软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007150]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向参数化动作空间的多智能体中心化策略梯度分解及其应用*

田树聪[1], 谢 愈[2], 张远龙[2], 周正春[1], 高 阳[3]

[1](西南交通大学 信息科学与技术学院, 四川 成都 611756)

[2](国防科技大学 智能科学学院, 湖南 长沙 410073)

[3](计算机软件新技术国家重点实验室 (南京大学), 江苏 南京 210023)

通信作者: 谢愈, E-mail: xieyu@nudt.edu.cn

摘 要: 近年来, 多智能体强化学习方法凭借 AlphaStar、AlphaDogFight、AlphaMosaic 等成功案例展示出卓越的决策能力以及广泛的应用前景. 在真实环境的多智能体决策系统中, 其任务的决策空间往往是同时具有离散型动作变量和连续型动作变量的参数化动作空间. 这类动作空间的复杂性结构使得传统单一针对离散型或连续型的多智能体强化学习算法不在适用, 因此研究能用于参数化动作空间的多智能体强化学习算法具有重要的现实意义. 提出一种面向参数化动作空间的多智能体中心化策略梯度分解算法, 利用中心化策略梯度分解算法保证多智能体的有效协同, 结合参数化深度确定性策略梯度算法中双头策略输出实现对参数化动作空间的有效耦合. 通过在 Hybrid Predator-Prey 场景中不同参数设置下的实验结果表明该算法在经典的多智能体参数化动作空间协作任务上具有良好的性能. 此外, 在多巡航导弹协同突防场景中进行算法效能验证, 实验结果表明该算法在多巡航导弹突防这类具有高动态、行为复杂化的协同任务中有效性和可行性.

关键词: 参数化动作空间; 多智能体强化学习; 中心化策略梯度分解; 多巡航导弹突防

中图法分类号: TP18

中文引用格式: 田树聪, 谢愈, 张远龙, 周正春, 高阳. 面向参数化动作空间的多智能体中心化策略梯度分解及其应用. 软件学报. http://www.jos.org.cn/1000-9825/7150.htm

英文引用格式: Tian SC, Xie Y, Zhang YL, Zhou ZC, Gao Y. Factored Multi-agent Centralised Policy Gradients with Parameterized Action Space and Its Application. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7150.htm

Factored Multi-agent Centralised Policy Gradients with Parameterized Action Space and Its Application

TIAN Shu-Cong[1], XIE Yu[2], ZHANG Yuan-Long[2], ZHOU Zheng-Chun[1], GAO Yang[3]

[1](School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China)

[2](College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China)

[3](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract: In recent years, multi-agent reinforcement learning methods have demonstrated excellent decision-making capabilities and broad application prospects in successful cases such as AlphaStar, AlphaDogFight, and AlphaMosaic. In the multi-agent decision-making system in a real-world environments, the decision-making space of its task is often a parameterized action space with both discrete and continuous action variables. The complex structure of this type of action space makes traditional multi-agent reinforcement learning algorithms no longer applicable. Therefore, researching for parameterized action spaces holds important significance in real-world application. This study proposes a factored multi-agent centralised policy gradients algorithm for parameterized action space in multi-agent settings. By utilizing the factored centralised policy gradient algorithm, effective coordination among multi-agent is ensured. After that, the output of the dual-

* 基金项目: 国家自然科学基金 (62173336, 92271108)
收稿时间: 2023-07-11; 修改时间: 2023-10-09, 2023-11-09; 采用时间: 2024-01-08; jos 在线出版时间: 2024-07-17

headed policy in the parameterized deep deterministic policy gradient algorithm is employed to achieve effective coupling in the parameterized action space. Experimental results under different parameter settings in the hybrid predator-prey scenario show that the algorithm has good performance on classic multi-agent parameterized action space collaboration tasks. Additionally, the algorithm's effectiveness and feasibility is validated in a multi-cruise-missile collaborative penetration tasks with complex and high dynamic properties.

Key words: parameterized action space; multi-agent reinforcement learning; factored centralised policy gradient; multi-cruise-missile collaborative penetration

多智能体强化学习 (multi-agent reinforcement learning, MARL)[1−3]是一类帮助多个智能体的联合策略能够实现全局奖励最大化的群体智能学习范式, 由于 MARL 所面向的群体智能决策更贴近实际应用需求, 使得 MARL 在近些年得到了广泛的关注并实现了快速发展. 目前, MARL 已经应用到了如自动驾驶、交通信号控制、无人机协同、编队集群部署等诸多领域[4−6], 并且在很多商业和军事的应用上表现出了巨大的潜力、前景和应用价值. 现有的性能优越的 MARL 算法通常是基于集中式训练分布式执行 (centralized training with decentralized execution, CTDE) 范式[7]. 在 CTDE 范式中, 所有智能体只能依据自身的观测信息分布式的决策出各自的动作, 同时基于全局状态信息和智能体的联合策略使用中心化评估网络引导所有智能体的策略网络朝着联合奖励最大化的梯度方向更新.

传统的 MARL 算法只能适用于离散型的动作空间如 QMIX[8], 或者连续型的动作空间如 MADDPG[9]. 在许多实际的多智能体决策应用中, 通常需要在同时具有离散动作变量和连续动作变量的参数化策略空间[10]中进行决策. 目前对于参数化动作空间下单智能体强化学习算法已经有了较为广泛的研究, 提出了如 PDDPG、P-DQN、HPPO、HyAR[11−14]等能够适用于参数化动作空间的强化学习方法. 然而, 整体而言面向参数化动作空间下 MARL 算法的相关研究工作还比较少, 这也是使得 MARL 难以推广到许多实际应用领域中的重要原因.

多巡航导弹协同突防打击是多域联合作战中的重要组成部分, 同时也是军事领域的重要发展方向[15,16]. 为了探索让巡航导弹在高动态对抗环境中精准打击目标的同时有效规避拦截弹的控制方法, 越来越多的专家学者开始关注利用强化学习的方法对巡航导弹的多元化突防参数进行智能控制[17,18], 希望通过设计出有效的强化学习智能决策方法, 以满足在强对抗的多巡航导弹协同突防博弈环境中多元化突防参数的高效决策需求.

针对上述提出的问题, 本文首先设计了一种能够应用于参数化动作空间的多智能体中心化策略梯度分解算法, 并且在开源强化学习环境 Hybrid Predator-Prey 场景[9]的多组参数设定下验证了该算法的效能. 同时将该算法应用到了具有参数化动作空间的多巡航导弹协同突防的场景中, 通过在仿真系统中的交互迭代训练, 最终收敛的算法模型能够有效协同控制多枚巡航导弹在有效规避威胁区/拦截弹的同时能够精准对目标制导. 通过在两类不同参数化动作空间任务场景下的实验验证了本文所提出新型多智能体算法对于解决参数化动作空间决策问题的可行性和有效性, 同时也为多巡航导弹智能协同决策问题提供了一种新的解决思路.

1 相关工作

1.1 分布式-部分可观测马尔可夫决策过程

马尔可夫决策过程 (Markov decision process, MDP) 是强化学习方法的一种本质表达形式, 具象化地表示出了强化学习方法中智能体与外部环境的迭代、交互、试错过程. 传统的 MDP 面向的是单智能体应用场景, 面对多智能体协作的应用场景时, 需要将 MDP 拓展为分布式-部分可观测马尔可夫决策过程 (decentralised partially observable Markov decision process, Dec-POMDP)[19].

Dec-POMDP 的定义由一个八元组 $G=\langle \mathcal{N},\mathcal{S},U,P,r,\Omega,O,\gamma\rangle$ 构成, 其中 $\mathcal{N}\equiv\{1,\ldots,n\}$ 表示所有智能体的集合, $\mathcal{S}$ 表示全局状态空间, U 表示智能体的动作空间, Ω 表示智能体的观测空间, $\gamma\in[0,1)$ 表示折扣因子. 在每个时间步长, 对于任意智能体 $i\in\mathcal{N}$ 通过观测信息函数 $O(s,i)$ 获得自身的观测信息 $o_i\in\Omega$, 其中 $s\in\mathcal{S}$. 每个智能体决策动作 $a_i\in U$, 所有智能体的动作共同形成联合策略 $\mathbf{a}=\langle a_1,\ldots,a_n\rangle\in\mathbf{U}\equiv U^n$. 外部环境响应联合策略后, 通过状态转移 $P(s'\mid s,\mathbf{a}):\mathcal{S}\times U\times\mathcal{S}\to[0,1]$ 将当前状态 s 更新为新的状态 s', 并获得全局联合奖励 $r(s,\mathbf{a})$.

在智能体与环境的不断交互训练中, 智能体会不断优化学习自己的随机策略 $\pi_i(a_i,\tau_i)$ 或者确定性策略 $\mu_i(\tau_i)$,

其中 τ_i 表示动作—观测的历史轨迹信息 $\tau_i \in T \equiv (\Omega \times U)$. 若智能体采取的是随机策略, 则通过所有智能体的联合随机策略 π 可以诱导出相应的联合 Q 函数 $Q^{\pi}(s_t, \mathbf{a}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{a}_{t+1:\infty}}[R_t \mid s_t, \mathbf{a}_t]$ 其中 $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ 表示累计折扣奖励. 同理, 若智能体采取的是确定性策略, 同样可以利用所有智能体的联合确定性策略 μ 诱导出相应的联合 Q 函数 $Q^{\mu}(s_t, \mathbf{a}_t)$.

1.2 参数化动作马尔可夫决策过程

在很多真实决策任务场景中, 其决策任务的动作空间往往同时包含了离散型动作变量和连续型动作变量. 这类动作空间对应的马尔可夫决策过程, 被称为参数化马尔可夫决策过程 (parameterized action Markov decision process, PAMDP)[10]. PAMDP 由一个五元组 $\langle \mathcal{S}, \mathcal{H}, P, r, \gamma \rangle$ 定义, 与标准 MDP 的差异在于 PAMDP 中的动作空间 $\mathcal{H}$ 表示参数化的动作空间. 参数化动作空间的定义如下所示:

$$\mathcal{H} = \{(k, x_k) \mid x_k \in \mathcal{X}_k \text{ for all } k \in \mathcal{K}\} \tag{1}$$

其中, $\mathcal{K} = 1, \ldots, K$ 表示一组离散的动作集合, 对于任意的 $k \in \mathcal{K}$, x_k 表示一组连续的动作参数, 动作对 (k, x_k) 表示参数化动作空间中的一组动作参数. 根据参数化动作空间的定义, 基于参数化动作的 Q 函数可被表示为 $Q^{\pi}(s, k, x_k)$.

