小型微型计算机系统
Journal of Chinese Computer Systems
DOI:10.20009/j.cnki.21-1106/TP.2025-0186
2026年5月 第5期
Vol.47 No.5 2026

复杂移动边缘计算场景中基于动态信任评估的任务卸载方案

程界猛[1],虞慧群[1,2],范贵生[1,2]

[1](华东理工大学 信息科学与工程学院,上海 200237)
[2](上海智慧能源工程技术研究中心,上海 201103)
E-mail:js_cjm2022@163.com

摘 要: 任务卸载是移动边缘计算中的关键研究方向.尽管现有研究在优化计算任务时延和能耗方面取得了显著进展,但多数研究未充分考虑边缘计算场景的复杂性.边缘环境中资源节点不可靠性,设备异构性及任务多样性等因素,影响了任务卸载决策和系统性能.因此,本文提出了基于动态信任评估的任务卸载方案.该方案结合了两种算法:首先,仿照人类社会信任的演化机制,在边缘网络中构建了设备间的信任关系,为任务卸载提供可靠的资源节点信息.通过该机制,可有效避免因设备故障,恶意攻击或资源不足等问题引发的卸载失败.其次,采用改进的Q-learning算法求解任务卸载问题,以实现系统成本最小化的优化目标.本文提出的方案相较于启发式方案,系统成本降低了16.3%,任务成功率提升了32.1%.同时,实验进一步验证了信任值机制在筛选可靠资源节点方面的有效性.

关 键 词: 移动边缘计算;任务卸载;强化学习;信任值
中图分类号: TP393 **文献标识码:** A **文 章 编 号:** 1000-1220(2026)05-1236-09

Task Offloading Scheme Based on Dynamic Trust Evaluation in Complex Moblie Edge Computing

CHENG Jiemeng[1], YU Huiqun[1,2], FAN Guisheng[1,2]

[1](School of Informaton Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)
[2](Shanghai Engineer Research Center of Smart Energy, Shanghai 201103, China)

Abstract: Task offloading represents a pivotal research direction within mobile edge computing. While existing studies have achieved remarkable progress in optimizing computation latency and energy consumption, most fail to adequately address the inherent complexities of edge computing environments. Factors such as the unreliability of resource nodes, device heterogeneity, and task diversity significantly affect offloading decisions and overall system performance. To address these challenges, this paper proposes a task offloading scheme grounded in dynamic trust evaluation. The proposed approach integrates two algorithms: Firstly, drawing inspiration from the evolution of trust mechanisms in human society, a trust relationship model between devices is established within the edge network. This mechanism ensures the provision of reliable resource node information for task offloading, effectively mitigating failures caused by device malfunctions, malicious attacks, or resource insufficiency. Subsequently, an improved Q-learning algorithm is employed to solve the task offloading problem, aiming to minimize system costs. Compared to heuristic scheme, the proposed scheme reduces system costs by 16.3% and improves task success rates by 32.1%. Furthermore, experimental results validate the effectiveness of the trust-value mechanism in identifying reliable resource nodes.

Keywords: mobile edge computing; task offloading; reinforcement learning; trust value

0 引 言

近年来,随着第五代通信技术(The Fifth Generation Mobile Communication Technology, 5G)和Wi-Fi 6的发展,用户端到网络接入端的时延已降至个位数毫秒[1],这促进了更多用户接入网络.根据思科发布的互联网行业报告[2],截止2023年全球互联网用户数量预计将达到53亿(占全球人口的66%),连接到IP网络的设备数量将是全球人口的3倍以上,其中物联网连接占到一半以上.当前,智慧城市[3],虚拟现实[4],增强现实[5],自动驾驶[6]等新兴行业及应用呈现蓬勃发展态势.然而,受限于计算资源,存储空间和电池容量等因素,移动设备难以满足日益增长的用户需求.为此,移动边缘计算(Moblie Edge Computing, MEC)应运而生,它成为了突破传统云计算瓶颈的有效解决方案[7].其核心思想是将计算资源和数据存储从远程云计算中心迁移至网络边缘,在网络接入点或基站部署边缘服务器,从而减少数据传输需求.

在MEC架构中,任务卸载是关键技术之一,其主要目标是将任务从用户设备转移到其他资源节点进行处理,从而减

收稿日期:2025-04-30 收修改稿日期:2025-05-21 基金项目:国家自然科学基金项目(62372174)资助. 作者简介:**程界猛**,男,1998年生,硕士研究生,研究方向为边缘计算、任务卸载;**虞慧群**,男,1967年生,博士,教授,CCF高级会员,研究方向为软件工程、大数据;**范贵生**,男,1980年生,博士,副研究员,CCF会员,研究方向为软件工程、可信计算.

轻设备负担,提升系统整体性能. 然而,设计高效的任务卸载算法面临诸多挑战. 首先,MEC 服务器的计算资源有限,合理调配该资源以应对随时间变化的计算负载是关键问题;其次,任务卸载受限于时延和能耗,通常被视为多目标优化问题,且已证明为 NP-hard 问题[8]. 因此,设计高效的任务卸载方案对于提升移动边缘计算系统的性能和用户体验(Quality of Experience,QoE)具有重要意义.

目前,任务卸载研究存在的问题主要体现在以下方面. 首先,现有研究未充分考虑边缘计算场景的复杂性,通常假设资源节点是可靠的. 然而,边缘设备的状态在现实场景中是动态变化的,甚至可能存在恶意节点. 若将任务卸载至恶意节点,会导致卸载失败,从而对用户体验产生负面影响. 其次,边缘设备的异构性与任务的多样性增加了卸载问题的复杂程度,现有研究也未进行考虑. 针对上述的局限性,本文提出了一种基于动态信任评估任务卸载方案,主要贡献如下:

1)针对资源节点的不可靠性,提出了基于信任值机制的轻量化评估模型. 该模型能够动态评估资源节点,有效提升用户的卸载服务质量. 信任值机制通过反馈信任和互动信任两个维度,在边缘场景中构建了设备间的信任关系,从而实现了在动态网络环境中对边缘设备可靠性的评估.

2)考虑了边缘计算场景的复杂性,本文提出了多样化、可扩展的任务卸载体系结构. 采用了云-边-端的卸载架构,支持设备间(Device-to-Device,D2D)通信,融合了多种异构设备,并且满足了用户对多类型任务的需求.

3)构建了以系统成本为优化目标的任务卸载问题,并将其建模为马尔科夫决策过程. 采用了改进的 Q-learning 算法求解该问题. 考虑到任务卸载问题的相似性,本文通过多次训练获得的 Q 表进行初始化,从而减少了边缘设备早期阶段的探索过程,提升算法性能.

4)通过仿真实验模拟了复杂边缘环境. 结果表明,本文提出的任务卸载方案能够有效降低系统成本,提高任务卸载的成功率,且在大规模设备集群的场景中表现突出. 此外,通过引入不同比例的恶意节点的对比实验,验证了信任值机制在筛选可靠资源节点方面的有效性.

1　相关工作

任务卸载问题通常被表述为数学优化问题,旨在找到最优或接近最优的解决方案. 该问题的求解需要通过定义目标函数以及选择合适的优化算法来实现.

