计 算 机 研 究 与 发 展
Journal of Computer Research and Development
DOI: 10.7544/issn1000-1239.202440189
61(8): 1904−1916, 2024

图多智能体任务建模视角下的协作子任务行为发现

李 超 李文斌 高 阳
(计算机软件新技术全国重点实验室(南京大学)南京 210023)
(chaoli1996@smail.nju.edu.cn)

Discovering Coordinated Subtask Patterns from a Graphical Multi-Agent Task Modeling Perspective

Li Chao, Li Wenbin, and Gao Yang
(*State Key Laboratory for Novel Software Technology* (*Nanjing University*), *Nanjing* 210023)

Abstract Numerous multi-agent tasks exhibit a nearly decomposable structure, wherein interactions among agents within the same interaction set are strong while interactions between different sets are weak. Efficiently modeling this structure and leveraging it to coordinate agents can enhance the learning efficiency of multi-agent reinforcement learning algorithms for cooperative multi-agent tasks, while existing work typically neglects and fails. To address this limitation, we model the nearly decomposable structure using a dynamic graph and accordingly propose a novel algorithm named coordinated subtask pattern (CSP) that enhances both local and global coordination among agents. Specifically, CSP identifies agents' interaction sets as subtasks and utilizes a bi-level structure to periodically distribute agents into multiple subtasks, which ensures accurate characterizations regarding their interactions on the dynamic graph. Based on the subtask assignment, CSP proposes intra-subtask and inter-subtask pattern constraints to facilitate both local and global coordination among agents. These two constraints ensure that partial agents within the same subtask are aware of their action selections and all agents select superior joint actions that maximize the overall task performance. Experimentally, we evaluate CSP across multiple maps of SMAC benchmark, and its superior performance against multiple baseline algorithms demonstrates its effectiveness on efficiently coordinating agents.

Key words multi-agent reinforcement learning; cooperative tasks; nearly decomposable structure; dynamic graph; coordination

摘 要 大量多智能体任务都表现出近似可分解结构,其中相同交互集合中智能体间交互强度大,而不同交互集合中智能体间交互强度小.有效建模该结构并利用其来协调智能体动作选择可以提升合作型多智能体任务中多智能体强化学习算法的学习效率.然而,目前已有工作通常忽视并且无法有效实现这一目标.为解决该问题,使用动态图来建模多智能体任务中的近似可分解结构,并由此提出一种名叫协作子任务行为(coordinated subtask pattern, CSP)的新算法来增强智能体间局部以及全局协作.具体而言,CSP算法使用子任务来识别智能体间的交互集合,并利用双层策略结构来将所有智能体周期性地分配到多个子任务中.这种分配方式可以准确刻画动态图上智能体间的交互关系.基于这种子任务分配,CSP算法提出

收稿日期: 2024-03-16; **修回日期**: 2024-05-10
基金项目: 国家自然科学基金项目(62192783, 62106100, 62276142);江苏省自然科学基金项目(BK20221441);江苏省产业前瞻与关键核心技术竞争项目(BE2021028);深圳市中央引导地方科技发展资金项目(2021Szvup056)
This work was supported by the National Natural Science Foundation of China (62192783, 62106100, 62276142), the Natural Science Foundation of Jiangsu Province (BK20221441), the Primary Research and Development Plan of Jiangsu Province (BE2021028), and the Shenzhen Fundamental Research Program (2021Szvup056).
通信作者: 李文斌(liwenbin@nju.edu.cn)

子任务内和子任务间行为约束来提升智能体间局部以及全局协作. 这2种行为约束确保相同子任务内的部分智能体间可以预知彼此动作选择，同时所有智能体选择优异的联合动作来最大化整体任务性能. 在星际争霸环境的多个地图上开展实验，实验结果表明CSP算法明显优于多种对比算法，验证了所提算法可以实现智能体间的高效协作.

关键词 多智能体强化学习；合作型任务；近似可分解结构；动态图；协作

中图法分类号 TP18

许多现实生活问题可以被建模为多智能体任务并尝试使用多智能体强化学习技术解决. 尽管近期工作已经在诸多领域取得了显著进展，例如自动车驾驶[1]、交通控制[2-4]、机器人群体控制[5]、量化交易[6]和传感网络[7]等，但是对于复杂多智能体任务而言，利用多智能体强化学习算法为智能体学习有效的协作策略仍然充满挑战. 一个主要问题是多智能体强化学习算法学习效率低下，通常需要成百上千万次试错才能学习到最优策略[8]. 此外，在多智能体任务中，多智能体强化学习算法需要在智能体联合动作空间中搜索最优联合策略. 因为智能体联合动作空间随智能体数量呈指数级增长，所以这进一步加剧了多智能体强化学习算法的效率低下问题，导致其难以实现多智能体任务中智能体协作策略的高效学习.

提升多智能体强化学习算法学习效率的一种有效手段是建模多智能体任务中潜在的智能体间交互结构，并且利用该结构信息去指导智能体策略学习，从而提升算法的学习效率. 基于这一思路，本文假设多智能体任务存在近似可分解结构[9]，并旨在针对该类多智能体任务实现智能体间的高效协作. 具体而言，近似可分解结构假设相同交互集合内智能体间交互强度大，而不同交互集合内智能体间交互强度小，但不可忽视. 这一结构广泛存在于现实生活场景中，例如在公司员工管理中，相同部门内的员工间交互相比不同部门员工间交互更加频繁. 同样，在足球比赛中，距离相近的球员间交互强度相比距离较远的球员间交互强度更大. 显而易见，对于具有近似可分解结构的合作型多智能体任务，智能体间协作包括相同交互集合内部分智能体间的局部协作以及不同交互集合全部智能体间的全局协作. 基于任务的近似可分解结构来设计协作机制以有效地建模智能体间局部和全部协作，可以缓解多智能体强化学习算法在智能体联合动作空间搜索带来的效率低下问题，进而提升协作策略的学习效率.

设计协作机制有效建模智能体间局部和全局协作需要解决2个问题：1）如何识别智能体交互集合；2）如何实现智能体间局部和全局协作. 目前主要有三大类工作尝试解决该问题.

第1类工作主要包括值函数分解算法[10-13]和多智能体策略梯度算法[14-20]. 该类算法忽视智能体间局部协作，直接优化智能体联合策略来实现智能体间全局协作. 然而，直接在智能体联合动作空间上搜索最优联合策略通常会导致算法学习效率低下，影响智能体间高效协作.

第2类工作[21-25]通过将全局动作值函数分解为定义在局部智能体联合动作上的局部值函数，并允许智能体间通信来实现智能体间局部和全局协作. 然而，该类算法严重依赖于智能体间通信，导致其无法适用于通信受限甚至不可用的现实生活多智能体场景.

第3类工作利用邻域[26]、分组[27-30]、角色[31-32]和子任务[33-36]等概念来识别智能体交互集合，并且限制相同交互集合内智能体遵循相似约束来实现智能体间局部协作. 同时该类算法通过学习全局动作值函数来实现智能体间全局协作. 然而，该类算法仍存在2个问题：1）当局部协作约束与任务目标相悖时，可能会影响智能体间协作性能；2）当智能体协作行为缺乏导致任务回报稀疏时，仅根据任务回报学习全局动作值函数可能会导致智能体间全局协作效率低下.

为设计高效协作机制来有效建模智能体间局部和全局协作，本文提出智能体动态协作图来建模多智能体任务中的近似可分解结构，并刻画随时间变化的智能体间交互关系，如图1所示. 其中节点表示智能体，节点间的边表示智能体间交互关系. 当智能体处于相同交互集合中时，其间交互强度较大；而不同交互集合智能体间交互强度小. 基于该形式化，多智能体强化学习算法可以通过有效建模智能体动态协作图中智能体间交互集合，并着重利用相同交互集合内相关智能体信息辅助其内每个智能体策略学习，从而提升智能体间协作策略学习效率. 根据这一

观点, 本文提出协作子任务行为(coordinated subtask pattern, CSP)算法. 具体而言, CSP 算法利用子任务来识别智能体交互集合, 并采用双层策略结构来协调智能体间的动作选择. 其中, 上层策略将所有智能体周期性地分配到多个子任务中, 而下层策略基于智能体子任务分配为其输出原子动作. 该上层策略受益于较小的联合子任务空间以及时序抽象的任务回报, 因此可以实现有效的子任务分配, 从而准确刻画智能体动态协作图上智能体间的交互关系. 基于这种子任务分配, CSP 算法提出子任务内行为约束确保相同子任务内的部分智能体间可以预知彼此动作选择, 之后每个智能体基于其局部信息和相同子任务内其他智能体动作预测信息选择动作来最大化任务回报, 由此提升相同子任务内智能体间局部协作; 同时, 针对智能体间全局协作, CSP 算法将智能体上层联合策略相应的时间差分误差作为子任务间行为约束指导智能体协作策略学习. 该约束可以为智能体下层策略提供稠密的监督信息, 鼓励所有智能体选择优异的联合动作来最大化该约束和任务回报, 由此实现智能体间全局高效协作.

