计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201000129

面向高维连续行动空间的蒙特卡罗树搜索算法

刘天星　李　伟　许　铮　张立华　戚骁亚　甘中学
复旦大学智能机器人研究院　上海 200433
季华实验室　广东 佛山 528000
(mliutianxing@126.com)

摘　要　蒙特卡罗树搜索(Monte Carlo Tree Search,MCTS)在低维离散控制任务中取得了巨大的成功。然而,在现实生活中许多任务需要在连续动作空间进行行动规划。由于连续行动空间涉及的行动集过大,蒙特卡罗树搜索很难在有限的时间内从中筛选出最佳的行动。作为蒙特卡罗树搜索的一个变种,KR-UCT(Kernel Regression UCT)算法通过核函数泛化局部信息的方式提高了蒙特卡罗树搜索在低维连续动作空间的模拟效率。但是在与环境交互的过程中,为了找出最佳的行动,KR-UCT在每一步都需要从头进行大量的模拟,这使得KR-UCT算法仅局限于低维连续行动空间,而在高维连续行动空间难以在有限的时间内从行动空间筛选出最佳的行动。在与环境交互的过程中,智能体可以获得环境反馈回来的信息,因此,为了提高KR-UCT算法在高维行动空间的性能,可以使用这些反馈信息剪枝树搜索过程来加快KR-UCT算法在高维连续行动空间的模拟效率。基于此,文中提出了一种基于策略-价值网络的蒙特卡罗树搜索方法(KR-UCT with Policy-Value Network,KRPV)。该方法使用策略-价值网络保存智能体与环境之间的交互信息,随后策略网络利用这些信息帮助KR-UCT算法剪枝KR-UCT搜索树的宽度;而价值网络则通过泛化不同状态之间的价值信息对蒙特卡罗树搜索在深度上进行剪枝,从而提高了KR-UCT算法的模拟效率,进而提高了算法在高维连续行动任务中的性能。在OpenAI gym中的4个连续控制任务上对KRPV进行了评估。实验结果表明,该方法在4个连续控制任务上均优于KR-UCT,特别是在6维的HalfCheetah-v2任务中,使用KRPV算法所获得的奖励是KR-UCT的6倍。

关键词:蒙特卡罗树搜索;高维连续行动空间;深度神经网络;强化学习;核回归UCT
中图法分类号　TP181

Monte Carlo Tree Search for High-dimensional Continuous Control Space

LIU Tian-xing,LI Wei,XU Zheng,ZHANG Li-hua,QI Xiao-ya and GAN Zhong-xue
Institute of AI and Robotics,Fudan University,Shanghai 200433,China
Jihua Laboratory,Foshan,Guangdong 528000,China

Abstract　Monte Carlo tree search (MCTS) has gained great success in low discrete control tasks. However,there are many tasks in real life that require selecting action sequentially in continuous action space. Kernel regression UCT (KR-UCT) is a successful attempt in low-dimensional continuous action space by using a pre-defined kernel function to exploit the similarity of different continuous actions. However,KR-UCT gets a poor performance when it comes to high-dimensional continuous action space,because KR-UCT does not use the interacting information between agent and the environment. And when it interacts with the environment,KR-UCT needs to perform a lot of simulations at each step to find the best action. In order to solve this problem,this paper proposes a method named kernel regression UCT with policy-value network (KRPV). The proposed method can filter out more representative actions from action space to perform MCTS and generalize the information between different states to pruning MCTS. The proposed method has been evaluated by four continuous control tasks of the OpenAI gym. The experimental results show that KRPV outperforms KR-UCT in all tested continuous control tasks. Especially for the six-dimensional HalfCheetah-v2 task,the rewards gained by KRPV are six-times of that of KR-UCT.

Keywords　Monte Carlo tree search,High dimensional continuous action space,Deep neural network,Reinforcement learning,Kernel regression UCT

到稿日期:2020-10-22　返修日期:2021-03-12　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:广东省季华实验室基金资助项目(X190021TB190);上海市科学技术委员会项目(19511132000)
This work was supported by the Jihua Laboratory Fund(X190021TB190) and Shanghai Committee of Science and Technology,China(19511132000).
通信作者:甘中学(ganzhongxue@fudan.edu.cn)

1 引言

在现实生活中，许多任务需要在连续动作空间进行行动规划，即在连续行动空间有序地选择一系列行动以完成特定的目的。例如，在机械臂的控制任务中，为了完成机械臂抓取任务，机械臂需要规划出最优的行动序列来高效地完成任务；在导航任务中，为了到达目的地，智能体需要规划出从出发地到目的地的最优路径；在乒乓球游戏中，为了获取胜利，选手们需要制定一系列行动策略来迷惑并击败对手。解决连续行动空间尤其是高维连续行动空间的行动规划问题有利于强化学习在实际生活中的应用，提高智能体的自动化水平，进而节省资源，减少一定的人力浪费。蒙特卡罗树搜索（MCTS）是在决策空间中随机抽取样本，随后根据搜索结果构建搜索树来解决行动规划问题的一种方法[1]。蒙特卡罗树搜索在离散控制和决策领域取得了巨大的成功，特别是与深度学习相结合之后，在国际象棋、Shogi 和 Atari 等游戏中均表现出超人类水平的性能[2-3]。然而由于蒙特卡罗树搜索固有的离散性，它只能局限于离散的行动空间。当涉及连续行动空间，特别是高维行动空间时，蒙特卡罗树搜索很难在有限的时间内从行动空间选出最佳的行动。

