中国科技期刊卓越行动计划项目入选期刊

控制与决策

CONTROL AND DECISION

抽象技术及其在蒙特卡洛树搜索中的应用研究综述

邵天浩, 程恺, 张宏军, 张可

引用本文:
邵天浩,程恺,张宏军,张可. 抽象技术及其在蒙特卡洛树搜索中的应用研究综述[J]. 控制与决策, 2024, 39(4): 1075−1094.

在线阅读 View online: https://doi.org/10.13195/j.kzyjc.2023.0652

抽象技术及其在蒙特卡洛树搜索中的应用研究综述

邵天浩, 程 恺†, 张宏军, 张 可

(陆军工程大学 指挥控制工程学院, 南京 210007)

摘 要: 抽象技术作为人工智能研究中高效拓展决策的重要组成部分,已广泛应用于大规模的决策问题. 蒙特卡洛树搜索虽然在众多决策领域取得了卓越成就,但是在现实决策问题中面临着决策空间巨大和规划周期很长的问题. 鉴于此,研究抽象技术及其在蒙特卡洛树搜索中的应用,从状态空间和动作空间两个角度出发分析抽象技术如何提升蒙特卡洛树搜索的决策能力,并对抽象蒙特卡洛树搜索研究中仍需要解决的问题和未来的研究方向作进一步展望.

关键词: 状态抽象; 动作抽象; 蒙特卡洛树搜索; 决策

中图分类号: TP181 **文献标志码:** A

DOI: 10.13195/j.kzyjc.2023.0652

引用格式: 邵天浩, 程恺, 张宏军, 等. 抽象技术及其在蒙特卡洛树搜索中的应用研究综述 [J]. 控制与决策, 2024, 39(4): 1075-1094.

A review of abstract technology and its application in Monte Carlo tree search

SHAO Tian-hao, CHENG Kai†, ZHANG Hong-jun, ZHANG Ke

(Command and Control Engineering College, Army Engineering University, Nanjing 210007, China)

Abstract: Abstract technology is an essential part of efficient decision-making in artificial intelligence research and has been widely used in large-scale decision-making problems. Although Monte Carlo tree search has achieved impressive results in many decision-making fields, it faces challenges of a vast decision space and long planning cycles in real-world decision-making problems. This paper investigates the application of abstract technology in Monte Carlo tree search and analyzes how it can enhance the decision-making ability of Monte Carlo tree search from the perspectives of state space and action space. Additionally, this paper provides further insights into the problems that still need to be addressed and future research directions in the study of abstract Monte Carlo tree search.

Keywords: state abstraction; action abstraction; Monte Carlo tree search; decision

0 引 言

抽象技术作为智能决策中理解和解决复杂问题最重要的技术之一,已被广泛应用于大规模的决策问题,通常方法是将决策中较细的元素组合或映射成粒度较大的元素,从而降低决策求解的复杂度[1-3].

Ho等[4]以人在复杂场景中能够做出正确决定为例子,从多个方面详细阐述了为何抽象技术能够有效地支持决策问题的求解. 以图1人在森林中进行徒步旅行为例,人在河流左侧,应该如何回到右侧的营地? 方案可能是“先穿过河到右侧,再走到营地”. 这一决策包含了抽象的概念,即人自然地将河流左侧和右侧映射为整体,将穿过河需要的动作组合成高层动作,即图1右侧所示. 抽象思维的存在,让人能够在复杂环境中进行分层决策,然而,计算机并没有抽象思维,智能体需要通过规则不断地与环境进行交互,尝试每一种可能的行动序列,最终得到类似于“首先需要绕过正前方这块石头,然后从右侧的小路一直往前走...”这样繁琐的底层行动序列.

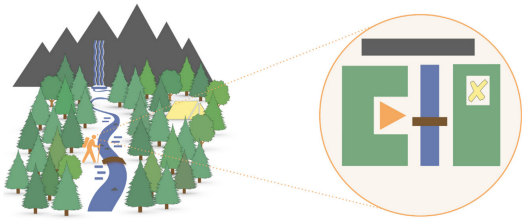

图 1 抽象行为示意图[4]

从案例可以看出,抽象思维在人工智能研究中是

收稿日期: 2023-05-15; **录用日期:** 2023-07-04.

基金项目: 国家自然科学基金项目(61806221); 国防科技创新特区项目(211-CXCY-A04-02-01-01).

†通讯作者. E-mail: chengkai911@126.com.

十分重要的. 目前,人工智能中的抽象技术研究主要集中在两种类型上:第一是状态抽象,状态抽象通过聚合环境的某些要素或者为它们分配共享的特征,将其视为相似的要素;第二是动作抽象,动作抽象通过将多个底层动作聚合成高级动作来完成.

蒙特卡洛树搜索(Monte Carlo tree search, MCTS)是一种用于解决序贯决策问题的强大方法[5],该方法以模拟的形式在决策空间中随机抽取样本并执行,根据结果的统计信息构建搜索树,以便在每个后续迭代中做出更优的选择,最终找到特定规划领域中的最优决策. 本质上来说,MCTS是一种以搜索树为代表的大型组合空间的决策算法,树的节点表示状态(问题的配置),节点到节点之间的边表示从一个状态到另一个状态的过渡(动作). 由于MCTS通过特有的树搜索方法平衡了探索(exploration)和开发(exploitation)两个因素,使其对人工智能方法产生了深远的影响.

自MCTS方法被提出以来,便成为许多人工智能问题研究的焦点. 最初,MCTS的研究聚焦在计算机围棋这项具有挑战性的任务中,并在早期使得计算机围棋达到业余5段水平. 在早期发展过程中,MCTS被成功应用到众多的决策领域,例如围棋[6-7]、VG(video game)[8-9]、RTS(real time strategy games)[10-11]等. 到2016年,MCTS与深度强化学习(deep reinforcem learning, DRL)的结合使得计算机围棋首次击败人类顶尖职业棋手[12-13],在人工智能领域引起轰动,从而掀起新一波人工智能研究的热潮. MCTS的研究在大数据和超级计算机的辅助下有了新的突破,在不同类型的游戏博弈问题上都有一定的应用[14],例如GGP(general game playing)[15-16]、GVGP(general video game playing)[17-18]、AVG(arcade video games)[19-20]和RTS[21-22]等,也在类似于《炉石传说》等不完全信息博弈游戏[23-25]以及各类规划和组合优化问题[26-29]中也有突破性的应用,同时,MCTS更多地作为深度强化学技术的辅助技术[30-34],使得各领域中训练的神经网络有更高的决策准确率.

MCTS通过搜索状态空间,从而建立关于特定状态下可用于决策的统计信息,每次迭代均包括4个阶段,如图2所示.

1) 选择阶段. MCTS从根节点开始对搜索树进行遍历,根据树策略(tree policy)不断选择下层节点,直到选择到叶节点(或结束节点).

2) 拓展阶段. MCTS在叶节点处概率随机地模拟执行一个动作,达到新的状态,将该状态所代表的新节点作为其子节点.

3) 模拟阶段. MCTS在拓展节点处依据模拟策略(rollout policy)模拟执行完决策的问题,直到达到终端状态(或指定深度),并获取特定的收益(奖励).

4) 反向传播阶段. MCTS将模拟阶段获得的收益从拓展节点开始反向上传至根节点,并更新整条路径上的统计信息.

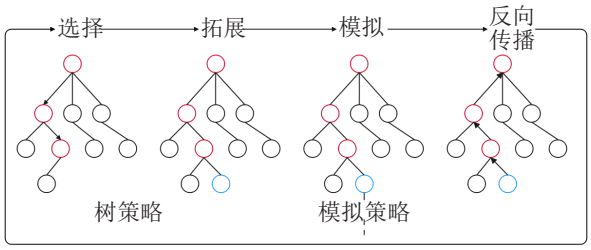

图 2 MCTS迭代过程

在上述4个阶段中,需要区分树策略(tree policy)和模拟策略(rollout policy)两个概念[14]. 树策略是指选择阶段使用的上层节点如何选择子节点的策略,其决定了如何平衡探索与开发两个因素,最常用的树策略即UCT(upper confidence bounds applied for trees)策略[35],UCT策略及其延伸策略是MCTS在众多领域成功的关键. 模拟策略是指模拟阶段从拓展节点开始一直到模拟结束所使用的策略,一般而言,模拟策略是一个完全随机的策略,因为MCTS的思想是通过大量的随机模拟来选择一个较好的确定性动作,但随着所需要解决的问题越来越多,模拟带来的算法复杂度呈指数上升,因此发展出使用启发式策略或神经网络替代随机策略的模拟策略,使得模拟过程能快速朝着需要的方向进行收敛,这也是MCTS近些年来成功的关键因素之一.

图3展示了自1996年以来与MCTS相关的文献数量随年份变化的曲线趋势,数据来源于Science Direct数据库,并对2023年的总体数据进行了预测. 从图3可以看出,近些年对MCTS的研究明显加快. MCTS的研究趋势主要分为3个阶段:一是MCTS被正式提出之前,蒙特卡洛思想已经被许多研究者应用到各个领域;二是从MCTS提出到深度学习广

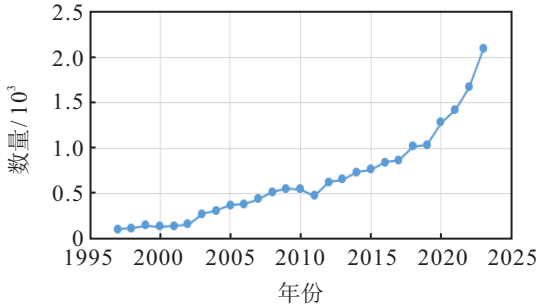

图 3 MCTS文献数量随年份变化曲线

泛流行之前,MCTS的研究主要集中在对MCTS方法本身的研究,包括树策略和模拟策略等;三是深度学习广泛流行尤其是AlphaGo被提出后,MCTS的研究更多地聚焦于如何辅助神经网络训练更快地收敛.

虽然MCTS在包括计算机围棋的多个领域已经取得了超越人类的性能,但随着现实需求的发展,其弊端也逐渐显现出来,即传统的MCTS无法处理大规模的决策问题. 由于MCTS模拟计算的复杂度随着决策问题的状态和动作空间而呈指数增长,当决策问题空间巨大时,即使是超级计算机也难以维持因多次模拟而产生的时间消耗. 当MCTS被应用于类似《王者荣耀》《星际争霸》等RTS游戏时,其性能一直不能达到超越人类的水平,即使与目前最先进的DRL相结合,也只能在特定任务或小场景中达到人类顶尖水平,并不通用. 进一步地说,人工智能的发展不仅仅是为了游戏博弈,更多的是解决现实问题,而在现实世界中,也不可避免地存在决策空间巨大和规划周期很长的问题,这导致MCTS的应用受到了一定的限制.

研究者将决策空间巨大和规划周期很长两个问题称为维度诅咒(curse of dimensionality)和历史诅咒(curse of history),而抽象的思想正好可以用于缓解这两大问题,即将原本扁平的MCTS方法转化为块状或层次状的MCTS方法,从而减小局部决策空间,降低局部规划周期.

现有文献中,将抽象技术应用到MCTS中的研究已经很多,但大多数局限于特定的场景使用具体的方法,没有对状态抽象和动作抽象两类方法进行整体性的归纳. 本文对抽象技术及其在MCTS中的应用研究现状进行归纳和总结,以探索研究中的不足之处,形成完整的基于抽象的MCTS研究体系,希望可以提升MCTS在现实问题中的求解能力.

本文首先将状态抽象技术分为状态聚类、状态转换图和神经网络3个类别,从各个角度介绍其在MCTS中的应用;然后将动作抽象技术分为经典动作抽象、状态转化图和神经网络3个类别,从各个角度介绍其在MCTS中的应用;最后详细介绍两种结合状态抽象和动作抽象方法的经典MCTS方法. 在上述基础上,对抽象技术在MCTS中的应用进行总结,指出当前研究的不足之处并对未来研究进行展望.

1　状态抽象及其在MCTS中的应用

状态抽象技术已经在决策领域得到广泛应用[36-38],提出和研究了许多与抽象有关的概念和方法,本节主要将状态抽象技术分为3个大类,后续将按照状态聚类、状态转化图和神经网络3个角度对状态抽象技术进行分类和总结,并介绍其在MCTS中的应用.

1.1　状态聚类

从状态聚类角度进行状态抽象是目前较为流行的一种做法,聚类是将对象的集合分成由类似的对象组成的多个类的过程. 由聚类所生成的簇是一组对象的集合,这些对象与同一个簇中的对象彼此相似,与其他簇中的对象相异.

从聚类角度考虑状态抽象是将不同状态按照一定的聚类规则或相似度度量方法[39]划分到不同的等价类中,每一个等价类便是一个抽象状态,其中的基状态都是相似的.

从马尔可夫决策过程(Markov decision process, MDPs)角度进行状态抽象是较早的一种做法. MDPs作为强化学习的理论依据,在MCTS中扮演着重要角色. MDPs是序贯决策的数学模型,用于在系统状态具有马尔可夫性质的环境中模拟智能体可实现的随机性策略与回报[40],其本质上是决策者(或智能体)通过采取具体动作,与现实环境(或计算机环境)发生交互,从而改变系统的状态并获得一定的回报奖励的循环过程.

