doi:10.12068/j.issn.1005－3026.2025.20230216

部分可观测环境中基于图强化的任务卸载与资源分配方法

代 钰[1]，景宗明[1]，杨 雷[2]，高 振[2]
（1.东北大学 软件学院，辽宁 沈阳 110169；2.东北大学 计算机科学与工程学院，辽宁 沈阳 110169）

摘 要：为了解决部分可观测环境中由于边缘服务器之间缺乏有效通信而导致的全局信息缺失问题，构建了基于图注意力机制的边缘服务器间沟通机制，将移动边缘计算（mobile edge computing，MEC）系统构建为图结构，使边缘服务器之间可以通过图中的边进行消息传递，进而间接得到MEC系统的全局状态信息．同时引入双注意力机制，使边缘服务器更多关注对策略优化更有用的通信消息，加快模型收敛速度并提高算法性能．仿真实验结果表明，与基线算法相比，本文所提出的算法可以有效降低任务完成时延与能耗，同时具有收敛速度快的优点．

关 键 词：移动边缘计算；深度强化学习；任务卸载；资源分配；消息通信
中图分类号：TP 393 **文献标志码**：A **文章编号**：1005-3026(2025)01-0009-10

A Graph Reinforcement-Based Approach to Task Offloading and Resource Allocation in Partially Observable Environment

DAI Yu[1]，*JING Zong-ming*[1]，*YANG Lei*[2]，*GAO Zhen*[2]
（1. School of Software，Northeastern University，Shenyang 110169，China；2. School of Computer Science & Engineering，Northeastern University，Shenyang 110169，China. Corresponding author：Dai Yu，E-mail：daiy@swc.neu.edu.cn）

Abstract：To address the issue of global information loss due to ineffective communication among edge servers in partially observable environment，an inter-edge server communication mechanism based on a graph attention mechanism is constructed，where the mobile edge computing（MEC）system is represented as a graph structure，allowing message passing between edge servers through the edges in the graph to indirectly obtain the global state information of the MEC system. The dual attention mechanism is introduced to enable agents to focus more on communication messages that are more useful for policy optimization，thereby accelerating the convergence speed of the model and improving algorithm performance. Simulation experimental results demonstrate that compared to baseline algorithms，the proposed algorithm effectively reduces task completion delay and energy consumption while exhibiting faster convergence speed.

Key words：mobile edge computing（MEC）；deep reinforcement learning；task offloading；resource allocation；message communication

近年来，随着手机、智能手表等移动设备的普及，出现了许多延迟敏感型和计算密集型应用，这也给移动设备在能耗与计算性能方面带来了更严峻的挑战．为了解决这个问题，学术界和产业界开始关注移动边缘计算（MEC）领域，通过将移动设备的任务卸载到边缘服务器，利用边缘服务器的计算资源，从而有效缓解移动设备的计算压力．一些研究人员提出采用启发式方法[1-3]解决此类问题，但这类方法只适用于任务大小、计算资源量都固定的MEC场景，这显然不能适应

收稿日期：2023-07-24
基金项目：国家重点研发计划项目（2021YFF0901205）．
作者简介：代 钰（1980—），女，辽宁沈阳人，东北大学副教授．

现实生活中的MEC场景.为此,一些研究提出使用基于值的深度强化学习(deep reinforcement learning,DRL)算法[4-5],通过维护Q函数来学习最优决策,还有一些研究选择使用基于策略的DRL算法[6-7],直接优化策略π.上述这些单智能体强化学习算法虽然在系统建模上规避了部分可观测问题,但在部署了多个边缘服务器的MEC系统中训练效果并不好,原因是每个智能体将其他智能体看作是环境中的一部分进行训练,其他智能体策略的变化导致环境变得非常不稳定,这并不利于强化学习模型的收敛,而且在现实环境中很难获取理想全局信息.

为了更好地解决移动边缘计算系统中的任务卸载和资源分配问题,一些研究提出使用多智能体深度强化学习(multi-agent deep reinforcement learning,MADRL)方法,多个智能体共享环境,智能体之间相互影响,共同实现最大化累积回报的目标[8-9].在MADRL中,训练和执行策略的方式有多种,通常根据训练和执行阶段是否集中化或分散化可以将MADRL的相关研究划分为3类:①基于分散式训练和执行的MADRL算法,例如IPPO算法.在训练过程中,每个智能体仅依赖于自己所观察到的局部状态和执行动作的反馈来更新自己的策略,而不考虑其他智能体的状态或行为.在执行阶段,智能体根据自己的局部观测和策略来选择动作.这类算法的主要缺陷在于难以处理复杂的协作或竞争关系.②基于集中化训练和执行的MADRL算法.这类算法在训练阶段,全局控制器能够获取所有智能体的状态和动作信息,共同学习一个全局最优策略.在执行阶段,所有动作的选择都是基于全局观测和策略来决定的.然而,这类算法的主要缺点在于计算成本较高,且不易于扩展.③基于集中式训练和分散式执行的MADRL算法,例如多智能体深度确定性策略梯度(multi-agent deep deterministic policy gradient,MADDPG),QMIX算法和COMA(counterfactual multi-agent)策略梯度算法.这类算法在训练阶段允许智能体共享全局信息,但在执行阶段,智能体仅根据其本地观测和策略进行动作选择,不再依赖全局信息.这类算法的优势在于,通过集中化训练,能够有效处理复杂的协作和竞争问题.同时,在执行阶段,这类算法具有较好的扩展性和较快的收敛速度.

尽管MADDPG和COMA算法已经取得了很有竞争力的优势.然而,在执行阶段,这类算法只能获得本地的观察结果,而不能了解环境的整体状况,这就会存在观测信息遗漏的可能性,导致算法陷入局部最优.因此,一些研究致力于采用加入循环神经网络(recurrent neural network,RNN)[10-11]的方法,使智能体能够记住最后一次通信时的信息,以此避免陷入局部最优.一些研究使用融合图神经网络(graph neural network,GNN)[12-13]的MADRL方法,在智能体之间构建有向图,使智能体可以交流他们的信息,如观测、意图或经验等,从而打破部分可观测和智能体之间通信受限的约束,让多个服务器协同工作以提供计算卸载服务,完成对任务完成情况和能耗的优化.但在较多边缘服务器的场景下,这些方法大都是无差别使用接收到的所有消息,没有根据消息的重要程度进行有必要的取舍,很容易出现有向图规模过大、训练时间长、训练结果难以收敛等问题.

