软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2023,34(4):1749–1764 [doi: 10.13328/j.cnki.jos.006708]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

结构交互驱动的机器人深度强化学习控制方法*

余 超[1], 董银昭[2], 郭 宪[3], 冯旸赫[4], 卓汉逵[1], 张 强[2]

[1](中山大学 计算机学院, 广东 广州 510006)

[2](大连理工大学 计算机科学与技术学院, 辽宁 大连 116081)

[3](南开大学 人工智能学院, 天津 300354)

[4](国防科技大学 系统工程学院, 湖南 长沙 410073)

通信作者: 余超, E-mail: yuchao3@mail.sysu.edu.cn

摘 要: 针对深度强化学习在高维机器人行为控制中训练效率低下和策略不可解释等问题, 提出一种基于结构交互驱动的机器人深度强化学习方法(structure-motivated interactive deep reinforcement learning, SMILE). 首先, 利用结构分解方法将高维的单机器人控制问题转化为低维的多关节控制器协同学习问题, 从而缓解连续运动控制的维度灾难难题; 其次, 通过两种协同图模型(ATTENTION 和 PODT)动态推理控制器之间的关联关系, 实现机器人内部关节的信息交互和协同学习; 最后, 为了平衡 ATTENTION 和 PODT 协同图模型的计算复杂度和信息冗余度, 进一步提出两种协同图模型更新方法 APDODT 和 PATTENTION, 实现控制器之间长期关联关系和短期关联关系的动态自适应调整. 实验结果表明, 基于结构驱动的机器人强化学习方法能显著提升机器人控制策略学习效率. 此外, 基于协同图模型的关系推理及协同机制, 可为最终学习策略提供更为直观和有效的解释.

关键词: 机器人控制; 深度强化学习; 结构分解; 可解释性

中图法分类号: TP18

中文引用格式: 余超, 董银昭, 郭宪, 冯旸赫, 卓汉逵, 张强. 结构交互驱动的机器人深度强化学习控制方法. 软件学报, 2023, 34(4): 1749–1764. http://www.jos.org.cn/1000-9825/6708.htm

英文引用格式: Yu C, Dong YZ, Guo X, Feng YH, Zhuo HK, Zhang Q. Structure-motivated Interactive Deep Reinforcement Learning for Robotic Control. Ruan Jian Xue Bao/Journal of Software, 2023, 34(4): 1749–1764 (in Chinese). http://www.jos.org.cn/1000-9825/6708.htm

Structure-motivated Interactive Deep Reinforcement Learning for Robotic Control

YU Chao[1], DONG Yin-Zhao[2], GUO Xian[3], FENG Yang-He[4], ZHUO Han-Kui[1], ZHANG Qiang[2]

[1](School of Computer Science and Engineering, Sun Yat-Sen University, Guangzhou 510275, China)

[2](School of Computer Science and Technology, Dalian University of Technology, Dalian 116081, China)

[3](School of Artificial Intelligence, Nankai University, Tianjin 300354, China)

[4](School of Systems Engineering, University of National Defense Science and Technology, Changsha 410073, China)

Abstract: This study proposes structure-motivated interactive deep reinforcement learning (SMILE) method to solve the problems of low training efficiency and inexplicable strategy of deep reinforcement learning (DRL) in high-dimensional robot behavior control. First, the high-dimensional single robot control problem is transformed into a low-dimensional multi-controllers control problem according to some structural decomposition schemes, so as to solve the curse of dimensionality in continuous control. In addition, SMILE dynamically outputs the dependency among the controllers through two coordination graph (CG) models, ATTENTION and PODT, in order to realize the information exchange and coordinated learning among the internal joints of the robot. In order to balance the computational complexity and information redundancy of the above two CG models, two different models, APODT and PATTENTION, are then proposed to update the CG, which can realize the dynamic adaptation between the short-term dependency and long-term dependency

* 基金项目: 国家自然科学基金(U1908214, 62076259); 腾讯犀牛鸟基金(JR202063)
收稿时间: 2021-09-30; 修改时间: 2021-11-28, 2022-03-30; 采用时间: 2022-05-17

among the controllers. The experimental results show that this kind of structurally decomposed learning can improve the learning efficiency substantially, and more explicit interpretations of the final learned policy can be achieved through the relational inference and coordinated learning among the components of a robot.

Key words: robotic control; deep reinforcement learning; structural decomposition; interpretation

近年来, 智能机器人受到学术界和产业界的广泛关注. 机器人的控制技术被认为是科技发展和社会进步的重要技术之一, 广泛应用于医疗手术、农业生产、工业制造、抗震救灾等领域[1]. 然而, 设计有效的控制器, 使机器人在复杂变化的未知环境中稳定地工作、自适应完成任务, 一直是机器人研究领域的难点[2].

作为机器学习的重要分支, 深度强化学习(deep reinforcement learning, DRL)成为一个蓬勃发展的研究领域, 在医疗健康[3]、娱乐游戏[4]、自动驾驶[5]、自然语言处理[6]和金融市场[7]等领域取得一系列研究成果. 深度强化学习结合了深度学习环境感知能力与强化学习环境交互能力, 在解决复杂的连续空间决策问题具有显著优势, 因而被广泛应用于机器人行为控制问题中[8], 如机器人行走[9]、机械臂抓取[10]、无人机飞行[11]等. 然而, 基于深度强化学习的机器人行为控制研究依然存在以下 3 个挑战.

(1) 策略探索空间存在维度灾难: 机器人行为控制问题属于高维度连续域决策问题, 例如: 对于仿生机器人行走控制问题, 其决策空间包括身体重要关节的位置、速度等 40 维状态信息以及 18 维行为状态信息, 决策空间高达 18^{40}. 指数级增长的维度灾难问题, 使得传统的基于降维和值估计的直接策略搜索方法难以在合理时间内收敛至有效的策略解;

(2) 机器人部件之间缺乏协同合作: 当前方法主要是将整个机器人看作单一的控制对象, 未能充分利用机器人显著的结构特性, 机器人部件之间缺乏有效的信息交互和协同机制, 因而最终控制策略的最优性无法得到保证, 或者即使满足最优性评估标准, 仍然可能出现不符合实际的奇异控制行为;

(3) 学习过程和策略缺乏可解释性: 当前方法主要遵循“端到端”的黑盒训练范式, 缺乏对机器人控制过程中部件之间依赖关系的建模与刻画, 从而无法对机器人的最终控制策略以及各部件之间的潜在关联性提供明确的解释.

针对以上问题, 本文提出了一种基于结构交互驱动的机器人深度强化学习方法(structure-motivated interactive deep reinforcement learning, SMILE). 首先, 依据物理结构将单一机器人控制问题分解为多个子控制器的协同学习问题, 从而有效缓解维度灾难问题. 例如: 机械臂可分解为上臂、前臂和手掌 3 个子部件, 分别由肩关节、肘关节和腕关节连接. 每个子关节能够独立地控制部件的运动, 通过关节之间的协同决策完成预定任务. 然后, 通过两种协同图模型, 注意力模型(attention-based, ATTENTION)和动态预测模型(predicting observable dynamic topology, PODT), 用以动态推理学习过程中子控制器之间的关联关系, 并将其他控制器的影响程度嵌入本地控制器的状态表示中, 进而实现子控制器之间的协同学习. 最后, 进一步提出两种不同的协同图模型更新方法 APDODT 和 PATTENTION, 实现控制器之间长期关联关系和短期关联关系的动态自适应调整, 用以平衡 ATTENTION 和 PODT 协同图模型的计算复杂度和信息冗余度. 实验结果表明: APODT 通过将控制器之间的长期关联关系融入到短期关联关系推理过程中, 在降低信息冗余的同时, 能最大程度提升机器人控制策略的学习效率. 此外, 通过展示协同图模型的动态过程, 可反映出机器人在不同运动姿势内部关节之间的潜在依赖关系, 为最终学习策略提供更为直观和有效的解释.

本文第 1 节介绍强化学习在机器人行为控制中的研究现状. 第 2 节介绍 SMILE 的流程图及组成部分.第 3 节介绍机器人结构分解方案、实验参数设置及结果分析. 第 4 节对本文所提算法及实验进行总结.

1 研究现状

2013 年, DeepMind 公司将强化学习和深度学习相结合, 提出深度 Q 网络(deep Q network, DQN)算法[12], 标志着深度强化学习的诞生. 2015 年又提出一种 DQN 的变体: Double-DQN 算法[13], 将动作选择和价值估计分开, 用以消除过度值估计的问题. Wang 等人[14]于 2016 年提出 Dueling-DQN 算法, 将 Q 值分解为状态价值和优势函数, 从而充分利用状态和动作的信息. 虽然 DQN 及其改进算法在 Atari 等具有离散动作空间的任务

中的表现已经接近人类水平, 但此类算法难以解决具有连续动态空间的机器人行为控制问题. Lillicrap 等人[15]在 2015 年提出的深度确定性策略梯度(deep deterministic policy gradient, DDPG)算法中引入确定性策略, 并应用于连续动作空间的机器人环境. Schulman 等人[16]在 2017 年提出的近端策略优化(proximal policy optimization, PPO)算法引入 Clip 函数, 使新旧策略在一定范围内波动, 从而提升算法的稳定性. Haarnoja 等人[17]在 2018 年提出的软行动器-评判器(soft actor-critic, SAC)算法在最大化期望奖励的同时也最大化策略熵, 防止策略过早地收敛到局部最优值, 从而提升算法的探索能力. 目前, PPO 和 SAC 是大规模连续控制问题中应用最广、效果最好的两种算法.