MDPs将具体规划问题建模为五元组M,有

$$M=\langle S,A,P,R,\gamma\rangle.$$

其中:S为状态空间,A为动作空间,P为状态转移函数,R为奖励函数,γ为折扣因子[41].

基础MDPs建模的值函数$v(s)$表示为

$$v(s)=\sum_{a\in A}\pi(a|s)\Big(R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)v(s')\Big),\tag{1}$$

其中:π为策略函数,括号中为状态-动作值函数$Q(s,a)$的计算方式.

MDPs问题的求解本质上是求式(1)的最优解,即计算一个最优策略$\pi^*(a|s)$,使得在状态空间中的所有状态下,所选动作a都能使值函数最大化,最优值函数用$v^*(s)$和$Q^*(a|s)$表示.

MDPs容易从奖励函数、状态转移概率和值函数等方面进行抽象表示,Li等[42]在结合早期状态抽象方法的基础上,为MDPs提供了状态抽象的统一处理,他们认为并不是所有的抽象都同等重要,有效地抽象必须保留一些对解决原始MDPs问题至关重要的信息,在这一思想基础上,其将MDPs中的抽象分为如下5类,每一类都对应相应的重要信息,其中的基状态指相同抽象状态下的基础状态:

1) 模型无关的抽象$\varphi_{\rm model}$.

基状态在任何动作下都有相同的奖励函数值和状态转移函数值.

2) Q^π无关的抽象φ_{Q^π}.

基状态在任何策略下执行任意相同的动作都有相同的Q函数值.

3) Q^*无关的抽象φ_{Q^*}.

基状态在最优策略π^*下执行任意相同的动作都有相同的Q函数值.

4) a^*无关的抽象φ_{a^*}.

一定有一个最优动作a^*,使得每个基状态在该动作下都有最优且相同的Q函数值.

5) Π^*无关的抽象φ_{π^*}.

一定有一个最优动作a^*,使得每个基状态在该动作下都有最优但可不同的Q函数值.

Li等[42]还证明了上述5类抽象方法的多个性质,其中最重要的两个性质如下:

1) $\varphi_{\rm model} \succcurlyeq \varphi_{Q^\pi} \succcurlyeq \varphi_{Q^*} \succcurlyeq \varphi_{a^*} \succcurlyeq \varphi_{\pi^*}$. 其中运算符$\succcurlyeq$被定义为: 有两个抽象函数$\varphi_1$和$\varphi_2$,对于任意基状态$s_1$和$s_2$,当$\varphi_1(s_1) = \varphi_1(s_2)$成立时,若$\varphi_2(s_1) = \varphi_2(s_2)$也成立,则称抽象函数$\varphi_1$比抽象函数$\varphi_2$更细化,记为$\varphi_1 \succcurlyeq \varphi_2$.

2) 对于抽象方法$\varphi_{\rm model}$、φ_{Q^π}、φ_{Q^*}和φ_{a^*},其抽象最优策略π^*同样也是基状态下的最优策略,而抽象方法φ_{π^*}的抽象最优策略可能是基状态下的次优策略.

上述两个性质对5种抽象方法的抽象粒度进行了理论上的比较,$\varphi_{\rm model}$、φ_{Q^π}、φ_{Q^*}和φ_{a^*}的抽象粒度依次提高,但其抽象后的最优策略仍然可以作为基状态下的最优策略来使用,φ_{π^*}的抽象粒度是最高的,但其抽象后的最优策略可能在基状态下并非最优. 这两个性质也符合实际情况,即随着抽象粒度的提高,模型会失去一些原有的信息要素,这虽然可以提高问题的求解效率,但可能会导致失去最优解.

Li等[42]从MDPs角度将状态抽象的方法分成5类,本质上是聚类的思想. 虽然该分类研究的时间较早,但近年来许多智能决策中的抽象方法仍然以上述5类方法为依据. Hostetler等[43-45]提出一种$\langle p, q\rangle$划分方法,将上述5种抽象方法进行统一,通过设置不同的p值和q值达到不同的抽象目的,并从理论上证明了该抽象方法的值函数相比于基本MDPs方法具有有限上界的性能损失.

虽然Li等[42]的分类方法目前仍被使用,但由于其较强的约束性,很难真正将抽象方法对应到5类中的具体某一类. 目前,更多研究者通过设置聚类函数将状态汇聚到不同的类型,其基础很大程度上也是MDPs抽象,但与Li等[42]提出的分类方法相比更宽松.

Abel等[46]和Ravindran等[47]提出AH (approximate homomorphism)状态抽象,使用构建近似模型,将上述5类方法的抽象条件从相同转变为近似,从而将基状态更便捷地进行抽象. Sokota等[48]通过拒绝在树中添加相似状态,提出IRSA (iteratively refining state abstractions)方法. Jiang等[49]提出一种由UCT轨迹诱导的局部抽象方法AS-UCT (abstractions of state UCT),使得良好的抽象可以被更快地寻找到,具体使用采样轨迹来计算状态的近似同态,也符合φ_{Q^*}的一部分特性,并在奥赛罗上证明了抽象可以带来UCT的性能提升. Feyzabadi等[50]利用代价函数衡量状态的相似性和连通性,对某个状态计算其与相邻状态的代价函数,然后按照聚类规则合并状态,若代价函数为奖励函数,则其符合$\varphi_{\rm model}$的一部分特性. 基于该方法提出的HCMDP (hierarchical constrained MDP)方法保持了原始模型中有价值的信息,在仿真规划问题的实验中均显著优于非抽象方法. Baram等[51]使用kmeans聚类、层次聚类等方法,将状态抽象分类,集成了SMDPs (semi Markov decision process)[52]和AMDPs (aggregated Markov decision process)[53],提出SAMDPs (semi-aggregated Markov decision process)模型,在网格世界和雅达利游戏中证明了状态抽象的方法有更高的性能. Wu等[54]将状态变量分为无动作和基于动作两类,定义聚合复杂度的概念衡量不同状态变量的聚合程度,并在ACPES (aggregation cyber physical energy systems)中证明了无动作状态变量的聚合性能优于基于动作状态变量的集合. Choe等[55]通过转换表[56]将许多类似的状态聚合到一起,提出一个《炉石传说》中的通用规划框MUCD (modified upper confidence bound for directed acyclic graph),并对未来的改进提供了开放式基线. Chen等[57]将具有到达相似状态所需成本的状态划为一类,降低了值函数迭代更新的成本,符合$\varphi_{\rm model}$的一部分特征,提出的VIAA (value iteration with adaptive aggregation)方法在各种模拟环境上的数值实验证明了抽象方法的鲁棒性,特别是随着MDPs问题规模的增加,值函数迭代的成本越来越低. Xu等[58]在最新的研究中通过动态聚类,不断聚合和拆分状态,使得MCTS能够在抽象状态层面运行,提出的EMCTS (elastic MCTS)方法在不同复杂度的RTS场景中都能构建比基础MCTS

更小的搜索树.

需要注意的是,上述抽象方法将规划主体看作状态的一部分,因此聚合的状态本质上是不同时刻的整体环境,但在一些场景中,规划主体如果独立于外界环境,只对外界环境进行抽象,则会有更好的规划效果. 如图4所示,左侧为原始状态,右侧为将原始状态划分为若干个区域,每个区域代表一个抽象状态.

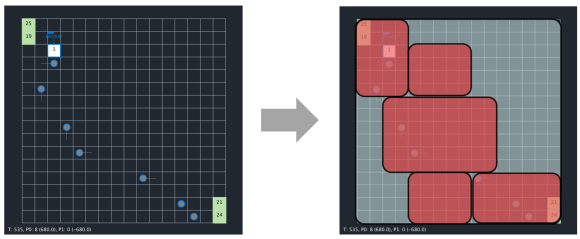

图 4　区域抽象示意图[59]

Ma 等[60-61] 在无人机多智能体规划中对路径点和智能体位置进行聚类,将其称为区域抽象,并在此基础上构造允许智能体根据区域抽象执行的抽象动作,提出了DDRP (dynamic domain reduction planning)方法. 仿真结果表明,在具有较大状态和动作空间的环境中,DDRP比使用标准MCTS方法有显著的改进. Ontanon[62] 提出的经典MCTS方法NaiveMCTS也使用了区域抽象的概念,通过在RTS游戏中划分区域,从同一时刻的角度对状态空间进行了抽象约简.

1.2　状态转换图

Bäckström 等[63] 从状态转换图 (state transition graphs, STG) 的角度构建了一个用于分析当前状态和动作抽象方法的统一框架,并且从规划角度建模了6种不同的抽象方法,其中包括3种状态抽象和3种动作抽象方法. 3种状态抽象的简要方式如下,实际操作中还要考虑动作的变幻:

1) ABSTRIPS 抽象 (abstrips-style abstraction, ABS).

ABSTRIPS抽象通过设置关键变量,将所有动作的前置条件限制为关键变量来进行抽象.

2) 变量投影 (variable projection, VP).

在ABS的基础上,去除抽象中的非关键变量,不保留基状态,完全使用高层抽象状态进行规划,这种抽象方法称为变量投影. ABS和VP方法的抽象过程如图5所示,v为关键变量,两者的不同之处在于是否保留非关键变量和基状态.

3) 变量域抽象 (variable-domain abstraction, VDA).

变量域抽象通过设置抽象域函数,将状态中的变量按照值域进行压缩,从而减少状态空间. VDA方法的抽象过程如图6所示,每个状态由(u,v)构成,抽象域函数为$h_u(0)=0, h_u(1)=h_u(2)=1, h_v(0)=h_v(1)=0, h_v(2)=2$,其中$h_u(0)=0$表示将基状态中$u=0$的状态映射到抽象状态$u=0$的状态中.

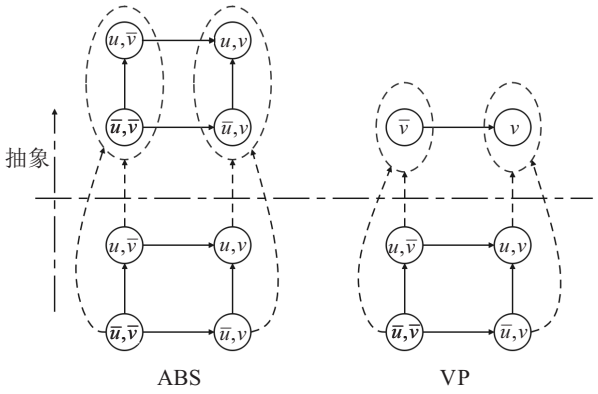

图 5　ABS和VP方法示意图[63]

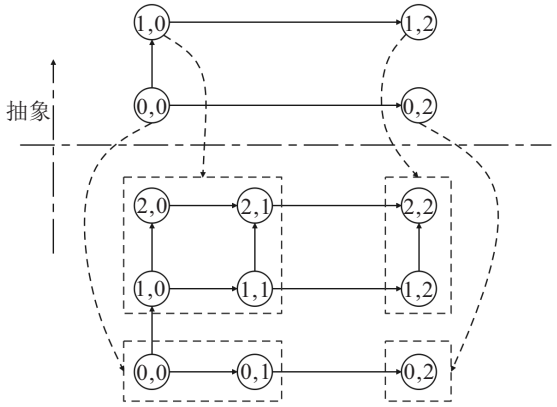

图 6　VDA方法示意图[63]

Bäckström 等[63] 从状态转换图的角度将状态抽象方法提炼为3类,最近也有学者从STG角度出发进行了研究. Nitii 等[64] 提出了混合关系域基于样本的规划器HYPE,通过去除不相关的事实,将具体状态概括为抽象状态,既符合φ_{model}的一部分特性,也符合VP抽象的一部分特性,本质是去除无效的状态(非关键变量的状态). Choe 等[55] 通过转换表[56] 提出通用规划框架MUCD,符合VDA抽象的一部分特性. Leurent 等[65] 将MCTS通过状态相似性转化成MCGS (Monte Carlo graph search),复用树中有相似属性的状态,其思想符合VP抽象的一部分特性,通过数值模拟表明MCGS在高维雅达利游戏中有较高的规划性能. Nashed 等[66] 通过设置关键变量k和h,判断当前状态与目标状态的可达性,将状态进行划分,提出的KHRA (K-H reach ability)方法具有ABS抽象的一部分特性,并在一组标准基线领域上证明了其优于几种强启发式方法.

1.3　神经网络

随着深度学习的快速发展,将神经网络与MCTS相结合已经越来越普遍,尤其是在AlphaGo被提出之

后,出现了一大批类似的方法.