本文将移动边缘计算系统中的任务卸载与资源分配问题建模为一个部分可观测马尔可夫过程,以基站作为强化学习过程中的智能体,给出了一种在部分可观测环境下基于MADDPG算法并借助图神经网络及双注意力机制改进的任务卸载与资源分配策略,借助双注意力机制缩小图的规模并增强智能体间消息通信的效果.仿真实验结果表明,在大多数情况下,本文所提出的算法可以有效降低任务完成时延与能耗,提高任务完成率,同时具有策略收敛速度快和更适应移动边缘网络的优点.

1 系统模型及环境建模

本文考虑了一个包含1个数据中心、多个边缘服务器以及多个移动用户设备的移动边缘网络,其中用户设备由$\mathcal{M}=\{1,2,\cdots,M\}$表示,边缘服务器由$\mathcal{N}=\{1,2,\cdots,N\}$表示.此外,本文将系统时间离散为多个回合,并且每个回合包含多个时间间隔,即$\mathcal{T}=\{1,2,\cdots,T\}$,$t$表示某一时间间隔.数据中心可以实现对移动边缘网络的全局观测,且认为数据中心具有足够的计算资源,因此忽略在数据中心进行数据处理而对边缘网络系统造成的影响.每个边缘服务器配备一个基站,作为边缘服务器对外访问的接口,同时也向数据中心汇报数据.用户设备通过无线信道连接到基站,用于发送卸载请求和传输计算任务等.每个用户设备维护1个任务队列,该队列用于存储需要在用

户设备本地计算的任务.移动边缘网络场景如图1所示.

每个用户设备在每一时隙$t\in\{1,2,\cdots,T\}$开始时产生一个计算密集且延迟敏感型的任务.第m个用户设备在时隙t产生的任务定义为$\boldsymbol{G}_m^t=[s_m^t,\ c_m^t,\ \tau_{m,\max}^t]$,其中$s_m^t$表示计算任务的数据量大小,$c_m^t$表示处理该计算任务所需要的CPU周期数,$\tau_{m,\max}^t$表示该任务的最大容忍时延.用户设备产生的这些任务可以卸载到不同的MEC服务器上执行,也可以在用户设备本地执行.每个时隙下,移动设备可以在系统所限定的范围内进行随机方向与距离的移动.

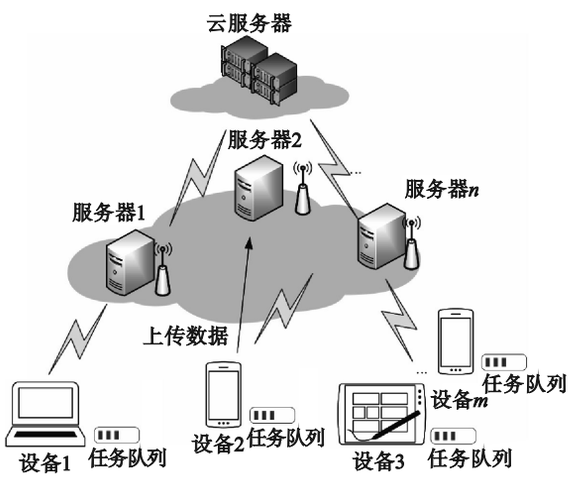

图1　移动边缘网络场景

Fig. 1　Mobile edge network scenario

1.1　通信模型

在时隙t内,如果边缘服务器在作卸载决策时选择将任务$\boldsymbol{G}_m^t$卸载到边缘服务器上执行,就需要将任务上传到边缘服务器n上.本文假设多个用户之间均分信道资源[14],并使用频分复用技术.根据香农公式[15],上行链路的传输速率可以定义为

$$r_{m,n}^t=\theta_m^t\mathrm{lb}\left(1+\frac{P_m\left(d_{m,n}\right)^{-\alpha}}{\sigma^2}\right).\tag{1}$$

其中:θ_m^t表示边缘服务器n对移动设备m分配的带宽;P_m为用户设备m的最大传输功率;$d_{m,n}$为用户设备m到边缘服务器n的距离,该距离可由GPS计算得出;α是路径损耗系数;$\left(d_{m,n}\right)^{-\alpha}$表示信道增益;$\sigma^2$表示用户设备$m$到边缘服务器$n$之间的最大噪声功率.

根据上行链路的传输速率$r_{m,n}^t$和任务$\boldsymbol{G}_m^t$的数据量大小s_m^t,任务的传输时延可以定义为

$$\tau_{m,\mathrm{trans}}^t=\frac{s_m^t}{r_{m,n}^t}.\tag{2}$$

根据传输时延$\tau_{m,\mathrm{trans}}^t$和最大传输功率$P_m$,任务的传输能耗可以定义为

$$E_{m,\mathrm{trans}}^t=P_m\tau_{m,\mathrm{trans}}^t.\tag{3}$$

由于计算任务返回的数据量通常比上传的数据量小得多[16],本文忽略计算结果返回的传输时延和能耗.

1.2　计算模型

本文设定边缘服务器接收的任务由服务器的计算单元分布式计算处理,边缘端计算时延可以定义为

$$\tau_{m,\mathrm{exe}}^t=\frac{c_m^t}{F_n w_m^t}.\tag{4}$$

其中:F_n表示边缘服务器总算力;w_m^t表示边缘服务器n分配给任务算力的比例.

根据式(2)和式(4)得到边缘端完成任务$\boldsymbol{G}_m^t$的总时延为

$$\tau_{m,\mathrm{offl}}^t=\tau_{m,\mathrm{trans}}^t+\tau_{m,\mathrm{exe}}^t.\tag{5}$$

边缘端完成任务$\boldsymbol{G}_m^t$的总时延$\tau_{m,\ \mathrm{offl}}^t$不应超过最大容忍时延$\tau_{m,\max}^t$.由于边缘服务器与用户设备相比拥有足够的计算资源,且通常由电网直接供电,本文不考虑边缘端的计算能耗.

本文定义用户设备m的计算能力为f_m,任务队列延时由q_m^t表示,本地执行任务所需要的计算时间$\tau_{m,\ \mathrm{loc}}^t$如式(6)所示,计算能耗$E_{m,\mathrm{loc}}^t$如式(7)所示.

$$\tau_{m,\mathrm{loc}}^t=\frac{c_m^t}{f_m}+q_m^t;\tag{6}$$

$$E_{m,\mathrm{loc}}^t=\kappa(f_m)^2c_m^t.\tag{7}$$

其中,κ为与设备芯片结构相关的能量系数[17].本地执行任务所需要的计算时间$\tau_{m,\ \mathrm{loc}}^t$也不应超过最大的任务容忍时延$\tau_{m,\max}^t$.