针对蒙特卡罗树搜索在连续行动空间中的问题，研究者们提出了许多先进的方法。使蒙特卡罗树搜索适应连续控制任务的一种直观的方法是对行动空间进行离散化，随后进行蒙特卡罗树搜索。但随行动空间维数的增长，蒙特卡罗树搜索需要考虑的行动集也呈指数级增长，这使得蒙特卡罗树搜索很难在有限次的模拟中找到最佳的行动。为了将蒙特卡罗树搜索算法拓展到连续空间中，Chaslot 等提出了逐步加宽（progressive widening）策略，即先从连续的行动空间筛选出一个初始的行动集合，而后随着模拟次数的增加不断向这个行动集合中增加新的行动，从而达到连续的目的[4]。Couëtoux 等在 2011 年提出了 cRAVE（Continuous Rapid Action Value Estimates）算法，利用高斯卷积通过计算节点与节点的相似性的方式快速评估该节点所对应的行动价值，从而将有限的统计信息泛化到不同兄弟节点中[5]。在逐步加宽策略和 cRAVE 算法的基础上，Yee 等提出了 KR-UCT 算法[6]。与 cRAVE 类似，KR-UCT 算法以核回归计算不同行动之间相似性的方式在所有探索过的动作和未探索的动作空间之间共享局部信息。核函数的利用，一方面使 KR-UCT 算法在反向回溯过程中将某一节点上的信息泛化到兄弟节点，另一方面使 KR-UCT 有启发地从连续行动空间中选择最需要探索的行动加入原有的行动集。核函数的使用提升了 KR-UCT 的模拟效率，使 KR-UCT 在相同模拟次数下对根节点行动集中的每个行动的评估更加准确，进而提升了蒙特卡罗树搜索在低维连续行动空间中的性能。

与原有的蒙特卡罗树搜索相似，只利用局部信息并不能使 KR-UCT 高效地从高维连续动作空间找到最佳的行动。但 KR-UCT 是一种无记忆的算法，即 KR-UCT 在某个与环境交互的时刻，KR-UCT 不能利用这一时刻之前所有与环境的交互信息。如果可以充分地利用这些交互信息，则可进一步提升蒙特卡洛树的效率。基于此，本文提出了一种基于策略-价值网络的蒙特卡罗树搜索方法（KRPV）。该方法利用深度神经网络保存 KR-UCT 与环境交互的信息，随后使用这些信息剪枝蒙特卡罗树搜索过程，实现在相同的模拟次数下尽可能准确地评估根节点处不同行动之间的价值，从而在有限的时间内筛选出最佳的行动。本文在 OpenAI 的 gym 平台上进行了 KRPV 与 KR-UCT 的对比实验[7]，实验证明，与 KR-UCT 算法对比，在 4 个连续控制场景中，本文提出的方法（KRPV）单个周期内所获得的奖励得到大幅提升。综上，本文的主要贡献如下：

（1）在 KR-UCT 的基础上，使用神经网络保存了 KR-UCT 与环境之间的交互信息；

（2）在 KR-UCT 的基础上，通过保存的交互信息对蒙特卡洛树搜索进行剪枝；

（3）在 gym 平台上不同维度的场景中对本文提出的算法进行了测试，实验证明了所提算法的优越性。

2 相关工作

Rémi 等在 2006 年将树搜索算法与蒙特卡罗取样方法相结合，提出了蒙特卡罗树搜索算法[8]。Kocsis 等将置信上限应用到树搜索中，提出了 UCT 树搜索算法[9]。为了将蒙特卡罗树搜索算法拓展到连续空间，Chaslot 等提出了逐步加宽的策略[4]。Couëtoux 等为解决逐步加宽中的欺骗问题，提出了 Double Progressive Widening 算法[10]。随后 Couëtoux 等在此基础上提出了 cRAVE 算法，利用高斯卷积将有限的统计信息泛化到不同兄弟节点中[5]。与 Couëtoux 等相似，Yee 等在 2016 年提出了 KR-UCT 算法，该算法一方面使用高斯核密度在反向传播过程中将某一节点上的信息泛化到兄弟节点，另一方面通过高斯核密度根据已有信息筛选出新的行动，并添加到行动集合中，该方法在冰壶游戏中取得了一定的效果[6]。

Sébastien 等另辟蹊径，提出了分层优化（Hierarchical Optimistic Optimization，HOO）的方法，该方法利用二叉树结构不断对连续行动空间进行分段，直到达到规定的深度，并根据行动空间分段过程中得到的统计信息从叶子节点中选择出最好的行动[11]。HOO 方法只是在每一步使用树结构对连续行动空间进行分段，不能进行长期规划。因此 Mansley 等在蒙特卡罗树中的每个节点利用 HOO 方法替代 UCB 进行行动的选择，并提出了 HOOT 算法（Hierarchical Optimistic Optimization applied to Trees，HOOT）[12]。与 HOOT 不同，Weinstein 等将 HOO 方法与环境动力模型相结合并提出了 HOLOP 算法，使 HOO 在分解连续行动空间的同时探索不同的行动序列，随后在这些序列中选出最佳的序列[13]。与 Mansley 等相似，Kim 等使用 VOO（Voronoi optimistic optimization）算法对连续行动空间进行分段及采样，并在蒙特卡罗树搜索的基础上使用 VOO 方法替代 UCB 算法，提出了 VOOT 算法，该方法在机器人规划任务中取得了一定的效果[14]。