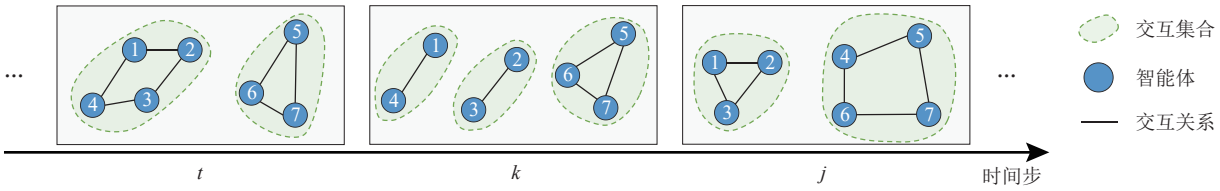


Fig. 1 Dynamic coordination graph of the nearly decomposable structure in multi-agent tasks

图 1 对于多智能体任务中近似可分解结构的动态协作图

本文在星际争霸[37]环境下对 CSP 算法进行评估. 实验结果表明该算法在多个地图上性能明显优于多个对比算法. 此外, 消融实验也验证了 CSP 算法所提子任务内和子任务间行为约束的有效性.

本文主要贡献包括 3 个方面:

1)利用智能体动态协作图来建模多智能体任务中的近似可分解结构, 并指出需要同时刻画智能体间局部和全局协作来实现协作策略的高效学习;

2)提出协作子任务行为算法, 该算法通过分配子任务来识别智能体间交互关系, 并提出子任务内和子任务间行为约束增强智能体局部和全局协作;

3)在星际争霸环境多个地图上的实验结果验证了本文所提算法的学习效率明显优于对比算法, 且消融实验验证了本文所提约束的有效性.

1 相关工作

在本节中, 我们根据目前多智能体强化学习算法实现智能体间协作的方式将其划分为三大类, 并简要介绍每一类工作的代表性算法.

大部分多智能体强化学习算法属于第 1 类工作, 该类算法通过直接学习智能体最优联合策略来实现智能体间全局协作. 其中值函数分解算法 VDN[10], QMIX[11], QTRAN[12] 和 QPLEX[13] 等通过学习一个可分解的全局动作值函数来更新所有智能体的去集中式效用函数(效用函数对应智能体的去集中式策略). 多智能体策略梯度算法例如 MADDPG[14], COMA[15], DOP[16], FACMAC[17], MAPPO[18], HAPPO[19] 以及 MADDPG-DC[20] 等利用集中式值函数指导智能体去集中式策略的学习. 尽管这一类算法可以克服多智能体任务中策略学习的非稳态问题, 但是它们并未显式利用多智能体任务中存在的近似可分解结构. 此外, 直接在智能体联合动作空间上搜索最优联合策略会导致策略学习效率低下, 影响智能体间高效协作.

第 2 类工作通过将全局动作值函数分解为定义在成对或局部智能体联合动作上的局部值函数来隐式地协调智能体局部和全局动作选择, 以增强智能体间局部和全局协作. 例如协同强化学习算法[21]通过学习一组定义在局部智能体联合动作上的局部值函数来近似全局动作值函数, 并利用通信传递技术来决定最优联合动作. 文献 [22] 将该算法扩展到网络化分布式部分可观测马尔可夫决策过程, 其利用分布式约束优化技术来协调多个预定义智能体交互集合的分布式学习. 随后, 文献 [23] 通过定义交互度量来动态识别智能体交互集合, 从而降低智能体间通信开销. 近期通信算法[24-25]在特定智能体间进行通信, 隐式地构建智能体间交互集合, 因此也属于这一类工作. 尽管这类算法利用通信实现智能体间有效协作, 但是当智能体间通信受限甚至不可用时, 该类算法难以生效.

第 3 类工作利用邻域、分组、角色和子任务等概念来识别智能体交互集合, 并限制相同交互集合内

智能体服从相似约束来实现智能体间局部协作. 具体而言, NCC[26] 预定义智能体邻域并约束相同邻域内智能体生成相似认知后验. GoMARL[27] 通过学习分组效用值函数对智能体进行动态分组, 同时鼓励智能体生成组内相似、组间差异的分组表征以及相应策略参数, 由此提升相同分组内智能体间局部协作. DCN[28] 首先利用图神经网络输出智能体观测表征, 随后利用 K-Means 聚类对智能体进行分组, 最后学习分组效用函数来刻画智能体间局部协作. GPLight[29] 利用图卷积网路编码智能体观测, 随后对齐具有相似轨迹的智能体相应观测表征, 由此实现相同组内智能体观测表征相似, 从而提升智能体间局部协作. AVGM[30] 将每个智能体可视范围内智能体集合视为其所属分组, 并构建分组效用函数协调组内智能体动作选择.

除邻域、分组外, 角色概念也被用于刻画智能体局部交互集合. 例如 ROMA[31] 为具有相似轨迹的智能体生成相似角色变量, 由此实现智能体交互集合划分. 同时, 该算法通过最大化相同集合内智能体彼此角色和轨迹间互信息来实现局部协作. 随后, RODE[32] 基于动作语义显式定义角色, 并限制智能体仅能执行其分配角色内所含动作. 除以上算法, 近期算法通过为智能体分配子任务来协调其动作选择. LDSA[33] 通过学习差异化子任务表征和定义在这些表征上的子任务策略来约束相同子任务内智能体行为相似. ALMA[34] 基于分层框架将所有智能体分配到预设子任务中, 并分别学习可分解的子任务值函数来实现各子任务内智能体协作. MACC[35] 将任务中实体视为子任务, 并为智能体学习子任务表征重构相应实体轨迹, 由此实现对相应子任务的隐式关注. CDS3[36] 基于智能体观测构建差异化子任务表征, 并设计面向智能体编号的行为多样性内在激励和部分共享的网络结构来提升探索, 由此实现子任务策略的高效学习.

尽管第 3 类工作通过邻域、分组、角色和子任务等概念来增强智能体间局部协作, 但是仍存在 3 个问题: 1) NCC, ALMA, MACC 等算法需要任务先验信息才能识别智能体交互集合; 2) NCC, GoMARL, GPLight, LDSA, ROMA, RODE 等算法约束相同交互集合内智能体认知或策略相似, 该局部约束可能与任务目标相悖并影响智能体协作性能; 3) 当智能体协作行为缺乏导致任务回报稀疏时, 该类算法仅根据任务回报学习全局动作值函数来实现智能体间全局协作这一做法可能会导致智能体全局协作效率低下. 相比之下, 本文算法预设子任务数量并基于双层策略结构实现智能体子任务自适应分配和识别, 无需任务先验信息. 同时, 相同子任务内智能体以彼此动作预测作为额外信息选择动作来最大化任务回报, 此时子任务内优化目标和任务目标一致, 因此保证智能体间局部协作能够提升任务整体性能. 此外, 针对任务回报稀疏问题, 本文算法基于上层策略设计稠密监督信息, 鼓励所有智能体选择优异的联合动作来最大化该约束和任务回报, 由此有效增强智能体间全局协作.

综上所述, 目前多智能体强化学习算法无法有效实现智能体间局部和全局协作. 为解决该问题, 本文提出子任务内和子任务间行为约束来增强智能体间局部和全局协作. 因此, 本文算法可以作为一种智能体间高效协作机制辅助多智能体强化学习算法, 由此提升智能体协作策略的学习效率.

2 背景知识

在本节中, 我们形式化本文拟解决的多智能体任务, 并对值函数分解算法进行简要概述.

2.1 去集中式部分可观测马尔可夫决策过程

对于要求智能体进行去集中式决策的合作型多智能体任务, 通常将其建模为去集中式部分可观测马尔可夫决策过程(decentralized partially observable Markov decision process, Dec-POMDP). Dec-POMDP 可以用多元组$(N,S,\boldsymbol{A},R,P,\gamma,Z,O)$表示, 其中$N=\{1,2,\cdots,n\}$表示智能体集合, S 表示状态空间. $\boldsymbol{A}=A^1\times A^2\times\cdots\times A^n$ 代表智能体联合动作空间, 其中 A^i 代表智能体 $i\in N$ 的局部动作空间. 在每一时间步 t, 每个智能体 $i\in N$ 根据其局部观测函数 $O^i(o_t^i|s_t)\in O$ 获取其局部观测 $o_t^i\in Z^i\in Z$, 随后根据其局部策略 π^i 选择动作 a_t^i. 基于所有智能体的联合动作 $\boldsymbol{a_t}=(a_t^1,a_t^2,...,a_t^n)$, 多智能体任务环境 s_t 根据状态转移函数 $P(s_{t+1}|s_t,\boldsymbol{a_t})$ 转移到下一状态 s_{t+1}, 同时根据回报函数 $R(s_t,\boldsymbol{a_t})$ 反馈给所有智能体共享回报 r_t. 为应对部分可观测问题, 目前算法利用智能体动作-观测历史 $\tau_t^i=(o_1^i,a_1^i,o_2^i,a_2^i,...,o_t^i)$ 进行动作选择, 即智能体学习局部策略 $\pi^i(a_t^i|\tau_t^i)$. 在 Dec-POMDP 中, 所有智能体的目标为学习最优联合策略 $\boldsymbol{\pi}=(\pi^1,\pi^2,...,\pi^n)$, 该联合策略可以最大化期望累计折扣回报 $\mathbb{E}_{\boldsymbol{\pi},P}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$, 其中 γ 为折扣因子.

