【摘要】

本论文将讨论如何使用蒙特卡洛方法为《炉石传说：魔兽英雄传》开发人工智能，进而指出作为博弈游戏，《炉石传说》十分适合用于人工智能研究。《炉石传说》目前已包含超过700张不同的卡牌，所有的这些卡牌都能够在不同程度上改变游戏的规则，这使得要为其开发一个完备的人工智能变得十分困难（尽管本文中只考虑《炉石传说》的一小部分卡牌）。除此之外，《炉石传说》的复杂度以及其作为卡牌游戏固有的随机性也为人工智能的开发增加了难度，这其中包括了不完全信息的搜索以及对手行为的预测。

蒙特卡洛搜索一直以来便是人工智能领域的著名算法，其延伸算法蒙特卡洛树搜索更是被用于构建了著名的围棋人工智能 AlphaGo 。作为一个搜索算法，它保证其能在任何时候结束搜索并返回当前已知的最优解，而且它是非启发的，在不加入任何领域特定知识的情况下它也能有很好的表现。在能够进行足够多数量的游戏模拟的情况下，蒙特卡洛搜索保证能够找出当前的最优步骤。

本文构建了基于规则和基于蒙特卡洛搜索的两种炉石传说人工智能，并通过实验对比了它们的决策强度差异。同时，通过添加启发式的优化与剪枝策略，展示了向蒙特卡洛搜索中加入领域特定知识是否能提高智能体的表现。除此之外，本文还浅述了蒙特卡洛树搜索实现炉石传说智能体的基本方法，但并未对该智能体进行试验测试。相关的实验将作为本文的后续工作。

【关键词】：人工智能；蒙特卡洛搜索；蒙特卡洛树搜索；炉石传说

[Abstract]

In this paper, we present Hearthstone: Heroes of Warcraft, a famous online card game, as an interesting test bed for AI research. Hearthstone has already released over 700 different kinds of cards. All these cards change the rules of the game to some extent, which causes developing a completed AI for this game extremely difficult (though this paper only takes into account a very small amount of all these Hearthstone cards), not to consider the stochastic and imperfect information characteristics of the game.

We discuss how to use Monte Carlo Search to develop AI player for Hearthstone. Monte Carlo Search has always been a classic algorithm in the field of AI research, as its descendant – Monte Carlo Tree Search – being the major component of the famous Go AI player – AlphaGo. As a search algorithm, Monte Carlo Search guarantees to return an temporarily-optimal result at any moment in time, while allowing the algorithm to run for longer period often improves the result. Additionally, Monte Carlo Search is aheuristic – the lack of domain-specific knowledge cannot prevent it from finding the optimal result.

For the experiment, we construct three different AI players for Hearthstone, being complete-random AI, rule-based AI and Monte Carlo Search AI respectively. Games between different AI players will be carried out to demonstrate the strength of these players. Heuristic optimizations will also be added to the Monte Carlo Search AI to examine if the additional domain-specific knowledge can improve the performance of Monte Carlo AI. On the other hand, we also demonstrate the principle of constructing a Hearthstone agent using Monte Carlo Tree Search, but the agent is not tested in the experiment. Detailed research on Monte Carlo Tree Search agent will be the future work of this paper.

[Keywords]: Artificial Intelligence; Monte Carlo Search; Monte Carlo Tree Search; Hearthstone

目 录

1 引言

《炉石传说：魔兽英雄传》（Hearthstone: Heroes of Warcraft，下简称“炉石传说”）是由暴雪娱乐推出的线上集换式卡牌游戏。2014 年 3 月 11 日，炉石传说登陆 Microsoft Windows 平台并在全世界发行。此后，炉石传说相继登陆了OS X、iOS和安卓平台，吸引了大批的忠实玩家。截止至 2015 年的 11 月，炉石传说已拥有超过 4 千万的注册用户，并为暴雪娱乐带来了每月超过 2 千万美金的利润。与此同时，暴雪娱乐也为炉石传说举办了世界级的比赛以吸引人气。两届炉石传说世界锦标赛分别在 2014 年和 2015 年的暴雪嘉年华上举行，两届的冠军选手均获得了 10 万美元的奖金，奖金池总额更是达到了 25 万美元。炉石传说在 2014 年游戏大奖（the Game Awards）上被评选为年度最佳手机游戏，并在 2015 年游戏大奖上被提名为年度电子竞技游戏。

作为卡牌游戏，炉石传说也有着像桥牌那样的随机性和不完全信息性。与桥牌等传统扑克牌游戏不同的是，玩家在炉石传说中使用的卡组不是固定的标准卡组，而是由玩家根据情况自行构建的卡组。炉石传说中的卡牌都能够在不同程度上改变游戏的规则，不同卡牌之间的相互影响更是大大提高了游戏的灵活性。在本论文所讨论的炉石传说构筑模式中，玩家需要从炉石传说的所有可用的卡牌中选取出 30 张卡牌构成自己的卡组。而炉石传说中所有可用的所有卡牌种数已经达到了 743 种，且每年均会有 200 ∼ 300 张新的卡牌被加入到游戏中。游戏本身有着极高的多样性，不同的卡组可能意味着完全不同的打法，因此要开发一个相对有竞争力的炉石传说人工智能更类似于开发一个普适游戏人工智能（General Game Playing）[1]。本质上，炉石传说的核心规则其实极为简单，其规则的多样性主要来自于卡组的变化。由此，炉石传说十分适合作为不完全信息游戏、随机游戏以及普适游戏的人工智能领域的研究对象。

这篇论文将讨论如何用蒙特卡洛方法搜索炉石传说的最优策略。基于蒙特卡洛方法的炉石传说智能体将与基于规则的智能体进行相互对弈，以判明何种数量

的对弈模拟可以使得蒙特卡洛智能体拥有与规则智能体同等的对弈能力。在实验中，蒙特卡洛智能体会先使用较弱的（随机）智能体进行模拟，以验证在模拟数量足够大的情况下，蒙特卡洛智能体是否能打败较强的规则智能体；而后，规则智能体将作为蒙特卡洛智能体的模拟策略，以验证在为蒙特卡洛智能体的模拟策略加入启发知识的情况下，蒙特卡洛智能体的对弈能力是否能有所提高。

1.1 研究现状

卡牌游戏是典型的不完全信息随机游戏，其中包括了对手手牌的不可见信息以及洗牌后随机的牌堆出牌顺序。信息隐藏与随机性两者相结合，使得卡牌游戏对于人类玩家或是人工智能开发来说都是十分有趣的领域 [2]。其中，被大量用于人工智能研究的卡牌游戏包括扑克牌和桥牌 [3]。

作为卡牌游戏，扑克牌（Poker）的一局对弈最多可容纳 8 名玩家。扑克牌的对弈决策包含了概率分析以及对手行为预测等多个方面 [4, 5]。要成为一个扑克牌高手通常需要根据自己现有的手牌来正确估计自己的当前形势，并据此来决定自己跟或不跟。相关的人工智能研究多数集中在研究如何通过基于现有的手牌进行多次模拟来判断当前有多大几率获胜 [6]。除此之外，也有相关研究在探索如何判断对手的决策强度，并以此对智能体进行调整。研究显示，贝叶斯分析可被用于根据玩家已有的出牌方式来判断他正在使用的策略，甚至判断出他何时会改变自己的策略 [7, 8]。

桥牌（Bridge）同属卡牌游戏，一局对弈包含 4 名被分为两队的玩家。在考虑到应用蒙特卡洛搜索算法可能可以使桥牌人工智能拥有更强的决策强度后，Ginsberg 运用了包含蒙特卡洛搜索算法、分部搜索 [9] 以及各类优化算法在内的相关技术开发出了一款名为 GIB 的桥牌人工智能。GIB 也是首款在决策强度上能与人类桥牌大师相提并论的桥牌人工智能。

事实证明，部分传统最小最大搜索算法所不能应付的游戏，使用蒙特卡洛搜索算法时可以有不错的效果。围棋本身属于完全信息游戏，但它与国际象棋 [10] 等不同的地方在于，其过大的分支系数使得运用暴力搜索的人工智能难以获得良

好的表现。在无法使用简单的搜索算法来开发围棋人工智能的情况下，人们提出使用蒙特卡洛方法 [11]。自此，各种不同的基于蒙特卡洛方法的搜索算法被相继提出 [12]，其中包括了将蒙特卡洛方法与策略搜索相结合的搜索算法 [13]以及后来的蒙特卡洛树搜索算法 [14]。所有的这些算法都基于蒙特卡洛方法的基本思想：与其在决策时考虑未来每一步所有可能的走法（进而产生一棵巨大的对弈树），倒不如只考虑当前所有可能的走法，然后使用一个随机或启发式的模拟策略将当前对弈模拟到终止状态，并根据模拟的对弈结果来更新该走法的收益值。这背后的本质在于，当我们无法验证未来所有可能的对弈状态变化时，足够数量的随机模拟对弈的结果可以近似反映走法的期望收益。

除此之外，研究人员还提出了一种名为赌博机决策（Bandit Based Planning）的方法 [15]。方法的名称来源于概率论中著名的多臂赌博机问题（Multi-armed Bandit Problem）。该问题的内容大致如下：给定一排若干数量的赌博机，所有的这些赌博机都有着各自不同的出奖几率，那么为了获取最高的期望收益，参与者应该以何种顺序去尝试这些不同的赌博机呢？Kocsis et al [15] 将一个可用于在多次尝试赌博机后获取最大收益的算法利用于对弈树搜索中，以找到当前的最优策略。由此，一个名为 UCT（Upper Conficence Bounds for Trees）的算法被提出。算法首先会对当前所有可用的策略进行一次采样，然后在一个概率模型的指导下再在所有这些策略中选取某个策略来再次采样。如此一来，算法便能很好地在穷尽某个策略（exploitation）和探索其他策略（exploration）之间获得良好的平衡，因为拥有高收益的策略将会被更多地采样，其他收益相对较低的策略也不会被完全抛弃，也会被不时地进行采样。而该决策方法已在围棋上获得了巨大的成功 [16, 17]。

与此同时，UCT 算法还被应用于普适游戏领域并获得了不错的成果 [18]。一款游戏人工智能的核心在于搜索和估价，其中搜索功能可用于预测游戏未来的走向而估价功能则可被运用于评估搜索功能发现的策略的收益。在普适游戏中，人工智能无法使用任何先验的与游戏有关的特定知识来对对弈状态进行估价，所有可用的知识只可以通过对弈的过程自行推断 [19]。由于蒙特卡洛搜索算法和 UCT

算法均无需依赖于任何启发式的估价函数，它们最终都能够在普适游戏领域获得巨大的成功 [20]。

在炉石传说中进行决策，最大的难点仍然在于隐藏的信息（对手的手牌不可见）以及随机性（对手和自己抽到的下一张牌不可知）。我们需要考虑到对手的手牌可能是炉石传说 743 种卡牌中的任何一张，所有暴力的枚举算法最终都必须应对无比巨大的分支系数（考虑到巨大的分支系数正是无法为围棋开发基于暴力搜索算法的人工智能的主要原因）。除此之外，在没有将对弈模拟至游戏结束的情况下，我们也难以开发出一个完备的效用函数对炉石传说的某个中间状态进行估价。

实际上，炉石传说并不是第一种这种类型的卡牌游戏。早在炉石传说之前，同为集换式卡牌游戏的万智牌就已经风靡全球。尽管在规则上存在着不同点，人们看到更多的是万智牌和炉石传说之间的相似之处：无论是英雄角色的定位、卡牌种类的差异还是生物对战，万智牌和炉石传说可谓是一脉相承。最关键的是，两款游戏同样有着极大的随机性和分支系数。在 2009 年， Cowling 和 Ward 将蒙特卡洛搜索应用于万智牌的手牌决策，通过与随机或基于规则的生物攻击和防御智能体进行相互组合构建出了 9 种不同的智能体，并进行相互对弈 [21]。对弈的结果显示，比起使用随机或是预定义规则进行决策的智能体，使用蒙特卡洛搜索进行手牌决策的智能体都有着相对较高的胜率，胜率的提高在 $4\% \sim 7\%$ 之间。到了 2012 年，他们和 Powley 一起深入研究，进而将蒙特卡洛树搜索算法应用于万智牌的手牌决策 [22]。在论文中，他们使用了确定化（Determinization）的方法来处理万智牌中带有随机效果的操作，并对比了对蒙特卡洛树搜索智能体的各项参数调整所带来的效果。结果显示，各项调整都能使智能体在胜率上提升 $5\% \sim 20\%$。