Chaslot 等对蒙特卡洛树搜索过程进行并行化，加快了模拟的效率[15]。随后 Kurzer 等将这些并行的方案扩展到连续行动空间，并在多智能体环境中取得了一定的效果[16]。

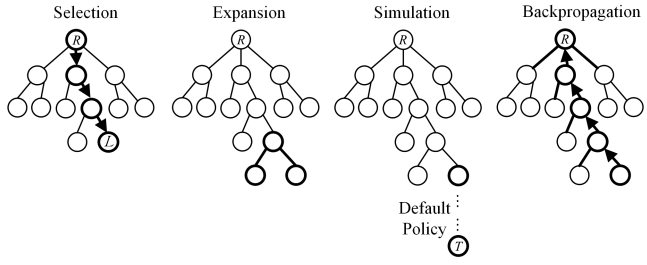
这些方法提供了很多宝贵的蒙特卡罗树搜索连续化方案，并在指定任务中取得了很好的效果。但这些方法中每个个体只能记忆一个时间步或者一个周期内的状态信息及行动信息，一旦智能体到达下一个时间步，这些方法需要从头开始探索环境中的状态和行动信息，造成了算力的极大浪费。为了解决蒙特卡罗树搜索中短时记忆的缺陷，Silver 等利用深度神经网络保存了蒙特卡罗树搜索过程中的交互信息，构成个体的长时记忆，并在此基础上提出了 AlphaZero 算法，随后在围棋游戏中首次击败人类[3]。独立于 AlaphZero，Anthony 等提出了专家迭代(Exit)的算法，该算法同样使用深度神经网络对蒙特卡罗树进行剪枝，并在 Hex 游戏中取得了一定的效果[17]。在 AlphaZero 的基础上，MuZero 使用深度神经网络对环境模型进行建模，并在 Atari 平台上在一系列具有挑战性的游戏中取得了超人的表现[2]。Kartal 等将蒙特卡罗树搜索与异步分布式深度强化学习方法相结合，并在双人迷你版 Pommerman 游戏中更快地收敛到一个比较好的策略[18]。MCTS-A3C 算法使用 A3C 算法取代了 MCTS 的走棋流程，并在森林火灾场景中取得了一定的效果[19]。Zhang 等提出了一种基于截断树搜索的分步逆向课程学习方法(RevCut Tree Search)，并在自行设计的单人游戏中达到了超人类的水平[20]。

Lee 等将 KR-UCT 与神经网络相结合，提出了 KR-DL-UCT 算法，使得 KR-UCT 在低维冰壶游戏场景中的性能得到大幅提升[21]。但 KR-DL-UCT 由于采用了离散的策略，因此局限于低维的游戏场景。上述方法分别在不同的领域获得了出色的性能，但这些方法的关注点要么是游戏博弈场景，要么是低维或离散的连续行动空间。为了将蒙特卡罗树搜索扩展到高维连续行动控制空间，本文在 KR-UCT 的基础上提出了 KRPV 算法。该方法在 KR-UCT 的基础上加入了策略网络和价值网络来保存智能体与环境交互的信息，随后用这些信息剪枝 KR-UCT 中树搜索的宽度和深度，从而提高 KR-UCT 的模拟效率。

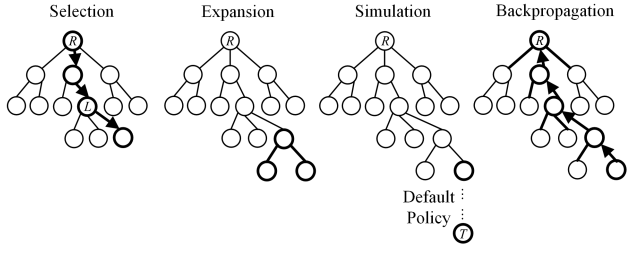
3 KR-UCT (Kernel Regression UCT)

蒙特卡罗树搜索是一种通过在决策空间随机取样并根据取样结果构建搜索树的方法，是 2006 年由 Rémi 将蒙特卡罗方法应用到游戏树搜索而提出的一种算法，并在许多棋盘游戏中获得了一定的效果[8]。然而，由于蒙特卡罗树固有的离散属性，蒙特卡罗树搜索只能局限于低维的离散行动空间。为了将蒙特卡罗树搜索应用到连续行动空间，Yee 等结合逐步加宽和快速价值估计的策略，一方面利用核密度函数从行动空间中选择有代表性的行动进行蒙特卡罗树搜索，另一方面通过核密度估计将模拟中获得的局部信息泛化到树结构中，从而缩短蒙特卡罗树在连续行动空间所需的时间，提高了蒙特卡罗树搜索在连续行动空间的性能[6]。

