面向工业场景的边-云协同大语言模型细粒度推理任务卸载

廖玲玲[1],陶 铭[1*],谢仁平[1],张 引[2],袁华强[1]
(1. 东莞理工学院计算机科学与技术学院(网络空间安全学院),广东东莞 523808;
2. 电子科技大学信息与通信工程学院,四川成都 611731)

摘 要: 大语言模型(Large Language Model,LLM)在任务推理等领域展现出卓越性能. 然而,面向复杂工业场景的实时高效推理仍是亟待解决的关键问题. 传统中心化云推理架构受限于长思维链(Chain of Thought,CoT)推理延迟与数据传输拥堵,难以满足复杂工业推理任务对低时延的严苛需求;边缘侧部署的轻量化LLM能实现快速响应,但是推理能力受限,难以保障推理质量. 为此,边-云协同推理成为必然选择. 然而,单一模态的LLM难以兼顾模态特性和任务需求,多模态LLM高昂的算力成本限制了其普适性;直接利用LLM推理复杂任务容易陷入固有的幻觉困境,影响推理质量. 因此,本文提出了一种基于边-云协同的LLM细粒度推理任务卸载框架,在边缘端部署轻量化专属模态LLM,充分适配特定数据模态,低时延高效处理简单任务;在云端部署具备强大推理能力的多模态深度LLM,执行复杂逻辑推理任务,保障推理质量. 将复杂LLM推理任务细粒度地划分为三个阶段,并构建有向无环图(Directed Acyclic Graph,DAG). 在此基础上,进一步提出通信与推理任务执行模型,并将LLM推理任务建模为总体推理时延与成本加权和的最小化问题. 通过证明该问题是离散马尔可夫决策过程(Markov Decision Process,MDP),针对动态环境中子任务特征与系统资源状态的复杂交互,设计了融合置信上界(Upper Confidence Bound,UCB)的动作选择机制和反事实多智能体策略梯度(COunterfactual Multi-Agent policy gradient,COMA)的问题求解方案UCB-COMA,实现子任务调度顺序与推理子任务执行位置的联合最优决策. 实验结果表明,本文方案的性能优于对比方案.

关键词: 大语言模型;边-云协同;任务卸载;深度强化学习;工业物联网
基金项目: 国家自然科学基金(No.62572122,No.62572099)
中图分类号: TP393.1 **文献标识码:** A **文章编号:** 0372-2112(2025)11-3880-14
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250411

Fine-Grained Inference Task Offloading for Large Language Model in Industrial Edge-Cloud Collaborative Scenarios

LIAO Ling-ling[1], TAO Ming[1*], XIE Ren-ping[1], ZHANG Yin[2], YUAN Hua-qiang[1]
(1. *School of Computer Science and Technology (School of Cyberspace Security), Dongguan University of Technology, Dongguan, Guangdong* 523808, *China*; 2. *School of Information and Communication Engineering, University of Electronics Science and Technology of China, Chengdu, Sichuan* 611731, *China*)

Abstract: Large language model (LLM) has exhibited exceptional performance in inference. However, achieving real-time and high-efficiency inference in complex industrial scenarios remains a significant challenge. Traditional centralized cloud-based inference architectures are constrained by the latency of long chain of thought (CoT) reasoning and transmission bottleneck, rendering them inadequate to meet the stringent low-latency requirements of complex industrial inference. Conversely, although lightweight LLM deployed on the edge can achieve rapid response, limited inference capabilities also compromise the inference quality. Therefore, edge-cloud collaborative inference emerges as an inevitable choice. However, single-modal LLM struggle to accommodate modality-specific characteristics and diverse task requirements, while the widespread applicability of multimodal LLM is limited by the high computational costs. Moreover, directly employing an LLM for complex inference often leads to hallucinations, undermining inference reliability. To address the issues, a fine-grained

收稿日期:2025-05-23;录用日期:2025-10-11;责任编辑:覃怀银
*通讯作者:陶铭

LLM inference task offloading framework based on edge-cloud collaboration is proposed in this paper. Specifically, lightweight and modality-specialized LLM are deployed on the edge to efficiently process simple tasks with minimal latency, while a powerful multimodal deep LLM resides in the cloud to execute complex logical reasoning tasks, ensuring inference quality. Complex LLM inference is decomposed into three stages and modeled as a directed acyclic graph (DAG). With this representation, the communication and inference models are constructed, and the LLM inference is formulated as a minimization problem of the weighted sum between overall inference latency and cost. With the proof that the investigated problem can be transferred into a discrete Markov decision process (MDP), considering the complex interactions between subtask features and dynamic system resource states, a solution named UCB-COMA, integrating the upper confidence bound (UCB)-based action selection mechanism with counterfactual multi-agent policy gradient (COMA), is designed to enable joint optimization of subtask scheduling order and executing position of inference subtask. Experimental results demonstrate that the performance of UCB-COMA is superior to that of comparison schemes.

Key words: large language model; edge-cloud collaboration; task offloading; deep reinforcement learning; industrial internet of things

Foundation Item(s): National Natural Science Foundation of China (No.62572122, No.62572099)

1 引言

随着工业智能化进程加速推进,大语言模型(Large Language Model,LLM)以其强大的推理与泛化能力,在工业数据分析、任务规划与异常解释等领域的应用价值日益凸显. 在传统推理架构中,LLM普遍采用云端集中式部署,以端到端的方式执行推理任务[1]. 然而,工业物联网设备产生的海量实时推理请求对响应效率具有严苛要求. 尽管云服务器具备强大的计算与存储能力,但LLM在处理复杂推理任务时,长思维链(Chain of Thought,CoT)将导致高能耗以及大延迟. 此外,受带宽约束,数据传输拥堵时有发生,进一步影响了推理响应效率. 因此,传统云部署模式的固有缺陷,使其难以满足工业场景对推理实时性的刚性需求.

为应对上述局限性,业界通过模型蒸馏、剪枝等优化技术推出了轻量化LLM[2,3],使其可在资源受限的边缘服务器上运行,有效缓解了云端推理的延迟问题. 然而,受限于参数量缩减,轻量化LLM的推理能力存在固有瓶颈[4]. 面对工业领域普遍存在的多模态异构数据输入以及涉及复杂逻辑推理的场景,轻量化LLM难以覆盖所有必要的推理路径和逻辑细节,易导致CoT推理过程断裂或逻辑谬误,无法保障推理质量,难以满足工业复杂任务对高质量推理的核心诉求.

为此,边-云协同推理架构成为突破工业场景实时高效推理难题的研究焦点[5]. 该架构通过边缘服务器部署轻量化LLM,发挥其低时延响应优势,同时依托云端强大算力支撑复杂推理,形成高效协同体系. 然而,工业场景中推理任务的多样性与数据模态的异构性,对协同架构提出了更高要求. 单一模态LLM难以兼顾不同数据模态特性与任务需求,多模态LLM虽然能够应对多种数据类型,但需要更多的训练数据和算力资源,导致成本攀升. 因此,仍需进一步优化设计边-云协同推理架构,实现任务与模态间一致性匹配,从而满足工业场景的个性化推理需求,并有效提升整体推理效率与资源利用率.

虽然合理的边-云协同架构在一定程度上可缓解推理时延与模态适配问题,但LLM在推理过程中仍存在固有缺陷,其内生的幻觉问题(可借助外部检测机制,如Sriramanan等人[6]提出的LLM-Check进行幻觉检测)会显著降低复杂工业场景下的推理可靠性. 具体而言,复杂LLM推理问题通常需要多步推理,模型单次生成长答案时易因注意力分散产生幻觉(如忽略关键约束或虚构事实). 为解决该问题,针对复杂推理任务的泛化性难题,Zhou等人[7]提出了从最少到最多的提示策略,通过将复杂问题拆解为一系列具有递进关系的子问题,利用先行子问题的答案为后续推理提供支撑;Khot等人[8]针对LLM在少样本场景下难以覆盖完整推理CoT的问题,提出了分解提示策略,将复杂任务解构为简单的子任务,并分配至专属LLM执行. 通过将复杂任务拆解为可靠的小单元,能够将幻觉风险限制在原子子问题内,从根本上隔离并缩短幻觉的传播路径,避免错误在长链推理中扩散放大,进而有效提升推理的准确性. 因此,为有效提升复杂LLM任务的推理质量,将复杂的LLM推理任务划分为多个具有逻辑依赖的子任务[9],并利用边-云协同架构支撑子任务的高效执行是一种行之有效的方法.

在边-云协同的工业LLM推理场景中,子任务的数据模态多样性、需求差异性,叠加网络带宽与计算负载的动态变化,使得推理任务卸载决策成为影响整体效率的关键问题. 低效的卸载决策可能导致额外的推理时延,显著增加计算成本[10]. 考虑时延、能耗、成本等因素的优化需求,研究人员设计了多种卸载策略,如置信上界(Upper Confidence Bound,UCB)算法[11]和深度强化学习(Deep Reinforcement Learning,DRL)算法[12,13]等. 然而,LLM推理任务内在的复杂依赖关系,使得卸

载策略的设计面临新的挑战. 子任务调度顺序直接影响边缘节点与云端的资源利用效率,且不同卸载位置产生的计算时延差异,可能改变关键路径任务的执行时序;此外,资源负载的动态变化要求卸载策略具备实时自适应能力. 这种时序约束与空间资源分配之间的强耦合性,使得单一维度的优化策略难以满足需求,亟需将子任务调度顺序与推理子任务执行位置决策进行联合优化[14]. 对比UCB等方法,DRL以其强大的高维状态空间建模能力与动态环境自适应能力,能够通过策略网络同时确定待调度子任务及对应子任务的推理执行位置,展现出更强的策略寻优能力.