1.3　问题定义

由上述通信模型和计算模型可知,完成任务$\boldsymbol{G}_m^t$的总时延τ_m^t和总能耗E_m^t可以通过式(8)和式(9)计算.

$$\tau_m^t=[\![x_m^t=0]\!]\tau_{m,\ \mathrm{loc}}^t+[\![x_m^t=1]\!]\tau_{m,\mathrm{offl}}^t,\tag{8}$$

$$E_m^t=[\![x_m^t=0]\!]E_{m,\mathrm{loc}}^t+[\![x_m^t=1]\!]E_{m,\mathrm{trans}}^t.\tag{9}$$

其中:$x_m^t\in\{0,1\}$,$x_m^t=0$表示在用户设备本地执行计算任务,$x_m^t=1$表示卸载到邻近的边缘服务器上,根据欧几里得距离[18]判断哪个服务器为邻近服务器,如果邻近服务器负载超限,则重新分配$\boldsymbol{G}_m^t$给空闲服务器执行;$[\![x_m^t=0]\!]$和$[\![x_m^t=1]\!]$为艾佛森括号[19]的应用,如果方括号内的条件满足则为1,不满足则为0.

由于任务卸载与资源分配的目标在于尽可能地减小计算任务完成时延与能耗，所以本文将系统的优化目标确定为在保证一定任务成功率的同时最小化任务完成时延与能耗，故将时隙t下任务$\boldsymbol{G}_m^t$的效用定义为

$$U_m^t = xs_m^t - y\tau_m^t - zE_m^t. \tag{10}$$

其中：xs_m^t表示任务成功完成所获得的奖励；$y\tau_m^t$为时延成本；zE_m^t为能耗成本；x,y,z为归一化系数，用来消除不同单位的差异.

本文的目标函数定义为

$$\max_{x_m^t,\theta_m^t,w_m^t} \mathbb{E}\left[\sum_{m=1}^{M}\sum_{t=1}^{T}\gamma^{(t-1)}U_m^t\right]. \tag{11}$$

$$\text{s.t. } x_m^t \in \{0,1\}, \forall\, m \in \mathcal{M}; \tag{11a}$$

$$0 \leqslant \theta_m^t \leqslant 1, \forall\, m \in \mathcal{M}, \forall\, n \in \mathcal{N}; \tag{11b}$$

$$0 \leqslant \sum_{m=1}^{M} F_n w_m^t \leqslant F_n; \tag{11c}$$

$$\tau_m^t \leqslant \tau_{m,\max}^t, \forall\, m \in \mathcal{M}. \tag{11d}$$

其中，$\gamma \in [0,1]$表示折扣因子，用来平衡即时奖励和未来奖励的重要性，γ越大，意味着未来的奖励越重要.式(11a)表示任务可以在用户设备本地执行，也可以卸载到边缘服务器上执行，是任务卸载决策；式(11b)表示任务卸载到边缘服务器上执行时为其分配的计算资源比例，是资源分配决策；式(11c)表示服务器为所有计算任务分配的算力不能超过自身总算力；式(11d)表示保证任务的完成时间满足此任务的最大容忍时延.

1.4 基于马尔可夫决策过程的问题建模

为了利用强化学习解决任务卸载和资源分配问题，本文将此问题建模为多智能体随机博弈过程，它可以看作是马尔可夫决策过程在多智能体环境中的扩展.本文将边缘服务器视为智能体，多智能体随机博弈过程的状态空间、动作空间和奖励函数定义如下：

状态空间(state space)：在时隙t，边缘服务器n从MEC环境得到本地观测$\boldsymbol{o}_n^t=[W_n^t, F_n^t, f_m^t, \boldsymbol{G}_m^t]$，包括服务器可用的带宽资源$W_n^t$和计算资源$F_n^t$、用户设备的计算资源$f_m^t$、任务相关信息$\boldsymbol{G}_m^t$.此外，本文将所有边缘服务器的联合状态定义为$\boldsymbol{o}^t=\{\boldsymbol{o}_1^t, \boldsymbol{o}_2^t, \cdots, \boldsymbol{o}_N^t\}$.

动作空间(action space)：边缘服务器负责决定每个时隙中不同用户设备的任务卸载策略和资源分配策略，进一步，本文将边缘服务器的动作定义为$\boldsymbol{a}_n^t=[x_1^t, x_2^t, \cdots, x_M^t, \theta_1^t, \theta_2^t, \cdots, \theta_M^t, w_1^t, w_2^t, \cdots, w_M^t]$. x_m^t表示任务卸载决策；θ_m^t表示带宽资源分配决策；w_m^t表示计算资源分配决策，$m=1,2,\cdots,M$.此外，本文定义所有边缘服务器的联合动作为$\boldsymbol{a}^t=\{\boldsymbol{a}_1^t, \boldsymbol{a}_2^t, \cdots, \boldsymbol{a}_N^t\}$.

奖励函数(reward function)：将式(11)定义的目标函数作为奖励函数r^t，目的在于使系统奖励最大化.

2 本文算法介绍

为了解决上文中所提到的多智能体MDP问题，考虑到智能体之间消息通信，本文引入图神经网络与双注意力机制，提出了基于MADDPG和图注意力(MADDPG and graph attention，MAGA)的任务卸载与资源分配算法.MAGA算法的网络结构如图2所示.