近年来, 深度强化学习方法被广泛应用于解决机器人行为控制问题. Haarnoja 等人[18]提出一种基于 Soft *Q*-learning 算法的最大熵策略, 并应用于真实机器人行为操控. 该方法通过组合现有技能来构建新的策略, 从而大大提高训练效率. Vecerik 等人[19]提出利用 DDPG 算法得到的演示数据来解决机器人学习过程中的稀疏奖励的问题. Bai 等人[20]提出一种基于改进 PPO 算法的无人飞行器控制方法. Lin 等人[21]在 SAC 算法的基础上提出了一种样本更加高效的 DRL-EG 算法, 进一步探索最优奖励值. 以上工作在整个机器人状态和动作空间内进行端到端训练和学习, 本质上无法解决机器人的维度灾难问题. 对于更为复杂的机器人环境, 此类算法依然面临学习效率低下且容易陷入局部最优等问题.

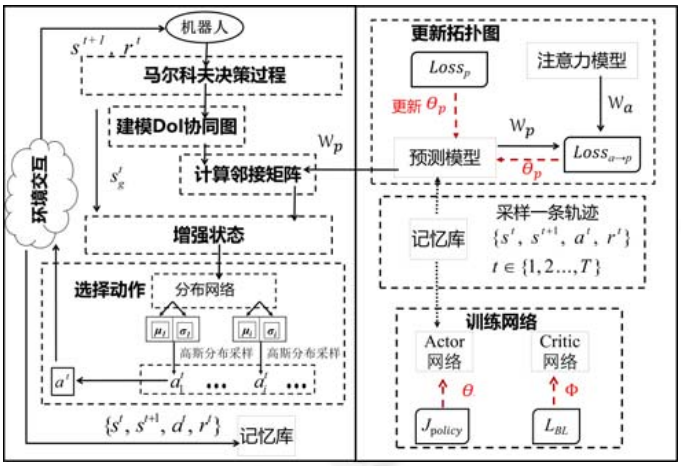
已有工作尝试从不同的角度来缓解机器人行为学习中的维度灾难和样本复杂性问题, 如多机器人并行学习[22]、人类演示学习[19]和机器人虚实迁移学习[23]等. 然而, 这些方法大都依赖额外的假设和限制(如专家演示轨迹、高逼真度的仿真平台), 方法通用性较差. 另一方面, 一些工作同样将单机器人分解成多个不同的关节, 将高维的机器人控制问题转化为低维的多关节协同控制问题, 进而缩小机器人学习训练的探索空间. 如: Dziomin 等人[24]将一个多轮移动机器人分解成不同关节并使用独立学习的方法进行训练; Busoniu 等人[25]利用集中式和分布式方法来解决两连杆机械手的控制问题, 通过对比两种学习方式的性能、收敛时间和计算条件资源, 证明了分布式的方法在更少的计算资源下能获得更好的学习性能. 这些工作大多是采用独立学习的训练方法, 机器人关节之间缺乏明确的、动态的信息交互和协同机制. 另一些研究应用图神经网络[26]或注意力机制[27,28]来学习机器人控制策略. 如: Wang 等人[26]提出了 NerveNet 方法, 其中每个控制器通过整合其邻节点的状态信息进行协同学习; Jiang 等人[27]应用注意机制来判断控制器是否应该在其可观测邻域与其他控制器进行通信. 以上工作均未讨论控制器之间的信用分配问题, 也缺乏对内部不同控制器之间依赖关系的显示量化与分析. 本文提出的方法通过计算不断变化的拓扑图建模控制器之间的动态依赖关系, 并利用控制器之间的相互重要性程度整合其他控制器的信息, 进而实现机器人内部关节之间的加权信息交互. 此外, 通过获取机器人不同运动姿势下控制器之间的关联性, 可为机器人行为决策过程中控制器之间依赖关系提供明确解释. 以上特点凸显了本文所提方法与现有方法的不同.

2 基于结构驱动的交互式学习方法

本文提出一种新型深度强化学习方法 SMILE, 用于解决机器人行为学习面临的维度灾难、部件间缺乏协同机制以及策略不可解释等问题. 如图 1 所示, 该算法由运动阶段(见第 2.1 节)和训练阶段(见第 2.2 节)组成.

运动阶段的功能是实现不同控制器(即关节)之间的信息交互, 并获得每个控制器执行的动作: 首先, 利用结构分解技术将整体机器人分解成多个控制器, 并提出信息交互程度(degree of interaction, DoI)的概念来建模不同控制器之间的依赖关系; 然后, 两种不同协同图模型 ATTENTION 和 PODT 来动态推理控制器之间的 DoI 值, 并将不同控制器的 DoI 嵌入到本地控制器的状态表示中, 从而实现控制器之间的信息传递; 最后, 每个控制器分别利用一个基于高斯分布采样的 Actor 网络选择动作并与环境交互;

训练阶段的功能是学习整体机器人的策略和动态更新协同图: 利用 PPO 算法[16]训练 Actor-Critic 网络, 并利用 Actor 网络学习的策略梯度指导协同图网络的更新.



(a) 运动阶段 (b) 训练阶段

图 1 SMILE 框架图

2.1 运动阶段

运动阶段首先利用机器人结构分解技术实现对整体机器人的行为/状态空间的分解并降维, 即: 将机器人的全局状态空 $s\in R^M$ 和动作空间 $a\in R^N$ 分解为一系列较小的子空间, 每个子空间由一个关节控制器管理, 具体表示为: $s_1,\ldots,s_l,\ldots,s_n,s_g=Divide(s)$和 $a_1,\ldots,a_l,\ldots,a_n,a_g=Divide(a)$, 其中, *Divide* 是结构分解器, s_g 代表被所有控制器共享的状态信息, n 是控制器的数量, s_l 和 a_l 分别表示第 l 个控制器的状态和动作信息. 不同的控制器在运动控制中扮演不同作用, 且控制器之间的依赖关系也在动态变化. 因此, 该阶段提出信息交互模型和协同图模型获取不同控制器之间的信息交互程度, 从而实现了机器人内部关节的协同控制.

2.1.1 信息交互模型

提出信息交互程度 DoI 的概念来建模不同控制器之间的依赖关系, 即关节之间的依赖程度的量化值. 具体表示为 $G=(V=\{A_i|i\in[n],W=\{w_{ij}|i,j\in[n]\})$, 其中, G, V 和 W 分别代表协同图、控制器集合和邻接矩阵(即协同图任意两个控制器之间权重的集合); A_i 代表第 i 个控制器; w_{ij} 表示是控制器 A_j 对 A_i 的重要程度, 即 DoI 的值. 根据邻接矩阵 W 的不同, 定义了 4 种不同的信息交互程度.

(1) 全局交互程度(global DoI, GDoI): 所有的权重 $w_{ij}=1$, 即每个控制器都完全了解全局状态信息, 并且认为其他所有控制器的信息同等重要;

(2) 独立交互程度(independent DoI, IDoI): $w_{ij}=1$ 且 $w_{ij}=0$, 即控制器仅可以获得自身信息;

(3) 动态交互程度(dynamic DoI, DDoI): w_{ij} 不是常数且动态变化, 即每个控制器都能动态获取来自其他控制器的信息;

(4) 物理交互程度(physical DoI, PDoI): $w_{ij}=1$ 且真实物理拓扑中控制器之间的权重 $w_{ij}=1$, 即每个控制器都可获得自身状态信息和真实物理连接控制器的信息.

2.1.2 协同图模型

基于 Attention 思想和状态预测思想, 分别提出了 ATTENTION 和 PODT 协同图模型, 用以具体计算机器人内部关节的动态协同图 G 和邻接矩阵 W(如图 2 所示). ATTENTION 的核心思想是: 利用神经网络输出正在查询的控制器 $\boldsymbol{Q}$ 和其他非查询控制器 $\boldsymbol{K}$ 之间的相关性, 从而建立动态协同图. 每个控制器的局部状态信息被输入到一个特征映射网络(F_{in}), 从而输出一个维度为 b 的特征向量 $f_l\in R^b$: $f_l=F_{in}(s_l)$; 查询的控制器 A_l 与任一控制器 A_j 的联合特征向量$\langle f_l,f_j\rangle$被传入到一个注意力网络中, 从而输出和它们之间的相似性值, 即 K_{lj}; 利用 *Softmax* 公式正规划, 获得控制器 A_j 对 A_i 的相关性权重 $w_{lj}^a=e^{K_{ll}^T\cdot K_{lj}}\Big/\sum_{k=1}^{n}e^{K_{ll}^T\cdot K_{lk}}$.