因此,应对复杂工业场景中高效LLM推理的现实挑战,本文提出了一种面向工业场景的边-云协同大语言模型细粒度推理任务卸载策略,有效降低推理时延与成本. 本文的主要贡献总结如下:

(1)提出了一种边-云协同的LLM细粒度推理任务卸载框架. 在边缘端部署专属模态轻量化LLM,利用低时延特性高效处理简单子任务,并在云端部署多模态深度LLM,负责复杂子任务的精细推理,从而实现两者的优势互补与高效协同,有效提升面向复杂工业场景的LLM推理效率.

(2)基于任务分解思想,将LLM推理流程解耦为三阶段有向无环图(Directed Acyclic Graph,DAG). 在此基础上,进一步构建了通信模型与推理任务执行模型,将复杂LLM推理任务建模为总体LLM推理时延与成本加权和的最小化优化问题.

(3)证明了目标优化问题满足马尔可夫性质,并将其建模为离散马尔可夫决策过程(Markov Decision Process,MDP). 通过融合UCB动作选择机制与反事实多智能体策略梯度(COunterfactual Multi-Agent policy gradient,COMA)算法,提出了一种高效的推理任务卸载方案UCB-COMA. 实现子任务调度顺序与推理子任务执行位置的联合最优决策.

2 相关工作

近年来,LLM凭借其卓越的语言理解与生成能力,在工业数据分析与智能决策等领域实现了突破性应用. 然而,随着模型参数规模急剧攀升,其对计算资源的需求呈指数级增长. 与此同时,工业场景中LLM推理任务对响应延迟具有严格要求,高算力需求-低延迟约束的矛盾成为全面提升LLM推理效率的关键瓶颈. 如何设计高效的LLM推理框架并优化推理任务卸载决策,成为研究热点.

2.1 边-云协同LLM推理框架

传统LLM推理依赖云端集中式部署,虽具备强大算力支撑,但面临网络通信延迟高、云端负载失衡以及运行成本激增等问题[15]. 为了克服上述问题,研究者们通过模型压缩技术(量化、剪枝、早退机制等)[16]实现LLM轻量化,使其可部署于边缘设备执行本地化推理,有效减少时延并缓解云端压力. 然而,受限于参数规模,轻量化LLM通常难以处理复杂LLM推理任务,其推理质量仍显不足. 为了充分发挥边缘侧和云端的优势,边-云协同推理架构应运而生,成为实现更高效、更灵活的推理服务的关键路径.

在此背景下,为扩展协同推理的应用场景,Hu等人[17]针对车联网场景,提出了一种基于多模态LLM的边-云协同架构,其中较小的多模态LLM部署于边缘侧,较大的多模态LLM则部署于云端,利用LLM强大的环境感知和常识性推理能力,实现协同驾驶辅助服务. 此外,考虑设备异构性、带宽限制、模型复杂性等因素,Zhang等人[18]提出了一种基于动态规划的模型切分与任务分配方法,将计算密集型LLM划分为多个可负担的子模块,在边缘与云端之间合理调度,从而最小化推理延迟.

2.2 LLM推理任务卸载

边-云协同架构的效能发挥,依赖于融合算力、网络条件、任务特性等多种因素的智能卸载决策. 为实现灵活、低延迟且高质量的边-云协同推理服务,研究者提出了多种LLM推理任务卸载策略. 面对边-云协作环境中任务需求的多样性与资源状态的动态变化,Yang等人[11]将LLM推理服务调度问题建模为组合多臂老虎机问题,并提出结合约束满足机制的UCB算法,提升服务调度与资源分配的灵活性与鲁棒性. He等人[19]提出一种基于主动推理的LLM推理任务卸载和资源分配方法,通过感知环境状态主动调整策略,增强LLM在边-云协同环境下的性能. 类似地,Ren等人[20]设计了一种基于身份管理的边-云协同推理框架,并结合基于智力的强化学习方法,优化LLM推理任务卸载和资源分配. 面向6G网络,Zhou等人[21]提出了一种边-云协同LLM部署策略,并引入基于上下文学习的任务卸载机制,在保证生成内容质量的同时,有效降低内容的生成与传输延迟.

2.3 研究动机

尽管边-云协同的LLM推理框架已取得一定研究进展,但在复杂工业场景的应用中仍存在诸多亟待解决的关键问题. 现有研究大多忽略了任务与模态之间的适配问题. 在实际工业场景中,LLM推理任务涉及的数据模态各不相同,针对特定模态定制的单一模态LLM难以满足多样化任务需求;多模态LLM虽具备处理多元数据的能力,但其更高的算力需求导致系统成本增加. 此外,当前LLM推理任务通常被视为不可分割的整体进行推理卸载处理,未能充分考虑任务的内

在结构及逻辑关系. 在复杂工业推理流程中,LLM容易在中间推理环节产生“幻觉”错误,这些错误会沿着推理链不断传递累积,严重削弱最终结果的可靠性. 更为重要的是,面对边-云环境中数据模态的多样性与系统资源的动态性,现有LLM推理卸载策略的适应性不足,难以在复杂动态环境下兼顾依赖关系、模态约束等耦合影响因素,导致无法实现高效、可靠的推理决策. 为此,本文提出了一种边-云协同的LLM细粒度推理任务卸载框架,并设计了一种结合UCB动作选择机制和COMA算法的推理卸载决策方案UCB-COMA,对子任务调度顺序与推理子任务执行位置进行联合优化,从而有效降低推理时延和成本.

3　系统框架

基于上述讨论,本文所提出的边-云协同的LLM细粒度推理任务卸载框架如图1所示.

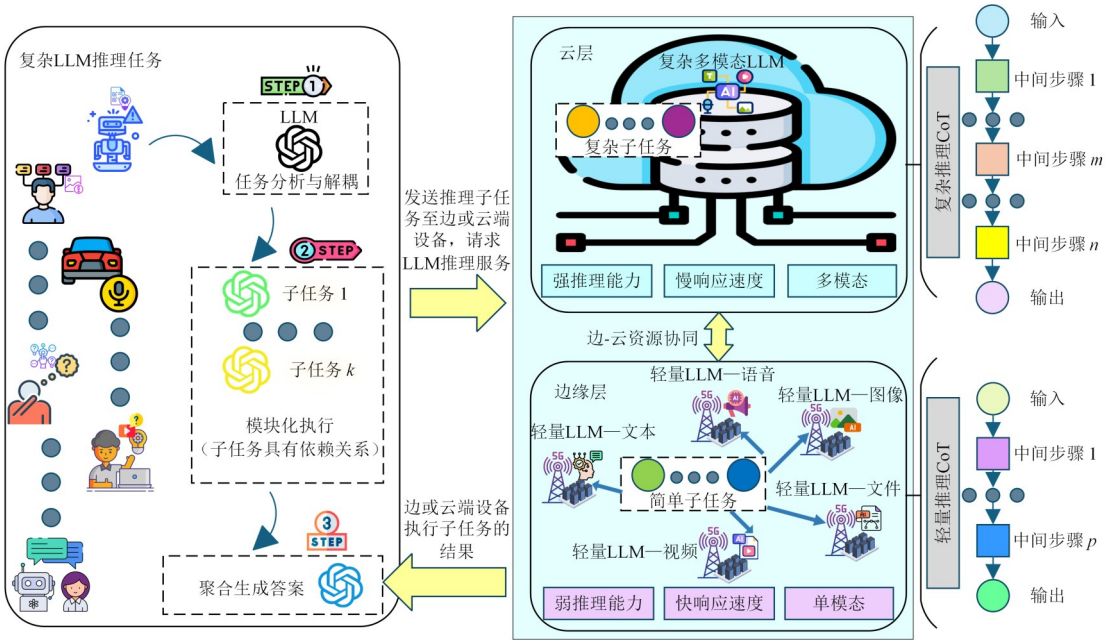

图1　边-云协同的LLM细粒度推理任务卸载框架

基于该框架,我们进一步构建了通信模型和推理任务执行模型,将工业场景中的复杂LLM推理任务建模为总体推理时延与成本加权和的最小化优化问题. 其中,所涉及的重要变量的符号表示及其含义如表1所示.

假设工业场景中有I个复杂LLM推理任务(如图像识别、文件处理等),表示为集合$\mathcal{L}=\{L_1,\cdots,L_i,\cdots,L_I\}$. 为有效减少复杂LLM推理任务的幻觉累积,提升推理效率,基于任务分解思想,将任务L_i细粒度划分为三个阶段:任务分析与解耦、模块化执行和聚合生成答案. 经任务分析与解耦阶段确定,模块化执行阶段可能包含多个具备依赖关系的子任务. 因此,L_i可以抽象为一个DAG:$G_i=(V_i,E_i)$. 其中,$V_i=\{L_{i,1},\cdots,L_{i,j},\cdots,L_{i,J}\}$表示$L_i$中所有子任务的集合. 子任务$L_{i,j}$可描述为一个四元组$\left(C_{i,j},D_{i,j},M_{i,j},R_{i,j}\right)$,其中$C_{i,j}$是完成$L_{i,j}$所需要的CPU周期数,$D_{i,j}$是执行$L_{i,j}$的输入数据量,$M_{i,j}$是$L_{i,j}$的待处理数据模态类型(如语音、文本等),$R_{i,j}$是$L_{i,j}$的输出数据量. E_i是有向弧的集合,表示L_i中子任务间的依赖关系,若$\left\langle L_{i,1},L_{i,2}\right\rangle\in E_i$,表示$L_{i,1}$必须在$L_{i,2}$调度执行之前完成,即$L_{i,1}$是$L_{i,2}$的前驱子任务.