实际上，确定化的思想并不新鲜，早前，该方法便已被用于处理一些概率决策问题 [23] 以及同为卡牌游戏的斗地主 [24]。除此之外，在上述的研究中，蒙特卡洛方法只被应用于了万智牌的手牌决策。万智牌和炉石传说的区别在于，万智牌的玩家回合分为多个阶段，包括出牌阶段以及生物攻击阶段。玩家只能在回合

的出牌阶段打出手牌，而在生物攻击阶段则只能对进行攻击的生物进行指派。上述的研究中并未将蒙特卡洛方法应用于生物的攻击和防御决策，因此其对万智牌决策的应用是不完全的。炉石传说的玩家回合并不包含多个阶段，手牌的使用和随从的攻击可以以任意顺序进行，进行的顺序不同会导致不同的结果。这也就导致了炉石传说智能体无法将手牌决策与攻击决策相互割裂。

最近，已有相关的研究团队开始将机器学习算法应用于炉石传说 [25]。尽管实验的结果并不理想，机器学习智能体在面对规则智能体时只能获得 10% 的胜率，但随着游戏进行次数的提升，智能体的胜率也有了上升的趋势，可见智能体已具备一定的学习能力。除此之外，也有研究团队在进行炉石传说卡组自动构建的研究 [26]。可见，作为一款新兴的卡牌游戏，炉石传说已经引起了研究人员的注意。

本文将在 Cowling 和 Ward 的研究的基础上，将蒙特卡洛搜索和蒙特卡洛树搜索分别应用于炉石传说，并对智能体进行部分的配置调优以测试各种不同的优化能在何种程度上提高智能体的决策强度。

1.2 论文结构

本文余下部分结构如下：第 2 章将用于讲述本文将使用的蒙特卡洛方法的基本概念；第 3 章讲述炉石传说的基本规则并给出形式化定义；第 4 章将详细描述本文实验所使用的规则智能体的组成；第 5 章则描述本文所使用的蒙特卡洛智能体；第 6 章为实验结果部分，并在第 7 章给出本文结论以及未来可能的工作方向。

2 预备知识

本章将对本文所使用方法的基本定义进行详细说明。其中，2.1 节将给出蒙特卡洛方法的基本定义，以让读者理解蒙特卡洛方法的主要思想。2.2 节将介绍经典的 K 臂赌博机问题，并说明如何将某个游戏状态中所有的可选操作视作一个 K 臂赌博机问题。2.3 节将给出如何使用蒙特卡洛搜索算法解决操作决策的 K 臂赌博机问题。2.4 节先从极小极大值算法开始，介绍博弈游戏使用博弈树遍历未来游戏状态的必然性，进而介绍蒙特卡洛搜索与博弈树的结合算法：蒙特卡洛树搜索。

2.1 蒙特卡洛方法

蒙特卡洛方法（Monte Carlo methods）实际上包括了一大类各种不同的计算算法，这类算法依赖于重复的随机采样来近似求取数值结果。蒙特卡洛方法主要起源于统计物理领域，被用于求出某些不可解的积分方程的近似值，现多被用于求解那些无法通过数学方法求解的物理或数学问题，而其重复采样求取近似值的思想更是被延伸到了其他各种领域。

Abramson [27] 首次展示了蒙特卡洛方法的随机采样可以被用于近似某个对弈决策的理论收益。这里使用 Gelly 和 Silver 在 [28] 中所提出的代数符号，那么某个对弈操作的期望收益可以表示为 Q 值如下：

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s)} \mathbb{I}_i(s,a) z_i \tag{2.1}$$

其中 $N(s,a)$ 为操作 a 在状态 s 下被选择的次数，$N(s)$ 为游戏从状态 s 被模拟的总次数，z_i 为从状态 s 开始的第 i 次模拟的结果；当操作 a 在第 i 次模拟被从状态 s 选中时，$\mathbb{I}_i(s,a)$ 为 1，否则为 0。

有一些蒙特卡洛方法在给定的对弈状态下会对所有的可以进行的操作进行均匀采样，这样的蒙特卡洛方法被称为平蒙特卡洛方法（flat Monte Carlo）。Ginsberg 曾利用这样的蒙特卡洛方法开发出了能与世界级选手相竞的

桥牌人工智能 [29]，可见这样的蒙特卡洛方法已有一定的实力。这样的方法的不足之处也是显而易见的：它完全无法对对手的行为进行任何预测 [30]。然而，在模拟时根据以往的经验偏向某些操作是完全可以提高所得的操作期望收益的准确度的。根据已有的操作期望收益，我们完全可以选择偏向那些拥有较高收益的操作。

2.2 赌博机方法

赌博机问题（bandit problems）是概率论中的经典问题。问题给定 K 个可进行的操作（例如，K个不同的赌博机），要求参与者在这些操作中进行选择，以使得在多次选取后获取最高的累计收益。每个操作所对应的收益分布是未知的，这使得参与者无法在这些操作中进行理论上的最优选择，他只能通过曾经进行的尝试的结果来估算每个操作背后可能对应的期望收益。这就产生了经典的穷尽-探索困境（exploitation-exploration dilemma）：参与者必须在“穷尽（exploitation）当前已知的最优操作”与“探索（exploration）其他现在不是最优但有可能在多次模拟后成为最优操作的操作”之间进行取舍。

一个 K 臂赌博机可被定义为随机变量 $X_{i,n}$，其中 $i \in [1, K]$ 代表第 i 个赌博机（赌博机的第 i 个“臂”）[15, 31, 32]，$X_{i,1}, X_{i,2}, \ldots$ 依次为尝试第 i 个赌博机时得到的结果，它们之间相互独立且一致地遵循着某个未知的分布法则，而该收益分布有着未知的期望值 μ_i。定义一个可以根据赌博机过往给出的奖励决定该尝试哪个赌博机的策略（policy）即可解决 K 臂赌博机问题。

2.2.1 后悔程度

给定的策略应能使得玩家的后悔程度（regret）最小。在 n 次尝试后，玩家的后悔程度可定义如下：

$$R_N = \mu^* n - \mu_j \sum_{j=1}^{K} \mathbb{E}[T_j(n)] \tag{2.2}$$

其中 μ^* 为期望收益最高的赌博机的期望收益，$\mathbb{E}[T_j(n)]$ 代表在这 n 次尝试中尝试了第 j 个赌博机的期望次数。换句话说，玩家的后悔程度可以被理解为因没能尝

试收益最高的赌博机所带来的损失期望。值得注意是，无论为任何时候，任何一个赌博机被选中的几率都不能为零，否则收益最高的赌博机可能会因为其他暂时拥有较高收益的赌博机而被搜索算法所忽略。为了保证这一点，我们应为每一个赌博机所观察到的收益值加上置信上界（Upper Confidence Bound）。

2.2.2 置信上界

为了解决 K 臂赌博机问题，搜索算法有必要引入置信上界，因为在任何时候，任何一个赌博机都可能是最优的赌博机。由 Auer et al 提出的名为 UCB1 的策略 [31] 可以在未预先设置任何与收益分布有关的启发知识的情况下使玩家的后悔程度随尝试次数 n 呈对数级增长（$O(\ln n)$）。该策略在每次选取赌博机时都会按照给定的公式为每个赌博机计算其对应的值，并选取所对应的值最大的赌博机。该公式定义如下：

$$UCB1 = \bar{X}_j + \sqrt{\frac{2\ln n}{n_j}} \tag{2.3}$$

其中 $\bar{X}_j$ 为第 j 个赌博机的平均收益，n_j 为第 j 个赌博机被尝试的次数，n 为尝试的总次数。不难看出，收益项 $\bar{X}_j$ 鼓励搜索算法穷尽（exploitation）目前平均收益最高的赌博机，而 $\sqrt{\frac{2\ln n}{n_j}}$ 项则鼓励搜索算法探索（exploration）其他尝试次数较少的赌博机。

2.3 蒙特卡洛搜索

在了解了赌博机问题以及用于解决赌博机问题的 UCB1 算法后，蒙特卡洛搜索算法的定义就变得十分清晰了。实际上，蒙特卡洛搜索的基本思路就是将给定的一系列可选操作视作一个 K 臂赌博机问题，通过大量的随机采样结合类似 UCB1 的算法来近似每个可选操作的期望收益，最终选出期望收益最高的操作。使用 UCB1 算法的蒙特卡洛搜索的基本流程如下：

初始化 对每个赌博机各尝试一次；

循环 选取使得公式

$$\bar{x}_j + C\sqrt{\frac{\ln n}{n_j}} \tag{2.4}$$

最大化的赌博机 j 进行尝试。其中 $\bar{x}_j$ 为赌博机 j 过往的平均收益，C 为一个 0 和 1 之间的常数，决定搜索算法尝试其他赌博机的可能性大小；n_j 为赌博机 j 过去被尝试的次数，$n = \sum_i n_i$ 为尝试次数的总和。

2.4 蒙特卡洛树搜索

蒙特卡洛搜索算法的缺陷在于，它只能枚举出当前玩家当前可选的所有操作，并通过随机或启发式的模拟采样来近似这些操作的期望收益。实际上，启发式或随机的模拟都无法对对手的行为进行准确的预测。尽管大量的随机采样可以确保所得期望收益在某种意义上算是比较可靠。但这样的期望收益是考虑到对手所有可能采用的策略后得出的结果，而实际上对手只会使用其中少数几种行为策略，因此这样得出的期望收益反而是不准确的。由此，蒙特卡洛搜索算法实际上无法对对手的行为作出准确预测，其决策的强度也会因此有所折扣。

2.4.1 极小极大值算法与博弈树

传统的人工智能算法多依赖树状结构来考虑对手可能的行为。这样的算法包括了经典的极小极大值算法。极小极大值算法需要构造一棵完整的博弈树，其中每个结点代表着一个游戏状态，子结点则代表着当前玩家在当前状态下可以进行的操作所对应的下一游戏状态。整棵博弈树会不断地向下延伸，直到叶子节点进入游戏的终止状态（已分出胜负）。此时，极小极大值算法依赖一个预定义的效用函数对终止状态（叶子结点）进行估价，进而根据终止态的价值计算从根结点到该叶子结点的路径上的结点的估价。

公式 2.5 给出了极小极大值算法计算结点收益值的方式。

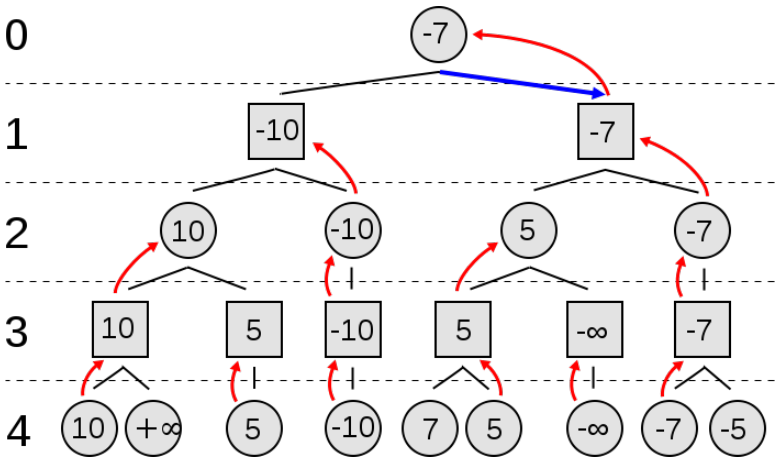

图 2.1 极小极大树

$$\text{MINIMAX-VALUE}(n) = \begin{cases} \text{UTILITY}(n) & \text{if } n \text{ is terminal state} \\ \text{MAX}_{s \in \text{SUCCESSORS}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is MAX node} \\ \text{MIN}_{s \in \text{SUCCESSORS}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is MIN node} \end{cases} \tag{2.5}$$