AlphaGo提取围棋盘面的48个特征并将其编码为特征矩阵,然后通过卷积神经网络(convolutional neural network, CNN)使得人类提取的特征最终抽象为机器自我理解的抽象特征[12],使用策略网络替代MCTS中的树策略,使用价值网络替代MCTS中的模拟策略,大大降低了MCTS的搜索空间,最终在围棋中取得了史无前例的成就,并且在后期发展到不需要人类训练数据即可达到目的,即AlphaGo Zero[13]. Guo等[67]利用分层设计奖励的思想,使用CNN从原始状态中自动学习特征,并将神经网络用于MCTS的底层模拟,不仅对状态空间进行编码抽象,还减少了模拟过程中的搜索空间,提出的PGRD-DL(policy-gradient for reward design with deep learning)框架提高了基础MCTS在多个雅达利游戏中的性能. Tang等[68]受到AlphaGo的启发,使用CNN和MCTS预测五子棋中的落子胜率,将网络作为模拟策略的模型,提出的ADP-UCT(adaptive dynamic programming UCT)方法在五子棋AI对抗中取得了良好效果. Barriga等[69]将神经网络的输出动作看作高层动作,并将状态进行卷积,从而降低MCTS搜索空间,提出的CST(combined strategy and tactics)方法在小型RTS游戏中取得了一定的性能. Zhang等[70]使用高级策略网络指导MCTS的模拟过程,也是将状态空间编码为向量从而实现抽象,提出的DUCT-Sf+CNB(determinized UCT silverfish chance node bucketing)框架在《炉石传说》游戏击败了较强的AI—Silverfish[71]. Swiechowski等[72]将MCTS与监督学习相结合,所提出的MCTS-SL(MCTS-supervised learning)框架也使用类似的方法增强《炉石传说》AI的性能. Pinto等[73]同样使用神经网络对格斗游戏的状态进行卷积抽象,网络输出高层动作,从而减少MCTS的搜索空间,其提出的Mogakumono智能体可以击败当时最先进的格斗游戏AI. Baier等[74]使用神经网络建模人类玩家在牌类游戏中的行为,并将其作为MCTS的模拟策略,降低了MCTS的搜索空间,使得MCTS有一定的模仿人类的能力,所提出的B-MCTS(Bias-MCTS)方法在《Spades》牌类游戏中保持了强大的竞争性. Wu等[75]基于AlphaGo思想提出了BV-ML-VN(board evaluation multi labelled value network)框架,在具有动态帖目的围棋中取得了良好性能,并且可以拓展到雅达利游戏,这是AlphaGo做不到的. 与BV-ML-VN类似, Yang等[76]提出Shiro方法,不需要提前知道帖目信息即可训练围棋AI模型,并且支持任何大小的围棋棋盘,同时具有一定的复用能力. Chang等[77]在AlphaGo Zero的基础上,将MCTS的奖励值从原来的赢、平或者输改为连续值,即得到或失去多少分,所提出的Big-Win方法大大提高了6×6奥赛罗游戏AI的强度. Gao等[78]也对AlphaGo Zero进行了拓展,将原来的双头网络输出改为三头网络输出,在策略、状态价值的基础上增加了动作价值的网络输出,更好地辅助MCTS进行搜索,所提出的PV-MCTS-3HNN(policy value MCTS 3 head neural network)框架在《Hex》卡牌游戏中验证了三头神经网络比双头神经网络表现更好,并且明显超过当时《Hex》游戏中最强的智能体MoHex-CNN. Swiechowski等[79]通过检查函数和附加约束将状态编码为32位向量,使得MCTS可以在更一般的概念上运行,所提出的GMCTS(granular MCTS)方法可以在多粒度上完成搜索,便捷了RTS游戏领域的建模. Goodman[80]在《Hanabi》牌类游戏中利用对抗数据训练神经网络,并且将其作为MCTS的模拟策略,从而降低MCTS的搜索空间,所提出的RD-ISMCTS(Re determinizing information set MCTS)框架建立在解决不确定博弈方法ISMCTS[23]的基础上. Pierrot等[81]利用模块化和层次思想,借助AlphaGo Zero训练神经网络,提出了一种新的强化学习框架AlphaNPI(alpha neural programmer interpreters),并在任意数量塔底的汉诺塔问题中取得了较高的性能. Hu等[82]为了解决现代数据并行框架调度性能弱的问题,训练了一个神经网络指导MCTS的模拟策略,通过关注更有前途的分支,显著提高了搜索效率,所提出的Spear方法比传统的调度方法提升了20%的性能. Abel等[83]在Rate-Distortion理论、Blahut-Arimoto方法和Information Bottleneck方法的基础上开发了一种用于计算状态抽象的方法DIBS(deterministic information bottleneck for state abstraction),将状态抽象视为压缩的新形式,是使用一个具有专家策略的MDPs分布训练一个网络,使其与专家策略接近从而实现抽象,同时分析了Li等[42]提出的抽象方法的缺点,即捕获无损抽象必然会降低学习效率. Abel等[84]使用深度DRL学习的方法,用神经网络表示抽象状态和具体的抽象动作提出了$\emptyset$-RO($\emptyset$-relative options)方法,并从数学上计算了其误差的下界,在传统的4房间寻路问题上证明了其性能优于传统的Q-Learning方法. Silver所在的Deepmind团队在AlphaGo的基础上提出了MuZero框架[85],通过将基于树的搜索与学习模型相结合,在

一系列具有挑战性和视觉复杂的领域中实现了超强的性能,并且不需要任何动力模型知识,是AlphaGo框架在超大搜索环境下的超强提升,但问题在于训练需要超大规模的算力. Dockhorn等[33]对RTS游戏《Dragon Age》的状态直接进行编码,通过评估函数评估抽象过程的优劣,所提出的HCMCTS (hybrid considering MCTS)方法在不同地图上的实验都表明了其性能优于基本的MCTS方法.

随着大数据和算力的快速发展,使用神经网络进行状态抽象的方法已经越来越普遍. 受到AlphaGo系列方法的启发,神经网络辅助MCTS搜索的方法主要有3类:

1) 将策略网络作为MCTS中的树策略.

将良好的策略网络作为MCTS的tree policy,可以使MCTS在选择阶段避免大量无效的探索,从而降低搜索空间.

2) 将使用价值网络作为MCTS中的模拟策略.

将良好的价值网络作为MCTS的rollout policy,可以使MCTS在模拟阶段快速评估叶节点的初始值,而不需要随机模拟到结束状态,从而降低搜索空间.

3) 使用CNN对原始状态进行深度抽象.

将状态或状态中的某些特征编码为特征向量或特征矩阵,通过CNN自动提取高层特征,如图7所示,可以看作是使用神经网络进行状态抽象,虽然人类可能无法理解多层卷积之后的特征代表什么,但机器可以从高层卷积中学习到更聚焦的特征.

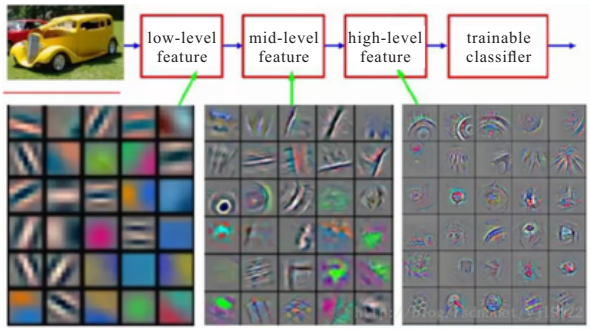

图7　CNN抽象示意图[86]

将上述具体方法按照神经网络辅助MCTS搜索的3种方法进行分类,如表1所示.

1.4　小　结

本节从状态聚类、状态转换图和神经网络3个角度对状态抽象技术及其在MCTS中的应用进行了分类和总结,图8对所介绍的技术和方法进行了直观地展示.

从图8可以看出,神经网络方法是目前研究较多的状态抽象技术,该类方法的发展大多源于AlphaGo系列方法的提出,但AlphaGo系列方法的框架在20世纪已经有所研究,因此该类方法的出现本质上源于近些年存储条件和计算能力的飞速提升. 此类方法虽然可以在部分决策问题中取得超越人类的性能,但也存在一定的缺陷,首先是网络模型对于领域的高度依赖使得此类方法难以复用;其次是训练代价过大导致其实用性较低;最后是神经网络可解释性问题导致决策可信度低.

状态聚类是一类理论基础比较强的抽象方法,通过设置聚类条件的宽松程度可以发现不同程度的抽象,其缺点在于高度的近似虽然能够简化问题,但会使搜索结果偏离最优值,因此需要衡量近似与最优之间的关系.

状态转换图是基于MCTS搜索结构的一类抽象方法,从图的角度考虑树节点的聚合和分离可以很大程度上对树进行剪枝,但缺点在于状态转换图的理论依据可能不符合MCTS的搜索过程,需要进一步研究两者的内在联系.

表1　神经网络辅助MCTS搜索分类表

方法	类别	方法	类别
AlphaGo (Zero)	1)、2)、3)	Big-Win	1)、2)、3)
PGRD-DL	2)、3)	PV-MCTS-3HNN	1)、2)、3)
ADP-UCT	2)	GMCTS	3)
CTS	3)	RD-ISMCTS	2)
DUCT-sf-CNB	2)、3)	AlphaNPI	1)、2)、3)
MCTS-SL	2)、3)	Spear	2)
Mogakumono	3)	DISB	1)2)
B-MCTS	2)	Ø-RO	3)
BV-ML-VN	1)、2)、3)	MuZero	1)、2)、3)
Shiro	1)、2)、3)	HCMCTS	3)

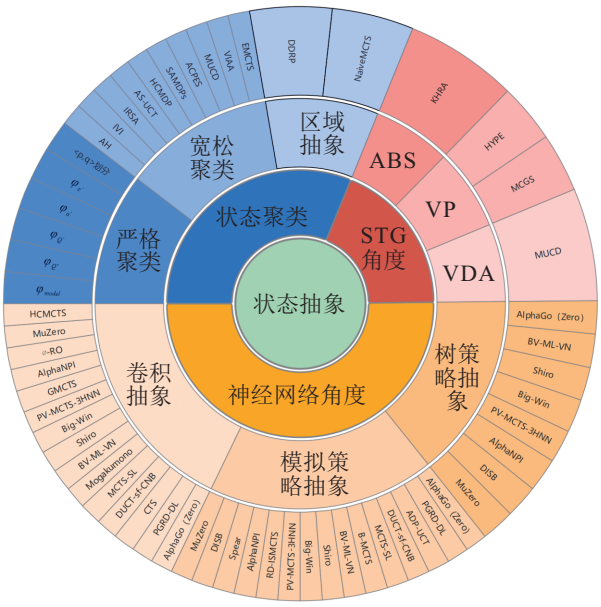

图8　状态抽象方法分类

2 动作抽象及其在MCTS中的应用

与状态抽象技术一样,动作抽象技术也被广泛应用于决策问题中. 本节主要将动作抽象技术分为3个大类,后续将从经典动作抽象、状态转化图和神经网络3个角度对动作抽象技术进行分类和总结,并介绍其在MCTS中的应用.

2.1 经典动作抽象

与状态抽象一样,从MDPs角度进行动作抽象是较早的一种做法,因为MDPs作为规划和强化学习的理论基础,很容易拓展到长时间的动作组合. 本节主要介绍MDPs基础上较为经典的几种动作抽象方法,以及具有聚类思想的机会事件桶方法.

1) 开局库(opening book, OB).

OB源自国际象棋AI的研究,在20世纪末,研究者发现重复地搜索会大大降低智能体的学习效率,因此将国际象棋的很多经典开局编码为动作组合(即OB). 通俗理解就是一个记忆库,将搜索或学习过程中见过的良好的行动序列记下来,可以认为是较早的动作抽象[87]. OB的出现使得国际象棋和早期围棋AI的性能在MCTS的辅助下得到快速提升[88]. 基于OB灵活性差的缺点,许多研究者提出了自适应和自学习的OB方法[89-90],并在简单博弈问题中有较好的应用.

OB虽然有一定的动作抽象思想,但其缺乏理论支撑,在21世纪后很大程度上被具有很强理论依据的选项(Option)理论所替代.

2) 选项(Option).

动作抽象最经典的方法Option, 由Sutton[52]于1999年提出,直到今天,Option理论依然被广泛用于MDPs建模中的动作抽象. Option建立在半马尔科夫决策SMDPs上[52],与普通的MDPs相比,SMDPs中状态转移函数被定义为联合概率分布$P(s',k|s,a)$,即在状态s中执行动作a、到达s'且经过k个时间步的概率. 图9能很好地展现MDPs、SMDPs与Option的关系.

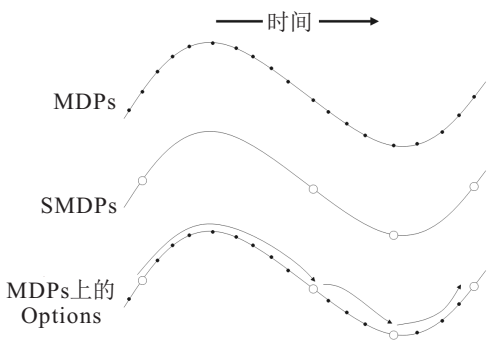

图9 MDPs、SMDPs与Option的关系[52]

一个Option可以用$o_{x\to y}$表示,即从抽象状态x到临近抽象状态y的映射,具体定义为三元组

$$o_{x\to y}=\langle \alpha,\pi,\beta\rangle.$$

其中:α为初始条件集,表示该Option只有状态满足α中的初始条件时才会被选择;π为局部策略,是S到底层动作空间A的映射;β为该Option的结束条件.