1.3 QMIX

QMIX[8]是一类典型的具有值分解网络结构的多智能体强化学习算法, 可应用于离散动作空间下的完全合作式多智能体协同任务. QMIX 算法是一类基于集中式训练分布式执行范式的算法, 其中分布式执行表现在每个智能体通过个体 Q 网络进行策略选择; 集中式训练则是将所有智能体的个体 Q 值以非线性复合的方式得到联合 Q 值, 通过联合 Q 值对联合策略进行中心化评估以实现对个体 Q 网络的参数优化更新, 其中复合个体 Q 值的权重由混合网络生成.

在 QMIX 算法中, 为了保证联合 Q 值下的最优联合策略与个体 Q 值下的最优个体策略的一致性, 需要保证 $Q_{\text{tot}}(s, \mathbf{u})$ 与 $Q_i(\tau_i, u_i)$ 之间满足如下的关系式:

$$\arg\max_{\mathbf{u}} Q_{\text{tot}}(s, \mathbf{u}) = \begin{pmatrix} \arg\max_{u_1} Q_1(\tau_1, u_1) \\ \vdots \\ \arg\max_{u_n} Q_n(\tau_n, u_n) \end{pmatrix} \tag{2}$$

公式 (2) 所示的约束关系被称为 IGM 条件, 为了保证联合 Q 值 $Q_{\text{tot}}(s, \mathbf{u})$ 与个体 Q 值 $Q_i(\tau_i, u_i)$ 之间满足 IGM 条件, 采用的方法是对 $Q_{\text{tot}}(s, \mathbf{u})$ 与 $Q_i(\tau_i, u_i)$ 之间进行单调性约束, 即限制 $Q_{\text{tot}}(s, \mathbf{u})$ 与 $Q_i(\tau_i, u_i)$ 之间的偏导关系非负, 如公式 (3) 所示.

$$\frac{\partial Q_{\text{tot}}(s, \mathbf{u})}{\partial Q_i(\tau_i, u_i)} \geqslant 0, \quad i = 1, \ldots, n \tag{3}$$

1.4 参数化动作空间下的多智能体算法

1.4.1 Deep MAPQN 和 Deep MAHHQN

针对参数化动作空间在多智能体协作任务中的算法设计问题, Fu 等人[20]在 2019 年提出的 Deep MAPQN 和 Deep MAHHQN 两种多智能体强化学习算法, 两种算法都是基于 QMIX 的结构进行设计的.

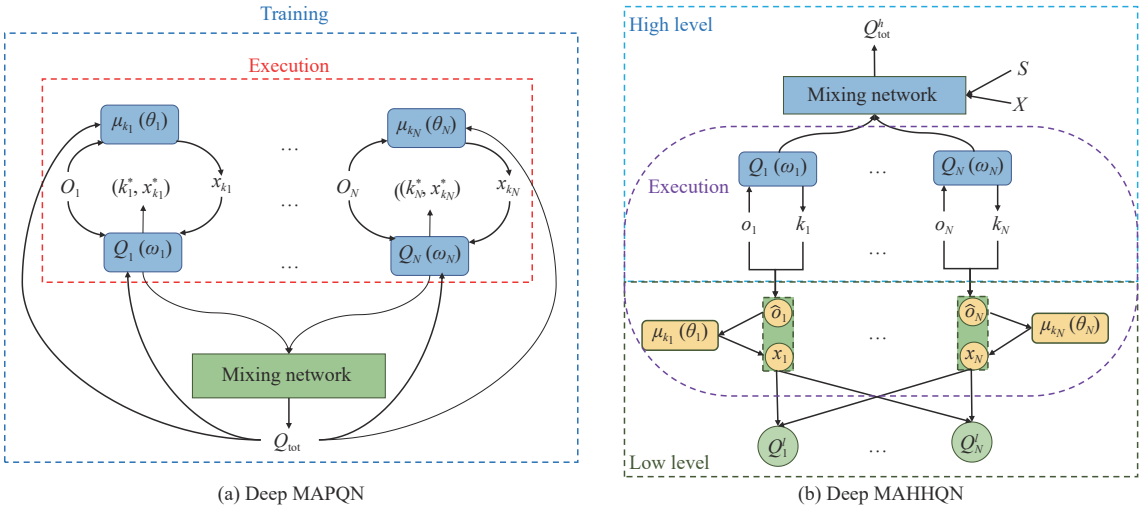
Deep MAPQN 是在 QMIX 的框架基础上, 利用 P-DQN[12]设计了分布式策略, 整个算法结构如图 1(a) 所示. 在 Deep MAPQN 中, 智能体 i 首先使用确定性策略网络 $\mu_{k_i}(o_i \mid \theta_i)$ 决策连续动作参数 x_{k_i}, 其次基于 x_{k_i} 再使用个体 Q 网络选择离散动作值 k_i. 整个参数化动作的决策过程如下所示:

$$(k_i, x_{k_i}) = \arg\max_{(k_i, x_{k_i})} Q_i(o_i, \mu_{k_i}(o_i \mid \theta_i); \omega_i) \tag{4}$$

Deep MAPQN 通过公式 (4) 所示的方式得到 $Q_i(o_i, x_i \mid \omega_i)$, 然后采用 QMIX 中的非线性复合机制得到联合 Q 值 $Q_{\text{tot}}(\mathbf{s}, \vec{k}, \vec{x}_k)$, 同时利用 QMIX 的更新机制引导所有网络参数进行更新.

Deep MAHHQN 是在 QMIX 的框架基础上, 分别设计了由上层和下层两层结构组成的分布式策略, 其中上层

网络是 QMIX 结构用于选择离散动作, 下层网络是 MADDPG[9]结构用于决策连续动作参数. 整个算法结构如图 1(b)所示. 与 Deep MAPQN 的决策顺序不同, 在 Deep MAHHQN 中智能体 i 首先使用上层网络进行离散动作的选择, 即 $k_i = \arg\max Q_i(o_i, k_i; \omega_i)$; 其次使用下层网络基于 k_i 再进行连续动作参数的决策, 即 $x_i = \mu_i(\varphi(o_i, k_i))$.



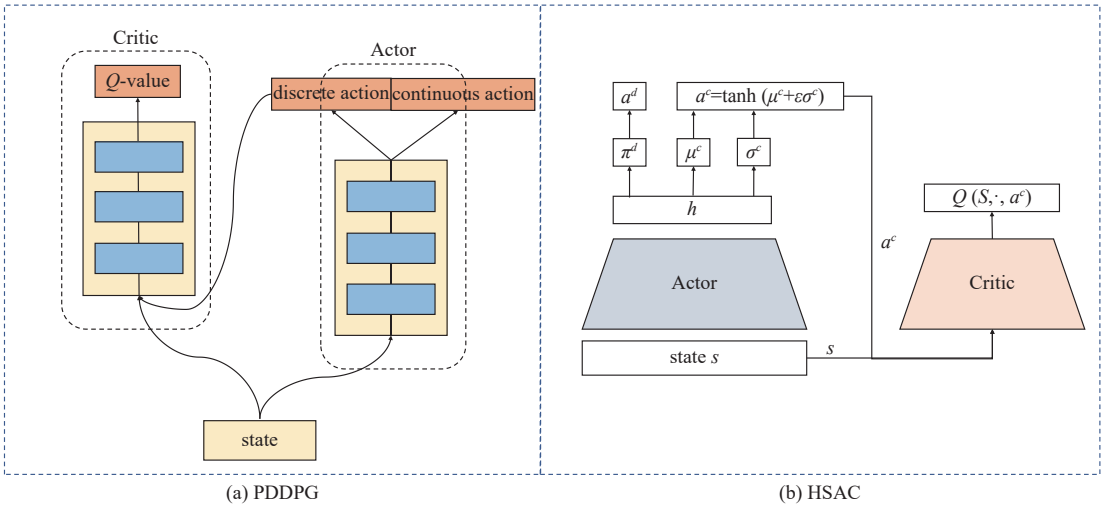
(a) Deep MAPQN (b) Deep MAHHQN

图 1 Deep MAPQN 和 Deep MAHHQN 的算法结构图

在 Deep MAHHQN 中上层网络采用了与 QMIX 相似的更新方式, 有所不同的地方在于 Deep MAHHQN 复合联合 Q 值 Q_{tot}^h 时需要将全局状态 $\mathbf{s}$ 以及所有联合智能体的连续动作参数 $\mathcal{X}$ 共同作为混合网络的输入. Deep MAHHQN 的下层网络则采用了与 MADDPG 相同的更新机制.

1.4.2 MAHDDPG 和 MAHSAC

针对参数化动作空间在多智能体协作任务中的算法设计问题, Hua 等人[21]在 2022 年提出了 MAHDDPG 和 MAHSAC 两种多智能体强化学习算法, 这两种算法都是基于 MADDPG[9]的中心化评估方式设计的. 其中 MAHDDPG 是将原本基于 DDPG[22]设计的分布式策略网络结构替换为了 PDDPG[11]的算法结构, 以此保证各智能体分布式策略网络在参数化动作空间下的耦合性. MAHSAC 是基于 HSAC[23]算法结构设计分布式策略网络, 同时利用 MADDPG 的中心化评估方式设计的多智能体算法. 其中 PDDPG 与 HSAC 的算法结构分别如图 2(a) 和图 2(b) 所示.



(a) PDDPG (b) HSAC

图 2 PDDPG 和 HSAC 的算法结构图

2 参数化的多智能体中心化策略梯度分解算法

在许多实际决策应用中, 其决策空间通常是同时包含了离散型动作变量和连续型动作变量的参数化动作空间, 目前已经有了许多关于参数化动作空间下的单智能体强化学习方法研究成果, 但是对于参数化动作空间下的多智能体强化学习方法的相关研究工作还比较少. 设计面向参数化动作空间下的多智能体算法时, 需要考虑两个问题: 1) 分布式策略网络如何与参数化动作空间进行耦合; 2) 中心化评估网络如何对引导分布式策略网络的协同更新.

针对具有参数化动作空间的多智能体协同决策算法设计, 本节提出了一种全新的强化学习方法——参数化的多智能体中心化策略梯度分解算法 (parameterized factored multi-agent centralised policy gradients, P-FACMAC). 该方法以基于值分解网络的多智能体中心化策略梯度分解算法 (factored multi-agent centralised policy gradients, FACMAC)[24]作为中心化评估架构, 通过值分解网络复合生成联合 Q 值的方法避免“信用分配”[25]问题的出现, 并引导对多智能体策略的协同更新. 在此基础上, 借鉴参数化深度确定性策略梯度算法 (parameterized deep deterministic policy gradient, PDDPG)[11]中处理参数化动作空间的策略网络设计, P-FACMAC 设计了与 PDDPG 相似的策略网络结构. 通过上述的设计, 使得 P-FACMAC 能够有效对具有参数化动作空间的多智能体协同任务进行决策和训练.