KR-UCT 在每次模拟搜索中包含了选择、扩展、模拟、反向传播和最终选择这 5 个过程[6]，如图 1 所示，其具体细节如下。



(a)节点 L 亟待扩展的情况



(b)节点 L 被充分访问的情况

图 1 KR-UCT 中的模拟搜索

Fig. 1 Simulation of KR-UCT

(1)选择过程(Selection)：每次模拟从根节点 R 开始，根据子节点选择策略递归地选择子节点，直到到达某一个满足条件的节点 L。这里，满足条件的节点要么是一个亟待扩展的节点，要么这个节点已经被充分访问。其中，一个亟待扩展的节点指没有孩子的节点，一个被充分访问的节点指这个节点在之前的模拟中被访问过一定的次数。子节点的选择策略为：

$$action=\arg\max_{a\in A} E(v|a)+C\sqrt{\frac{\log\sum_{b\in A}W(b)}{W(a)}} \tag{1}$$

$$E(v|a)=\frac{\sum_{b\in A}K(a,b)\overline{v_b}n_b}{\sum_{b\in A}K(a,b)n_b} \tag{2}$$

$$W(a)=\sum_{b\in A}K(a,b)n_b \tag{3}$$

其中，A 是节点的行动集合；$E(v|a)$是行动 a 的价值期望；$W(a)$是通过行动 a 的兄弟行动的访问次数对 a 的访问次数的估计；$K(a,b)$是行动 a 与行动 b 之间的核函数，可以看作行动 a 与行动 b 之间的相似性；n_b 是行动 b 的真实访问次数；$\overline{v_b}$是行动 b 的真实价值。

(2)扩展过程(Expansion)：当节点 L 是一个亟待扩展的节点，KR-UCT 随机从行动空间初始化一个行动集，随后从该行动集中随机选择一个行动进行模拟过程。当节点 L 是已经被充分访问过的节点，KR-UCT 向节点的行动集中添加新的行动。如果一个行动与行动集中最好的行动相似，同时又不能被现有行动集很好地表示，那么这个行动作为新行动被添加到行动集中。新行动的筛选公式为：

$$newAction=\arg\max_{K(action,a)>\tau} W(a) \tag{4}$$

(3)模拟过程(Simulation)：扩展过程结束后，KR-UCT 通过一个默认的策略随机走棋直到终端节点 T。

(4)反向传播过程(Backpropagation)：根据终端节点的价

值信息 v_T 和方程(1)－方程(3)，计算从节点 L 到节点 R 的路径上所有节点的价值信息及其兄弟节点的相关信息。

(5)最终选择过程：在经过充分的模拟之后，KR-UCT 从根节点中选择最好的行动与环境进行交互，选择行动的方式如下：

$$action=\arg\max_{a\in A} E(v|a)-C\sqrt{\frac{\log\sum_{b\in A}W(b)}{W(a)}} \tag{5}$$

4 KRPV

KR-UCT 虽然在一定程度上提高了蒙特卡罗树在连续行动空间的模拟效率，但在高维行动空间仍需要耗费大量的时间进行模拟搜索来寻找最佳的行动。纵观 KR-UCT 的整个过程，我们发现模拟搜索的时间主要耗费在扩展过程和模拟过程中。在扩展过程中，KR-UCT 随机从行动空间选择一组行动进行初始化，随后在模拟搜索中逐步向最佳的行动靠拢。在模拟过程中，为了评估根节点的价值，KR-UCT 需要模拟终端节点。因此，为了加快 KR-UCT 在高维搜索空间的效率，一方面需要一个策略函数在扩展过程中生成一组较好的行动来进行搜索，从而缩短找到最佳行动的时间，另一方面需要一个价值函数，截断 KR-UCT 中的模拟过程。而 KR-UCT 是一种无记忆性的算法，即 KR-UCT 在第 t 步不能使用前 $t-1$ 步与环境交互的信息。因此利用这些交互信息来训练策略函数和价值函数并以此剪枝蒙特卡罗树搜索过程，可以作为提升 KR-UCT 在高维连续行动空间的效率的一种方法。基于此，本文提出了一种基于策略-价值网络的蒙特卡罗树搜索方法(KRPV)。该方法利用深度神经网络保存 KR-UCT 与环境交互的信息，随后使用这些信息剪枝蒙特卡罗树搜索过程，从而实现在相同的模拟次数下尽可能准确地评估根节点处不同行动之间的价值。

算法 1 KRPV 算法

输入：(K，N)

输出：(θ，φ)

1. 初始化神经网络的参数 θ，φ；
2. 初始化数据池 B；
3. for k←1 to K do
   3.1. 初始化状态 s_0；
   3.2. while(not done)
      3.2.1. 从 KR-FPV-UCT 模块中获取当前状态 s_t 下的动作 a_t^*；
      3.2.2. 执行动作 a_t^*，环境反馈奖励 r_t 和下一步的状态 s_{t+1}；
      3.2.3. 将(s_t，a_t^*，r_t，s_{t+1})保存到数据池 B 中；
   3.3. for n←1 to K do
      3.3.1. 从数据池中采样一批数据；
      3.3.2. 根据这些数据和式(6)－式(8)更新神经网络的参数.