简单而言，我方优势越大，终止态的收益越高。代表我方行动的博弈树结点会被标记为 MAX 结点，其收益值等于其所有子结点收益值的最大值；代表敌方行动的结点被标记为 MIN 结点，其收益值等于其所有子结点收益值的最小值。这么做相当于考虑对手总是会以最优的决策进行游戏，以此来提高根结点直接子结点所代表的操作的期望收益的可靠性。

然而，极小极大值算法的缺陷在于，它首先必须完整地构造出整棵博弈树。对于像井字棋这样的状态较少的游戏固然是无足轻重，但对于现代游戏来说，游戏的状态数量往往是极大的，完整构造整棵博弈树通常是不现实的。除此之外，极小极大值算法还依赖一个对终止状态进行估价的效用函数。同样，对于复杂的现代游戏来说，开发一个准确的效用函数是极为困难的。极小极大值算法的优点在于，它和博弈树的结合确实能使智能体考虑对手可能进行的操作，并做出最优的决策。那么有没有可能即保留极小极大值算法的优点，又摒弃它的缺点呢？答案就是蒙特卡洛树搜索。

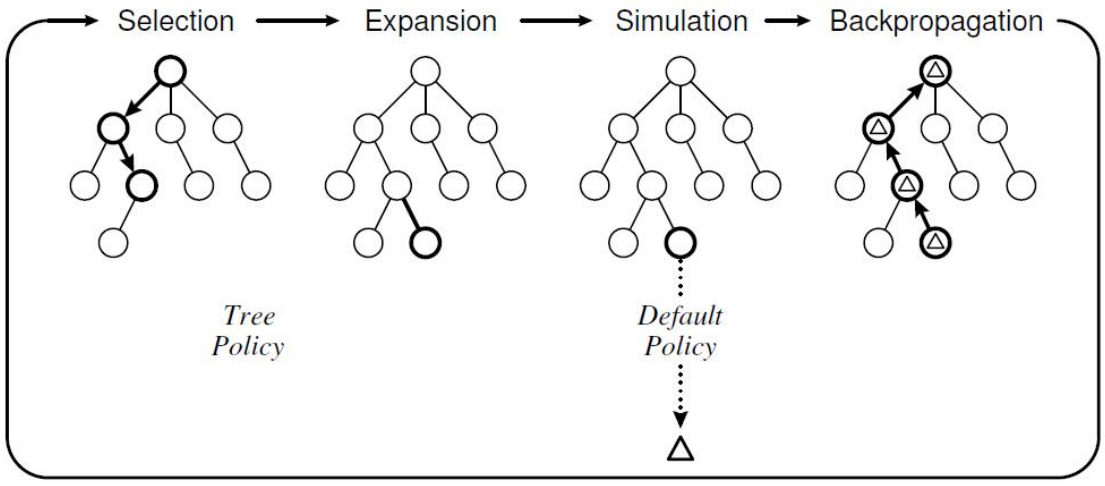

图 2.2 蒙特卡洛树搜索的迭代循环

2.4.2 蒙特卡洛树搜索算法的基本流程

蒙特卡洛树搜索算法（Monte Carlo Tree Search）实为将蒙特卡洛搜索算法与博弈树相结合而成的算法。比起像极大极小值那般依赖效用函数来得出期望收益，蒙特卡洛树搜索算法将每个结点及其子结点均视作一个 K 臂赌博机问题，使用蒙特卡洛搜索的思想得出子结点的期望收益。因此，蒙特卡洛树搜索算法的核心思想依然是基于大量的随机采样来对结点的期望收益进行近似。

蒙特卡洛树搜索的基本流程包括不断循环进行的迭代（选择结点并随机采样），并在迭代的过程中渐渐扩展已有的博弈树。如同蒙特卡洛搜索，迭代循环会一直持续，直到达成某些预定义的条件为止。树中的每个结点依然代表每个不同的游戏状态，其所有的子结点则代表当前玩家在当前状态下所有可进行的操作所对应的下一游戏状态。蒙特卡洛树搜索的一个循环可以被分为如下四步 [33]：

1. **选择**：从根节点开始，蒙特卡洛树搜索会递归地在每层都使用预定义的树策略（Tree Policy）选择一个子节点，并进入该子节点，直到找到一个可扩展的结点。如果一个结点并不代表非终结状态，且仍有未发现的子结点，那么我们说这个结点是可扩展的。

2. **扩展**：根据当前玩家在当前结点所代表的游戏状态下可进行的所有操作，为该结点产生子结点。

3. **模拟**：使用预定义的模拟策略（Default Policy）对这些新添加的子结点进行模拟，一直模拟到游戏结束并根据模拟结果产生出此次模拟的收益值。

4. **反向传播**：从该子结点开始，将其此次模拟的收益值一直向上传播，直到传播至根结点。

算法的过程涉及到了两个蒙特卡洛树搜索所使用的策略：

1. **树策略**：树策略用于选择需要扩展的结点。当给定一个结点时，树策略首先判断该结点是否需要扩展。如无需扩展，即该结点所有子结点已被发现，那么树策略将从所有的子结点中选择一个结点，指导蒙特卡洛树搜索沿该子结点继续向下寻找。

2. **模拟策略**：蒙特卡洛树搜索在对选定结点进行模拟时使用的策略。蒙特卡洛树搜索会使用该策略一直模拟至游戏结束。通常会使用完全随机的智能体作为模拟策略，但在模拟策略中加入一定的启发知识也有可能提升智能体的表现。

值得注意的是，蒙特卡洛树搜索对于博弈树的构建是不完全的，是非对称的（asymmetric）。蒙特卡洛树搜索会通过预定义的树策略，找寻到整棵博弈树中相对有价值的部分，并只对该部分进行探索，其它被认定为没有价值的子树则不会被构建。最终的效果便是整棵博弈树会偏向比较有价值的方向进行生长。由此，蒙特卡洛树搜索便有效规避了博弈树过大的风险。不过，尽管如此，随着扩展层数的增加，树的结点数依然会呈指数级增长，因此蒙特卡洛树在扩展到一定程度后便有可能需要停止扩展。除此之外，蒙特卡洛树搜索扩展出的博弈树的结构本身在某种程度上也能反映智能体对游戏的理解，我们也许也能从中获取关于游戏本身的有用信息。

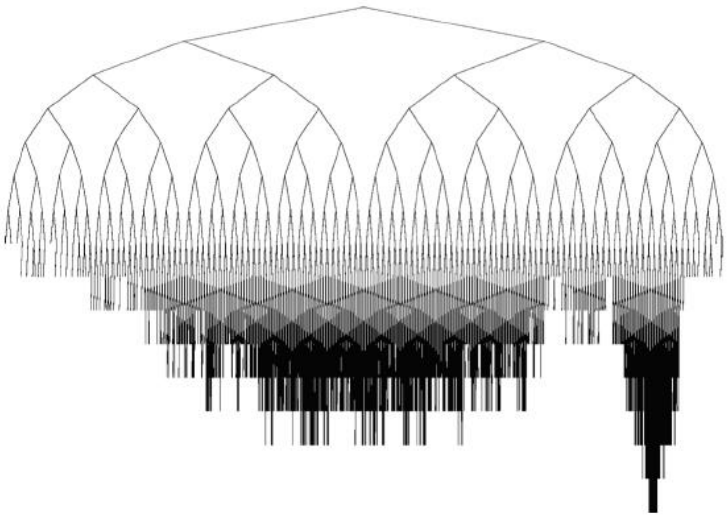

图 2.3 蒙特卡洛树搜索算法构建的非对称博弈树

作为蒙特卡洛搜索的延伸算法，蒙特卡洛树搜索保留了蒙特卡洛搜索非启发的特点：整个算法不依赖任何游戏特定知识，也不依赖任何用于对局面进行估价的效用函数。只要游戏状态的转换可以被构造成树状结构，蒙特卡洛树搜索可以在不做大幅改动的情况下直接应用于任何游戏。实际上，尽管完整构建博弈树的极小极大值算法显得不现实，但从理论层面来讲，极小极大值算法给出的答案确实是真正的最优解。无奈的是，即便是优化过后的有限深度的极小极大值算法依然过于依赖效用函数。对于像国际象棋这种在很多年前便已研发出可靠的效用函数的游戏固然是最好的，但对于像围棋或是炉石传说这样难以开发效用函数的游戏，极小极大值算法的应用便变得不可能了。

尽管蒙特卡洛树搜索确实可以完全脱离启发知识的存在，但加入特定的启发知识往往可以使算法在性能和表现上获得一定的提升。事实上，现在大多数的顶级围棋人工智能都加入了不少的启发知识，这些启发知识往往都是可以通过棋盘的模式进行判断的机器学习算法 [34]。这些启发知识并不需要是完备的，因为蒙特卡洛树搜索的随机采样特性能够弥补这一点。启发知识的加入只是为了让蒙特卡洛树搜索在进行决策时能更好地偏向那些看起来更“合理”的选择。尽管如此，启发知识的加入仍然是有代价的：完全随机的模拟采样保证了模拟时的效率，而启发知识的加入往往会大幅降低模拟的效率。除此之外，启发知识的加入

也会使得蒙特卡洛树搜索的结果产生偏差，具体是向好的方向偏还是不好的方向偏则取决于加入的启发知识是否能够很好的反应实际的情况。如果加入的启发知识无法准确地反映真实情况，启发知识的加入反而会降低蒙特卡洛树搜索的决策强度。因此，应加入何种程度的启发知识也是值得研究的方向，本文的实验部分也会针对这个问题进行相关测试。

除此之外，蒙特卡洛树搜索仍然保留了蒙特卡洛搜索的随时性（anytime）：每一次模拟的结果都将会被立刻反向传播至根节点，这保证了在任何一次迭代结束后，所有结点所保存的期望收益都是最新的。由此，蒙特卡洛树搜索确保了智能体可以在任何时候结束迭代并返回当前期望收益最高的操作，尽管进行更多的迭代往往能够更进一步提升其所得结果的可靠性。

2.4.3 UCB1 算法的蒙特卡洛树应用：UCT 算法

如 2.4.2 节所述，蒙特卡洛树搜索的核心思想在于通过大量的随机采样来近似求得当前状态可进行所有操作的期望收益。在重复迭代的过程中，蒙特卡洛树搜索会不断地扩展已有的博弈树，以实现对完整博弈树的局部搜索，如图 2.2 与图 2.3 所示。具体一棵博弈树会以怎样的方式被构建取决于蒙特卡洛树搜索在迭代的过程中如何选择进行扩展的结点，也即是其所使用树策略的具体实现。在将每一次选取子结点进行深入探索视为独立的多臂赌博机问题的过程中，蒙特卡洛树搜索通过使用多次的蒙特卡洛模拟来近似求得每个子结点的期望收益，也正对应着每个赌博机所拥有的拥有未知分布规则的随机奖励。

2.2 节提到了使用 UCB1 算法结合蒙特卡洛模拟来解决多臂赌博机问题（蒙特卡洛搜索）。 UCB1 算法本身简单高效，同时也确保了后悔程度随尝试次数的增长处于最优增长率的常数倍数范围内（$O(\ln n)$）。因此， UCB1 算法的这些特性也使得它适合用于解决蒙特卡洛树搜索中的多臂赌博机问题，而在蒙特卡洛树搜索中，每一次的子结点选取都可被视为多臂赌博机问题。由此，UCT 算法（Upper Confidence Bounds for Trees）作为 UCB1 算法与蒙特卡洛树搜索的结合诞生了。

UCT 算法在本质上与 UCB1 算法的区别并不大：在给定一系列的子结点时，UCT 算法将选取使得如下公式所得值最大化的子结点 j：

$$UCT = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{2.6}$$

其中 n 为当前结点（父结点）曾被访问的次数，n_j 为子结点 j 曾被访问的次数，而 C_p 为一个大于 0 的常数。该公式与 UCB1 算法的公式 2.4 相比并无太大的变化，它仍然可以通过调整 C_p 变量的值来改变算法在探索与穷尽之间的取舍。在多数情况下，C_p 可被直接设为 $1/\sqrt{2}$ [32]。

UCT 算法剩余的部分与 2.4.2 节的描述基本相同：在搜索算法从根结点通过公式 2.6 找到可扩展的结点后，搜索算法便会对该结点进行扩展。而后，搜索算法会使用预设的模拟策略为新添加的子节点进行模拟直到游戏结束。最后，游戏结束后的终止状态对应的收益会沿博弈树反向传播至根节点。

