# 基于深度强化学习的多用户边缘计算任务卸载调度与资源分配算法

邝祝芳[1)] 陈清林[1)] 李林峰[1)] 邓晓衡[2)] 陈志刚[2)]

[1)]（中南林业科技大学计算机与信息工程学院 长沙 410004）

[2)]（中南大学计算机学院 长沙 410083）

**摘 要** 移动边缘计算(Mobile Edge Computing，MEC)把计算和存储等资源部署在网络边缘以满足某些对延迟要求苛刻的应用.用户设备可以通过无线网络将计算任务整体或者部分卸载到边缘服务器执行从而降低延迟和本地耗能，进而获得良好的用户体验.现有传统优化算法在MEC卸载决策和资源分配方面是可行的，但传统优化算法并不很适合高实时性的MEC系统.深度强化学习可以通过与传统优化算法不同的方式，建立尝试-奖励反馈机制，通过积累经验进行学习，从而完成优化目标.本文考虑移动边缘计算网络中多用户多任务卸载的情况下，研究任务卸载中的卸载决策和任务调度以及服务器资源分配的问题，以最小化系统延迟和传输耗能、本地耗能为目标，基于深度强化学习提出了一种多用户多任务下的任务卸载调度与资源分配算法，在上层服务器分配资源确定的情况下，提出基于贪心策略的流水车间调度算法解决了任务卸载决策和卸载调度问题，下层采用强化学习方法优化服务器资源分配问题.仿真结果表明，所提出的方法在降低延迟和本地耗能方面有比较优越的性能.

**关键词** 移动边缘计算；深度强化学习；任务卸载；任务调度；资源分配

**中图法分类号** TP393 **DOI号** 10.11897/SP.J.1016.2022.00812

# Multi-user Edge Computing Task offloading Scheduling and Resource Allocation Based on Deep Reinforcement Learning

KUANG Zhu-Fang[1)] CHEN Qing-Lin[1)] LI Lin-Feng[1)] DENG Xiao-Heng[2)] CHEN Zhi-Gang[2)]

[1)](*School of Computer and Information Engineering*, *Central South University of Forestry and Technology*, *Changsha* 410004)

[2)](*School of Computer Science and Engineering*, *Central South University*, *Changsha* 410083)

**Abstract** Mobile Edge Computing (MEC) deployments computing and storage resources at the edge of a network to meet some latency-critical applications. The user device can offload the computing tasks to the edge server in whole or in part through the wireless network so as to reduce the latency and local power consumption, thus obtaining a good user experience. The existing traditional optimization algorithm is feasible in the aspects of MEC offloading decision and resource allocation, but the traditional optimization algorithm is not very suitable for the high real-time MEC system. Deep reinforcement learning is different from the traditional optimization algorithm by establishing the trial-reward feedback mechanism, and learning by accumulating experience, so as to achieve the optimization goal. In the case of multi-user and multi-task

收稿日期：2021-01-26；在线发布日期：2021-08-07.本课题得到国家自然科学基金资助项目(62072477，61309027)、湖南省自然科学基金资助项目(2018JJ3888)、湖南省教育厅优秀青年项目(18B197)、国家重点研发计划项目(2018YFB1700200)、智慧物流技术湖南省重点实验室课题资助项目(2019TP1015)、中国高校产学研创新基金(No.2021FNA04009)资助.**邝祝芳**(通信作者)，博士，教授，硕士生导师，中国计算机学会(CCF)高级会员，主要研究领域为物联网、边缘计算、大数据分析、人工智能. E-mail：zfkuangcn@163.com. **陈清林**(通信作者)，硕士研究生，主要研究领域为边缘计算. E-mail：1311085095@qq.com. **李林峰**，硕士，主要研究领域为边缘计算. **邓晓衡**，博士，教授，博士生导师，中国计算机学会(CCF)高级会员，主要研究领域为无线通信与网络、边缘计算. **陈志刚**，博士，教授，博士生导师，中国计算机学会(CCF)高级会员，主要研究领域为网络与分布式计算、边缘计算.