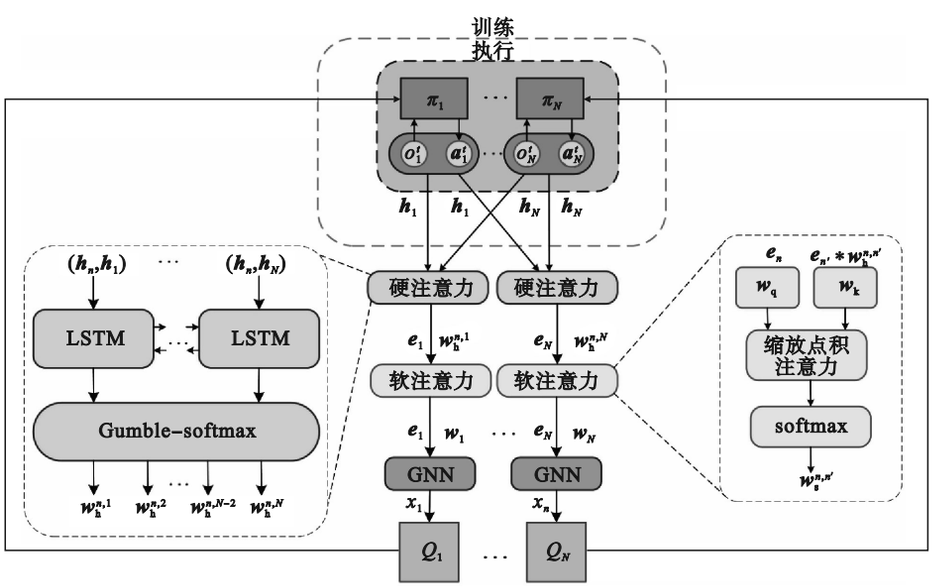

图2 MAGA算法的网络结构

Fig. 2 Network structure of MAGA algorithm

MAGA算法将边缘服务器之间的关系构建为一个图$G=(v,e)$,由节点集v和边集e组成,其中每个节点表示一个边缘服务器,每条边表示两个边缘服务器之间的交互关系,初始状态下默认每个边缘服务器之间都存在交互关系,即所有节点都成对连接.考虑1个部分可观测环境,在每个时间间隔t,每个边缘服务器n得到一个局部观测$\boldsymbol{o}_n^t$,通过多层感知机(multilayer perceptron,MLP)网络编码为特征向量$\boldsymbol{h}_n$,然后使用$\boldsymbol{h}_n$通过注意力机制来学习边缘服务器之间的关系.需要注意的是,边缘服务器之间的关系需要通过硬注意力和软注意力机制得出.

定义1　硬注意力是指选择输入序列某一个位置上的信息,这迫使模型只关注重要元素,而忽略其他元素.然而硬注意力机制是基于采样来选择元素的,所以是不可微的,不能通过端到端反向传播直接学习注意力权重.

定义2　软注意力计算的是元素的重要性分布,在选择信息的时候,不是从N个信息中只选择一个,而是计算N个输入信息的加权平均后,再输入到神经网络中计算.由于软注意力机制是可微的,模型能够计算出注意力权重的梯度,并逐步调整这些权重来优化结果,从而通过端到端训练方式实现更好的表现.由于元素的重要性分布通常可以转化为[0,1]连续分布问题,所以处理过程中通常使用softmax激活函数,然而该函数通常会将非零概率分配给不相关的元素,这削弱了对真正重要元素的关注.

MAGA算法使用硬注意力模型输出独热编码向量得到图G中2个节点之间的边是否存在,保留存在交互关系的边得到交互子图,以缩小图的规模,减小训练复杂度.此外,由于交互关系的重要性不同,训练了一个软注意力机制模型来学习每条边的权重.通过这种方式,可以得到边缘服务器n的子图G_n,其中边缘服务器n仅与需要交互的边缘服务器存在边的连接,边的权重描述了交互关系的重要程度,利用GNN获取子图G_n的向量表示.加入双注意力机制网络后基于MADDPG算法的MAGA算法结构如图2所示.

在硬注意力网络层,MAGA算法将边缘服务器n,n'的嵌入向量合并为特征$(\boldsymbol{h}_n,\boldsymbol{h}_{n'})$,其中$n'\in\{1,2,\cdots,N\}$,且$n\neq n'$,将特征$(\boldsymbol{h}_n,\boldsymbol{h}_{n'})$输入到长短期记忆(long short-term memory,LSTM)网络模型中得到二进制输出权重0或1,其中0表示两个边缘服务器之间不存在交互边,1表示存在交互边.传统LSTM的输出仅取决于当前时隙和前一时隙的输入,而忽略了后一时隙的输入信息.为了避免这种情况,本文采用了双向长短期记忆网络(bi-directional long short-term memory,Bi-LSTM))模型[20],即两个独立的LSTM模型分别从头尾两端开始计算,每次将前向LSTM与后向LSTM的输出叠加进行硬注意力权重计算.

由于硬注意力机制是基于采样来选择元素的,无法进行梯度反向传播,所以本文采用Gumble-softmax[21]解决此问题.最终,通过硬注意力网络层得到的权重为

$$\boldsymbol{w}_{\mathrm{h}}^{n,n'}=\mathrm{gum}\Big(f\Big(\mathrm{LSTM}\big(\boldsymbol{h}_n,\boldsymbol{h}_{n'}\big)\Big)\Big). \tag{12}$$

其中:gum(·)为Gumble-softmax函数;$f(\cdot)$表示一个全连接层;LSTM(·)表示使用了LSTM模型.

在软注意力网络层,采用基础的注意力机制处理硬注意力网络层保留下来的边,为这些边确定重要性权重.将边缘服务器n的本地观测$\boldsymbol{o}_n^t$和动作$\boldsymbol{a}_n^t$输入到单层MLP网络,通过嵌入函数$\boldsymbol{e}_n=g_n(\boldsymbol{o}_n^t,\boldsymbol{a}_n^t)$计算得到嵌入表达$\boldsymbol{e}_n$,其中$g_n$为一个MLP.通过权重矩阵$\boldsymbol{w}_{\mathrm{k}}$将嵌入表达$\boldsymbol{e}_n$转化为键向量,通过权重矩阵$\boldsymbol{w}_{\mathrm{q}}$将嵌入表达$\boldsymbol{e}_n$转化为查询向量,使用基于点积模型的打分函数进行相似性打分,然后将这些分数通过softmax函数归一化.最终,通过软注意力网络层得到的权重表示为

$$\boldsymbol{w}_{\mathrm{s}}^{n,n'}\propto\exp\left(\boldsymbol{e}_{n'}^{\mathrm{T}}\boldsymbol{w}_{\mathrm{k}}^{\mathrm{T}}\boldsymbol{w}_{\mathrm{q}}\boldsymbol{e}_n\boldsymbol{w}_{\mathrm{h}}^{n,n'}\right). \tag{13}$$

在训练过程中,为每一个边缘服务器计算一个动作价值函数.边缘服务器n的动作价值函数为

$$Q_n\big(\boldsymbol{o}^t,\boldsymbol{a}^t\big)=X_n\Big(Z_n\big(\boldsymbol{o}_n^t,\boldsymbol{a}_n^t\big),x_n\Big). \tag{14}$$