2.1 P-FACMAC 算法设计

为了方便对 P-FACMAC 算法进行直观地理解, 本节首先对 P-FACMAC 设计中包含的两类基础算法 PDDPG 和 FACMAC 进行介绍; 在此基础上, 详细介绍 P-FACMAC 的算法框架以及该算法训练更新机制.

2.1.1 PDDPG 和 FACMAC

为了解决高维连续动作空间中智能决策问题, Lillicrap 等人提出了深度确定性策略梯度算法 (deep deterministic policy gradient, DDPG)[22]. DDPG 算法最初是用于高维连续动作空间中, 可以通过对 DDPG 算法中的策略网络输出使用 Gumbel-Softmax[26]采样方式, 使其适用于离散型动作空间的应用场景中. 由于 DDPG 算法通过不同的采样机制能够分别适用于离散型动作空间以及连续型动作空间, 基于这一特性 Hausknecht 等人[11]设计出了策略网络具有双头输出结构的 PDDPG 算法. PDDPG 的策略网络输出中一个输出离散型动作变量, 另一个输出连续型动作变量, 通过这样的方式保证了 PDDPG 与参数化动作空间的耦合性.

如图 2(a) 中所示 PDDPG 的主体结构与 DDPG 基本相同, PDDPG 采用了与 DDPG 相同的训练更新机制. 具体来说, PDDPG 中的价值评估网络 (Critic 网络) 与策略网络 (Actor 网络) 的训练更新方式为:

$$\begin{cases} L_Q(\theta^Q)=\mathbb{E}_{(s_t,a_t,r_t,s_{t+1})\sim\mathcal{D}}\left[(Q(s_t,a_t)-(r_t+\gamma Q'(s_{t+1},\mu'(s_{t+1}))))^2\right] \\ \nabla_{\theta^\mu}\mu=\mathbb{E}_{s_t\sim\mathcal{D}}\left[\nabla_a Q(s_t,a\mid\theta^Q)\nabla_{\theta^\mu}\mu(s_t)|_{a=\mu(s_t)}\right] \end{cases} \tag{5}$$

QMIX 及其部分变体算法[27,28]在具有挑战性的多智能体星际挑战任务 (starcraft multi-agent challenge, SMAC)[29]场景中展现出卓越的效果, 但是 QMIX 基于值函数的算法本质使其只能应用于离散型动作空间的多智能体协同场景中. 为了推广 QMIX 算法的应用领域, Peng 等人将 QMIX 和 DDPG 进行结合设计了能够适用于多智能体连续控制任务的 FACMAC 算法[24], FACMAC 的算法结构如图 3 所示.

图 3(a) 表示 FACMAC 算法中分布式策略网络, 图 3(b) 表示 FACMAC 算法中基于值分解的中心化评估网络. 在 FACMAC 算法中每个智能体都有对应的 Actor 网络和 Critic 网络, Actor 网络根据智能体当前的观测信息决策出相应的行为动作, Critic 网络则生成用于评估该智能体在当前观测信息下行为动作价值的个体 Q 值. 通过这样的方式, 使得 FACMAC 能够根据应用场景的需求, 有针对性的设计应用于离散型动作空间或者连续型动作空间的 Actor 网络. 基于 FACMAC 值分解网络的算法结构, 将各智能体 Critic 网络生成个体 Q 值使用混合网络进行非线性复合, 得到用于评估联合动作效能的联合 Q 值. 联合 Q 值的非线性复合方式如下所示:

$$Q_{\text{tot}}=g(s,Q_1,\ldots,Q_N;\omega) \tag{6}$$

在 FACMAC 算法中 ω, $\{\theta_a\}_{a=1}^n$, $\{\phi_a\}_{a=1}^n$ 分别表示混合网络 g, 智能体的个体评估网络 $\{Q_a\}_{a=1}^n$, 智能体的个体策

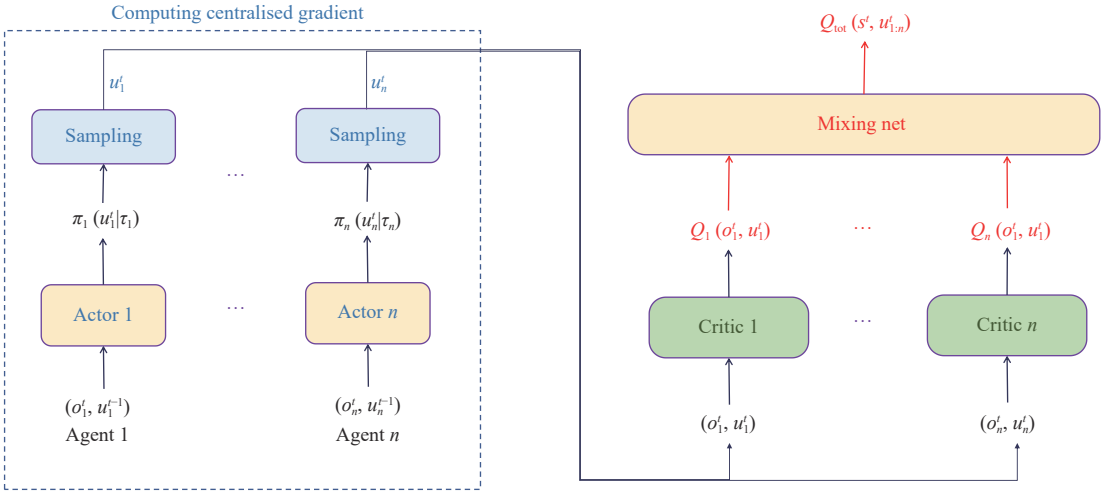
略网络 $\{\mu_a\}_{a=1}^n$ 的网络参数. 首先参数 ω, $\{\theta_a\}_{a=1}^n$ 更新方式如下所示:

$$\begin{cases}\mathcal{L}(\omega,\theta_1,\ldots,\theta_n)=\mathbb{E}_{(s_t,\tau_t,\mathbf{u}_t,r_t,s_{t+1},\tau_{t+1})\sim\mathcal{D}}\left[(y_{\text{tot}}-Q_{\text{tot}}(s,\mathbf{u};\omega,\theta_1,\ldots,\theta_n))^2\right]\\ y_{\text{tot}}=r+\gamma Q_{\text{tot}}(s',\mathbf{u}';\omega^-,\theta_1^-,\ldots,\theta_n^-)\end{cases} \tag{7}$$

在参数 ω, $\{\theta_a\}_{a=1}^n$ 更新的基础上, 以中心化策略梯度估计的方式更新所有智能体的策略网络参数 $\{\phi_a\}_{a=1}^n$, 具体的更新方式如下所示:

$$\nabla_\Phi J(\boldsymbol{\mu})=\mathbb{E}_{(s_t,\mathbf{u}_t,r_t,s_{t+1})\sim\mathcal{D}}\left[\nabla_\Phi\boldsymbol{\mu}\nabla_\mu Q_{\text{tot}}(s,\mu_1(\tau_1),\ldots,\mu_n(\tau_n))\right] \tag{8}$$

其中, $\Phi=\{\phi_1,\phi_2,\ldots,\phi_n\}$ 表示所有智能体当前策略网络参数的集合. 根据实际使用场景可以设置所有智能体共享一套策略网络参数以及共享一套个体评估网络参数, 即 $\phi_1=\phi_2=\ldots=\phi_n$, $\theta_1=\theta_2=\ldots=\theta_n$.



(a) FACMAC 算法中分布式策略网络　　(b) FACMAC 算法中基于值分解的中心化评估网络

图 3　FACMAC 的算法框架图

2.1.2　P-FACMAC 算法

在 FACMAC 算法中, 通过对策略网络的输出使用不同的采样机制使得该策略网络能够分别适用于不同类型的动作空间. 对于同时具有离散动作变量和连续动作变量的参数化动作空间, FACMAC 则无法同时输出离散型和连续型的动作变量. 为了能够解决具有参数化动作空间的多智能体协作任务, 本节在 FACMAC 算法的基础上进行了改进, 将策略网络设计为 PDDPG 中双头输出的形式, 设计了能够同时输出离散型和连续型的动作变量的多智能体强化学习算法 P-FACMAC, P-FACMAC 的算法结构如图 4 所示.

图 4(a) 表示 P-FACMAC 算法中分布式策略网络, 图 4(b) 表示 P-FACMAC 算法中基于值分解的中心化评估网络. P-FACMAC 的主体算法框架与 FACMAC 相同, 每个智能体都有对应的 Actor 网络和 Critic 网络, 但是 P-FACMAC 中各智能体的 Actor 网络为双头结构分别输出离散型动作变量 u_a^T 以及连续型动作变量 u_a^P, Critic 网络接收当前的观测信息以及参数化的动作变量生成个体 Q 值. 所有智能体的个体 Q 值通过混合网络进行非线性复合得到用于评估联合动作效能的全局联合 Q 值, P-FACMAC 算法中混合网络的网络结构如图 5 所示.

在图 5 中 W_1 和 b_1 分别表示第 1 次复合的权重和偏置, W_2 和 b_2 分别表示第 2 次复合的权重和偏置, $|\cdot|$ 表示取元素的绝对值的操作符, 通过操作符 $|\cdot|$ 对 W_1 和 W_2 中所有元素取绝对值以此保证复合得到联合 Q 值能够满足公式 (2) 中的 IGM 条件.