KRPV 主要由神经网络标签的生成和训练以及神经网络与 KR-UCT 的结合两大部分构成，伪代码如算法 1 所示。神经网络生成和训练部分主要由策略模块和价值模块组成，其中策略模块用于初始化蒙特卡罗树搜索时所需考虑的行动集，将一些不好的行动尽早排除在外，从而达到剪枝 KR-UCT 中搜索树的宽度和加快模拟速度的目的；价值模块则在叶子节点处截断搜索，从而达到剪枝 KR-UCT 中搜索树的深度及加快模拟速度的目的。策略模块和价值模块与 KR-UCT 的融合构成了 KR-UCT-PV 模块，最后利用 KR-UCT-PV 模块与环境的交互信息完成对策略模块和价值模块的训练。

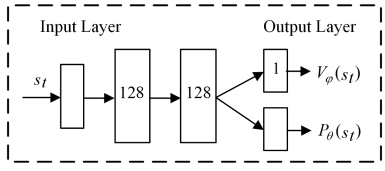
4.1 策略网络模块

KR-UCT 的扩展过程中使用了一个固定的随机分布初始化节点行动集，即随机地对行动空间进行采样，随后进行蒙特卡罗树搜索。使用不可学习的随机分布不能区分出哪些行动可以带来回报，哪些行动会带来惩罚，因此不能从行动空间中筛选出有代表性的行动。针对这个问题，本文使用一个可学习的随机分布学习在不同状态下各个行动的概率分布，随后根据该分布从行动空间中筛选出有代表性的行动构成节点的初始行动集，进行蒙特卡罗树搜索，从而在进行蒙特卡罗树模拟之前排除不合适的行动。

如图 2 所示，策略网络由输入层、两个隐藏层和输出层构成，其中输入层和输出层神经单元的个数由具体的任务决定。输入层、两个隐藏层的参数与价值网络模块共享。本文使用 Beta 分布对策略网络进行建模，策略网络的输出作为 Beta 分布的两个参数。由于策略网络生成的策略分布用于从行动空间中筛选出一组好的行动，随后蒙特卡罗树搜索进一步对这些行动进行评估，从而找出这一组行动中最佳的行动，因此蒙特卡罗树搜索模拟后推荐的最佳行动不劣于策略网络推荐的行动。故可以将蒙特卡罗树搜索当作老师，而将策略网络当作学生，使策略网络模仿蒙特卡罗树搜索与环境交互时的行动序列，即使策略网络在相同情况下执行这段行动序列的概率最大化。策略网络的标签可以从蒙特卡罗树搜索与环境的交互中获得，即蒙特卡罗树搜索推荐的最佳行动的概率为 1，其他行动的概率为 0。KRPV 随后根据式(6)对策略网络进行训练。

$$L_\theta=\frac{1}{T}\Sigma-\int q(a)\log p_\theta(a)\,\mathrm{d}a=-\frac{1}{T}\Sigma\log p_\theta(a^*) \tag{6}$$

其中，$q(a)$为策略模块的标签，$p_\theta(a^*)$为策略网络的输出，a^*为蒙特卡罗树搜索模拟之后从根节点选择出的最佳行动。



注：$V_\varphi(s_t)$为价值模块，$P_\theta(s_t)$为策略网络模块

图 2 神经网络结构

Fig. 2 Structure of neural network

4.2 价值网络模块

在 KR-UCT 的模拟过程中，KR-UCT 通过默认策略随机从叶子节点走到终端节点，随后通过终端节点的价值对叶子节点的价值进行估计。但是在高维连续行动空间，由于行动空间巨大，为了得到对状态价值的准确估计，需要进行大量的随机走棋。为了加快评估过程，本文使用神经网络对不同状态之间的价值进行泛化，截断蒙特卡罗树搜索中的随机走棋过程，

从而在加快模拟速度的同时提高对状态价值评估的准确性。

如图 2 所示，价值网络模块由输入层、两个隐藏层和输出层构成，其中输入层的神经单元的个数由具体的控制任务确定。两个隐藏层的神经元个数为 128，输出层由一个神经单元构成。价值网络模块的标签由智能体与环境交互时环境反馈的奖励决定，即式(7)，随后利用有监督学习的方式与式(8)进行训练。

$$\hat{V}(s)=\sum_{i=t}^{T}\gamma^{i-t}R_i \tag{7}$$

$$L_{\varphi}=\frac{1}{T}\sum(V_{\varphi}(s)-\hat{V}(s))^2 \tag{8}$$

其中，$\hat{V}(s)$是状态 s 的价值标签，R_i 是智能体在第 i 步的奖励，$V_{\varphi}(s)$为神经网络对状态 s 的价值估计。

4.3 KR-PV-UCT

KR-PV-UCT 的模拟与 KR-UCT 相同且都包含 5 个过程，但其具体细节与 KR-UCT 不同，这 5 个过程分别是选择过程、扩展过程、评估过程、反向传播过程和最终选择过程。

(1)选择过程(Selection)：每次模拟从根节点 R 开始，根据子节点选择策略选择子节点，直到到达某一个满足条件的节点 L。这里，满足条件的节点要么是一个亟待扩展的节点，要么这个节点已经被充分访问。判断一个节点是否被充分访问由式(12)确定[10]。子节点的选择策略如式(1)－式(3)及式(9)所示。

$$m(s)=c_{pw}\times n(s)^{\kappa} \tag{9}$$

其中，$m(s)$为节点 s 的行动集的大小，$n(s)$为节点 s 的访问次数，c_{pw}和 κ 为超参数。

(2)扩展过程(Expansion)：当节点 L 是一个亟待扩展的节点，根据策略网络从行动空间初始化一个行动集 $a_i\sim P_{\theta}(L)$，随后从行动集中随机选择一个行动进行模拟。当节点 L 已经被充分访问，KRPV 向节点的行动集中添加新的行动。如果一个行动与行动集中最好的行动相似，同时又不能被行动集很好地表示，那么这个行动作为新行动被添加到行动集中。筛选行动的公式为：

$$newAction=\underset{K(action,a)>\tau}{\arg\max}W(a) \tag{10}$$

在实际过程中，为了筛选出新行动，我们先随机从行动空间随机取样一组行动，随后在这组行动中筛选出满足式(10)的行动作为新行动加入对应的行动集。

(3)评估过程(Evaluation)：当扩展过程完成后，我们使用价值网络模块对新的叶子节点进行评估，得到叶子节点的价值 $v_{\varphi}(L)$，从而省略随机走棋过程。