PODT 模型的流程图如图 2 右图所示, 其核心思想是: 利用预测状态和真实状态的差异来定义每个控制器

之间的相关性, 从而建立协同图 G_{PCG}. 将控制器 A_l 的状态 s_l 输入到一个状态预测网络 $\boldsymbol{P}$, 从而预测其他控制器的状态信息. 控制器 A_j 的真实状态 s_j 和预测状态 $P(s_l)[j]$之间的预测误差 e_{lj} 可以表示为: $e_{lj}=\|s_j-P(s_l)[j]\|_2$; 任意两个控制器之间的长期预估误差 E_{lj} 可以表示为一条连续轨迹 $\{s_l^t, s_l^{t+1}\}$, $t\in\{1,2,...,T\}$ 中预估误差 e_{lj} 的均值: $E_{lj}=\frac{1}{T}\sum_{t=1}^{T}e_{lj}$; 对于每个控制器 A_l, 预测模型仅保留最大的 K 个长期预估误差 E_{lj}, 而其他的值被初始化为 0, 其中, K 是一个超参数, 控制每个控制器的边数. 将选定的 K 个相关性权重正规化, 从而获得邻接矩阵 W_{K_PCG}; 最后, 为了同时考虑机器人的物理结构, PODT 模型利用机器人物理拓扑 W_p 来进一步矫正邻接矩阵 W_{K_PCG}, 从而获得最终的 PODT 的邻接矩阵 W_{PCG}: $W_{PCG}=\eta W_{K\_PCG}+(1-\eta)*W_p$, 其中, η代表控制权衡选择偏差的超参数(根据机器人物理拓扑定义 W_p, 即: 当两个关节真实连接时, W_p 中对应的边权重赋值为 $1/n$, 否则赋值为 0).

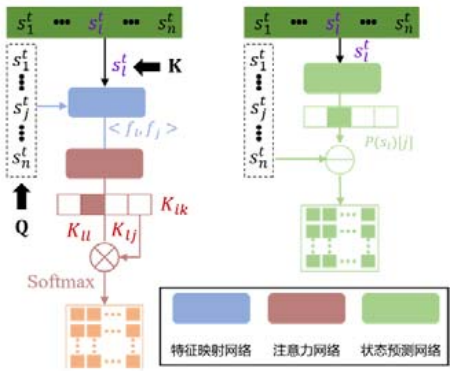


图 2 ATTENTION(左图)和 PODT 模型(右图)

综上, 图 3 展示了 SMILE 运动阶段的具体流程图.

- 步骤 1. 控制器 A_l 的局部状态 s_l 和全局状态信息 s_g 分别被输入到一个特征映射网络(F_{in})和一个全局特征映射网络(F_g), 输出特征向量 $f_l\in R^b$ 和全局特征向量 $\hat{f}_g\in R^b$, 如下: $f_l=F_{in}(s_l)$, $\hat{f}_g=F_g(s_g)$;
- 步骤 2. 利用 DoI 来整合其他控制器的状态信息, 从而计算每个智能的增强状态 $\hat{s}_l$. 此步骤实现了控制器之间的信息传递: $\hat{f}_l=CONCAT(f_1*w_{l1},...,f_j*w_{lj},...,f_n*w_{ln})$, $\hat{s}_l=\hat{f}_l+\hat{f}_g$, 其中, $\hat{f}_l$ 代表增强特征; *CONCAT* 代表一个连接器; n 是控制器的数量; w_{lj} 表示控制器 A_l 对控制器 A_j 的依赖程度, 即 DoI 值;
- 步骤 3. 利用一个高斯多层神经网络(Gaussian MLP, Γ)来逼近动作的生成过程, 输入为每个控制器的增强状态, 输出为高斯分布采样的均值μ_l 和方差σ_l, 如下: $\mu_l,\sigma_l=\Gamma(\hat{s}_l)$;
- 步骤 4. 利用高斯采样的方法获得每个控制器将执行的动作. 此时, 整体机器人执行所有控制器的联合动作 a 可以表示为: $a=CONCAT(a_1,\ldots,a_l,\ldots,a_n)$.

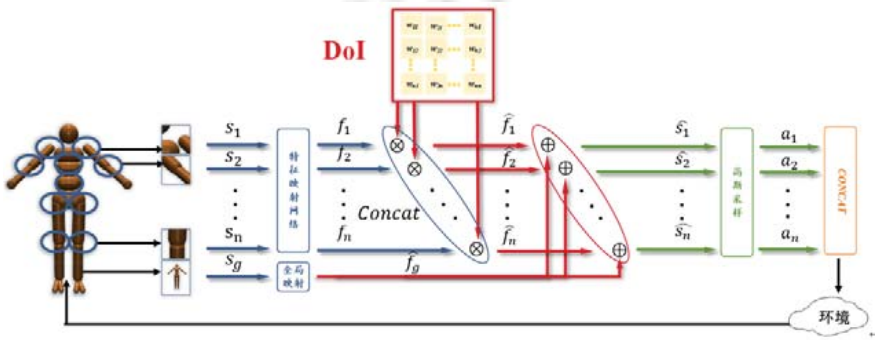


图 3 SMILE 运动阶段的流程图

2.2 训练阶段

2.2.1 策略学习模型

SMILE 的策略学习模型采用 PPO 算法[16]的思想来实现, 其中, Actor 网络(即高斯多层神经网络)的输入为对应控制器的增强状态 $\hat{s}_l$, 输出为该控制器行为高斯分布的均值和方差. Actor 网络用于学习对应控制器的最优策略, 通过更新参数θ来最大化折扣回报 J_{policy}; Critic 网络的输入为机器人的全局状态 s, 输出为预估回报 $V_\Phi(s)$, 用于评估整体机器人的累计折扣奖励值, 通过更新参数Φ来最小化预估回报损失 L_{BL}, 如下:

$$J_{policy}=\frac{1}{T}\sum_{t=1}^{T}\min\left\{\frac{\pi_\theta(a\mid s)}{\pi_\theta^{old}(a\mid s)}\hat{A},clip\left(\frac{\pi_\theta(a\mid s)}{\pi_\theta^{old}(a\mid s)},1-\varepsilon,1+\varepsilon\right)\hat{A}\right\} \tag{1}$$

$$L_{BL}=-\frac{1}{T}\sum_{t=1}^{T}\hat{A}=-\frac{1}{T}\sum_{t=1}^{T}(\sum_{t'>t}\gamma^{t'>t}r^{t'}-V_\Phi(s))^2 \tag{2}$$

其中, ε为一个超参数.

2.2.2 协同图更新模型

ATTENTION 模型实时输出协同图 G_{ACG}, 利用短期(一个 step)的状态信息来计算邻接矩阵 W_{ACG}, 从而获得精准的 DoI. 该模型的缺点是计算量大, 而且仅考虑短期运动过程中的权重信息, 导致 DoI 值变化较快. 相反, PODT 模型离线输出协同图 G_{PCG}, 融合了长期(一个 episode)的状态信息来计算邻接矩阵 W_{PCG}, 可以获得更加稳定的 DoI 值. 该模型的缺点是长期估计误差 E_{lj} 会导致信息缺失. 因此, DoI 值仅仅是控制器之间依赖关系的近似估计值. 为了平衡长期状态信息和短期状态信息(即权衡计算复杂度和信息冗余度), 提出了两种不同的协同图模型更新方法 APOTD 和 PATTENTON, 来输出更加精准的和稳定的 DoI. 同时, 为了考虑整体机器人的策略性能, 利用策略梯度 J_{policy} 进一步矫正协同图网络.

APODT 的核心思想是: 利用 PODT 模型输出 DoI 值, 同时利用 ATTENTION 模型的邻接矩阵 W_{ACG} 动态矫正 DoI, 如图 4(左)所示. 具体流程如下.

- 步骤 1. 在每个回合后, 利用 PODT 输出 DoI, 然后利用损失函数 L_{θ_p} 来更新状态预测网络 $\boldsymbol{P}$, 如下:

$$L_{\theta_p}=\sum_{j=1}^{n}\sum_{l=1}^{n}(s_j-P(s_l)[j])^2 \tag{3}$$

- 步骤 2. 每隔若干个回合后, 利用 W_{ACG} 和 W_{PCG} 的均方差计算损失函数 $loss_{p\leftrightarrow a}$ 来更新注意力网络的参数θ_a, 如下所示:

$$loss_{p\leftrightarrow a}=\|W_{ACG}-W_{PCG}\|_2 \tag{4}$$

- 步骤 3. 为了动态探索最优的协同图和邻接矩阵, 利用机器人的整体策略 J_{policy} 和损失函数 $loss_{p\leftrightarrow a}$ 的组合来更新状态预测网络的参数θ_p, 梯度公式如下所示:

$$g_\Theta=\nabla_\Theta J_{policy}+\tau\nabla_\Theta loss_{p\leftrightarrow a},\ \text{for}\ \Theta\in\theta_p \tag{5}$$

其中, τ为平衡的超参数, 用于控制修正的更新梯度.

PATTTENTION 模型的核心思想是: 利用 ATTENTION 模型输出 DoI 值, 同时利用 PODT 模型的邻接矩阵 W_{PCG} 动态矫正 DoI, 如图 4(右)所示. 具体流程如下.

- 步骤 1. 机器人运动的每一步时, 利用 ATTENTION 模型计算 DoI;
- 步骤 2. 每个回合后, 利用 PODT 模型输出邻接矩阵 W_{PCG}, 并利用损失函数 L_{θ_p} 来优化状态预测网络 $\boldsymbol{P}$ 的参数θ_p;
- 步骤 3. 为了动态探索最优的协同图和邻接矩阵, 利用策略梯度 J_{policy} 和损失函数 $loss_{p\leftrightarrow a}$ 的组合来更新注意力网络的参数θ_a, 梯度公式如下所示:

$$g_\Theta=\nabla_\Theta J_{policy}+\tau\nabla_\Theta loss_{p\leftrightarrow a},\ \text{for}\ \Theta\in\theta_a \tag{6}$$

APODT 方法融合了 ATTENTION 和 PODT 的优点, 相比 PATTENTION 具有更低的计算复杂度. 表 1 给出了 APODT 的伪代码: 首先, 初始化策略参数θ、注意力网络的参数θ_a、状态预测网络 $\boldsymbol{P}$ 参数θ_p, 初始化邻接矩阵 W_{PCG} 等参数(第 1 行); 在每个训练的时间步 t 时, 机器人的全局状态被划分成若干控制器的局部状态(第

2 行–第 4 行); 然后, 利用 DoI 计算增强状态并采样每个控制器的动作(第 5 行–第 9 行); 机器人执行联合动作 a 并与环境交互(第 10 行–第 13 行); 利用 PODT 输出 DoI 并更新状态预测网络(第 14 行、第 15 行); 训练 Actor 网络和 Critic 网络(第 16 行); 利用 ATTENTION 模型计算邻接矩阵 W_{ACG} 并计算损失函数 $loss_{p\leftrightarrow a}$(第 18 行、第 19 行); 最后, 利用机器人的整体策略 J_{policy} 和损失函数 $loss_{p\leftrightarrow a}$ 对 DoI 值进行更新(第 19 行、第 20 行).