$o_{x\to y}$中的局部策略π是Option内部的底层动作选择策略,用于两个相邻基状态之间的状态转换,因此可以认为动作抽象对问题进行了分层求解,原本的总策略被划分为若干个局部策略,每个局部策略π_o由一个o来具体计算. 定义总策略Π是状态到Option集合的映射,即$\Pi:S\to O$,则进行动作抽象后的MDPs问题的分层解可以定义为$\xi=\{\Pi,\pi_{o1},\pi_{o2},\pi_{o3},\ldots\}$.

分层解ξ可以指导智能体选择具体的动作,由下列3个过程循环实现:首先,根据t时刻的状态s_t,依据总策略Π选择一个$o=\Pi(s_t)$;然后,智能体将依据o的局部策略π_o来选择具体的底层动作执行,直到在$t+k$时刻满足o的结束条件;最后,智能体依据总策略Π选择一个新的$o'=\Pi(s_{t+k})$继续执行动作.

对MDPs策略进行分层建模后,需要对其状态值函数(V函数)和状态-动作值函数(Q函数)进行修正,使其符合新的建模方法.

令$V^{\Pi}(s)$表示在状态为s时依据策略Π执行动作可以得到的期望回报,对应于基础MDPs中的$v(s)$,令$Q^{\Pi}(s,o)$表示在状态为s时执行o后再依据策略Π执行动作可以得到的期望回报,对应于基础MDPs中的$Q(s,a)$. 对式(1)进行修正,可以得到分层建模下的$V^{\Pi}(s)$与$Q^{\Pi}(s,o)$之间的关系为

$$V^{\Pi}(s)=\sum_{o\in O}\Pi(o|s)\Big(V^{\pi_o}(s)+\sum_{s'\in S}\gamma^k P(s',k|s,o)V^{\Pi}(s')\Big),\tag{2}$$

其中括号内为$Q^{\Pi}(s,o)$的计算方式. 可以看出,$V^{\pi_o}(s)$对应式(1)中的$R(s,a)$,即奖励值,但式(2)是总策略的值函数迭代过程,而Option并不是具体可执行的动作,因此需要用另一组局部值函数来迭代求解局部策略.

$V^{\pi_o}(s)$表示在状态为s时依据局部策略π_o执行动作可以得到的期望回报,迭代计算公式为

$$V^{\pi_0}(s)=\sum_{a\in A}\pi_0(a|s)\Big(R(s,a)+\gamma\sum_{s'\in S}P(s'|s,a)V^{\pi_0}(s')\Big),\tag{3}$$

其中括号内为$Q^{\pi_0}(s,a)$的计算方式.

从Option提出以来,大量研究者都将其思想

应用到具体的动作抽象中[91]. Menashe 等[92] 基于MCTS提出了一种新的基于模型的强化学习方法T-UCT (transition UCT), 使用宏动作跨越更大的状态空间, 并忽略了不太可能成功的轨迹, 从一定程度上解决了强化学习样本稀疏的问题. Barriga 等[10] 提出PS (puppet search) 方法, 将Option看作木偶的肢体, 定义关键点为木偶的关节, 通过移动肢体组合成木偶的不同形态, 从而实现搜索功能, 实验表明其在星际争霸AI对抗中减少了搜索空间, 使对抗性游戏的树搜索变得可行. Li 等[93] 基于Option提出CTA (core task abstraction) 框架, 学习RL中的相关转换函数, 将Option再次进行抽象, 相似的任务被划分到同一片段, 并共享相同的转移函数, 实验表明其在网格世界中的性能优于MAXQ方法. Waard 等[94] 将Option与MCTS相结合提出OMCTS (option MCTS) 方法, 并在具有高水平行动计划的视频游戏中表现优于纯MCTS. Barriga 等[69] 将Option作为高层神经网络的输出动作, 提出的CST方法在小型RTS游戏中获得了优于PS的性能. Baram 等[51] 将Option用于分析神经网络的策略, 使得网络具有一定的可解释性, 所提出的SAMDPs方法在雅达利游戏中获得了较好的性能. Moraes 等[22] 通过使用相同的Option减少动作搜索空间, 并基于纯MCTS提出了A3N (asymmetric action abstractions Naïve MCTS) 框架, 在小型RTS游戏中优于当时最先进的搜索方法. Kurzer 等[95] 提出一种异构环境中使用宏动作的自动驾驶车辆分散协同规划方法, 宏动作允许在多个时间步骤上进行拓展, 所提出的DeCoH-MCTS (decentralized cooperative hierachical) 框架能够在异构环境下通过学习到的宏动作实现有效的协同规划. Gabor 等[96] 提出一种自动生成Option的方法, 并与MCTS相结合提出SMCTS (subgoal MCTS) 方法, 其性能在网格世界和俄罗斯方块中好于纯MCTS. Dockhorn 等[33] 将RTS游戏《Dragon Age》的动作划分为6大类, 即6类Option, 并作为神经网络的输出, 所提出的HCMCTS方法性能优于基本的MCTS方法. Ouessai 等[97] 通过为智能体预先定义好一组启发式的脚本动作, 使得Option可以被参数化, 所提出的ParaMCTS方法和EvoPMCTS方法性能在小型RTS的多个游戏中优于当时一些最先进的智能体.

同Option方法相似, MCTS还可以与某些分层规划方法相结合, 即使用MCTS来辅助分层规划, 较为经典的是使用MCTS辅助层次任务网 (hierarchical task network, HTN) 进行规划, 这类方法统称为MCTS-HTN. HTN作为智能规划技术的重要组成部分, 通过对任务的分解和冲突的消解来寻求完成任务的可行方案, 其基本思想是将规划领域的特殊知识提取出来, 用于将该领域复杂抽象的任务递归地分解成越来越小的子任务, 直到分解后的子任务都可以通过特定的规划动作直接完成为止[98-99]. 从一定程度上看, HTN中的复合任务类似于一个Option, 而该复合任务的分解方法类似于Option中的策略.

Wichlacz 等[100] 提出将MCTS应用到HTN规划中, 并且在实验中验证使用MCTS作为HTN的搜索算法比使用传统的搜索算法或启发式搜索算法得到的规划结果更好, 或者能更快地得到规划结果. 在同时期, Shao 等[28] 也提出基于MCTS的HTN规划方法, 使用MCTS来为复合任务选择最佳的分解方法, 解决了HTN规划依赖于分解方法排列顺序的问题, 并且可以推广到行动结果不确定的规划问题上. Goldman[101] 也采用了将MCTS用于选择分解方法的思路, 并将其应用到部分可观察的MDPs (partially observable MDPs) 问题的HTN在线规划算法中.

3) MAXQ值函数分解.

MAXQ本质上是SMDPs的一个实例, 其将MDPs分层分解, 并且定义了分层策略$\Pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$, 每个子策略都对应一个MDPs[102-103].

MAXQ定义了值函数投影

$$V^{\Pi}(i,s) = V^{\Pi}(\pi_i(s),s) + \sum_{s',k}\gamma^k P_i(s',k|s,\pi_i(s))V^{\Pi}(i,s'), \quad (4)$$

其中$V^{\Pi}(i,s)$为使用分层策略Π, 在状态s下执行任务i (依据i所对应的策略π_i) 最终得到的奖励期望. 进一步可以将Q函数表示为

$$Q^{\Pi}(i,s,a) = V^{\Pi}(a,s) + \sum_{s',k}\gamma^k P_i(s',k|s,a)Q^{\Pi}(i,s',\Pi(s')). \quad (5)$$

其中: a为i的子任务; $Q^{\Pi}(i,s,a)$为在状态s下先执行i下的子任务a得到的奖励期望, 由执行子任务a的奖励期望$V^{\Pi}(a,s)$和沿着策略Π完成整个任务i的总奖励期望两部分组成, 后者定义为完成函数, 有

$$C^{\Pi}(i,s,a) = \sum_{s',k}\gamma^k P_i(s',k|s,a)Q^{\Pi}(i,s',\Pi(s')). \quad (6)$$

因此Q函数又可以表示为

$$Q^{\Pi}(i,s,a) = V^{\Pi}(a,s) + C^{\Pi}(i,s,a). \quad (7)$$

$V^{\Pi}(i,s)$根据i是否为原子任务, 可以表示为

$$V^{\Pi}(i,s)=\begin{cases}Q^{\Pi}(i,s,\pi_i(s)),\ i\text{为复合任务};\\ \sum\limits_{s'}P(s'|s,i)R(s'|s,i),\ i\text{为原子任务}.\end{cases}\tag{8}$$

至此,MAXQ的值函数分解可以理解为,若要计算顶层任务0的值函数,则通过分层策略Π可以不断地将任务分解为任务a_1、任务a_2、任务a_3,直到某个任务分解出来的首个子任务a_n为原子任务为止. $V^{\Pi}(0,s)$的分解计算公式如下:

$$V^{\Pi}(0,s)=V^{\Pi}(a_n,s)+C^{\Pi}(a_{n-1},s,a_n)+\ldots+C^{\Pi}(0,s,a_1),\tag{9}$$

$$V^{\Pi}(a_n,s)=\sum_{s'}P(s'|s,a_n)R(s'|s,a_n).\tag{10}$$

基于上述分解过程,通过采样轨迹递归更新子任务的完成函数即可学习到所有值函数.

同时,MAXQ的提出者还提出两种基于MAXQ的学习方法—MAXQ-0和MAXQ-Q,使MAXQ可以应用到更广泛的领域.

Vien等[104]在MAXQ的基础上使用多元密度方法与贝叶斯理论相结合,将宏动作添加到动作空间中一起学习,并给宏动作定义自身的价值函数,由此提出的BA-HMDP (Bayes-adaptive hierarchical MDP)框架在小型出租车打车问题中的性能优于纯MAXQ方法. Li等[105]基于MAXQ提出CSRL (context sensitive RL)框架,通过仿真对每个子任务进行评估,在不同层次之间发展知识共享和行动选择,解决了MAXQ中任务需要详细指定和多个类似任务必须分开学习的问题. Capobianco等[106]将MAXQ思想应用于策略生成的分层值函数迭代,使用HTN定义高层动作,所提出的HI-VAL (iterative learning of hierarchical value functions)方法在机器人抓取领域可以较好地探索不同状态[107].

4) 层次抽象机(hierarchies of abstract machine, HAMs).

HAMs由Parr等[108-109]提出,引入有限状态机概念表示MDPs状态空间中的区域策略,将层次任务过程转化为MDPs策略空间中的有限结构. 这些机器可以在层次结构控制器被调用时为MDPs任务生成动作,也可以实现一些不确定的行为,调用其他状态机,并在最后将控制权返回给调用实体[107].

HAMs的理论基础也是SMDPs模型,因此与Option和MAXQ相同,也可以认为是动作抽象的经典方法. Bai等[110-111]后续提出HAMs的改进方法HAMQ-INT,在使用HAM时自动查找内部转换,递归地缩短了计算Q值的时间[107].

5) 机会事件桶(chance event bucket, CEB).

与状态抽象类似,可以从聚类角度考虑动作抽象,将不同动作按照一定的聚类规则或聚类函数划分到不同的等价类中.

在MCTS中,动作聚类的方法主要通过CEB实现,其是指利用聚类规则或聚合函数,将相似的动作或具有相似结果的动作放在同一个桶里,同一个桶中的动作被称为一个机会事件,当MCTS过程中需要采样时,会直接采样机会事件桶,而不是采样所有的动作空间,从而降低动作空间的大小. Zhang等[70]提出的DUCT-Sf + CNB框架不仅使用神经网络抽象状态空间,而且使用机会事件减小搜索的分支因子,最终在《炉石传说》游戏击败了较强的AI — Silverfish[71].

需要注意的是,Option、MAXQ和HAMs等方法严格来说是“时间抽象”,是将不同时刻的动作进行高层抽象的行为,而CEB是在决策的同时刻将不同动作进行划分的行为.

2.2 状态转化图

Bäckström等[63]建模了3类不同的动作抽象方法,其简要的抽象方式如下,实际操作中还要考虑状态的变幻.

1) 删除冗余动作(removing redundant actions, RRA).

删除冗余动作即取消部分或全部多余的行动,从而抽象掉多余的信息,如图10左侧所示,动作b是冗余动作,因此可以删除.

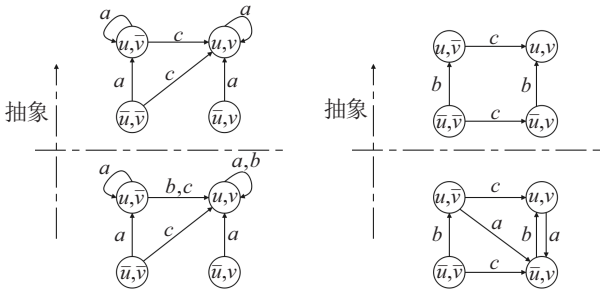

图10 RRA和IDL方法[63]

2) 忽略删除列表(ignoring delete lists, IDL).

忽略删除列表是指当某个状态变量达到规划目标时,导致其再次改变的动作都被删除,如图10右侧所示,假设规划目标是(u,v),当规划目标到达时,导致其改变的动作a被删除.

3) 基于地标的代理(landmark-based surrogates, LBS).