(4)反向传播过程：根据对新叶子节点的价值评估、式(3)及式(9)－式(12)，更新从节点 L 到节点 R 的路径及兄弟节点的信息。

$$n_l=n_l+1 \tag{11}$$

$$v_l=\frac{\sum_i R_l^i+v_{l+1}^i}{n_l} \tag{12}$$

其中，n_l 为节点 l 的访问次数，$R_l^i+v_{l+1}^i$为每次访问节点 l 时根据搜索树下一层节点价值得到的节点 l 的价值。

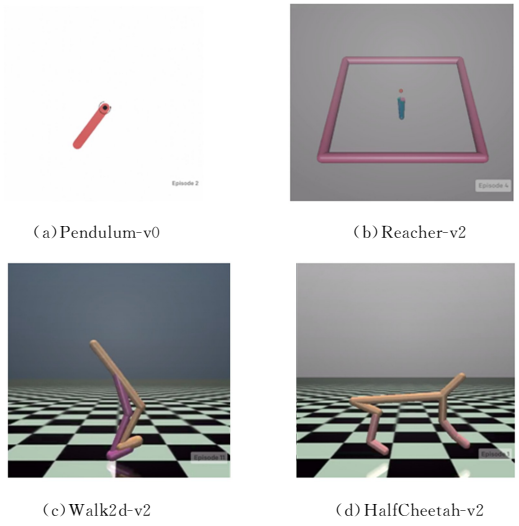
(5)最终选择过程：在经过充分的模拟之后，KR-FPV-UCT 从根节点中选择最好的行动与环境进行交互，选择行动的方式如下：

$$action=\underset{a\in A}{\arg\max}\,E(v|a)-C\sqrt{\frac{\log\sum_{b\in A}W(b)}{W(a)}} \tag{13}$$

5 实验

5.1 实验设计

为了测试 KRPV 的效果，本文在 OpenAI 的 gym 平台的 4 个场景[7]对 KRPV 算法进行了测试，如图 3 所示，这 4 个场景分别为：Pendulum-v0 (1 维的连续行动空间，3 维的状态空间)、Reacher-v2 (2 维的连续行动空间，11 维的状态空间)、Walk2d-v2(6 维的连续行动空间，17 维的状态空间)以及 HalfCheetah-v2(6 维的连续行动空间，17 维的状态空间)。本文涉及的实验相关参数如表 1 所列。



(a)Pendulum-v0 (b)Reacher-v2

(c)Walk2d-v2 (d)HalfCheetah-v2

图 3 测试场景

Fig. 3 Test cases

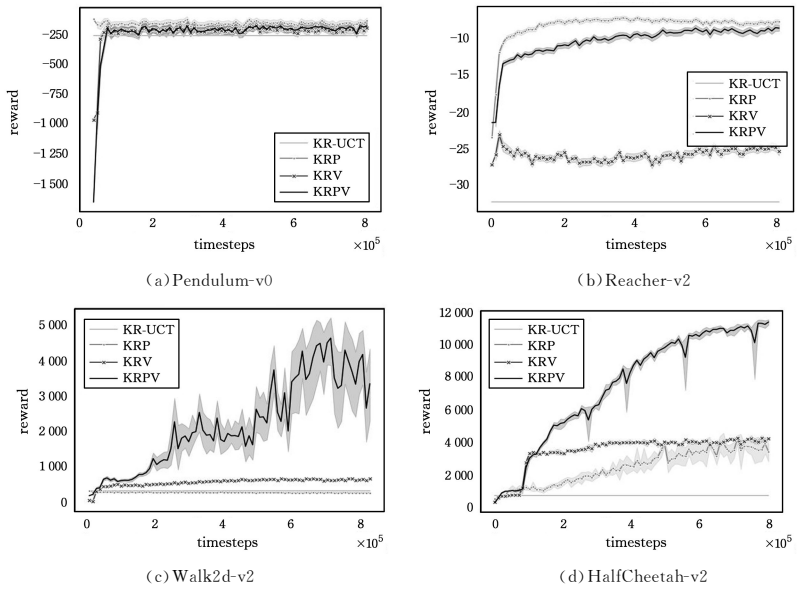
表 1 参数设置

Table 1 Parameter settings

Parameter	Value
Batch Size	32
Optimizer	RMSprop
Learning Rate	3.0×10^{-4}
Total time steps	8×10^{5}
Simulation Times	50
Discount(γ)	0.98
Kappa(κ)	0.6
c_{pw}	1.0
τ	2.0
C	0.5

5.2 KRPV 与 KR-UCT 的比较

为了探索策略模块与价值模块对 KR-UCT 的提升效果，本节将 KR-UCT、KRV(KR-UCT＋价值模块)、KRP(KR-UCT＋策略模块)与 KRPV(KR-UCT＋策略模块＋价值模块)这 4 种方法进行了对比。图 4 是这 4 种方法在不同场景下的训练图像，横轴为实验过程中智能体与环境交互的次数，纵轴为与环境交互过程中智能体的每个周期所获得的奖励。表 2 列出了训练结束后，在 4 个场景、4 个不同的环境种子下测得的几种方法的性能。