传统的数学优化方法在资源分配和网络调度问题中广泛应用,主要优点在于能够将问题进行数学建模并使用现有的优化方案求解. Ning 等人[9]针对物联网中云边协同优化的任务卸载问题,考虑单用户场景采用分支定界算法;多用户场景构建混合整数线性规划模型进行求解. Du 等人[10]提出了联合优化卸载决策,资源分配与功率带宽的多目标模型. 通过半定松弛与随机化生成卸载策略,结合了混合非线性整数规划及拉格朗日对偶分解,实现了资源动态分配. 传统的数学优化方法适用于小规模优化问题. 当面临大规模,复杂的动态网络环境时,往往表现出较高的计算复杂度和较差的适应性.

启发式的算法可以迅速找到局部最优解,但是无法保证获得全局最优解. 在面对复杂,动态的环境下,启发式算法具有更好的适应性,灵活性. Zhu 等人[11]将任务卸载表述为一个双目标最小化问题,基于线性规划优化和二进制粒子群优化,提出了动态任务分配框架,并使用现实世界的出租车痕迹进行模拟. Ketykó 等人[12]对一个可归约为多背包问题的多用户 MEC 卸载模型进行了评估,提出了一个精确的算法和一个启发式算法的行为,并展示了这些算法的性能结果. Guo 等人[13]对超密集环境下的移动边缘计算卸载问题,提出了一种启发式贪心卸载方案.

使用人工智能方法为任务卸载提供了高效,灵活且智能的解决方案,尤其适合处理复杂,动态和大规模的卸载任务. Qu 等人[14]提出了一种基于深度元强化学习的卸载算法,该算法将多个并行 DNN(Deep Neural Network)与 Q-learning 学习相结合. 通过综合深度学习,强化学习,元学习等优势,可以快速灵活地从动态环境中获得最优卸载策略. Lu 等人[15]提出了深度强化学习来解决大规模异构 MEC 中集群的多服务节点卸载问题和移动任务的多依赖问题,利用 LSTM(Long Short-Term Memory)网络层和候选网络集对 DQN(Deep Q-Network)算法进行改进. Zhu 等人[16]采用多智能体的深度强化学习方法,使用历史信息进行训练模型,并对环境做出实时的卸载决策. Atcor 和 Critic 网络在数据中心进行训练,车辆以分布式的方式执行决策.

对于解决任务卸载的优化问题,研究人员采用了丰富的算法,除上述的方法外,还可使用博弈论以及控制理论解决卸载问题,Jošilo 等人[17]考虑了在边缘计算系统中将无线资源分配给一组设备,设备可以自行决定是否使用边缘资源来减少计算任务的负担,从而最小化完成时间. 并且采用了 Stackelberg 博弈来模拟设备与操作者之间的相互作用. Mao 等人[18]提出了基于绿色能源收集的 MEC 系统的卸载策略,以时延和任务失败的成本作为优化目标,提出了基于 Lyapunov 优化的动态任务卸载算法.

2　系统模型与问题描述

本文的系统模型如图 1 所示,采用云-边-端的三层边缘计算架构,其中包含多个 MEC 服务器以及单个远程云计算中心,且在边缘侧分布着异构边缘设备.

异构设备被定义为集合 $S=\{\{d_i\},\{l_i\},\{r_i\}\}$,其中$\{d_i\}$代表区域内的移动设备. $\{l_i\}$为固定设备,具备较多的计算资源. $\{r_i\}$为物联网的传感器设备,其计算资源有限,且不可移动,作为任务卸载服务的消费者. 在边缘网络中部署了固定数量的 MEC 服务器,通常位于基站附近,用于处理来自用户的任务. 除 MEC 服务器外,还需部署信任代理服务器,为区域内的边缘设备提供信任服务.

本文提出的任务卸载方案如图 2 所示. 首先,边缘设备在生成任务数据后,根据强化学习输出的动作进行卸载决策. 策略具有多样性,任务可以卸载到 MEC 服务器或者远程云计算中心. 支持 D2D 通信的特性使任务可以卸载到其他边缘设备. 为优化卸载决策,方案结合了信任值机制,在边缘网络中维护了设备间的信任关系,目的是筛选出可靠的资源节点. 随后设备选择资源节点将任务卸载执行. 根据任务完成的状态

将信息反馈给信任代理服务器,由其更新信任值.

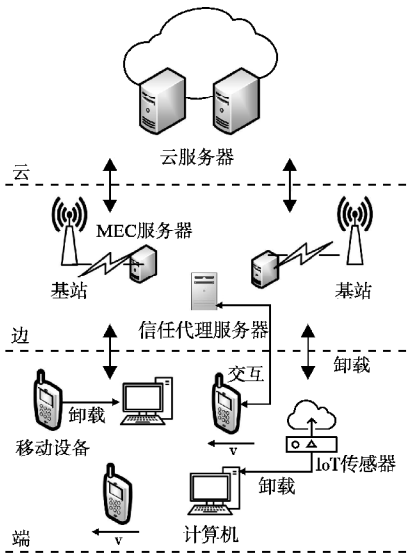

图 1 系统模型

Fig. 1 System model

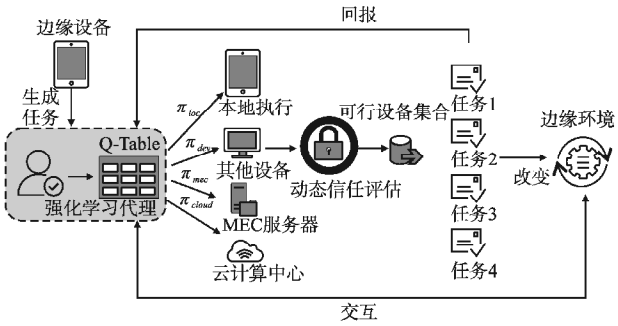

图 2 任务卸载方案流程图

Fig. 2 Task offloading scheme flowchart

2.1 信任模型

为构建动态信任评估的算法框架,需在设备集群中建立多维的信任模型,以形式化定义信任来源与量化方法. 信任是主观评价,用于衡量某个个体是否可靠. 此概念可扩展到边缘网络中,通过信任评估来判断设备的可靠性. 在移动边缘计算中,设备的可靠性直接影响了任务卸载的成功率,因此,信任评估成为优化卸载决策的关键因素. 本文首先采用设备最近的任务卸载成功率作为其历史行为的衡量标准,以定义反馈信任值(Feedback Trust,FT)[19].

考虑在时刻 t,设备 s_i 将任务卸载到设备 s_j. 任务卸载产生的结果 $X_{i,j}^t \in \{0,1\}$,其中 $X_{i,j}^t=1$ 表示本次任务卸载成功,反之 $X_{i,j}^t=0$ 表示失败. $X_{i,j}^t$ 的结果将作为 s_j 的历史行为反馈给信任代理服务器. 在感知到来自 s_i 的反馈后,信任代理服务器将按照时间顺序维护序列 Q_j 来记录 s_j 的历史行为,该序列被定义为:

$$Q_j=\{X_{i,j}^t,X_{i,j}^{t+1}\cdots,X_{i,j}^{t+n},\cdots X_{i+k,j}^{t+n}\} \tag{1}$$

其中,$X_{i,j}^t,X_{i,j}^{t+1},X_{i,j}^{t+n}$ 为设备 s_i 对设备 s_j 所提供的任务卸载结果的反馈. 而 $X_{i+k,j}^{t+n}$ 则是其他设备 s_{i+k} 在 $t+n$ 时刻的对 s_j 的反馈. 利用该序列,可计算设备 s_j 在 $\Delta t=n$ 时间段内的任务卸载服务成功率为:

$$\eta_{\Delta t}=\frac{\sum X_{\forall s\in\{S\},j}^t}{|Q_j|} \tag{2}$$

其中,$|Q_j|$ 表示序列 Q_j 中的元素数量,也即任务卸载服务次数. 定义 $\eta_{\Delta t}$ 为设备 s_j 在 Δt 时间内的反馈信任值. 由于计算的是设备 s_j 的历史行为,可能会出现各个时间段内行为重要性不同的情况. 从社会的视角来看,若根据历史行为判断一个人的可靠性,依据的是其最近的行为,因为久远的历史行为对当前的影响较小. 同理,在设备的信任评估中,近期的行为对评估该设备的可靠性具有更大权重.