表1　重要变量的符号表示及其含义

符号	含义
$L_{i,j}$	LLM推理任务L_i的第j个子任务
$C_{i,j}$	完成$L_{i,j}$所需要的CPU周期数
$D_{i,j}$	$L_{i,j}$的输入数据量
$M_{i,j}$	$L_{i,j}$的待处理数据模态类型
$R_{i,j}$	$L_{i,j}$的输出数据量
LLM_n	在第n个边缘服务器部署的轻量LLM
LLM_{N+1}	在云端部署的多模态深度LLM
$\alpha_{i,j}$	推理子任务$L_{i,j}$的执行位置决策变量
$\mathrm{ST}_{i,j}^{\alpha_{i,j}}$	$\mathrm{LLM}_{\alpha_{i,j}}$最早开始执行$L_{i,j}$的时刻
$\mathrm{PR}_{i,j}$	$L_{i,j}$的前驱子任务$L_{i,k}$的集合
$\mathrm{EN}_{i,j}^{\alpha_{i,j}}$	$L_{i,j}$在$\mathrm{LLM}_{\alpha_{i,j}}$上执行完成的时刻
$\mathrm{Cost}_{i,j}^{\alpha_{i,j}}$	$L_{i,j}$在$\mathrm{LLM}_{\alpha_{i,j}}$上的执行成本

由于推理子任务可能涉及不同的服务请求,为支持多样化推理需求,在边缘侧部署N个单模态轻量化LLM,记为$\mathrm{LLM}=\{\mathrm{LLM}_1,\cdots,\mathrm{LLM}_n,\cdots,\mathrm{LLM}_N\}$(第$n$个

边缘服务器部署LLM_n),并在云端部署一个具备强大推理能力的多模态LLM,记为LLM_{N+1}. 进一步地,为$L_{i,j}$定义一个推理子任务执行位置决策变量$\alpha_{i,j}\in\{1,\cdots,n,\cdots,N+1\}$. 若$\alpha_{i,j}=n$,表示$L_{i,j}$被卸载至边缘侧$\mathrm{LLM}_n$上执行. 若$\alpha_{i,j}=N+1$,表示$L_{i,j}$被卸载至云端$\mathrm{LLM}_{N+1}$上执行.

3.1 通信模型

上述系统框架中,考虑实际工业场景需求,本文采用如下方式定义所涉实体之间的通信模型. 设备采用无线链路与接入点通信,接入点采用有线链路经核心网与云服务器通信. 考虑核心网的数据传输延迟相对较大,为此,设备与云服务器之间的通信仅考虑有线链路部分,且上行/下行数据传输速率分别表示为$r^{\mathrm{ul}}_{i,N+1}$和$r^{\mathrm{dl}}_{i,N+1}$. 边缘服务器之间、边缘服务器与云服务器之间均采用有线链路连接,数据传输速率分别表示为r_e和r_{ec}. 设备与边缘服务器之间采用无线链路连接. 由于多个复杂LLM推理任务可能被同时卸载至同一边缘服务器,传统的共享信道方式可能导致严重的信道干扰,降低通信质量. 为此,本文采用正交频分多址(Orthogonal Frequency Division Multiple Access,OFDMA)技术,划分正交子信道,并为每个设备与边缘服务器之间的通信分配独立子信道,最大限度地减少干扰. 考虑传输速率受发射功率、信道质量、带宽及噪声等因素影响,基于香农定理,设备与边缘服务器之间的上行链路传输速率可定义为$r^{\mathrm{ul}}_{i,n}=W\cdot\log_2\left(1+\dfrac{P^d_i\cdot G_{i,n}}{\sigma^2}\right)$,其中$W$为设备与边缘服务器之间的通信带宽,$P^d_i$为设备的发射功率,$G_{i,n}$为设备与边缘服务器之间的信道增益,$\sigma^2$为高斯噪声功率. 同理,下行链路的数据传输速率定义为$r^{\mathrm{dl}}_{i,n}=W\cdot\log_2\left(1+\dfrac{P^e_n\cdot G_{i,n}}{\sigma^2}\right)$,其中$P^e_n$为边缘服务器的发射功率.

3.2 推理任务执行模型

在边-云协同支持下,LLM推理子任务$L_{i,j}$可以被卸载到边缘服务器或云端执行. 由于边缘服务器的计算资源有限,多个子任务可能在同一时间被卸载至同一边缘节点,因此,需要考虑因计算资源争用导致的排队延迟. 对于云端,由于其拥有更强大的计算能力,可以及时处理子任务,本文忽略云端的排队延迟.

3.2.1 边端执行

当$\alpha_{i,j}=n$时,推理子任务$L_{i,j}$将被卸载到第n个边缘服务器上部署的LLM_n上执行. 由于子任务之间存在复杂的依赖关系,在数据层面,只有当在云端或其他边缘服务器上处理的前驱子任务$L_{i,k}$执行完成,并将结果数据传输至LLM_n后,$L_{i,j}$才可启动执行. 为此,$L_{i,j}$在LLM_n上执行前的数据等待时刻$\mathrm{WT}^{n,\mathrm{data}}_{i,j}$可定义为$\mathrm{WT}^{n,\mathrm{data}}_{i,j}=\max\left\{\max\limits_{L_{i,k}\in\mathrm{PR}_{i,j}\wedge\alpha_{i,k}\neq n}\left\{\zeta_{i,k}\cdot\left(\mathrm{EN}^{N+1}_{i,k}+\mathrm{TR}^{N+1,n}_{i,k}\right)+\left(1-\zeta_{i,k}\right)\cdot\left(\mathrm{EN}^{\alpha_{i,k}}_{i,k}+\mathrm{TR}^{\alpha_{i,k},n}_{i,k}\right)\right\}\right\}$,其中$\mathrm{PR}_{i,j}$是$L_{i,j}$的前驱子任务集合,$\zeta_{i,k}$是一个二进制变量,若$L_{i,k}$在边缘服务器上执行,$\zeta_{i,k}=0$,否则$\zeta_{i,k}=1$,$\mathrm{EN}^{N+1}_{i,k}$是$L_{i,k}$在云端部署的$\mathrm{LLM}_{N+1}$上执行的完成时刻,$\mathrm{TR}^{N+1,n}_{i,k}=\dfrac{R_{i,k}}{r_{\mathrm{ec}}}$是$L_{i,k}$将输出数据从云端$\mathrm{LLM}_{N+1}$传输至$\mathrm{LLM}_n$的时延,$\mathrm{EN}^{\alpha_{i,k}}_{i,k}$是$L_{i,k}$在第$\alpha_{i,k}$个边缘服务器上部署的$\mathrm{LLM}_{\alpha_{i,k}}$执行的完成时刻,$\mathrm{TR}^{\alpha_{i,k},n}_{i,k}=\dfrac{R_{i,k}}{r_e}$是$L_{i,k}$将输出数据从$\mathrm{LLM}_{\alpha_{i,k}}$传输到$\mathrm{LLM}_n$的时延. 因此,$\mathrm{LLM}_n$最早开始执行$L_{i,j}$的时刻$\mathrm{ST}^n_{i,j}$须满足$\mathrm{ST}^n_{i,j}\geqslant\mathrm{WT}^{n,\mathrm{data}}_{i,j}$.

此外,在计算资源层面,由于$L_{i,j}$需要等待在同一服务器上前序排队的任务$L_{q,k}$执行完毕后释放计算资源,才可启动执行,$\mathrm{ST}^n_{i,j}$也须满足$\mathrm{ST}^n_{i,j}\geqslant\max\limits_{L_{q,k}\in\mathrm{UN}^n_{i,j}}\left\{\left(1-\zeta_{q,k}\right)\cdot\mathrm{EN}^n_{q,k}\right\}$,其中$\mathrm{UN}^n_{i,j}$是在$L_{i,j}$之前被卸载到$\mathrm{LLM}_n$上的子任务集合,$\mathrm{EN}^n_{q,k}$是$L_{q,k}$在$\mathrm{LLM}_n$上执行的完成时刻. 综合考虑数据依赖和资源争用,$L_{i,j}$在$\mathrm{LLM}_n$上的最早开始执行时刻表示为$\mathrm{ST}^n_{i,j}=\max\left\{\mathrm{WT}^{n,\mathrm{data}}_{i,j},\max\limits_{L_{q,k}\in\mathrm{UN}^n_{i,j}}\left\{\left(1-\zeta_{q,k}\right)\cdot\mathrm{EN}^n_{q,k}\right\}\right\}$. 当轮到$L_{i,j}$在$\mathrm{LLM}_n$执行时,$L_{i,j}$的输入数据$D_{i,j}$需传输至$\mathrm{LLM}_n$,此过程引发的数据传输延迟为$\mathrm{TR}^n_{i,j}=\dfrac{D_{i,j}}{r^{\mathrm{ul}}_{i,n}}$. 数据传输完成后,开始执行$L_{i,j}$. 执行时延可定义为$T^{n,\mathrm{com}}_{i,j}=\dfrac{C_{i,j}}{f_n}$,其中$f_n$是第$n$个边缘服务器的计算能力. 综合数据传输延迟与执行时延,$L_{i,j}$在LLM_n上的执行完成时刻$\mathrm{EN}^n_{i,j}$可表示为$\mathrm{EN}^n_{i,j}=\mathrm{ST}^n_{i,j}+\mathrm{TR}^n_{i,j}+T^{n,\mathrm{com}}_{i,j}$.

为量化计算资源利用的经济成本,本文引入价格机制定义执行成本. $L_{i,j}$在LLM_n上的执行成本可定义为$\mathrm{Cost}^n_{i,j}=\dfrac{C_{i,j}}{10^9}\cdot\lambda_n$,其中$\lambda_n$是第$n$个边缘服务器的计算资源单价(单位:\$/GHz).