所有智能体的个体 Q 值通过混合网络生成复合权重并基于公式 (6) 所示的方式进行非线性复合得到联合 Q 值 $Q_{\text{tot}}(s_t,u_{1:n}^P,u_{1:n}^T)$. 在 P-FACMAC 算法中 ω, $\{\theta_a\}_{a=1}^n$, $\{\phi_a\}_{a=1}^n$ 分别表示混合网络 g, 智能体的个体评估网络 $\{Q_a\}_{a=1}^n$, 智能体的个体策略网络 $\{\mu_a\}_{a=1}^n$ 的网络参数. P-FACMAC 采用了与 FACMAC 相同的算法框架, 因此 P-FACMAC 中的 ω, $\{\theta_a\}_{a=1}^n$ 参数采用了与公式 (7) 相同的方式进行参数更新, 参数 $\{\phi_a\}_{a=1}^n$ 采用了与公式 (8) 相同的中心化策略梯

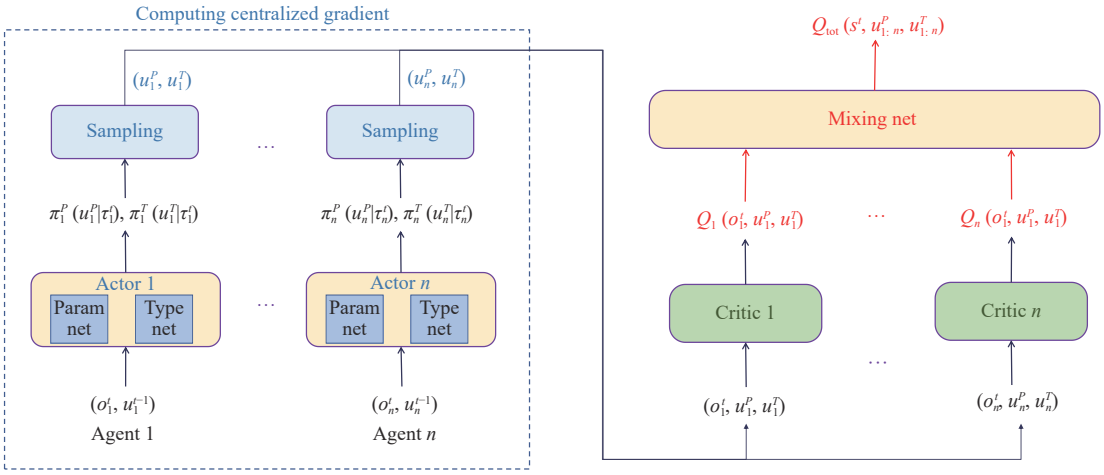
度的更新机制进行更新.

参数 ω, $\{\theta_a\}_{a=1}^n$ 更新方式为:

$$\begin{cases}\mathcal{L}(\omega,\theta_1,\ldots,\theta_n)=\mathbb{E}_{(s_t,\tau_t,\mathbf{u}_t^P,\mathbf{u}_t^T,r_t,s_{t+1},\tau_{t+1})\sim\mathcal{D}}\left[(y_{\text{tot}}-Q_{\text{tot}}(s,\mathbf{u}^P,\mathbf{u}^T;\omega,\theta_1,\ldots,\theta_n))^2\right]\\ y_{\text{tot}}=r+\gamma Q_{\text{tot}}(s',(\mathbf{u}^P)',(\mathbf{u}^T)';\omega^-,\theta_1^-,\ldots,\theta_n^-)\end{cases}\tag{9}$$

参数 $\Phi=\{\phi_a\}_{a=1}^n$ 的更新方式为:

$$\nabla_\Phi J(\boldsymbol{\mu})=\mathbb{E}_{(s_t,\tau_t,\mathbf{u}_t^P,\mathbf{u}_t^T,r_t,s_{t+1},\tau_{t+1})\sim\mathcal{D}}\left[\nabla_\Phi\boldsymbol{\mu}\nabla_{\boldsymbol{\mu}}Q_{\text{tot}}(s,\mu_1(\tau_1),\ldots,\mu_n(\tau_n))\right]\tag{10}$$



(a) P-FACMAC 算法中分布式策略网络　　(b) P-FACMAC 算法中基于值分解的中心化评估网络

图 4　P-FACMAC 的算法结构图

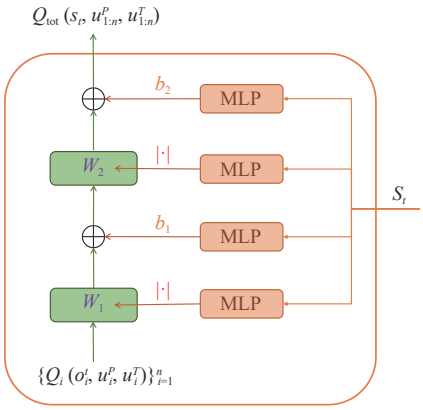


图 5　P-FACMAC 中混合网络的网络结构

同样, P-FACMAC 算法在使用过程中可以根据实际场景需求, 设置所有智能体共享一套策略网络参数以及共享一套个体评估网络参数, 即 $\phi_1=\phi_2=\ldots=\phi_n$, $\theta_1=\theta_2=\ldots=\theta_n$.

基于上述的更新方式, P-FACMAC 算法的伪代码如算法 1 所示.

算法 1. P-FACMAC.

1. 初始化所有智能体的个体评估网络 $\{Q_i(\tau^i,(u^P,u^T)_i;\theta_i)\}_{i=1}^n$, 个体策略网络 $\{\mu_i(\tau_i;\phi_i)\}_{i=1}^n$, 混合网络 $g(\cdot;\omega)$
2. 初始化目标网络参数 $\{\theta_i^-\leftarrow\theta_i\}_{i=1}^n$, $\{\phi_i^-\leftarrow\phi_i\}_{i=1}^n$, $\omega^-\leftarrow\omega$

3. 初始化经验池 $\mathcal{D}$

4. **for** $episode = 1$ to M **do**

5. 推演环境初始化, 并获取初始全局状态 s_0, 所有智能体的局部观测信息 $\{\tau_0^i\}_{i=0}^n$

6. **for** $t = 1$ to max-episode-length **do**

7. 对于第 i 个智能体, 决策出参数化动作 $(u_t^P, u_t^T)_i = \mu_i(\tau_t^i)$

8. 推演环境执行所有智能体的联合动作 $(\mathbf{u}_t^P, \mathbf{u}_t^T) = \{(u_t^P, u_t^T)_i\}_{i=1}^n$, 并获得全局联合奖励态 r_t, 更新全局状态 s_{t+1} 以及所有智能体的观测信息 $\{\tau_{t+1}^i\}_{i=1}^n$

9. 将 t 时刻的样本经验 $[s_t, \{\tau_t^i\}_{i=1}^n, (\mathbf{u}_t^P, \mathbf{u}_t^T), r_t, s_{t+1}, \{\tau_{t+1}^i\}_{i=1}^n]$ 保存至经验池 $\mathcal{D}$

10. $s_t \leftarrow s_{t+1}$, $\{\tau_t^i\}_{i=1}^n \leftarrow \{\tau_{t+1}^i\}_{i=1}^n$

11. **if** Training **then**

12. 从经验池 $\mathcal{D}$ 中随机批采样 $[s_h, \{\tau_h^i\}_{i=1}^n, (\mathbf{u}_h^P, \mathbf{u}_h^T), r_h, s_{h+1}, \{\tau_{h+1}^i\}_{i=1}^n]_{h=1}^N$

13. $y_{\text{tot}} = r + \gamma Q_{\text{tot}}(s', (\mathbf{u}^P)', (\mathbf{u}^T)'; \omega^-, \theta_1^-, \ldots, \theta_n^-)$

14. 更新混合网络参数 ω, 个体评估网络参数 $\{\theta_i\}_{i=1}^n$:

15. $\mathcal{L}(\omega, \theta_1, \ldots, \theta_n) = \mathbb{E}_{(s_t, \tau_t, \mathbf{u}_t^P, \mathbf{u}_t^T, r_t, s_{t+1}, \tau_{t+1}) \sim \mathcal{D}} \left[\left(y_{\text{tot}} - Q_{\text{tot}}(s, \mathbf{u}^P, \mathbf{u}^T; \omega, \theta_1, \ldots, \theta_n) \right)^2 \right]$

16. 更新个体策略网络参数和 $\{\phi_i\}_{i=1}^n$:

17. $\nabla_\Phi J(\mu) = \mathbb{E}_{(s_t, \tau_t, \mathbf{u}_t^P, \mathbf{u}_t^T, r_t, s_{t+1}, \tau_{t+1}) \sim \mathcal{D}} [\nabla_\Phi \mu \nabla_\mu Q_{tot}(s, \mu_1(\tau_1), \ldots, \mu_n(\tau_n))]$

18. 更新目标网络参数:

19. $\omega^- \leftarrow \tau\omega + (1-\tau)\omega^-$

20. $\{\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-\}_{i=1}^n$

21. $\{\phi_i^- \leftarrow \tau\phi_i + (1-\tau)\phi_i^-\}_{i=1}^n$

22. **end if**

23. **end for**

24. **end for**

2.2 P-FACMAC 算法实验

第 2.1 节对 P-FACMAC 的算法框架设计以及网络参数的更新方式进行了详细的描述, 本节将对 P-FACMAC 算法性能进行对比分析. 首先选择了具有参数化动作空间的 Hybrid Predator-Prey 开源场景作为算法的测试环境, 同时选择了 Deep MAPQN、Deep MAHHQN、MAHDDPG、MAHSAC 这 4 种能够适用于参数化动作空间的多智能体强化学习算法与 P-FACMAC 进行算法性能的对比分析.

2.2.1 实验环境的简介

本节的实验是在 PettingZoo 多智能体环境库中进行的, 选择了其中 MPE 环境下 Predator-Prey 场景作为 P-FACMAC 算法的验证实验环境 (Predator-Prey 场景对应着 MPE 库中的 simple_tag 包). 图 6 是一个具有 3 个捕食者、1 个猎物和 2 个障碍物在二维空间中的 Predator-Prey 场景示例, 这类场景中捕食者和猎物智能体具有合作与竞争关系, Predator-Prey 场景最初在文献 [9] 中被用于验证 MADDPG 的算法性能. 在图 6 中红色的圆圈表示捕食者, 捕食者的特点是速度比较慢; 绿色的圆圈表示猎物, 与捕食者相比猎物具有更快的速度; 灰色的大圆圈表示障碍物, 用于阻挡捕食者在部分方向上的移动, 在每次推演开始时都会在整个二维空间中随机生成相应数量的障碍物. 在 Predator-Prey 场景中, 捕食者因为速度比较慢, 因此需要合作完成对猎物的捕捉, 同时还要注意在移动过程中对障碍物进行规避.