来自 [35] 的算法 1 和算法 2 为 UCT 算法基本过程的伪代码。其中，每个结点 v 包含 4 个相关变量，包括：其对应的游戏状态 $s(v)$ 、进入该结点所需进行的操作 $a(v)$ 、总模拟收益 $Q(v)$ 和曾经的访问次数 $N(v)$ 。项 $\Delta\ (v,p)$ 代表收益向量 Δ 中属于玩家 p 和结点 v 的元素。主函数 UCTSEARCH 的返回值为 $a(\text{BESTCHILD}(v_0,0)$，即为通往最高收益子结点的操作 a 。尽管如此，蒙特卡洛树搜索也可以选择返回访问次数最多的子结点而不是收益最高的子结点，尽管在多数情况下这两种方式都会指向同一个结点。算法 2 给出了蒙特卡洛树搜索反向传播的伪代码，其中 BACKUP 函数为常规的反向传播方式，而 BACKUPNEGAMAX 函数则适用于由两个互相敌对的玩家进行轮流操作的零和游戏，如炉石传说。BACKUPNEGAMAX 函数通过在结点层数变换时反转模拟收益 Δ ，使得每一层的结点所对应的期望收益均对应于智能体所代表的玩家的收益，而不是当前玩家的收益。于此，蒙特卡洛树搜索便拥有类似极小极大值算法的模拟对手的能力了。

算法 1 UCT 算法（Part 1）

function UCTSEARCH(s_0)
  create root node v_0 with state s_0
  while within computational budget **do**
    $v_l \leftarrow$ TREEPOLICY(v_0)
    $\triangle \leftarrow$ DEFAULTPOLICY($s(v_l)$) BACKUP($v_0, 0$)
  end while
  return a(BESTCHILD($v_0, 0$))
end function

function DEFAULTPOLICY(s)
  while s is non-terminal **do**
    choose $a \in A(s)$ uniformly at random
    $s \leftarrow f(s, a)$
  end while
  return reward for state s
end function

function TREEPOLICY(v)
  while v is nonterminal **do**
    if v not fully expanded **then**
      return EXPAND(v)
    else
      $v \leftarrow$ BESTCHILD(v, C_p)
    end if
  end while
  return v
end function

function EXPAND(v)
  choose $a \in$ untried actions from $A(s(v))$
  add a new child v' to v with $s(v') = f(s(v), a)$ and $a(v') = a$
  return v'
end function

function BESTCHILD(v, c)
  return $\max_{v' \in \text{ children of } v} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$
end function

算法 2 UCT 算法（Part 2）

```
function BACKUP(v, △)
    while v is not null do
        N(v) ← N(v) + 1
        Q(v) ← Q(v)+ △ (v, p)
        v ← parent of v
    end while
end function

function BACKUPNEGAMAX(v, △)
    while v is not null do
        N(v) ← N(v) + 1
        Q(v) ← Q(v)+ △ (v, p)
        △← − △
        v ← parent of v
    end while
end function
```

3　炉石传说基本规则

本章主要讲述炉石传说游戏的基本规则，详细介绍炉石传说卡牌的分类以及在对弈中玩家可操作的游戏实体，并给出炉石传说游戏状态及回合流程的形式化表示。

《炉石传说：魔兽英雄传》，简称“炉石传说”，是由暴雪娱乐推出的一款免费线上卡牌游戏。炉石传说的世界观依托于暴雪推出的另一款经典游戏《魔兽世界》，将魔兽世界中的英雄、职业及其对应的法术技能变成了炉石传说中的卡牌。炉石传说包括多种游戏模式，有构筑模式、单人模式、竞技场模式和乱斗模式，而本论文只考虑其中的构筑模式。

在构筑模式中，玩家可以使用自己预先构筑好的卡组来匹配对手，进行游戏。玩家在选定卡组对应的英雄职业后，可以在该英雄的所有职业专属卡和中立卡之中精心挑选，构成一副由30张卡牌组成的卡组。每副卡组中，同种的卡牌最多只能存在两张。除此之外，稀有等级为“传说”的卡牌，一种最多只能存在一张。“传说”卡牌为炉石传说中稀有度最高的卡牌，与其他卡牌相比也有着强得多的效果。

炉石传说的完整游戏规则是十分复杂的，考虑到现存的 743 张卡牌大多数都能够在一定程度上改变游戏的规则，卡牌与卡牌之间的组合更是可以让游戏千变万化。其中尤以“传说”卡牌为甚。“传说”卡牌的效果多为随机效果，运气的好坏很可能直接决定玩家最终会赢还是输。图 3.1 为炉石传说中一张很经典的传说卡牌，炎魔之王拉格纳罗斯。不难想象，玩家花费大量费用召唤出拉格纳罗斯以后，其随机攻击目标的价值将直接决定玩家此次召唤是否值得。除此之外，不同卡组之间的强度存在着本质的区别，使用不同的卡组进行测试可能导致实验结果产生差异。因此，本论文的实验只保留炉石传说最基本的规则，并选取部分具有代表性的卡牌组成一副固定的测试卡组进行测试。

炉石传说的一局对局由两名相互敌对的玩家构成。一名玩家所能操控的游戏

图 3.1 炎魔之王拉格纳罗斯

实体包括英雄、英雄能力、英雄武器、法力水晶、手牌、牌堆以及在场的随从。图 3.2 给出了炉石传说对弈局面的基本表示。

双方的目标均是将对方英雄的生命值下降至 0 以下，英雄生命值首先到达 0 以下的玩家输掉对局，两名英雄的生命值同时到达 0 以下则双方都输掉对局。炉石传说的对局采用回合轮换制，两名玩家先后轮流地进入自己的回合。玩家只能在自己的回合中进行操作，当进入对方的回合时则无法进行任何操作。

在对局开始时会随机决定两名玩家的先后手。先手玩家可以抽 3 张牌，并在这 3 张牌中选择任意的牌将其重新洗入牌堆，再抽牌直到手中有 3 张牌作为自己的起始手牌。后手玩家则可以在对局开始时抽 4 张牌，并在进行同样的选牌过程后将其作为自己的起始手牌。同时，后手玩家还会额外获得一张“硬币”卡[1]。随后，对局先从先手玩家的回合开始。

在对局中两名玩家都会拥有各自的法力水晶。双方在对局开始前均不拥有任何法力水晶，但每次回合开始时，玩家均获得一枚额外的法力水晶，直到玩家拥

[1]硬币：0 费用法术卡，使玩家当前回合的法力值 +1

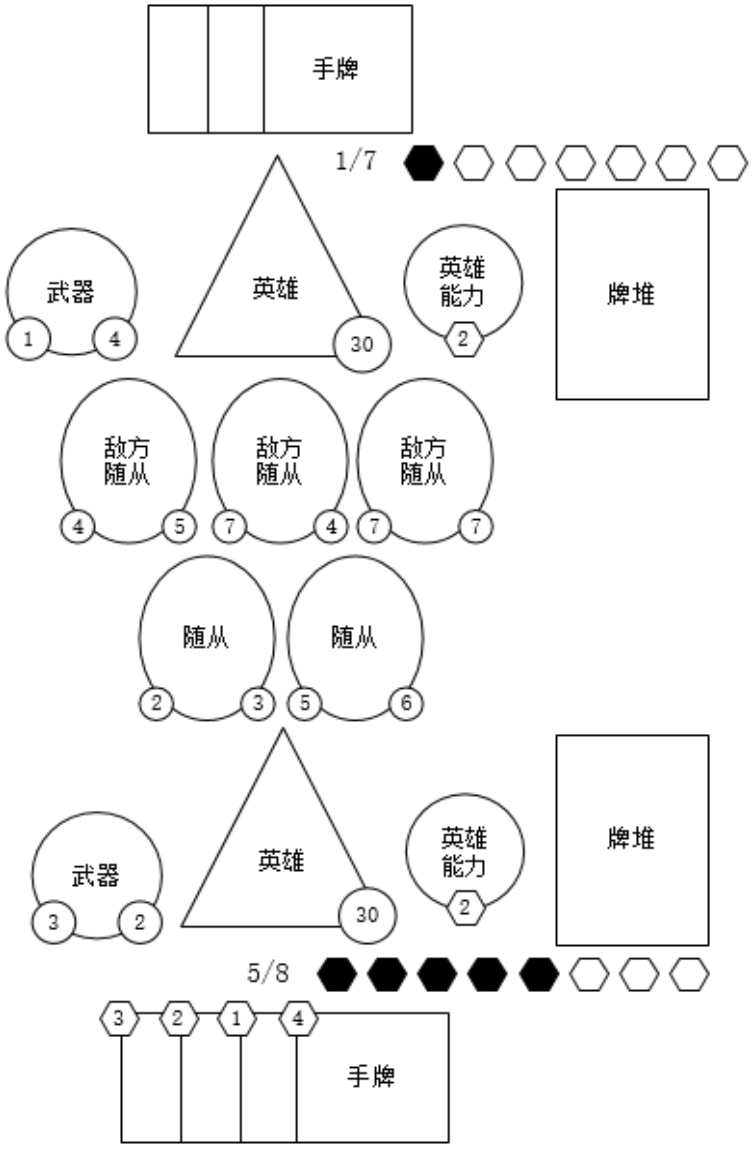

图 3.2 炉石传说局面基本表示

有的法力水晶达到 10 枚。法力水晶的数量代表着玩家在每个回合可支配的法力数。每一张卡牌都有着其对应的法力消耗，玩家使用卡牌将会消耗对应数量的法力值。通常消耗越大的卡牌会有着越强的效果。

在玩家的回合开始时，玩家从牌堆中抽一张牌，且所有法力水晶将重新充满。如果玩家的牌堆已经被抽空，那么玩家将会在每次试图抽卡时受到疲劳伤害。在每次造成疲劳伤害后，疲劳伤害会从 1 点伤害开始向上递增（ 2 点、 3 点、 4 点...）。除此之外，玩家的手中最多容纳 10 张牌。当手牌数达到上限后，玩家从牌堆中抽出的卡都将被直接烧毁（消失）。因此，根据对方卡组的策

略适当控制抽卡频率也是很重要的决策。在炉石传说中便存在着名为“爆牌贼”的盗贼卡组。该卡组最主要的策略是通过大量触发对手的抽牌动作，烧毁对手牌堆中大部分的卡牌，并最终令对手死于疲劳伤害。

进入己方回合后，玩家即可任意进行操作，包括使用卡牌、命令随从和英雄攻击以及使用英雄能力。不同的卡牌有着不同的效果。按类别来分的话，可以分为法术卡、随从卡和武器卡，分别可用于使用法术、召唤随从或为英雄装备武器以进行攻击。正如前文所说，消耗越大的卡牌意味着越强的效果，也就是越强的法术、越强的随从和越强的武器。

图 3.3 炉石传说卡牌基本信息

图 3.1 与图 3.3 给出了炉石传说中三类卡牌的实例。其中，图 3.1 给出的“炎魔之王拉格纳罗斯”为随从卡，使用随从卡可以召唤对应的随从。每张随从卡都包含五个基本信息：位于牌面中部的随从名称、位于牌面左上角的法力消耗 C 、位于牌面左下角的攻击力 A 、位于牌面右下角的生命值 H 和位于牌面下侧中部的效果描述。在接下来的内容中，若未加额外说明，随从将用 $C/A/H$ 来表示。若法力消耗本身无关紧要时，也可能直接表示为 A/H。

图 3.3 的左图给出的“鹰角弓”为武器卡，使用武器卡可为己方英雄装备对应的武器。每张武器卡同样包含五个基本信息，但位于牌面右下角的是武器的耐久度 D。在一般状态下，英雄的攻击力为 0，无法进行攻击。英雄装备武器后，英雄将获得武器的攻击力加成，并可以进行攻击。在每次攻击后，武器都会失去 1 点耐久度。当耐久度为 0 时，武器将会损毁，英雄本身也将扣除对应的攻击力。除了武器以外，少部分法术同样可以提高英雄的攻击力。同样，武器可以被表示为 $C/A/D$ 或 A/D。