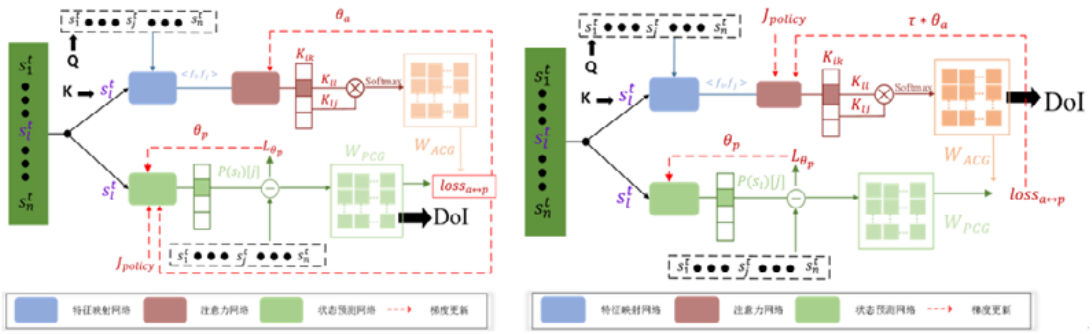


图 4 APODT 模型(左)和 PATTENTION 模型(右)

表 1 APODT 算法

算法 1. APODT.

初始化状态 $s\in R^M$, 动作 $a\in R^N$, 奖励衰减因子 γ, 学习率 λ, 运动步长 T, Actor 网络参数 θ, Critic 网络参数 Φ, 初始化 W_{PCG}, 注意力网络的参数 θ_a, 状态预测网络 $\boldsymbol{P}$ 参数 θ_p;
重复 $ep\in 1,\ldots,episode$:
  重复 $t=1,\ldots,T$:
    划分机器人的状态 s 为: $s_1,\ldots,s_l,\ldots,s_n,s_g$;
    重复 $l<n$:
      特征映射 $f_l=F_{in}(s_l)$;
    结束循环
    利用特征映射网络和全局特征映射网络获得增强特征 $\hat{f}_l$ 和全局特征向量 $\hat{f}_g$;
    获得增强状态 $\hat{s}_l=\hat{f}_l+\hat{f}_g$ 并且拟合高斯分布 $\mu_l,\sigma_l=\Gamma(\hat{s}_l)$;
    机器人执行动作 $a=CONCAT(a_1,\ldots,a_l,\ldots,a_n)$;
    与环境交互, 获得下一步状态和奖励值, 并储存到记忆库中;
结束循环
利用 PODT 模型生成协同图 G_{PCG} 和 DoI(邻接矩阵 W_{PCG});
更新状态预测网络 $\boldsymbol{P}$ 的参数 θ_p: $L_{\theta_p}=\sum_{j=1}^{n}\sum_{l=1}^{n}(s_j-P(s_l)[j])^2$;
调用公式(1)和公式(2), 分别训练 Actor 网络和 Critic 网络;
如果 $ep\%Batch==1$, 则:
  利用 ATTENTION 模型计算邻接矩阵 W_{ACG};
  计算损失函数 $loss_{p\leftrightarrow a}=\|W_{ACG}-W_{PCG}\|_2$;
  更新状态预测网络 $\boldsymbol{P}$ 的参数 θ_p: $g_\Theta=\nabla_\Theta J_{policy}+\tau\nabla_\Theta loss_{p\leftrightarrow a}$, for $\Theta\in\theta_p$.

3 实验和结果

3.1 实验环境

实验环境基于 MuJoCo 平台, 其中包含多款机器人环境, 如 Swimmer, Hopper, Walker, Half-Cheetah 和 Humanoid 等. 不同机器人环境中的状态维度、动作维度和自由度等信息见表 2.

表 2 不同机器人的复杂度

机器人环境	状态维度(S)	动作维度(A)	自由度	连接杆	运动是否终止
Swimmer	8	2	2	3	否
Hopper	11	3	3	4	是
Walker	17	6	6	7	是
Half-Cheetah	17	6	6	9	否
Humanoid	342	17	17	13	是

图 5 给出了以上 5 种机器人环境的结构分解图. 以 Half-Cheetah 为例, 即图 5(d)和图 5(i), 机器人被分为 6 个关节(即控制器). 设 $State=(p_0,p_1,\ldots,p_7,v_8,v_9,\ldots,v_{16})$和 $Action=(a_0,a_1,a_2,a_3,a_4,a_5)$分别表示 17 维的状态和 6 维的动作, 其中, $P_m(m\in[0,7])$, $v_n(n\in[8,16])$和 $a_k(k\in[0,5])$分别表示机器人的位置、速度和关节的角度. 经结构分解后, (p_0,p_1,v_8,v_9,v_{10})代表控制整 Half-Cheetah 机器人位置或速度的全局状态; p_2, p_3, p_4, p_5, p_6 和 p_7 分别控制控制器 A_1–A_6 的位置; v_{11}, v_{12}, v_{13}, v_{14}, v_{15} 和 v_{16} 分别控制控制器 A_1–A_6 的速度; a_0, a_1, a_2, a_3, a_4 和 a_5 分别控制控制器 A_1–A_6 的动作. 此时, Half-Cheetah 的每个控制器在二维的状态空间和一维的动作空间中进行决策, 这与直接在原始的状态和动作空间中搜索相比大大降低了复杂度. 表 3 给出了分解后每个控制器管理的状态和动作的维度信息.

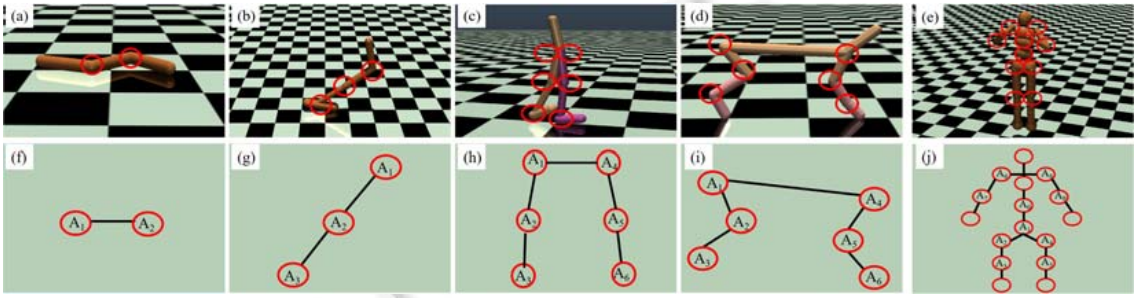


图 5　4 种不同机器人环境的结构分解图解

表 3　状态和动作空间的分解

环境	Swimmer		Hopper		Walker		Half-Cheetah		Humanoid	
控制器	状态空间	动作空间	状态空间	动作空间	状态空间	动作空间	状态空间	动作空间	状态空间	动作空间
A_1	(p_2,v_{11})	(a_0)	(p_2,v_{11})	(a_0)	(p_2,v_{11})	(a_0)	(p_2,v_{11})	(a_0)	(p_5,p_6,p_{28},v_{29})	(a_0,a_1)
A_2	(p_3,v_{12})	(a_1)	(p_3,v_{12})	(a_1)	(p_3,v_{12})	(a_1)	(p_3,v_{12})	(a_1)	(p_7,v_{30})	(a_2)
A_3			(p_4,v_{13})	(a_2)	(p_4,v_{13})	(a_2)	(p_4,v_{13})	(a_2)	$(p_8,p_9,p_{10},v_{31},v_{32},v_{33})$	(a_3,a_4,a_5)
A_4					(p_5,v_{14})	(a_3)	(p_5,v_{14})	(a_3)	(p_{11},v_{34})	(a_6)
A_5					(p_6,v_{15})	(a_4)	(p_6,v_{15})	(a_4)	$(p_{12},p_{13},p_{14},v_{35},v_{36},v_{37})$	(a_7,a_8,a_9)
A_6					(p_7,v_{16})	(a_5)	(p_7,v_{16})	(a_5)	(p_{15},v_{38})	(a_{10})
A_7									$(p_{16},p_{17},v_{39},v_{40})$	(a_{11},a_{12})
A_8									(p_{18},v_{41})	(a_{13})
A_9									$(p_{19},p_{20},v_{32},v_{43})$	(a_{14},a_{15})
A_{10}									(p_{21},v_{44})	(a_{16})

3.2 参数设置

在型号为 Xeon E5-2630 v3 的 CPU 上进行训练和测试; 显存为 3839MB, GPU 型号为 Nvidia Quadro K2200 (4GB/Nvidia), RAM 内存容量为 32GB. 本文提出的 SMILE 包含 7 种不同的交互程度的方法: GDoI, IDoI, PDoI 以及 4 种 DDoI(ATTENTION, PODT, APODT 和 PATTENTION). 将机器人获得的平均奖励值收敛速度或累积奖励值的大小作为算法的评估标准. 对比主要强化学习算法包括 CEM 算法[29]、REINFORCE 算法[30]、AC 算法[31]、DDPG 算法[15]、PPO 算法[16]和 SAC 算法[17]. 表 4 和表 5 分别给出了 SMILE 和现有 DRL 算法的主要参数. 表 6 给出了 SMILE 中神经网络的参数设置. 仿真环境每个运行持续 2 000 回合(Swimmer 除外, 为 500 回合). 机器人连续运动 300 步或直到摔倒后一个回合结束. 为了消除实验的随机性, 结果为独立运行 5 次的平均值.