其中:$\boldsymbol{o}^t=\{\boldsymbol{o}_1^t,\boldsymbol{o}_2^t,\cdots,\boldsymbol{o}_N^t\}$表示所有智能体的全局观测信息;$\boldsymbol{a}^t=\{\boldsymbol{a}_1^t,\boldsymbol{a}_2^t,\cdots,\boldsymbol{a}_N^t\}$表示所有智能体的动作信息;$X_n$为一个两层的MLP网络;$Z_n$为一个单层的MLP网络;$x_n$表示所有其他边缘服务器对当前边缘服务器$n$的贡献,可使用具有强大编码能力的GNN得到.为了方便,本文将其他智能体的贡献进行加权求和得到x_n:

$$x_n=\sum\nolimits_{n'\neq n}\boldsymbol{w}_{n'}v_{n'}=\sum\nolimits_{n'\neq n}\boldsymbol{w}_{n'}h\Big(\boldsymbol{V}g_n\big(\boldsymbol{o}_{n'}^t,\boldsymbol{a}_{n'}^t\big)\Big). \tag{15}$$

式中:$\boldsymbol{w}_{n'}$为注意力权重;$v_{n'}$为边缘服务器n的嵌入表示,通过一个嵌入函数编码,然后通过共享矩阵$\boldsymbol{V}$进行变换;$h(\cdot)$是逐元素非线性函数.

Citic网络输出Q值后,通过策略梯度更新Actor网络,策略梯度可表示为式(16).通过最小化损失函数更新Citic网络,损失函数可表示为式

(17).最后利用软更新操作更新目标网络相关的参数.

$$\nabla_{\theta_n} J(\boldsymbol{\mu}_n)=$$
$$\mathbb{E}_{\boldsymbol{o}^t\sim\mathcal{D},\boldsymbol{a}^t\sim\pi}\left[\nabla_{\theta_n}\boldsymbol{\mu}_n(\boldsymbol{a}_n^t|\boldsymbol{o}_n^t)\nabla_{\boldsymbol{a}_n^t}Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^t,\boldsymbol{a}^t)|_{\boldsymbol{a}_n^t=\boldsymbol{\mu}_n(\boldsymbol{o}_n^t)}\right], \quad (16)$$
$$L(\theta_n)=\mathbb{E}_{\boldsymbol{o}^t,\boldsymbol{a}^t,r^t,\boldsymbol{o}^{t+1}}[(Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^t,\boldsymbol{a}^t)-y)^2]. \quad (17)$$

其中：$y=r^t(\boldsymbol{o}^t,\boldsymbol{a}^t)+\gamma Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^{t+1},\boldsymbol{a}^{t+1})|_{\boldsymbol{a}_n^{t+1}=\boldsymbol{\mu}_n(\boldsymbol{o}_n^{t+1})}$；$\nabla_{\theta_n}J(\boldsymbol{\mu}_n)$表示关于参数$\theta_n$的目标函数$J(\boldsymbol{\mu}_n)$的梯度,目标函数$J$通常表示期望的回报或奖励,$\boldsymbol{\mu}_n$是策略函数,用来选择动作;$\mathbb{E}_{\boldsymbol{o}^t\sim\mathcal{D},\boldsymbol{a}^t\sim\pi}$为期望运算符,表示对来自数据集$\mathcal{D}$中的状态$\boldsymbol{o}^t$和从策略$\pi$中采样的动作$\boldsymbol{a}^t$的期望值,换句话说,这是在样本数据上计算平均值;$\nabla_{\theta_n}\boldsymbol{\mu}_n(\boldsymbol{a}_n^t|\boldsymbol{o}_n^t)$表示策略$\boldsymbol{\mu}_n$对参数$\theta_n$的梯度,$\boldsymbol{\mu}_n(\boldsymbol{a}_n^t|\boldsymbol{o}_n^t)$是基于观测$\boldsymbol{o}_n^t$选择动作$\boldsymbol{a}_n^t$的概率;$\nabla_{\boldsymbol{a}_n^t}Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^t,\boldsymbol{a}^t)$表示$Q$函数对动作$\boldsymbol{a}^t$的梯度,$Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^t,\boldsymbol{a}^t)$是在状态$\boldsymbol{o}^t$和动作$\boldsymbol{a}^t$下的$Q$值函数(也称为动作-值函数),表示采取该动作后的预期回报;$\boldsymbol{a}_n^t=\boldsymbol{\mu}_n(\boldsymbol{o}_n^t)$表示在观测$\boldsymbol{o}_n^t$下,策略$\boldsymbol{\mu}_n$所选择的动作$\boldsymbol{a}_n^t$;$L(\theta_n)$表示损失函数,用于更新参数$\theta_n$以最小化预测的值$Q$和目标$y$之间的差距;$\mathbb{E}_{\boldsymbol{o}^t,\boldsymbol{a}^t,r^t,\boldsymbol{o}^{t+1}}$表示对状态$\boldsymbol{o}^t$、动作$\boldsymbol{a}^t$、奖励$r^t$和下一状态$\boldsymbol{o}^{t+1}$进行期望运算;$Q_n^{\boldsymbol{\mu}}(\boldsymbol{o}^t,\boldsymbol{a}^t)-y$表示损失项,表示当前$Q$值和目标值$y$之间的平方误差.

综上所述,加入图注意力机制的MADDPG算法训练流程见算法1.