(a)Pendulum-v0 (b)Reacher-v2

(c)Walk2d-v2 (d)HalfCheetah-v2

注：横轴为智能体与环境交互的次数，纵轴为不同方法的表现

图 4 KRPV，KRV，KRP 及 KR-UCT 在不同场景下训练过程中的效果

Fig. 4 Performance of KRPV，KRV，KRP and KR-UCT

表 2 KRPV，KRV，KRP，KR-UCT 在训练结束后 4 个场景下的测试效果

Table 2 Performance of KRPV，KRV，KRP，KR-UCT in 4 cases after training

Algorithm	KR-UCT	KRP	KRV	KRPV
HalfCheetah-v2	1 660.54± 758.56	3 838.64± 1 334.54	4 099.19± 166.72	**11 483.50± 338.88**
Walker2d-v2	264.97± 59.57	295.24± 63.68	409.58± 219.48	**3 355.87± 1 585.19**
Reacher-v2	−33.48± 2.86	−9.17± 3.11	−26.44± 3.68	**−9.03± 3.36**
Pendulum-v0	−186.71± 112.38	**−167.77± 86.74**	−202.15± 112.50	−213.55± 115.05

注：表中的数值为在特定场景下特定的方法在 4 个随机种子上的平均性能及方差

从图 4 和表 2 来看，相较于 KR-UCT 而言，KRP 方法和 KRV 方法在这 4 个场景中所获得的奖励之和均有所提升，这表明策略模块和价值模块对 KR-UCT 有一定的提升作用。其中 KRP 方法在低维连续行动空间(Pendulum-v0，Reacher-v2)有较大的提升，而 KRV 在高维连续空间有较大的提升。这是因为在低维的连续行动场景中，智能体的状态空间维度也较低，蒙特卡罗树搜索通过随机走棋过程足以评估出行动集中各个状态的价值，使用价值模块替换随机走棋过程并不能使算法在单个周期内获得更多的奖励；但策略网络却可以帮助蒙特卡罗树从行动空间筛选出一组比较好的行动进行搜索，而不再需要考虑一些没有价值的行动。

在高维的连续行动空间，KRP 与 KR-UCT 的对比反映了策略在高维行动空间对 KR-UCT 的性能并没有太大的提升，然而 KRPV 与 KRV 的对比表明策略模块确实可以为 KR-UCT 的性能带来巨大的提升。这是因为在高维的连续行动空间，蒙特卡罗树通过随机走棋过程对状态的估值不够准确，使得蒙特卡罗树很难找出最佳的行动，因此策略网络不能从蒙特卡罗树搜索与环境交互中学到任何知识，从而在高维连续行动空间上基本失效。但通过 KRP 与 KRPV 在高维连续行动空间的对比可知，当采用价值网络对不同状态之间的信息进行泛化时，价值网络对状态的评估较为准确，从而能够帮助蒙特卡罗树找出最佳的行动，进而可引导策略网络区分出不同动作的好坏，而策略网络反哺蒙特卡罗树搜索，使蒙特卡罗树搜索在高维连续行动空间达到更高的性能。

KRP，KRV 与 KR-UCT 之间的对比实验也表明了价值网络在高维连续行动空间可以泛化不同行动之间的信息，可以提高对状态价值评估的准确性；KRPV 与 KRV 的对比表明，相比固定的策略，策略网络可以从行动空间筛选出一组值得考虑的行动集进行蒙特卡罗树搜索，从而减少蒙特卡罗树的宽度，使得蒙特卡罗树在较短的时间内找出最佳的行动与环境交互。综合表 2 可以看出，KRPV 算法在 3 个场景中取得了最好的效果，但在 Pendulum-v0 场景中不如其他 3 个场景。但从 KRPV，KRP，KRV 和 KR-UCT 在 Pendulum-v0 场景中的对比来看，造成 KRPV 效果较差的原因是价值模块在低维场景中对状态的估计劣于直接采用随机走棋过程对状态价值的评估，当将价值模块换为随机走棋过程，即将 KRPV 改为 KRP 时，算法便达到了最好的效果。

5.3 参数分析

本节主要探索了模拟次数对 KRPV 效果的影响，并在 HalfCheetah-v2 场景中在不同模拟次数下对 KRPV 算法进行了测试。实验结果如图 5 所示，实验表明，随着模拟次数的增加，KRPV 算法在 HalfCheetah 场景中的效果逐渐变好。这是因为模拟次数的增加使蒙特卡罗树进行了更多次的模拟，进而提升了对行动集中行动价值评估的准确性，使其更好地发现最佳的行动，提升了蒙特卡罗树搜索的效果。

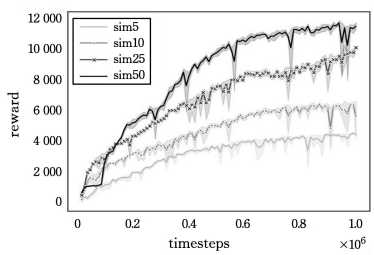


图 5 KRPV 在不同模拟次数下的表现

Fig. 5 Performance of KRPV in different simulation times

从图 5 可以看出，随着模拟次数的增加，单位模拟次数对效果的提升呈现下降的趋势。这是因为当模拟次数增加到一定程度时，蒙特卡罗树对行动集中的行动评估逐渐稳定，继续增加模拟次数对行动价值评估的影响微乎其微。