2.2 值函数分解算法

本文基于的值函数分解算法为智能体学习去集中式策略. 值函数分解算法如 VDN、QMIX、QTRAN

等将全局动作值函数Q^{total}分解为每个智能体的去集中式效用函数$Q^i(\tau^i,a^i)$(效用函数定义在智能体局部动作-观测历史τ^i和局部动作a^i上,对应每个智能体的去集中式策略π^i),由此可以通过Q^{total}来更新Q^i. 为保证Q^{total}和Q^i最优动作的一致性,其两者间分解关系需要满足局部-全局最大化(individual-global max, IGM)原则. IGM 原则定义为:

$$\arg\max_{\boldsymbol{a}} Q^{\text{total}}(\boldsymbol{\tau},\boldsymbol{a})=\begin{pmatrix}\arg\max\limits_{a^1} Q^1(\tau^1,a^1)\\ \arg\max\limits_{a^2} Q^2(\tau^2,a^2)\\ \vdots\\ \arg\max\limits_{a^n} Q^n(\tau^n,a^n)\end{pmatrix}, \tag{1}$$

其中$\boldsymbol{\tau}$代表智能体联合动作-观测历史. 基于 IGM 原则,Q^{total}对应智能体联合动作空间上的贪婪联合动作,其可以通过对每个智能体Q^i进行局部贪婪操作得到.

以 QMIX 算法为例,该算法通过提出单调性约束来遵循 IGM 原则. 其单调性约束定义为:

$$\frac{\partial Q^{\text{total}}(\boldsymbol{\tau},\boldsymbol{a},s)}{\partial Q^i(\tau^i,a^i)}\geqslant 0,\ \forall i\in N. \tag{2}$$

并通过学习一个混合网络f来计算Q^{total},计算公式定义为:

$$Q^{\text{total}}(\boldsymbol{\tau},\boldsymbol{a},s)=f\left(Q^1(\tau^1,a^1),Q^2(\tau^2,a^2),\cdots,Q^n(\tau^n,a^n),s\right). \tag{3}$$

QMIX 算法以状态s作为超参网络输入来生成混合网络f的参数,并且限制混合网络f的权值非负以满足单调性约束. 在计算得到Q^{total}后,所有智能体的去集中式效用函数Q^i即可利用Q^{total}相应的时间差分损失(temporal difference loss, TD-loss)进行更新.

3 协作子任务行为算法

本节详细介绍 CSP 算法. 我们首先使用智能体动态协作图建模多智能体任务中的近似可分解结构,并利用双层策略结构进行子任务分配. 之后提出子任务内和子任务间行为约束来增强智能体间局部和全局协作. 最后总结算法整体学习目标及流程.

3.1 图多智能体任务建模

本文提出智能体动态协作图建模多智能体任务中的近似可分解结构,其中节点代表智能体,节点间的边表示智能体间的交互关系. 如图 2 所示,对于一个 6 智能体任务,在某一时间步时,相同交互集合内智能体间交互强度大,而不同交互集合智能体间交互强度小,且智能体间交互关系随时间发生变化. 智能体动态协作图具体定义为:

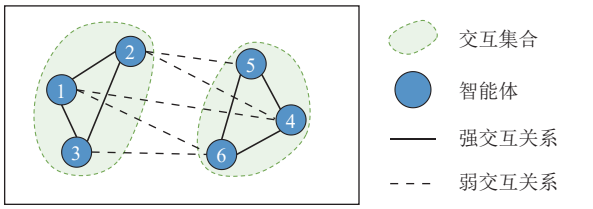


Fig. 2 Agent dynamic coordination graph of a 6-agent task at certain timestep

图 2 一个 6 智能体任务在某一时间步的智能体动态协作图

定义 1. 智能体动态协作图. 给定一个无向图$G=(N,\boldsymbol{E},T)$,其中N代表节点集合,$E_t\in\boldsymbol{E}$代表时间步$t\in T$时的边集合. 在节点集合N中,每个节点$i\in N$代表智能体实体. 对于任意 2 个智能体节点$i,j\in N$,如果在时间步t时 2 节点间存在边$e^{ij}\in E_t$,则表示相应智能体i和j在当前时间步t存在交互关系,且边的权重代表智能体间的交互强度.

建模智能体动态协作图可以准确描述多智能体任务中智能体间的实时交互关系,并由此识别智能体交互集合. 此时,每一智能体在更新策略时需要更多考虑相同交互集合内其他强交互智能体信息,而较少考虑不同交互集合内弱交互智能体信息,由此提升智能体间协作策略的学习效率. 文献 [38] 所提的算法通过遍历所有成对智能体并考虑每一智能体对另一智能体局部状态转移和累积折扣回报的影响来衡量智能体间交互强度. 文献 [23] 所提的算法通过定义每个智能体在缺失另一个智能体子集协作时的效用损失来衡量其间交互强度. 然而,当多智能体任务中智能体数量过多时,该类算法遍历所有可能的智能体组合会导致复杂度较大,阻碍智能体协作策略高效学习.

为实现智能体动态协作图的高效建模,CSP 算法使用子任务来识别智能体交互集合,并利用双层策略中的上层策略将智能体周期性地分配到预设的子任务中. 其中去集中式上层策略可以实现智能体去集中式子任务分配,由此克服之前算法需要遍历所有可能的智能体组合带来的可扩展性差的问题. 此外,上层策略可以利用时序抽象的任务回报实现子任务高效分配,从而最大化任务回报. 因此,CSP 算法可以实现复杂多智能体任务中智能体交互集合的高效识别,由此近似刻画智能体动态协作图中智能体间交互关系.

3.2 双层策略结构

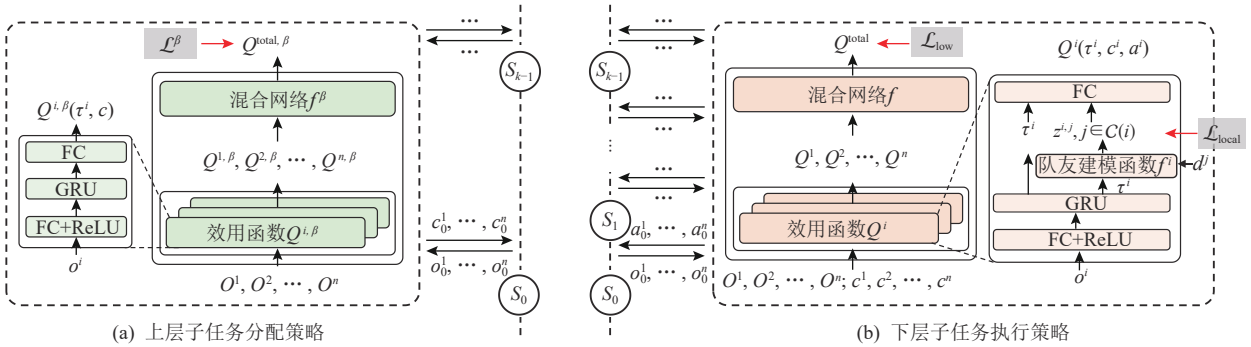
CSP 算法为每个智能体i学习 2 部分策略: 1)子

任务分配策略$\beta^i:\mathcal{T}^i\to C^i$：基于智能体$i$局部动作-观测历史$\tau^i\in\mathcal{T}^i$选择子任务$c^i\in C^i$；2）子任务执行策略$\pi^i:\mathcal{T}^i,C^i\to A^i$：基于智能体$i$局部动作-观测历史$\tau^i\in\mathcal{T}^i$和所属子任务$c^i\in C^i$输出原子动作$a^i\in A^i$.

