计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications [doi: 10.15888/j.cnki.csa.009636]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

优化的协作多智能体强化学习架构①

刘　玮, 程　旭, 李浩源

(南京信息工程大学 计算机学院、网络空间安全学院, 南京 210044)
通信作者: 刘　玮, E-mail: 2821908422@qq.com

摘　要: 在现实环境中, 许多任务需要多个智能体的协作来完成, 然而智能体之间通常存在着通信受限和观察不完整的问题. 深度多智能体强化学习 (Deep-MARL) 算法在解决这类具有挑战性的场景中表现出卓越的性能. 其中 QTRAN 和 QTRAN++是能够学习一类广泛的联合动作-价值函数的代表性方法, 且同时具备强大的理论保证. 然而, 由于依赖于单一联合动作-价值估计量以及忽视了对智能体观察的预处理, 使得 QTRAN 和 QTRAN++的性能受到了影响. 本文提出了一种称为 OPTQTRAN 的新算法, 其在 QTRAN 和 QTRAN++的性能基础上取得了显著的提升. 首先, 本文引入了一种双联合动作-价值估计量的结构, 利用一个分解网络模块计算额外的联合动作-价值. 为了确保准确计算联合动作-价值, 本文设计了一个自适应网络模块, 有效促进了值函数学习. 此外, 本文引入了一个多元网络结构, 将智能体的观察分组到不同的单元中, 以有效估计各智能体的效用函数. 在广泛使用的 StarCraft 基准测试中进行的多场景实验表明, 与最先进的多智能体强化学习方法相比, 本文的方法表现出更卓越的性能.

关键词: 强化学习; 智能博弈; 多智能体强化学习; 智能体协作

引用格式: 刘玮,程旭,李浩源.优化的协作多智能体强化学习架构.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/9636.html

Optimized Architecture for Cooperative Multi-agent Reinforcement Learning

LIU Wei, CHENG Xu, LI Hao-Yuan

(School of Computer Science, Nanjing University of Information Science & Technology, Nanjing 210044, China)

Abstract: Numerous real-world tasks require the collaboration of multiple agents, often with limited communication and incomplete observations. Deep multi-agent reinforcement learning (Deep-MARL) algorithms show remarkable effectiveness in tackling such challenging scenarios. Among these algorithms, QTRAN and QTRAN++ are representative approaches capable of learning a broad class of joint-action value functions with strong theoretical guarantees. However, the performance of QTRAN and QTRAN++ is hindered by their reliance on a single joint action-value estimator and their neglect of preprocessing agent observations. This study introduces a novel algorithm called OPTQTRAN, which significantly improves upon the performance of QTRAN and QTRAN++. Firstly, the study proposes a dual joint action-value estimator structure that leverages a decomposition network module to compute additional joint action-values. To ensure accurate computation of joint action-value estimators, it designs an adaptive network that facilitates efficient value function learning. Additionally, it introduces a multi-unit network that groups agent observations into different units for effective estimation of utility functions. Extensive experiments conducted on the widely-used StarCraft benchmark across diverse scenarios demonstrate that the proposed approach outperforms state-of-the-art MARL methods.

Key words: reinforcement learning (RL); intelligent games; multi-agent reinforcement learning (MARL); agent collaboration

① 收稿时间: 2024-02-27; 修改时间: 2024-05-06; 采用时间: 2024-05-09; csa 在线出版时间: 2024-09-24

1 介绍

在过去的 10 年中, 多智能体强化学习 (MARL) 问题引起了显著的关注. 在 MARL 中, 系统内的智能体学习如何协调以获得最大化累积的全局奖励[1]. 许多复杂的现实问题, 如控制机器人群体[2]和自动驾驶[3], 可以被建模为合作型 MARL 问题, 并在商业应用中具有巨大潜力. 然而, 合作型 MARL 受到可扩展性和部分可观察性问题的阻碍. 随着代理数量的不断增加, 联合状态-动作空间呈指数级增长, 导致了网络训练的维度灾难. 此外, 智能体必须根据局部的动作-观察历史做出个体决策, 然而这并不能准确反映整体系统的真实状态.

为了解决上述问题, 一种称为“中心化训练分布式执行”(CTDE) 的流行范式被提出. CTDE 范式在训练期间通过汇总系统中的所有信息来优化个体的分散策略, 从而提供了完全集中的值函数. 一旦训练完成, 执行时智能体只需要考虑自己的局部动作-观察历史进行决策. 最近, 诸如 VDN[4]和 QMIX[5]等方法均采用了 CTDE 范式. 然而, 由于值函数分解具有单调约束性, 这两种方法在联合状态-动作空间评估全局 Q 值时可能面临困难. 因此, QTRAN[6]提出了一种更一般的分解形式, 即采用软正则化进行约束以保证分解的准确性与通用性. 由于优化问题的约束在计算上是难以处理的, 因此, QTRAN 必须使用两个惩罚项来放宽这些约束, 结果产生了次优解. 为了解决 QTRAN 在理论和实际应用中的局限性, QTRAN++被提出, 其采用了非固定估计量、更精细的损失函数和用于联合动作-价值估计量的混合网络架构. 这些变化显著提升了网络的收敛性能. 然而, 由于 QTRAN 和 QTRAN++过于依赖单一的联合动作-价值估计量, 它们在复杂环境中的效果受到限制, 这可能导致收敛速度缓慢与估计不准确, 且 QTRAN++的半单调混合网络进一步加剧了这一问题. 此外, QTRAN 和 QTRAN++忽略了对各智能体观察信息的预处理, 而是直接将它们输入神经网络, 使得整个网络需要大量的反向传播来学习到最优解.

本文提出一种新的基于值的算法, 称为 OPTQTRAN, 它缓解了上述提到的种种限制. 该算法在 StarCraft multi-agent challenge (SMAC)[7], 一个复杂的 MARL 环境中取得了最先进的结果. OPTQTRAN 对智能体的训练过程进行了关键性修改, 首先, 本文提出了一种新颖的双联合动作-价值估计量的结构, 与 QTRAN 和 QTRAN++明显不同. 具体来说, 该结构引入了一个分解网络模块来计算额外的联合动作-价值并优化整体损失函数. 此外, 本文引入了一个自适应网络来计算联合动作-价值, 解决了 QTRAN++的半单调混合网络中估计偏差的问题. 最后, 本文引入了一个多单元网络模块, 将智能体的观察分组到不同的单元中, 以使效用函数估计量携带更多基本信息. 本文在 SMAC 环境中进行了对 OPTQTRAN 方法的广泛评估, 将其与 6 个 MARL 基准算法在 6 个不同场景中进行比较. 值得注意的是, OPTQTRAN 在所有实验中都取得了最先进的性能. 此外, 本文进行了消融实验以展示 OPTQTRAN 算法中每个模块的有效性. 实验结果表明, 每个模块在实现 OPTQTRAN 的卓越性能方面都发挥了关键作用. 根据本文的实验, 我们相信 OPTQTRAN 方法可以成为未来研究 MARL 任务的强大基准.