算法1:加入图注意力机制的MADDPG算法

1. 随机初始化Critic网络和Actor网络
2. 初始化目标Critic网络和目标Actor网络
3. 初始化经验回放池$\mathcal{D}$
4. 初始化Actor和Critic网络参数、折扣因子γ、最大训练回合数X、每个回合的最长步数T
5. for $j=1$ to X do
6. 初始化每个边缘服务器的状态$\boldsymbol{o}_n^t$,全局观测信息o^t
7. for $t=1$ to T do
8. 边缘服务器n的局部观测信息$\boldsymbol{o}_n^t$编码为特征向量$\boldsymbol{h}_n$
9. 根据式(12)计算硬注意力层输出$\boldsymbol{w}_{\mathrm{h}}^{n,n'}$
10. 根据式(13)计算软注意力层输出$\boldsymbol{w}_{\mathrm{s}}^{n,n'}$
11. 根据式(15)计算智能体间贡献度x_n
12. 边缘服务器选择动作:$\boldsymbol{a}_n^t=\boldsymbol{\mu}_n(\boldsymbol{h}_n,x_n)$
13. 执行联合动作$\boldsymbol{a}^t=\{\boldsymbol{a}_1^t,\boldsymbol{a}_2^t,\cdots,\boldsymbol{a}_N^t\}$并获得奖励$r^t$和下一状态$\boldsymbol{o}^{t+1}$
14. 将元组$(\boldsymbol{o}^t,\boldsymbol{a}^t,r^t,\boldsymbol{o}^{t+1})$存入经验回放池$\mathcal{D}$
15. $\boldsymbol{o}^t\leftarrow\boldsymbol{o}^{t+1}$
16. for $n=1$ to N do
17. 从$\mathcal{D}$中任抽S个样本$(\boldsymbol{o}^t,\boldsymbol{a}^t,r^t,\boldsymbol{o}^{t+1})$
18. 根据式(14)计算$Q_n(\boldsymbol{o}^t,\boldsymbol{a}^t)$
19. 根据式(17)更新Critic网络
20. 根据式(16)更新Actor网络
21. end for
22. end for
23. end for

3 实验与性能评估

与文献[22-23]类似,本文考虑一个具有4~8个边缘服务器和10~60个移动用户设备的移动边缘网络环境,移动用户随机分布在基站周围.时间间隔设置为1 s,即用户设备每秒产生一个计算密集且延迟敏感型任务.将任务大小设定为2~5 Mb,可容忍时延为200 ms,所需CPU计算能力为0.2×10^9~1×10^9 Hz.参考文献[24-25],用户设备传输功率设定为1 W,本地CPU频率为1 GHz.服务器CPU频率设定为20 GHz,传输功率谱密度为-174 dBm/Hz,传输带宽为10 MHz.

3.1 神经网络参数设置

为了确定Actor网络和Critic网络最佳的学习率,图3为不同学习率对系统奖励的影响.本文定义l_{A}为Actor网络的学习率,l_{C}为Critic网络的学习率.从图3中可以看出,当$l_{\mathrm{A}}=0.01, l_{\mathrm{C}}=0.1$或$l_{\mathrm{A}}=0.001, l_{\mathrm{C}}=0.1$时,MAGA算法是可以收敛的,但由于$l_{\mathrm{A}}=0.01, l_{\mathrm{C}}=0.1$这组学习率设置过大,导致神经网络使用过大的更新步长,使算法在局部最优解处收敛.当$l_{\mathrm{A}}=0.000\,1, l_{\mathrm{C}}=0.001$时,算法并不能收敛,这是因为选取的学习率过低,神经网络更新速度过慢,此时需要训练更多的回合数才能使算法收敛.所以本文最后将Actor网络的学习率设置为0.001,将Critic网络的学习率设置为0.01.

在1.3节中提到,γ为决定未来奖励的折扣因子,取值在0~1之间,用来平衡即时奖励和未来奖励的重要性,γ越大,意味着未来的奖励越重要.为了确定折扣因子的取值,图4为不同折扣因子对系统奖励的影响.从图4中可以看出,当折扣因

子为0.95时,MAGA算法具有最好的奖励表现.这是因为当$\gamma=0.4$时,过低的折扣因子使智能体优先考虑近期奖励,忽视远期回报;当$\gamma=0.99$时,很高的折扣因子使智能体过分考虑行为对MEC系统的长期影响,这对提高算法训练的最终性能是不利的.因此最终将折扣因子γ的取值定为0.95.

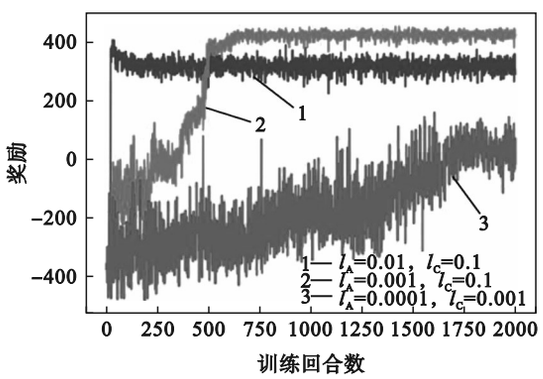

图3　不同学习率对系统奖励的影响

Fig. 3　Impact of different learning rates on system rewards

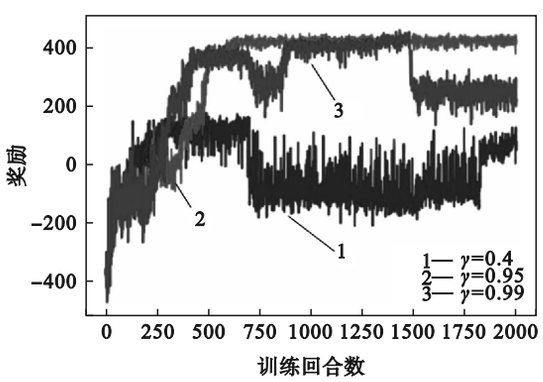

图4　不同折扣因子对系统奖励的影响

Fig. 4　Impact of different discount factors on system rewards

3.2　MAGA算法实验结果与分析

为了证明本文提出的任务卸载与资源分配方案有效,还将MAGA算法与下面2种不同的计算卸载方案进行了对比.

随机卸载策略(random offloading to edge scheme,ROES):所有任务随机分配到边缘服务器上或是用户设备本地执行.

基于MADDPG算法的策略:使用MADDPG基本算法,智能体之间不进行消息通信,执行动作时依靠局部观测完成任务卸载与资源分配.

只包含软注意力网络层的MAGA算法:将MAGA算法中的硬注意力网络层去除,即考虑图结构中所有的边,只利用软注意力网络层确定所有边的权重,以验证硬注意力网络层在MAGA算法中的作用,以下简称no-HardAtt.

实验共训练2 000个回合,每回合的最大迭代步数设置为100.由于奖励值是通过加权和的方式进行计算,因此奖励值是一个无量纲指标.从图5中可以看出,MAGA算法在获得系统奖励方面优于MADDPG算法,这是因为将MEC系统建模为图结构后,边缘服务器可以感知到其他边缘服务器的决策和相对应用户设备的状态信息,多个边缘服务器协同作出可以使系统奖励更高的决策.此外,图的构建使每个智能体获得的信息更多,这并不利于模型的收敛,本文加入软硬注意力网络结构以尽可能提取有利信息,但软硬注意力网络的加入又会使MAGA算法的网络结构比MADDPG算法更加复杂,所以MAGA算法的收敛速度会比MADDPG算法慢一点,波动比MADDPG算法大一点.

