加，因此选择的策略更加保守，表现也不如纯策略为加注的表现。纯策略为弃牌时，基本不会输，所以并无参考意义。

使用改进的 9-bucketing 策略，其余训练方式不变，在与纯策略的的对局中，弃牌次数较之前均有所减少。不管是纯策略为叫注还是加注的对局中，性能均有所提升。

本文的程序设计中起始双方筹码均为 50，小盲注下 1，大盲注下 2。以后叫注筹码为 1，加注筹码为 2，以此统计净胜筹码方式表示双方输赢情况，如表 3.1，3.2 所示。

	胜率	净胜筹码
叫注（call）	61.4%	1.13
加注（raise）	56.2%	0.75

表 3.1 传统 9-bucketing 策略与纯策略的对局表现

	胜率	净胜筹码
叫注（call）	67.5%	1.42
加注（raise）	63.7%	1.28

表 3.2 改进的 9-bucketing 策略与纯策略的对局表现

如图 3.10 是纯策略为叫注时改进的 bucketing 策略与原策略在 200 局游戏中的净胜筹码。

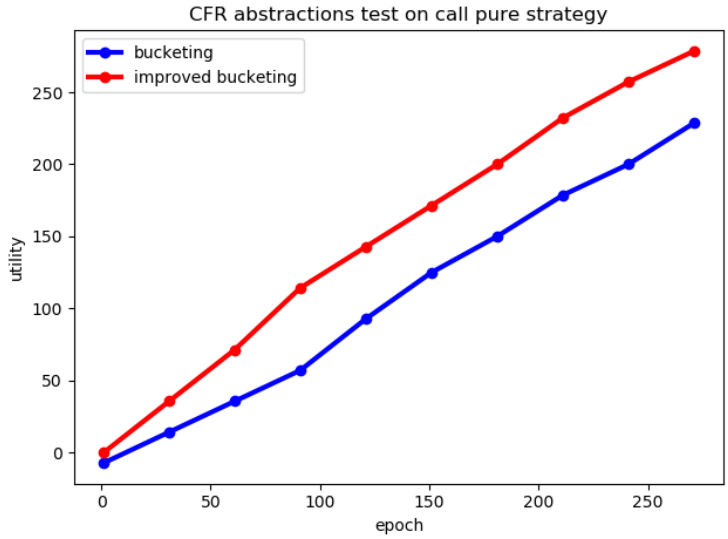

图 3.10　叫注纯策略时两种抽象方法在 200 局游戏中的净胜筹码

3.4.4.　防对手建模策略的损失

机器博弈中，防止被对手建是很重要的一点，尤其对于一些形式比较简单的正则博弈如剪刀石头布，如果为了寻找纳什均衡解而采用遗憾最小化算法，很容易被对手建模，虽然说德州扑克的游戏状态数量庞大，纳什均衡解也是近似值，但是为了一定程度规避掉这些风险，还是需要采取一定措施，由于本课题并不的实现场景并不需要太高智能的防建模策略，因此采用了最常见的轮盘赌转法，计算策略值后并不是直接选取值最大的动作，而是按照策略值大小所占的权重分不到[0,1]之间，然后每次产生[0,1]之间均匀分布的随机数，落到哪个动作区间就选取哪个动作，因此宏观角度在训练足够的前提下还是有着很大的概率选择最优策略的，而对于非最优的策略也会有较高机率规避，当然也要承担因为没有选择最优策略而带来的损失。所以对于本次测试全部为纯策略，去掉防止对手建模的轮盘赌转法后更多带来的是选择最优策略的收益。如表 3.3 所示。

	胜率	净胜筹码
叫注（call）	72.4%	1.73
加注（raise）	66.3%	1.45

表 3.3 不使用轮盘赌转的改进 9-bucketing 策略与纯策略的对局表现

3.5. 对比分析

改进的 9-bucketing 策略，由于更加关注游戏对局的实时数据，因此效率也略优于传统的策略。并且由于改进的 9-bucketing 策略分区近似均分，也就意味着任意组合的牌型落在每个区的概率是相等的，在传统 9-bucketing 策略里位于第一 bucketing 的手牌组合，在改进后至少也能落在第六 bucketing 及以上的分区里，对于策略的选取同样具有参考性，并且将原本牌力较小但与公共牌组合后能称为较大牌型的手牌力度提升不少，一定程度上可避免盲目弃牌带来的损失，主观表现上，这种改进的 9-bucketing 使得策略选取更加大胆。

规避对手建模的算法实验也基本符合预期。虽然训练出的均衡策略并不是针对纯策略的最优策略，但是在与纯策略的测试中也有不错的表现，去掉轮盘赌转法后，几乎可以保证每一次都是采取最优策略，因此性能有着较明显的提升。