offloading in mobile edge computing networks, the offloading decision, task scheduling and server resource allocation are studied in order to minimize system delay, transmission energy consumption and local energy consumption. Based on deep reinforcement learning, we propose a task offload scheduling and resource allocation algorithm for multi-user and multi-task. In the upper layer, with giving the resource allocation of MEC server, the flow shop scheduling algorithm based on greedy strategy is proposed to solve the problem of task offloading decision and offloading scheduling. In the lower layer, the reinforcement learning method is used to optimize the problem of resource allocation. Simulation results show that the proposed method has superior performance in reducing system latency and local energy consumption.

**Keywords** Mobile edge computing; deep reinforcement learning; task offloading; task scheduling; resource allocation

## 1 引　言

近年来，越来越多的移动设备连接到互联网上，加上严格的5G要求，为设计网络基础设施带来了新的挑战[1]. 许多时延敏感的服务如自动驾驶汽车、交互式游戏和电子医务（以及其他）等工业垂直行业需要安全、低延迟和可靠的通信. 为了满足这些严格的要求，计算资源必须移到离用户更近的地方，从网络的核心移到边缘，其中服务器位于用户附近，以最小化回程延迟[2]. 移动边缘计算（Mobile Edge Computing，MEC）是一种支持移动用户设备（User Equipments，UEs）实现计算敏感应用的新兴技术[3]. 将MEC集成到物联网（Internet of Things，loT）中，可以将计算能力和能量有限的loT设备的计算密集型和延迟敏感任务转移到网络边缘，从而为设备提供高质量的服务[4].

随着各种移动应用程序的广泛采用，设备以非常快的速度产生大量数据，这给移动核心网络和回程链路带来了巨大的负担[5]. 为了降低移动设备的时延和能量消耗，在移动边缘计算中采用了计算卸载策略[6]. 为了使系统时延和能耗最小化，需要在移动设备和MEC服务器之间共同分配通信和计算资源. 最近有很多在MEC系统卸载和资源分配联合设计方面的研究工作在进行[7-9].

针对单用户MEC系统，文献[10]研究了任务卸载调度和传输功率分配的联合问题. 在文献[11]中，研究了超低功率雾计算交换同步无线信息和功率传输（Simultaneous Wireless Information and Power Transfer，SWIPT）单用户MEC系统中的资源分配问题. 在文献[12]中，考虑到功率受限和不可预测的任务，以单用户处理能力最大化为目标的功率分配问题对MEC系统进行了研究，提出了一种二元搜索充水算法. 在文献[13]中，设计了一种迭代启发式的MEC资源分配算法，以动态进行卸载决策. 然而在文献[11]-[13]中，没有考虑任务卸载调度问题，单用户MEC系统中的联合任务部分卸载决策、卸载调度和资源分配仍然是一个有待解决的问题.

对于多用户MEC场景，有一些最近的研究. 在文献[14]中，考虑了一个具有多个用户和单个MEC服务器的非正交多路访问（Non-Orthogonal Multiple-Access，NOMA）MEC系统，通过使用强化学习（RL）算法Deep Q-network（DQN）在不事先知道其他用户动作的情况下选择同时卸载的用户，得到最优组合状态，使系统卸载延迟最小. 在文献[15]中，研究多用户MEC系统中的资源分配问题，通过利用回归算法求解通信资源（子载波）的分配问题并合理分配通信和计算资源，在延迟约束下实现系统能量消耗最小. 在文献[16]中，为了进一步提高MEC的接入容量和最小化所有用户的计算开销，利用频谱效率较优的非正交多路访问技术，共同优化计算卸载决策，通信和计算资源分配. 在文献[17]中，针对NOMA用户，研究了不同上传时延与共信道干扰之间的相互作用，提出了一种计算卸载方案，通过对卸载决策和资源分配的联合优化来降低用户的平均卸载延迟. 在文献[18]中，研究了在一个基站服务的多个用户之间进行卸载决策和资源分配的问题，以达到最优的系统用户效用，即任务延迟和能量消耗之间的权衡. 在文献[19]中，提出一种基于强化学习的状态-动作-奖赏-状态-动作（RL-SARSA）算法来解决边缘服务器的资源管理问题，并通过优化卸载决策来最小化系统成本（包括能量消耗和计算时延）.