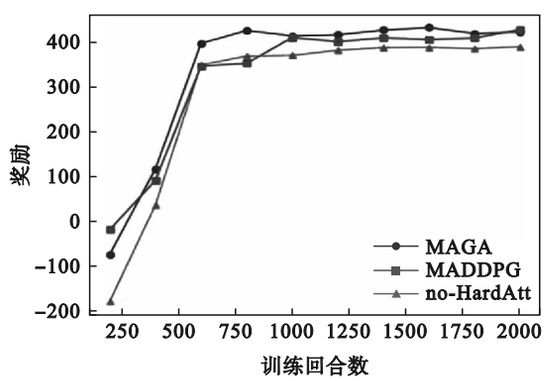

图5　不同算法的系统奖励

Fig. 5　System rewards of different algorithms

为了证明MAGA算法中的硬注意力网络层可以用于断开不存在通信关系的智能体之间的边,从而减小图结构的规模,简化训练难度,提升训练速度,本文将no-HardAtt算法与MAGA算法进行对比.从图5中可以看出,在迭代训练600个回合左右时,MAGA算法所获得的系统奖励趋于稳定,而no-HardAtt算法至少需要训练1 000个回合才能趋于稳定.这是因为去除硬注意力网络层后,每个边缘服务器之间的边都会被考虑在内,对于那些不存在交互关系的边缘服务器,软注意力网络层也会为其分配一定的权重,作为边缘服务器消息通信中的部分信息.但在真实的MEC系统场景中,一个边缘服务器并不需要与整个MEC系统中的其他所有边缘服务器都进行消息通信,往往只需要与周围几个邻近的边缘服务器通信即可.综上所述,与no-HardAtt算法相比,加

入硬注意力网络层的MAGA算法收敛速度更快且更具有应用价值.

图6展示了增加训练回合数对平均任务完成时间成本的影响.平均任务完成时间成本包括在最大容忍时延内完成任务的时延和未在最大容忍时延内完成任务所消耗的时间成本,降低平均任务完成时间成本可以促使计算任务更快地被完成,从而提高用户服务质量.从图6中可以看出,MAGA算法的平均任务完成时间成本最低,在5.52 s左右.ROES的平均任务完成时间成本最高,这是因为所有计算任务被随机分配到边缘服务器上或是用户设备本地执行,随着任务量的增大,这在一定程度上导致一些任务不能被成功卸载,产生了大量的未在最大容忍时延内完成任务所消耗的时间成本.

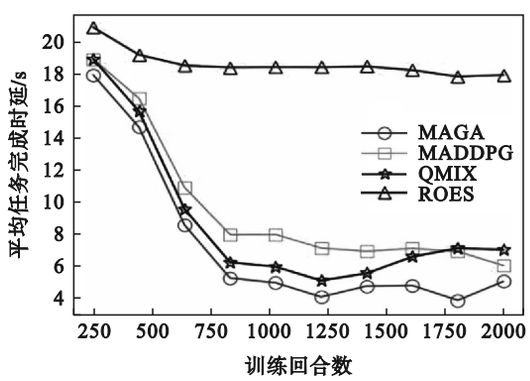

图6 不同算法的平均任务完成时延

Fig. 6 Average task completion delay of different algorithms

图7为不同算法在迭代训练过程中的平均能耗.由于边缘服务器可以一直处于供电状态,所以本文只考虑用户设备能耗.从图7中可以看出,随着训练回合数的增加,MADDPG算法和MAGA算法都可以减少完成计算任务的能耗.这是因为本地执行计算任务的用户设备能耗与用户设备CPU频率的平方成正比,是远大于将任务传输到边缘服务器上执行所花费的传输能耗的,MADDPG算法和MAGA算法通过确定最优的卸载决策,控制了选择在本地任务执行的比例,从而让更多的计算任务得以卸载到边缘服务器上执行.随机卸载策略的卸载动作是随机的,导致了用户设备平均能耗无法收敛.

最后进行消融实验,分析双注意力机制网络结构和图神经网络的有效性.其中MAGA表示拥有两者,no-Att表示没有使用双注意力机制网络缩小图的规模,no-Graph表示没有将MEC系统模型构建为图结构,而是直接在MADDPG的基础上融入注意力机制模块.

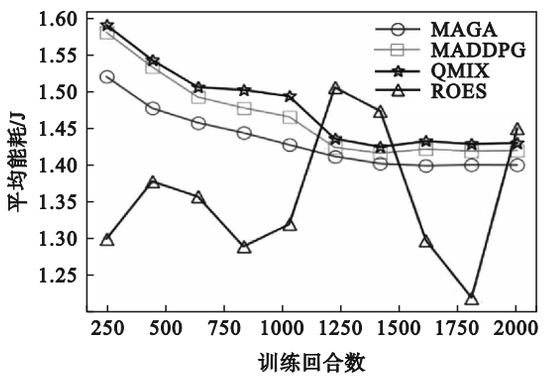

图7 不同算法的平均能耗

Fig. 7 Average energy consumption of different algorithms

图8展示了三种算法,在包含20个移动用户设备,4个边缘服务器的MEC系统仿真环境下的收敛性.从图8中可以看出,没有添加双注意力网络的no-Att算法无法收敛,而其他两个算法均能收敛.这是因为将MEC系统建模为图做消息通信会增加一部分图结构信息,复杂的数据结构会增大模型的训练复杂度,从而使模型无法收敛.而双注意力机制有效删减了图结构信息中的无效信息,提高了智能体在图结构下的通信质量.从图中可以看出,no-Graph算法的收敛要早于MAGA算法.这是由于no-Graph算法下的智能体通信不够充分,因此更容易陷入局部最优.相比之下,MAGA算法所获得的系统奖励比no-Graph算法更高,这说明图神经网络的加入增强了智能体间的交互质量,间接得到了更有利于决策的全局观测信息,有助于智能体之间协同找到更好的任务卸载与资源分配策略.