2 相关工作

CTDE 范式在多智能体强化学习领域取得了显著的流行, 其假设在训练期间可以获得完整的全局状态信息, 而训练完成后智能体可以仅根据局部观测信息来执行策略. 目前许多种类的方法都采用了 CTDE 范式来训练智能体, 包括基于策略和基于值的方法. 基于策略的方法通常采用演员-评论家 (actor-critic algorithm) 框架, 其中独立的演员用于实现分布式执行. 反事实的多智能体策略梯度 (COMA)[8]通过估算反事实基线来解决信用分配问题, 并使用联合评论家网络训练个体策略. MADDPG[9]扩展了 DDPG[10]算法, 以一种集中的方式学习合作和竞争性游戏的个体策略. 另一个例子是 MAAC[11], 它在评论家网络中引入了注意力机制以增强可扩展性. 最近, ROMA[12]提出了一个以角色为导向的框架, 利用带有正则化项的深度强化学习和角色条件策略来学习角色. FOP[13]将最大熵多智能体强化学习分解为局部策略, 并证明了其收敛性. 此外, IRAT[14]构建了个体策略和团队策略, 并应用差分约束以缓解稀疏奖励.

在基于值的方法中, VDN 将联合动作-价值估计量表示为各个智能体的效用函数之和. QMIX 通过引入混合网络来捕捉联合动作-价值估计量中各智能体效用函数之间的非线性单调关系, 以此对 VDN 进行了扩展, 但也保留着和 VDN 一样的问题, 即没有摆脱单调

性假设关系. Qatten 引入了多头注意力机制, 在理论研究发现的基础上对联合动作-价值估计量进行了近似分解. QPLEX[15]则采用了双工对抗网络架构来分解联合价值函数. RODE[16]提出了一种基于角色的多智能体强化学习方法, 利用角色选择器将联合动作空间分解为与不同角色对应的有限动作空间. PAC[17]提出了一种使用最优联合动作生成反事实预测的方法, 涉及编码局部观察、优化交互信息以及使用变分推断生成辅助信息. 最近, QTRAN 被提出以消除 QMIX 中的联合动作-价值估计量中的单调性假设问题. QTRAN 通过神经网络的训练目标强制将联合动作-价值估计量分解为各智能体效用函数之和, 而不是直接将其分解. 然而, QTRAN 在理论基础和实际应用之间仍存在差距, 为了解决这个问题提出了 QTRAN++. 首先, 通过修改优化损失函数, QTRAN++稳定了训练过程. 其次, 通过利用一个非固定的真实动作-价值估计量, 使得在训练中其和转化的联合动作-价值函数间进行模仿学习, 大幅提升训练效率.

另外, 最近已经提出了几种方法来解决 QMIX 局限性的问题. Mahajan 等人[18]提出了一种专门的探索算法 MAVEN 来解决智能体的探索问题. Yang 等人[19]提出了 Q-DPP 算法来激励智能体获得多样化的行为模型并协调它们之间的探索行为. Rashid 等人[20]则提出了 CW-QMIX 和 OW-QMIX 算法, 通过使用加权投影让智能体能学习到更多更好的联合动作. 最后, Pan 等人[21]提出了 RES 算法, 通过向基准 Q 值添加一个正则项来惩罚偏离以此改善 QMIX 的过度估计问题.

3 准备工作

3.1 分散部分可观察马尔可夫决策过程 (DEC-POMDP)

本文将一个完全合作的多智能体任务建模为一个由元组$G=\langle \mathbb{N},S,A,P,\Omega,r,\gamma\rangle$定义的 DEC-POMDP, 其中$\mathbb{N}=\{1,2,\cdots,n\}$是智能体的有限集合, $s\in S$ 是全局状态的有限集合. 在每个时间步, 每个智能体$i\in\mathbb{N}$在全局状态s上选择一个动作$a_i\in A\equiv\{A^{(1)},\cdots,A^{(|A|)}\}$, 形成一个联合动作$a\equiv[a_i]_{i=1}^{n}\in A\equiv A^n$. 之后可以得到一个联合奖励$r(s,a)$并且使当前全局状态转移到下一个全局状态$s'\sim P(\cdot\mid s,a)$. $\gamma\in[0,1)$是一个折扣因子. 本文考虑一个部分可观察的设置, 其中每个智能体i根据观测概率函数$O(o_i\mid s,a_i)$收到一个个体的部分观测$o_i\in\Omega$. 每个智能体 i 有一个动作-观测历史$\tau_i\in T\equiv(\Omega\times A)^*$, 并构建其个体策略$\pi_i(a\mid\tau_i)$来共同最大化团队性能. 本文使用$\tau\in T\equiv T^n$表示联合动作-观测历史. 形式化的目标函数是找到一个联合策略$\pi=\langle\pi_1,\cdots,\pi_n\rangle$来最大化联合价值函数$V^{\pi}(s)=E\left[\sum_{t=0}^{\infty}\gamma^t r_t\mid s_0=s,\pi\right]$. 在策略搜索中的另一个感兴趣的点是联合动作-价值函数$Q^{\pi}(s)=r(s,a)+\gamma E_{s'}[V^{\pi}(s')]$.

3.2 个体-全局最大 (IGM) 条件

IGM 原则[5]已经成为实现基于值的 CTDE 的一种流行的方法. 其具体表现为以中心化的方式训练智能体, 使智能体能够访问其他智能体或全局状态的信息, 并且在分布式执行阶段, 每个智能体能够基于其局部的动作-观察历史做出自己的决策. 通过使用 IGM 原则, 智能体做出的决策可以在个体性能与全局最优性间做出平衡, 从而提高系统的整体性能. 本文使用$Q_{jt}(\boldsymbol{\tau},\boldsymbol{a})$表示联合动作-价值, $[Q_i(\tau_i,a_i)]_{i=1}^{n}$表示个体动作-价值, $i\in\mathbb{N}$. 因此, IGM 被定义如下:

$$\forall\boldsymbol{\tau}\in T,\arg\max Q_{jt}(\boldsymbol{\tau},\boldsymbol{a})=\left(\arg\max_{a_1\in A}Q_1(\tau_1,a_1),\cdots,\arg\max_{a_n\in A}Q_n(\tau_n,a_n)\right)\tag{1}$$

4 方法

在本节中, 本文将展示 OPTQTRAN 的算法框架. 首先, 在第 4.1 节中对算法的网络结构进行概述, 然后在第 4.2 节中介绍分解网络模块, 即实现了双联合动作-价值估计量结构并为此设计一个新的损失函数, 最后, 在第 4.3 节中, 将呈现两个模块: 一个用于联合动作-价值估计量的自适应网络模块, 以及另一个用于效用函数的多单元网络模块.