图 3.3 的右图给出了名为“冰冻陷阱”的法术卡。法术卡只包含名称、法力消耗和效果三个信息，使用法术卡可以施放对应的法术。除此之外，“冰冻陷阱”为奥秘卡。使用奥秘卡可以为我方设置一个奥秘。奥秘会在敌方回合时激活，并在满足特定条件时触发。

在命令随从和英雄进行攻击时，可以对对方的随从或是英雄进行攻击。当攻击对方攻击力不为 0 的随从时，攻击的随从或英雄会受到对方随从的反击，受到与目标随从攻击力等同的伤害。随从在生命值下降到 0 以下时会死亡。考虑到一名玩家同时只能控制 7 个随从，当达到上限时将无法召唤更多的随从，因此适当地牺牲随从也有一定的战略意义。通常来讲，同一个英雄或随从在一个回合中都只能攻击一次，合理地选择它们各自的攻击目标和顺序也可能带来截然不同的战术收益。

英雄能力的使用同样需要消耗一定的法力，且通常每回合只能使用一次。与卡牌不同的是，卡牌在使用过后便会消失，某一张卡牌在一场对局中可用的次数是有限的，而英雄能力在整场对局中的使用次数则是没有限制的。

图 3.4 给出了炉石传说中回合的基本流程。

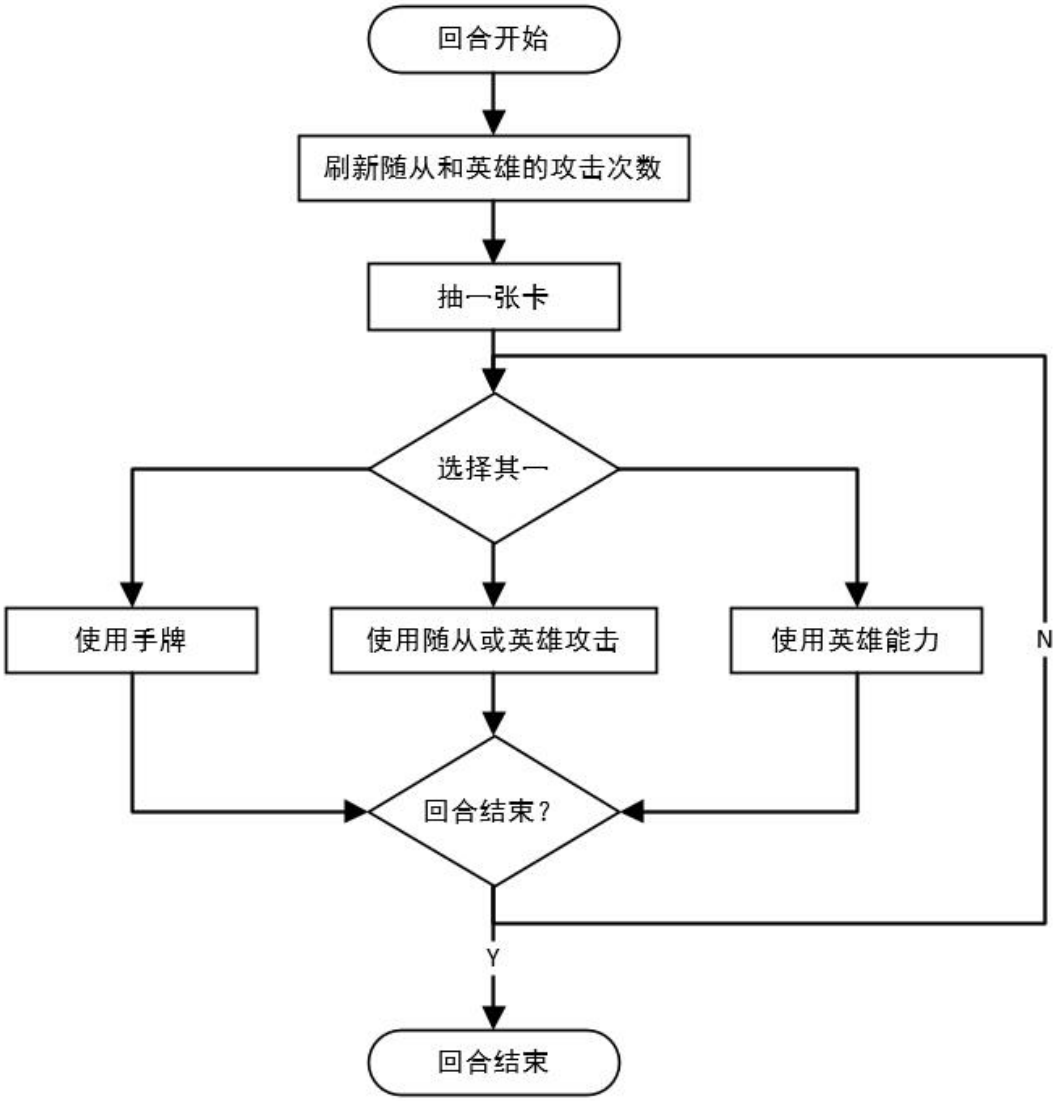

图 3.4 炉石传说基本流程

4 基于规则的智能体

本章主要讲述实验部分用于作为智能体参考对手的规则智能体的详细设计。4.1 节先从实验使用的测试环境开始，详细介绍实验时智能体所使用的固定卡组；4.2 节开始介绍本文所使用的规则智能体的详细设计，并给出相关的伪代码；4.3 节将在规则智能体的基础上修改得出新的随机规则智能体，该智能体将在实验部分用于验证启发知识对蒙特卡洛智能体决策强度的影响。

4.1 测试环境

在炉石传说中，不同卡组的强度是不一致的，部分卡组会本质上强于其他卡组。因此，本论文所有的智能体均会使用相同的卡组，卡组的组成如表 4.1 所示。

该卡组借鉴了以往出现过的许多胜率稳定的猎人卡组，其中包含的卡牌的法力消耗跨度较大，包括了从消耗小且体量小的随从卡到消耗大且体量大的随从卡，以及一定的法术卡。除了法力消耗的跨度，随从卡的效果包含了炉石传说中绝大部分的典型随从效果，其中包括战吼[1]、亡语[2]、潜行[3]、冲锋[4]以及圣盾[5]；法术卡则包含了指向性与非指向性法术，以及部分奥秘卡。卡组中甚至还包括了名为“火车王里艾诺”和名为“加兹瑞拉”的传说卡牌。作为传说卡牌，“火车王里艾诺”和“加兹瑞拉”对于猎人卡组都是很有用处的卡牌，同时也拥有比其他卡牌更复杂的效果。综上所述，这个卡组十分适合用于测试智能体对各种不同效果的卡牌的处理能力。

在给定的测试环境下，智能体需要进行的决策包括如下：

1. 决定使用哪张手牌；

[1]战吼：战吼效果会在随从被召唤时触发。

[2]亡语：亡语效果会在随从阵亡时触发。

[3]潜行：潜行随从无法作为敌方法术或随从直接攻击的目标。潜行随从在进行攻击后将不再潜行。

[4]冲锋：随从被召唤后会进入疲劳状态，无法攻击，直至下一回合。拥有冲锋效果的随从在被召唤后不会进入疲劳状态，可以直接攻击。

[5]圣盾：拥有圣盾的随从将免疫下一次受到的伤害，并失去圣盾。

表 4.1 测试智能体使用的猎人职业卡组

名称	类型	消耗	攻击	生命	效果	数量
猎人印记	法术	0			使一个随从的生命值变为1	2
奥术射击	法术	1			造成2点伤害	2
叫嚣的中士	随从	1	2	1	**战吼**：本回合中， 使一个随从获得+2攻击力	2
狼人渗透者	随从	1	2	1	**潜行**	2
爆炸陷阱	法术	2			**奥秘**：当你的英雄受到攻击时， 对所有敌人造成2点伤害	2
冰冻陷阱	法术	2			**奥秘**：当一个敌方随从进攻时， 将其移回到所有者手中， 并使其法力值消耗增加(2)	2
鬼灵爬行者	随从	2	1	2	**亡语**：召唤两个1/1的鬼灵蜘蛛	2
鹰角弓	武器	3	3	2	每当有一张你的**奥秘**牌被揭示时， 便获得+1耐久度	2
杀戮命令	法术	3			造成3点伤害。如果你控制 野兽，则改为造成5点伤害	2
关门放狗	法术	3			战场上每有一个敌方随从， 便召唤一个1/1并具有**冲锋**的猎犬	2
蜘蛛坦克	随从	3	3	4		2
冰风雪人	随从	4	4	5		2
火车王里艾诺	随从	5	6	2	**冲锋**，**战吼**：为你的对手 召唤两只1/1的雏龙	1
石拳食人魔	随从	6	6	7		2
加兹瑞拉	随从	7	6	9	每当该随从受到伤害， 便获得攻击力翻倍	1
强袭坦克	随从	8	7	7	**圣盾**	2

2. 决定使用哪个随从或英雄进行攻击，以及攻击的目标；

3. 决定是否使用英雄能力，以及英雄能力的目标。

除此之外，智能体还需要决定上述操作的先后次序。

英雄或随从的攻击决策是完全信息的：所有可进行攻击的角色和可被攻击的目标都是可穷举的，攻击后的效果也是确定的。大部分英雄能力使用后的效果也是确定的，但少部分职业的英雄能力本身也存在一定的随机性。表 4.1 所示卡组所属职业的英雄能力的效果是确定的。手牌使用的决策则是不完全信息的：是否

使用某一张手牌还需要考虑下一张可能抽到的牌，因为特定卡牌之间的配合可以在炉石传说之中打出更好的效果；除此之外，对方的手牌和牌堆的信息也是隐藏的，我方手牌的使用同样需要考虑这些信息。

本文构建了一个可进行随机决策的智能体用以测试其他智能体的决策强度。除此之外，本文还将额外使用了两个不同强度的基于规则的智能体。

4.2 专家规则智能体

首先，本节将详细介绍强度稍高的专家规则智能体所使用的启发知识细节。

算法 3 规则智能体决策方法 PRODUCEMOVE

```
1:  function PRODUCEMOVE(S)
2:      S ← current game state
3:
4:      A = action sequence ← ∅
5:      a ← null
6:      repeat
7:          a ← PLAYCARD(S)
8:          if a = null then
9:              a ← ATTACK(S)
10:         end if
11:         if a = null then
12:             a ← PLAYHEROPOWER(S)
13:         end if
14:         if a ≠ null then
15:             A ← A ∪ a
16:         end if
17:     until a = null
18:     return A
19: end function
```

函数 PRODUCEMOVE （算法 3）作为规则智能体产生操作序列的主框架。首先， PRODUCEMOVE 会获取到当前的游戏状态，并依次进行手牌使用、随从攻击和使用英雄能力的决策。其中， PLAYCARD 、 ATTACK 和 PLAYHEROPOWER 函数分别用于这三种决策。在完全枚举所有可进行的操作后， PRODUCEMOVE 将返回当前的操作序列A。

算法 4 使用手牌 PLAYCARD

```
1:  function PLAYCARD(s)
2:      S ← current game state
3:      P_A ← current active player
4:      P_O ← opponent of P_A
5:
6:      A_O = (opponent attackers) ← (A_1/H_1, A_2/H_2, ..., A_m/H_m)
7:
8:      if Σ_i A_i ≥ P_A.hero.health then
9:          return PLAYEMERGENCYCARD(S) if it is not null
10:     end if
11:
12:     C_M = (minion cards in the hand of P_A) ← (C_1, C_2, ... C_n)
        where C_1.cost ≥ C_2.cost ≥ ... ≥ C_n.cost
13:
14:     for C_i in C_M do
15:         if C_i.cost ≤ P_A.mana then
16:             return play minion card C_i
17:         end if
18:     end for
19:
20:     return null
21: end function
```