3.2.2 云端执行

同理,若$\alpha_{i,j}=N+1$,推理子任务$L_{i,j}$将在云端部署的LLM_{N+1}上执行. 由于$L_{i,j}$的前驱子任务$L_{i,k}$可能在边缘侧LLM上执行,因此需要等待$L_{i,k}$的输出数据传输至云端LLM_{N+1}. 为此,$L_{i,j}$在LLM_{N+1}上的最早开始执行时刻$\mathrm{ST}^{N+1}_{i,j}$可定义为$\mathrm{ST}^{N+1}_{i,j}=\max\limits_{L_{i,k}\in\mathrm{PR}_{i,j}}\left\{\left(1-\zeta_{i,k}\right)\cdot\left(\mathrm{EN}^{\alpha_{i,k}}_{i,k}+\mathrm{TR}^{\alpha_{i,k},N+1}_{i,k}\right)\right\}$,其中$\mathrm{EN}^{\alpha_{i,k}}_{i,k}$是$L_{i,k}$在第$\alpha_{i,k}$个边缘服

务器上部署的$\mathrm{LLM}_{\alpha_{i,k}}$上执行完成的时刻,$\mathrm{TR}_{i,k}^{\alpha_{i,k},N+1}=\frac{R_{i,k}}{r_{\mathrm{ec}}}$是$L_{i,k}$将输出数据$R_{i,k}$从$\mathrm{LLM}_{\alpha_{i,k}}$传输至云端$\mathrm{LLM}_{N+1}$的时延.$L_{i,j}$在$\mathrm{LLM}_{N+1}$上的执行时延可以表示为$T_{i,j}^{N+1,\mathrm{com}}=\frac{C_{i,j}}{f_{N+1}}$,其中$f_{N+1}$是云服务器的计算能力.因此,考虑$L_{i,j}$的输入数据传输时延$\mathrm{TR}_{i,j}^{N+1}=\frac{D_{i,j}}{r_{i,N+1}^{\mathrm{ul}}}$,$L_{i,j}$在$\mathrm{LLM}_{N+1}$执行完成的时刻$\mathrm{EN}_{i,j}^{N+1}$可表示为$\mathrm{EN}_{i,j}^{N+1}=\mathrm{ST}_{i,j}^{N+1}+\mathrm{TR}_{i,j}^{N+1}+T_{i,j}^{N+1,\mathrm{com}}$.

此外,$L_{i,j}$在LLM_{N+1}的执行成本可以表示为$\mathrm{Cost}_{i,j}^{N+1}=\frac{C_{i,j}}{10^9}\cdot\lambda_{N+1}$,其中$\lambda_{N+1}$表示云服务器的计算资源单价(单位:\$/GHz).

3.3　问题定义

L_i的子任务存在依赖关系且部分子任务可并行处理,L_i的总执行时延可定义为最晚调度子任务的执行结束时间与最早调度子任务的开始执行时间的差.L_i的最终执行结果需回传至本地设备,其完整推理时间可定义为$T_i=\left(\max\limits_{j\in\{1,2,\cdots,J\}}\mathrm{EN}_{i,j}^{\alpha_{i,j}}-\min\limits_{j\in\{1,2,\cdots,J\}}\mathrm{ST}_{i,j}^{\alpha_{i,j}}\right)+\mathrm{TR}_{i,J}^{l}$.其中,$\mathrm{TR}_{i,J}^{l}$是将最终结果从边缘或云服务器回传到本地设备的时延,且$\mathrm{TR}_{i,J}^{l}=\frac{R_{i,J}}{r_{i,\alpha_{i,J}}^{\mathrm{dl}}}$.$L_i$的完整推理成本可表示为所有子任务推理成本的累加,即$\mathrm{Cost}_i=\sum\limits_{j=1}^{J}\mathrm{Cost}_{i,j}^{\alpha_{i,j}}$.

在工业场景中,推理延迟与生产效率呈负相关关系.为保障工业生产效率,必须严格控制推理延迟.此外,考虑工业生产的成本,也必须严格控制推理任务执行所需的资源成本.为此,针对复杂工业LLM推理任务,亟需在保障推理实时性的同时,有效控制资源使用的成本,考虑将总体推理任务处理时延和推理成本的加权和作为优化目标.据此,在本文所考虑的场景中,LLM细粒度推理任务卸载可以建模为一个带约束的优化问题:

$$\begin{aligned}&\left\{\min_{\alpha_{i,j},\ \zeta_{i,j}}\ \sum_{i=1}^{I}\left(\gamma\cdot T_i+\left(1-\gamma\right)\cdot\mathrm{Cost}_i\right)\right\}\\ \text{s.t. C1:}\ &\alpha_{i,j}\in\left\{1,\cdots,n,\cdots,N+1\right\},\\ \text{C2:}\ &\zeta_{i,j}\in\left\{0,1\right\},\\ \text{C3:}\ &T_i\leqslant T_{\max},\\ \text{C4:}\ &M_{i,j}\in\Theta_{\alpha_{i,j}},\\ \text{C5:}\ &\mathrm{ST}_{i,j}^{n},\ \mathrm{ST}_{i,j}^{N+1}\end{aligned}\tag{1}$$

其中,$\gamma(\gamma\in[0,1])$是加权因子,用于平衡推理时延与成本的权重;$\Theta_{\alpha_{i,j}}$表示$\mathrm{LLM}_{\alpha_{i,j}}$所支持的模态类型的集合;C1表示$L_{i,j}$需卸载至某一边缘服务器或云端服务器进行推理;C2表示$L_{i,j}$的推理位置限定于边缘端或云端;C3表示推理任务处理时延不得超过最大可接受时延$T_{\max}$;C4表示卸载$L_{i,j}$时必须满足模态约束,即只能将其卸载至支持$L_{i,j}$模态类型的服务器进行执行;C5表示执行$L_{i,j}$前必须满足数据与资源约束,即$L_{i,j}$必须在其所有前驱子任务完成、结果数据成功传输至目标服务器,且目标服务器处于空闲状态时方可开始执行.

4　解决方案

在动态变化的工业场景中,LLM推理任务卸载决策过程需满足多重约束条件.为高效求解该优化问题,LLM推理任务的卸载策略必须具备强大的自适应更新能力.为此,通过证明LLM推理任务卸载可建模为离散MDP,提出了融合UCB动作选择机制与COMA算法的UCB-COMA方案,实现子任务调度顺序与推理子任务执行位置的联合最优决策.

4.1　MDP问题公式化

由于LLM推理任务卸载问题已建模为总体推理时延与成本加权和的最小化优化问题,考虑LLM推理任务的依赖关系、模态限制、子任务调度顺序与推理子任务执行位置的多维耦合特性,为实现复杂动态工业场景下的高效决策,可将目标优化问题转化为离散MDP问题[22],具体证明如下.

证明　在本文研究场景中,定义了I个复杂LLM推理任务.为此,可定义I个智能体并构成集合$\mathcal{A}\triangleq\{\mathrm{ag}_1,\cdots,\mathrm{ag}_i,\cdots,\mathrm{ag}_I\}$.每个智能体$\mathrm{ag}_i$在离散决策时刻观测当前状态$s_i$,从离散联合动作空间中选择动作$a_i$(包括待调度子任务与推理子任务执行位置).执行$a_i$后,环境状态受动作影响转移到新的状态$s_i'$,并生成即时奖励$r_i$.$\mathrm{ag}_i$基于新状态$s_i'$继续选择下一动作,与环境持续交互.由此,在LLM推理任务的卸载过程中,动作选择仅依赖于当前状态,与之前的历史状态无关.同样,状态转移概率满足$p\left(s_{i+1}|s_i,a_i,\cdots,s_0,a_0\right)=p(s_{i+1}|s_i,a_i)$,即当前时刻的状态仅由前一时刻的状态与动作决定,与之前的其他状态和动作条件无关.并且决策过程以一定间隔推进,时间步是离散的.因此,本文考虑的LLM推理任务卸载问题可以转化为离散MDP.

证毕.

为实现子任务调度顺序与推理子任务执行位置的协同决策,离散MDP的主要组件定义如下.

4.1.1　状态空间

当智能体ag_i为L_i执行决策时,其所观测到的状态s_i由推理子任务执行状态、服务器(边缘与云端)特征及LLM推理任务拓扑特征共同构成.具体而言,推理子任

务执行状态表示为长度等于子任务数量的二值编码向量$\boldsymbol{P}_i$,元素取值为1表示对应子任务已完成,0表示未完成. 该向量实时更新,动态反映子任务的执行进度. 服务器特征(记为$\mathcal{S}$),涵盖所有边缘服务器与云服务器的计算能力、推理成本、部署的LLM支持的模态类型、与本地设备的数据传输速率及当前任务队列的排队时延. 考虑到L_i可被建模为DAG,传统的手工特征往往只能刻画节点的静态属性,难以有效捕捉子任务间的依赖关系. 由于图卷积神经网络(Graph Convolutional Network,GCN)通过在节点特征更新的过程中自适应地聚合邻居信息,既能捕捉子任务自身特征,又能编码整个DAG的拓扑结构[23],因此,本文采用两层GCN对DAG任务加以编码,每层隐藏维度均为64. 随后,通过全局均值池化将节点级别的隐含表征整合为固定维度的图表示,进而经一层线性映射,将其转换为输出维度为16的任务状态向量$\mathbf{Task}_i$. 该过程高效而全面地将结构化任务信息转化为适用于DRL的LLM推理任务特征表示,可为后续策略学习奠定可靠基础. 综上,s_i可以表示为$s_i=\left(\boldsymbol{P}_i,\mathcal{S},\mathbf{Task}_i\right)$.

4.1.2 动作和奖励空间

由于子任务调度顺序受限于依赖关系,而推理子任务执行位置决策受限于模态要求,两者的联合决策直接影响推理时延与成本. 因此,将待调度子任务与子任务执行位置联合建模为动作空间,表示为$a_i=\left(L_{i,j},\alpha_{i,j}\right)$.