4.1 网络结构概述

如图 1 所示, 本文采用了超网络 (hypernetwork)[22]为 3 个混合网络生成权重参数. 超网络是采用一个神经网络来为另一个神经网络生成权重参数的网络结构. 需要注意的是为普通混合网络 (common mixing network) 生成的权重参数是任意值, 而为单调混合网络 (monotonic mixing network) 生成的权重参数是大于 0 的, 即 QMIX 中所使用的网络结构. 本文将普通混合网络和单调混合网络分别定义为$f_{\text{mix}}(Q_1,\cdots,Q_n;\theta_{jt}^{(1)}(s))$和$f_{\text{mix}}(Q_1,\cdots,Q_n;\theta_{jt}^{(2)}(s))$, 其中$f_{\text{mix}}$代表了超网络中使用的全连接层网络. 之后, 将$\tau_i$和$a_i$输入多元网络模块

(multi-unit module) 进行模块化处理, 然后将其输出作为效用函数的输入, 最终得到效用函数Q_i (utility Q_i). 接着将Q_i分别送入分解网络模块 (decomposition module) 和 3 个混合网络, 其输出分别定义为Q'_{jt}, Q^1_{jt}, Q^2_{jt}和Q_{tran}. 通过将Q^1_{jt}和Q^2_{jt}输入自适应网络模块 (adaptive module), 最终得到联合动作-价值估计量Q_{jt}.

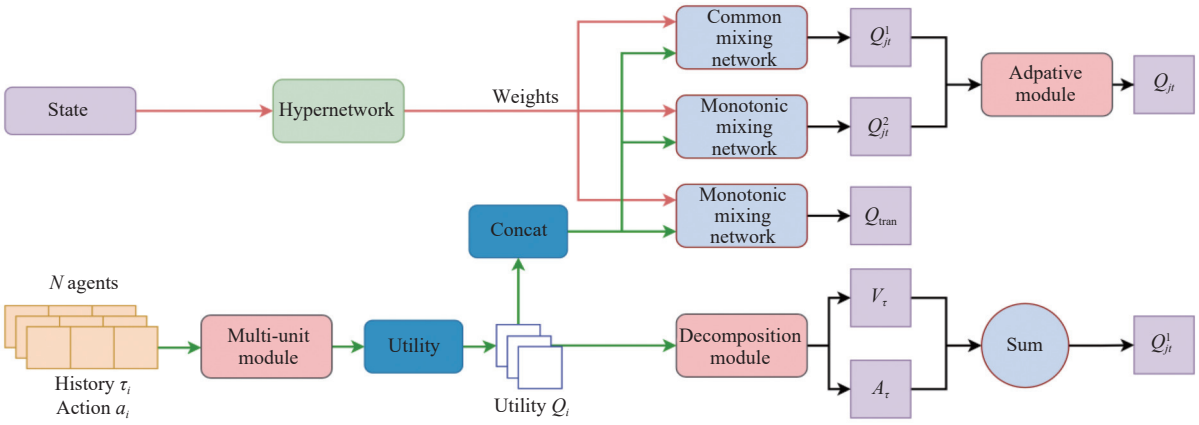


图 1 整体算法结构

4.2 分解网络模块

如图 2 所示, 本文将效用函数的输出值$Q_i(\tau_i,a_i)$作为分解网络模块的输入, 之后将$Q_i(\tau_i,a_i)$分解为两个单独的函数, 即状态价值函数$V_i(\tau_i)$和优势函数$A_i(\tau_i,a_i)$. 这一对抗结构在 QPLEX 中已经被证明了其有效性. 为了计算每个智能体的$A_i(\tau_i,a_i)$和$V_i(\tau_i)$, 本文采用式 (2) 和式 (3):

$$A_i(\tau_i,a_i)=Q_i(\tau_i,a_i)-V_i(\tau_i) \tag{2}$$

$$V_i(\tau_i)=\max_{a_i}Q_i(\tau_i,a_i) \tag{3}$$

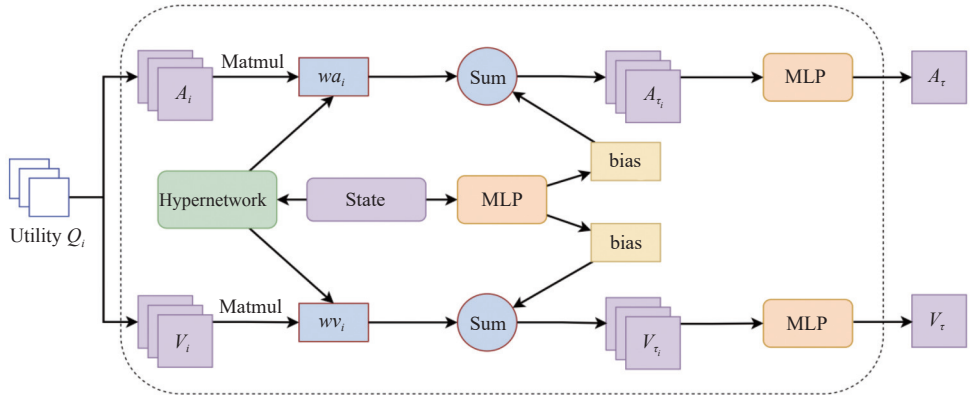


图 2 分解网络模块结构

然后为了将每个智能体的动作-观测历史τ_i扩展为联合动作-观测历史$\boldsymbol{\tau}$, 本文采用超网络为优势函数和状态价值函数分别生成参数wa_i和wv_i, 超参数的输入值为全局状态s. 此处wa_i和wv_i的取值范围不加限制, 为任意值. 之后, 根据式 (4) 和式 (5) 就得到了在联合动作-观测历史$\boldsymbol{\tau}$下的状态价值函数和优势函数:

$$A_i(s,\boldsymbol{\tau},a_i)=wa_i(s)A_i(\tau_i,a_i)+b_i(s) \tag{4}$$

$$V_i(s,\boldsymbol{\tau})=wv_i(s)V_i(\tau_i)+b_i(s) \tag{5}$$

之后本文采用两个多层网络感知机 (MLP) 将$V_i(s,\boldsymbol{\tau})$转换为$V_{\text{tot}}(s,\boldsymbol{\tau})$, $A_i(s,\boldsymbol{\tau},a_i)$转换为$A_{\text{tot}}(s,\boldsymbol{\tau},\boldsymbol{a})$, 同时为了最大化$A_{\text{tot}}(s,\boldsymbol{\tau},\boldsymbol{a})$和$V_{\text{tot}}(s,\boldsymbol{\tau})$的泛化性, 本文对 MLP 参数的正负不加以限制. 最后对联合动作-价值函数$Q'_{jt}(s,\boldsymbol{\tau},\boldsymbol{a})$做如下表述:

$$Q'_{jt}=V_{\text{tot}}(s,\boldsymbol{\tau})+A_{\text{tot}}(s,\boldsymbol{\tau},\boldsymbol{a}) \tag{6}$$

其中, 本文对联合动作-价值函数做了符号简化, 即将$Q'_{jt}(s,\boldsymbol{\tau},\boldsymbol{a})$定义为$Q'_{jt}$. Q'_{jt}是对$Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a})$的另一种表达

方式, 引入它是为了在 QTRAN++训练目标的基础上加入一个新的训练目标L_{adv}. 其数学表达如下所示:

$$\begin{cases} L_{\text{adv1}} = (Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) - Q'_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}))^2 \\ L_{\text{adv2}} = (V_{\text{tot}}(s,\boldsymbol{\tau}) - (Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) + Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})) \times 0.5)^2 \\ L_{\text{adv3}} = (Q'_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) - (r + \gamma Q_{jt}^{\text{target}'}(s',\boldsymbol{\tau}',\bar{\boldsymbol{a}}')))^2 \\ L_{\text{adv}} = \lambda_{\text{adv1}} L_{\text{adv1}} + \lambda_{\text{adv2}} L_{\text{adv2}} + \lambda_{\text{adv3}} L_{\text{adv3}} \end{cases} \tag{7}$$