3.6. 改进与不足

一些数据显示，输的局中很多都是因为弃牌而输的，尤其在纯策略为加注时，这种情况更加明显，因此，需要一些策略来改变这种保守的决策，不要被对手的诈唬策略取巧。改进的 9-bucketing 策略一定程度上避免了轻易弃牌的缺点，从本质上讲，是因为与底牌结合避免武断的弃掉牌面价值虽然不大但是与公共牌组合后潜在价值较高的手牌，所以胜率较之前有所提升。但由于目前并未在大数据

上测试，当前的分区方式未必是最优，由于训练中也有纯策略的情况，个人感觉也可能是没有达到最佳的训练次数。而这种改进的 9-bucketing 策略能否真的超越之前的方法，也还需要进一步大量实验数据的支持和验证。

3.7. 本章小结

本章主要是针对本课题的理论知识进行实验设计。先是介绍了针对本课题设计的一款两人有限注德州扑克的应用平台，作为实践 CFR 算法的基础，然后将 CFR 引入德州扑克的生成策略中，通过反复迭代训练出具有一定智能的机器博弈体。同时介绍了改进的卡牌抽象策略以及程序设计的数据模型。接着讲述了自己在远程服务器的测试过程，并给出测试结果以及对比分析。

4. 总结与展望

选择德州扑克的课题，很大程度上是觉得有趣好玩。自己一直比较喜欢棋牌类的游戏，对于斗地主，麻将，象棋，围棋，军棋等均有浓厚的兴趣，虽然此前对于德州扑克一无所知，但是出于对棋牌类游戏的热爱还是选择了这个题目，而前几年 AlphaGo 在围棋上大显神威，令我对棋牌游戏与人工智能的结合心向往之。通过这次毕设，我觉得收获颇丰，不但学会了德州扑克的玩法技巧，面向对象程序设计能力也有所提升，同时对机器博弈这一概念也颇有感触。

在基于虚拟遗憾最小化的两人有限注德州扑克这一题目中，我的工作主要分为对机器博弈，德州扑克，纳什均衡等背景理论知识的了解，对虚拟遗憾最小化算法的理解学习，编写德州扑克的应用程序，CFR 算法以及抽象方法，最后利用远程服务器进行测试。各部分总结如下：

⑴ 虚拟遗憾最小化是一种高效的遍历算法，经常用来处理大规模扩展式博弈，通过遍历整个决策树，计算并存储每个节点采取不同策略的遗憾值，在多次迭代中更新每个节点的遗憾值和策略值，进行下一轮的策略选择，在德州扑克中根据不同历史动作的游戏状态结合手牌情况进行决策。使用 CFR 算法处理两人有限注德州扑克，首先需要编写两人德州扑克的应用程序，能实现两人对局后，再引入虚拟遗憾最小化算法不断调整学习最优策略，最终实现一个具有较高智能水平的德州扑克博弈系统。

⑵ 德州扑克的游戏状态数量较大，需要引入合适的抽象方法降低时间空间复杂度，最常见的有行为抽象和卡牌抽象，行为抽象一般对加注次数和可选动作加以限制，卡牌抽象最长使用的是 9-bucketing 策略，对两张手牌牌值做牌力

判断并分区，落在同一分区的手牌组合可采取相同的策略，一定程度上能降低空间复杂度。本文提出改进的 9-bucketing 策略，判断的不再是每个节点当前两张手牌的牌力，而是手牌与公共牌的最大组合，将所有情况均分区到 9 个桶里，根据场上不同阶段公共牌的变化动态调整策略。这种方法相对更符合实际情况，并且在小规模纯策略的测试中也表明，这种改进的 9-bucketing 策略在不引起时间空间复杂度增加的前提下，性能有所改善。

⑶ 最后由于受疫情影响，无法到达学校使用实验室服务器进行测试，而该题目对 CPU 计算性能要求较高，最终在网上租用云服务器测试，完成了主要内容，并对改进的 9-bucketing 策略进行验证分析，得到了预期结论。

现如今关于机器博弈和德州扑克算法的研究蓬勃发展，不断开启新的篇章。虽然非完备信息机器博弈已经取得了丰硕的成果，但很多技术如 CFR 算法的时间和存储开销问题一直未能很好解决，德州扑克仍有很大研究提升空间。