函数 PLAYCARD （算法 4）用于产生使用手牌的决策。首先，在 6 ∼ 10 行， PLAYCARD 会首先计算对方所有可攻击角色的攻击力总和，并将结果与我方英雄剩余生命值进行比较。如果敌方攻击力总和超过了我方英雄剩余生命值，意味着对方在下一个回合就能将我方击败，那么 PLAYCARD 就会调用 PLAYEMERGENCYCARD 函数，产生用于自保的手牌使用方式。当 PLAYEMERGENCYCARD 函数不能给出任何自保方式时， PLAYCARD 便会采取原本的出牌策略继续进行，留由后面的 ATTACK 函数和 PLAYHEROPOWER 函数来化除险境。在 12 ∼ 18 行， PLAYCARD 函数会选择手牌中可用的法力消耗最高的随从卡，并使用该卡。如果手牌中不存在任何可用的随从卡， PLAYCARD 函数则会返回 null ，表示所有可进行的手牌使用已枚举完毕。

算法 5 使用手牌进行自保 PLAYEMERGENCYCARD

```
1: function PLAYEMERGENCYCARD(s)
2:     S ← current game state
3:     P_A ← current active player
4:     C_Mt = (taunt minion cards in the hand of P_A) ← (C_1, C_2, ... C_n)
   where C_1.cost ≥ C_2.cost ≥ ... ≥ C_n.cost
5:
6:     for C_i in C_Mt do
7:         if C_i.cost ≤ P_A.mana then
8:             return play minion card C_i
9:         end if
10:    end for
11:
12:    return null
13: end function
```

函数 PLAYEMERGENCYCARD （算法 5）用于在检测到敌方可攻击角色的攻击力综合超过我方英雄剩余生命值时，使用现有的手牌进行自保。PLAYEMERGENCYCARD 本身的逻辑十分简单，仅仅只是选择手牌中可用的法力消耗最高的嘲讽随从并打出。嘲讽随从的存在可以使得敌方角色无法直接攻击我方英雄，以此便能达到保护我方英雄的目的。

函数 ATTACK （算法 6）用于枚举随从与英雄的攻击动作。在第 4 行，ATTACK 函数会首先获取所有可进行攻击的友方角色，并对其按攻击力进行降序排序。第 6 ∼ 8 行用于判断边界情况：当不存在任何可进行攻击的角色时，意味着 ATTACK 函数已完成所有角色攻击决策的枚举，因此返回 null 。第 10 ∼ 16 行用于枚举所有可攻击的敌方目标。首先 ATTACK 函数会尝试获取可攻击的敌方嘲讽随从，若不存在，则尝试攻击敌方威胁较大的随从。如何定义某个敌方随从对我方“威胁较大”是比较复杂的问题。本文仅将攻击力超过一定阈值的随从视作威胁较大的随从。若依旧不存在威胁较大的随从时，则将所有敌方目标选定为可攻击的备选目标。第 18 ∼ 22 行将从攻击力最高的我方角色开始，为其选定最优的攻击目标。首先在第 18 ∼ 20行， ATTACK 函数会尝试获取一个该我方角色可以杀死的生命值最大的敌方随从，并指定我方角色攻击该目标。如果不存在这样

算法 6 角色攻击 ATTACK

1: **function** ATTACK(s)
2: $S \leftarrow$ current game state
3:
4: $As =$ (friendly attackers) $\leftarrow (A_1/H_1, A_2/H_2, \ldots, A_m/H_m)$ where $A_1 \geq A_2 \geq \ldots \geq A_m$
5:
6: **if** As is empty **then**
7: **return** null
8: **end if**
9:
10: $Ts =$ (enemy taunt minions) $\leftarrow (a_1/h_1, a_2/h_2, \ldots, a_n/h_n)$ where $h_1 \geq h_2 \geq \ldots \geq h_m$
11: **if** Ts is empty **then**
12: $Ts =$ (enemy dangerous minions)
13: **end if**
14: **if** Ts is empty **then**
15: $Ts =$ (all enemy targets)
16: **end if**
17:
18: **if** there is $T \subseteq Ts$ with $h_j \leq A_1$ (for all $a_j/h_j \in T$) **then**
19: **return** attack $\{a_j/h_j | a_j/h_j \in M \wedge h_j = max(h_1, h_2, ...)$ for all $a_k/h_k \in M$ with A_1/H_1
20: **end if**
21:
22: **return** attack enemy hero with A_1/H_1
23: **end function**

的目标，即该我方角色无法直接杀死任何敌方角色，第 22 行则会直接指定我方角色攻击敌方英雄，以求通过激进地压低敌方生命值为对方增加压力，从而拉大优势、赢得对局。

最终， PRODUCEMOVE 函数调用 PLAYHEROPOWER 函数，在仍有剩余法力值的情况下使用英雄能力。

正如前文所述，炉石传说是很复杂的游戏，如此简单的规则智能体固然无法与人类玩家相提并论。不难看出，该规则智能体只能对随从卡牌做出反应，法术

卡牌与武器卡牌都不会被规则智能体使用。这么做的原因在于，炉石传说中的武器卡牌和法术卡牌变化度极大，卡牌之间的共同性极小，要开发能处理这类卡牌的规则智能体十分复杂。有趣的是，由于炉石传说的受欢迎程度之大，在进行排名比赛时往往可以在较低的排名中匹配到使用其他规则智能体进行自动游戏以获益的玩家，即所谓的“脚本”。在与这些脚本进行对战时，本文定义的规则智能体往往表现不俗，能在多数情况下取胜。由此，尽管确实无法与人类玩家相提并论，本文使用的规则智能体已具有一定的决策强度。

4.3 随机规则智能体

另一个本文使用的强度较低的规则智能体，实为在 4.2 节所述规则智能体的基础上加上了一定的随机因素组合而来。在本文的实验中，蒙特卡洛智能体将会使用 4.2 节所述的专家规则智能体作为模拟策略，以求证在蒙特卡洛方法的模拟策略中加入启发知识是否能提高蒙特卡洛智能体的表现。遗憾的是，蒙特卡洛方法所得结果的可靠性来源于大量随机采样结果的统计，不带有任何随机因素的专家规则智能体有可能导致蒙塔卡洛方法模拟所得的结果产生偏差，反而导致智能体的强度有所下降。因此，通过将随机智能体与规则智能体相结合，也许能在某种程度上消除启发知识所带来的缺陷。具体效果将会在本文的实验部分进行验证。

相比于专家规则智能体，随机规则智能体引入概率 $p \in [0, 1]$ ，并做出如下改动：

- PRODUCEMOVE：作为规则智能体产生决策的主框架函数，随机规则智能体无需对 PRODUCEMOVE 函数做出修改。随机规则智能体仍将以相同的顺序产生决策。
- PLAYCARD：随机规则智能体在进入 PLAYCARD 函数后，有 $1 - p$ 的可能性使其随机地使用卡牌。具体的做法是在所有可用的手牌中随机选择一张并使用。
- ATTACK：随机规则智能体在进入 ATTACK 函数后，有 $1 - p$ 的可能性使

其随机地为我方角色指派攻击目标。具体的做法是，首先被选中的我方角色有 50% 的可能选择不攻击；选择攻击时，攻击的目标也会随机地进行选择。

- PLAYHEROPOWER：随机规则智能体的 PLAYHEROPOWER 函数未做任何改动。

简单而言，随机规则智能体是完全随机智能体和专家规则智能体的集合，而概率变量 p 定义了它们之间所占的比重：当 $p = 0$ 时，随机规则智能体将无法进入专家规则智能体的决策逻辑，完全使用随机智能体进行决策，因此效果上完全等同于完全随机智能体；反之，当 $p = 1$ 时，随机规则智能体将无法进入随机智能体的决策逻辑，完全使用专家规则智能体进行决策，因此效果上完全等同于专家规则智能体；在其他情况下，p 值越大意味着智能体强度更大，也意味着智能体所携带的启发知识更多。

5 蒙特卡洛智能体

本章主要讲述本文所使用的蒙特卡洛智能体的主要设计。5.1 节首先给出将蒙特卡洛搜索应用于炉石传说的基本方法；5.2 节首先讨论了蒙特卡洛树搜索如何使用确定化的思想来应对像炉石传说这样的隐藏信息随机游戏，进而讨论了为何朴素的蒙特卡洛树搜索无法被直接应用于炉石传说，并在最后给出可对蒙特卡洛树搜索进行的相关优化。

5.1 蒙特卡洛搜索智能体

除了随机和规则智能体外，本文主要测试的智能体由蒙特卡洛搜索进行实现。具体而言，本文使用的蒙特卡洛智能体实现了可用于解决多臂赌博机问题的 UCB1 算法 [31]。在多臂赌博机问题中，参与者需要在若干个随机出奖的赌博机之间进行选择，而使用 UCB1 算法能确保参与者在足够多次的尝试后获得最高的收益。

根据 2.3 节所述的 UCB1 算法的基本过程，将其应用于炉石传说也并非难事。蒙特卡洛智能体的决策过程基本如下：

1. 根据当前游戏状态枚举出智能体在当前回合可以进行的所有操作序列；
2. 为每个可进行的操作序列均将游戏模拟至游戏结束，并保存模拟的结果；
3. 进入 UCB1 算法的循环阶段，使用公式 2.4 选取操作序列并将游戏模拟至游戏结束，然后保存模拟的结果；
4. 在满足一定的预设条件后结束循环，并返回平均收益 $\bar{x}_j$ 最高的操作序列。

5.2 蒙特卡洛树搜索智能体

上一节所描述的蒙特卡洛搜索智能体的缺陷在于其无法将操作过程中可能触发的隐藏信息纳入考虑，因为所有可能的操作序列在搜索开始前便以结束枚举。

能触发隐藏信息的操作在炉石传说随处可见，包括如图 3.1 中的炎魔之王拉格纳罗斯的随机效果，或是如图 5.1 中的会触发抽牌动作的卡牌。而使用蒙特卡洛树搜索则可以考虑这些可能被触发的隐藏信息。

图 5.1 能触发隐藏信息的炉石传说卡牌

5.2.1 确定化

确定化（Determinization）是一种可被用于为带随机和隐藏信息的游戏制作人工智能的办法，它又被称为完全信息蒙特卡洛（Perfect Information Monte Carlo）。对于给定的一局带有随机因素和隐藏信息的对弈，它的**确定化**即为与其等效的完全信息的确定型对弈，其中当前的对弈状态将根据智能体所知悉的信息集进行生成，而所有原本的概率事件在确定化的版本中，它们的结果都是确定的。比如说，卡牌游戏中最典型的游戏信息包括对手与我方牌堆出牌的顺序以及对手的手牌，那么在对应的确定化游戏中，双方的手牌和牌堆都是可见的。那么，智能体就可以通过已经知悉的信息推断出隐藏信息的可能结果，并从中选择若干个结果组成若干个确定化版本，而后对确定化的对弈采用智能体所采用的算

法进行决策，最终通过合并这些决策来决定在原本的隐藏信息的对弈下所应进行的决策。

在 [22] 一文中，确定化的蒙特卡洛树搜索被应用于万智牌的手牌决策。在决策时，智能体会为每一个确定化构造一棵独立的蒙特卡洛树。对于每一棵蒙特卡洛树，它们的所有隐藏信息的结果都会在构造前预先随机确定。以抽卡为例，当蒙特卡洛树进入状态 s 后需要进行抽卡，那么智能体将对抽卡的动作进行一次随机采样，得出一个拥有固定结果的结果操作 a，而操作 a 将使蒙特卡洛树从状态 s 进入状态 s'。那么，在之后每一次蒙特卡洛树进入状态 s 时，智能体便不会再进行采样，而是让蒙特卡洛树直接进入状态 s'。

确定化的使用无疑是必然的。考虑到万智牌的一副卡组中通常包含 60 张卡，其中大约可包含 15 种不同的卡。如果在蒙特卡洛树向下延伸的过程中每次遇到随机事件都考虑所有可能的结果的话，蒙特卡洛树的分支系数将会以 $O((n!)^2)$ 的速度上升。例如，一个回合中可进行的操作可能有 $5 \sim 6$ 种，那么第 1 层的分支系数可能就会达到 $75 \sim 90$，第 2 层就会达到 $7000 \sim 8000$，而在第 3 层的时候就上百万了。在这种情况下，如果仍需要蒙特卡洛树给出相对可靠的结果，那么模拟的次数同样也需要一同增长，最终导致蒙特卡洛树搜索难以在有限的时间内做出决策。尽管炉石传说中的一副卡组只包含 60 张卡，但分支系数上升的速度依旧是相同的。