以上研究主要考虑多用户的MEC卸载场景，

以最小化能耗或延迟为目标.然而,在多用户MEC卸载系统中,任务卸载调度一直没有得到重视.从本质上讲,多用户MEC系统场景下的联合卸载决策和卸载调度问题是一个具有挑战性的问题.

除了考虑多用户MEC系统中的计算卸载问题外,还考虑了异构MEC系统.在文献[20]中,提出了一种基于MEC和云计算的车辆网络中向汽车卸载服务的协同方法,解决了联合优化计算卸载决策和计算资源分配问题.在文献[21]中,提出了两种有效的分散算法,供设备在成本最小和时间公平的资源分配策略下实现卸载决策的计算均衡.在文献[22]中,提出了一个云和边缘计算系统,该系统包括终端层、边缘计算层和云数据中心(Cloud Data Centers, CDCs)层.在此基础上,设计了一种利润最大化的协同计算卸载和资源分配算法,使系统的利润最大化,并保证任务的响应时间限制得到严格满足.在文献[23]中,研究了具有异构云(包括边缘云和远程云)的多用户MEC系统的任务卸载问题,在任务的有界延迟要求下,最小化多个移动设备的总能量消耗.在文献[24]中,研究了单用户MEC系统中的部分卸载调度与功率分配问题,在保证任务的传输功率约束的情况下,使执行延迟和能量消耗的加权和最小.在文献[20]-[23]中,主要考虑了异构MEC系统中任务卸载决策与资源分配,但是没有研究卸载调度问题.在文献[24]中主要研究了异构MEC系统中任务卸载决策、卸载调度问题和功率分配问题,但是没有考虑多用户问题.多用户MEC中任务卸载、调度和资源分配的联合问题尚未得到深入研究.

在本文中,考虑了多个具有多个计算任务的移动设备,以及加载和执行任务的并行实现.在多用户MEC系统中,我们设计了一种高效、低延迟的移动设备MEC卸载机制.这种机制联合优化任务卸载、调度和资源分配问题,最大限度地降低了系统的能耗,同时也减少了系统的延迟.本文的贡献主要体现在3个方面:(1)对具有多用户多任务的MEC系统的任务部分卸载调度和资源分配问题进行了建模,研究了任务卸载决策、卸载调度和功率分配的联合问题,目标为最小化计算系统执行延迟和能耗的加权和;(2)提出了一个基于强化学习的多用户多任务卸载调度和资源分配算法(Multi-user Multi-task Offloading Scheduling and Power allocation algorithm based on DQL, MMOA-DQN);(3)在上层的服务器分配频率确定的情况下,提出基于贪心策略的流水车间调度算法解决了任务卸载决策和卸载调度问题.下层采用强化学习方法优化服务器频率分配问题.

本文第2节介绍了网络模型与问题描述;第3节详细介绍文中算法的主要步骤;第4节通过实验对比文中提出的算法的性能;第5节是总结.

## 2 网络模型和问题描述

### 2.1 网络模型

MEC系统由一个配备MEC服务器的无线基站(eNodeB, eNB),$K$个移动设备组成,表示为$r=\{U_1,U_2,\cdots,U_K\}$;每个移动设备$U_i$有$N$个独立的计算任务,$U_i$的任务集合表示为$\Gamma_i=\{T_{i,1},T_{i,2},\cdots,T_{i,N}\}$;MEC服务器是一个计算强大的服务器,可以为移动设备提供高强度的计算服务.移动设备将计算任务卸载给MEC服务器,MEC服务器处理后将计算结果返回给相应的设备.我们考虑移动边缘计算系统中$K$个移动设备,如图1所示.移动设备通过将任务转移到MEC服务器,MEC服务器为每个用户分配服务器资源来降低用户设备的耗能和延迟.本模型中,边缘服务器存储未处理任务的缓存足够大,且用户设备和边缘服务器均配备一个单核CPU,所有任务按照输入CPU的顺序依次处理.