联合动作的有效性与否靠奖励值体现. 为引导策略在满足系统约束的同时最小化优化目标,奖励函数设计为多层级结构. 为避免子任务调度顺序违背依赖关系,根据未完成的前驱子任务数量n,设定惩罚项$\eta_1=\vartheta\cdot n$,其中ϑ是固定惩罚因子. 为抑制模态不匹配的动作选择,引入惩罚项$\eta_2=\vartheta\cdot m$,其中m为二进制变量($m=1$表示卸载位置违反模态约束,$m=0$表示满足约束). 当所有约束均满足时(即$\varrho=1$,ϱ为约束满足指示变量),给予正向激励θ,强化合法动作的选择. 为确保L_i满足最大可接受时延约束,引入惩罚项$\eta_3=\vartheta\cdot\max\left(0,\left(\mathrm{EN}_{i,j}^{\alpha_{i,j}}-T_{\max}\right)\right)$. 综合,奖励值$r_i$可定义为$r_i=-\eta_1-\eta_2+\varrho\cdot\theta-\eta_3-\gamma\left(\mathrm{TR}_{i,j}^{\alpha_{i,j}}+T_{i,j}^{\alpha_{i,j},\mathrm{com}}\right)-\left(1-\gamma\right)\mathrm{Cost}_{i,j}^{\alpha_{i,j}}$.

4.2 UCB-COMA

在复杂动态的工业场景中,LLM推理任务卸载面临环境时变性与多约束耦合的双重挑战. 基于DRL的解决方案因其卓越的环境自适应能力与高维状态处理能力,成为理想选择. 针对多智能体协同决策中的个体贡献评估与全局协作平衡的核心难题,COMA通过引入集中式评论家网络与反事实基线机制,在联合决策中展现出平衡“个体收益”与“团队协同”的独特优势[24]. 为进一步提升算法在非平稳环境中的决策效率,本文创新性地将UCB机制[25]嵌入COMA框架,提出UCB-COMA的问题求解方案,其整体框架如图2所示.

COMA框架主要采用演员-评论家的思想,包含多个策略网络、一个集中式的Q网络以及对应的目标Q网络. 在训练开始时,将Q网络的参数完整拷贝到目标Q网络. 随后,每隔固定步数,将当前Q网络的参数再次同步到目标Q网络,保证目标值计算的稳定性. Q网络接受全局状态S和所有智能体的联合动作$A=\left(a_1,\cdots,a_i,\cdots,a_I\right)$作为输入,输出对应的真实回报估计$Q(S,A)$. 为使得策略能够更快地朝着最优策略的方向学习,对于一个经验样本(S,A,R,S'),其中S、A、R、S'分别表示所有智能体的状态、动作、奖励和下一状态,使用均方时序差分误差作为Q网络的损失函数[26],$\mathcal{L}_{\mathrm{critic}}=E_{(S,A,R,S')}\left[\left(y-Q(S,A)\right)^2\right]$,其中$y=r+\psi\cdot Q'(S',A')$,$r$是所有智能体的累计即时奖励,$\psi$是折扣因子,$Q'(S',A')$是在下一状态$S'$和下一动作$A'$时由目标Q网络给出的目标$Q$值.

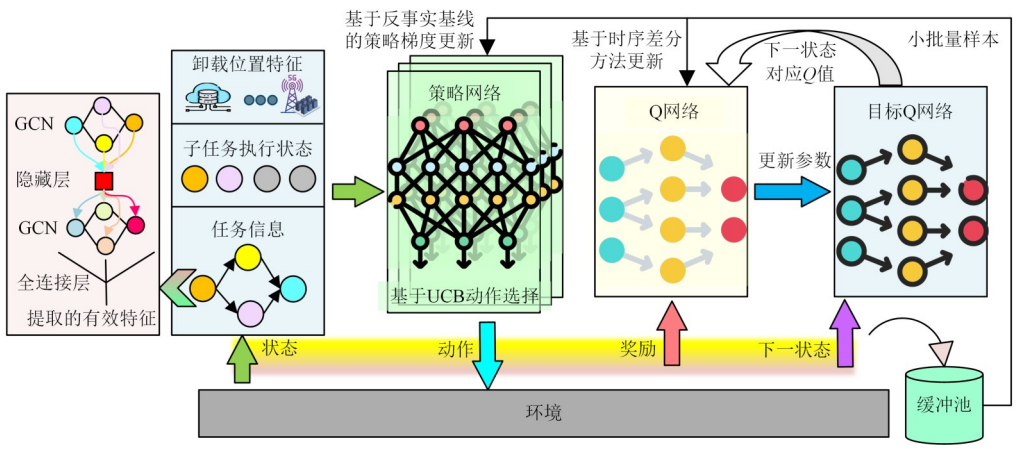

图2 UCB-COMA的问题求解方案框架

在多智能体强化学习中,每个智能体一般只能基于自身的局部观测做出决策,无法直接获取全局状态.同时,各智能体的动作对全局回报的贡献也各不相同.因此,为了促进多智能体协作,需要一种机制来衡量每个智能体执行的动作对整体目标的真实贡献.受差分奖励思想(每个智能体在执行完动作并获得一个全局奖励之后,假设它当初执行的是一个其他动作并计算出可能会获得的奖励,通过比较这两个奖励值来判断执行的动作对整体有多大的贡献)的启发,Jakob N. Foerster等人在COMA中引入"反事实基线"的概念,为每个智能体ag_i构造专属的优势函数.在保持其他智能体的动作A^{-i}不变的情况下,智能体ag_i选择其他动作时所能获得的加权Q值为反事实基线$B_i\left(S,A^{-i}\right)=\sum_{a_i'}\left(\omega_{\varphi_i}\left(a_i'\mid s_i\right)Q\left(S,\left(a_i',A^{-i}\right)\right)\right)$,其中$\omega_{\varphi_i}\left(a_i'\mid s_i\right)$是第$i$个智能体在其观察$s_i$下选择动作$a_i'$的概率,$Q\left(S,\left(a_i',A^{-i}\right)\right)$是在全局状态$S$时智能体执行动作$\left(a_i',A^{-i}\right)$对应的$Q$值.相应地,比较第$i$个智能体的真实动作$a_i$所对应的$Q$值与$B_i\left(S,A^{-i}\right)$的增益,就得到了衡量该动作贡献的优势值,可表示为$\mathcal{A}_i(S,A)=Q(S,A)-B_i\left(S,A^{-i}\right)$.

据此,第i个智能体的策略梯度可表示为$\nabla_{\varphi_i}J_i=E_{S,A}\left[\nabla_{\varphi_i}\ln\omega_{\varphi_i}\left(a_i\mid s_i\right)\mathcal{A}_i(S,A)\right]$.其中,优势函数$\mathcal{A}_i(S,A)$有效降低了对其他智能体动作随机性的敏感度,显著减少了梯度方差,并有效提升了多智能体协作的稳定性.

然而,在多智能体协同卸载这一高维离散的联合动作决策问题中,尽管COMA算法通过集中式Q网络与反事实基线有效解决了信用分配问题并降低了策略梯度方差,但其探索机制仍存在固有局限.具体而言,第i个智能体的策略网络接收观察s_i,输出对应动作组合的选择概率分布;而在训练初期,该分布近乎均匀,依赖该分布进行随机或按最大概率采样,相当于在庞大的组合动作空间中进行无导向的盲目探索,难以在有限训练步数内发现高性能动作组合.这种探索-利用失衡会导致探索效率低下、收敛速度缓慢,且易陷入局部最优,使得原生COMA难以在动态环境中快速定位高价值解空间.

考虑到UCB作为一种基于置信上界的乐观估计策略,已被广泛应用于多臂老虎机及强化学习场景,且其核心优势在于通过维护每个动作的期望回报估计及不确定性度量,优先选择"置信上界(即估计回报与不确定性之和)"最大的动作,从而实现探索与利用的平衡.为此,我们提出在动作选择层面引入UCB的乐观探索思想,将策略网络的利用信号与UCB提供的不确定性度量相结合,在保持COMA优势的同时显著增强采样阶段的探索效率,从而加速收敛并提高在动态工业场景下的鲁棒性.具体而言,我们将策略网络输出的动作对数概率作为利用项,叠加经典的UCB探索项,共同构成一个联合评分.对于第i个智能体,其在观察s_i下对待调度子任务$a^i_{\mathrm{sub}_j}$和推理子任务执行位置$a^i_{\mathrm{loc}_k}$的评分$g_i(\mathrm{sub}_j)$和$g_i(\mathrm{loc}_k)$分别定义为

$$g_i(\mathrm{sub}_j)=\ln\omega_{\varphi_i}\left(a^i_{\mathrm{sub}_j}|s_i\right)+\varepsilon\sqrt{\frac{2\ln N_i}{n_i\left(a^i_{\mathrm{sub}_j},a^i_{\mathrm{loc}_k}\right)+1}}\tag{2}$$

$$g_i(\mathrm{loc}_k)=\ln\omega_{\varphi_i}\left(a^i_{\mathrm{loc}_k}|s_i\right)+\varepsilon\sqrt{\frac{2\ln N_i}{n_i\left(a^i_{\mathrm{sub}_j},a^i_{\mathrm{loc}_k}\right)+1}}\tag{3}$$

其中,$\ln\omega_{\varphi_i}\left(\cdot\mid s_i\right)$是策略网络输出的对数概率,体现当前模型的利用能力;N_i为第i个智能体的累计尝试次数;$n_i\left(a^i_{\mathrm{sub}_j},a^i_{\mathrm{loc}_k}\right)$是$\left(a^i_{\mathrm{sub}_j},a^i_{\mathrm{loc}_k}\right)$的执行次数;$\varepsilon$是探索因子,会随训练轮数线性衰减,用于平衡探索与利用.在训练时,每一步都优先选择得分更高的待调度子任务和推理子任务执行位置,从而在保持策略网络已有经验优势的同时,通过UCB探索项有效挖掘罕见或未充分尝试的动作组合,加速整体收敛过程.