对于每个智能体i, CSP算法利用双层策略结构中的上层策略实现其子任务分配策略$\beta^i:\mathcal{T}^i\to C^i$, 而子任务执行策略$\pi^i:\mathcal{T}^i,C^i\to A^i$使用下层策略来实现. 当在多智能体任务中进行交互时, 上层子任务分配策略β^i每隔k个时间步将智能体i分配到某一子任务$c^i\in C^i$中；之后, 智能体i遵循该子任务并利用下层子任务执行策略π^i选择并输出原子动作$a^i\in A^i$.

3.2.1 子任务分配策略

为解决子任务空间未知问题, CSP算法引入参数m来指定子任务数量, 并使用独热编码(one-hot encoding)表示子任务. 如图3所示, 本文通过学习一个子任务效用函数$Q^{i,\beta}(\tau^i,c)$来实现每个智能体i的子任务分配策略β^i. $Q^{i,\beta}(\tau^i,c)$以智能体i局部动作-观测历史作为输入并输出所有可能子任务$c\in C^i$相应的Q值. 每隔k个时间步, 智能体i基于这些Q值选择子任务c^i, 并将其发送给下层子任务执行策略.



(a) 上层子任务分配策略 (b) 下层子任务执行策略

Fig. 3 Architecture of CSP algorithm

图3 CSP算法结构

同时, 为协调所有智能体的子任务分配策略, CSP算法引入一个混合网络f^β, 该混合网络根据所有智能体的子任务效用函数$Q^{i,\beta}$来计算全局子任务值函数$Q^{\text{total},\beta}$, 进而$Q^{i,\beta}$可以通过$Q^{\text{total},\beta}$利用任务回报进行更新, 其损失函数定义为：

$$\mathcal{L}^\beta(\theta^\beta,\omega^\beta)=\mathbb{E}_{\mathcal{D}}\left[\left(\sum_{t'=0}^{k-1}r_{t+t'}+\gamma\max_{\boldsymbol{c}'}\bar{Q}^{\text{total},\beta}(\boldsymbol{\tau}_{t+k},\boldsymbol{c}')-Q^{\text{total},\beta}(\boldsymbol{\tau}_t,\boldsymbol{c}_t)\right)^2\right],\tag{4}$$

其中$\boldsymbol{c}_t=(c_t^1,c_t^2,\cdots,c_t^n)$表示时间步$t$所有智能体的联合子任务分配, 而$\boldsymbol{c}'$表示下一时间步$t+k$所有智能体的联合子任务分配. 相比下层策略, 上层策略遵循更快的时间尺度, 即每隔k个时间步执行1次子任务选择. θ^β和ω^β分别代表$Q^{i,\beta}$和混合网络f^β的参数, 并且所有智能体共享$Q^{i,\beta}$网络参数. $\bar{Q}^{\text{total},\beta}$表示一个周期性更新的目标全局子任务值函数. 此外, 本文通过从经验回放池$\mathcal{D}$中采样数据来近似计算期望.

3.2.2 子任务执行策略

在接收到上层子任务分配策略输出的子任务c^i后, 智能体i在之后的k个时间步内遵循该子任务并利用下层子任务执行策略选择原子动作. 如图3(b)所示, CSP算法通过学习效用函数$Q^i(\tau^i,c^i,a^i)$来实现智能体i的子任务执行策略π^i. $Q^i(\tau^i,c^i,a^i)$以智能体i局部动作-观测历史τ^i和所属子任务c^i为输入, 输出所有可能原子动作$a^i\in A^i$相应Q值并由此选择动作.

3.3 行为约束

仅将子任务c^i作为每个智能体i子任务执行策略额外输入无法保证智能体间协作行为增强. 为解决该问题, CSP算法提出子任务内和子任务间行为约束来增强智能体间局部和全局协作.

3.3.1 子任务内行为约束

对于每个智能体i, 本文使用$C(i)$表示其所处子任务c^i的智能体集合. 为增强$C(i)$内智能体间局部协作, CSP算法对$C(i)$内其余智能体$j\in C(i),j\neq i$进行行为建模, 并利用该行为信息辅助智能体i的子任务执行策略进行动作选择. 具体而言, CSP算法为每个智能体i学习一个队友建模函数$f^i(\tau^i,d^j)$, 其以智能体i动作-观测历史τ^i和待建模队友编号d^j为输入, 输出一个多元高斯分布的均值$\mu^{i,j}$和方差$\sigma^{i,j}$. 随后, CSP算法从相应高斯分布$\mathcal{N}(\mu^{i,j};\sigma^{i,j})$中采样变量$z^{i,j}$作为智能体$i$对$C(i)$内其他智能体$j$的行为表征.

为保证该变量$z^{i,j}$能够准确刻画智能体j的策略信息, CSP算法最大化变量$z^{i,j}$和智能体j的动作a^j间互信息, 该优化目标定义为：

$$I(z^{i,j};a^j|\tau^i,d^j)=H(z^{i,j}|\tau^i,d^j)-H(z^{i,j}|\tau^i,d^j,a^j),\quad(5)$$

其中$I(\cdot;\cdot)$代表互信息, $H(\cdot)$代表信息熵. 直观来说, 如果$z^{i,j}$与a^j相关性小, 则a^j无法提供$z^{i,j}$相关信息, 此时$H(z^{i,j}|\tau^i,d^j,a^j)$衰退为$H(z^{i,j}|\tau^i,d^j)$, 相应地, $z^{i,j}$和a^j间互信息$I(z^{i,j};a^j|\tau^i,d^j)$趋近于0. 相反, 若两者间关联性大, $H(z^{i,j}|\tau^i,d^j,a^j)<H(z^{i,j}|\tau^i,d^j)$, 导致$I(z^{i,j};a^j|\tau^i,d^j)$变大. 因此, 通过最大化互信息$I(z^{i,j};a^j|\tau^i,d^j)$可以增强变量$z^{i,j}$与智能体$j$的动作$a^j$间的相关性, 实现$z^{i,j}$对$a^j$的准确建模.

针对互信息项$I(z^{i,j};a^j|\tau^i,d^j)$, 本文将其重写为:

$$I(z^{i,j};a^j|\tau^i,d^j)=\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log\frac{p(z^{i,j};a^j|\tau^i,d^j)}{p(z^{i,j}|\tau^i,d^j)p(a^j|\tau^i,d^j)}\right]=\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log\frac{p(z^{i,j}|a^j,\tau^i,d^j)}{p(z^{i,j}|\tau^i,d^j)}\right],\quad(6)$$

其中$p(z^{i,j}|a^j,\tau^i,d^j)$为未知条件概率分布. 本文引入变分概率分布$q^\phi(z^{i,j}|a^j,\tau^i,d^j)$对其近似:

$$\begin{aligned}I(z^{i,j};a^j|\tau^i,d^j)=&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log\frac{q^\phi(z^{i,j}|a^j,\tau^i,d^j)}{p(z^{i,j}|\tau^i,d^j)}\right]+\\&D_{\mathrm{KL}}\left[p(z^{i,j}|a^j,\tau^i,d^j)\|q^\phi(z^{i,j}|a^j,\tau^i,d^j)\right]\geqslant\\&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log\frac{q^\phi(z^{i,j}|a^j,\tau^i,d^j)}{p(z^{i,j}|\tau^i,d^j)}\right]=\\&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log q^\phi(z^{i,j}|a^j,\tau^i,d^j)\right]-\\&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}\left[\log p(z^{i,j}|\tau^i,d^j)\right].\end{aligned}\quad(7)$$

由于队友建模函数$f^i(\tau^i,d^j)$以τ^i和d^j作为输入来输出$z^{i,j}$, 因此当给定τ^i,d^j时$z^{i,j}$与a^j相独立, 此时:

$$\begin{aligned}I(z^{i,j};a^j|\tau^i,d^j)=&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}[\log q^\phi(z^{i,j}|a^j,\tau^i,d^j)]-\\&\mathbb{E}_{z^{i,j},a^j,\tau^i,d^j}[\log p(z^{i,j}|\tau^i,d^j)]=\\&\mathbb{E}_{a^j,\tau^i,d^j}\left[\int(p(z^{i,j}|\tau^i,d^j)\log q^\phi(z^{i,j}|a^j,\tau^i,d^j)-\right.\\&\left.p(z^{i,j}|\tau^i,d^j)\log p(z^{i,j}|\tau^i,d^j))\mathrm{d}z^{i,j}\right]=\\&\mathbb{E}_{a^j,\tau^i,d^j}\left[-D_{\mathrm{KL}}\left[p(z^{i,j}|\tau^i,d^j)\|q^\phi(z^{i,j}|a^j,\tau^i,d^j)\right]\right].\end{aligned}\quad(8)$$

因此, 本文定义损失函数作为子任务内行为约束:

$$\mathcal{L}_{\mathrm{local}}(\theta^{\mathrm{team}},\phi)=\sum_{i\in N}\sum_{\substack{j\in C(i),\\ j\neq i}}\mathbb{E}_{\mathcal{D}}\left[D_{\mathrm{KL}}\left[p(z^{i,j}|\tau^i,d^j)\|q^\phi(z^{i,j}|a^j,\tau^i,d^j)\right]\right],\quad(9)$$

其中θ^{team}表示队友建模函数f^i的参数, ϕ表示变分概率分布q^ϕ的参数. 通过优化该损失, 每个智能体i可以通过变量$z^{i,j}$实现对相同子任务集合$C(i)$内其他智能体行为的准确建模. 随后对于每个智能体i, CSP算法合并其局部动作-观测历史τ^i和相同子任务中其他智能体j相应$z^{i,j}$来计算$Q^i(\tau^i,c^i,a^i)$. 通过行为建模, 相同子任务内智能体间可以准确预知彼此动作选择, 由此实现相同子任务内智能体间局部协作增强.