基于地标的代理通过设置一组地标变量(可以是原始变量的组合、拆分,也可以是新的状态变量)来

减少规划和搜索中出现的环状结构,其原理与IDL类似,即某个地标变量达到规划目标时,导致其再次改变的动作都被删除. LBS与IDL的不同之处在于,地标变量的设置使得原始的状态变量不需要作为删除的依据,更具有灵活性.

Baier等[112]将启发式动作约简算法Beam与MCTS相结合提出了BMCTS方法,通过预先评估节点的价值在搜索过程中只拓展有限数量的节点,符合RRA的一部分特性. Jiang等[49]提出的AS-UCT除了使用状态抽象外,还通过删除冗余动作进行动作抽象,符合RRA的一部分特性. Graf等[113-114]提出自适应的模拟策略MCTS-AP (MCTS with adaptive playouts),通过删除不必要的动作减少模拟过程的搜索空间,符合RRA的一部分特性. Ontañón[11]在NaiveMCTS方法的基础上,使用贝叶斯先验模型估计智能体的动作概率,从而减少搜索树中的节点数目,符合RRA的一部分特性,与纯MCTS方法相比,其提出的IMCTS (informed MCTS)方法在小型RTS中取得了更高的分数. Subramanian等[115]通过偏好来禁止某些动作,通过启发式函数在rollout过程中将一些不符合条件的动作删除,符合IDL和LBS的一部分特性,所提出的OCMCTS (option and constraint MCTS)方法在吃豆人游戏实验中取得了比纯MCTS更高的分数. Moraes等[22]提出的A3N框架也通过禁止某些动作来减少动作空间,符合IDL和LBS的一部分特性,最终在小型RTS游戏中优于当时最先进的搜索方法. Pinto等[73]在使用Option的基础上,通过在不同状态下过滤某些Option进一步减少动作空间,符合IDL和LBS的一部分特性,所提出的框架训练的AI智能体Mogakumono在格斗游戏中与当年的冠军智能体水平相当,但需要的专家知识更少. Cook[116]将动作按照是否可逆进行压缩从而实现动作抽象,符合IDL的一部分特性,提出的MCTS-R (MCTS with reversibility compression)方法在推箱子游戏中比纯MCTS以及几个变种更利于解决搜索中稀疏采样带来的影响. 一些研究者使用脚本组合和逐步不剪枝来减少过滤动作空间,符合LBS的一部分特性,所提出的P-MCTS (portfolio MCTS)[117]和P-MCTS-PU (P-MCTS progressive unpruning)[118]方法在RTS中取得了比普通MCTS更好的性能.

2.3　神经网络

前文已经介绍了大量使用神经网络辅助MCTS搜索的方法,并且将它们归纳为3个类型. 从本质上讲,使用神经网络辅助MCTS搜索,同时包含了状态抽象和动作抽象,因为无论使用策略网络替代tree policy,还是将价值网络作为rollout polict,都是从整体上降低MCTS的搜索空间,使得MCTS在树的构建过程中减少了大量的非必要探索. 因此本节不再重点介绍使用神经网络进行动作抽象的方法,但需要另外说明的是,当MCTS本身就在高层状态上构建搜索树时,策略网络的输出有时会是宏观动作,即类似于Option的方式. 例如OCMCTS方法、CST方法、GMCTS方法以及HCMCTS方法等,便是将提前设计好的或约束的Option作为策略网络的输出,并结合高层状态抽象减小MCTS的搜索空间.

2.4　小　结

本节从经典动作抽象、状态转换图和神经网络3个角度对动作抽象技术及其在MCTS中的应用进行了分类和总结,图11对介绍的技术和方法进行了直观地展示.

图11　动作抽象方法分类

在动作抽象中,除神经网络外,使用最多的是经典动作抽象中的Option方法,其优点在于可以将一段时间的动作组合成高层行动,从而使搜索过程变得简单,并且每个Option内部都满足MDPs条件,有很强的理论支撑. 缺点在于Option需要人工提前制定,并且需要为每个Option学习内部策略,从而使需要学习的参数量增加.

在许多关于抽象方法描述的文献中,并没有很好地对抽象方法与MCTS相结合的方式进行区分,本文将这种方式分为两大类:一类如图12所示,其中蓝色、红色和黄色节点分别表示根节点、原始节点和抽象节点,虚线框表示抽象过程,使用抽象技术对MCTS的同层状态节点进行等价划分,这种方法可以不考

虑对动作的抽象,也可以将动作考虑为状态抽象的一种条件,即状态-动作对抽象;另一种如图 13 所示,对 MCTS 的不同层状态节点进行等价划分,这种方法更多地考虑了动作抽象,例如 Option 的思想. 在实际应用中,两者的使用可以是结合的,但更多的抽象必然会引起较大的误差,即最终所选动作可能并非最优,因此如何选择抽象方法将是未来研究的方向之一.

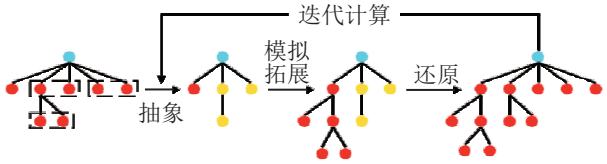

图 12 抽象方法与 MCTS 同层结合示意图

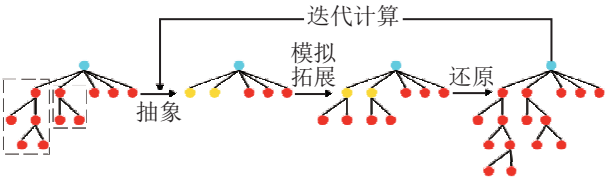

图 13 抽象方法与 MCTS 非同层结合示意图

3 状态-动作抽象

通过抽象方法减少搜索空间能够大大加快 MCTS 的探索速度,无论是状态抽象还是动作抽象,都有广泛的研究和应用,当然也会同时使用两种抽象方法提升 MCTS 性能[84,119-120],其中较为经典和通用的是在 POMCPs (partially observable Monte Carlo planning) 框架[121-122] 上提出的 H-POMCP (hierarchical POMCP) 方法[104,123] 和基于状态-动作对抽象的 ASAP-UCT 方法[124-126].

3.1 H-POMCP (hierarchical POMCP) 方法

H-POMCP 方法是基于 POMCP 框架的分层搜索方法,其整体结构很好地展示了如何使用 MCTS 来解决 POMDPs 问题.

POMDPs 将环境部分可观察的规划问题建模为七元组 M',有

$$M' = \langle S, A, Z, P, R, \Omega, \gamma \rangle.$$

其中: Z 为观测空间,指决策者或智能体实际观察到的状态的集合, Ω 为观测函数,用 $\Omega(z'|s,a)$ 表示,指在状态 s 时执行动作 a、观测到的环境状态为 z' 的概率. 在 s 下执行 a 的简单示意图如图 14 所示, s' 表示下一时刻系统的所有状态集合, z' 表示下一时刻决策者或智能体能观测到的状态集合.

为了更好地建模真实的部分可观察问题,POMDPs 引入了两个新的概念: 历史 (history) 和信念状态 (belief state). 历史 h 表示观测与动作的交替组合 $h = \{z_0, a_0, z_1, a_1, \ldots, a_{t-1}, z_t\}$; 信念状态 $b(s|h)$ 表

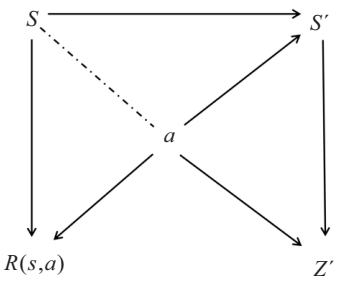

图 14 POMDPs 执行动作示意图

示当前历史 h 下实际状态为 s 的概率.

POMDPs 建模的问题可以等价地转换为定义在历史空间或信念空间上的 MDPs 问题,定义在历史空间上的 POMDPs 贝尔曼方程可以表示为

$$v(h) = \sum_{a\in A}\pi(a|h)\Big(R(h,a) + \gamma\sum_{z'\in Z}\Pr(z'|h,z)v(haz')\Big), \tag{11}$$

$$\pi(a|h) = \sum_{s\in S}\pi(a|s)b(s|h), \tag{12}$$

$$R(h,a) = \sum_{s\in S}R(s,a)b(s|h), \tag{13}$$

$$\Pr(z'|h,a) = \sum_{s\in S}\Omega(z'|s,a)b(s|h). \tag{14}$$

其中: $\Pr(z'|h,a)$ 表示在历史 h 时执行动作 a 得到的观察状态为 z' 的概率, haz' 表示在历史 h 时执行动作 a 得到观测状态为 z' 后组合而成的下一时刻的历史.

H-POMCP 方法的主要思路是将任务划分为原子任务和宏观任务,然后利用 POMCP 框架求解,但是由于宏观任务的存在,树节点上的任务完成有延迟 (需要多步执行),这导致 POMDPs 问题中蕴含了 SMDPs 问题,而 H-POMCP 方法则能较好地解决这一问题.

与 Shao 等[28] 的方法类似, H-POMCP 将 MCTS 用于选择宏观任务 i 在信念状态 b 下的最佳行为 (最佳行为可以是一个原子任务,也可以是一个宏观任务). 对于每个宏观任务 i,都有一棵单独为其服务的搜索树, MCTS 每次都从信念状态 b 中采样一个具体状态 s,并在 s 下进行模拟. 模拟 N 次后,一棵搜索树便构建完成,此时只需要按照 Q 值选择根节点的最大子节点,就是任务 i 需要完成的第一个最佳行为. 图 15 直观地展示了 H-POMCP 中搜索树的构建过程. 图 15 中蓝色节点表示信念状态,每个方框都表示一棵搜索树或子搜索树,方框的左上角表示该搜索树属于哪个任务. 可以看出,当信念状态为 b_1 时,任务 i 有两个可选行为 a_1 和 a_2, H-POMCP 会为 a_1 构建搜索树,返回在信念状态 b_1 下完成 a_1 的奖励 r,即 MAXQ 中的值函

数,并将信念状态更新为b_2. 进一步搜索b_2时需要完成任务i剩余部分的行为,并返回奖励R',即MAXQ中的完成函数. r与R'结合生成最终奖励R,用于更新节点或边上的N值和Q值.

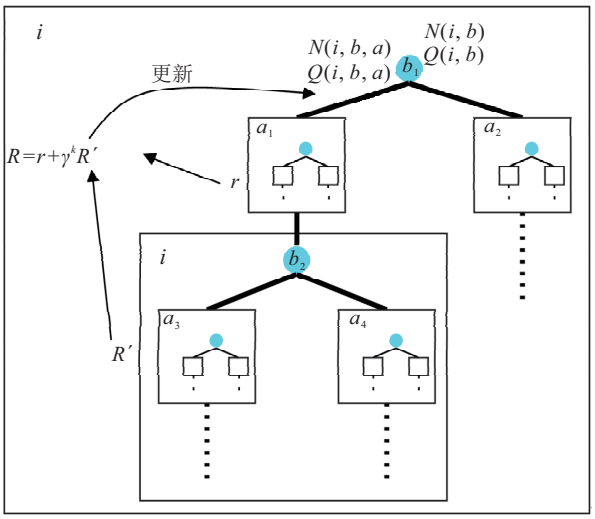

图 15　H-POMCP中搜索树构建过程

3.2　ASAP-UCT方法

Anand团队提出了状态-动作对(state-action pairs, SAPs)的抽象,用于发现规划或博弈中更多的对称性.

令$P_{\text{air}} = S \times A$为定义在MDPs上的SAPs集合,$\epsilon \subseteq S \times S$为状态对上的等价关系,$X$表示关系$\epsilon$上的等价类集合,$\mu_\epsilon : S \to X$为状态映射到对应等价类的等价函数,$e \subseteq P_{\text{air}} \times P_{\text{air}}$为SAPs上的等价关系,$U$为关系$e$上的等价类集合,$\mu_e : P \to U$为SAPs映射到对应等价类的等价函数. 有了上述声明,可以递归地定义状态等价和SAPs等价.

1) 状态等价.

状态s与s'是等价的$(\mu_\epsilon(s) = \mu_\epsilon(s'))$当且仅当

$$\exists a \in \text{Apply}(s) \text{ and } \exists a' \in \text{Apply}(s'),$$
$$\text{s.t.} \mu_e(s, a) = \mu_e(s', a').$$

其中Apply(s)表示返回状态s下所有的可行动作集合. 状态等价的前提是已经有SAPs等价和μ_e函数.

2) SAPs等价.

两个SAPs是等价的$(\mu_e(s, a) = \mu_e(s', a'))$当且仅当:

① $\forall x \in X, \sum\limits_{s_t \in S} \Pi[\mu_\epsilon(s_t) = x] P(s_t|s, a) = \sum\limits_{s_{t'} \in S} \Pi[\mu_\epsilon(s_{t'}) = x] P(s'_t|s', a')$;

② $R(s, a) = R(s', a')$.

其中条件①中的Π为指示函数. 上述条件说明,若要使两个SAPs等价,则其转换到的每个抽象状态的转换概率之和应该相同,同时有相同的奖励函数值. SAPs等价的前提是已经有了状态等价和μ_ϵ函数.