其中, $\bar{\boldsymbol{a}}$表示“最优的动作”, 由于$\boldsymbol{a} \equiv [a_i]_{i=1}^n$, 即$\bar{\boldsymbol{a}}$表示每个智能体都选取最优的动作, 反映到函数上就是$Q_i(\tau_i, a_i)$选取最大值. 同理, $\tilde{\boldsymbol{a}}$则表示“最差的动作”. 为了加快整体网络的收敛速度, 本文采用损失函数L_{adv1}来训练Q_{jt}和Q'_{jt}, 训练时不对二者进行固定, 使它们相互模仿学习. 这种方法的好处是Q_{jt}可以从Q'_{jt}中学习到更多采样, Q'_{jt}作为一个指导者的角色, 同时可以防止两者训练时大幅度的相互偏移, 维持训练的稳定.

为了加快训练初期的速度, 本文采用损失函数L_{adv2}将$V_{\text{tot}}(s,\boldsymbol{\tau})$的值限制在$Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}})$和$Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})$之间, 在强化学习理论中$V_{\text{tot}}(s,\boldsymbol{\tau})$的值也是在二者之间, 而此处本文进行了强制约束. 如图 3 所示, 真实的$V_{\text{tot}}(s,\boldsymbol{\tau})$值用绿色标记, 由于联合动作空间的指数增长, 实际算法中并不直接计算该真实值而是采用各种近似值代替.

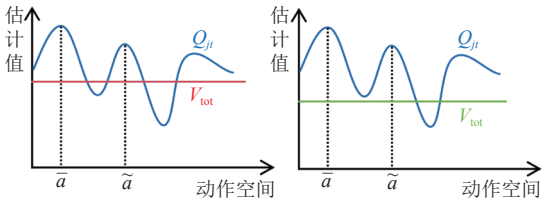


图 3 V_{tot} 的近似估计

此处, 本文采用红色的线来给$V_{\text{tot}}(s,\boldsymbol{\tau})$一个训练初期的初始值, 即$Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}})$和$Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})$之和的中间值, 同时约束$Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}})$的值大于$Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})$. 更新过后的损失函数$L_{\text{newadv2}}$定义如下:

$$\begin{aligned} L_{\text{newadv2}} =& (Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) - \max\{Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}), Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})\})^2 \\ &+ L_{\text{adv2}} \end{aligned} \tag{8}$$

其中, 在训练时本文将$(Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) + Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})) \times 0.5$和$Q_{jt}(s,\boldsymbol{\tau},\tilde{\boldsymbol{a}})$进行固定, 得益于对$V_{\text{tot}}(s,\boldsymbol{\tau})$的限制, 根据式 (6) 训练$Q'_{jt}$将变得更容易, 接下来根据损失函数$L_{\text{adv3}}$采用强化学习中的标准时序差分学习对$Q'_{jt}$进行更新. 此外, 本文对损失函数$L_{\text{adv1}}$和$L_{\text{adv2}}$添加了$\lambda$值进行约束, 并保留了 QTRAN++的损失函数$L_{\text{QTRAN++}}$, 其定义如下:

$$L_{\text{QTRAN++}} = L_{\text{td}} + \lambda_{\text{opt}} L_{\text{opt}} + \lambda_{\text{nopt}} L_{\text{nopt}} \tag{9}$$

$$L_{\text{opt}} = (Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) - Q_{\text{tran}}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}))^2 \tag{10}$$

$$\begin{cases} L_{\text{nopt}} = \begin{cases} (Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) - Q_{\text{tran}}(s,\boldsymbol{\tau},\boldsymbol{a}))^2, \\ \quad \text{if } Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) \geqslant Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) \\ (Q_{clip}(s,\boldsymbol{\tau},\boldsymbol{a}) - Q_{\text{tran}}(s,\boldsymbol{\tau},\boldsymbol{a}))^2, \\ \quad \text{if } Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) < Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}}) \end{cases} \\ Q_{clip}(s,\boldsymbol{\tau},\boldsymbol{a}) = clip(Q_{\text{tran}}(s,\boldsymbol{\tau},\boldsymbol{a}), Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}), Q_{jt}(s,\boldsymbol{\tau},\bar{\boldsymbol{a}})) \end{cases} \tag{11}$$

$$L_{\text{td}} = (Q_{jt}(s,\boldsymbol{\tau},\boldsymbol{a}) - (r + \gamma Q_{jt}^{\text{target}}(s',\boldsymbol{\tau}',\bar{\boldsymbol{a}}')))^2 \tag{12}$$

其中, $clip(\cdot, L_1, L_2)$函数表示将输入限制在$[L_1, L_2]$之间. $(s,\boldsymbol{\tau},\boldsymbol{a})$和$(s',\boldsymbol{\tau}',\bar{\boldsymbol{a}}')$表示从马尔可夫决策过程中连续时间步收集的对象. $\bar{\boldsymbol{a}}' = [\bar{a}'_1, \cdots, \bar{a}'_N]$是一组最优动作, 使效用函数取得最大值, 即$\bar{a}'_i = \arg\max_{a_i} Q_i(\tau_i, a_i), i \in \mathbb{N}$. 综上, 本文算法的损失函数可以如下定义:

$$L = L_{\text{QTRAN++}} + \lambda_{\text{adv1}} L_{\text{adv1}} + \lambda_{\text{adv2}} L_{\text{adv2}} + \lambda_{\text{adv3}} L_{\text{adv3}} \tag{13}$$

4.3 自适应网络模块与多元网络模块

在本节中将具体介绍自适应网络模块与多元网络模块的架构, 以及它们对网络结构优化所起的重要作用. 首先, 自适应网络模块为Q_{jt}^1和Q_{jt}^2增加了外部系数, 其目的是加强Q_{jt}的泛化性, 使得Q_{jt}可以更准确地逼近其真实值. 其次多元网络模块通过为输入添加额外有价值的信息来提升效用函数估计量的训练效率.

● 自适应网络模块. 在 QTRAN++中, 其只是将Q_{jt}^1与Q_{jt}^2进行简单相加来得到Q_{jt}, 与之不同, 本文采用超网络来为Q_{jt}^1与Q_{jt}^2生成权重参数. 在图 4 中本文将指出 QTRAN++的局限性, 其中蓝色曲线 Non-Monotonic-Q 和红色曲线 Monotonic-Q 分别代表Q_{jt}^1和Q_{jt}^2, 前者表示网络采用非单调性参数, 后者表示网络采用单调性参数. 图 4 中灰色曲线表示 Non-Monotonic-Q+Monotonic-Q 的值, 绿色曲线表示Q_{jt}可能的真实值, 简单的相加形式很大程度上不能良好的逼近真实值. 如果真实的Q_{jt}中有一些“陡峭”的值, 这一误差情况可能会进一步被放大.