PettingZoo 中内置的 Predator-Prey 场景需要同时对捕食者和猎物进行控制, 因此这是一个竞争—合作式的任务场景. 为了得到一个完全合作式的任务场景, 本节参考了文献 [24] 中方法, 为猎物设计了一套基于硬编码的启发式行动策略, 该策略的核心思想是在任何时刻都让猎物朝着远离与距离自己最近的捕食者的方向移动. 在原始

的 Predator-Prey 场景中设置的动作空间只能支持离散型动作空间 (行动的方位 $[x^+,x^-,y^+,y^-]$) 或者连续型动作空间 (不同行动方位上的连续速度增量, 速度增量的取值区间 $\in[0,1]$). 为了能够支持算法在参数化动作空间下的性能验证, 将原始的 Predator-Prey 场景中离散动作空间和连续动作空间进行组合. 在新的动作空间设置中, 先决策行动的离散方位 $\{(x^+,y^+),(x^+,y^-),(x^-,y^+),(x^-,y^-)\}$, 再根据选择出的行动方位决策对应方位上的连续型速度增量. 通过上述两项设计可以将原本的 Predator-Prey 场景变成了具有参数化动作空间的 Hybrid Predator-Prey 场景.

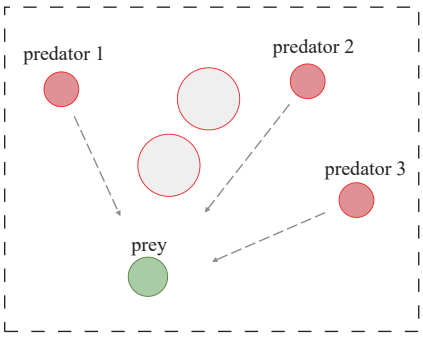


图 6 Predator-Prey 实验环境

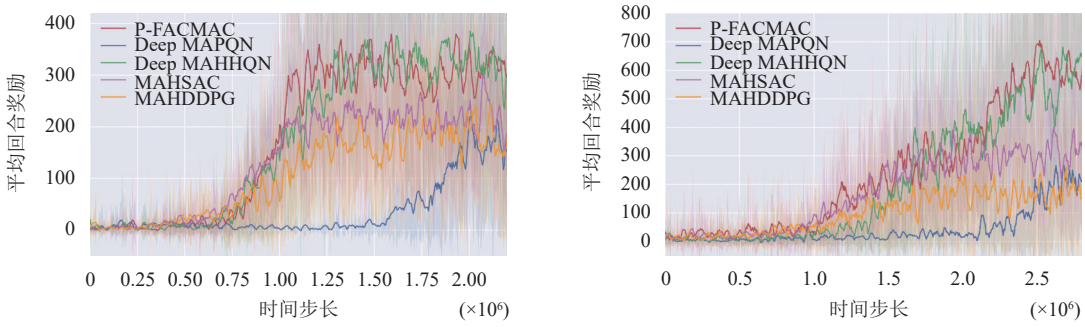
2.2.2 实验效果的对比分析

本节在 Hybrid Predator-Prey 场景中分别设置 3 组不同 predators-prey-obstacles 参数作为验证 P-FACMAC 算法性能的实验场景, 其中场景 1 的参数为 3 predators, 1 prey, 2 obstacles; 场景 2 的参数为 5 predators, 1 prey, 3 obstacles; 场景 3 的参数为 6 predators, 2 prey, 4 obstacles. 其中 P-FACMAC 算法在 Hybrid Predator-Prey 场景的训练过程中超参数取值如表 1 所示.

表 1 Hybrid Predator-Prey 场景中 P-FACMAC 的训练超参数

超参数项	取值
参数 ω 和 $\{\theta_a\}_{a=1}^n$ 的优化器	Adam优化器
参数 ω 和 $\{\theta_a\}_{a=1}^n$ 的学习率	5E–5
参数 $\{\phi_a\}_{a=1}^n$ 的优化器	Adam优化器
参数 $\{\phi_a\}_{a=1}^n$ 的学习率	5E–5
混合网络 g 是否使用目标网络	是
智能体个体评估网络 $\{Q_a\}_{a=1}^n$ 是否使用目标网络	是
所有智能体个体评估网络是否共享网络参数	是
智能体个体策略网络 $\{\mu_a\}_{a=1}^n$ 是否使用目标网络	是
所有智能体个体策略网络是否共享网络参数	是
经验池大小	100 000
软更新系数 τ	0.05
折扣因子 γ	0.85
批采样大小batch_size	256
算法训练周期	每收集25帧样本训练1次

P-FACMAC 及其对比算法 Deep MAPQN、Deep MAHHQN、MAHSAC、MAHDDPG 在 3 组不同参数设置的 Hybrid Predator-Prey 场景中的训练效果如图 7 所示. 算法在训练过程中, 每经过 5 000 条样本的训练更新后对当前模型进行 10 次推演测试, 将 10 次测试中的累积奖励值保存并记录下来, 统计 10 次测试中的平均值和标准差绘制成如图所示的平均回合奖励的变化趋势.



(a) 场景 1: 3 predators, 1 prey and 2 obstacles

(b) 场景 2: 5 predators, 1 prey and 3 obstacles

(c) 场景 3: 6 predators, 2 prey and 4 obstacles

图 7 Hybrid Predator-Prey 场景中平均回合奖励的变化趋势

根据图 7(a)–图 7(c) 所示的平均回合奖励变化趋势可知, 在场景 1 中 P-FACMAC 与 Deep MAHHQN 算法经过大约 1 200 000 个时间步长的采样训练后, 算法模型基本收敛, 最终收敛模型的平均回合奖励大约为 350 左右. 在场景 2 中 P-FACMAC 与 Deep MAHHQN 算法经过大约 2 500 000 个时间步长的采样训练后, 算法模型基本收敛, 最终收敛的平均回合奖励大约为 600 左右. 在场景 3 中 P-FACMAC 算法经过大约 1 800 000 个时间步长的采样训练后, 算法模型基本收敛, 最终收敛的平均回合奖励大约为 1 100 左右. 通过 3 组场景中的实验可以发现 P-FACMAC 与 Deep MAHHQN 这 2 种算法随着迭代次数的增加训练的模型能够逐渐收敛到较高收益水平, 而 Deep MAPQN、MAHSAC、MAHDDPG 这 3 种算法则陷入了局部收敛的状态.

根据图 7(a)–图 7(c) 所示的算法实验效果, 并结合场景特点、算法结构特点对 5 种算法的收敛性问题进行分析.

在 Hybrid Predator-Prey 场景中, 捕食者应该首先明确需要移动的方位 (离散型变量), 在确定需要移动的增量 (连续型变量), 这样的决策机制才能更加高效的帮助捕食者捕获到猎物. 因此在该场景中, 策略网络应该优先决策离散型动作变量然后在决策连续型动作变量, 或者基于一套策略网络同时决策输出离散型动作变量和连续型动作变量, 基于这两种策略选择机制的决策算法能够更加适用于 Hybrid Predator-Prey 场景.

对于 Deep MAPQN 算法, 该算法的策略网络是基于 P-DQN 设计的, 而 P-DQN 需要先决策连续型动作在此基础上进一步决策离散动作. 因此 Deep MAPQN 的策略选择结构是影响其在 Hybrid Predator-Prey 场景中性能的重要原因.

对于 MAHSAC、MAHDDPG 算法, 虽然这两种算法的策略网络能够同时决策离散型动作和连续型动作, 但是 MAHSAC、MAHDDPG 采用的是 MADDPG 的中心化评估结构, 即所有智能体拥有单独的中心化评估网络, 通过各自的中心化评估网络引导各自的策略网络更新. 随着智能体数量增加, 会使得中心化评估网络的训练周期和训练成本也随之增加, 并且容易造成中心化评估网络在引导策略网络更新时陷入局部收敛的情况. 根据图 7(a)–图 7(c) 所示也可以发现在 Hybrid Predator-Prey 场景中 MAHSAC 和 MAHDDPG 算法随着智能体数量的增加, 其

性能也逐渐下降.

对于 P-FACMAC、Deep MAHHQN 算法, P-FACMAC 的策略网络是基于 PDDPG 设计的能有基于同一套策略网络同时输出离散动作和连续动作, Deep MAHHQN 则是先决策离散型动作再决策连续型动作, 两种算法的策略选择机制都适用于 Hybrid Predator-Prey 场景. 同时两种算法都是基于 QMIX 的中心化评估机制, 相较于 MADDPG 的中心化评估机制, 这类评估机制能够更加高效的引导所有智能体的策略朝着联合收益最大化的方向更新. 在根据图 1(b) 中的 Deep MAHHQN 算法结构图可知, Deep MAHHQN 算法分别使用了上层网络结构和下层网络结构. 相较于 P-FACMAC 的算法设计, Deep MAHHQN 包含了更多更复杂的网络结构, 这样设计的一个缺点就是随着智能体数量的增加、任务难度的提升, 该算法训练收敛的时间也将会更长.

3 P-FACMAC 在多巡航导弹协同突防中的应用

在第 2 节中介绍了一种应用于参数化动作空间的多智能体强化学习算法 P-FACMAC, 并且在开源强化学习训练环境 PettingZoo 中的 Hybrid Predator-Prey 场景验证了 P-FACMAC 方法对于解决多智能体参数化动作空间任务的可行性. 为了进一步验证 P-FACMAC 算法在解决面向实际任务时的有效性和可行性, 将 P-FACMAC 算法应用到多巡航导弹协同突防的智能决策任务场景中, 基于采样训练后的 P-FACMAC 模型, 能够智能决策控制多枚巡航导弹达到超过 80% 的突防成功率.

在本节的多巡航导弹协同突防任务场景中, 其智能决策的目标是控制巡航导弹尽可能绕飞规避威胁区, 或者以尽可能短的时间通过威胁区, 并且控制的巡航导弹必须要能够成功到达目标. 在多巡航导弹协同突防任务场景中, 控制的巡航导弹数量不少于 5 枚, 假设防御方部署了两类不同的反导拦截系统, 对应拦截区域 A 和 B. 其中威胁区 A 拦截区域半径为 100 km, 部署数量为 2 个; 威胁区 B 拦截区域半径为 50 km, 部署数量为 3 个. 通过有效控制巡航导弹在突防过程中侧向机动、纵向机动、悸动加减速、隐身、干扰 (其中悸动加减速、隐身、干扰执行有约束限制) 这 5 种突防策略, 实现巡航导弹对威胁区的规避、进入威胁区后对其中的拦截弹的规避并快速通过该威胁区、能准确到达目标.

3.1 多巡航导弹协同突防的 Dec-POMDP 建模

因为 P-FACMAC 算法是一种强化学习方法, 因此要将这种方法应用到多巡航导弹协同突防的任务场景中首先需要对整个突防任务进行 Dec-POMDP 建模, 即根据设计巡航导弹的观测空间、全局状态空间、动作空间以及奖励函数.

(1) 观测空间

本任务场景中巡航导弹的观测信息包括以下内容.