结束语 本文针对 KR-UCT 在高维连续空间的问题，提出了一种基于策略-价值网络(KRPV)的蒙特卡罗树搜索方法。该方法通过学习策略-价值网络的方式泛化智能体与环境之间交互的全局信息，学习到的策略网络可以帮助 KR-UCT 从高维连续行动空间筛选出最值得考虑的一组行动进行评估，价值网络则通过泛化不同状态之间的价值信息对蒙特卡罗树搜索进行剪枝，从而提高了 KR-UCT 在高维连续行动任务中的性能。

然而，KRPV 收集的数据是由蒙特卡罗树搜索推荐的轨迹所形成的，而策略网络是通过模仿这些轨迹得以训练的，导致所训练的策略网络比较贪婪，进而导致了 KRPV 一定程度上的贪婪，因而难以在需要探索的场景中找到最佳策略。因此，如何改善 KRPV 在需要扩展的场景中的效果是我们未来的工作。

致 谢 感谢关春博士、刘创博士、田小禾同学和张校志同学耐心地回答和帮忙解决所遇到的一些问题；感谢上海智能机器人工程技术研究中心(Shanghai Engineering Research Center of AI & Robotics)、智能机器人教育部工程研究中心(Engineering Research Center of AI & Robotics, Ministry of Education)和北京深度奇点科技有限公司(Beijing Deep Singularity Technology Co., Ltd.)对本工作提供的支持。

参 考 文 献

[1] BROWNE C B, POWLEY E, WHITEHOUSE D, et al. A Survey of Monte Carlo Tree Search Methods[J]. IEEE Transactions on Computational Intelligence & AI in Games, 2012, 4(1): 1-43.

[2] SCHRITTWIESER J, ANTONOGLOU I, HUBERT T, et al. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model[J]. arXiv: 1911.08265.

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

[4] CHASLOT G, WINANDS M, VAN DEN HERIK H J, et al. Progressive strategies for monte-carlo tree search[J]. New Mathematics and Natural Computation, 2008, 4(3): 343-357.

[5] COUËTOUX A, MARIO M. Continuous rapid action value estimates [C]// Asian Conference on Machine Learning. 2011: 19-31.

[6] YEE T, VILAM L, BOWLING M. Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty[C]// IJCAI 2016. AAAI Press, 2016: 690-697.

[7] BROCKMAN G, CHEUNG V, PETTERSSON L, et al. OpenAI Gym[J]. arXiv: 1606.01540.

[8] RÉMI C. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search[C]// Proceeding of the International Conference on Computer & Games. 2006: 72-83.

[9] KOCSIS L, SZEPESVÁRI C. Bandit based monte-carlo planning [C]// 17th European Conference on Machine Learning. 2006: 282-293.

[10] COUËTOUX A, HOOCK J B. Continuous Upper Confidence Trees[C]// International Conference on Learning and Intelligent Optimization. 2011: 433-445.

[11] SÉBASTIEN B, RÉMI M, STOLTZ G, et al. Online Optimization in X-Armed Bandits[J]. Advances in Neural Information Processing Systems, 2009: 201-208.

[12] MANSLEY C R, WEINSTEIN A, LITTMAN M L. Sample-Based Planning for Continuous Action Markov Decision Processes[C]// International Conference on International Conference on Automated Planning & Scheduling. AAAI Press, 2011.

[13] WEINSTEIN A, LITTMAN M L. Bandit-based planning and learning in continuous-action markov decision processes[C]// Proceedings of the Twenty-Second International Conference on International Conference on Automated Planning and Scheduling. 2012: 306-314.

[14] KIM B, LEE K, LIM S, et al. Monte Carlo Tree Search in Continuous Spaces Using Voronoi Optimistic Optimization with Regret Bounds [C]// AAAI 2020. 2020: 9916-9924.

[15] CHASLOT G M B, WINANDS M H, VAN DENHERIK H J. Parallel monte-carlo tree search[C]// International Conference on Computers and Games. 2008: 60-71.

[16] KURZER K, CHRISTOPH H, MARIUS Z J. Parallelization of Monte Carlo Tree Search in Continuous Domains[J]. arXiv: 2003.13741.

[17] ANTHONY T, TIAN Z, BARBER D. Thinking Fast and Slow with Deep Learning and Tree Search[C]// Neural Information Processing Systems. 2017: 5360-5370.

[18] 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.

[19] SUBRAMANIAN S G, CROWLEY M. Combining MCTS and A3C for Prediction of Spatially Spreading Processes in Forest Wildfire Settings[C]// Canadian Conference on Artificial Intelligence. 2018: 285-291.

[20] ZHANG H, CHENG F, XU B, et al. RevCuT Tree Search Method in Complex Single-player Game with Continuous Search Space[C]// 2019 International Joint Conference on Neural Networks (IJCNN). 2019: 1-8.

[21] LEE K, KIM S, CHOI J, et al. Deep Reinforcement Learning in Continuous Action Spaces: a Case Study in the Game of Simulated Curling[C]// Proceedings of the 35th International Conference on Machine Learning. 2018: 2937-2946.

LIU Tian-xing, born in 1996, postgraduate. His main research interests include reinforcement learning, machine learning and collective intelligence evolution.

GAN Zhong-xue, born in 1952, Ph. D, distinguished professor. His main research interests include basic theory and key technologies of collective intelligence, autonomous intelligent robot and flexible automatic control.