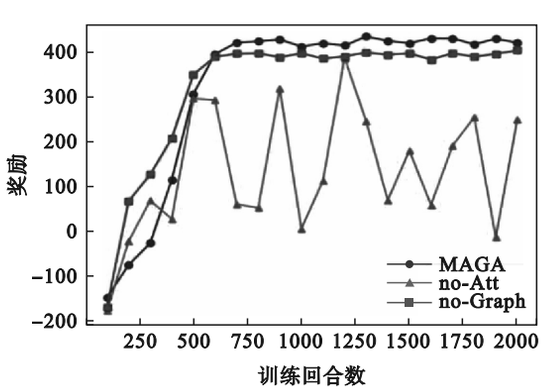

图8 三种算法的收敛性

Fig. 8 Convergence of the three algorithms

本文探究了移动设备的数量对于算法时延的影响,见图9.从图9中可以看出,本文提出的算法实现了最低的任务执行时延.例如,相对于其他的算法,本文提出的算法在时延方面降低了10%以上.这是因为本文将MEC系统构建为图结构,使边缘服务器之间可以通过图中的边进行消息传递,从而感知MEC系统的全局状态信息,最终提高算法性能.

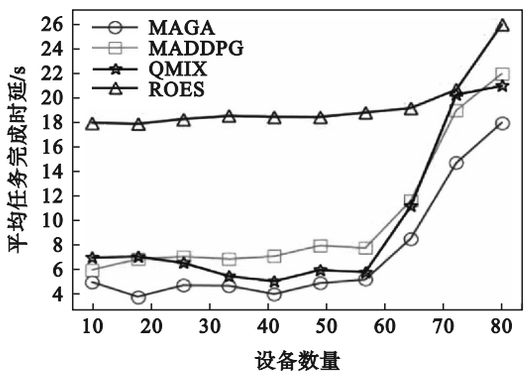

图9　移动设备的数量对于算法时延的影响

Fig. 9　Impact of the number of mobile devices on algorithm delay

4　结　　语

本文研究了在移动边缘计算环境下,使用多智能体强化学习进行任务卸载与资源分配算法时环境中存在的部分可观测问题,提出了一种基于图神经网络与双注意力机制的任务卸载与资源分配算法,构建了基于图注意力的智能体间沟通机制.加入双注意力机制后的图注意力机制能让边缘服务器使用图中的边进行有选择性的消息传递,进而更有效地间接得到MEC系统的全局状态信息,最终解决部分可观测问题.实验证明本文提出的MAGA算法可以有效收敛,与对比算法相比,实现了显著的性能提升.本文所使用的系统环境并未考虑移动设备之间的异构性与可能存在的分布特点,之后针对模拟环境仿真度欠缺的问题进行环境和算法的优化与调整.

参考文献:

[1] Chen C, Chen L L, Liu L, et al.Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks[J].*IEEE Access*, 2020, 8: 18863-18873.

[2] Zhao L, Zhang E C, Wan S H, et al. MESON: a mobility-aware dependent task offloading scheme for urban vehicular edge computing[J].*IEEE Transactions on Mobile Computing*, 2023, 23(5): 4259-4272.

[3] Pu L J, Chen X, Mao G Q, et al.Chimera: an energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications[J]. *IEEE Internet of Things Journal*, 2019, 6(1): 84-99.

[4] Feng L, Zhou Y, Liu T, et al.Energy-efficient offloading for mission-critical IoT services using EVT-embedded intelligent learning[J]. *IEEE Transactions on Green Communications and Networking*, 2021, 5(3): 1179-1190.

[5] Qian Y C, Wang R, Wu J, et al. Reinforcement learning-based optimal computing and caching in mobile edge network[J]. *IEEE Journal on Selected Areas in Communications*, 2020, 38(10): 2343-2355.

[6] Shi J M, Du J, Shen Y, et al. DRL-based V2V computation offloading for blockchain-enabled vehicular networks[J]. *IEEE Transactions on Mobile Computing*, 2022, 22(7): 3882-3897.

[7] Chen J, Xing H L, Xiao Z W, et al.A DRL agent for jointly optimizing computation offloading and resource allocation in MEC[J].*IEEE Internet of Things Journal*, 2021, 8(24): 17508-17524.

[8] Zhao N, Ye Z Y, Pei Y Y, et al. Multi-agent deep reinforcement learning for task offloading in UAV-assisted mobile edge computing[J].*IEEE Transactions on Wireless Communications*, 2022, 21(9): 6949-6960.

[9] Tang L, Du Y C, Liu Q H, et al. Digital-twin-assisted resource allocation for network slicing in industry 4.0 and beyond using distributed deep reinforcement learning[J]. *IEEE Internet of Things Journal*, 2023, 10(19): 16989-17006.

[10] Ye Z H, Wang K, Chen Y N, et al.Multi-UAV navigation for partially observable communication coverage by graph reinforcement learning[J]. *IEEE Transactions on Mobile Computing*, 2022, 22(7): 4056-4069.

[11] Tang M, Wong V W S.Deep reinforcement learning for task offloading in mobile edge computing systems[J]. *IEEE Transactions on Mobile Computing*, 2020, 21(6): 1985-1997.

[12] Gao Z, Yang L, Dai Y. Fast adaptive task offloading and resource allocation in large-scale MEC systems via multiagent graph reinforcement learning[J].*IEEE Internet of Things Journal*, 2023, 11(1): 758-776.

[13] Yan J, Bi S Z, Zhang Y J A. Offloading and resource allocation with general task graph in mobile edge computing: a deep reinforcement learning approach[J]. *IEEE Transactions on Wireless Communications*, 2020, 19(8): 5404-5419.

[14] Zhu X Y, Luo Y Y, Liu A F, et al. Multiagent deep reinforcement learning for vehicular computation offloading in IoT[J].*IEEE Internet of Things Journal*, 2021, 8(12): 9763-9773.

[15] Shannon C E.A mathematical theory of communication[J]. *The Bell System Technical Journal*, 1948, 27(3): 379-423.

[16] Gao Z, Yang L, Dai Y. Large-scale cooperative task offloading and resource allocation in heterogeneous mec systems via multi-agent reinforcement learning[J]. *IEEE Internet of Things Journal*, 2023, 11(2): 2303-2321.

[17] Zhai Z W, Wu Q, Yu S, et al.FedLEO: an offloading-assisted decentralized federated learning framework for low earth orbit satellite networks[J]. *IEEE Transactions on Mobile Computing*, 2023, 23(5): 5260-5279.

[18] Chen Q J, Yang C, Lan S L, et al. Two-stage evolutionary search for efficient task offloading in edge computing power networks[J]. *IEEE Internet of Things Journal*, 2024, 11(19): 30787-30799.

(下转第25页)