3.3.2 子任务间行为约束

对于子任务间智能体全局协作, 之前算法根据任务回报学习全局动作值函数来实现该目标. 然而, 当学习过程中智能体协作行为缺乏导致任务回报稀疏时, 全局动作值函数学习效率低下, 影响智能体间全局协作. 为解决该问题, CSP算法提出子任务间行为约束辅助全局动作值函数高效学习. 具体而言, CSP算法利用上层子任务分配策略每个决策间隔相应的时间差分误差(temporal difference error, TD-error)为下层子任务执行策略生成内在回报, 其定义为:

$$r_h^{\mathrm{intrinsic}}=\frac{1}{k}\left(\delta^\beta_{t:t+k-1}\right),\ \forall h\in\{t,t+1,t+2,\cdots,t+k-1\},\quad(10)$$

其中$\delta^\beta_{t:t+k-1}$表示上层智能体联合子任务分配策略在其单个决策间隔$(t,t+k-1)$对应的TD-error, 定义为:

$$\delta^\beta_{t:t+k-1}=\sum_{t'=0}^{k-1}r_{t+t'}+\gamma\max_{\boldsymbol{c}'}\bar{Q}^{\mathrm{total},\beta}(\boldsymbol{\tau}_{t+k},\boldsymbol{c}')-Q^{\mathrm{total},\beta}(\boldsymbol{\tau}_t,\boldsymbol{c}_t),\quad(11)$$

其中各元素定义与式(4)相同. 根据式(10), 对该决策间隔内的每个时间步$h\in\{t,t+k-1\}$, CSP算法将$\delta^\beta_{t:t+k-1}$按k等分, 由此为下层智能体联合子任务执行策略生成每一个时间步的内在回报$r_h^{\mathrm{intrinsic}}$.

在CSP算法的双层策略结构中, 相比下层策略, 上层策略受益于较小的联合子任务空间以及时序抽象回报, 因此上层策略可以在更大的决策尺度上识别下层策略学习进度. 当$\delta^\beta_{t:t+k-1}\geqslant 0$时, 其表示智能体下层联合策略在该时间间隔内执行了优异的联合原子动作并收到了较大任务回报, 进而导致上层策略值函数的大幅度更新. 因此, 本文利用上层策略的TD-error识别优异的下层联合原子动作并将其作为内在回报指导下层策略学习, 有助于提升下层联合策略的学习效率, 实现子任务间智能体全局协作增强.

3.4 总体学习目标

对于上层子任务分配策略, CSP算法根据式(4)对其进行更新. 对于下层子任务执行策略, CSP算法利用任务回报r_t和内在回报$r_t^{\mathrm{intrinsic}}$对其进行更新, 其相应损失函数定义为:

$$\mathcal{L}_{\mathrm{low}}(\theta,\omega)=\mathbb{E}_{\mathcal{D}}\left[\left(r_t+\alpha\times r_t^{\mathrm{intrinsic}}+\gamma\max_{\boldsymbol{a}_{t+1}}\bar{Q}^{\mathrm{total}}(\boldsymbol{\tau}_{t+1},\boldsymbol{a}_{t+1})-Q^{\mathrm{total}}(\boldsymbol{\tau}_t,\boldsymbol{a})\right)^2\right],\quad(12)$$

其中α是一权重系数, θ表示下层子任务执行策略相应效用函数$Q^i(\tau^i,c^i,a^i)$的参数(智能体间共享该效用函数参数). 对于下层策略, CSP算法也利用一个混合网络f(参数为ω)来组合所有智能体$Q^i(\tau^i,c^i,a^i)$得到全局动作值函数$Q^{\mathrm{total}}(\boldsymbol{\tau},\boldsymbol{a})$. $\boldsymbol{a}_{t+1}$表示下一时间步

$t+1$所有智能体的联合原子动作，$\boldsymbol{a}_t$表示当前时间步t所有智能体的联合原子动作. $\bar{Q}^{\text{total}}$基于Q^{total}参数进行周期性更新. 基于经验回放池$\mathcal{D}$中样本，CSP 算法利用式(12)实现对Q^i以及f的更新.

总而言之，CSP 算法的损失函数定义为：

$$\mathcal{L}=\mathcal{L}_{\text{low}}(\theta,\omega)+\mathcal{L}^{\beta}(\theta^{\beta},\omega^{\beta})+\lambda\mathcal{L}_{\text{local}}(\theta^{\text{team}},\phi), \tag{13}$$

其中λ表示权重系数. 通过优化式(13)损失函数，CSP 算法实现相同子任务内和不同子任务间智能体局部和全局协作增强，从而提升智能体策略学习效率.

CSP 算法运行流程如算法 1 所示. CSP 算法首先需要初始化参数设置以及各模块网络参数，然后行①~②表示每一智能体利用双层策略与多智能体任务环境交互采集数据并存放到经验回放池$\mathcal{D}$中，其中上层策略每隔k个时间步为智能体i分配子任务c^i，下层策略接收c^i作为额外输入选择原子动作a_t^i输出. 行②~⑤表示当训练回合开始时，CSP 算法从经验回放池$\mathcal{D}$随机采样批训练数据$\mathcal{B}$，随后基于$\mathcal{B}$利用式(13)计算损失函数$\mathcal{L}$，最终利用梯度下降方法对相关模块参数进行更新. 以上流程重复进行，直到算法运行回合数满足预设最大回合数终止.

算法 1. CSP 算法.

输入：初始化算法参数设置、各模块参数；

输出：所有智能体去集中式策略.

① while 未完成最大训练回合数时

② for 每个回合中的每一个时间步t

③ for 每一智能体i

④ If $t\%\ k=0$

⑤ 利用子任务分配策略β^i选择c^i；

⑥ 利用子任务执行策略π^i选择a_t^i；

⑦ end if

⑧ end for

⑨ end for

⑩ 采集数据存入$\mathcal{D}$中；

⑪ If 训练回合

⑫ 从$\mathcal{D}$中随机采样批训练数据$\mathcal{B}$；

⑬ 根据式(13)计算损失函数$\mathcal{L}$；

⑭ 利用梯度下降更新相关模块参数；

⑮ end if

⑯ end while

4 实　　验

在本节中，我们设计实验来回答 2 个问题：1)CSP 算法是否可以通过增强智能体间局部和全局协作来提升智能体协作策略学习效率(见 4.2 节)；2)如果 CSP 算法能够实现协作策略的高效学习，那么哪个模块对其性能提升贡献较大？(见 4.3 节). 此外，本节对实验中所采用的测试任务、对比算法以及算法参数设置给予简要概述(见 4.1 节和 4.4 节).

4.1 测试任务和对比算法

为测试算法有效性，本文在星际争霸[37]环境(SMAC)的多个地图上将 CSP 算法与多个基准算法进行比较. 测试任务场景和基准算法的简要描述如下.

4.1.1 测试任务

在星际争霸环境中，本文选择 6 个作战地图 3s5z、1c3s5z、3s_vs_5z、5m_vs_6m、2c_vs_64zg 和 MMM2 作为测试任务. 这些任务包含 2 方单元，其中友方单元由本文算法控制，而敌方单元由内置规则控制. 友方单元需要协作击杀敌方单元来获胜. 这些地图根据友方和敌方单元配置被划分为容易、困难和超级困难 3 种难度，本文在表 1 中对这些地图进行概述.

Table 1 Descriptions of Selected Maps in SMAC

表 1 星际争霸环境下选用地图的描述

地图信息	难度	作战单元配置
3s5z	简单	友方：3 Stalkers, 5 Zealots 敌方：3 Stalkers, 5 Zealots
1c3s5z	简单	友方：1 Colossus, 3 Stalkers, 5 Zealots 敌方：1 Colossus, 3 Stalkers, 5 Zealots
3s_vs_5z	困难	友方：3 Stalkers 敌方：5 Zealots
5m_vs_6m	困难	友方：5 Marines 敌方：6 Marines
2c_vs_64zg	困难	友方：2 Colossi 敌方：64 Zerglings
MMM2	超级困难	友方：1 Medivac, 2 Marauders, 7 Marines 敌方：1 Medivac, 3 Marauders, 8 Marines

4.1.2 基准算法

为回答问题 1，本文将以下 6 个算法作为基准算法.