1) 巡航导弹 id (主要用于便于决策网络区分不同的巡航导弹); 2) 巡航导弹状态 (0: 未发射, 1: 正在飞行, 2: 成功击中目标, 3: 已被拦截); 3) 巡航导弹正在执行的策略; 4) 巡航导弹是否进入威胁区 (1: 未进入, 2: 已进入); 5) 对该巡航导弹威胁程度最高的威胁区 id (若巡航导弹进入威胁区, 则是所进入威胁区的 id); 6) 上述“威胁区威胁半径”与“巡航导弹与该威胁区距离”的比值; 7) 角度信息: 巡航导弹速度的朝向角度、巡航导弹与目标连线的夹角; 8) 巡航导弹与目标的相对位置信息: 巡航导弹经度与目标经度之差、巡航导弹纬度与目标纬度之差; 9) 巡航导弹前方一定阈值范围内的威胁区数量; 10) 巡航导弹距离目标的距离 (做归一化处理); 11) 悸动加减速、隐身、干扰 3 项突防策略可执行的约束; 12) 已被干扰的威胁区平台 id.

(2) 状态空间

本任务场景中状态空间的设计主要包含了巡航导弹相关信息以及威胁区相关的信息.

① 巡航导弹相关信息

巡航导弹 id、巡航导弹状态、巡航导弹正在执行的策略、对该巡航导弹威胁程度最高的威胁区 id、“威胁区威胁半径”与“巡航导弹与该威胁区距离”的比值、巡航导弹速度的朝向角度、巡航导弹与目标连线的夹角、航弹经度与目标经度之差、巡航导弹纬度与目标纬度之差、巡航导弹前方一定阈值范围内的威胁区数量、巡航导弹

与目标的距离、“悸动加减速、隐身、干扰”这 3 项突防策略可执行的约束.

② 威胁区相关信息

威胁区域编号 id、威胁区是否被干扰、威胁区的位置信息“经度、纬度”、威胁区域半径大小.

(3) 动作空间

在本任务场景中巡航导弹可控的策略参数包括: ① 机动: 决策速度倾角、速度方位角; ② 悸动加减速: 决策加减速时间; ③ 隐身: 决策隐身时间; ④ 干扰: 决策干扰时间. 在整个策略空间中速度倾角、速度方位角、加减速时间、隐身时间、干扰时间都可以视为连续型的动作变量. 在本任务场景中假设悸动加减速、隐身、干扰不能同时执行, 因此可以将当前时刻执行悸动加减速、隐身、干扰中的哪一种, 或者 3 种都不执行视为一组离散型的动作变量.

为了验证 P-FACMAC 在本任务场景中的算法性能, 可以将动作空间进一步设计为参数化动作空间, 具体的设计方法如下.

1) 3 类有约束上限策略的执行选项, 其中, 0: 不执行, 1: 执行悸动加减速, 2: 执行隐身, 3: 执行干扰 (离散型动作变量); 2) 决策速度倾角 (连续型动作变量); 3) 决策速度方位角 (连续型动作变量); 4) 决策悸动加减速时间 (连续型动作变量); 5) 决策隐身时间 (连续型动作变量); 6) 决策干扰时间 (连续型动作变量).

根据上述设置, 本任务场景中的动作空间可以被设计为包含一组离散型动作变量和五维连续动作变量的参数化动作空间. 在决策动作参数的过程中, 需要根据约束条件设计相应的 Action Mask[28]保证决策动作参数的合理性. 需要考虑的约束条件包括加减速、隐身、干扰的累积执行时长不能超过 20 个时间步长, 同一时刻只能执行加减速、隐身、干扰中的一种或者不执行.

(4) 奖励函数

本任务场景中的目标是在尽可能规避威胁区或用尽可能短的时间安全通过威胁区的前提下能够成功突防打击到作战目标. 基于此任务目标, 为巡航导弹的实时策略设计了 3 类子奖励函数, 分别为: 1) 与目标的相对位置关系奖励; 2) 通过威胁区的时间奖励; 3) 成功打击到目标的奖励. 3 类子奖励函数的具体设计如下.

① 与目标相对位置关系奖励

巡航导弹在向目标飞行的过程中, 规避威胁区时会影响巡航导弹驶向偏离目标的航线, 从而导致巡航导弹无法有效到达目标. 需要设计一个关于与目标相对位置关系的奖励函数, 当巡航导弹偏离目标轨迹过大时, 能有效引导其向目标轨迹回归. 该奖励主要包含两项指标, 一个是巡航导弹速度的朝向角度以及巡航导弹与目标连线的夹角, 两个角度之间的关系, 两个角度相对差异越大, 说明偏离目标程度越大. 另一个指标是巡航导弹与目标之间的相对距离, 该距离越大也说明偏离目标的程度越大.

基于上述两项指标, 与目标相对位置关系奖励的具体设计如下式所示, 其中各指标的符号定义为: θ 表示巡航导弹速度的朝向角 (单位: 弧度), μ 表示巡航导弹与目标连线的夹角 (单位: 弧度), d 表示巡航导弹与目标之间的相对距离 (单位: km), R_1 表示与目标相对位置关系奖励. R_1 的计算方式为:

$$R_1 = \cos(|\theta - \mu|)/(d/1000) - 0.5 \times (d/1000).$$

② 通过威胁区的时间奖励

当巡航导弹通过威胁区时, 在威胁区中飞行的时间越长, 则会有越高的概率被威胁区发射的拦截弹拦截, 因此希望巡航导弹尽可能规避威胁区, 或者运用悸动加减速、隐身、干扰等突防措施提高突防能力. 因此通过威胁区的时间奖励使用巡航导弹已经进入威胁区的时间作为刻画指标, 巡航导弹进入威胁区的时间越长, 则说明该枚巡航导弹越有可能被拦截.

基于上述这项指标, 通过威胁区的时间奖励的具体设计如下所示, 其中指标的符号定义为: T 表示已经进入威胁区的时间, R_2 表示通过威胁区的时间奖励. R_2 的计算方式为:

$$R_2 = \begin{cases} -0.08 \times T, & T \leqslant 50; \\ -4 - 0.04 \times (T - 50), & T > 50. \end{cases}$$

③ 成功打击目标的奖励

成功打击目标的奖励是用于评估巡航导弹是否成功到达目标的最终奖励, 使用 R_3 表示. 若巡航导弹成功到达目标, 则 $R_3=20$, 反之则 $R_3=0$. 当巡航导弹进入到目标区域的 30 km 的范围内时, 可以通过导弹自身的导引头实现对目标的精准制导, 因此当巡航导弹进入到目标区域的 30 km 的范围内时即可认为该导弹已经成功到达目标.

3.2 基于 P-FACMAC 智能协同突防决策算法设计与集成

在第 3.1 节中对整个多巡航导弹协同突防的任务场景进行了 Dec-POMDP 建模, 将巡航导弹的决策参数设计成了参数化的动作空间, 结合 P-FACMAC 策略网络中双头的输出结构设计了如图 8 所示的多巡航导弹协同突防的策略网络架构. 策略网络将观测信息作为输入, Type net 和 Param Net 分别决策输出离散型的动作变量 (突防策略的类型) 和连续型的动作变量 (突防策略的具体参数). 由于悸动加减速、隐身、干扰 3 项干扰策略在执行过程会受到一定的约束条件, 因此为 Type net 设计了 Action Mask 准则[30], 并且通过 Gumbel-Softmax 采样[26]的方式保证 Type net 能够按照 Action Mask 准则有效输出. 根据多巡航导弹仿真推演系统的响应决策指令的数据格式, 将 Type net 和 Param net 输出的决策参数进行联合编码, 保证其能被仿真推演系统所响应.

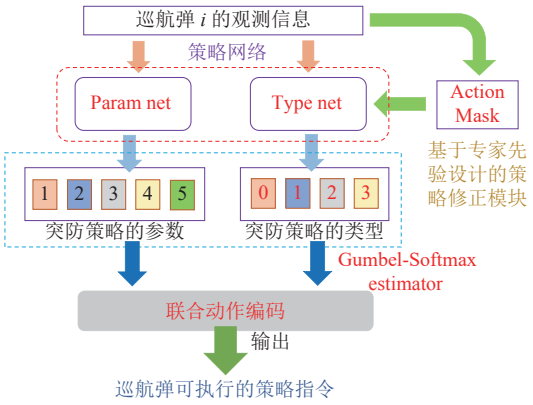


图 8 多巡航导弹协同突防任务中策略网络的架构

为了便于 P-FACMAC 智能决策算法的实现和训练, 本节基于 Python 编程语言进行 P-FACMAC 算法的设计与开发, 同时为了能够将基于 Python 的智能决策算法与基于 C++的多巡航导弹协同突防的仿真推演系统进行集成, 使用了 socket 中的 UDP 通信协议实现智能决策算法与仿真推演系统之间数据传输. 智能决策算法接收仿真系统当前的态势信息, 仿真系统则接收智能决策算法决策出的策略指令, 通过数据收发的方式实现了智能决策算法与仿真推演系统的实时交互. 集成了 P-FACMAC 智能决策算法的仿真推演系统总体结构如后文图 9 所示, 图 9 中包含了整套系统中数据的流向、训练样本的采集、智能算法的决策机制和训练架构等.

3.3 实验结果与分析

本节关于多巡航导弹协同突防的实验场景部署设置为, 在某区域内分别部署了 3 个发射点和 3 个目标点 (将管控的巡航导弹分别部署在 3 个发射点, 同时为每 1 枚巡航导弹设置 1 个需要到达的目标点), 在发射点和目标点之间部署了 2 个威胁区 A 以及 3 个威胁区 B. 所部署的多巡航导弹协同突防任务场景如图 10 所示.

在上述场景部署的基础上, 分别设置管控 5 枚巡航导弹和 6 枚巡航导弹进行协同突防的任务. 在管控不同数量巡航导弹进行协同突防的场景下, 对 P-FACMAC 算法在智能协同突防应用中的性能验证. 其中 P-FACMAC 算法在多巡航导弹协同突防场景的训练过程中超参数取值如表 2 所示.