为体现时间权重的特性,首先需要对设备 s_j 的历史行为序列 Q_j 按照时间片长度 τ 进行划分,将其分为 n/τ 个不同的时间片. 然后在这些时间片上根据公式(2)求出反馈信任为 $\{\eta_1,\eta_2,\cdots,\eta_{n/\tau}\}$. 反馈信任 $\eta_{n/\tau}$ 比 η_1 更加重要,因为 $\eta_{n/\tau}$ 的信息是近期产生的,反映了设备当前的可靠性状况.

为衡量某个特定时间片内的反馈信任权重值,本文设计了时间衰减函数 $\delta(t)$ 来计算反馈信任的权重值. 其中 t 表示当前时间片到最新时间片的差值. 假设当前时间片为 t,最新的时间片为 $t+\Delta t$,则计算权重值为 $\delta(\Delta t)$. 由此计算当前时间片的反馈信任值:

$$FT_{\forall s\in\{S\},s_j}^{\Delta t}=\delta(\Delta t)\cdot\eta_{\Delta t} \tag{3}$$

其中,$FT_{\forall s\in\{S\},s_j}^{\Delta t}$ 表示在 Δt 时间段内的反馈信任值. 将所有时间片的反馈信任值累加并归一化处理,从而得到设备 s_j 的反馈信任值为:

$$FT_{\forall s\in\{S\},s_j}=\frac{\sum_{\Delta t=1}^{t}\delta(\Delta t)\cdot\eta(\Delta t)}{\sum_{\Delta t=1}^{t}\delta(\Delta t)} \tag{4}$$

其中,$FT_{\forall s\in\{S\},s_j}$ 表示设备 s_j 的反馈信任值,该值基于 s_j 的历史行为,作为任务卸载服务提供者的反馈评价.

然而,仅依赖反馈信任值会出现一些问题. 一方面,如果发生共谋攻击[20],即设备之间相互提供虚假的反馈,信任代理服务器可能会计算出错误的信任值;另一方面,偶然性问题也会影响信任值的准确性,尤其是在新节点或服务次数较少的节点中. 因此,为修正反馈信任值,本文引入互动信任(Interaction Trust,IT)的概念. 从人类社会的视角来看,两者之间的互动次数越多,表明关系越熟悉,这为判断可靠性提供了重要的参考.

设备 s_j 为设备 s_i 提供任务卸载服务后,设备 s_i 除了将卸载结果 $X_{i,j}^t$ 反馈给信任代理服务器,还要记录与设备 s_j 的互动次数. 为此,设备 s_i 将维护元组 I_{s_i,s_j} 来记录与其他设备的互动次数,具体表示为:

$$I_{s_i,s_j}=\{(s_i,s_j,N)\,|\,s_j\in\{S\},N\geqslant 0\} \tag{5}$$

其中,I_{s_i,s_j} 表示设备 s_i 和 s_j 的互动次数,记作 N. 根据互动的次数计算出互动信任,即:

$$IT_{s_i,s_j}=\frac{I_{s_i,s_j}}{\sum_{k=1}^{n}I_{s_i,s_k}} \tag{6}$$

其中,n 为 s_i 互动元组的记录数量,IT_{s_i,s_j} 为设备 s_i 对 s_j 的互动信任,该值已归一化的处理. 如果 s_i 和 s_j 之间并没有产生互动,则互动信任被设置为默认值 W. 此外,在长时间未发送互动的前提下,互动信任也被设置为 W,因此互动信任是动态的.

全局信任(Global Trust,GT)可通过加权融合反馈信任和互动信任得出,它将作为任务卸载决策中的权威信息. 设备 s_j 对于 s_i 的全局信任为 GT_{s_i,s_j},计算为:

$$GT_{s_i,s_j}=FT_{s_i,s_j}+\alpha IT_{s_i,s_j},\alpha>0 \tag{7}$$

调整权重 α 的数值可影响互动信任对全局信任的贡献.

在设备 s_i 进行任务卸载决策前,需向信任代理服务器发送请求以获取区域内所有设备的信任状态. 该信息将作为优化卸载决策的重要参考. 此外,为降低网络请求的频率,该信息会被缓存至设备并定期更新.

2.2　时延和能耗计算

本节将对任务生命周期中所产生的时延与能耗进行建模,并以此提出任务卸载的优化问题.

在边缘网络中,设备 s_i 在随机时刻 t 产生任务 T_k,其中 T_k^s 表示任务的数据大小,T_k^m 表示执行任务所需的计算资源. 任务卸载过程中,任务数据通过无线信道传输至目标资源节点,此阶段时延与 T_k^s 成正比. 任务到达目标节点后,系统将拉取相应的应用服务并进行任务执行,其执行时间与 T_k^m 成正比.

根据系统模型的设计,定义卸载策略为 π. 当任务生成后,它将被传递至任务编排器(即设备本身)进行处理,由优化算法决定任务的具体卸载策略,包括 4 种:任务本地执行,卸载到其他设备,卸载到 MEC 服务器,卸载到云计算中心.

$$\pi\in\{\pi_{loc},\pi_{dev},\pi_{mec},\pi_{cloud}\} \tag{8}$$

卸载策略可归类为两种形式:本地执行与卸载执行. 针对不同策略,分别对任务产生的时延和能耗进行计算.

1)本地执行

当任务卸载策略 $\pi=\pi_{loc}$ 时,表示任务在本地执行,无需进行卸载. 在此情况下,任务执行所产生的时延由两部分组成:一是任务本地执行的时延;二是等待时延,发生在任务生成速率较高的情形,即任务生成速度超过本地编排器的处理速度,导致任务需排队等待调度. 为简化时延模型,使用平均等待时延 $\bar{t}$ 替代任务调度的时延. 设本地设备 s_i 的计算能力为 $f_{s_i}^{loc}$,则任务时延计算为:

$$D^{loc}=\frac{T_k^m}{f_{s_i}^{loc}}+\bar{t} \tag{9}$$

当任务在本地执行时,仅需考虑任务在本地计算过程中所产生的能量消耗. 设备 s_i 在执行单位指令能量消耗为 $J_{s_i}^c$,则任务在本地算产生的能耗计算为:

$$E^{loc}=J_{s_i}^c\cdot T_k^m \tag{10}$$

2)卸载执行

当任务被卸载到其他边缘设备上执行时,设备 s_i 通过无线信道将任务数据 T_k^s 传输至目标设备 s_j,然后 s_j 消耗计算资源执行任务,并返回处理结果. 由于返回结果的数据大小通常远小于任务本身,因此可忽略传输结果产生的时延. 假设 s_j 的计算能力为 f_{s_j},任务卸载产生的时延可计算为:

$$D^{dev}=\frac{T_k^m}{f_{s_j}}+\frac{T_k^s}{R_{s_i,s_j}}+\bar{t} \tag{11}$$

其中,R_{s_i,s_j} 为设备 s_j 与设备 s_j 之间的最大比特率,根据香农-哈特计算公式[21]得出,具体计算如下:

$$R_{s_i,s_j}=B_{s_i,s_j}\log_2\left(1+\frac{S}{\sigma^2}\right) \tag{12}$$

其中,B_{s_i,s_j} 为设备 s_j 与设备 s_j 之间的网络带宽,假设信道噪声为高斯白噪声,方差为 σ^2,信号功率 S 取决于路径损耗传播模型. 在卸载策略为 π_{mec} 时,此时目标设备变为 m_j,任务执行的总时延可以计算为:

$$D^{mec}=\frac{T_k^m}{f_{m_j}}+\frac{T_k^s}{R_{s_i,m_j}}+\bar{t} \tag{13}$$

其中,f_{m_j} 表示 MEC 服务器的计算能力,R_{s_i,m_j} 为设备 s_i 与 MEC 服务器之间的最大比特率. 同理,任务在远程云上执行产生的总时延为:

$$D^{cloud}=\frac{T_k^m}{f_{c_j}}+\frac{T_k^s}{R_{s_i,c_j}}+\bar{t} \tag{14}$$

其中,f_{c_j} 表示远程云计算中心 c_j 的计算能力,R_{s_i,c_j} 表示设备 s_i 与远程云之间的最大比特率.

若任务卸载执行,则任务产生的能耗来源于两方面:一是本地设备发送任务数据时的能耗;二是目标设备执行任务时的能耗. 设 s_i 每发送一个比特产生的能量消耗为 J_{s_i},当目标设备是 s_j 时,能耗计算为:

$$E^{dev}=J_{s_i}\cdot T_k^s+J_{s_j}^c\cdot T_k^m \tag{15}$$

其中 $J_{s_j}^c$ 表示在 s_j 执行单位指令的能量消耗. 当目标设备是 m_j 和 c_j 时,设在 MEC 服务器,远程云计算中心上执行单位指令的能量消耗为 $J_{m_i}^c J_{c_i}^c$,那么任务执行产生的能耗 E^{mec} 以及 E^{cloud} 可以由公式(16)、公式(17)得到:

$$E^{mec}=J_{s_i}\cdot T_k^s+J_{m_j}^c\cdot T_k^m \tag{16}$$

$$E^{cloud}=J_{s_i}\cdot T_k^s+J_{c_j}^c\cdot T_k^m \tag{17}$$

2.3　移动模型

本节定义了移动设备的运动规则. 在系统模型中,考虑了设备的移动性,这为任务卸载带来了挑战. 例如,当 $\pi=\pi_{dev}$ 时,任务被卸载到边缘设备,但设备间的无线通信距离 Υ 是有限的,如果设备之间的距离超出该范围,则任务可能失败. 本文采用了随机游走模型,在 t 时刻 s_i 设备的位置坐标为 $L_{s_i}(x(t),y(t))$,随后设备将会选择随机方向 θ,以速度 v 进行移动,在 $t+\Delta t$ 时刻设备的位置坐标由公式(18)、公式(19)更新. 其中,方向角 $\theta\in[0,2\pi)$.

$$x(t+\Delta t)=x(t)+v\cdot\Delta t\cdot\cos\theta \tag{18}$$

$$y(t+\Delta t)=y(t)+v\cdot\Delta t\cdot\sin\theta \tag{19}$$

2.4　问题描述

本节将提出具体的任务卸载优化问题. 假设在某段 Δt 时间内,设备 $s\in\{S\}$ 产生的任务集合为 $T=\{T_1,T_2,\cdots,T_k\}$. 对于任务 T_k,设备生成卸载决策 π,随后任务被发送到资源节点上执行. 在此过程中产生时延为 D_s^π 和能耗 E_s^π,定义任务 T_k 的成本 $C_{s,k}(\pi)$ 为时延和能耗的加权和,即:

$$C_{s,k}(\pi)=\alpha D_s^\pi+\beta E_s^\pi \tag{20}$$

其中,α 和 β 为权重参数,满足 $\alpha>0,\beta>0$. 进一步定义区域内的系统成本为所有任务产生的成本,即:

$$C_{\forall s\in\{S\},k}(\pi)=\sum_{n=1}^{k}(\alpha D_s^\pi+\beta E_s^\pi) \tag{21}$$

通过优化卸载决策来最小化成本函数 $C_{\forall s\in\{S\},k}(\pi)$,任务卸载问题可以被归结为 P1:

$$\min C_{\forall s\in\{S\},k}(\pi)$$

满足以下约束:

$$C1:\sum_{k=1}^{x}T_k^m\leqslant f_s$$

$$C2:\forall k, T_k^{finished} \leqslant T_k^d$$

$$C3:\pi \in \{\pi_{loc}, \pi_{dev}, \pi_{mec}, \pi_{cloud}\}$$

$$C4:\| L_{s_i} - L_{s_k} \| = \leqslant Y$$

在任务卸载的约束条件中,约束 C1 要求设备 s 的计算资源不得超过其所能承载的任务所需资源的总和. X 是在设备 s 执行的任务数量. 约束 C2 要求计算任务必须在指定的截至期限内完成,未能按时完成的任务被视作执行失败. 约束 C3 规定任务卸载的决策空间,根据系统设计任务可以本地执行,卸载到 MEC 服务器,卸载到远程云计算中信,卸载到其他边缘设备. 约束 C4 要求若任务卸载到附近设备,则两者的距离必须在无线通信范围内,保证任务卸载过程中通信的可行性.

除系统成本外,考虑到边缘场景的复杂性,若任务卸载到不可靠的资源节点,可能导致任务执行失败,进而影响用户的体验质量. 因此,必须引入动态评估的信任机制来筛选可靠的资源节点,以确保任务执行的可靠性并最大化用户的 QoE.

3 基于动态信任评估的任务卸载方案

3.1 动态信任评估方法

当卸载决策 $\pi = \pi_{dev}$ 时,需要从可行设备集合 $\{S'\}$ 中选取目标资源设备. 在复杂的边缘网络环境中,$\{S'\}$ 是通过信任值机制筛选出的,这些设备在资源充足且可靠性较高的前提下被选中. 通过使用动态信任评估算法,可以不断更新并确定可行设备集合 $\{S'\}$,从而优化任务卸载决策. 这种信任评估机制有助于确保任务被卸载到高可信度的设备上,以提高任务执行的成功率,并有效提升用户的体验质量.

在任务卸载过程中,资源节点的可用计算资源是关键因素. 若资源不足,卸载就可能失败. 因此,在选取可行设备集合时,还需要综合考虑节点当前的负载情况(Load Condition, LC). 本文综合考虑了当前任务的数据长度度 T_k^m,资源节点的计算能力 f_d 以及 CPU 使用率. 该值越小,说明资源节点的处理能力越强,资源利用越高效,更适合执行卸载任务. 将该值与全局信任值加权融合,得到每个资源节点的综合评分. 根据评分筛选出最终的可行设备集合 $\{S'\}$,并将该集合反馈给任务编排器 s_i,作为任务卸载的依据. 动态信任评估算法的流程如表 1 所示.