1)GoMARL[27]. 该算法通过学习分组效用值函数对智能体进行动态分组，同时鼓励智能体生成组内相似、组间差异的分组表征以及相应策略参数，由此增强相同分组内智能体间的局部协作.

2)LDSA[33]. 该算法学习差异化子任务向量表征，并将子任务策略定义在其相应表征上，由此实现相同子任务内智能体间局部协作增强.

3)ROMA[31]. 该算法基于智能体局部动作-观测历史轨迹生成角色，并为行为相似的智能体生成相似角色变量来实现智能体间局部协作增强.

4)RODE_ND. RODE[32]基于动作语义划分角色，并利用双层策略结构实现角色分配. 然而，该算法需

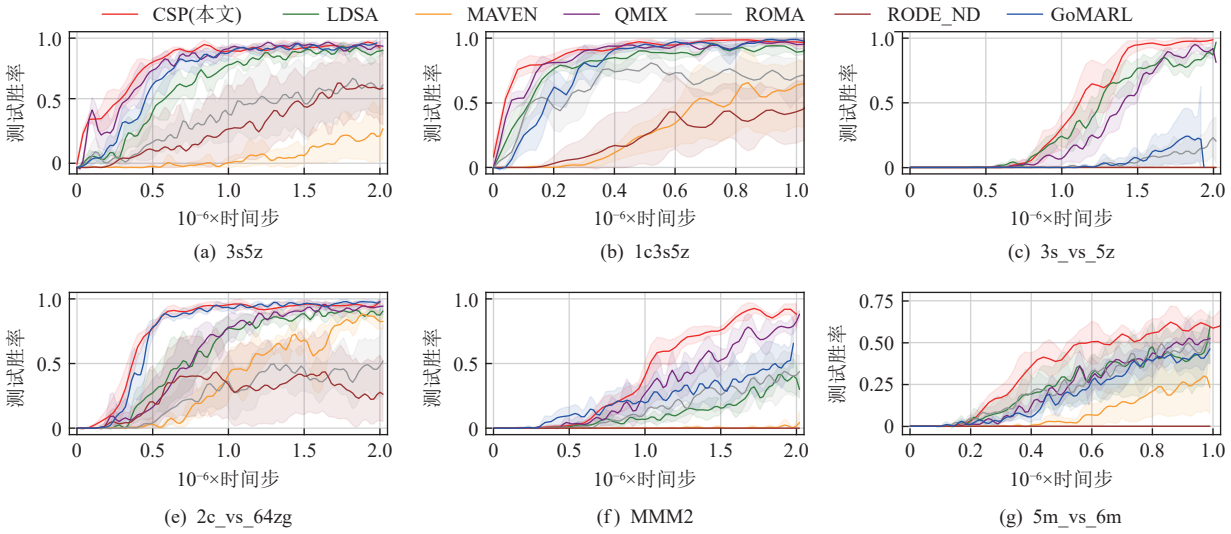
要利用任务先验知识来实现角色有效划分(例如在星际争霸中，RODE 强制将 6 个移动相关动作包含在所有角色中)．本文通过移除该操作来验证其通用性．

5)MAVEN[39]．该算法在回合开始时生成一个全局变量并约束所有智能体生成相应联合轨迹，由此实现智能体联合行为多样化．该算法忽视智能体间局部协作，直接在智能体联合动作空间开展全局协作，本文将其作为基准算法来验证刻画局部协作的必要性．

6)QMIX[11]．该算法通过学习可分解的全局动作值函数来更新智能体去集中式效用函数(策略)，并未显式刻画智能体间局部协作结构．因此本文也将其作为基准算法来验证刻画智能体间局部协作的必要性．

4.2 对比实验结果

CSP 算法与以上基准算法的对比实验结果如图 4 所示．具体而言，CSP 算法在地图 2c_vs_64zg，MMM2 以及 5m_vs_6m 上明显优于其他基准算法，并且在其他地图 3s5z、1c3s5z 和 3s_vs_5z 上也表现出优异性能．相比之下，LDSA 算法尽管可以在地图 3s5z、1c3s5z 和 3s_vs_5z 上学习到有效策略击败敌方单元并实现较高的测试胜率，但是其性能仍弱于 CSP 算法．



注：图中实线代表 5 个随机种子设定下所有算法的平均测试胜率，而阴影区域代表标准差．

Fig. 4 Comparison results of our algorithm against multiple baselines on six maps in SMAC

图 4 星际争霸环境 6 个地图上本文算法与多个基准算法的对比结果

因为 LDSA 算法仅根据任务回报学习全局动作值函数来保证智能体间全局协作，所以这一结果验证了有效增强智能体间全局协作有利于提升智能体协作策略学习效率．在困难地图 2c_vs_64zg、5m_vs_6m 和超级困难地图 MMM2 下，地图难度大导致联合协作动作较难出现，因此任务回报稀疏．此时 LDSA 算法仅利用全局动作值函数保证智能体间全局协作的做法无法令其快速学习到有效的协作策略．这一结果进一步验证了高效增强智能体间全局协作的必要性．

QMIX 算法通过学习可分解的全局动作值函数来直接实现智能体间的全局协作，并且在地图 3s5z、1c3s5z 和 3s_vs_5z 上表现出优异性能．这可能归因于 QMIX 算法提出的混合网络能够实现隐式的信度分配，由此隐式刻画智能体交互集合并实现智能体间局部协作增强．然而，QMIX 算法对于智能体局部协作的隐式刻画严重依赖于任务回报，当任务难度较大导致任务回报稀疏时，其难以生效．QMIX 算法在地图 2c_vs_64zg、MMM2 以及 5m_vs_6m 下的实验学习效率低下验证了这一观点．

GoMARL 算法通过学习分组效用值函数对智能体进行分组，同时鼓励智能体生成组内相似、组间差异的分组信息和策略参数以增强相同分组内智能体间局部协作．然而，该算法仍然依赖于任务回报学习全局动作值函数以实现智能体间全局协作．当任务回报稠密时，该算法可以高效学习分组效用值函数和全局动作值函数，由此提升智能体协作策略学习效率，因此在地图 3s5z，1c3s5z 下可以快速学习到有效策略并实现较高胜率．然而，当地图难度大导致任务回报稀疏时，这些值函数无法有效更新，导致智能体间局部和全局协作效率低下，因此 GoMARL 算法在地图 3s_vs_5z、5m_vs_6m 和 MMM2 上表现较差．

此外，GoMARL 算法在地图 2c_vs_64zg 上优于 LDSA 算法与 QMIX 算法，这可能是因为该算法通过学习分组效用值函数能够为智能体实现更优的信度分配，由此提升智能体协作策略学习效率.

MAVEN 和 ROMA 算法在所有地图上都无法实现策略的高效学习. 这可能是因为 MAVEN 中提出的全局行为多样化约束和 ROMA 中提出的局部协作约束与原始任务目标冲突，导致智能体策略优化偏移任务目标，无法实现智能体间高效协作. RODE_ND 算法同样无法学习到有效策略实现较高测试胜率，这一结果表明 RODE 算法严重依赖于任务先验知识，当任务先验知识不可用时，其无法实现有效的角色划分.

4.3 消融实验结果

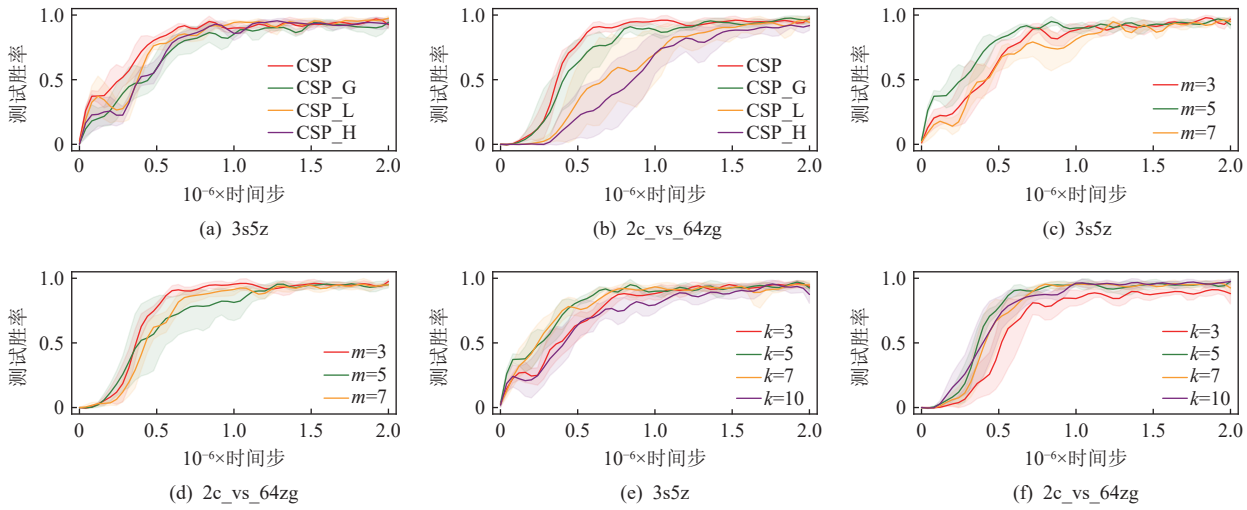
针对问题 2，本节开展消融实验来评估 CSP 算法中各模块对其性能的影响，这些模块包括：子任务内行为约束、子任务间行为约束、双层策略结构、预设子任务数量和子任务分配间隔. 对于前 3 个模块，本文引入以下基准算法对其影响进行评估：

1）CSP_L. 该基准算法通过移除子任务间行为约束（即设置 $\alpha=0$ 将 $r_t^{\text{intrinsic}}$ 从式（12）中去除）来评估子任务间行为约束对于 CSP 算法性能的影响.