对于以目标状态为导向的无限步长MDPs,所有目标状态都被认为是等价的,即$\forall s, s' \in$ Goal,有$\mu_\epsilon(s) = \mu_\epsilon(s')$. 对于有限步长的情况,最大步长相同的目标状态被认为是等价的,即$\forall s, s' \in$ Goal且$\text{depth}(s) = \text{depth}(s')$,有$\mu_\epsilon(s) = \mu_\epsilon(s')$. 有了上述两条基本规则:迭代使用状态抽象和SAPs抽象的规则,可以不断拓展抽象集合,直到收敛.

上述框架称为ASAP(abstractions of state-action pairs)框架,可以揭示大量的对称性,从而极大程度上降低搜索空间.

在ASAP框架之前,已经有AS框架[49]和ASAM(abstractions of states with action mappings)框架[47]被提出,Anand团队证明了AS和ASAM均是ASAP的特殊情况,ASAP可以通过设置参数退化为AS或ASAM,ASAP可以找到AS和ASAM计算出的所有抽象.

以ASAP框架为基础,Anand团队提出了ASAP-UCT方法,使用MCTS中的4个步骤不断地在抽象树上进行拓展和回溯,从而使搜索树更加完备. 每当过了一段提前设置的时间后,ASAP-UCT便会将抽象树反向拓展为原始搜索树,并在原始搜索树上再次计算抽象树,然后继续在新抽象树上运行MCTS的4个步骤,直到用完决策时间,最终得到一个最优动作.

在原始搜索树基础上计算抽象树的过程是从原始搜索树的底层节点开始,通过迭代使用状态等价和SAPs等价两个条件,一直到根节点为止,最终生成抽象搜索树. 需要注意的是,根据Jiang等[49]的观点,还未拓展完全的状态节点被认为是等价的.

ASAP-UCT方法示意图如图16所示,其中蓝色、红色、黄色节点分别表示根节点、基状态节点和抽象状态节点.

在几个MDPs基准领域的实验评估表明,ASAP-UCT得到的策略比纯MCTS方法有很大的质量提升,但是ASAP-UCT方法在计算抽象和MCTS过程中是不相交的,若计算的抽象不准确,则会导致模拟资源的浪费. 为此,Anand团队进一步提出OGA-UCT(on-the-go abstractions UCT)方法[127],通过增量抽象计算,在原有搜索树上实时更新抽象结果,计算抽象时不再需要将抽象搜索树打平为原始搜索树,从而更好地利用计算资源. OGA-UCT还可以通过UCT搜索将抽象计算集中在最重要的地方,剔除某些抽象状态,进一步减少搜索空间.

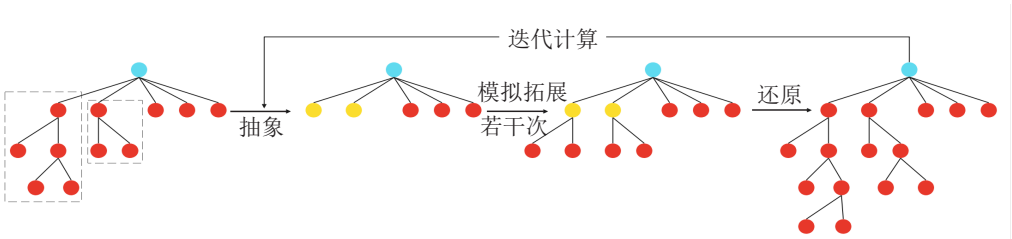

图 16 ASAP-UCT方法示意图

3.3 小 结

本节详细介绍了两种同时使用状态抽象和动作抽象的MCTS方法.

H-POMCP方法的优点在于:有很强的分层概念,将MCTS发展到了分层MCTS;有很强的理论基础,可以很好地与RL结合;可以解决规划动作具有持续性的问题,也可以推广至动作执行效果不确定的问题;可以进行在线规划,Lu等[128]提出的HMCTS-OP(HMCTS-based online planning)方法便是将H-POMCP应用到不对称的对抗环境中进行在线规划. H-POMCP方法的缺点在于:需要大量领域知识来构建任务之间的层次结构;分层结构虽然能够降低搜索空间,但效率有限,还需要进一步研究更先进的树剪枝技术;在多智能体规划中,H-POMCP缺乏智能体之间的通信能力,只能为单个智能体规划自身的行动.

ASAP-UCT方法的优点在于:将之前的AS与ASAM框架进行了统一,可以通过设置参数使ASAP转化为AS或ASAM,通用性较强;通过递归定义状态等价和SAPs等价,发现了大量MCTS中的抽象,使得决策问题可以最大程度地简化,同时最大程度保留决策结果的最优性. ASAP-UCT方法的缺点在于:与H-POMCP方法相比有一定的理论缺陷,因为ASAP-UCT中两个等价的状态-动作对可能有不等价的父节点,所以无法将其对应的MDPs模型进行抽象化,因为这会导致马尔科夫条件失效. 这也是情理之中的,因为大量的抽象虽然可以提升搜索的速度,但同时也会降低其结果的可靠性. 为了给ASAP框架增加理论支撑,Maillard等[129-130]将ASAP引入到具有平均奖励准则且不重置的MDPs问题中,提出的CUCRL2(Class Upper Confidence Reinforcement Learning)方法在产生大量等价类的基础上更具有理论依据.

4 未来展望

尽管抽象技术已经很大程度地缓解了MCTS中遇到的搜索空间爆炸问题,但还有一些尚未考虑的问题有待研究,现有的研究也存在一些不足的地方需要改进.

1) 动态选择抽象方法的研究.

目前已经有大量的抽象方法被提出,但如何选择最好的抽象方法,令MCTS能够最大程度地提升搜索效率并最低程度地降低搜索性能是一个值得研究的问题. 同时,在MCTS搜索中能否动态改变抽象方法,即随着搜索的进行,使用不同的抽象方法对状态或动作空间进行简化,从而合理利用规划资源,更好地衡量快速搜索和最优决策之间的标准,也需要进一步论证.

已经有一些研究者将该思想应用到具体问题中,例如Nashed等[66]讨论了如何选择不同的状态抽象来平衡时间与性能之间的关系;VIAA方法[57]动态地将具有相似成本的状态汇聚在一起;ASAP-UCT方法[124]和EMCTS方法[58]也有类似的思想,即不断聚合、拆分搜索树的节点来动态抽象. 虽然动态选择抽象方法的研究已经有了雏形,但缺乏整体框架和理论支撑,需要进一步研究具有理论依据的动态选择方式,真正使搜索时间与搜索性能之间达到动态平衡.

2) 提升抽象下决策的可信度.

MCTS与神经网络相结合是目前研究的热点,在MCTS的帮助下,深度神经网络已经在许多领域取得了优异成绩. 深度网络可解释性差是其应用中的通病,而许多现实中的决策问题,例如车辆行驶、家庭机器人、智能作战任务规划等,都需要决策算法给出为什么选择最终的决策动作,而不是单单给出最佳动作,因此提升抽象下决策的可信度,提高深度神经网络的可解释性,是未来的研究点之一.

目前,有研究者试图通过增加人类指令来指导深度神经网络在重要决策问题上的可信度,但还没有应用到具有MCTS的深度神经网络中,相信人类指令可以给抽象MCTS提供更高的可信度[131-132].

3) MCTS与分层深度强化学习的结合.

当前分层深度强化学习(HDRL)已经成为深度学习和强化学习中的研究热点[133-134],例如腾讯AI实验室通过HDRL训练的智能体在《王者荣耀》[135]游戏

和《我的世界》[136]游戏中分别取得了优秀的成绩. 分层的思想天生就自带了抽象的概念,如果将HDRL的思想与MCTS相结合,相信可以进一步增强决策的速度和准确度.

目前,有许多研究者已经对HDRL进行了研究[137-139],也有一些相对应的理论支撑[140-141]. 但将HDRL与MCTS结合的应用还比较少,未来的研究可以聚焦在两个方面:一是将类似于H-POMCP的框架作为核心,直接构建两个层次的深度网络,使用上层MCTS的结果指导下层MCTS,程恺等[142]尝试将人类知识作为搭建上层网络的依据,将数据作为搭建下层网络的依据,如果能构建知识牵引和数据驱动的HDRL方法,并将MCTS加入其中,相信可以获得更高的效能;二是以HDRL为核心,使用MCTS在各个层次进行数据采样,将获得的高质量数据用于网络的训练,而训练的网络又可以指导MCTS采集更高质量的数据[19,143],从而迭代提升模型的决策能力,相信这种迭代优化过程将会成为HDRL研究中最重要的方法之一.

4) 自动学习抽象过程.

上述讨论的抽象方法除了神经网络外都是通过人工制定抽象规则或抽象函数来对状态或动作进行约简,有时需要各个领域专家的介入,例如基于HTN的MCTS方法便需要提前编写规划中的领域知识. 为了缓解规划者的负担,能否与神经网络直接进行抽象类似,通过神经网络或别的方法,自动地学习抽象过程,可以从规划问题的规则本身,或从规划领域的现有数据中学习,通过神经网络判断两个状态或动作是否可以抽象到一起,从而使手工编写抽象规则变为自动生成抽象规则.

目前,已经有研究者开始这方面的研究,例如Silver等[144]为学习状态和动作抽象开发了一个新的框架,通过自动学习符号组件和神经组件达到抽象的目的,但该方面的研究仍属于少数,相信未来此思路可以有更好的研究和实际应用.

5) 抽象MCTS与其他技术的有机结合.

本文介绍的抽象MCTS方法的理论依据主要以MDPs为主,或是以神经网络技术为主,但在规划和决策问题中,还有其他重要的理论和方法,例如贝叶斯理论或博弈论中的方法,如何将抽象MCTS与其他更多的技术进行有机结合也具有较大的研究价值.

目前,有一些研究者将贝叶斯理论应用到抽象MCTS的研究中,例如Wang等[145]使用贝叶斯理论编码模型中对状态的先验知识,通过维持模型参数的概率分布表示模型参数的不确定性,并且提出MC-BRL (Monte Carlo Bayesian reinforcement learning)方法,用于解决POMDPs问题;Sharma等[146]允许计算风险敏感的贝叶斯自适应策略,以最佳方法权衡UCT中的探索和开发;管延霞等[147]则利用并行化的方式优化MCTS的搜索过程. 希望未来有更多的先进技术来支撑抽象MCTS的发展.

5 结 语

抽象技术作为人工智能研究中高效拓展决策的重要组成部分,通过将原本扁平的MCTS方法转化为块状或层次状的MCTS方法,减小局部决策空间,降低局部规划周期,从而较好地缓解MCTS中维度诅咒和历史诅咒两个问题. 本文从状态抽象和动作抽象两个方面对MCTS中的抽象应用进行了梳理和分类,并详细介绍了两种经典的抽象MCTS方法,最后预测了抽象MCTS未来可能的研究趋势,相信该技术的发展在未来可以更好地解决现实中的决策问题.

参考文献(References)

[1] Konidaris G. On the necessity of abstraction[J]. Current Opinion in Behavioral Sciences, 2019, 29: 1-7.

[2] Shanahan M, Mitchell M. Abstraction for deep reinforcement learning[J/OL]. 2022, arXiv: 2202.05839.

[3] Xu L, Perez-Liebana, D, Dockhorn A. Towards applicable state abstractions: A preview in strategy[C]. Proceedings of the 5th Multi-disciplinary Conference on Reinforcement Learning and Decision Making. Piscataway: IEEE, 2022: 1-7.

[4] Ho M K, Abel D, Griffiths T L, et al. The value of abstraction[J]. Current Opinion in Behavioral Sciences, 2019, 29: 111-116.

[5] Coulom R. Efficient selectivity and backup operators in Monte-Carlo tree search[C]. Computers and Games. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007: 72-83.

[6] Chen K H. Dynamic randomization and domain knowledge in Monte-Carlo tree search for go knowledge-based systems[J]. Knowledge-Based Systems, 2012, 34: 21-25.

[7] Baier H, Winands M H M. Time management for Monte Carlo tree search[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2016, 8(3): 301-314.

[8] Pepels T, Winands M H M, Lanctot M. Real-time Monte Carlo tree search in ms pac-man[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2014, 6(3): 245-257.

[9] Guo X, Singh S, Lee H, et al. Deep learning for realtime atari game play using of line Monte Carlo tree search planning[C]. Proceedings of the 2014 Neural Information Processing Systems. Piscataway: IEEE, 2014: 3338-3346.

[10] Barriga N, Stanescu M, Buro M. Puppet search: Enhancing scripted behavior by look-ahead search with applications to real-time strategy games[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2021, 11(1): 9-15.

[11] Ontañón S. Informed Monte Carlo tree search for real-time strategy games[C]. IEEE Conference on Computational Intelligence and Games. Santorini, 2017: 1-8.

[12] 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.

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

[14] wiechowski M, Godlewski K, Sawicki B, et al. Monte Carlo tree search: A review of recent modifications and applications[J]. Artificial Intelligence Review, 2023, 56(3): 2497-2562.

[15] Sironi C F, Winands M H M. Comparing randomization strategies for search-control parameters in Monte-Carlo tree search[C]. IEEE Conference on Games. London, 2019: 1-8.