3.2 基于强化学习的任务卸载方法

在提出的任务卸载优化问题 P1 中,需要优化延迟和能耗的加权(即系统成本). 由于任务卸载涉及设备,边缘服务器和云服务器之间的动态复杂交互,传统的优化方法难以高效处理这一情况. 为了解决这一问题,本文将 P1 建模为马尔可夫决策过程(Markov Decision Process, MDP),并采用强化学习方法进行求解[22]. MDP 框架的核心元素定义如下:

1)状态空间(State Space)

根据系统模型的设计,对于区域内的设备 s_i,其在时刻 t 的状态空间定义为 S_t.

$$S_t = \{\rho_{loc}, \bar{\rho}_{mec}, \bar{\rho}_{cloud}, f_s, T_k^m, T_k^d\} \tag{22}$$

其中,ρ_{loc} 表示当前设备 s_i 的资源利用率,其计算方式为剩余计算资源与总计算资源的比值,反映了设备的空闲计算资源水平. $\bar{\rho}_{mec}$ 和 $\bar{\rho}_{cloud}$ 分别表示场景中 MEC 服务器和远程云计算资源的平均利用率. f_s, T_k^m, T_k^d 分别代表设备当前的计算能力,任务所需的计算资源以及任务的截至期限.

表 1 动态信任评估方法

Table 1 Dynamic trust evaluation method

算法 1. 基于信任机制的动态评估算法

输入: 设备 s_j 和任务 T_k;无线通信范围 Y,其他的设备集合为 D = $\{d_1, d_2, \cdots, d_n$ 以及时间窗口 $t = \{1, 2, \cdots, t\}$;设备 CPU 利用率 U_{d_n};交互记录 I_{s_j,d_n} 以及设备间距离 r_{s_j,d_n},任务执行状态 X^t

开始:

If t > 0

For each $i = 1$ to n **do**

If $Y > r_{s_j,d_n}$ **then**

设备 s_j 向信任代理服务器发送任务状态 $X^t_{s_j,d_i}$

信任代理服务器计算时间权重

根据公式(4)计算反馈信任值 FT_{s_j,d_i}

根据公式(6)计算交互信任值 IT_{s_j,d_i}

根据公式(7)计算全局信任值 GT

计算设备 d_n 的资源可用性 LC:

$$LC \leftarrow U_{d_i} \cdot \frac{T_k^m}{f_{d_i}}$$

End if

End each

信任代理服务器计算所有设备并进行打分:

$$f \leftarrow \alpha' GT + \beta' \frac{1}{LC}$$

信任代理服务器选取评分前 K 个的设备加入集合 $\{S'\}$

End if

输出: $\{S'\}$

结束

2)动作空间(Action Space)

根据观察到的状态,任务卸载决策的空间由任务编排器 s_i 在 t 时刻的可行动作集合构成,即:

$$A_t = \{\pi_{loc}, \pi_{dev}, \pi_{mec}, \pi_{cloud}\} \tag{23}$$

对于异构设备,其动作空间存在差异. 例如,传感器设备 r_i 由于自身计算资源有限,它的动作空间不包括本地执行策略 π_{loc}.

3)奖励(Reward)

奖励是环境根据卸载决策对任务编排器的即时反馈,并以标量形式表示. 奖励函数用于引导任务编排器的学习过程. 在系统中,设备 s_i 在时刻 t 观察环境后作出卸载决策 a_t,并根据任务执行的结果返回相应的奖励 R_t

$$R_t = \begin{cases} C_{s,k}(\pi) & if \quad X = 1 \\ N_0 \cdot C_{s,k}(\pi) & if \quad X = 0 \end{cases} \tag{24}$$

其中,$X = 1$ 表示任务执行成功的情况,此时即时奖励为本次卸载产生的成本. 为与本文提出的优化问题 P1 的目标保持一致,当 $X = 0$ 表示任务执行失败时,使用较大的常数因子 N_0 乘以卸载成本作为负向反馈. 任务编排器在每次决策中会选择回报最小的动作作为卸载决策,从而引导任务编排器实现最小的长期累积成本,以达到最小化系统成本的目标.

在强化学习中,代理的目标是最大化长期累积收益. 代理从某一状态 s_t 开始执行策略 π 直到某个终止状态. 从时间 t

开始的长期累积收益定义为:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (25)$$

其中,G_t 是从时间步长 t 开始的累积奖励. γ 是折扣因子取值在 0 ~ 1 之间,表示未来奖励的重要性,较小的折扣因子意味着更注重短期奖励. R_{t+k+1} 是在时间步 $t+k+1$ 的即时奖励. 定义代理执行卸载决策 a 获得的期望回报为 $Q(s_t,a)$.

$$Q(s_t,\pi) = E[G_t \mid S_t = s_t, A_t = a] \quad (26)$$

$Q(s_t,a)$ 又被称之为动作价值函数或 Q 值,用于评估不同卸载决策的优劣,从而引导代理选择最优策略. 当 $Q(s_t,a_1) > (s_t,a_2)$ 时,可以认为在状态 s_t 下,选择卸载动作 a_1 所能获得的期望回报比 a_2 大.

Q-learning 是一种基于时序差分学习的算法,其核心思想是通过迭代更新动作价值函数 $Q(s_t,a)$ 来逼近最优策略. 它广泛应用于各种决策问题,尤其适用于离散状态空间和动作空间的场景[23].

在 Q-learning 中需要维护一张 $M \times N$ 大小的 Q 表,其中 M 是所有的状态数量,N 动作空间的大小. 算法首先初始化 Q 表,通常将所有值设为 0 或随机值. 然后算法通过迭代过程进行学习. 具体骤如下:首先选择当前状态 s_t,并采用 ϵ - greedy 的探索策略选择动作 a;接着执行动作 a,将任务本地执行或者进行卸载,并根据环境的即时反馈获得回报 r,同时观察下一个状态 s_{t+1};根据公式(27)更新 Q 值,直到 Q 值收敛为止,此时终止算法.

$$Q(s_t,a) = (1-\alpha)Q(s_t,a) + \alpha(r + \gamma \max_a Q(s_{t+1},a')) \quad (27)$$

其中,r 是执行动作 a 的即时奖励,γ 是为折扣因子,表示未来回报的权重. α 是学习率,决定了旧的 Q 值在更新过程中对新值的影响程度. 该算法可以实现动态边缘网络中的自适应任务卸载决策.

表 2　不同类型任务的属性

Table 2　Attributes of different types of tasks

属　性	时延敏感型	计算密集型	综合型
执行时延(秒)	6	30	10
任务长度(KB)	150	800	400
所需资源(MIPS)	12000	600000	20000
数量分布(%)	30	30	40
生成速率(个/分钟)	40	20	20

表 3　异构设备的属性

Table 3　Attributes of heterogeneous devices

属性	移动设备	计算机	传感器
移动支持能力	√	×	×
任务生成能力	√	√	√
卸载服务支持	√	√	×
任务编排功能	√	√	√
区域分布(%)	35	25	40
计算能力	25000	100000	–

根据状态空间的定义,需要对所有状态 S_t 进行离散化处理,从而生成状态数据. 具体来说,ρ_{loc},$\bar{\rho}_{mec}$ 以及 $\bar{\rho}_{cloud}$ 表示利用率,数据的范围从 0% ~ 100%. 因此,记 0% ~ 20% 记作状态“1”,21% ~ 40% 记作状态“2”,依此类推. 状态 f_s 表示设备当前的计算能力,0 ~ 5000 记作状态“1”,5000 ~ 10000 记作状态“2”,10000 ~ 20000 记作状态“3”,20000 ~ 60000 记作状态“4”,60000 以上记作状态“5”. T_k^m 表示任务所需的计算资源,0 ~ 15000 记作状态“1”,15000 ~ 30000 记作状态“2”,30000 以上记作状态“3”. 最后,T_k^d 表示任务的截止期限,0 ~ 8 记作状态“1”,8 ~ 16 记作状态“2”,16 以上记作状态“3”. 由于动作空间 A_t 本身就是离散的,因此无需进一步处理. 以上离散化的依据来自于模拟任务的属性以及边缘设备的属性,如表 2、表 3 所示. 根据离散化的结果,状态空间总共有 2304 个不同的状态,而动作空间有 4 种不同的动作,即 $M = 2304$,$N = 4$. 边缘设备需要有存储 2304 × 4 表格大小的空间.