2）CSP_G. 该基准算法通过去除子任务内行为约束 $\mathcal{L}_{\text{local}}$ 来评估其对 CSP 算法性能的影响.

3）CSP_H. 该基准算法同时去除 $\mathcal{L}_{\text{local}}$ 和 $r_t^{\text{intrinsic}}$ 来评估 CSP 算法性能提升是否归因于双层策略结构.

本文在地图 3s5z 和 2c_vs_64zg 上将 CSP 算法与以上基准算法进行比较，实验结果如图 5（a）（b）所示. 在地图 3s5z 上，CSP_G 算法性能明显弱于 CSP 算法，这一结果表明子任务内行为约束可以通过建模其他智能体行为来提升子任务内智能体间协作，移除该约束会导致协作策略学习效率低下. CSP_L 算法性能介于 CSP 和 CSP_G 之间，表明通过子任务间行为约束增强智能体间全局协作可以有效提升协作策略学习效率. 值得注意的是 CSP_G 和 CSP_H 性能相近，这可能是因为 3s5z 难度较低导致任务回报稠密，此时全局动作值函数可以被高效学习来增强智能体间全局协作，因此子任务间的行为约束效果并不显著.



注：图中实线代表 5 个随机种子设定下所有算法的平均测试胜率，而阴影区域代表标准差.

Fig. 5 Ablation studies regarding major modules of CSP algorithm on maps of 3s5z and 2c_vs_64zg

图 5 地图 3s5z 和 2c_vs_64zg 下 CSP 算法主要模块的消融实验

在地图 2c_vs_64zg 下，CSP_G 算法性能略弱于 CSP 算法，表明当智能体数量较少时，利用子任务内行为约束实现智能体行为建模作用较小. CSP_L 算法性能相较 CSP 算法明显下降，表明子任务间行为约束对算法实现协作策略高效学习具有重要影响. 相比之下，CSP_H 算法效果最弱，表明 CSP 算法性能提升主要归因于所提出的子任务内和子任务间的行为约束，而非双层策略结构.

对于预设子任务数量和子任务分配间隔模块，本文将 CSP 算法中的子任务数量参数 m 设置为 3, 5, 7，子任务分配间隔参数 k 设置为 3, 5, 7, 10，其他模块保持不变. 在这些参数设置下，本文在地图 3s5z 和 2c_vs_64zg 下运行 CSP 算法. 如图 5（c）~（f）所示，当地图中智能体数量较多时，CSP 算法在 $m=5$ 时效果较优；当地图中智能体数量较少时，CSP 算法在 $m=3$ 时效果较优. 此外，CSP 算法在 $k=5$ 时效果稳定且较优. 因此，本文在所有地图下默认设置 $m\leqslant 5$ 以及 $k=5$，如表 2 所示.

Table 2 Hyper-Parameter Settings for CSP Algorithm on All Maps

表 2 所有地图下 CSP 算法超参数设置

地图	m	k	α	λ
3s5z	5	5	0.001	1.0
1c3s5z	3	5	0.001	1.0
2c_vs_64zg	3	5	0.001	1.0
3s_vs_5z	3	5	0.001	1.0
5m_vs_6m	3	5	0.001	1.0
MMM2	3	5	0.001	0.1

4.4 算法参数设置

对于基准算法,本文利用其论文中的开源代码来开展实验,其中算法参数已经在星际争霸环境下微调. 本文基于 PyMARL[37] 框架实现所有算法以保证算法间的公平对比. 对于 CSP 算法,实验所用服务器包括一台配置为 2×22-core Intel® Xeon® Gold 6238 CPUs、512 GB RAM, 8 NVIDIA GTX2080Ti GPUs 的服务器和一台配置 2×12-core Intel® Xeon® CPU E5-2650 v4 CPUs、256 GB RAM、8 NVIDIA GTX1080Ti GPUs 的服务器.

5 结 论

本文使用动态图来建模多智能体任务中广泛存在的近似可分解结构并提出协作子任务行为算法来实现智能体间高效协作. 该算法利用子任务来识别智能体交互集合,并提出子任务内和子任务间行为约束来实现智能体间局部和全局协作增强. 星际争霸下的对比实验验证了本文算法能够有效提升智能体协作策略学习效率,消融实验验证了所提约束的有效性.

未来工作主要包括 3 个方面:

1)本文所提智能体动态协作图可以有效刻画多智能体任务中的近似可分解结构. 未来工作将尝试使用子图分割算法、图信息传递等技术来实现智能体动态协作图的高效建模,进而提升智能体协作效率.

2)本文算法需要预设子任务数量,其中子任务并无实际意义且引入额外的算法参数. 未来工作将尝试利用计算机视觉领域的语义分割模型来识别多智能体任务中已有实体,并将部分实体作为子任务,由此利用实体相关信息指导子任务内智能体协作.

3)本文算法依靠双层策略结构实现子任务分配策略和子任务执行策略学习. 然而,上下层策略同时更新会导致非稳态问题并影响策略学习. 未来工作将聚焦于预训练智能体间局部协作策略,随后将其固定作为下层子任务执行策略并单独学习上层子任务分配策略来实现具体多智能体任务中的协作策略学习.

作者贡献声明: 李超提出算法思路,完成实验并撰写论文;李文斌和高阳提出指导意见并修改论文.

参 考 文 献

[1] Cao Yongcan, Yu Wenwu, Ren Wei, et al. An overview of recent progress in the study of distributed multi-agent coordination[J]. IEEE Transactions on Industrial Informatics, 2012, 9(1): 427−438

[2] Wang Xiaoqiang, Ke Liangjun, Qiao Zhimin, et al. Large-scale traffic signal control using a novel multiagent reinforcement learning[J]. IEEE Transactions on Cybernetics, 2020, 51(1): 174−187

[3] Gao Han, Luo Juan, Cai Qianya, et al. An intelligent traffic signal coordination method based on asynchronous decision-making[J]. Journal of Computer Research and Development, 2023, 60(12): 2797−2805 (in Chinese)
(高涵, 罗娟, 蔡乾娅, 等. 一种基于异步决策的智能交通信号协调方法[J]. 计算机研究与发展, 2023, 60(12): 2797−2805)

[4] Zheng Yingying, Zhou Junlong, Shen Yufan, et al. Time and energy-sensitive end-edge-cloud resource provisioning optimization method for collaborative vehicle-road systems[J]. Journal of Computer Research and Development, 2023, 60(5): 1037−1052 (in Chinese)
(郑莹莹, 周俊龙, 申钰凡, 等. 时间和能量敏感的端—边—云车路协同系统资源调度优化方法[J]. 计算机研究与发展, 2023, 60(5): 1037−1052)

[5] De Souza C, Newbury R, Cosgun A, et al. Decentralized multi-agent pursuit using deep reinforcement learning[J]. IEEE Robotics and Automation Letters, 2021, 6(3): 4552−4559

[6] Zhang Weipeng, Zhang Ning, Yan Junchi, et al. Auto uning of price prediction models for high-frequency trading via reinforcement learning[J]. Pattern Recognition, 2022, 125: 108543

[7] Xu Jing, Zhong Fangwei, Wang Yizhou. Learning multi-agent coordination for enhancing target coverage in directional sensor networks[C/OL]//Advances in Neural Information Processing Systems. 2020[2024-05-10]. https://proceedings.neurips.cc/paper/2020/hash/7250eb93b3c18cc9daa29cf58af7a004-Abstract.html