表 4　SMILE 中主要参数

参数	数值	含义
Torch.seed	6	CPU 的随机种子
Maxstep	300	行走步长上限
Env.seed	2 180	机器人环境的随机种子

表 4 SMILE 中主要参数(续)

参数	数值	含义
Episode	2 000	机器人行走的回合数
Replay	10 000	经验池
Sample	32	PODT 中采样大小
Batch	50	小批量数据大小
Runs	6	算法被执行的次数
M or B	10	Actor 或 Critic 的小批量数据
T	10	每隔 T 步计算优势函数
K	3	PODT 中选择最大边的数量
η	0.1	PODT 权重矫正参数
γ	0.99	奖励值衰减
λ	e^{-4}	学习率
τ	0.01	APODT 中的超参数

表 5 不同 DRL 算法中的参数

参数	AC	CME	DDPG	REINFORCE	PPO
γ	$1e^{-2}$	$1e^{-4}$	$1e^{-3}$	$2e^{-2}$	$2e^{-3}$
λ	0.99	0.9	0.99	0.995	0.99
Hidden layer	32	10	400	20	64
Replay	None	$5e^{5}$	$5e^{4}$	None	$1e^{4}$
Batch	None	150	64	None	32
ε	None	None	None	None	0.2

表 6 SMILE 中神经网络的设置

神经网络	输入维度	隐藏层维度	输出维度
F_g	s_g	None	36
Actor (Gaussian MLP)	36	None	2
F_{in}	S	36	36 (b)
Attention MLP	36	None	10 (h)
$\boldsymbol{P}$	S	128	S
Critic	S	64	1

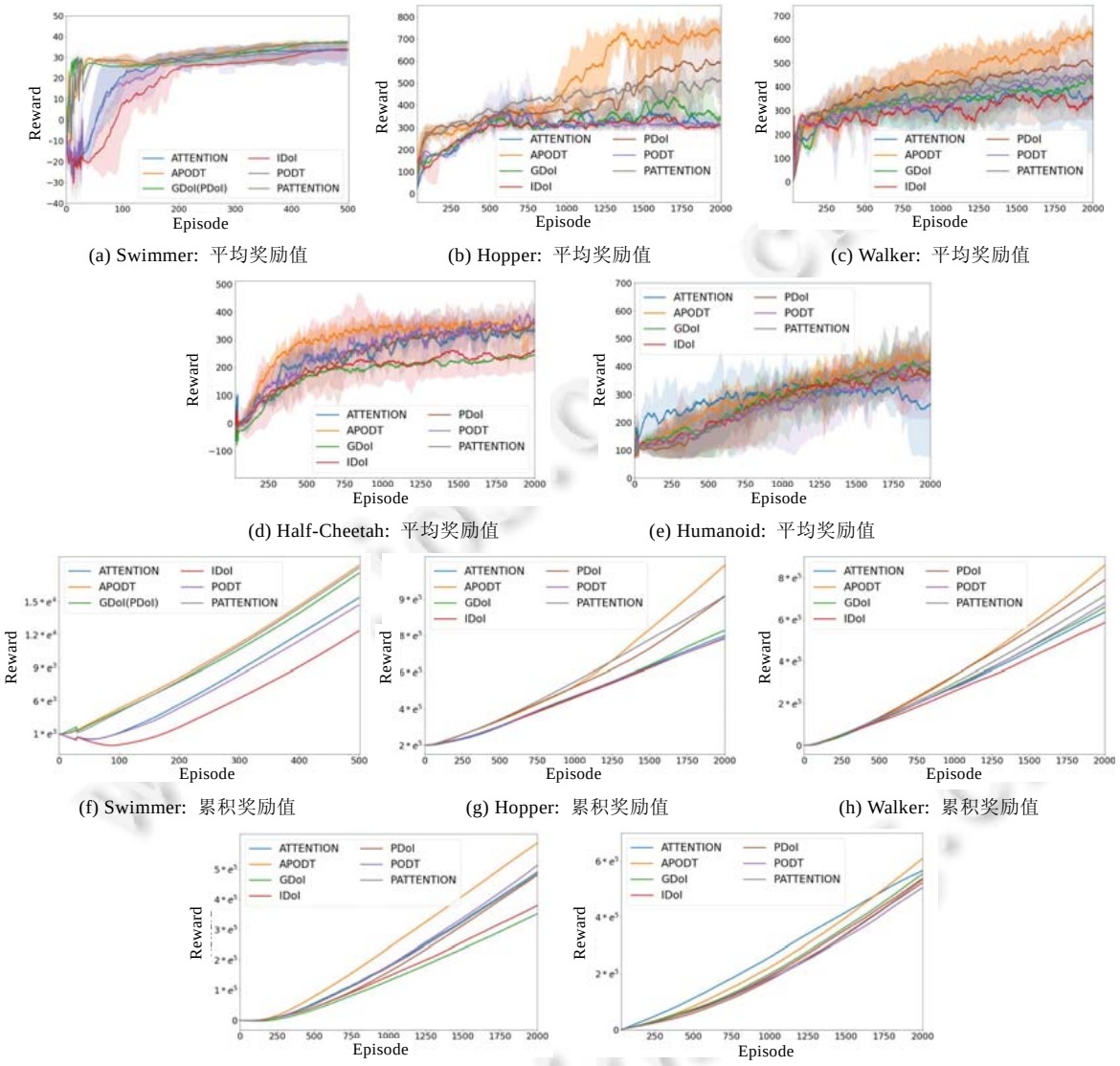
3.3 实验结果

3.3.1 不同 DoI 计算方法的对比

图 6 分别对比了不同的 DoI 计算方法在 5 种机器人环境的平均奖励值和累积奖励值. 在最简单的机器人环境 Swimmer 中, PATTENTION 和 APODT 的性能接近, 略优于 GDoI(PDoI)算法; 三者的性能明显优于 ATTENTION, PODT 和 IDoI. 在 Hopper 和 Walker 中, PATTENTION 的性能明显低于 APODT 和 PDoI, 但依旧优于 GDoI. 在 Half-Cheetah 中, PDoI, ATTENTION 和 PODT 的性能与 PATTENTION 接近, 且明显优于 GDoI 和 IDoI. 以上结果表明, 控制器之间的信息交互有助于学习性能的提升. 相反, 由于控制器之间缺乏协同和信息传递, IDoI 在所有算法中的性能表现最差. 在 GDoI 中, 所以控制器的状态信息被同等地加权到本地状态嵌入表示中, 而机器人的一些控制器之间可能并不需要信息共享进行协同, 冗余的状态信息将可能导致学习效率的下降.

在最为复杂的类人机器人环境中, APODT 和 PATTENTION 效果前期低于 ATTENTION, 但是后期可以探索到更优的控制策略; ATTENTION 和 PODT 的表现相对较差, 后期甚至低于 IDoI; GDoI, IDoI 和 PDoI 的效果相近, 且明显低于 APODT.

4 种不同的 DDoI 算法都能够捕获控制器之间的动态依赖关系, 但是在不同的机器人环境中具有明显的差异. 其中, ATTENTION 方法需要实时在线地计算获得精准的 DoI, 较高的计算复杂度, 极大地影响了学习效率; PODT 方法可以离线地计算权重获得近似估计的 DoI 值, 因而计算复杂度相对较小, 但与此同时, 也可能会导致重要协同信息的缺失, 从而陷入局部最优. 因此, ATTENTION 和 PODT 仅适用于较简单的机器人环境(即 Hopper, Walker 和 Half-Cheetah), 而在更为复杂的环境如 Humanoid 机器人中不再具有优势. 相对于 IDoI 和 GDoI, PDoI 掌握了机器人的物理拓扑信息, 有效地解决了信息传递和信息冗余的问题, 从而对简单的机器人环境(即 Hopper, Walker 和 Half-Cheetah)的学习效果具有明显提升. 但是, 在具有复杂物理拓扑结构的 Humanoid 环境中, 仅仅依靠物理信息不足以学习到最优策略. 由此可见, 信息冗余、信息缺失和物理连接信息都会对学习性能产生显著影响. 因此, 一个协同图模型精准高效的更新方法至关重要. APODT 方法结合 PDoI, ATTENTION 和 PODT 的优点, 在低计算复杂度和低信息冗余度之间进行平衡, 从而获得了更加精准的 DoI 值来进行信息传递. 与此同时, APODT 还可以获得物理连接之外的重要性关系. 因此, APODT 在所有机器人环境中均能获得最优的性能表现.