由于任务卸载问题具有较强的相似性,本文采用了多次训练得到的 Q 表进行初始化,这样可以有效减少代理在初期阶段的无效探索,从已有的知识中进行学习.

3.3　算法复杂度分析

本节将对所提出算法进行复杂度分析. 首先是动态信任评估算法,在算法中,设备 s_i 提供了反馈信息,在边缘网络中该类型设备总数量为 m,满足无线通信距离条件 Y 的资源节点的数量为 n. 从空间复杂度的角度来看,信任代理服务器需存储双维度的信任信息,全局信任信息以及设备负载状态信息. 信任代理服务器的空间复杂度为 $O(mn+n)$,近似为 $O(mn)$. 对于每个边缘设备,需要接收来自信任代理服务器的可行设备集合 $\{S\}$,并缓存在本地,该集合中包含 k 个元素. 因此,每个边缘设备的空间复杂度为 $O(k)$. 从时间复杂度的角度来看,设当前时间为 t,时间片数量为 $q = \frac{t}{\tau}$,在每个时间片上,需要以权重的方式累加所有反馈信任值,因此其时间复杂度为 $O(qmn)$,计算互动信任的时间复杂度为 $O(mn)$,而计算资源节点负载状态的时间复杂度为 $O(n)$. 综上,动态信任评估算法的时间复杂度近似为 $O(tmn)$.

在基于表格的 Q-learning 框架中,算法维护了一个大小为 $M \times N$ 的表格存储 Q 值. M 为状态空间的大小,即 $|S|$,N 为动作空间的大小,即 $|A|$. 因此空间复杂度为 $O(|S| \times |A|)$. 表格型的 Q-learning 算法在一次单步更新中,通过 Q 表访问并计算覆盖 Q 值的时间复杂度为 $O(1)$. 此外,对于离散化的有限状态-动作空间下的 Q-learning 算法,已从数学角度证明其是收敛的[24]. 收敛速度通常与状态空间的大小 $|S|$ 呈指数关系,且通过合理设置初始 Q 值,可以加速算法收敛的过程[25].

4　实验结果与分析

4.1　实验设置

为了验证本方案在复杂边缘环境中的性能,使用了基于改进版 PureEdgeSim[26] 进行了仿真实验. PureEdgeSim 构建于 CloudSim Plus 之上,既支持组件间的离散事件通信,又具备真实可信的边缘计算基础设施建模能力. 该平台采用高度模块化设计,可根据本文所述系统模型灵活替换各功能模块,从而模拟复杂的边缘场景.

为模拟移动边缘计算场景,采用如表 4 所列参数进行配

置. 仿真的总时长设置为 10 分钟,考虑实际环境中的障碍物(如墙壁),信号衰减与干扰,将无线通信有效距离设为 40 米;设备移动速度取 1.5m/s,接近行人平均速度. 为凸显 MEC 在低延迟,高吞吐量及能耗优化方面的优势,本地无线局域网与云计算中心的带宽分别为 100Mbps 和 20Mbps.

表 4 仿真参数汇总

Table 4 Summary of simulation parameters

参 数	取值
仿真持续时间	10 分钟
仿真区域面积	40000 平方米
D2D 无线通信传播距离	40 米
区域内设备最小数量	70 个
区域设备最大数量	200 个
设备移动速度	1.5 米/秒
本地局域网带宽	100Mbps
云计算中心带宽	20Mnps
发送每比特能耗	5×10^{-8}焦耳

通过将区域内设备数量从 70 增加至 200,来模拟从稀疏到密集的异构边缘网络,并获得实验结果. 在此过程中,设备的初始位置通过随机生成,并采用随机游走模型进行位置更新. 每个边缘设备会以一定速率随机生成计算任务,任务类型涵盖了边缘场景中用户对计算任务的多样化需求. 随着设备规模的扩大,生成的任务数量也随之增加. 此外,对于 Q-learning 算法的超参数设置:学习率 $a=0.4$ 和折扣因子 $\gamma=0.3$ 分别控制学习更新的速度与未来回报的权重.

4.2 结果分析

4.2.1 不同算法的平均等待对比

本实验比较了本地执行(Local),启发式算法(Greedy)与强化学习算法(Q-learning)算法在不同设备数量中的 $\bar{t}$ 表现.

实验结果如图 3 所示. 本地执行算法的平均等待时延为 0.17 ~ 0.26 秒之间,为 3 种算法中最低的. 这是因为本地执行算法只需考虑单个设备的计算资源,不需要通过网络进行资源调度,所以任务调度简单. 但是,本地计算无法通过卸载的方式来优化任务执行.

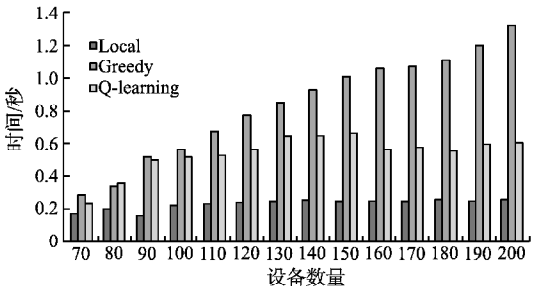

图 3 不同类型算法的平均等待时延

Fig. 3 Average waiting delay of different algorithms

Greedy 算法作为启发式算法,可快速找到近似最优解,实现简单,适用于小规模设备集群. 但是在大规模的设备场景下,Greedy 的搜索空间增大,耗费更多的调度时间. Q-learing 算法在设备数量为 70 时,平均等待时延为 0.23 秒;设备数量增加到 200 台设备时为 0.61 秒,其增长幅度明显低于 Greedy 算法. 实验表明,基于强化学习的 Q-learing 算法相比于启发式算法能够动态适应复杂环境,有效地减少了任务的平均等待时延.

4.2.2 不同任务生成速率下的设备负载分析

本实验将调节任务的生成速率,范围为 10 ~ 60 个任务每秒(TPS). 目的是评估在不同任务生成速率下,启发式算法以及强化学习算法在云端,边缘设备和终端设备上的负载分配性能以及任务丢失率的差异.

实验结果如图 4 所示. 随着任务生成速率的增加,云端,边端,终端的负载均呈现上升的趋势,同时任务丢失率增加显著上升,从 8.37% 增加至 40.71%. 在任务生成速率较大的时候,云端,边端,终端的平均负载都处于较高的水平,这反映了 Greedy 算法作为启发式方法,简单地卸载到资源最充足的节点上,因此无法考虑到长远的优化和全局信息,可能会导致某些设备负载过重,使任务丢失率迅速上升.