本文对自适应网络模块的详细阐述如图 5. 上方的超网络将Q_{jt}^2作为输入, 下方的超网络将Q_{jt}^1作为输入. 本文通过 abs 绝对值函数将权重 Weight 1 和 Weight 2

设为正值, 即让神经网络来调整Q_{jt}^1和Q_{jt}^2的正负, 而权重则控制二者比例大小. 之后将 Weight 1 和 Weight 2 拼接在一起与Q_{jt}^1和Q_{jt}^2进行矩阵相乘, 最后相加得到Q_{jt}. 为了防止权重过大或者过小, 本文使用归一化函数将 Weight 1 和 Weight 2 的大小规范化到 0–2 之间. 改进后的Q_{jt}定义如下:

$$Q_{jt} = w_1(Q_{jt}^2)Q_{jt}^1 + w_2(Q_{jt}^1)Q_{jt}^2 + b(s) \tag{14}$$

其中, w_1和w_2是对 Weight 1 和 Weight 2 的简写. $b(s)$代表了偏差项, 是一个以全局状态s为输入的 MLP 所得到的值. 本文所提出的自适应网络模块使得Q_{jt}有着“补偿”特性, 即超网络生成权重时必须考虑其输入的影响, 使得梯度在更新时朝着目标方向移动. 如权重w_1必须考虑Q_{jt}^2的影响, 在一些任务场景中, w_1可能需要对此进行“补偿”, 为了使梯度准确更新而降低自身的值. 这样可以确保联合动作-价值估计量被优化到其可能达到的最准确的值.

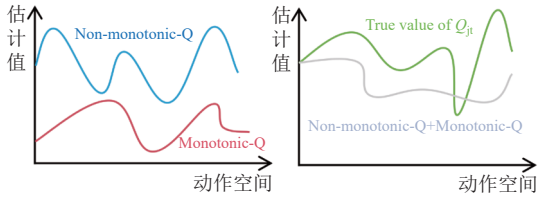


图 4　QTRAN++的局限性

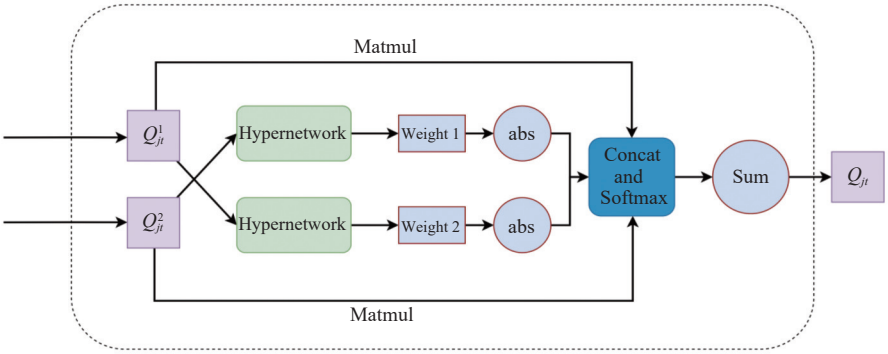


图 5　自适应网络模块结构

● 多元网络模块. 在许多 MARL 任务中, 如 SMAC 基准环境, 效用函数通常采用 DRQN 网络为每个智能体生成$Q_i(\tau_i, a_i)$. 需要指出的是个体智能体的观测τ_i会随着环境总体智能体数量的增加而成倍增加, 这会导致维度灾难, 从直观上来讲, 输入神经网络的数据维度越多整体训练速度就越慢. 因此为了缓解这一问题, 本文采用多元网络模块对输入进行模块化处理.

模块的具体细节如图 6 所示. 这一方法的主要思路是为 DRQN 网络[23]的输入增加一个可训练的参数, 以此来缓解学习压力. 首先, 智能体的观测τ_i由 3 部分组成, 即智能体自身状态, 友军状态, 敌军状态, 假设智能体的观测维度为m, 本文将观测进行主动分类, 标记为 self-unit, ally-unit, 和 enemy-unit, 每一个单元的维度取决于其相对应的属性维度. 之后采用 3 个单独的超网络为每一个单元生成权重, 超网络的输入为每一个单元本身. 由于一次性输入 DRQN 网络的智能体观测数量为待训练智能体的数量, 因此需要对每一个智能体的观测进行模块化处理. 此处定义单元的观测体数量为(j,k,n), 即对于一个智能体来说j=1 代表自身数量为 1, k=友军数量, n=敌军数量, 其各自维度为(ds,da,de), 所以对于 self-unit 的超网络的输入维度是(j,ds), 同理对于 ally-unit 的超网络输入是(k,da), 对于 enemy-unit 的超网络输入是(n,de). 注意$m = j\times ds + k\times da + n\times de$. 超网络的输出表示为$(V,D,H)$, $V \in (j,k,n)$代表单元的观测体数量, $D \in (ds,da,de)$表示每个单元的维度, H则代表了所需要的 DRQN 网络的输出维度. 每个超网络依据上述关系可以得出自身输出, 分别定义输出参数为ws, wa, we. 最后将得到的参数与各单元自身相乘, 然后将得到的乘积值与偏差项相加以获得最终输出, 其中的偏差项与自适应网络模块中的相同. 相比于将输入直接传入 DRQN, 明显本文的方法更具有优势, 可以使输入携带更多有效的信息, 如超网络所提供的参数可以提供对每个模块单独的评估信息, 即权重的大小, 在训练时为了使损失最小, 超网络必须对每一个单

元进行合理的评估, 给出合适的权重. 此外多元网络模块可以实现各单元精确的梯度传递, 并允许按每个单元的需求进行参数调整, 最终实现更高效的参数更新和更好的强化学习性能.