[16] 梁思立, 姜桂飞, 陈泰劼, 等. 基于蒙特卡洛树搜索的通用博弈系统的构建与优化研究[J]. 数据与计算发展前沿, 2022(3): 66-77.
(Liang S L, Jiang G F, Chen T J, et al. Optimizing Monte Carlo tree search for general game playing[J]. Frontiers of Data & Computing, 2022(3): 66-77.)

[17] Joppen T, Moneke M U, Schröder N, et al. Informed hybrid game tree search for general video game playing[J]. IEEE Transactions on Games, 2018, 10(1): 78-90.

[18] Guerrero-Romero C, Louis A, Perez-Liebana D. Beyond playing to win: Diversifying heuristics for GVGAI[C]. IEEE Conference on Computational Intelligence and Games. New York, 2017: 118-125.

[19] Kartal B, Hernandez-Leal P, Taylor M E. Action guidance with MCTS for deep reinforcement learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2019, 15(1): 153-159.

[20] Zhou H W, Gong Y C, Mugrai L, et al. A hybrid search agent in pommerman[C]. Proceedings of the 13th International Conference on the Foundations of Digital Games. Malmö, 2018: 1-4.

[21] Preuss M, Risi S. A games industry perspective on recent game AI developments[J]. KI-Künstliche Intelligenz, 2020, 34(1): 81-83.

[22] Moraes R, Mariño J, Lelis L, et al. Action abstractions for combinatorial multi-armed bandit tree search[J]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2018, 14(1): 74-80.

[23] Cowling P I, Powley E J, Whitehouse D. Information set Monte Carlo tree search[J]. IEEE Transactions on Computational Intelligence and AI in Games, 2012, 4(2): 120-143.

[24] 殷彤辉. 基于MCTS与DRQN的多人非完备信息博弈研究[D]. 北京: 北京交通大学, 2021.
(Yin T H. Research on multi-player incomplete information game based on MCTS and DRQN[D]. Beijing: Beijing Jiaotong University, 2021.)

[25] 肖凌峰. 基于注意力机制和蒙特卡洛树搜索的二打一扑克博弈算法研究[D]. 西安: 西安电子科技大学, 2022.
(Xiao L F. Research on two-to-one poker game algorithm based on attention mechanism and Monte Carlo tree search[D]. Xi'an: Xidian University, 2022.)

[26] Horcas J M, Galindo J A, Heradio R, et al. A Monte Carlo tree search conceptual framework for feature model analyses[J]. Journal of Systems and Software, 2023, 195: 111551.

[27] Li W J, Cao J, Zhou B, et al. Multi-cloud service provision based on decision tree and two-layer Restricted Monte Carlo tree search[J]. Internet of Things, 2023, 22: 100751.

[28] Shao T H, Zhang H J, Cheng K, et al. The hierarchical task network planning method based on Monte Carlo tree search[J]. Knowledge-Based Systems, 2021, 225: 107067.

[29] Jooken J, Leyman P, Wauters T, et al. Exploring search space trees using an adapted version of Monte Carlo tree search for combinatorial optimization problems[J]. Computers & Operations Research, 2023, 150: 106070.

[30] Liu P S, Zhou J Z, Lv J C. Exploring the first-move balance point of Go-Moku based on reinforcement learning and Monte Carlo tree search[J]. Knowledge-Based Systems, 2023, 261: 110207.

[31] Takada K, Iizuka H, Yamamoto M. Reinforcement learning to create value and policy functions using minimax tree search in hex[J]. IEEE Transactions on Games, 2020, 12(1): 63-73.

[32] Dipendra M, Mikael H, Akshay K, et al. Kinematic state abstraction and provably efficient rich-observation reinforcement learning[C]. Proceedings of the 2020 International Conference on Machine Learning. Vienna, 2020: 6961-6971.

[33] Dockhorn A, Hurtado-Grueso J, Jeurissen D, et al. Game state and action abstracting Monte Carlo tree search for general strategy game-playing[C]. IEEE Conference on Games. Copenhagen, 2021: 1-8.

[34] Naman S, Siddharth S. Using deep learning to bootstrap abstractions for hierarchical robot planning[C]. Proceedings of the 21st International Conference on Autonomous Agents and Multi-agent Systems. Virtual, 2022: 1183-1191.

[35] Kocsis L, Szepesvári C. Bandit based Monte-Carlo planning[C]. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006: 282-293.

[36] Oliehoek F, Witwicki S, Kaelbling L. A sufficient statistic for influence in structured multiagent environments[J].

Journal of Artificial Intelligence Research, 2021, 70: 789-870.

[37] Shah D, Xu P, Lu Y, et al. Value function spaces: Skill-centric state abstractions for long-horizon reasoning[J/OL]. 2021, arXiv: 2111.03189.

[38] Pol E, Hoof H, Oliehoek F A, et al. Multi-agent MDP homomorphic networks[C]. Proceedings of the 2022 International Conference on Learning Representations. Virtual, 2022: 1-8.

[39] García J, Visús Á, Fernández F. A taxonomy for similarity metrics between Markov decision processes[J]. Machine Learning, 2022, 111(11): 4217-4247.

[40] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. New York: MIT Press, 2018: 1-3.

[41] Bellman R E. Dynamic programming[M]. Princeton: Princeton University Press, 1957.

[42] Li L, Walsh T J, Littman M. Towards a unified theory of state abstraction for MDPs[C]. Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics. Fort Lauderdale: Springer, 2006: 1-10.

[43] Hostetler J, Fern A, Dietterich T. State aggregation in Monte Carlo tree search[C]. Proceedings of the AAAI Conference on Artificial Intelligence, 2014, 28(1): 2446-2452.

[44] Hostetler J. Monte Carlo tree search with fixed and adaptive state abstractions[D]. Corvallis: Oregon State University, 2017.

[45] Hostetler J, Fern A, Dietterich T. Sample-based tree search with fixed and adaptive state abstractions[J]. Journal of Artificial Intelligence Research, 2017, 60: 717-777.

[46] Abel D, Hershkowitz D E, Littman M L. Near optimal behavior via approximate state abstraction[C]. Proceedings of the 33rd Internation Conference on Machine Learing. Piscataway: IEEE, 2016: 2915-2923.

[47] Ravindran B, Barto A G. Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes[J]. International Journal of Mineral Processing, 2004, DOI: 10.1016/S0301-7516(03)00069-3.

[48] Sokota S, Ho C, Ahmad Z, et al. Monte Carlo tree search with iteratively refining state abstractions[C]. Proceedings of the 35th Conference on Neural Information Processing Systems. Virtual, 2021, 34: 18698-18709.

[49] Jiang N, Singh S, Lewis R. Improving UCT planning via approximate homomorphisms[C]. Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems. France: ACM, 2014: 1289-1296.

[50] Feyzabadi S, Carpin S. Planning using hierarchical constrained Markov decision processes[J]. Autonomous Robots, 2017, 41(8): 1589-1607.

[51] Baram N, Zahavy T. Spatio-temporal abstractions in reinforcement learning through neural encoding[C]. Proceedings of the 5th International Conference on Learning Representations. France, 2017.

[52] Sutton R S, Precup D, Singh S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning[J]. Artificial Intelligence, 1999, 112(1/2): 181-211.

[53] Moore A W. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces[C]. Machine Learning Proceedings 1991. Amsterdam: Elsevier, 1991: 333-337.

[54] Wu J J, Jia Q S. On state aggregation in a class of cyber physical energy systems[C]. The 37th Chinese Control Conference. Wuhan, 2018: 6160-6165.

[55] Choe J S B, Kim J K. Enhancing Monte Carlo tree search for playing hearthstone[C]. IEEE Conference on Games. London, 2019: 1-7.

[56] Childs B E, Brodeur J H, Kocsis L. Transpositions and move groups in Monte Carlo tree search[C]. IEEE Symposium on Computational Intelligence and Games. Perth, 2009: 389-395.

[57] Chen G T, Gaebler J D, Peng M, et al. An adaptive state aggregation algorithm for Markov decision processes[J/OL]. 2021, arXiv: 2107.11053.

[58] Xu L J, Hurtado-Grueso J, Jeurissen D, et al. Elastic Monte Carlo tree search with state abstraction for strategy game playing[C]. IEEE Conference on Games. Beijing, 2022: 369-376.

[59] Xu L, Perez-Liebana D, Dockhorn A. Towards applicable state abstractions: A preview in strategy games[C]. Proceedings of the 2022 Multi-disciplinary Conference on Reinforcement Learning and Decision Making. Piscataway: IEEE, 2022: 1-7.

[60] Ma A, Ouimet M, Cortés J. Dynamic domain reduction for multi-agent planning[C]. International Symposium on Multi-Robot and Multi-Agent Systems. Los Angeles, 2018: 142-149.

[61] Ma A, Ouimet M, Cortés J. Hierarchical reinforcement learning via dynamic subspace search for multi-agent planning[J].Autonomous Robots, 2020, 44(3/4): 485-503.

[62] Ontanon S. The combinatorial multi-armed bandit problem and its application to real-time strategy games[C]. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. New York: AAAI, 2023: 58-64.

[63] Bäckström C, Jonsson P. A framework for analysing state-abstraction methods[J]. Artificial Intelligence, 2022, 302: 103608.

[64] Nitti D, Belle V, Laet T, et al. Planning in hybrid relational MDPs[J]. Machine Learning, 2017, 106(12): 1905-1932.

[65] Leurent E, Maillard O A. Monte-Carlo graph search: The value of merging similar states[C]. Proceedings of the 12th Asian Conference on Machine Learning. Virtual, 2020, 129: 577-592.

[66] Nashed S B, Svegliato J, Bhatia A, et al. Selecting the partial state abstractions of MDPs: A metareasoning

approach with deep reinforcement learning[C]. IEEE/RSJ International Conference on Intelligent Robots and Systems. Kyoto, 2022: 11665-11670.

[67] Guo X X, Singh S, Lewis R, et al. Deep learning for reward design to improve Monte Carlo tree search in ATARI games[J/OL]. 2016, arXiv: 1604.07095.

[68] Tang Z T, Zhao D B, Shao K, et al. ADP with MCTS algorithm for gomoku[C]. IEEE Symposium Series on Computational Intelligence. Athens, 2017: 1-7.

[69] Barriga N A, Stanescu M, Buro M. Combining strategic learning and tactical search in real-time strategy games[C]. Proceedings of the 30th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Little Cottonwood Canyon, 2017: 9-15.

[70] Zhang S Y, Buro M. Improving hearthstone AI by learning high-level rollout policies and bucketing chance node events[C]. IEEE Conference on Computational Intelligence and Games. New York, 2017: 309-316.

[71] noHero123, silverfish[DB/OL]. (2016-01-07) [2023-05-10]. https://github.com/noHero123/silverfish.

[72] Swiechowski M, Tajmajer T, Janusz A. Improving hearthstone AI by combining MCTS and supervised learning algorithms[C]. IEEE Conference on Computational Intelligence and Games. Maastricht, 2018: 1-8.

[73] Pinto I P, Coutinho L R. Hierarchical reinforcement learning with Monte Carlo tree search in computer fighting game[J]. IEEE Transactions on Games, 2019, 11(3): 290-295.

[74] Baier H, Sattaur A, Powley E J, et al. Emulating human play in a leading mobile card game[J]. IEEE Transactions on Games, 2019, 11(4): 386-395.

[75] Wu T R, Wu I C, Chen G W, et al. Multilabeled value networks for computer go[J]. IEEE Transactions on Games, 2018, 10(4): 378-389.

[76] Yang B H, Wang L, Lu H, et al. Learning the game of go by scalable network without prior knowledge of Komi[J]. IEEE Transactions on Games, 2020, 12(2): 187-198.

[77] Chang N Y, Chen C H, Lin S S, et al. The big win strategy on multi-value network: An improvement over AlphaZero approach for 6x6 Othello[C]. Proceedings of the 2018 International Conference on Machine Learning and Machine Intelligence. Ha Noi, 2018: 78-81.

[78] Gao C, Müller M, Hayward R. Three-head neural network architecture for Monte Carlo tree search[C]. Proceedings of the 27th International Joint Conference on Artificial Intelligence. Sweden, 2018: 3762-3768.

[79] Swiechowski M, Slzak D. Granular games in real-time environment[C]. IEEE International Conference on Data Mining Workshops. Singapore, 2019: 462-469.

[80] Goodman J. Re-determinizing information set Monte Carlo tree search in hanabi[J/OL]. 2019, arXiv: 1902.06075.

[81] Pierrot T, Ligner G, Reed S, et al. Learning compositional neural programs with recursive tree search and planning[J/OL]. 2019, arXiv: 1905.12941.

[82] Hu Z M, Tu J, Li B C. Spear: optimized dependency-aware task scheduling with deep reinforcement learning[C]. IEEE 39th International Conference on Distributed Computing Systems. Dallas, 2019: 2037-2046.

[83] Abel D, Arumugam D, Asadi K, et al. State abstraction as compression in apprenticeship learning[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 3134-3142.

[84] Abel D, Umbanhowar N, Khetarpal K, et al. Value preserving state-action abstractions[C]. Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics. Virtual, 2020: 1639-1650.

[85] Schrittwieser J, Antonoglou I, Hubert T, et al. Mastering Atari, go, chess and shogi by planning with a learned model[J]. Nature, 2020, 588(7839): 604-609.