在综合 2.4 节所述的蒙特卡洛树搜索基本原理与 5.1 节所述的蒙特卡洛搜索智能体的具体实现后，蒙特卡洛树搜索智能体的具体实现便十分直观，在此便不再赘述。

值得注意的是，蒙特卡洛树搜索与蒙特卡洛搜索在执行步骤上的差异在于，蒙特卡洛树搜索需要对一棵树进行不断地扩展，每一次迭代都有可能需要为某个结点扩充子结点。随着迭代数的增加，蒙特卡洛树的深度不断增加，在一次搜索中发生的可进行操作枚举的次数也会逐渐增加，而蒙特卡洛搜索则只需要在搜索的最开始的时候进行一次操作枚举。在实际测试时，蒙特卡洛树搜索智能体的时间占用在蒙特卡洛搜索智能体的十倍以上，性能的瓶颈就在于操作枚举的步骤。

对于炉石传说而言，由于可进行的三种决策的顺序是不定的，以不同的顺序执行同样的操作也有可能意味着不同的结果，因此随着游戏进行到大约第 6 回合时，暴力枚举所能获取的可执行的操作序列便可能达到数万个，而这其中只有少数几个是有价值的。因此直接将朴素的蒙特卡洛树搜索应用于炉石传说是不实际的。

5.2.2 相关优化

尽管蒙特卡洛树搜索智能体的实现十分直观，但鉴于蒙特卡洛树的深度会随着模拟次数的增加而增加，在不对蒙特卡洛树搜索进行一定的优化的情况下，智能体恐怕难以在有限的时间内给出可靠的结果。可对蒙特卡洛树搜索进行的性能方面的优化包括如下两点：

- **合并确定化采样**：确定化的思想要求在每次需要进行抽卡时都对抽卡的结果进行一次随机采样。实际上这样的效果等同于确定了双方牌组的出牌顺序，因此完全可以在构建确定化所拥有的蒙特卡洛树之前，提前确定该确定化所对应的双方牌组顺序以及对手的手牌，如此一来便无需在遍历蒙特卡洛树的过程中再进行采样。
- **操作剪枝**：蒙特卡洛树搜索的执行效率很大程度上取决于当前可进行操作序列的数量。当操作数量增加时，蒙特卡洛树搜索的耗时也会以很快的速度上升。除此之外，操作数量的减少也使得在固定模拟次数的情况下，不同的操作可以被进行更多次的模拟，使得其近似得出的期望收益更加可靠。通过预定义的基于规则的启发知识，智能体可在开始蒙特卡洛树搜索之前对部分显然无益的操作进行移除，以免智能体浪费时间为这些无益的操作进行模拟。值得注意的是，在对操作进行移除的过程中一定要确保该操作不可能为最优解，否则不准确的移除反而将导致智能体决策强度的下降。
- **启发式操作枚举**：在上一节的末尾，我们讨论了朴素蒙特卡洛树搜索应用于炉石传说的瓶颈。暴力枚举并不适合用于可操作实体数较多的炉石传说，为了解决这个性能瓶颈必须为枚举算法加入一定的启发知识，使其能

像人类玩家一样只去枚举有价值的操作序列。

除此之外，也有其他的一些优化方式可被用于蒙特卡洛树搜索智能体以提高其决策强度：

- **启发式模拟**：如 2.4.2 节所述，适当地在模拟策略中加入启发知识而不是使用完全随机的模拟策略可以使蒙特卡洛树搜索更好地模拟对手的行为。尽管完全随机的模拟策略确保了期望收益的可靠性，但完全随机的模拟策略往往需要更多的模拟次数来给出同样可靠的期望收益，而且所给出的期望收益是针对所有类型的对手的。如果只考虑智能体所面对的真实对手，其往往只可能属于其中的少数几种类型，那么完全随机模拟得出的期望收益反而是不准确的。

  值得注意的是，尽管启发式模拟没有了随机模拟的这些缺点，但启发式模拟本身的启发知识使得智能体的结果产生了偏差。当这些启发知识不能很好地模拟实际的对手时，其效果反而要比随机模拟来得更差。本文的实验部分将验证所加入启发知识的数量对智能体决策强度的影响。

- **收益递减**：实际上，使用蒙特卡洛树搜索进行决策的智能体有一个很大的特点，就是在拥有比较大的优势时决策强度反而开始下降。这种行为的成因在于，在智能体拥有比较大的优势时，其所能执行的操作大部分都会拥有比较高的期望收益，此时由于有限的模拟次数永远都是不充分的，加之操作间的期望收益差距过小，某些看起来不大合理的操作反而可能得出最高的期望收益。这种情况在智能体进入劣势时便会消失，在此时智能体往往能做出极其精确的最优操作。尽管对于多数游戏来说，在优势情况下的随意决策往往影响很小，但在炉石传说中，由于拥有极强效果的卡牌过多，而且回合数越高时，玩家可支配的法力水晶越多，对手打出这样的卡牌并最终翻盘的可能性也更高，因此炉石传说的智能体即便是在优势很大的情况下也应该继续通过尽可能拉大优势来尽早结束游戏。

  在蒙特卡洛树搜索进行反向传播的过程中，通过在每一层结点为传播的模

拟收益乘上一个递减系数 $\lambda \in (0, 1]$ 即可让智能体懂得尽可能早地结束游戏，因为如果某一种操作会使得游戏以更多的回合数才能取胜，尽管胜利的模拟收益仍为 1，但在逐层乘上递减系数后，操作对应的结点本身收到的收益反而比较小。

6 实验结果

本章为论文的实验结果部分。本论文的实验将测试 4 节和第 5 章提到的四种智能体，分别为随机智能体、专家规则智能体和两种蒙特卡洛搜索智能体。智能体将使用同一副卡组进行对弈，以免不同卡组之间的强度差异导致实验结果发生偏移。测试卡组的组成已在表 4.1 中给出。在 6.1 节，四种智能体首先将进行两两对弈并给出各自的胜场数，以让读者对不同智能体的平均决策强度差异有大概的认知；在 6.2 节中，4.3 节提及的随机规则智能体将用作蒙特卡洛智能体的模拟策略，以测试启发式的先验知识对蒙特卡洛智能体决策强度的影响。

表 6.1 测试智能体

智能体名称	智能体描述
Random	完全随机的智能体
Rule-based	基于预定义规则的智能体
MC Random	使用随机智能体进行模拟的基于蒙特卡洛搜索的智能体
MC Rule-based	使用规则智能体进行模拟的基于蒙特卡洛搜索的智能体

6.1 智能体之间的决策强度差异

本实验将用于测试不同智能体之间的决策强度的差异。智能体们将会进行相互对战。每组智能体各进行 200 次对战，并最终给出各个智能体在各种不同的组合中的胜场数。其中，蒙特卡洛智能体默认在对所有操作进行一次尝试后会额外进行 500 次迭代。测试时使用的计算机[1]可以在 1 秒内完成 500 次的模拟游戏。后面的实验结果也表明 500 次的迭代已足以让蒙特卡洛智能体做出足够强度的决策。本实验的结果也将用作后续实验的基准数据。

表 6.2 给出了此次实验的结果。不难看出，随机智能体比起规则智能体和蒙特卡洛智能体都弱得多。基于随机模拟的蒙特卡洛智能体在面对规则智能体时可以有 56% 的胜率，可见即使是未添加任何先验知识的蒙特卡洛智能体已能与拥有大量先验知识的规则智能体并驾齐驱。值得注意的是，当采用规则智能体进行模

[1]Intel Core i5-3210M @ 2.50GHz 2.50GHz

表 6.2 智能体对弈胜场数

VS.	Random	Rule-based	MC Random	MC Rule-based
Random	100	2	0	0
Rule-based	198	100	88	60
MC Random	200	112	100	140
MC Rule-based	200	140	60	100

拟时，蒙特卡洛智能体拥有了来自规则智能体的先验知识，同时也拥有了蒙特卡洛搜索返回最优解的保证，因此可以看到，基于规则模拟的蒙特卡洛智能体在面对规则智能体时的胜率比起基于随机模拟的蒙特卡洛智能体有了显著的提升（提升至 70%）。更为有趣的是，基于规则模拟的蒙特卡洛智能体与基于随机模拟的蒙特卡洛智能体直接对战时，基于规则的智能体反而处于劣势。

6.2 迭代次数与启发知识对蒙特卡洛智能体决策强度的影响

由 2.3 节可知，蒙特卡洛搜索通过不断地重复采样以近似每个操作的期望收益，采样次数越多意味着得到的期望收益越准确。该过程类似于近似值逐渐逼近正确值的过程，理论上可知当采样次数达到一定程度后，采样次数的提高所能带来的可靠性的提高会变得微乎其微。本次实验将调整上述实验中使用的两个蒙特卡洛智能体的迭代次数，以测试蒙特卡洛智能体在模拟次数逐渐增加时会有怎样的表现。

图 6.1 为随机规则模拟的蒙特卡洛智能体与规则智能体进行对战时，蒙特卡洛智能体的胜率的变化。其中每个结点代表的数据为蒙特卡洛智能体在给定的迭代次数和 p 值下，与专家规则智能体进行 500 场随机对弈后蒙特卡洛智能体的胜率。当 $p = 1$ 时，随机规则智能体的行为与规则智能体的行为是完全一致的；当 $p = 0$ 时，随机规则智能体的行为与随机智能体的行为是完全一致的。尽管变化不是很大，但仍能清楚地看到，在 $p = 1$ 时，随着迭代次数的增多，蒙特卡洛智能体的胜场数有上升的趋势。蒙特卡洛智能体的胜率在迭代数超过 100 后趋于稳定，稳定在了 67% 胜率附近。这个数字与 6.1 节得出的胜率颇为接近。与之形成对比的是，当 $p = 0$ 时，蒙特卡洛智能体的表现在迭代数较低的情况下较差，当

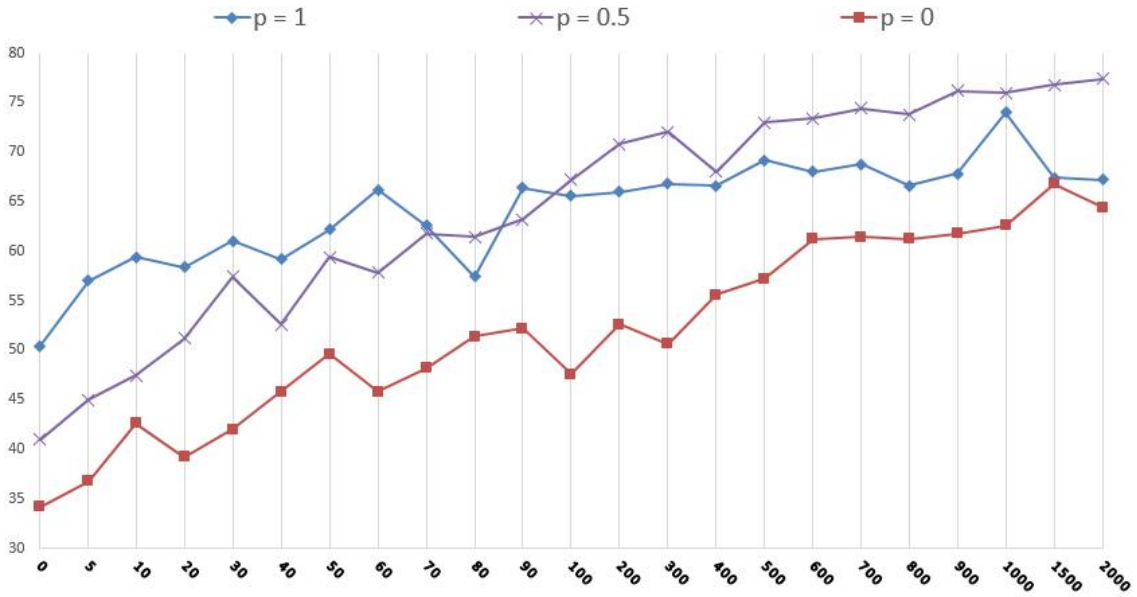

图 6.1 基于随机规则模拟的蒙特卡洛智能体 vs. 规则智能体

迭代次数达到 100 以上时蒙特卡洛智能体的胜率才达到了 50% 。随着迭代数的增加，蒙特卡洛智能体的决策强度仍在不断增加，当迭代数达到 2000 时，蒙特卡洛智能体的胜率已达到 60% ，与 $p = 1$ 时的表现比较接近。除此之外，图中还给出了当 $p = 0.5$ 时蒙特卡洛智能体的胜率的走势。有趣的是，在迭代数达到 100 时，蒙特卡洛智能体的胜率已达到了与 $p = 1$ 时相同的水平，而且仍在不断上升，最终超过了 $p = 1$ 时的表现，并在迭代数达到 2000 时达到了 75% 以上的胜率。