综上所述,本文所提出的UCB-COMA算法流程如算法1所示.

5　实验与分析

5.1　实验设置

仿真实验在Ubuntu 18.04.6 LTS操作系统下开展,基于PyTorch深度学习框架进行程序开发.在硬件配置方面,本文设置3台边缘服务器,分别部署支持文本、图像和音频处理的轻量化LLM,其计算能力f_n依次为2.0 GHz、2.3 GHz和2.5 GHz;同时配置一台云服务器,部署支持多模态类型的深度LLM,f_{N+1}为10 GHz.边缘服务器推理成本λ_n设定为0.01 \$/GHz,λ_{N+1}为0.03 \$/GHz[27].在数据传输速率方面,分别设定r_e、$r^{\mathrm{ul}}_{i,N+1}$、$r^{\mathrm{dl}}_{i,N+1}$和r_{ec}为22 Mbps、3 Mbps、12 Mbps和12 Mbps[28].为充分验证UCB-COMA算法在多样化任务下的性能与泛化能力,如图3所示,我们精心选取了五种在结构和模态上具有高度代表性的典型工业推理任务DAG作为实验用例,包括文本分析、图像分析、音频分析、视觉问答及音频-文本分析.这些DAG覆盖了任务卸载中最核心的拓扑模式(线性、分支-合并、多源异构融合)和模态组合(单模态文本/图像/音频、多模态图文、多模态音频文本),确保算法能够有效地迁移到具有相似拓扑或模态特征的新未知任务上.此外,对于不同模态的子任务,其数据量设置如下,文本类型子任务的输入和结果数据量范围均为[1,3] Kbit;图像、音频和多模态类型子任务的输入数据量范围为[0.1,

算法1 UCB-COMA算法

输入:学习率lr,ψ,训练批次大小B,经验回放池$\mathcal{D}$,训练迭代次数$\mathbb{E}$

输出:最优的LLM推理任务卸载决策

1. 随机初始化策略网络参数φ_i、Q网络和目标Q网络参数
2. 将N_i、$n_i\left(a^i_{\text{sub}_j},a^i_{\text{loc}_k}\right)$、训练次数step和更新次数upt_step设为0
3. FOR eps=1 TO $\mathbb{E}$
4. 环境初始化
5. IF NOT done THEN
6. 收集训练数据(S,A,R,S'),其中A基于UCB动作选择机制产生
7. 记录R的总奖励,并将(S,A,R,S')存入$\mathcal{D}$
8. IF step%32==0 THEN
9. 从$\mathcal{D}$中采样B个样本
10. 基于$\mathcal{L}_{\text{critic}}$更新Q网络
11. 基于$B_i\left(S,A^{-i}\right)$,$\mathcal{A}_i(S,A)$和$\nabla_{\varphi_i}J_i$更新策略网络
12. IF upt_step==10 THEN
13. 更新目标Q网络,upt_step=0
14. ELSE
15. upt_step+=1
16. END IF
17. END IF
18. step$\leftarrow$step+1, $S\leftarrow S'$
19. END IF
20. 计算该轮的平均奖励、推理完成时延及推理成本
21. eps+=1
22. END FOR
23. 输出最优的LLM推理任务卸载决策

0.3]Mbit、[0.2,0.4]Mbit和[0.3,0.6]Mbit,结果数据量范围为[2,4]Kbit、[3,5]Kbit和[2,6]Kbit. 同时,设定单位比特所需的CPU周期数为1 900. 在每个任务的仿真用例中,节点的相关数据量均在上述范围内随机生成. 这种参数扰动机制进一步增强了对同一任务族内不同实例的适应性,提升了实验的鲁棒性. 本实验的其他关键参数设置如表2所示.

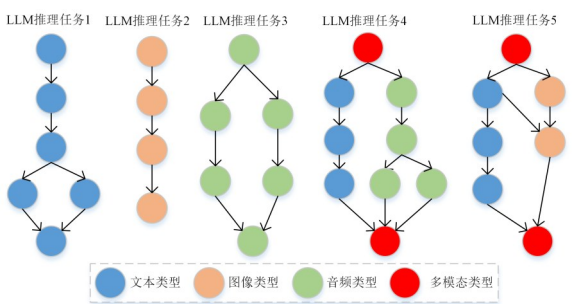

图3 LLM推理任务实验图

5.2 性能评估

为全面评估UCB-COMA的性能,本文选取多智能体深度确定性策略梯度(Multi-Agent Deep Deterministic Policy Gradient,MADDPG)[29]、多智能体近端策略优化(Multi-Agent Proximal Policy Optimization,MAPPO)[30]、COMA、约束满足上置信区间(Constraint Satisfaction-Upper Confidence Bound,CS-UCB)[11]及随机(Random)五种典型卸载算法进行对比分析. 五种对比算法的原理概述如下:

(1)MADDPG:为每个智能体分别构建策略网络与Q网络,融合策略梯度方法与Q学习. 此外,在训练过程中引入目标网络稳定策略更新,策略网络联合生成待调度子任务与推理子任务执行位置,Q网络基于全局状态与联合动作进行价值评估.

(2)MAPPO:作为近端策略优化(Proximal Policy Optimization,PPO)在多智能体场景的扩展,MAPPO采用中心化值网络学习全局价值函数,策略网络负责输出联合决策,利用裁剪后的优势函数在全局信息下对策略进行更新.

(3)COMA:基于演员-评论家架构的核心思想构建,是面向多智能体协作问题的强化学习算法. 通过反事实基线显著降低策略梯度方差,并借助集中式评论家网络对协作决策的全局奖励进行精准评估.

(4)CS-UCB:该算法融合了约束满足机制与UCB算法,其核心逻辑为通过约束满足机制筛选出所有符合给定约束条件的动作,再从这些有效动作中选取UCB值最大的动作执行.

(5)随机:该算法在子任务调度顺序与推理子任务执行位置的选择上均采用随机方式,不依赖任何环境或任务特征.

在W=6 MHz和$T_{\max}$=3.0 s的条件下,图4展示了六种卸载方案在700个训练轮次下的奖励变化趋势. 为了提高可读性,图中曲线使用了指数移动平均(Exponential Moving Average,EMA)进行平滑,阴影区域表示原始未平滑数据. 由于未考虑子任务内部的复杂逻辑依赖关系与模态类型约束,Random策略完全无目标导向,奖励值始终在低水平振荡,未出现有效优化. 相比

表2 关键参数设置

参数	值
W	{5,5.5,6} MHz
P_i^d	0.3 W
P_n^e	0.8 W
σ^2	−43 dBm
γ	0.5
$T_{\max}$	{2.5,2.8,3.0} s
ϑ	1.0
θ	2.0
lr	1×10^{-3}
ψ	0.95

之下,其他五种算法的奖励均随着训练轮次的增加而逐步提升,体现了结构化决策的优势. 具体而言,MADDPG依托确定性策略与噪声探索机制保证了训练稳定性,奖励曲线呈缓慢上升趋势,但由于高维、多智能体联合动作空间中存在局部最优陷阱,且对局部扰动敏感,其整体性能提升受限. MAPPO需进行裁剪式更新,微小的策略调整可能引起联合动作分布变动,尽管相较MADDPG获得一定奖励提升,但曲线波动显著. 由于显式考虑了候选动作约束违规的"严重性",CS-UCB在前期能获得较高奖励;然而,在存在复杂依赖关系、模态与时延耦合的工业场景中,子任务间的资源竞争易导致次优决策,进而影响整体效率. 相比之下,COMA得益于反事实基线机制对策略梯度方差的有效抑制,并借助集中式评论家网络精准评估全局回报,能够有效捕捉子任务间的协同依赖关系与模态约束,使动作更新更具针对性. 相较于MADDPG、MAPPO和CS-UCB,其奖励提升更显著,验证了多智能体协作优化的有效性. 基于此,UCB-COMA在COMA框架中引入UCB动作选择机制,形成了"探索-利用"动态平衡策略,在训练前期通过UCB的置信区间机制加速高价值卸载策略的发现,促使奖励迅速攀升,加速收敛;在训练后期凭借COMA的稳定优化能力维持策略更新的收敛性,最终获得最高奖励值. 这一表现充分验证了在复杂动态工业场景下,UCB-COMA算法对细粒度LLM推理子任务调度与卸载决策的高效性和可靠性.

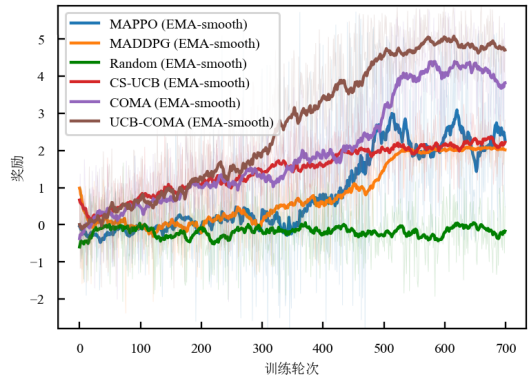

图4　在W=6 MHz和T_{max}=3.0 s的条件下,六种卸载方案的奖励变化趋势

在工业场景中,无线网络的动态波动特性对LLM推理任务卸载策略的适应性提出了严苛要求. 为验证算法在通信条件变化下的鲁棒性,图5在T_{max}=3.0 s的约束下,对比了六种算法在不同边-端W下的细粒度LLM推理任务平均奖励表现. 随着带宽的提升,数据传输时延降低,为边端与云端之间的子任务动态调度创造了更优条件. 然而,不同策略对带宽变化的适应能力存在差异. MAPPO算法受限于多维度耦合约束(任务依赖关系、模态适配需求等),其在线策略更新机制难以快速适应通信条件波动,导致策略调整与带宽改善的协同性不足,平均奖励未能得到提升. 借助噪声注入维持策略连续性,MADDPG保持平稳更新,但由于局部探索能力有限,难以充分发掘通信条件优化带来的调度潜力,平均奖励提升有限,暴露出自适应能力短板. Random策略因完全脱离网络状态与系统资源进行决策,其平均奖励在不同带宽条件下几乎无变化,凸显非自适应性决策的局限性. 相比之下,CS-UCB借助可行性过滤与UCB的探索-利用权衡机制,COMA则凭借反事实基线机制有效抑制策略梯度方差,使智能体能够一定程度上感知通信变化对卸载决策的影响. 因此,二者在不同带宽条件下保持相对稳定的平均奖励,整体性能优于MAPPO、MADDPG及Random. UCB-COMA进一步融合UCB动作选择机制与COMA的稳定优化能力,能快速识别通信条件变化下的最优卸载策略. 实验结果表明,UCB-COMA不仅能够充分释放带宽优化带来的性能增益,同时在动态通信环境中展现出卓越的适应性与鲁棒性,验证了其在工业场景下边-云协同LLM推理任务调度与卸载中的实用性与高效性.