(a) Swimmer: 平均奖励值 (b) Hopper: 平均奖励值 (c) Walker: 平均奖励值

(d) Half-Cheetah: 平均奖励值 (e) Humanoid: 平均奖励值

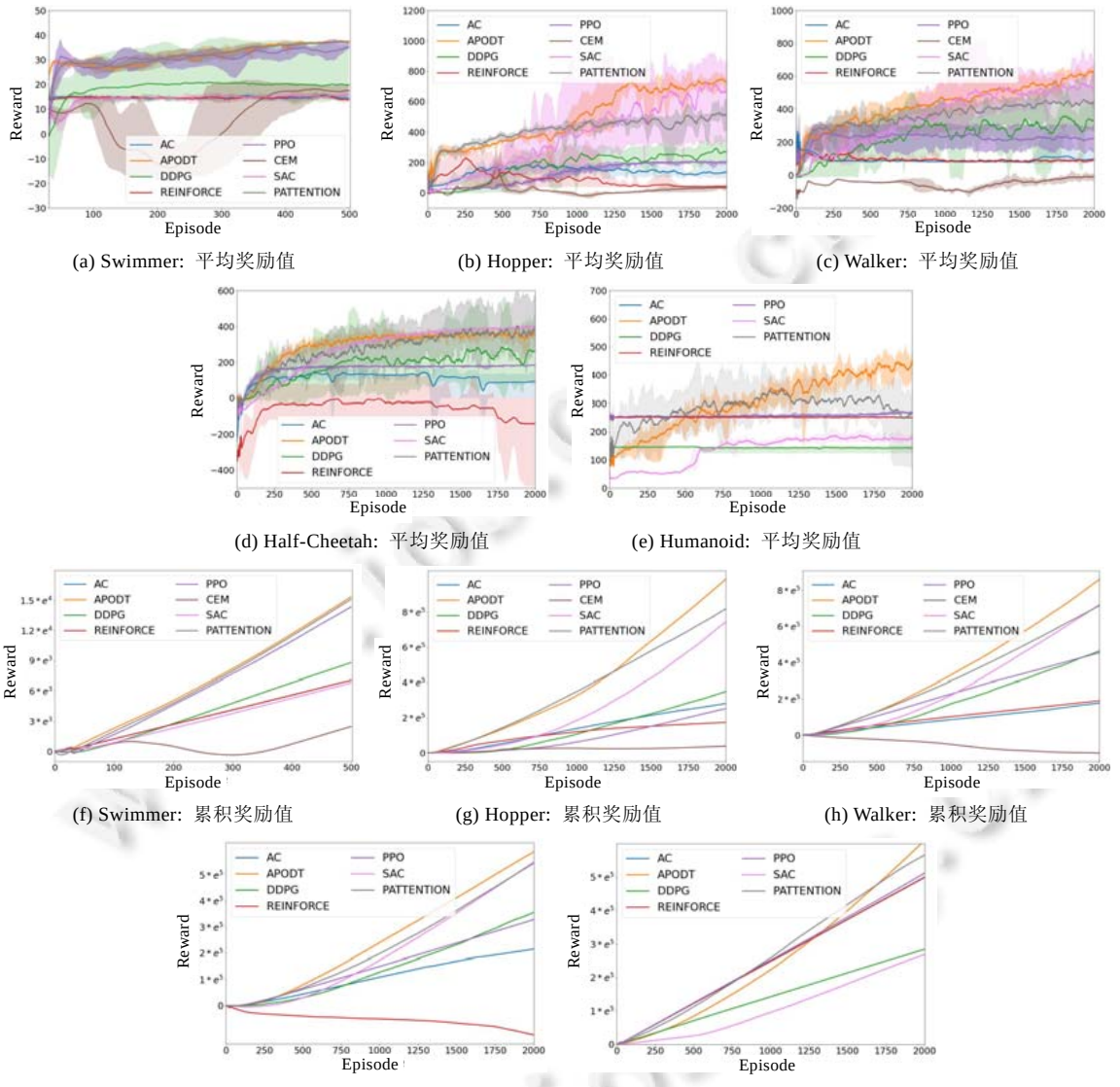
(f) Swimmer: 累积奖励值 (g) Hopper: 累积奖励值 (h) Walker: 累积奖励值

(i) Half-Cheetah: 累积奖励值 (j) Humanoid: 累积奖励值

图 6 不同交互程度的算法在 5 种机器人环境中平均奖励值和累积奖励值的对比

3.3.2 不同 DRL 算法的对比

图 7 将 APODT 和 PATTENTION 与现有的 DRL 算法进行对比. 由图可知: 在低维环境(如 Swimmer)中, APODT 和 PATTENTION 的性能略优于 PPO, 但明显优于其他算法; 在 Hopper, Walker 和 Half-Cheetah 环境中, APODT 的性能明显高于所有算法; PATTENTION 在前期具有优势, 但后期表现低于 SAC; 在 Humanoid 环境中, 传统的强化学习算法表现极差, 极易陷入局部最优且收敛缓慢. 综上所述, 在所有环境中, APODT 可以最快学到最优策略, 证明了机器人内部关节的信息传递对学习性能的提升.



(a) Swimmer: 平均奖励值 (b) Hopper: 平均奖励值 (c) Walker: 平均奖励值

(d) Half-Cheetah: 平均奖励值 (e) Humanoid: 平均奖励值

(f) Swimmer: 累积奖励值 (g) Hopper: 累积奖励值 (h) Walker: 累积奖励值

(i) Half-Cheetah: 累积奖励值 (j) Humanoid: 累积奖励值

图 7 不同深度强化学习算法在 5 种机器人环境中平均奖励值和累积奖励值的对比

此外, 表 7 对比了机器人不同学习的阶段的平均奖励值(不同回合的最高奖励值以粗体标记). 4 种 DDoI 方法由于需要动态地计算协同图, 在学习初期通常不具有优势. 经过初期的试错期后, 4 种 DDoI 方法(尤其是 APODT)在后期明显优于其他算法.

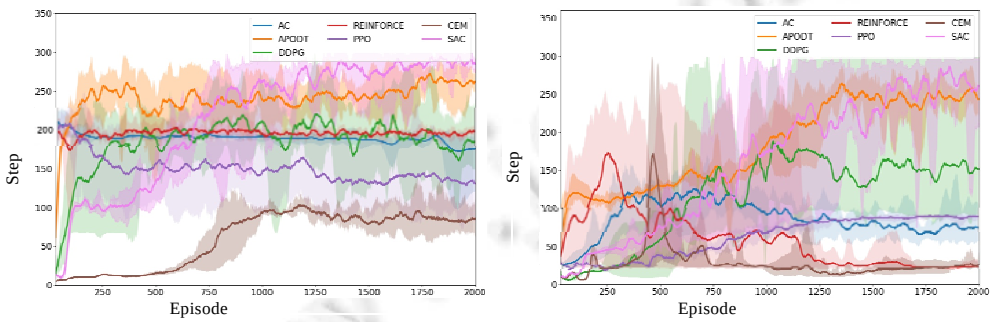
图 8 对比了 APODT 和不同 DRL 算法在 Hopper, Walker 和 Humanoid 环境中的运动步长. 在 3 种环境中, APODT 收敛最快且性能最好. 即: 机器人最快学会连续奔跑, 且能持续较长时间. 由于 Swimmer 和 Half-Cheetah 不存在任务终止的情况, 每回合都可以执行到最大步数(300 步), 因此运动步长在这两种环境中不具有对比意义.

表 7 不同算法在不同学习阶段的奖励值

环境	Episode	AC	CEM	DDPG	PPO	REINFORCE	SAC
Swimmer	1	**17.0±8.1**	−11.1±7.6	0.9±5.0	−34.1±2.1	15.3±3.7	−9.4±1.3
	1 000	14.25±0.9	−10.2±6.7	20.5±5.0	31.1±4.3	14.2±1.7	14.7±0.4
	2 000	14.37±1.0	17.5±1.7	19.7±6.6	35.2±1.0	13.7±1.1	15.9±0.4
Hopper	1	41.9±1.0	−2.8±0.7	1.8±0.0	35.5±0.5	43.6±1.6	8.4±5.2
	1 000	155.2±44.3	−11.4±17.1	185.4±176.1	147.1±17.6	82.7±44.1	368.0±204.4
	2 000	134.7±31.1	34.8±15.1	377.0±98.4	202.2±10.0	3.5±3.5	667.9±319.7
Walker	1	73.3±15.3	−64.9±3.2	1.8±0.0	**74.1±14.8**	68.8±15.8	−2.9±2.6
	1 000	86.4±8.8	206.4±42.0	310.0±115.7	289.6±74.1	90.0±11.7	419.1±121.4
	2 000	100.7±33.9	246.7±27.8	330.3±66.0	270.0±14.9	93.3.3	528.5±131.7
Half-Cheetah	1	240.9±12.1	44667.1±57700.6	**1.9±0.1**	**74.1±14.8**	292.3±13.2	−130.8±52.1
	1 000	126.6±27.9	83879.5±5557.3	229.8±224.3	289.6±74.1	25.5±94.1	336.5±9.2
	2 000	91.2±92.6	44861.1±6547.5	261.3±183.2	270.0±14.9	139.1±352.7	**399.4±8.4**
Humanoid	1	255.8±10.0		163.1±1.4	**258.1±10.2**	255.8±10.0	63.1±6.4
	1 000	251.2±1.4	None	142.1±4.7	253.2±5.9	250.8±1.3	154.1±9.1
	2 000	251.0±0.8		142.1±10.8	269.4±5.1	250.6±0.9	179.5±9.5