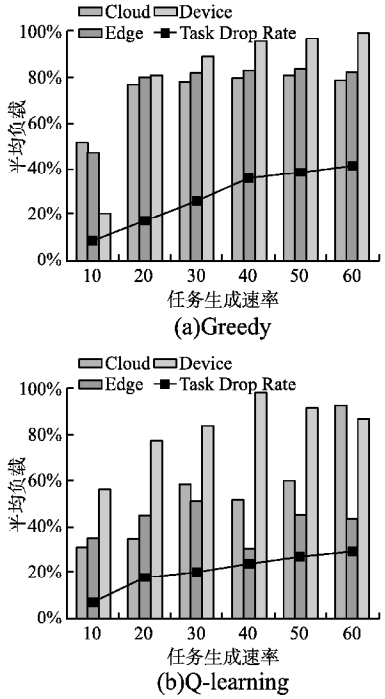

图 4 云-边-端层级负载随任务生成速率变化对比

Fig. 4 Comparative analysis of cloud-edge-device hierarchical load variations across task generation rates

与 Greedy 相比,Q-learning 的任务丢失率略低. 任务丢失率从 7.02% 在低任务生成速率时开始,逐渐上升至 28.70%,显示出任务丢失的增多趋势,但其增幅相对较小. 这表明 Q-learning 算法能够在每一步决策中考虑到设备的负载状态,并通过持续学习优化决策,避免了启发式决策的局限性. 尤其在任务生成速率较高时,Q-learning 通过调整任务卸载策略,可以有效降低任务丢失率. 实验表明,Q-learning 在高负载、复杂任务卸载场景下表现出更强的适应性,能够有效减少任务丢失.

4.2.3 不同算法在系统成本与任务成功率上的对比分析

本实验对不同算法在系统成本和任务成功率的表现进行对比,以评估其在不同设备规模中的性能. 对比算法如下:

1)启发式算法(Greedy),Greedy 算法在每次任务卸决策时,会评估所有可用节点的资源状况,并优先选择资源最充足的节点作为任务的执行目标.

2)基础 Q-learning 算法(Base Q-learning),该算法基于强化学习框架,通过试探和学习优化任务卸载决策,以奖励反馈更新 Q 表,逐步逼近最优策略. 基础 Q-learning 从随机初始化的 Q 表开始,通过交互不断提升卸载性能.

3)改进 Q-learning(Modified Q-learning),该算法通过使用训练好的 Q 表初始化. 相比基础 Q-learning,减少了试探次数,更快适应任务卸载场景的动态变化.

4)基于信任机制的改进 Q-learning 算法(Proposed),本文提出的算法在改进 Q-learning 的基础上,引入信任值机制,筛选可靠的资源节点供强化学习代理决策使用.

系统成本的实验结果如图 5(a)所示. 随着设备数量从 70 增加到 200,所有算法的任务卸载成本总体呈上升趋势. 这反映了设备规模增加对系统性能带来的影响.

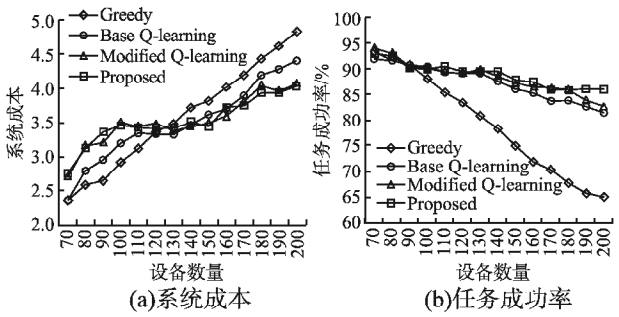

图 5　不同算法的系统性能对比

Fig. 5　Comparison of system performance of different algorithms

Greedy 算法在设备规模较小时性能最佳,因其优先选择空闲资源最多的设备进行卸载. 随着设备规模增加,其性能显著下降. 基础的 Q-learning 算法在小规模设备场景中性能较差,因其探索阶段决策开销较高,初始性能表现不佳. 但随着设备数量增加,其性能逐渐变好,体现强化学习在复杂场景中的适应性. 改进 Q-learning 通过经验初始化优化了性能,在设备数量超过 120 后优于基础 Q-learning. 本文提出的算法在大规模设备集群中(设备数量为 200)性能表现最优,对比启发式算法系统成本降低了 16.3%. 但是和改进 Q-learning 没有拉开差距,这是因为在计算系统成本时没有考虑失败的任务.

在任务成功率指标上,实验结果如图 5(b)所示. Greedy 算法在小规模环境中表现优异,但在大规模场景下不能满足任务截止期限,导致任务执行成功率快速下降. 基础 Q-learning 的成功率随设备数量增加而下降,但幅度小于 Greedy 算法,体现了强化学习算法在动态环境中的适应能力. 改进 Q-learning 表现更稳定,大多数情况下优于基础 Q-learning,尤其在设备数量较多时表现突出. 初始化 Q 表减少了初期无效探索,使其在大规模设备场景下仍然保持较高任务成功率. 而本文提出的算法在任务成功率上表现最优,尤其在大规模设备场景中,对比启发式算法提升了 32.1%. 这表明,通过信任值机制可以有效选择可靠的资源节点,减少了任务卸载失败的情况.

4.2.4　信任值评估性能对比实验

本实验在边缘计算场景中引入了 10%,20% 以及 30% 的恶意节点,以研究恶意节点对任务成功率的影响. 恶意节点的存在会破坏任务卸载过程,导致任务卸载失败. 对用户体验(QoE)产生显著的负面影响. 因此,研究恶意节点对任务卸载过程的影响对于提升系统的抗干扰能力具有重要意义.

实验结果如图 6(a)所示,随着恶意节点比例从 10% 增加至 30%,Greedy 算法的任务成功率迅速下降,尤其在设备数量较多时更明显. 例如,在 200 台设备时,任务成功率从 60.88% 降至 43.35%,降幅达 17.53%. 这表明恶意节点在大规模设备环境中对资源分配和任务成功率的干扰更加显著. 基础的 Q-learning 和改进 Q-learning 的实验结果如图 6(b)和图 6(c)所示,两者在恶意节点比例增加时任务成功率均呈现下降的趋势,但改进 Q-learning 表现优于基础 Q-learning,其优势来源于使用训练好的 Q 表进行初始化,从而减少初期试探并提升卸载效率.

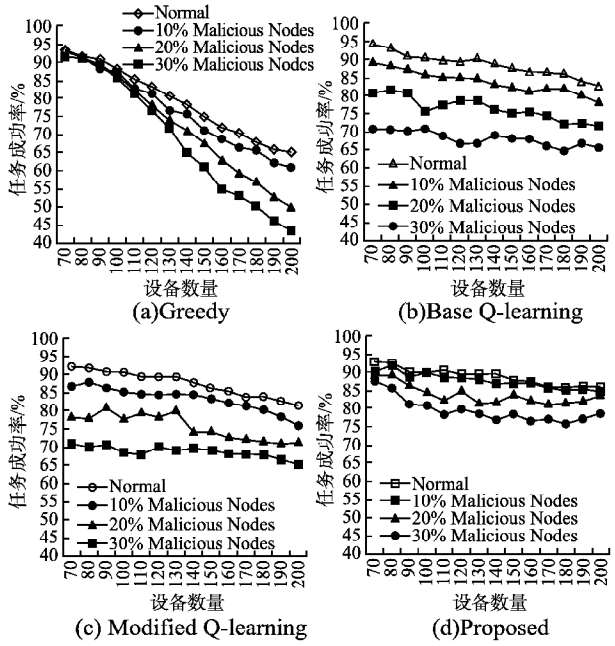

图 6　不同恶意节点比例下的任务卸载成功率比较

Fig. 6　Comparison of task success rate under different proportions malicious nodes

本文提出的卸载方案的实验结果如图 6(d)所示,在 30% 恶意节点条件下,任务成功率从 87.59% 降至 78.6%,降幅仅为 8.99%. 实验表明,在大规模设备集群中,提出方案的任务成功率为 78.6%,显著高于 Modified Q-learning 的 65.17% 和 Greedy 算法的 43.35%,分别优化了 20.61% 和 81.31%. 这证明了通过引入信任值机制,使任务优先分配给高可靠性资源节点,可有效降低恶意节点的干扰,提升任务卸载的成功率.