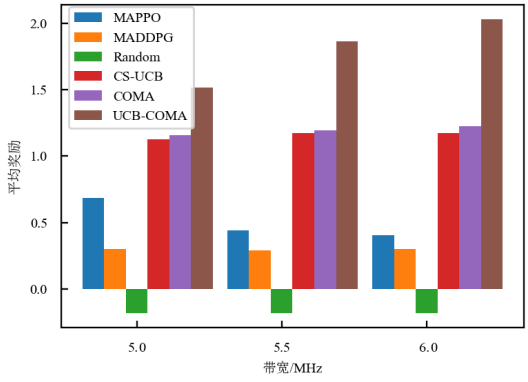

图5　在T_{max}=3.0 s的约束下,六种算法在不同边-端W下的平均奖励表现

在T_{max}=3.0 s的条件下,图6系统对比了不同边-端W下各算法的归一化推理时延与成本加权和(归一化至[0.1,1]区间). 实验结果表明,Random由于决策完全脱离网络状态与任务特性,其加权和无法形成有效优化,印证了非结构化决策在动态环境中的局限性. 虽然CS-UCB能够利用约束条件过滤动作,在奖励表现上占有一定优势,但在多重约束的复杂耦合下,优化目标存在一定牺牲,导致加权和反而最高. MAPPO与MADDPG分别受限于裁剪更新的不稳定性与局部噪声探索机制,难以有效利用带宽改善带来的更大调度空间优化决策,导致加权和居高不下,其在多维度耦合约束下对通信环境变化的适应性不足. COMA凭借反事实基线机制,整体加权和低于MADDPG与MAPPO,但

是由于缺乏动态的探索-利用平衡机制，面对带宽变化时难以快速调整子任务调度顺序与子任务执行位置．在多维度约束与带宽波动交织的复杂工业场景中，甚至出现加权和随带宽改善反向升高的现象，反映出策略优化对环境变化的滞后性．相比之下，UCB-COMA将UCB动作选择机制与COMA框架深度融合，通过动态平衡策略网络输出动作概率与选择次数，实现了高价值卸载策略的高效发现与稳定利用．在不同带宽条件下，均能保持最低加权和．不仅显著优化了推理时延与成本的综合指标，更在工业边-云协同LLM推理任务中，展现出对动态通信环境的卓越适应能力与全局最优决策优势．

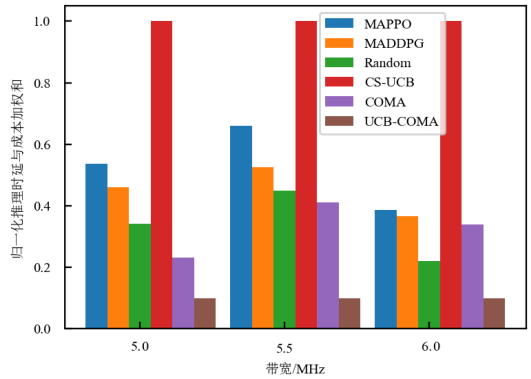

图6 在T_{max}=3.0 s的条件下，不同边-端W下的归一化推理时延与成本的加权和

在W=6 MHz条件下，图7对比了不同T_{max}下六种算法的平均奖励．Random由于完全不依赖任务特性与网络状态，其策略选择无差别，放宽或收紧时延约束对其平均奖励几乎无影响．在多维度约束下对联合动作空间的探索覆盖不足，导致对T_{max}变化的敏感度低，难以及时调整子任务调度顺序和推理子任务执行位置，MADDPG与MAPPO未能借助放宽的时延约束实现性能改进．相比之下，CS-UCB能通过约束满足机制剔除明显无效动作，并结合UCB实现探索-利用平衡；COMA利用反事实基线隔离个体动作对全局回报的影响，使每个智能体能够明确自身动作的贡献并减少策略更新的冲突．因此，二者在不同T_{max}下维持了相对稳定的奖励水平．然而，由于CS-UCB在多耦合约束下易受次优资源竞争影响，且COMA固有的探索与利用机制仍存在局限，二者的平均奖励始终不及UCB-COMA．在边-云协同LLM推理框架下，UCB-COMA算法将UCB的动作选择机制与COMA相结合，使算法在时延约束变化时能够迅速在探索与利用之间切换，实现子任务调度与推理子任务执行位置选择的联合优化，避免奖励塌缩，充分彰显了其在工业场景下的鲁棒性与高效性．

在W=6 MHz条件下，图8对比了不同T_{max}下六种

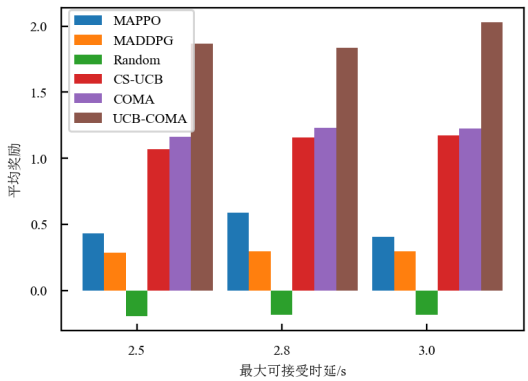

图7 在W=6 MHz条件下，不同T_{max}下六种算法的平均奖励

算法的归一化推理时延与成本加权和．UCB-COMA算法凭借独特的设计，在复杂动态环境中展现出卓越的适应能力．其通过策略网络深度融合子任务特征与系统资源状态，以概率形式动态生成推理子任务的调度顺序与执行位置．在此基础上，UCB-COMA创新性地将UCB探索机制与历史动作选择次数相结合，通过置信区间动态调整探索-利用平衡，有效规避了因重复选择高负载推理节点导致的时延与成本激增问题，始终取得最低的加权和，验证了其在复杂工业场景中的强大性能．相比之下，COMA算法通过反事实基线有效降低策略梯度方差，并借助集中式评论家网络实现全局决策价值的准确评估，在不同T_{max}下均能维持相对合理的加权和水平．然而，其策略更新依赖于网络初始分布，缺乏动态探索机制，导致其在应对复杂多变的工业环境时灵活性不足，加权和始终高于UCB-COMA．进一步对比MADDPG与MAPPO算法，二者在适应性方面存在明显短板．MADDPG依赖噪声驱动的局部探索，易陷入高维动作空间的次优解陷阱；MAPPO受限于裁剪机制对策略更新的约束，在T_{max}变化时难以快速适应．为此，二者的加权和均高于COMA与UCB-COMA．因完全脱离任务或网络特征，Random方式的加权和同样劣于UCB-COMA，印证了结构化决策机制在工业边-云协同LLM推理中的必要性．值得注意的是，CS-UCB虽在约束感知上有优势，但在多重耦合约束下可能为避免违规而牺牲总体优化目标，因而出现最高的加权和．

为进一步验证UCB-COMA算法的鲁棒性，在T_{max}=3.0 s和W=6 MHz的条件下，通过模拟边缘节点发生故障的情形（故障发生概率分别设置为0、0.1和0.2），图9对比了不同故障发生概率下UCB-COMA算法的平均奖励、平均时延与成本的加权和．随着故障发生概率的增加，UCB-COMA的整体性能呈下降趋势．然而，得益于UCB动作选择机制与COMA的深度融合，UCB-COMA在节点状态动态变化时能快速切换，可实现子任务调度顺序与执行位置的联合优化，因此，平均奖励的下降幅度较为微弱．即便在故障概率为0.2的极端场景下，

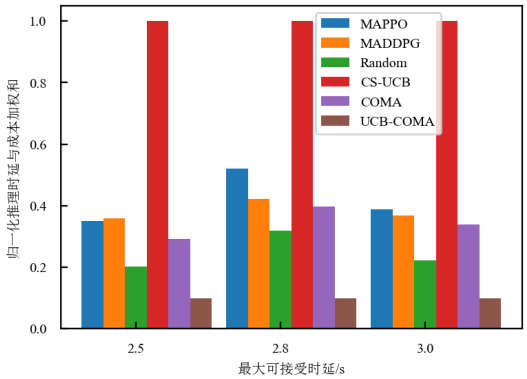

图8　在 $W=6$ MHz 条件下,不同 T_{max} 下六种算法的归一化推理时延与成本加权和

相较于图5~图8中无故障场景下其他算法的性能表现,UCB-COMA的整体性能仍具有明显优势,充分验证了算法的鲁棒性与高效性. 此外,受边缘节点偶发故障影响,更多子任务需卸载至云端执行,导致平均时延与成本的加权和出现一定增长. 这一现象从侧面印证了UCB-COMA对动态环境变化的适应能力.