根据第 2.2.2 节中的实验对比分析, 选择了在 Hybrid Predator-Prey 场景中与 P-FACMAC 性能持平的 Deep MAHHQN 算法作为 P-FACMAC 在多巡航弹协同突防场景中的对比算法. 在两组不同管控巡航导弹数量场景中

的训练效果如图 10 所示, 算法在训练过程中, 将每个回合的回合累积奖励记录下来并分别绘制成如图 11(a) 和图 11(b) 所示的回合奖励的变化趋势图. 根据图 11(a) 可知, 在 5 枚巡航导弹协同突防的实验中, P-FACMAC 算法经过大约 350 000 个回合的采样训练算法逐渐趋于收敛, Deep MAHHQN 算法经过大约 400 000 个回合的采样训练算法逐渐趋于收敛, 两者最终达到的收益函数基本一致. 根据图 11(b) 可知, 在 6 枚巡航导弹协同突防的实验中, P-FACMAC 算法经过大约 1 400 000 个回合的采样训练算法逐渐趋于收敛, Deep MAHHQN 算法经过大约 1 400 000 个回合的采样训练算法逐渐趋于收敛, 但是 Deep MAHHQN 收敛时的回合收益值低于 P-FACMAC 的回合收益值.

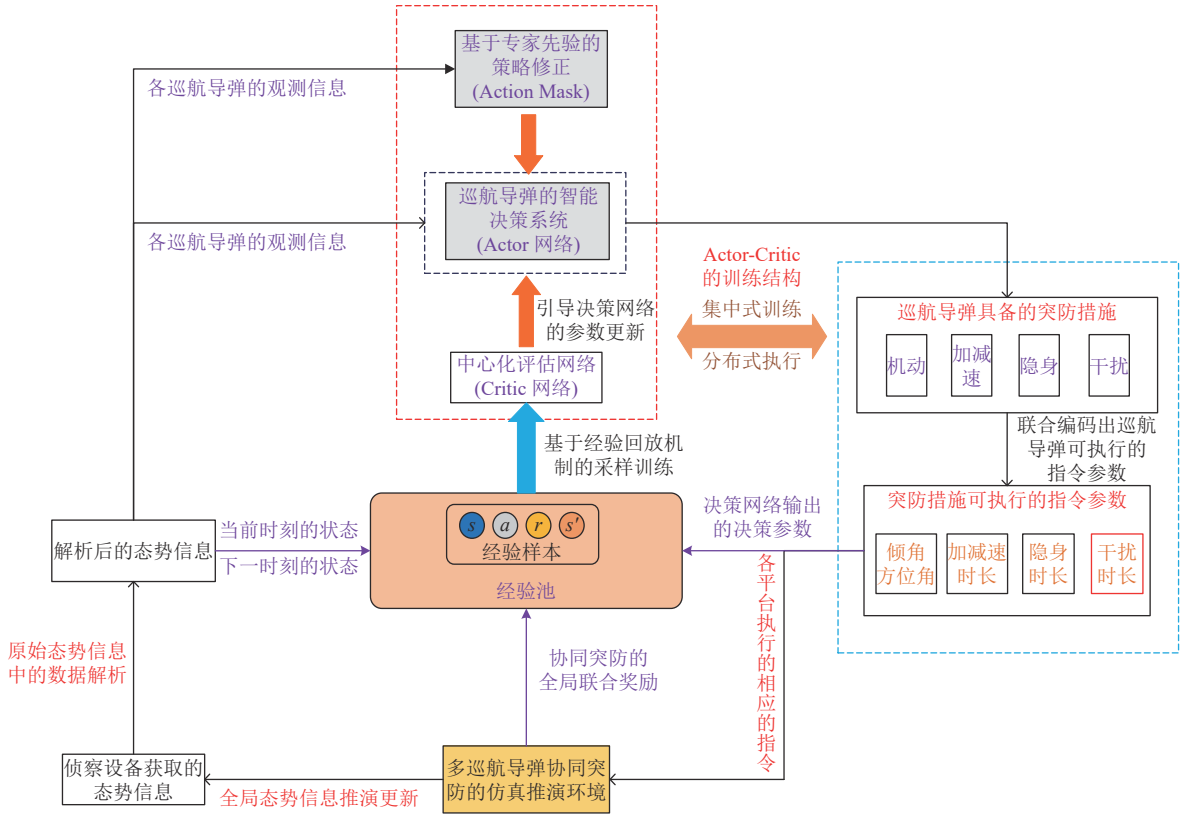


图 9　基于 P-FACMAC 智能决策算法的仿真推演系统总体结构

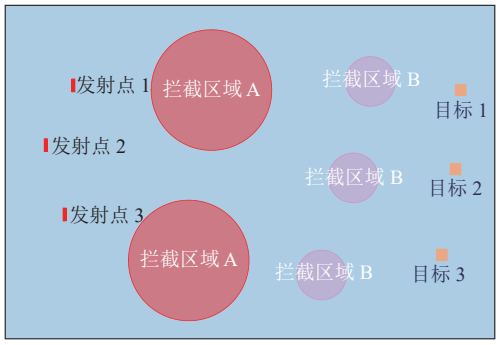
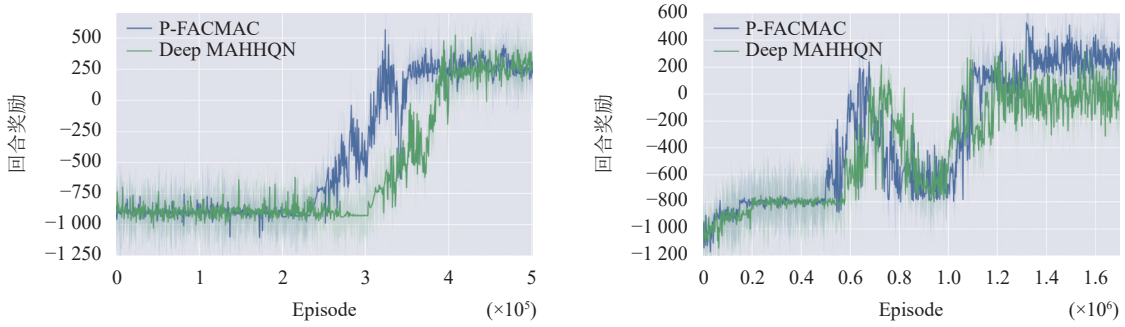


图 10　多巡航导弹协同突防任务场景图

表 2　多巡航导弹协同突防场景中 P-FACMAC 的训练超参数

超参数项	取值
参数 ω 和 $\{\theta_a\}_{a=1}^n$ 的优化器	Adam优化器
参数 ω 和 $\{\theta_a\}_{a=1}^n$ 的学习率	1E–4
参数 $\{\phi_a\}_{a=1}^n$ 的优化器	Adam优化器
参数 $\{\phi_a\}_{a=1}^n$ 的学习率	1E–4
混合网络 g 是否使用目标网络	是
智能体个体评估网络 $\{Q_a\}_{a=1}^n$ 是否使用目标网络	是
所有智能体个体评估网络是否共享网络参数	是
智能体个体策略网络 $\{\mu_a\}_{a=1}^n$ 是否使用目标网络	是
所有智能体个体策略网络是否共享网络参数	是
经验池大小	100 000
软更新系数 τ	0.01
折扣因子 γ	0.99
批采样大小batch_size	256
算法训练周期	每收集50帧样本训练1次



(a) 5 枚巡航导弹的协同突防实验　　(b) 6 枚巡航导弹的协同突防实验

图 11　多巡航导弹协同突防实验中回合奖励变化趋势

通过训练过程中记录的日志文件对两种算法在多巡航弹协同突防场景中的算法收敛性问题进行分析. 在 5 枚巡航导弹协同突防的实验中, 两种算法最终的收敛模型都能控制 5 枚巡航导弹全部到达对应的目标. 在 6 枚巡航导弹协同突防的实验中, P-FACMAC 最终的收敛模型能控制 6 枚巡航导弹全部到达对应的目标, Deep MAHHQN 最终的收敛模型只能控制 4–5 枚巡航到达对应的目标.

在 6 枚巡航导弹协同突防的实验中, P-FACMAC 算法在大约 500 000–900 000 回合训练中出现了一个先上升再下降的震荡趋势, 在这部分阶段中智能算法通过前期的学习逐渐能够控制了 4 枚巡航导弹到达目标, 另外还有 2 枚巡航导弹在目标区域边缘徘徊. 在本部分的实验设定中 P-FACMAC 的所有智能体共享策略网络参数, 在引导另外 2 枚巡航导弹向目标区域靠近的过程中, 对另外 4 枚巡航导弹的决策也产生不利因素从而导致这 4 枚导弹都未能到达目标区域, 相较于初始的探索阶段离各自的目标区域很远, 在这个训练期间 6 枚导弹虽然都没到达目标区域但是都在各自的目标区域附近探索. 又经过一段时间的采样训练, 策略网络逐渐掌握了如何让 6 枚导弹都能全部到达目标区域. Deep MAHHQN 算法在大约 600 000–1 000 000 回合训练中也出现了一个先上升再下降的震荡趋势, 震荡原因与 P-FACMAC 类似, 在震荡区间后 Deep MAHHQN 也逐渐开始上升, 最终的收敛效果是能稳定让 4 枚巡航导弹到达目标, 1 枚巡航导弹处于目标区域边界附近 (有时能进入目标区域, 有时则会偏离目标区域), 另外 1 枚巡航导弹则总是到达距离目标区域较远的地方.

根据图 11(a) 和图 11(b) 的算法效果, 可以发现与 Deep MAHHQN 相比 P-FACMAC 能够收敛得更快, 同时随着控制智能体的增加, 作战任务难度的提高, 在相同时间内 P-FACMAC 表现的性能更佳. 这进一步验证了第 2.2.2 节中的分析, Deep MAHHQN 的设计中包含了更多更复杂的网络结构, 而 P-FACMAC 中的网络设计更加简洁, 因此在面对智能体数量更多、任务复杂度更高的场景时, P-FACMAC 训练收敛的时间会更短.

为了验证上述训练好的 P-FACMAC 模型对于多巡航导弹协同突防决策的有效性, 基于训练收敛的 P-FACMAC 模型进行智能决策, 首先在训练场景下进行 500 次突防实验, 统计其协同突防成功率 (协同突防的成功是指所有的巡航导弹都未被拦截, 并且能到达目标区域 30 km 范围内, 否则认为协同突防未成功); 其次在两组新场景 (与训练场景相比, 新场景更改了发射点到达目标点的对应关系) 下分别进行 500 次的仿真实验, 统计并记录其协同突防成功率. 管控不同数量的巡航导弹分别在训练场景、新场景 1、新场景 2 下各进行 500 次突防实验的突防成功次数以及突防成功率如表 3 所示.