本课题已经使用了极为简化的行为抽象方法，但是仍然受到计算量的制约，可见硬件仍然是当前大规模非完备信息的扩展式博弈发展的最大限制。另一方面，个人认为，一切的博弈行为均是人类社会映射在数学空间的抽象模型，脱离了实际情况，单纯的利用统计方法研究模型内部的数学关系，是达不到最佳的效果的。因此我认为未来德州扑克算法发展应该向实际情况靠拢。真正的游戏局势往往瞬息万变，无论是对场上对手的细节动作，面部表情等有用的信息的捕捉，还是对当前牌型以及潜在价值的判断，人类选手也都比当前最优的德州扑克智能博弈机器也要可靠的多，如果机器博弈也能像人类一样学会利用这些有价值的场上信息，加上本就远远强于人类选手的计算能力，机器博弈的发展将迎来又一次突破。

致谢

四载岁月惯优游，红衰翠减物华休，且将往事忆从头。

吟情渺渺心悠悠，欲挥剑气荡清秋，须倾杯盏一醉酬。

弹指间，大学已成为过去，欢笑中夹杂着叹息，收获中混搭着遗憾；事情的开头往往猜不到结局，疫情当下，举世患难，没想到竟然以这样一种特殊的方式告别度过四年的学校。犹记刚入校时，也曾一腔热血，万丈豪情，后来屡屡碰壁，几度迷惘，值得庆幸的是，总算在行将告别时看见了柳暗花明，去开辟另一段道路。只要有希望，就值得为之付出一切。

感谢在一起相处的室友，自卓 1601 班的同学，启明学院智能车队的同学；大家都彼此留下深刻的印象，三位室友各有特色，其中有着班级同学的学习楷模，成绩天花板，引领着宿舍的学习之风，大家性格都很活泼，从早期日常报团取暖，到后来彼此心照不宣；虽然从此各奔天涯，但相信大学四年成为各自记忆中最宝贵的片段。班里的同学也都很热心，虽然刚入学时有些不适应，但后来也认识了不少好友，这也许会是以后人生中最宝贵的一笔财富。自己大学很长一段时间都是在启明学院的智能车队度过的，在这里更是结交了不少身怀绝技的朋友，而关于比赛的经历对自己来说更像是一种磨练吧。

同样更要感谢的是，那些学识渊博，敬业和蔼的老师们，自己大学期间受到过很多老师的帮助。首先感谢的是毕设指导老师罗云峰老师，感谢罗老师在毕设期间给予的谆谆指导；也要感谢智能车队的指导老师何顶新老师，感谢何老师在车队给予我们的帮助和人生指导，车队才能取得那么多好成绩；最后也要感谢大四新加入实验室的导师袁烨老师，在我急需项目经历时给让我进入实验室；同时感谢教师班主任张征老师，给予我们班无微不至的关怀和帮助 ····· 感谢以上

老师的同时，也要感谢直接指导我毕设的居奇学长，实验室的程骋学姐，以及车队的马志朋队长和黄宗恒学长等，因为他们的指导，我在人生前进的路上少走了许多弯路。

虽然大学即将结束，但追逐梦想的脚步永远不会停止；希望接下来的人生会有更多的精彩，现实值得我去拥有，未来值得我去期待。冷月遗旧梦，朝霞吟春风；为偿多劫愿，浩荡赴前程。

参考文献

[1] Shannon C E. Mag P. Series 7[J]. 1950(41): 256-275.

[2] J. Schaeffer, R. Lake, P. Lu, M. Bryant, AI Magazine 17, 21 (1996).

[3] M. Campbell, A. J. Hoane Jr., F. Hsu, Artificial Intelligence 134, 57 (2002).

[4] D. Ferrucci, IBM Journal of Research and Development 56, 1:1 (2012).

[5] J. Bronowski, The ascent of man, Documentary (1973). Episode 13.

[6] 安波.人工智能与博弈论——从阿尔法围棋谈起[J].中国发展观察,2016(06):13+17.

[7] Russell S J, Norvig P, Canny J F, et al. Artificial Intelligence: a Modern Approach[M]. Vol. 2. Cambridge, MA, USA: Prentice hall Englewood Cliffs,1995.

[8] Zinkevich, Martin & Johanson, Michael & Bowling, Michael & Piccione, Carmelo. (2007). Regret Minimization in Games with Incomplete Information.. 2008.

[9] Science; Studies from University of Alberta Further Understanding of General Science (Heads-up limit hold'em poker is solved)[J]. Science Letter,2015.

[10] C. Babbage, Passages from the Life of a Philosopher (Longman, Green, Longman, Roberts, and Green, London, 1864). Chapter 34.

[11] M. Zinkevich, M. Littman, Journal of the International Computer Games Association 29, 166 (2006). News item.

[12] Billings D. Computer poker[M]. Alberta, Canada: Department of Computing Science, University of Alberta, 1995.

[13] Berliner H J. Backgammon Computer Program Beats World Champion[J]. Artificial Intelligence. 1980, 14(80): 205-220.