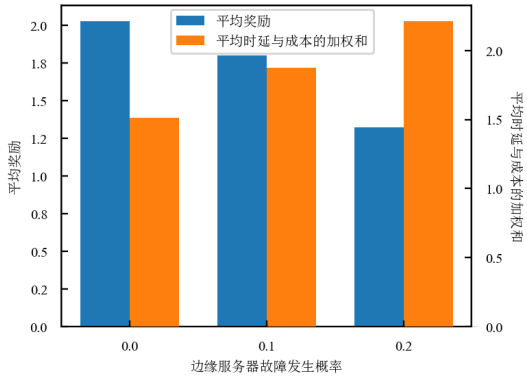

图9　不同故障发生概率下UCB-COM算法的平均奖励、平均时延与成本的加权和

6　结论

面向复杂工业任务的LLM推理,单模态LLM的模态适配局限性、多模态LLM的高算力成本及推理CoT的幻觉风险并存,致使高效实时推理面临严峻挑战. 为此,本文提出了一种边-云协同的LLM细粒度推理任务卸载框架,通过边缘侧的轻量化专属模态LLM与云端深度多模态LLM的协同机制,为工业场景提供高效优质的推理服务. 创新性地将复杂LLM推理任务解耦为多个细粒度子任务,采用DAG建模任务分解与依赖关系,并将LLM推理任务卸载建模为最小化总体推理时延与成本加权和的目标优化问题. 通过证明该目标优化问题可转化为离散MDP问题,进而综合考虑工业场景的时变性与多约束耦合特性,设计了一种融合UCB动作选择机制和COMA的问题求解方案UCB-COMA,实现子任务调度顺序和推理子任务执行位置的动态协同决策,为复杂LLM推理任务的高效执行提供有力支持. 实验结果表明,本文所提出的UCB-COMA在平均奖励和归一化推理时延与成本加权和的关键性能指标方面,均优于对比算法.

未来,如何进一步融合更多类型的轻量级多模态模型,持续提升整体推理效率,仍然是一个极具价值且值得深入探索的研究方向.

参考文献

[1] ZHANG C, YU M, WANG W, et al. Mark: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving[C]//Proceedings of 2019 USENIX Annual Technical Conference (USENIX ATC). California: USENIX Association, 2019: 1049-1062.

[2] MA X, FANG G, WANG X. LLM-pruner: On the structural pruning of large language models[J]. Advances in Neural Information Processing Systems, 2023, 36: 21702-21720.

[3] 张青龙, 韩锐, 刘驰. 云边协同大模型块粒度重训方法[J]. 电子学报, 2025, 53(2): 287-300.
ZHANG Q L, HAN R, LIU C. Cloud-edge collaborative retraining of foundation models at the block granularity[J]. Acta Electronica Sinica, 2025, 53(2): 287-300. (in Chinese)

[4] TANG X H, LIU F G, XU D S, et al. LLM-assisted reinforcement learning: Leveraging lightweight large language model capabilities for efficient task scheduling in multi-cloud environment[J]. IEEE Transactions on Consumer Electronics, 2025, 71(2): 5631-5644.

[5] ZHANG Z Y, ZHAO Y, LI H, et al. DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference[J]. IEEE Transactions on Mobile Computing, 2024, 23(10): 9042-9059.

[6] SRIRAMANAN G, BHARTI S, SADASIVAN V S, et al. LLM-check: Investigating detection of hallucinations in large language models[J]. Advances in Neural Information Processing Systems, 2024, 37: 34188-34216.

[7] ZHOU D, SCHARLI N, HOU L, et al. Least-to-most prompting enables complex reasoning in large language models[EB/OL]. (2023-04-16)[2025-03-23]. https://arxiv.org/abs/2205.10625.

[8] KHOT T, TRIVEDI H, FINLAYSON M, et al. Decomposed prompting: A modular approach for solving complex tasks[EB/OL]. (2023-04-11)[2025-03-23]. https://arxiv.org/abs/2210.02406.

[9] 秦龙, 武万森, 刘丹, 等. 基于大语言模型的复杂任务自

主规划处理框架[J]. 自动化学报, 2024, 50(4): 862-872.
QIN L, WU W S, LIU D, et al. Autonomous planning and processing framework for complex tasks based on large language models[J]. Acta Automatica Sinica, 2024, 50(4): 862-872. (in Chinese)

[10] TAO M, LIAO L L, XIE R P, et al. Bidding-enabled resource pricing for computation offloading in 6G vehicle-to-edge networks[J]. IEEE Transactions on Intelligent Transportation Systems, 2025, 26(10): 17838-17850.

[11] YANG Z M, YANG Y H, ZHAO C, et al. PerLLM: Personalized inference scheduling with edge-cloud collaboration for diverse LLM services[EB/OL]. (2024-05-23) [2025-03-23]. https://arXiv.org/abs/2405.14636.

[12] SEID A M, BOATENG G O, MARERI B, et al. Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network[J]. IEEE Transactions on Network and Service Management, 2021, 18(4): 4531-4547.

[13] LIAO L L, TAO M, DONG A N, et al. Graph-convolutional-network-enabled task offloading for industrial image recognition in digital twin edge networks[J]. IEEE Internet of Things Journal, 2025, 12(15): 29176-29188.

[14] 崔玉亚, 张德干, 张婷, 等. 一种面向移动边缘计算的多用户细粒度任务卸载调度方法[J]. 电子学报, 2021, 49(11): 2202-2207.
CUI Y Y, ZHANG D G, ZHANG T, et al. A multi-user fine-grained task offloading scheduling approach of mobile edge computing[J]. Acta Electronica Sinica, 2021, 49(11): 2202-2207. (in Chinese)

[15] LIN L, LIAO X F, JIN H, et al. Computation offloading toward edge computing[J]. Proceedings of the IEEE, 2019, 107(8): 1584-1607.

[16] 高晗, 田育龙, 许封元, 等. 深度学习模型压缩与加速综述[J]. 软件学报, 2021, 32(1): 68-92.
GAO H, TIAN Y L, XU F Y, et al. Survey of deep learning model compression and acceleration[J]. Journal of Software, 2021, 32(1): 68-92. (in Chinese)

[17] HU Y Q, YE D D, KANG J W, et al. A cloud-edge collaborative architecture for multimodal LLM-based advanced driver assistance systems in IoT networks[J]. IEEE Internet of Things Journal, 2025, 12(10): 13208-13221.

[18] ZHANG M J, SHEN X M, CAO J N, et al. EdgeShard: Efficient LLM inference via collaborative edge computing[J]. IEEE Internet of Things Journal, 2025, 12(10): 13119-13131.

[19] HE Y, FANG J C, YU F R, et al. Large language models (LLMs) inference offloading and resource allocation in cloud-edge computing: An active inference approach[J]. IEEE Transactions on Mobile Computing, 2024, 23(12): 11253-11264.

[20] REN Y Z, ZHANG H J, YU F R, et al. Industrial Internet of Things with large language models (LLMs): An intelligence-based reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2025, 24(5): 4136-4152.

[21] ZHOU H, HU C M, YUAN D, et al. Generative AI as a service in 6G edge-cloud: Generation task offloading by in-context learning[J]. IEEE Wireless Communications Letters, 2025, 14(3): 711-715.

[22] TAO M, LI X Q, FENG J, et al. Multi-agent cooperation for computing power scheduling in UAVs empowered aerial computing systems[J]. IEEE Journal on Selected Areas in Communications, 2024, 42(12): 3521-3535.

[23] CHEN M, WEI Z, HUANG Z, et al. Simple and deep graph convolutional networks[C]//Proceedings of Machine Learning Research (PMLR). Cambridge: PMLR, 2020, 119: 1725-1735.

[24] FOERSTER J, FARQUHAR G, AFOURAS T, et al. Counterfactual multi-agent policy gradients[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2018, 32(1): 2974-2982.

[25] CARPENTIER A, LAZARIC A, GHAVAMZADEH M, et al. Upper-confidence-bound algorithms for active learning in multi-armed bandits[M]//Algorithmic Learning Theory. Berlin, Heidelberg: Springer, 2011: 189-203.

[26] BRADTKE S J, BARTO A G. Linear least-squares algorithms for temporal difference learning[J]. Machine Learning, 1996, 22(1): 33-57.

[27] SHAH-MANSOURI H, WONG V W S, SCHOBER R. Joint optimal pricing and task scheduling in mobile cloud computing systems[J]. IEEE Transactions on Wireless Communications, 2017, 16(8): 5218-5232.

[28] ZHANG Q, YANG Y Y, YI C Y, et al. Energy- and cost-aware offloading of dependent tasks with edge-cloud collaboration for human digital twin[J]. IEEE Internet of Things Journal, 2024, 11(17): 29116-29131.

[29] LI S H, WU Y, CUI X Y, et al. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient[J]. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, 33(1): 4213-4220.

[30] WU Z, YU C, YE D, et al. Coordinated proximal policy optimization[J]. Advances in Neural Information Processing Systems, 2021, 34: 26437-26448.

作者简介

廖玲玲　女,2001年4月出生于江西省赣州市. 现为东莞理工学院计算机科学与技术学院硕士研究生. 主要研究方向为工业物联网.
E-mail: liaolingling@dgut.edu.cn

陶　铭　男,1986年6月出生于安徽省马鞍山市. 现为东莞理工学院计算机科学与技术学院教授. 主要研究方向为工业物联网.
E-mail: taom@dgut.edu.cn

谢仁平　男,1989年1月出生于湖南省娄底市. 现为东莞理工学院计算机科学与技术学院特聘副研究员. 主要研究方向为工业物联网.
E-mail: xierenping@dgut.edu.cn

张　引　男,1986年10月出生于江西省九江市. 现为电子科技大学信息与通信工程学院研究员、博士生导师. 主要研究方向为边缘计算、物联网.
E-mail: zhangyin123@uestc.edu.cn

袁华强　男,1966年12月出生于湖南省衡阳市. 现为东莞理工学院计算机科学与技术学院教授. 主要研究方向为工业物联网.
E-mail: yuanhq@dgut.edu.cn