表 3　基于 P-FACMAC 智能决策算法的突防成功率

巡航导弹协同数量	训练场景		新场景1		新场景2	
	成功次数 (次)	成功率 (%)	成功次数 (次)	成功率 (%)	成功次数 (次)	成功率 (%)
5枚	500	100	437	87.3	428	85.6
6枚	500	100	410	82	417	83.4

4 结　论

强化学习方法的快速发展, 为智能决策的发展提供了新的学习范式. 虽然强化学习方法在棋牌、游戏等领域已经能够超越人类的职业选手, 但是在面对具有强对抗、多平台协同管控、决策参数多元化等特点的复杂军事对抗博弈场景时, 很多 MARL 方法表现出的能力还不足以应对挑战, 其核心原因之一是现有的 MARL 方法通常只能处理单一的离散或者连续动作空间, 而无法适用于更契合实际应用的参数化动作空间. 本文将处理参数化动作空间的 PDDPG 算法以及多智能体算法 FACMAC 进行耦合设计, 利用前者设计分布式策略网络, 利用后者设计中心化评估网络, 所提出的 P-FACMAC 算法能够有效应用于具有参数化动作空间的多智能体协同决策任务. 同时通过实验进一步验证了 P-FACMAC 对于解决多巡航导弹协同突防这类复杂决策任务下算法决策能力. 在未来, 我们还将探索面向更加多元化参数化决策动作且平台之间决策参数异构化的复杂场景中, 多智能强化学习方法的高效应用.

References:

[1] Liang XX, Feng YH, Ma Y, Cheng GQ, Huang JC, Wang Q, Zhou YZ, Liu Z. Deep multi-agent reinforcement learning: A survey. Acta Automatica Sinica, 2020, 46(12): 2537–2557 (in Chinese with English abstract). [doi: 10.16383/j.aas.c180372]

[2] Chai JJ, Li WF, Zhu YH, Zhao DB, Ma Z, Sun KW, Ding JSY. UNMAS: Multiagent reinforcement learning for unshaped cooperative scenarios. IEEE Trans. on Neural Networks and Learning Systems, 2023, 34(4): 2093–2104. [doi: 10.1109/TNNLS.2021.3105869]

[3] Nguyen TT, Nguyen ND, Nahavandi S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE Trans. on Cybernetics, 2020, 50(9): 3826–3839. [doi: 10.1109/TCYB.2020.2977374]

[4] Gupta JK, Egorov M, Kochenderfer MJ. Cooperative multi-agent control using deep reinforcement learning. In: Proc. of the 2017 Workshops on Autonomous Agents and Multiagent Systems. São Paulo: Springer, 2017. 66–83. [doi: 10.1007/978-3-319-71682-4_5]

[5] Zhang LX, Zhang RX, Wu T, Weng R, Han MH, Zhao Y. Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles. IEEE Trans. on Neural Networks and Learning Systems, 2021, 32(12): 5435–5444. [doi: 10.1109/TNNLS.2021.3084685]

[6] Tan T, Bao F, Deng Y, Jin A, Dai QH, Wang J. Cooperative deep reinforcement learning for large-scale traffic grid signal control. IEEE Trans. on Cybernetics, 2020, 50(6): 2687–2700. [doi: 10.1109/TCYB.2019.2904742]

[7] Kraemer L, Banerjee B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing, 2016, 190: 82–94. [doi: 10.1016/j.neucom.2016.01.031]

[8] Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S. QMIX: Monotonic value function factorisation for deep multi-

agent reinforcement learning. In: Proc. of the 35th Int'l Conf. on Machine Learning. Stockholm: PMLR, 2018. 4292–4301.

[9] Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I. Multi-agent actor-critic for mixed cooperative-competitive environments. In: Proc. of the 31st Int'l Conf. on Neural Information Processing Systems. Long Beach: Curran Associates Inc., 2017. 6382–6393.

[10] Masson W, Ranchod P, Konidaris GD. Reinforcement learning with parameterized actions. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. Phoenix: AAAI, 2016. 1934–1940. [doi: 10.1609/aaai.v30i1.10226]

[11] Hausknecht MJ, Stone P. Deep reinforcement learning in parameterized action space. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan, 2016. 1–12.

[12] Xiong JC, Wang Q, Yang ZR, Sun P, Zheng Y, Fu HB, Zhang T, Liu J, Liu H. Parametrized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. arXiv:1810.06394, 2018.

[13] Fan Z, Su R, Zhang WN, Yu Y. Hybrid actor-critic reinforcement learning in parameterized action space. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: AAAI, 2019. 2279–2285.

[14] Li BY, Tang HY, Zheng Y, Hao JY, Li PY, Wang Z, Meng ZP, Wang L. HyAR: Addressing discrete-continuous action reinforcement learning via hybrid action representation. In: Proc. of the 10th Int'l Conf. on Learning Representations. OpenReview.net, 2022. 1–22.

[15] Cui YM, Wang HX, Zheng CS, Hu RG. Pursuit-evasion game decision technology of highspeed vehicles. Journal of Command and Control, 2021, 7(4): 403−414 (in Chinese with English abstract).

[16] Zhang WM, Huang SP, Huang JC, Zhu C, Ding ZY. Analysis on multi-domain operation and its command and control problems. Command Information System and Technology, 2020, 11(1): 1–6 (in Chinese with English abstract). [doi: 10.15908/j.cnki.cist.2020.01.001]

[17] Gao A, Dong ZM, Ye HB, Song JH, Guo QS. Loitering munition penetration control decision based on deep reinforcement learning. Acta Armamentarii, 2021, 42(5): 1101–1110 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-1093.2021.05.023]

[18] Liang XX, Feng YH, Huang JC, Wang Q, Ma Y, Liu Z. Novel deep reinforcement learning algorithm based on attention-based value function and autoregressive environment model. Ruan Jian Xue Bao/Journal of Software, 2020, 31(4): 948–966 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5930.htm [doi: 10.13328/j.cnki.jos.005930]

[19] Oliehoek FA, Amato C. A Concise Introduction to Decentralized POMDPs. Cham: Springer, 2016. [doi: 10.1007/978-3-319-28929-8]

[20] Fu HT, Tang HY, Hao JY, Lei ZH, Chen YF, Fan CJ. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: AAAI, 2019. 2329–2335.

[21] Hua HZ, Zhao RW, Wen GX, Wu KG. A further exploration of deep multi-agent reinforcement learning with hybrid action space. In: Proc. of the 32nd Int'l Conf. on Artificial Neural Networks. Heraklion: Springer, 2023. 1–12. [doi: 10.1007/978-3-031-44223-0_1]

[22] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D. Continuous control with deep reinforcement learning. In: Proc. of the 4th Int'l Conf. on Learning Representations. San Juan, 2016. 1–14.

[23] Delalleau O, Peter M, Alonso E, Logut A. Discrete and continuous action representation for practical RL in video games. arXiv:1912.11077, 2019.

[24] Peng B, Rashid T, de Witt CAS, Kamienny PA, Torr PHS, Böhmer W, Whiteson S. FACMAC: Factored multi-agent centralised policy gradients. In: Proc. of the 35th Conf. on Neural Information Processing Systems. 2021. 12208–12221.

[25] Foerster JN, Farquhar G, Afouras T, Nardelli N, Whiteson S. Counterfactual multi-agent policy gradients. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 2974–2982. [doi: 10.1609/aaai.v32i1.11794]

[26] Jang E, Gu SX, Poole B. Categorical reparameterization with Gumbel-Softmax. In: Proc. of the 5th Int'l Conf. on Learning Representations. Toulon: OpenReview.net, 2017. 1–13.

[27] Mahajan A, Rashid T, Samvelyan M, Whiteson S. MAVEN: Multi-agent variational exploration. In: Proc. of the 33rd Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 684.

[28] Rashid T, Farquhar G, Peng B, Whiteson S. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In: Proc. of the 34th Int'l Conf. on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 855.

[29] Samvelyan M, Rashid T, de Witt CS, Farquhar G, Nardelli N, Rudner TGJ, Hung CM, Torr PHS, Foerster J, Whiteson S. The StarCraft multi-agent challenge. In: Proc. of the 18th Int'l Conf. on Autonomous Agents and MultiAgent Systems. Montreal: Int'l Foundation for Autonomous Agents and Multiagent Systems, 2019. 2186–2188.

[30] Huang S, Ontañón S. A closer look at invalid action masking in policy gradient algorithms. In: Proc. of the 35th Int'l Florida Artificial Intelligence Research Society Conf. Beach, 2022. [doi: 10.32473/flairs.v35i.130584]

附中文参考文献:

[1] 梁星星, 冯旸赫, 马扬, 程光权, 黄金才, 王琦, 周玉珍, 刘忠. 多 Agent 深度强化学习综述. 自动化学报, 2020, 46(12): 2537–2557. [doi: 10.16383/j.aas.c180372]

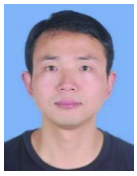
[15] 崔雅萌, 王会霞, 郑春胜, 胡瑞光. 高速飞行器追逃博弈决策技术. 指挥与控制学报, 2021, 7(4): 403−414.

[16] 张维明, 黄松平, 黄金才, 朱承, 丁兆云. 多域作战及其指挥控制问题探析. 指挥信息系统与技术, 2020, 11(1): 1–6. [doi: 10.15908/j.cnki.cist.2020.01.001]

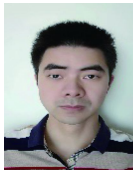
[17] 高昂, 董志明, 叶红兵, 宋敬华, 郭齐胜. 基于深度强化学习的巡飞弹突防控制决策. 兵工学报, 2021, 42(5): 1101–1110. [doi: 10.3969/j.issn.1000-1093.2021.05.023]

[18] 梁星星, 冯旸赫, 黄金才, 王琦, 马扬, 刘忠. 基于自回归预测模型的深度注意力强化学习方法. 软件学报, 2020, 31(4): 948–966. http://www.jos.org.cn/1000-9825/5930.htm [doi: 10.13328/j.cnki.jos.005930]

田树聪(1996－), 男, 博士生, 主要研究领域为多智能体强化学习, 安全强化学习.



周正春(1978－), 男, 博士, 教授, 博士生导师, 主要研究领域为机器学习, 电子信息对抗.



谢愈(1982－), 男, 博士, 副研究员, 主要研究领域为智能规划与决策, 飞行任务规划.

高阳(1972－), 男, 博士, 教授, 博士生导师, CCF 杰出会员, 主要研究领域为强化学习, 多智能体系统, 计算机视觉, 大数据分析.

张远龙(1989－), 男, 博士, 讲师, 主要研究领域为飞行器智能规划与决策.