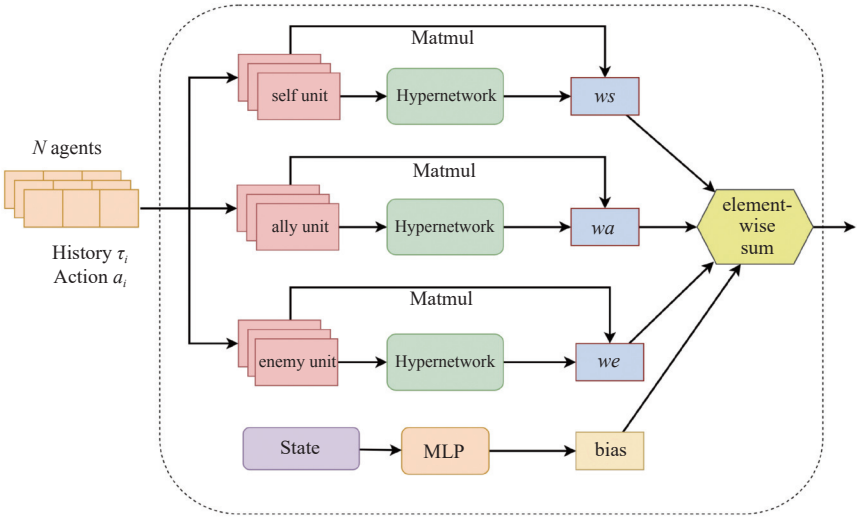


图 6 多元网络模块结构

5 实验

5.1 矩阵游戏

在本节中将研究图 7 与图 8 中两个具有挑战性的矩阵游戏, 它们描绘了基本的合作型多智能体任务, 在这两个游戏中想要跳出局部最优的情况是十分困难的. 整体实验设置包括了两个智能体且每个智能体可以选择 3 个动作, 其中最优的联合动作为$(A^{(1)},A^{(1)})$. 为了在联合动作空间中收集足够多的数据, 本文采用了均匀数据分布, 通过这个预先设定的数据集, 我们可以从优化的角度来集中研究多智能体关于 Q 值学习的最优性. 表中的所有算法结果都经过了完全探索, 使用了探索因子ε-greedy, ε初始值为 1, 在 5 000 个时间步内逐步衰减, 确保所有可能的潜在状态都被完全探索. 图 7 显示, 除了 QPLEX 和 QMIX 收敛到局部最优, 其他算法都达到了最佳效果. 此处的 QPLEX 采用了 MLP 网络生成重要性权重而不是多头注意力机制, 其原文中也论述了两种方法的效果基本没有差异. QPLEX 与 OPTQTRAN-Q'_{jt}相比, 表明重要性权重这一特性一定程度上影响了表达能力. QTRAN++-Q_{jt}和 OPTATRAN-Q_{jt}之间的对比表明, 后者对于一些特定状态的估计更为准确, 如联合动作$(A^{(2)},A^{(2)})$. 在图 8 中, 所有算法都收敛到了最优值, 除了 QMIX. 此外, OPTATRAN-Q_{jt}对于大部分的值估计得都比 QTRAN++-Q_{jt}准确, 如对于联合动作$(A^{(2)},A^{(2)})$的估计, 这可能由于 QTRAN++ 中采用了与 QMIX 结构相同的混合网络, 而 QMIX 对于这些联合动作的估计也不准确. 相反, OPTQTRAN 采用了双联合动作-价值估计量的结构有效解决了这一问题.

5.2 在线数据收集训练

5.2.1 实验设置

在 SMAC 环境中对不同的算法进行评估, 这是一个在线训练平台, SMAC 模拟了由不同策略控制的智能体之间的战斗场景, 并且将所有训练需要的信息实时返回. 每一个智能体接收一个本地观测, 其中包括了自身和附近所有可视单位的信息, 如彼此间的距离、位置、生命值、护盾等. 此外, 在智能体训练期间还会提供一个包含了所有智能体信息的全局状态. 本文将 OPTQTRAN 与 6 个主流的基准算法 (QMIX、QTRAN、VDN、Qatten、QPLEX 和 QTRAN++) 在 6 个不同难度级别的地图场景进行了比较. 所有智能体的策略网络都采用 DRQN 结构, 包含了两个 64 维的全连接层和一个 64 维的 GRU. 混合网络是一个 32 维度的隐藏层, 且使用 ELU 激活函数. 超网络采用两个 64 维的全连接层, 激活函数使用 ReLU. 同时使用学习率为 0.0005 的 RMSProp 优化器对所有神经网络进行训练, 动作选择时使用逐渐减小的探索因子ε-greedy, 在 500 000 个时间步内从 1 逐步减小到 0.05. 对于折扣因子γ将其设置为 0.99. 重放缓冲区的大小设置为 5 000 轮, mini-batch 的大小设置为 32. 训练时间根据地图场景的不同, 范围为 8–24 h, 采用 NVIDIA 3080 显卡进行训练.

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**8.00**	−12.00	−12.00
$A^{(2)}$	−12.00	0.00	0.00
$A^{(3)}$	−12.00	0.00	0.00

(a) 矩阵1

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**8.00**	−11.99	−11.99
$A^{(2)}$	−11.99	3.33	1.81
$A^{(3)}$	−11.99	3.43	1.81

(b) Q_{jt} of QTRAN++

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	3.99	−11.98	−11.96
$A^{(2)}$	−11.97	0.01	0.00
$A^{(3)}$	−11.98	**4.00**	0.00

(c) Q_{jt} of QPLEX

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**7.95**	−11.97	−11.97
$A^{(2)}$	−11.97	0.00	0.00
$A^{(3)}$	−11.97	0.00	0.00

(d) Q_{jt} of OPTQTRAN-Q'_{jt}

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**7.97**	−12.00	−12.00
$A^{(2)}$	−12.02	0.00	0.00
$A^{(3)}$	−12.02	0.00	0.00

(e) Q_{jt} of OPTQTRAN-Q_{jt}

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	−7.99	−7.99	−7.99
$A^{(2)}$	−7.99	−0.03	−0.03
$A^{(3)}$	−7.99	**−0.02**	−0.03

(f) Q_{jt} of QMIX

图 7 矩阵游戏 1

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**8.00**	−12.00	−12.00
$A^{(2)}$	−12.00	6.00	0.00
$A^{(3)}$	−12.00	0.00	6.00

(a) 矩阵2

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**8.00**	−11.99	−12.00
$A^{(2)}$	−12.00	2.99	3.00
$A^{(3)}$	−12.00	2.99	3.00

(b) Q_{jt} of QTRAN++

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**7.99**	−11.99	−11.99
$A^{(2)}$	−11.99	6.00	0.00
$A^{(3)}$	−11.99	0.00	6.00

(c) Q_{jt} of QPLEX

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**8.00**	−12.00	−10.35
$A^{(2)}$	−11.99	5.99	0.00
$A^{(3)}$	−12.00	0.00	5.99

(d) Q_{jt} of OPTQTRAN-Q'_{jt}

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	**7.99**	−11.97	−11.16
$A^{(2)}$	−12.00	6.04	0.03
$A^{(3)}$	−12.01	−0.01	5.91

(e) Q_{jt} of OPTQTRAN-Q_{jt}

a_2 \ a_1	$A^{(1)}$	$A^{(2)}$	$A^{(3)}$
$A^{(1)}$	−7.99	−7.99	−7.99
$A^{(2)}$	−7.99	3.00	**3.00**
$A^{(3)}$	−7.99	3.00	3.00

(f) Q_{jt} of QMIX

图 8 矩阵游戏 2

每经过 10 000 个时间步对算法进行性能评估, 将探索因子设置为 0, 使智能体停止随机决策而是根据训练的网络输出策略, 然后执行 32 轮对抗测试, 基于测试的胜率评估算法性能, 即我方智能体成功击败敌方智能体的轮数在 32 轮测试中所占的百分比. 本文展示的结果是 5 次独立实验的中位性能, 带有 25%–75% 的阴影置信区间. 其中分解网络模块的超网络层数为 2, 维度为 64, 该模块中的所有 MLP 均为 2 层且每层 32 维. 自适应网络模块中超网络和生成偏差的 MLP 均为 3 层且每层 32 维. 多元网络模块的超网络为 2 层且每层 64 维. 损失函数权重为$\lambda_{\text{opt}} = 2$, $\lambda_{\text{nopt}} = 1$, $\lambda_{\text{adv1}} = 0.2$, $\lambda_{\text{adv2}} = 0.2$, $\lambda_{\text{adv3}} = 1.0$.