[14] Papp D R. Dealing with Imperfect Information in Poker[M]. University of Alberta, 1999, 12(24): 43 -54.

[15] Billings D et al. Approximating Game-theoretic Optimal Strategies for Full-scale Poker[C]//In Proceeding of IJCAI, 2003: 661-668.

[16] Bowling M, Risk N A, Bard N, Billings D, Burch N, Hawkin J, Holte R et al. A demonstration of the Polaris poker system[C]//In Proceeding of International Conference on Autonomous Agents and MultiAgent Systems 2009, 2009:1391-1392.

[17] V. L. Allis, Searching for solutions in games and artificial intelligence, Ph.D. thesis, University of Limburg (1994).

[18] D. Koller, A. Pfeffer, Artificial Intelligence 94, 167 (1997).

[19] D. Billings, et al., Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (2003), pp. 661 – 668.

[20] 王骄，徐心和．计算机博弈：人工智能的前沿领域——全国大学生计算机博弈大赛[J]．计算机教育，2012(7):14-18.

[21] 徐心和，王骄．中国象棋计算机博弈关键技术分析[J]．小型微型计算机系统,2006, 27(6):961-969.

[22] 沈学东．基于风险模型的德州扑克博弈系统的研究[D].哈尔滨工业大学,2014.

[23] 李景棚．非完备信息博弈估值算法的研究[D].哈尔滨工业大学,2014.

[24] Jiří Čermák,Viliam Lisý,Branislav Bošanský. Automated construction of bounded-loss imperfect-recall abstractions in extensive-form games[J]. Artificial Intelligence,2020,282.

[25] Parkash Chander,Myrna Wooders. Subgame-perfect cooperation in an extensive

game[J]. Journal of Economic Theory,2020,187.

[26] 刘小山,唐晓嘉.基于囚徒困境博弈的理性、信息与合作分析[J].西南大学学报(社会科学版),2019,45(01):21-30.

[27] 李冰. 重复囚徒困境博弈中群体策略演化研究[D].大连理工大学,2018.

[28] Guilherme Carmona,Konrad Podczeck. Pure strategy Nash equilibria of large finite-player games and their relationship to non-atomic games[J]. Journal of Economic Theory,2020,187.

[29] Francesca Parise,Sergio Grammatico,Basilio Gentile,John Lygeros. Distributed convergence to Nash equilibria in network and average aggregative games[J]. Automatica,2020,117.

[30] 关毅.人工智能在多人桌德州扑克比赛中战胜世界顶尖选手[J].自然杂志,2019,41(04):241.

[31] Branislav Bošanský,Viliam Lisý,Marc Lanctot,Jiří Čermák,Mark H.M. Winands. Algorithms for computing strategies in two-player simultaneous move games[J]. Artificial Intelligence,2016,237.

[32] F. Southey, et al., Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence (2005), pp. 550 – 558.

[33] I. V. Romanovskii, Soviet Mathematics 3, 678 (1962).

[34] A. Gilpin, T. Sandholm, Journal of the ACM 54 (2007).

[35] J. Shi, M. L. Littman, Revised Papers from the Second International Conference on Computers and Games (2000), pp. 333 – 345.

[36] 代佳宁. 基于虚拟遗憾最小化算法的非完备信息机器博弈研究[D].哈尔滨工

业大学,2017.

[37] 胡裕靖,高阳,安波.不完美信息扩展式博弈中在线虚拟遗憾最小化[J].计算机研究与发展,2014,51(10):2160-2170.

[38] 滕雯娟. 基于虚拟遗憾最小化算法的德州扑克机器博弈研究[D].哈尔滨工业大学,2015.

[39] 胡开亮. 基于状态抽象和残局解算的二人非限制性德州扑克策略的研究[D].哈尔滨工业大学,2017.

[40] 吴天栋. 非完备信息机器博弈算法及对手模型的研究[D].武汉理工大学,2018.

[41] 王帅,雷跃明.一种德州扑克的牌力评估方法[J].计算机工程与科学,2017,39(07):1352-1358.

[42] 毛建博. 基于虚拟自我对局的多人非完备信息机器博弈策略研究[D].哈尔滨工业大学,2018.

[43] ESC 云服务器实例规格族官方文档

[44] Zhou, Y., Ren, T., Li, J., Yan, D., Zhu, J.\ 2018.\ Lazy-CFR: fast and near optimal regret minimization for extensive games with imperfect information.\ arXiv e-prints arXiv:1810.04433.

[45] Heinrich J, Lanctot M, Silver D. Fictitious Self-Play in Extensive-Form Games[C]//International Conference on Machine Learning. 2015:805-813.