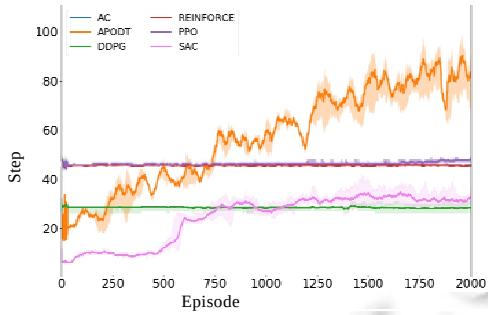
表 7 不同算法在不同学习阶段的奖励值(续)

环境	GDoI	IDoI	PDoI	ATTENTION	PODT	PATTENTION	APODT
Swimmer	−9.8±10.03	−6.8±13.0	−9.8±10.3	−7.5±6.5	18.1±3.4	−11.8±3.5	−11.1±2.4
	29.1±1.4	29.0±1.3	29.1±1.4	**33.6±7.3**	29.8±5.2	29.6±4.4	32.8±2.3
	37.0±1.8	37.7±1.5	37.0±1.8	34.9±7.5	33.2±7.2	34.3±3.3	**37.4±0.3**
Hopper	**69.1±3.2**	67.8±1.9	60.8±5.6	68.4±2.5	65.9±2.3	68.8±2.8	66.9±2.3
	294.3±2.9	329.6±31.8	362.4.3±46.6	327.6±28.2	320.5±60.9	415.2±44.4	**396.2±83.0**
	402.3±84.2	338.2±35.5	594.2±163.8	311.6±19.4	332.6±37.5	512.7±108.6	**675.1±53.3**
Walker	−8.2±1.2	−8.2±1.2	−5.6±0.7	−8.3±1.5	−8.3±1.5	−8.3±1.4	−8.3±1.4
	349.4±126.1	304.6±22.9	408.5±166.9	279.5±140.8	339.5±51.0	369.3±37.6	**453.9±89.7**
	410.9±39.8	359.0±23.0	486.9±222.7	349.1±222.7	358.9±37.5	445.8±131.2	**627.1±31.1**
Half-Cheetah	−87.6±67.5	119.0±35.7	−100.5±45.6	−119.0±45.2	107.0±52.4	−114.0±27.0	114.0±27.0
	284.5±69.9	319.9±17.7	287.9±65.5	212.3±87.2	275.2±46.1	344.1±51.5	**344.1±81.0**
	308.1±54.6	334.9±44.8	358.5±90.6	248.5±104.4	372.0±24.8	353.1±29.3	353.1±32.3
Humanoid	100.6±1.7	100.6±1.6	98.5±12.3	103.2±4.4	102.2±3.3	102.0±3.6	100.0±1.1
	289.1±64.5	292.7±46.9	286.7±16.1	317.9±83.1	249.6±92.3	282.5±65.5	**316.6±26.2**
	397.0±76.0	378.5±61.2	377.1±58.4	266.2±71.8	349.7±73.4	417.2±82.7	**445.1±35.0**



(a) Walker (b) Half-Cheetah

图 8 APODT 与不同算法中在 3 种不同机器人中平均运动步数的对比

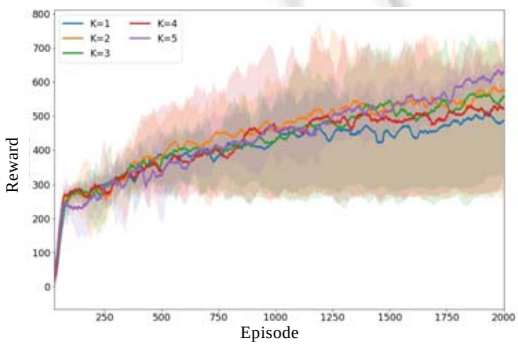


(c) Humanoid

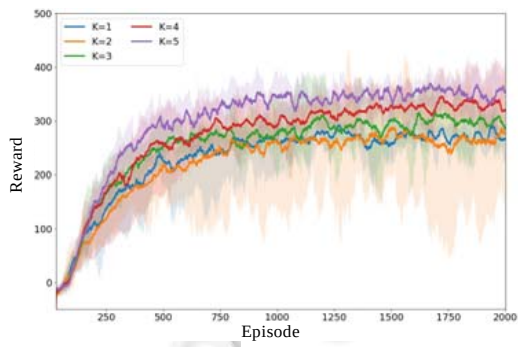
图 8　APODT 与不同算法中在 3 种不同机器人中平均运动步数的对比(续)

3.3.3　消融实验

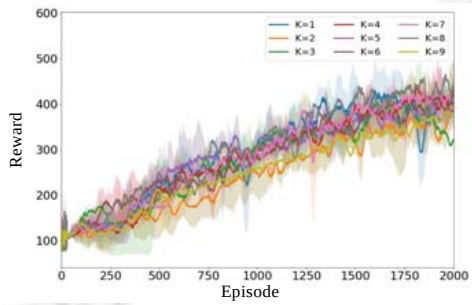
图 9 对比了不同 K 值的 APODT 算法在 3 种机器人环境中的性能. 由图可知. 越大的 K 值并不能保证最好的学习效率. 例如: 在 Walker (具有 6 个关节)中, 在中 K=2 或 K=3 时学习效果最好; 在 Half-Cheetah (具有 6 个关节)中, 具有全连接图(K=5)的 APODT 的性能明显最优; 在 Humanoid (10 个关节)中, K=8 时具有最优的收敛速率. 可见: APODT 的性能随着 K 的变化而略微变化, 信息冗余在不同程度上影响控制器的学习性能. 如何自适应地探索最优的 K 值, 是未来工作研究的方向之一.



(a) Walker



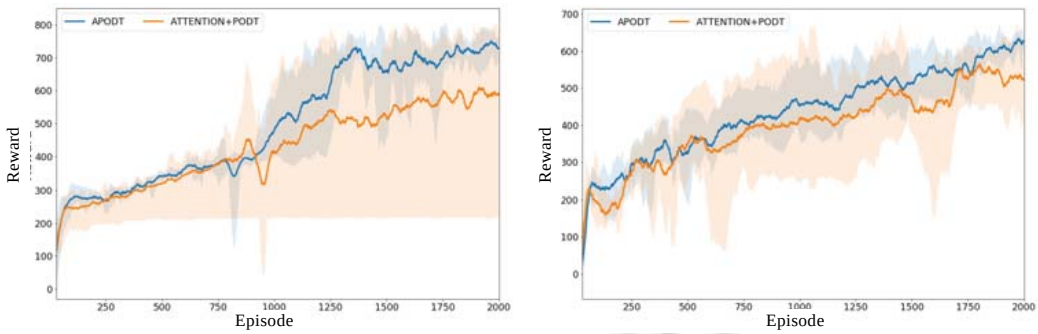
(b) Half-Cheetah



(c) Humanoid

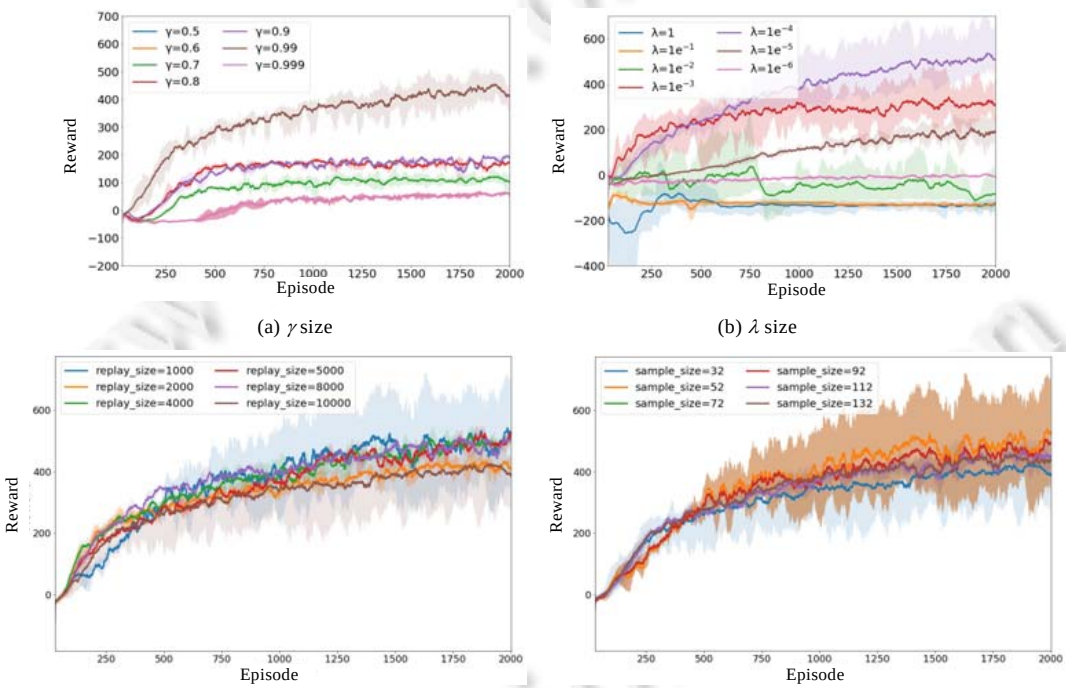
图 9　具有不同 K 值的 APODT 算法的性能

图 10 中, ATTENTION+PODT 代表将两种协同图模型输出邻接矩阵的平均来计算 DoI 值. 由图可知: 直接平均会导致协同图变化过快, 因此, 机器人关节无法快速调整姿势以适应 DoI 的变化. 因此, 一个合理计算 DoI 的模型对于最终算法的稳定性至关重要. 此外, 图 11 展示了 APODT 部分参数的对比图.