5.2.2 实验结果

实验结果对比如图 9 所示. 本文的评估涵盖了一系列具有挑战性的任务场景, 现有的方法难以在不同的场景中保持一致的性能, 而 OPTQTRAN 始终能够达到几近最优的结果. 例如, 在图 9(a) 中, QTRAN++、Qatten 和 QMIX 存在局部最优问题, 表现为胜率一开始增加, 然后突然急剧下降. 相反, OPTQTRAN 学习到了一种有效的策略, 并避免了这个问题. 在图 9(b) 和图 9(c) 中, 两支队伍拥有相同类型的智能体, 但对手数量比我方多出一个, 在这种情况下, 即使是策略选择中的微小偏差也可能导致失败, 需要智能体在进攻和生存间做出平衡. 值得注意的是, 虽然 QTRAN 获得了很高的胜率, 但仍然不及 OPTQTRAN 的表现. 在图 9(d) 中呈现了这样一个场景, 其中 3 个我方追随者必须在大部分回合中引诱敌对狂热者, 导致延迟奖励问题. OPTQTRAN 比除了 QPLEX 之外的所有其他基

准算法都表现优异. 在图 9(e) 和图 9(f) 的高度具有挑战性的场景中, 这些场景要求智能体之间进行有效的协作决策, 本文的算法相对于基准算法表现出优越的性能. 特别是图 9(f) 对其他算法而言, 这是一个相当困难的情况, 而本文的算法仍旧可以取得一定的胜率.

可视化分析如图 10. 在图 10(a) 和 10(b) 中, 双方各剩下 4 个智能体, 以人的角度分析, 此时最优的策略应该是集中火力攻击敌对单位, 然而, QTRAN++倾向于学习一个“局部最优”策略, 即智能体远离敌对单位以避免受伤. 相反, OPTQTRAN 学到了“最佳”策略, 即所有智能体都攻击敌人.

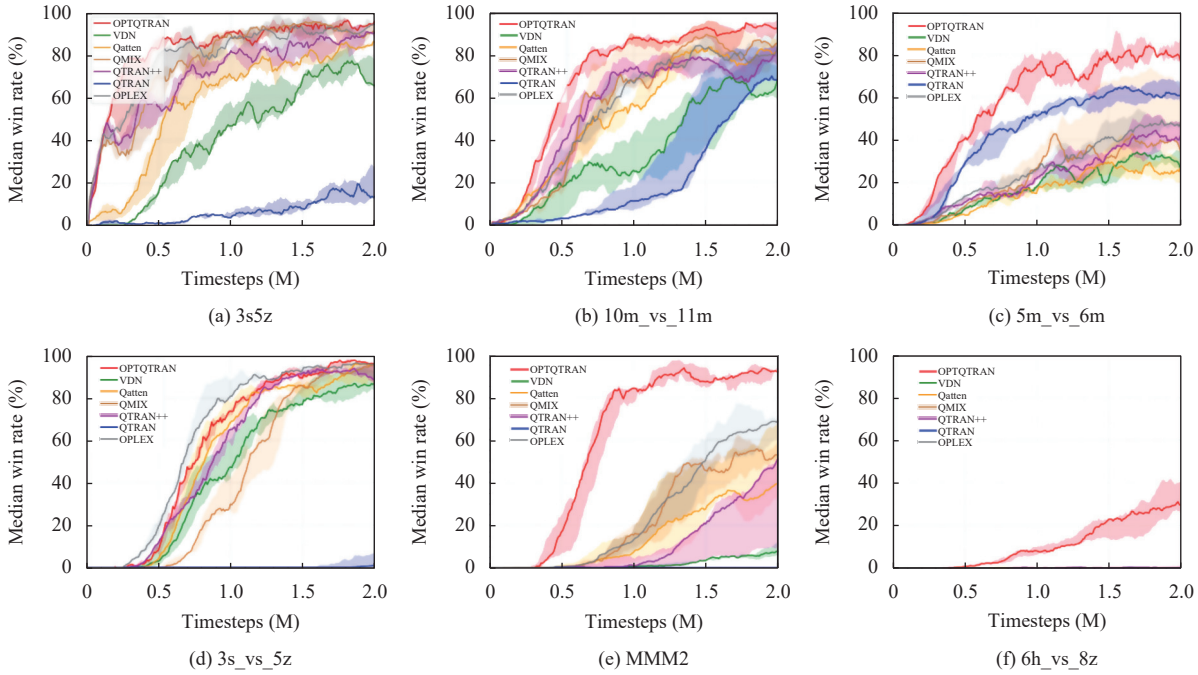


图 9 实验结果对比

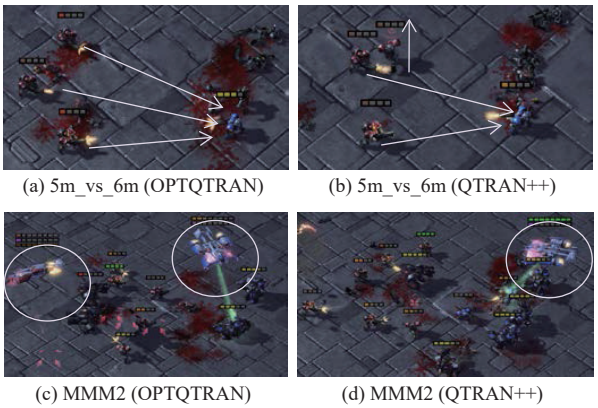


图 10 可视化策略分析

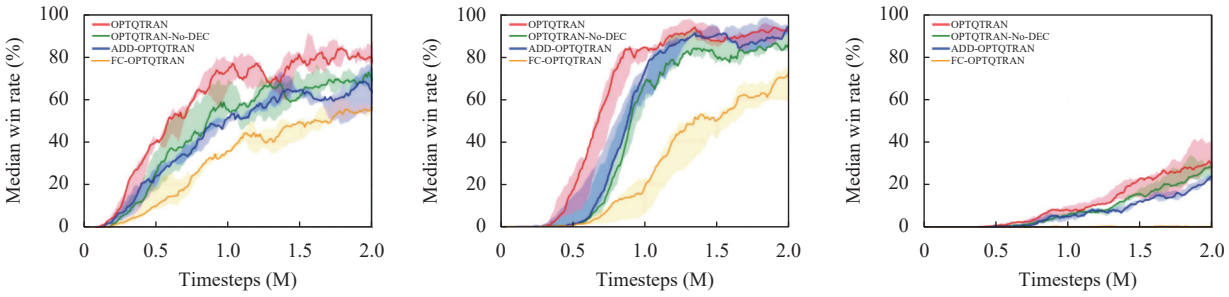
此外, 在图 10(c) 和图 10(d) 中, 想要获胜必须在摧毁敌方治疗单位的同时保护我方的治疗单位. 在这种情况下, 生命值并不是关键因素, 保护重要的友方单位和摧毁重要的敌方单位才至关重要. QTRAN++显然未能学到该获胜的策略, 反而在对局开始时选择快速歼灭敌方普通单位, 陷入局部最优导致了最终的失败. 相反, OPTQTRAN 成功地避免了这一情况, 选择在保护治疗单位的同时攻击敌方治疗单位. 综上实验结果表明, 在这些具有挑战性的情景中, OPTQTRAN 显著优于 QTRAN++.