[8] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518(7540): 529−533

[9] Simon H A. The architecture of complexity[J]. Proceedings of the American Philosophical Society, 1962, 106(6): 467−482

[10] Sunehag P, Lever G, Gruslys A, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward[C]//Proc of the Int Conf on Autonomous Agents and Multi-Agent Systems. Richland, SC, USA: International Foundation for Autonomous Agents

and Multiagent Systems/ACM, 2018: 2085−2087

[11] Rashid T, Samvelyan M, De Witt C S, et al. Monotonic value function factorization for deep multi-agent reinforcement learning[J]. The Journal of Machine Learning Research, 2020, 21(1): 7234−7284

[12] Son K, Kim D, Kang W J, et al. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning[C/OL]//Proc of Int Conf on Machine Learning. New York: PMLR, 2019[2024-05-10]. https://proceedings.mlr.press/v97/son19a

[13] Wang Jianhao, Ren Zhizhou, Liu Terry, et al. QPLEX: Duplex dueling multi-agent q-learning[C/OL]//Proc of Int Conf on Learning Representations. 2021[2024-05-10]. https://openreview.net/forum?id=Rcmk0xxIQV

[14] Lowe R, Wu Y I, Tamar A, et al. Multi-agent actor-critic for mixed cooperative-competitive environments[C]//Advances in Neural Information Processing Systems. Red Hook, NY: Curran Associates Inc, 2017, 30: 6379−6390

[15] Foerster J, Farquhar G, Afouras T, et al. Counterfactual multi-agent policy gradients[C]//Proc of the AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2018, 32(1): 2974−2982

[16] Wang Yihan, Han Beining, Wang Tonghan, et al. DOP: Off-policy multi-agent decomposed policy gradients[C/OL]//Proc of Int Conf on Learning Representations. 2021[2024-05-10]. https://openreview.net/forum?id=6FqKiVAdI3Y

[17] Peng Bei, Rashid T, Schroeder de Witt C, et al. FACMAC: Factored multi-agent centralised policy gradients[C/OL]//Advances in Neural Information Processing Systems. 2021[2024-05-10]. https://proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract.html

[18] Yu Chao, Velu A, Vinitsky E, et al. The surprising effectiveness of ppo in cooperative multi-agent games[C/OL]//Advances in Neural Information Processing Systems. 2022[2024-05-10]. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html

[19] Zhong Yifan, Kuba J G, Feng Xidong, et al. Heterogeneous-agent reinforcement learning[J]. Journal of Machine Learning Research, 2024, 25: 1−67

[20] Ding Shifei, Du Wei, Guo Lili, et al. Multi-agent deep deterministic policy gradient method based on double critics[J]. Journal of Computer Research and Development, 2023, 60(10): 2394−2404 (in Chinese)
(丁世飞, 杜威, 郭丽丽, 等. 基于双评论家的多智能体深度确定性策略梯度方法[J]. 计算机研究与发展, 2023, 60(10): 2394−2404)

[21] Guestrin C, Lagoudakis M, Parr R. Coordinated reinforcement learning[C]//Proc of Int Conf on Machine Learning, San Francisco, CA: Morgan Kaufmann Publishers Inc, 2002, 2: 227−234

[22] Zhang Chongjie, Lesser V. Coordinated multi-agent reinforcement learning in networked distributed POMDPs[C]//Proc of the AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2011: 764−770

[23] Zhang Chongjie, Lesser V. Coordinating multi-agent reinforcement learning with limited communication[C]//Proc of the Int Conf on Autonomous Agents and Multi-Agent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013: 1101−1108

[24] Das A, Gervet T, Romoff J, et al. Tarmac: Targeted multi-agent communication[C/OL]//Proc of Int Conf on Machine Learning. New York: PMLR, 2019[2024-05-10]. https://proceedings.mlr.press/v97/das19a.html

[25] Kim W, Park J, Sung Y. Communication in multi-agent reinforcement learning: intention sharing[C/OL]//Proc of Int Conf on Learning Representations. 2021[2024-05-10]. https://openreview.net/forum?id=qpsl2dR9twy

[26] Mao Hangyu, Liu Wulong, Hao Jianye, et al. Neighborhood cognition consistent multi-agent reinforcement learning[C]//Proc of the AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI , 2020, 34(5): 7219−7226

[27] Zang Yifan, He Jinmin, Li Kai, et al. Automatic grouping for efficient cooperative multi-agent reinforcement learning[C/OL]//Advances in Neural Information Processing Systems. 2024[2024-05-10]. https://proceedings.neurips.cc/paper_files/paper/2023/hash/906c860f1b7515a8ffec02dcdac74048-Abstract-Conference.html

[28] Liu Weiwei, Peng Linpeng, Wen Licheng, et al. Decomposing shared networks for separate cooperation with multi-agent reinforcement learning[J]. Information Sciences, 2023, 641: 119085

[29] Liu Yilin, Luo Guiyang, Yuan Quan, et al. GpLight: Grouped multi-agent reinforcement learning for large-scale traffic signal control[C/OL]//Proc of the Int Joint Conf on Artificial Intelligence. 2023[2024-05-10]. https://www.ijcai.org/proceedings/2023/23

[30] Liu Shanqi, Hu Yujing, Wu Runze, et al. Adaptive value decomposition with greedy marginal contribution computation for cooperative multi-agent reinforcement learning[C]//Proc of the Int Conf on Autonomous Agents and Multi-Agent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2023: 31−39

[31] Wang Tonghan, Dong Heng, Lesser V, et al. ROMA: Multi-agent reinforcement learning with emergent roles[C/OL]//Proc of the Int Conf on Machine Learning. New York: PMLR, 2020[2024-05-10]. https://proceedings.mlr.press/v119/wang20f

[32] Wang Tonghan, Gupta T, Mahajan A, et al, et al. RODE: Learning roles to decompose multi-agent tasks[C/OL]//Proc of the Int Conf on Learning Representations. 2021[2024-05-10]. https://openreview.net/forum?id=TTUVg6vkNjK

[33] Yang Mingyu, Zhao Jian, Hu Xunhan, et al. LDSA: Learning dynamic subtask assignment in cooperative multi-agent reinforcement learning[C/OL]//Advances in Neural Information Processing Systems. 2022[2024-05-10]. https://proceedings.neurips.cc/paper_files/paper/2022/hash/0b4145b562cc22fb7fa50a2cd17c191d-Abstract-Conference.html

[34] Iqbal S, Costales R, Sha F. ALMA: Hierarchical learning for composite multi-agent tasks[C/OL]//Advances in Neural Information Processing Systems. 2022[2024-05-10]. https://proceedings.neurips.

cc/paper_files/paper/2022/hash/2f27964513a28d034530bfdd117ea31d-Abstract-Conference.html

[35] Yuan Lei, Wang Chenghe, Wang Jianhao, et al. Multi-agent concentrative coordination with decentralized task representation[C/OL]//Proc of the Int Joint Conf on Artificial Intelligence. 2022[2024-05-10]. https://www.ijcai.org/proceedings/2022/85

[36] Li Chenghao, Wang Tonghan, Wu Chengjie, et al. Celebrating diversity with subtask specialization in shared multiagent reinforcement learning[J]. IEEE Transactions on Neural Networks and Learning Systems, 2023, 1−15

[37] Samvelyan M, Rashid T, De Witt C S, et al. The starcraft multi-agent challenge[C]//Proc of the Int Conf on Autonomous Agents and Multi-Agent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2019: 2186−2188

[38] Wang Tonghan, Wang Jianhao, Wu Yi, et al. Influence-based multi-agent exploration[C/OL]//Proc of the Int Conf on Learning Representations. 2020[2024-05-10]. https://openreview.net/forum?id=BJgy96EYvr

[39] Mahajan A, Rashid T, Samvelyan M, et al. MAVEN: Multi-agent variational exploration[C/OL]//Advances in Neural Information Processing Systems. 2019[2024-05-10]. https://papers.nips.cc/paper_files/paper/2019/hash/f816dc0acface7498e10496222e9db10-Abstract.html



Li Chao, born in 1996. PhD candidate. Student member of CCF. His main research interests include reinforcement learning, multi-agent systems, and task modeling.

李　超, 1996 年生. 博士研究生. CCF 学生会员. 主要研究方向为强化学习、多智能体系统、任务建模.



Li Wenbin, born in 1991. PhD, associate researcher. Member of CCF. His main research interests include machine learning, meta learning, and continual learning.

李文斌, 1991 年生. 博士, 副研究员. CCF 会员. 主要研究方向为机器学习、元学习、持续学习.



Gao Yang, born in 1972. PhD, professor. Committee member of CCF. His main research interests include multi-agent reinforcement learning, game theory, and machine learning.

高　阳, 1972 年生. 博士, 教授. CCF 委员会委员. 主要研究方向为多智能体强化学习、博弈论、机器学习.