图 6.2 为 $p = 1$ 的蒙特卡洛智能体与 $p = 0$ 的蒙特卡洛智能体进行对战时，$p = 1$ 的蒙特卡洛智能体的胜场数的变化。与猜想的结果截然不同的是，随着两个智能体的迭代次数的增加， $p = 1$ 的蒙特卡洛智能体的胜场数在逐渐下降。当迭代次数达到 500 以上时，其胜率已经下降到了 35% 附近。

以上几次对弈测试的结果显示出了十分有趣的规律。首先比起随机蒙特卡洛智能体，规则蒙特卡洛智能体在与规则智能体对弈时，其胜率可以以更少的迭代次数（100）稳定在较高的胜率（65%）。尽管随机蒙特卡洛智能体表现不如规则蒙特卡洛智能体，但随机蒙特卡洛智能体也在一定的迭代次数后（500），胜场数也趋于稳定（63%）。当 $p = 0.5$ 时，蒙特卡洛智能体的胜率并未止步于 65% ，

图 6.2 基于规则模拟的蒙特卡洛智能体 vs. 基于随机模拟的蒙特卡洛智能体

而是一路攀升，最终全面超越了其在 $p = 1$ 时的表现。除此之外，当两个蒙特卡洛智能体进行相互对弈时， $p = 1$ 的蒙特卡洛智能体由于启发知识的加入反而输给了 $p = 0$ 的蒙特卡洛智能体。

蒙特卡洛搜索算法本身无法对对手的行为作出任何预测。完全随机的蒙特卡洛智能体由于不包含任何启发知识，在无法对对手的行为作出精准预测的情况下，它得出的操作序列的期望收益与现实情况可能是不匹配的。但随着迭代次数的增加，大量的随机采样往往能保证期望收益最终趋近于真实值。大概也正因如此，随机模拟的蒙特卡洛智能体需要更多的迭代来让胜场数趋于稳定，稳定后的胜场数也与 $p = 1$ 的蒙特卡洛智能体较为接近。当 $p = 1$ 时，蒙特卡洛智能体由于加入了启发知识，它能更好地预测规则智能体的行为。但正是由于启发知识的加入，蒙特卡洛智能体的迭代采样不是随机的，这就导致了它得出的期望收益是有偏差的，不适用于所有类型的对手，而完全随机的蒙特卡洛智能体没有这样的局限。因此随着迭代次数的增加，规则蒙特卡洛智能体渐渐被随机蒙特卡洛智能体超越，且差距越来越大。

7 结语

7.1 结论

本文在第 2 章详细介绍了蒙特卡洛搜索和蒙特卡洛树搜索的基本定义，并在第 3 章介绍了炉石传说的基本规则。第 4 章给出的本文实验的基本测试环境，并给出了测试中使用的规则智能体的详细定义。第 5 章给出了蒙特卡洛搜索与蒙特卡洛树搜索应用于炉石传说的基本方法，在 5.2 节末尾讨论了将蒙特卡洛树搜索直接应用于炉石传说所遇到的瓶颈，并在 5.2.2 节给出了可进行的优化。

第 6 章构建了四个不同的炉石传说智能体并使其进行两两对弈。最终得知，本文使用的规则智能体比普通的随机智能体有强得多的决策强度。完全不包含先验知识的 $p = 0$ 的蒙特卡洛智能体能够轻易击败最弱的随机智能体，与拥有大量先验知识的规则智能体对弈时亦能拥有不错的表现。$p = 1$ 的蒙特卡洛智能体加入了规则智能体所拥有的先验知识，在决策强度上有了一定的提升，能比随机蒙特卡洛智能体更快趋于稳定，稳定后的胜率也比随机蒙特卡洛智能体稍高。这意味着，蒙特卡洛搜索的加入能够提高智能体的决策强度，规则智能体的先验知识的加入同样能够提高智能体的决策强度，而先验知识的加入与蒙特卡洛搜索算法之间并不冲突。蒙特卡洛搜索意味着无启发知识的搜索型智能体，规则智能体则意味着依赖启发知识的启发型智能体。它们代表着截然不同的人工智能研究方向，而两者相结合所带来的成功意味深远。不幸的是，先验知识的加入导致规则蒙特卡洛智能体的表现发生了偏差：它能比随机蒙特卡洛智能体更好地应对规则智能体，但在面对其他类型的对手时的表现却有所下降。

除此之外，实验部分还验证了迭代次数对蒙特卡洛智能体决策强度的影响。可见，对于炉石传说而言，500 次的蒙特卡洛迭代已足以让蒙特卡洛搜索算法近似求得各个操作的期望收益，更多的迭代已不能继续得到更好的结果。考虑到 500 次的迭代对现代计算机而言只需要 1 秒的处理时间，蒙特卡洛人工智能在性能上也是可以接受的。

7.2 未来的工作方向

蒙特卡洛搜索固然能带来智能体决策强度的提高，但先验知识的加入对蒙特卡洛智能体的提升也是显而易见的。炉石传说人工智能仍然可以通过各种方式加入更多的先验知识：无论是基于规则的智能体，还是利用学习算法使得智能体拥有学习能力，本文的实验结果都确保了它们的加入能够更好地模拟对手的行为，为智能体的决策强度带来提高。在炉石传说这样的不完全信息游戏中，对手行为预测是十分重要的。而要对对手进行预测，加入启发知识并不是唯一的方法。实验中，规则模拟与蒙特卡洛搜索算法的结合所带来的决策强度的提升，也预示了基于启发式规则与基于博弈树搜索这两个方向的人工智能研究并不是相互矛盾的，两个方向的研究进展最终都可以通过相互结合来创造出更强大的智能体。

除此之外，本文并未能成功地将蒙特卡洛树搜索应用于炉石传说，主要的瓶颈在于炉石传说一个回合可进行的操作总数可由于可操作实体数量的上升暴增到数万以上，严重影响了蒙塔卡洛树扩展的效率，使得蒙特卡洛树搜索无法在可接受的时限内完成足够数量的迭代。未来需要考虑为炉石传说设计一套启发式的操作枚举系统，以期能以更高地效率排除价值较低的操作，进而使得蒙特卡洛树搜索能在可接受的时限内完成迭代。

参考文献

[1] Björnsson Y, Finnsson H. Cadiaplayer: A simulation-based general game player [J]. Computational Intelligence and AI in Games, IEEE Transactions on, 2009, 1(1):4–15.

[2] Bowling M, Johanson M, Burch N, et al. Strategy evaluation in extensive games with importance sampling [C]. Proceedings of the 25th international conference on Machine learning. ACM. 2008: 72–79.

[3] Schaeffer J. A gamut of games [J]. AI Magazine, 2001, 22(3):29.

[4] Billings D, Burch N, Davidson A, et al. Approximating game-theoretic optimal strategies for full-scale poker [C]. IJCAI. 2003: 661–668.

[5] Billings D, Davidson A, Schaeffer J, et al. The challenge of poker [J]. Artificial Intelligence, 2002, 134(1):201–240.

[6] Billings D, Davidson A, Schauenberg T, et al. Game-tree search with adaptation in stochastic imperfect-information games [C]. Computers and Games. Springer, 2004: 21–34.

[7] Baker R J, Cowling P I. Bayesian opponent modeling in a simple poker environment [C]. Computational Intelligence and Games, 2007. CIG 2007. IEEE Symposium on. IEEE. 2007: 125–131.

[8] Baker R J, Cowling P I, Randall T W, et al. Can opponent models aid poker player evolution? [C]. Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On. Ieee. 2008: 23–30.

[9] Ginsberg M L. Partition search [C]. AAAI/IAAI, Vol. 1. 1996: 228–233.

[10] Campbell M, Hoane A J, Hsu F h. Deep blue [J]. Artificial intelligence, 2002, 134(1):57–83.

[11] Brügmann B. Monte carlo go [R]. Citeseer, 1993.

[12] Chaslot G, Saito J T, Bouzy B, et al. Monte-carlo strategies for computer go [C]. Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium. 2006: 83–91.

[13] Cazenave T, Helmstetter B. Combining Tactical Search and Monte-Carlo in the Game of Go [J]. CIG, 2005, 5:171–175.

[14] Chaslot G, Winands M, Uiterwijk J, et al. Progressive strategies for monte-carlo tree search [C]. Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007). 2007: 655–661.

[15] Kocsis L, Szepesvári C. Bandit based monte-carlo planning [C]. Machine Learning: ECML 2006. Springer, 2006: 282–293.

[16] Wang Y, Gelly S. Modifications of UCT and sequence-like simulations for Monte-Carlo Go [J]. CIG, 2007, 7:175–182.

[17] Lee C S, Wang M H, Chaslot G, et al. The computational intelligence of MoGo revealed in Taiwan's computer Go tournaments [J]. Computational Intelligence and AI in Games, IEEE Transactions on, 2009, 1(1):73–89.

[18] Finnsson H, Björnsson Y. Simulation-Based Approach to General Game Playing [C]. AAAI. vol 8. 2008: 259–264.

[19] Clune J. Heuristic evaluation functions for general game playing [C]. AAAI. vol 7. 2007: 1134–1139.

[20] Sharma S, Kobti Z, Goodwin S. Knowledge generation for improving simulations in uct for general game playing [C]. AI 2008: Advances in Artificial Intelligence. Springer, 2008: 49–55.

[21] Ward C D, Cowling P I. Monte Carlo search applied to card selection in Magic: The Gathering [C]. Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on. IEEE. 2009: 9–16.

[22] Cowling P I, Ward C D, Powley E J. Ensemble determinization in monte carlo tree search for the imperfect information card game magic: The gathering [J]. Computational Intelligence and AI in Games, IEEE Transactions on, 2012, 4(4):241–257.

[23] Yoon S W, Fern A, Givan R. FF-Replan: A Baseline for Probabilistic Planning. [C]. ICAPS. vol 7. 2007: 352–359.

[24] Powley E, Whitehouse D, Cowling P. Determinization in Monte-Carlo tree search for the card game Dou Di Zhu [J]. Proc. Artif. Intell. Simul. Behav., York, United Kingdom, 2011, 17–24.

[25] Taralla D, Qiu Z, Sutera A, et al. Decision Making from Confidence Measurement on the Reward Growth using Supervised Learning: A Study Intended for Large-Scale Video Games [C]. Proceedings of the 2016 International Conference on Agents and Artificial Intelligence.

[26] Goes L W, da Silva A R, Rezende J, et al. HoningStone: Building Creative Combos with Honing Theory for a Digital Card Game [J].

[27] Abramson B. Expected-outcome: A general model of static evaluation [J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1990, 12(2):182–193.

[28] Gelly S, Silver D. Monte-Carlo tree search and rapid action value estimation in computer Go [J]. Artificial Intelligence, 2011, 175(11):1856–1875.

[29] Ginsberg M L. GIB: Imperfect information in a computationally challenging game [J]. Journal of Artificial Intelligence Research, 2001, 303–358.

[30] Browne C. The dangers of random playouts [J]. ICGA Journal, 2011, 34(1):25–26.

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

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

[33] Chaslot G, Bakkes S, Szita I, et al. Monte-Carlo Tree Search: A New Framework for Game AI. [C]. AIIDE. 2008.

[34] 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):484–489.

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

致　谢

在此论文完成之际，谨向我的导师致以衷心的感谢和崇高的敬意。我的导师细心地教导了我学术论文的基本写作过程，给予了大量的指导，特别是在毕业论文的选题过程中，给了非常多有效的指引和建议，使得这篇论文能以较高水平顺利完成。

衷心感谢我的家人对我的关心和理解，感谢你们一直以来的支持和鼓励。

最后，向百忙中审阅论文的各位老师致以崇高的敬意和深深的感谢。