(a) Hopper (b) Walker

图 10 APODT 与直接结合算法在两种机器人环境中的平均奖励值的对比



(a) γ size (b) λ size

(c) replay size (d) sample size

图 11 具有大小不同参数的 APODT 算法在 Half-Cheetah 机器人中的性能

3.3.4 可解释性分析

通过分析协同图的变化, SMILE 能够对机器人运动过程中内部关节的协同关系进行直观解释. 图 12 显示了 APODT 在 Half-Cheetah 环境中的协同图示意图, 其中, 黄色箭头表示双向连接, 即两个控制器都考虑对方的信息; 白色箭头表示的单向连接, 箭头指向的控制器必须考虑尾部的控制器的信息进行协调; 红色圆圈表示被其他控制器注意力(权重)最高的关节. 图 12(a)显示了机器人正常行走时的姿态和协同图. 为了保持平稳行走, 所有的主要关节(即大腿前部和后部、膝盖和脚踝)之间都会有状态信息的交互. 当机器人跳跃时, 即图 12(b), 信息传递主要集中在后大腿, 表明它的信息被其他控制器共享, 使得其他关节在准备跳跃时可以做出协同动作. 当机器人着陆时, 即图 12(c), 后大腿和前脚踝是最重要的关节, 对其他关节的运动影响最大. 综上, SMILE 能够利用协同图来捕捉机器人在不同运动姿势下内部关节之间的潜在依赖关系.

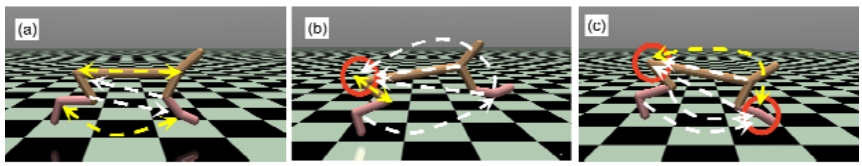


图 12 在行走(a)、跳跃(b)和着陆(c)时，对 Half-Cheetah 内部依赖关系的解释

4 结 论

本文提出了一种基于结构交互驱动的机器人深度强化学习方法 SMILE，用以动态推理机器人内部关节的依赖关系，从而实现高效的策略学习. 该方法首先利将整体机器人的分解成多个控制器，从而减少解决整个任务所需的搜索空间；然后，利用协同图模型定义控制器之间的交互程度，通过融合其他控制器状态信息的增强状态，实现机器人内部关节的信息传递；最后，提出了两种改进的协同图模型更新方法，提升最终算法的稳定性和有效性. 在不同的 DoI 算法中，ATTENTION 具有较高的计算复杂度，因此更适合处理低维的机器人控制问题；而 PODT 可以降低计算复杂度，但同时也因信息缺失而容易陷入局部最优，仅适用于中-低复杂度的机器人；APODT 通过计算复杂度和信息冗余度之间的平衡，既减少了协同图模型的复杂度，又可以有效避免信息冗余，因此在所有机器人环境中均能获得最优的性能表现. 在未来工作中，我们计划进一步完善机器人结构分解模型，并利用真实机器人平台(如 Nao V6 机器人和 Cython E300 机器臂)验证算法的有效性.

References:

[1] Saridis G. Intelligent robotic control. IEEE Trans. on Automatic Control, 1983, 28(5): 547−557.

[2] Yu C, Wang D, Ren J, *et al*. Decentralized multiagent reinforcement learning for efficient robotic control by coordination graphs. In: Proc. of the Pacific Rim Int'l Conf. on Artificial Intelligence. Cham: Springer, 2018. 191−203.

[3] Yu C, Liu J, Nemati S, *et al*. Reinforcement learning in healthcare: A survey. ACM Computing Surveys (CSUR), 2021, 55(1): 1−36.

[4] Ye D, Liu Z, Sun M, *et al*. Mastering complex control in moba games with deep reinforcement learning. Proc. of the AAAI Conf. on Artificial Intelligence, 2020, 34(4): 6672−6679.

[5] Yu C, Wang X, Xu X, *et al*. Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs. IEEE Trans. on Intelligent Transportation Systems, 2019, 21(2): 735−748.

[6] He J, Chen J, He X, *et al*. Deep reinforcement learning with a natural language action space. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics. 2016. 1621−1630.

[7] Chen S, Luo W, Yu C. Reinforcement learning with teacher-student framework in future market. In: Proc. of the Int'l Conf. on Algorithms, High Performance Computing, and Artificial Intelligence (AHPCAI 2021), Vol.12156. SPIE, 2021. 61−68.

[8] Arulkumaran K, Deisenroth MP, Brundage M, *et al*. Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 2017, 34(6): 26−38.

[9] Li Z, Cheng X, Peng XB, *et al*. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In: Proc. of the 2021 IEEE Int'l Conf. on Robotics and Automation (ICRA). IEEE, 2021. 2811−2817.

[10] Li Y, Hao X, She Y, *et al*. Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning. Aerospace Science and Technology, 2021, 109: 106446.

[11] Fu S, Tang Y, Wu Y, *et al*. Energy-Efficient UAV-enabled data collection via wireless charging: A reinforcement learning approach. IEEE Internet of Things Journal, 2021, 8(12): 10209−10219.

[12] Mnih V, Kavukcuoglu K, Silver D, *et al*. Playing atari with deep reinforcement learning. In: Proc. of the NIPS Deep Learning Workshop. 2013.

[13] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. Proc. of the AAAI Conf. on Artificial Intelligence. 2016, 30(1).

[14] Wang Z, Schaul T, Hessel M, *et al*. Dueling network architectures for deep reinforcement learning. In: Proc. of the Int'l Conf. on Machine Learning. 2016. 1995−2003.

[15] Lillicrap TP, Hunt JJ, Pritzel A, *et al*. Continuous control with deep reinforcement learning. In: Proc. of the Int'l Conf. on Learning Representations. 2016.

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

[17] Haarnoja T, Zhou A, Abbeel P, *et al*. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proc. of the Int'l Conf. on Machine Learning (PMLR). 2018. 1861−1870.

[18] Haarnoja T, Pong V, Zhou A, *et al*. Composable deep reinforcement learning for robotic manipulation. In: Proc. of the 2018 IEEE Int'l Conf. on Robotics and Automation (ICRA). IEEE, 2018. 6244−6251.

[19] Vecerik M, Hester T, Scholz J, *et al*. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. arXiv:1707.08817, 2017.

[20] Bai X, Lu C, Bao Q, *et al*. An improved PPO for multiple unmanned aerial vehicles. Journal of Physics: Conf. Series. IOP Publishing, 2021, 1757(1): 012156.

[21] Lin K, Gong L, Li X, *et al*. Exploration-cguidance for robot control. arXiv:2002.12089, 2020.

[22] Gu S, Holly E, Lillicrap T, *et al*. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In: Proc. of the 2017 IEEE Int'l Conf. on Robotics and Automation (ICRA). IEEE, 2017. 3389−3396.

[23] Hanna J, Stone P. Grounded action transformation for robot learning in simulation. In: Proc. of the AAAI Conf. on Artificial Intelligence. 2017. 3834−3840.

[24] Dziomin U, Kabysh A, Golovko VA, *et al*. A multi-agent reinforcement learning approach for the efficient control of mobile robot. In: Proc. of the IEEE Int'l Conf. on Intelligent Data Acquisition & Advanced Computing Systems. IEEE, 2013.

[25] Busoniu L, De Schutter B, Babuska R. Decentralized reinforcement learning control of a robotic manipulator. In: Proc. of the 2006 9th Int'l Conf. on Control, Automation, Robotics and Vision. IEEE, 2006. 1−6.

[26] Wang T, Liao R, Ba J, *et al*. Nervenet: Learning structured policy with graph neural networks. In: Proc. of the Int'l Conf. on Learning Representations. 2018. 1−26.

[27] Jiang J, Lu Z. Learning attentional communication for multi-agent cooperation. Advances in Neural Information Processing Systems, 2018, 31.

[28] Hoshen Y. Vain: Attentional multi-agent predictive modeling. Advances in Neural Information Processing Systems, 2017, 30.

[29] Szita I, Lörincz A. Learning tetris using the noisy cross-entropy method. Neural Computation, 2006, 18(12): 2936−2941.

[30] Williams RJ. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 1992, 8(3-4): 229−256.

[31] Peters J, Schaal S. Natural actor-critic. Neurocomputing, 2008, 71(7−9): 1180−1190.

余超(1985－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为智能体与多智能体系统, 强化学习, 智能机器人.

董银昭(1995－), 男, 博士生, CCF 学生会员, 主要研究领域为强化学习, 智能机器人.

郭宪(1985－), 男, 博士, 副教授, 博士生导师, 主要研究领域为强化学习, 多智能体技术, 博弈论等在机器人领域中的研究和应用.

冯旸赫(1985－), 男, 博士, 副教授, 博士生导师, 主要研究领域为智能指挥控制, 强化学习.

卓汉逵(1982－), 男, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为智能规划, 强化学习, 自然语言处理, 机器人行为控制.

张强(1971－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为机器行为与人机协同, 生物计算.