[86] Zeiler M D, Fergus R. Visualizing and understanding convolutional networks[C]. Computer Vision — ECCV 2014. Cham: Springer International Publishing, 2014: 818-833.

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

[88] Chaslot G M, Hoock J B, Perez J, et al. Meta Monte Carlo tree search for automatic opening book generation[C]. Proceedings of the 21st International Joint Conference on Artificial Intelligence Workshop on General Intelligence in Game Playing Agents. New York: AAAI, 2009: 7-12.

[89] Nagashima J, Hashimoto T, Iida H. Self-playing-based opening book tuning[J]. New Mathematics and Natural Computation, 2006, 2(2): 183-194.

[90] Gaudel R, Hoock J B, Pérez J, et al. A principled method for exploiting opening books[C]. Computers and Games. Berlin, Heidelberg: Springer, 2011: 136-144.

[91] Dobre M S. Low-resource learning in complex games[D]. Edinburgh: University of Edinburgh, 2019.

[92] Menashe J, Stone P. Monte Carlo hierarchical model learning[C]. Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. Turkey, 2015: 771-779.

[93] Li Z R, Narayan A, Leong T Y. A core task abstraction approach to hierarchical reinforcement learning [C]. Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems. Singapore, 2016: 1411-1412.

[94] de Waard M, Roijers D M, Bakkes S C J. Monte Carlo tree search with options for general video game playing[C]. IEEE Conference on Computational Intelligence and Games. Santorini, 2017: 1-8.

[95] Kurzer K, Zhou C Y, Marius Zöllner J. Decentralized cooperative planning for automated vehicles with hierarchical Monte Carlo tree search[C]. IEEE Intelligent Vehicles Symposium. Changshu, 2018: 529-536.

[96] Gabor T, Peter J, Phan T, et al. Subgoal-based temporal abstraction in Monte-Carlo tree search[C]. Proceedings of the 28th International Joint Conference on Artificial

Intelligence. Macao, 2019: 5562−5568.

[97] Ouessai A, Salem M, Mora A M. Evolving action pre-selection parameters for MCTS in real-time strategy games[J]. Entertainment Computing, 2022, 42: 100493.

[98] 王红卫, 刘典, 赵鹏, 等. 不确定层次任务网络规划研究综述[J]. 自动化学报, 2016, 42(5): 655-667.
(Wang H W, Liu D, Zhao P, et al. Review on hierarchical task network planning under uncertainty[J]. Acta Automatica Sinica, 2016, 42(5): 655-667.)

[99] 邵天浩, 张宏军, 程恺, 等. 层次任务网络中的重新规划研究综述[J]. 系统工程与电子技术, 2020, 42(12): 2833-2846.
(Shao T H, Zhang H J, Cheng K, et al. Review of replanning in hierarchical task network[J]. Systems Engineering and Electronics, 2020, 42(12): 2833-2846.)

[100] Wichlacz J, Höller D, Torralba A, et al. Applying Monte-Carlo tree search in HTN planning[J]. Proceedings of the International Symposium on Combinatorial Search, 2021, 11(1): 82-90.

[101] Goldman R P. Solving POMDPs online through HTN planning and Monte Carlo tree search[C]. Proceedings of the 4th International Conference on Automated Planning and Scheduling Workshop on Hierarchical Planning. Beijing, 2021: 57-61.

[102] Dietterich T G. The MAXQ method for hierarchical reinforcement learning[C]. Proceedings of the 15th International Conference on Machine Learning. New York, 1998: 118-126.

[103] Dietterich T G. Hierarchical reinforcement learning with the MAXQ value function decomposition[J]. Journal of Artificial Intelligence Research, 2000, 13: 227-303.

[104] Vien N A, Lee S, Chung T. Bayes-adaptive hierarchical MDPs[J]. Applied Intelligence, 2016, 45(1): 112-126.

[105] Li Z R, Narayan A, Leong T Y. An efficient approach to model-based hierarchical reinforcement learning[C]. Proceedings of the AAAI Conference on Artificial Intelligence. Argentina: AAAI, 2017: 3583-3589.

[106] Capobianco R, Riccio F, Nardi D. HI-VAL: Iterative learning of hierarchical value functions for policy generation[C]. Proceedings of the International Conference on Intelligent Autonomous Systems, Springer, 2018: 414-427.

[107] Lu L N, Gu X Q, Zhang W P, et al. Research on learning method based on hierarchical decomposition[C]. Chinese Automation Congress. Hangzhou, 2020: 5413-5418.

[108] Parr R, Russell S. Reinforcement learning with hierarchies of machines[C]. Proceedings of the 10th International Conference on Neural Information Processing Systems. New York: MIT, 1997: 1043-1049.

[109] Parr R E. Hierarchical control and learning for Markov decision processes[M]. Berkeley: University of California, 1998.

[110] Bai A J, Russell S. Efficient reinforcement learning with hierarchies of machines by leveraging internal transitions[C]. Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, 2017: 1418-1424.

[111] Bai A, Russell S. Speeding Up HAM learning with internal transitions[C]. Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making. Michigan, 2017.

[112] Baier H, Winands M H M. Monte-Carlo tree search and minimax hybrids with heuristic evaluation functions[C]. IEEE Conference on Computational Intelligence and Games. Piscataway: IEEE, 2012: 227-233.

[113] Graf T, Platzner M. Adaptive playouts in Monte-Carlo tree search with policy-gradient reinforcement learning[C]. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015: 1-11.

[114] Graf T, Platzner M. Adaptive playouts for online learning of policies during Monte Carlo tree search[J]. Theoretical Computer Science, 2016, 644(6): 53-62.

[115] Subramanian K, Thomaz A. Efficient exploration in Monte Carlo tree search using human action abstractions[C]. Proceedings of the 30th Conference on Neural Information Processing Systems. Spain, 2016: 1-10.

[116] Cook M. Monte Carlo tree search with reversibility compression[C]. IEEE Conference on Games. Copenhagen, 2021: 1-8.

[117] Dockhorn A, Hurtado-Grueso J, Jeurissen D, et al. Portfolio search and optimization for general strategy game-playing[C]. IEEE Congress on Evolutionary Computation. Kraków, 2021: 2085-2092.

[118] Perez-Liebana D, Guerrero-Romero C, Dockhorn A, et al. Generating diverse and competitive play-styles for strategy games[C]. IEEE Conference on Games. Copenhagen, 2021: 1-8.

[119] Rosenfeld A, Taylor M E, Kraus S. Speeding up tabular reinforcement learning using state-action similarities[C]. Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. Brazil, 2017: 1722-1724.

[120] Presented A D. Efficient probabilistic reasoning using partial state-space exploration[D]. Amherst: University of Massachusetts Amherst, 2019.

[121] Silver D, Veness J. Monte-Carlo planning in large POMDPs[C]. Proceedings of the 24th Annual Conference on Neural Information Processing Systems. Piscataway: IEEE, 2010: 1-9.

[122] Bai A, Srivastava S, Russell S. Markovian state and action abstractions for MDPs via hierarchical MCTS[C]. Proceedings of the 25th International Joint Conference on Artificial Intelligence. Argentina: AAAI, 2016: 3029-3037.

[123] Vien N A, Toussaint M. Hierarchical Monte-Carlo planning[J]. Proceedings of the AAAI Conference on Artificial Intelligence. New York: AAAI, 2015: 3613-3619.

[124] Anand A, Grover A, Singla P, et al. ASAP-UCT: Abstraction of state-action Pairs in UCT[C]. Proceedings of the 24th International Joint Conference on Artificial

Intelligence. Argentina: AAAI, 2015: 1509-1515.

[125] Anand A, Grover A, Mausam, et al. A novel abstraction framework for online planning[C]. Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. Turkey, 2015: 1901-1902.

[126] Anand A. Lifting techniques for sequential decision making and probabilistic inference[C]. Proceedings of the 25th International Joint Conference on Artificial Intelligence. Argentina: AAAI, 2016: 3972-3973.

[127] Anand A, Noothigattu R, Mausam, et al. OGA-UCT: On-the-go abstractions in UCT[C]. Proceedings of the International Conference on Automated Planning and Scheduling. London, 2016: 29-37.

[128] Lu L N, Zhang W P, Gu X Q, et al. HMCTS-OP: Hierarchical MCTS based online planning in the asymmetric adversarial environment[J]. Symmetry, 2020, 12(5): 719.

[129] Maillard O A, Asadi M. Upper confidence reinforcement learning exploiting state-action equivalence[Z]. HAL-01945034, 2018.

[130] Asadi M, Talebi M S, Bourel H, et al. Model-based reinforcement learning exploiting state-action equivalence[J/OL]. 2019, arXiv: 1910.04077.

[131] Chen V, Gupta A, Marino K. Ask your humans: Using human instructions to improve generalization in reinforcement learning[C]. Proceedings of the 2021 International Conference on Learning Representations. Virtual, 2021: 1-22.

[132] Xu S S, Wang H J, Wu Y. Grounded reinforcement learning: Learning to win the game under human commands[C]. Proceedings of the 36th Conference on Neural Information Processing Systems. Virtual, 2022: 1-28.

[133] 王童, 李骜, 宋海荦, 等. 基于分层深度强化学习的移动机器人导航方法[J]. 控制与决策, 2022, 37(11): 2799-2807.
(Wang T, Li A, Song H L, et al. Navigation method for mobile robot based on hierarchical deep reinforcement learning[J]. Control and Decision, 2022, 37(11): 2799-2807.)

[134] 赵铭慧, 张雪波, 郭宪, 等. 基于分层强化学习的通用装配序列规划算法[J]. 控制与决策, 2022, 37(4): 861-870.
(Zhao M H, Zhang X B, Guo X, et al. A general assembly sequence planning algorithm based on hierarchical reinforcement learning[J]. Control and Decision, 2022, 37(4): 861-870.)

[135] Wu B. Hierarchical macro strategy model for MOBA game AI[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 1206-1213.

[136] Lin Z C, Li J Y, Shi J N, et al. JueWu-MC: Playing minecraft with sample-efficient hierarchical reinforcement learning[C]. Proceedings of the31st International Joint Conference on Artificial Intelligence. Austria, 2022: 3257-3263.

[137] Goyal J, Madan A, Narayan A, et al. ASD: A framework for generation of task hierarchies for transfer in reinforcement learning[C]. Neural Information Processing. Cham: Springer International Publishing, 2018: 313-325.

[138] Menashe J, Stone P. Escape room: A configurable testbed for hierarchical reinforcement learning[C]. Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems. Virtual, 2019: 2123-2125.

[139] Dalal G, Hallak A, Mannor S, et al. SoftTreeMax: Policy gradient with tree search[J/OL]. 2022, arXiv: 2209.13966.

[140] Allen C, Parikh N, Gottesman O, et al. Learning Markov state abstractions for deep reinforcement learning[J/OL]. 2021, arXiv: 2106.04379.

[141] Starre R A N, Loog M. An analysis of abstracted model-based reinforcement learning[J/OL]. 2022, arXiv: 2208.14407.

[142] 程恺, 陈刚, 余晓晗, 等. 知识牵引与数据驱动的兵棋AI设计及关键技术[J]. 系统工程与电子技术, 2021, 43(10): 2911-2917.
(Cheng K, Chen G, Yu X H, et al. Knowledge traction and data-driven wargame AI design and key technologies[J]. Systems Engineering and Electronics, 2021, 43(10): 2911-2917.)

[143] Soemers D J N J, Piette É, Stephenson M, et al. Learning policies from self-play with policy gradients and MCTS value estimates[C]. IEEE Conference on Games. London, 2019: 1-8.

[144] Silver T, Chitnis R, Kumar N, et al. Inventing relational state and action abstractions for effective and efficient bilevel planning[J/OL]. 2022, arXiv: 2203.09634.

[145] Wang Y, Won K S, Hsu D, et al. Monte Carlo Bayesian reinforcement learning[C]. Proceedings of the 29th International Coference on International Conference on Machine Learning. New York, 2012: 795-802.

[146] Sharma A, Harrison J, Tsao M, et al. Robust and adaptive planning under model uncertainty[J]. Proceedings of the International Conference on Automated Planning and Scheduling, 2021, 29: 410-418.

[147] 管延霞, 刘逊韵, 刘运韬, 等. 面向多智能体博弈的并行蒙特卡洛树搜索算法研究[J]. 计算机工程与科学, 2022, 44(12): 2128-2133.
(Guan Y X, Liu X Y, Liu Y T, et al. A parallel Monte Carlo tree search algorithm for multi-agent game[J]. Computer Engineering and Science, 2022, 44(12): 2128-2133.)

作者简介

邵天浩 (1996−), 男, 博士生, 从事智能决策、深度学习等研究, E-mail: 296749641@qq.com;

程恺 (1983−), 男, 副教授, 博士, 从事智能决策、效能评估等研究, E-mail: chengkai911@126.com;

张宏军 (1963−), 男, 教授, 博士生导师, 从事数据与知识工程、计算机仿真理论等研究, E-mail: jsnjzhj@263.com;

张可 (1996−), 女, 博士生, 从事智能决策、神经网络可解释性等研究, E-mail: 2387303531@qq.com.