5　结束语

本文面向复杂场景中的任务卸载问题,设计了基于动态评估的任务卸载方案,该方案由两部分算法构成:一是动态信

任评估机制,利用包含时间衰减特性的双维度信任模型对边缘设备的可靠性进行量化,从而筛选出可靠的资源节点以支持任务卸载;二是将任务卸载问题建模为马尔可夫决策过程,提出了基于改进 Q-learning 的方法来优化任务卸载决策. 最后,通过仿真实验模拟了复杂的边缘场景,实验结果表明,所提出的卸载方案能够有效识别不可靠资源节点,并在任务成功率和系统成本两个指标中显著优于启发式算法. 同时,在大规模设备集群的环境中,该方案表现出较强的适应性,验证了其在实际应用中的潜力.

虽然本文取得了一定的研究进展,但是仍然有可以改进的地方. 首先基于表格的 Q-learning,在面对状态和动作空间急剧增大的场景时,容易出现维度灾难的问题,目前可以引入深度神经网络作为动作价值函数的近似器,代替传统的表格形式. 其次,本文聚焦于边缘网络中的任务卸载安全问题,未来研究可进一步在场景上拓宽,利用加密技术和隐私保护机制,来确保用户任务卸载的安全性与隐私性.

References:

[1] Schneider M, Haag F, Khalil A K, et al. Evaluation of communication technologies for distributed industrial control systems: concept and evaluation of 5G and WiFi 6[J]. Procedia CIRP, 2022, 107: 588-593.

[2] Cisco. Annual internet report (2018-2023) white paper[EB/OL]. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2023.

[3] Nguyen H, Nawara D, Kashef R. Connecting the indispensable roles of IoT and artificial intelligence in smart cities: a survey[J]. Journal of Information and Intelligence, 2024, 2(3): 261-285.

[4] Hu F, Deng Y, Saad W, et al. Cellular-connected wireless virtual reality: Requirements, challenges, and solutions[J]. IEEE Communications Magazine, 2020, 58(5): 105-111.

[5] Siriwardhana Y, Porambage P, Liyanage M, et al. A survey on mobile augmented reality with 5G mobile edge computing: architectures, applications, and technical aspects[J]. IEEE Communications Surveys & Tutorials, 2021, 23(2): 1160-1192.

[6] Garikapati D, Shetiya S S. Autonomous vehicles: evolution of artificial intelligence and the current industry landscape[J]. Big Data and Cognitive Computing, 2024, 8(4): 42, doi: 10.3390/bdcc8040042.

[7] Saeik F, Avgeris M, Spatharakis D, et al. Task offloading in edge and cloud computing: a survey on mathematical, artificial intelligence and control theory solutions[J]. Computer Networks, 2021, 195: 108177, doi: 10.1016/j.comnet.2021.108177.

[8] Islam A, Debnath A, Ghose M, et al. A survey on task offloading in multi-access edge computing[J]. Journal of Systems Architecture, 2021, 118: 102225, doi: 10.1016/j.sysarc.2021.102225.

[9] Ning Z, Dong P, Kong X, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things[J]. IEEE Internet of Things Journal, 2018, 6(3): 4804-4814.

[10] Du J, Zhao L, Feng J, et al. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee[J]. IEEE Transactions on Communications, 2017, 66(4): 1594-1608.

[11] Zhu C, Tao J, Pastor G, et al. Folo: latency and quality optimized task allocation in vehicular fog computing[J]. IEEE Internet of Things Journal, 2018, 6(3): 4150-4161.

[12] Ketykó I, Kecskés L, Nemes C, et al. Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing[C]//European Conference on Networks and Communications, 2016: 225-229.

[13] Guo H, Liu J, Zhang J. Computation offloading for multi-access mobile edge computing in ultra-dense networks[J]. IEEE Communications Magazine, 2018, 56(8): 14-19.

[14] Qu G, Wu H, Li R, et al. DMRO: a deep meta reinforcement learning-based task offloading framework for edge-cloud computing[J]. IEEE Transactions on Network and Service Management, 2021, 18(3): 3448-3459.

[15] Lu H, Gu C, Luo F, et al. Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning[J]. Future Generation Computer Systems, 2020, 102: 847-861, doi: 10.1016/j.future.2019.07.019.

[16] Zhu X, Luo Y, Liu A, et al. Multiagent deep reinforcement learning for vehicular computation offloading in IoT[J]. IEEE Internet of Things Journal, 2020, 8(12): 9763-9773.

[17] Jošilo S, Dán G. Wireless and computing resource allocation for selfish computation offloading in edge computing[C]//IEEE INFOCOM-IEEE Conference on Computer Communications, 2019: 2467-2475.

[18] Mao Y, Zhang J, Letaief K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices[J]. IEEE Journal on Selected Areas in Communications, 2016, 34(12): 3590-3605.

[19] Yuan J, Li X. A multi-source feedback based trust calculation mechanism for edge computing[C]//IEEE INFOCOM-IEEE Conference on Computer Communications Workshops, 2018: 819-824.

[20] Kong W, Li X, Hou L, et al. A reliable and efficient task offloading strategy based on multifeedback trust mechanism for IoT edge computing[J]. IEEE Internet of Things Journal, 2022, 9(15): 13927-13941.

[21] Aghapour Z, Sharifian S, Taheri H. Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments[J]. Computer Networks, 2023, 223: 109577, doi: 10.1016/j.comnet.2023.109577.

[22] Zhao X, Liu M, Li M. Task offloading strategy and scheduling optimization for internet of vehicles based on deep reinforcement learning[J]. Ad Hoc Networks, 2023, 147: 103193, doi: 10.1016/j.adhoc.2023.103193.

[23] Dab B, Aitsaadi N, Langar R. Q-learning algorithm for joint computation offloading and resource allocation in edge cloud[C]//IFIP/IEEE Symposium on Integrated Network and Service Management, 2019: 45-52.

[24] Evendar E, Mansour Y. Learning rates for Q-learning[J]. Journal of Machine Learning Research, 2003, 5: 1-25, doi: 10.1007/3-540-44581-1_39.

[25] Koenig S, Simmons R G. Complexity analysis of real-time reinforcement learning[C]//Association for the Advance of Artificial Intelligence, 1993: 99-105.

[26] Mechalikh C, Taktak H, Moussa F. PureEdgeSim: a simulation framework for performance evaluation of cloud, edge and mist computing environments[J]. Computer Science and Information Systems, 2021, 18(1): 43-66.