5.2.3 消融实验

消融实验旨在评估 OPTQTRAN 方法中每个组件的影响. 首先我们移除了分解网络模块, 引入了只使用一个联合动作-价值估计量的 OPTQTRAN-NO-DEC 变体. 接下来, 我们将 OPTQTRAN 与 ADD-OPTQTRAN 进行比较, 后者采用了一个加法形式计算 Q_{jt}^1、Q_{jt}^2 来替换自适应模块. 此外, 我们通过与 FC-OPTQTRAN 进行比较来评估多元网络模块的影响, FC-OPTQTRAN 采用了 MLP 网络替代超网络. 这种系统的评估使我们能够深入了解我们方法中这些改进的有效性.

在图 11 中展示了消融实验的结果, 即 OPTQTRAN

的每个模块在改进其性能方面的有效性, 实验共涵盖了 3 种不同的任务场景. 从结果中, 我们观察到分解和自适应网络模块在一定程度上提高了整体模型的性能, 而多元网络模块在实现卓越性能方面发挥了至关重要的作用. 我们认为多单元网络模块之所以能够显著加速训练, 是由于超网络所带来的额外有效信息. 在图 11(a)、(b) 中可以明显看出在去除了分解网络模块和替换了自适应网络模块后, 模型整体的性能出现了一定程度的下滑, 而在图 11(c) 中, 用 MLP 网络替换超网络导致性能显著下降, 更突显了超网络在为每个单元高效生成适当权重中所发挥的关键作用.



(a) 5m_vs_6m (b) MMM2 (c) 6h_vs_8z

图 11 消融实验

6 总结

本文提出了一种基于 CTDE 执行范式的协作多智能体强化学习方法并命名为 OPTQTRAN. 该方法具有许多突出优势, 包括通过对损失函数和网络结构的优化来加速训练, 通过先进的联合动作-价值估计量的计算来提高网络泛化性能, 以及将多元网络模块用于更具信息性的效用函数估计量. 为了评估 OPTQTRAN 的有效性, 本文在具有挑战性的 SMAC 环境中进行了实验. 实证结果表明, 该方法建立了一个新的卓越性能基准, 这些发现为开发更高效的 MARL 算法提供了宝贵的见解, 期待本文的工作能够继续激发该领域更多的研究.

参考文献

1 Busoniu L, Babuska R, De Schutter B. A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2008, 38(2): 156–172. [doi: 10.1109/TSMCC.2007.913919]

2 Yogeswaran M, Ponnambalam SG, Kanagaraj G. Reinforcement learning in swarm-robotics for multi-agent foraging-task domain. Proceedings of the 2013 IEEE Symposium on Swarm Intelligence. Singapore: IEEE, 2013. 15–21.

3 Cao YC, Yu WW, Ren W, *et al*. An overview of recent progress in the study of distributed multi-agent coordination. IEEE Transactions on Industrial Informatics, 2013, 9(1): 427–438. [doi: 10.1109/TII.2012.2219061]

4 Sunehag P, Lever G, Gruslys A, *et al*. Value-decomposition networks for cooperative multi-agent learning based on team reward. Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems. Stockholm, 2018. 2085–2087.

5 Rashid T, Samvelyan M, De Witt CS, *et al*. Monotonic value function factorisation for deep multi-agent reinforcement learning. Journal of Machine Learning Research, 2020, 21(178): 1–51.

6 Son K, Kim D, Kang WJ, *et al*. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. Proceedings of the 36th International Conference on Machine Learning. Long Beach: PMLR, 2019. 5887–5896.

7 Samvelyan M, Rashid T, de Witt CS, *et al*. The StarCraft multi-agent challenge. Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems. Montreal, 2019. 2186–2188.

8 Foerster J, Farquhar G, Afouras T, *et al*. Counterfactual multi-agent policy gradients. Proceedings of the 32nd AAAI Conference on Artificial Intelligence. New Orleans: AAAI, 2018. 2974–2982.

9 Lowe R, Wu Y, Tamar A, *et al*. Multi-agent actor-critic for mixed cooperative-competitive environments. Proceedings of the 31st International Conference on Neural Information Processing Systems. Long Beach: Curran Associates Inc.,

2017. 6382–6393.
10 Lillicrap TP, Hunt JJ, Pritzel A, *et al*. Continuous control with deep reinforcement learning. Proceedings of the 4th International Conference on Learning Representations. San Juan: ICLR, 2016.
11 Iqbal S, Sha F. Actor-attention-critic for multi-agent reinforcement learning. Proceedings of the 36th International Conference on Machine Learning. Long Beach: PMLR, 2019. 2961–2970.
12 Wang TH, Dong H, Lesser V, *et al*. ROMA: Multi-agent reinforcement learning with emergent roles. Proceedings of the 37th International Conference on Machine Learning. PMLR, 2020. 9876–9886.
13 Zhang TH, Li YH, Wang C, *et al*. FOP: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. Proceedings of the 38th International Conference on Machine Learning. PMLR, 2021. 12491–12500.
14 Wang L, Zhang YP, Hu YJ, *et al*. Individual reward assisted multi-agent reinforcement learning. Proceedings of the 39th International Conference on Machine Learning. Baltimore: PMLR, 2022. 23417–23432.
15 Wang JH, Ren ZZ, Liu T, *et al*. QPLEX: Duplex dueling multi-agent Q-learning. Proceedings of the 9th International Conference on Learning Representations. OpenReview.net, 2021.
16 Wang TH, Gupta T, Mahajan A, *et al*. RODE: Learning roles to decompose multi-agent tasks. Proceedings of the 9th International Conference on Learning Representations. OpenReview.net, 2021.
17 Zhou HH, Lan T, Aggarwal V. PAC: Assisted value factorisation with counterfactual predictions in multi-agent reinforcement learning. Proceedings of the 36th International Conference on Neural Information Processing Systems. New Orleans: Curran Associates Inc., 2022. 1146.
18 Mahajan A, Rashid T, Samvelyan M, *et al*. MAVEN: Multi-agent variational exploration. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2019. 684.
19 Yang YD, Wen Y, Wang J, *et al*. Multi-agent determinantal Q-learning. Proceedings of the 37th International Conference on Machine Learning. PMLR, 2020. 10757–10766.
20 Rashid T, Farquhar G, Peng B, *et al*. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2020. 855.
21 Pan L, Rashid T, Peng B, *et al*. Regularized Softmax deep multi-agent Q-learning. Proceedings of the 34th International Conference on Neural Information Processing Systems. Vancouver: Curran Associates Inc., 2021. 1365–1377.
22 Ha D, Dai AM, Le QV. Hypernetworks. Proceedings of the 5th International Conference on Learning Representations. Toulon: OpenReview.net, 2017.
23 Hausknecht M, Stone P. Deep recurrent Q-learning for partially observable MDPs. Proceedings of the 2015 AAAI Fall Symposia. Arlington: AAAI, 2015. 29–37.

(校对责编: 张重毅)