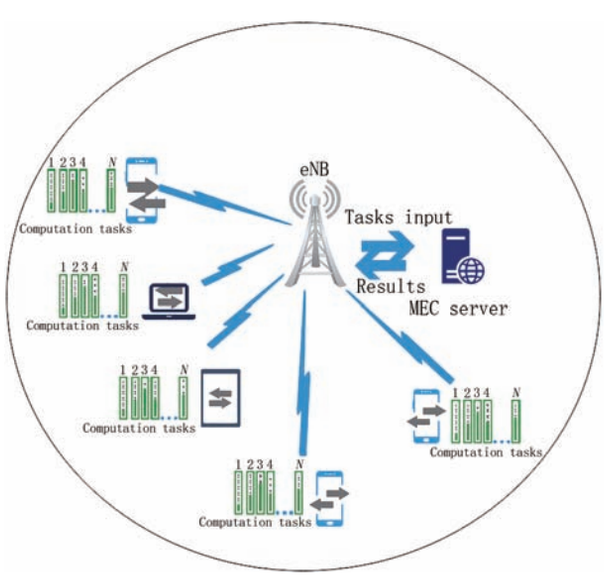



图1 系统模型

### 2.2 问题描述

系统共$K$个移动设备,所有移动设备所有任务集合表示为$G=\{T_{i,j}|1\leqslant i\leqslant K,1\leqslant j\leqslant N\}$,共$N^*$

$K$个独立的计算任务需要处理，其中所有的任务都可以被卸载和调度. 每个任务$T_{i,j}$由$(D_{i,j}, C_{i,j})$两个参数描述，对于移动设备$U_i$的任务$T_{i,j}$，$D_{i,j}$(以bit为单位)表示处理任务所需要的数据量，表示(以CPU周期为单位)处理该任务的每单位数据所需要的CPU周期数. 在本文中，我们主要研究移动设备计算资源有限的情况，且每个任务都不能进一步划分；每个移动设备只装备一个天线，每次只能传输一个任务；所有用户均分传输带宽；因此，传输移动设备$U_i$的任务$T_{i,j}$到边缘服务器的速率表示为

$$R_{i,j}=\frac{W}{K}\log_2\left(1+\frac{g_0(L_0/L_i)^{\theta}p}{N_0 w/K}\right) \tag{1}$$

其中，$W$为传输带宽，$g_0$为路径损耗常数，$L_0$为相对距离，$L_i$为用户$i$到服务器的实际距离，$\theta$为路径损耗指数，$N_0$为噪声功率谱密度，$p$表示移动设备卸载任务$T_{i,j}$到边缘服务器的传输功率. 每个任务既可以选择在本地执行也可以在服务器执行，因此我们引入一个卸载决策变量$X=\{x_{i,j}\}$表示任务卸载情况：

$$x_{i,j}=\begin{cases}1 & \text{任务卸载执行时}\\ 0 & \text{任务本地执行时}\end{cases} \tag{2}$$

为了使服务器CPU在传输过程中不空载，因此对所有边缘设备$\Upsilon=\{U_1, U_2, \cdots, U_K\}$的所有卸载任务进行调度，将它们按照一定顺序进行排序并存入卸载任务集合$S=\{S_1, \cdots, S_i, \cdots, S_K\}$. 卸载任务集合中的任意子集$S_i$为有序集合，共有$Ns$个任务，记为：$S_i=[S_{i,1}, S_{i,2}, \cdots, S_{i,Ns}]$，每个任务的传输功率为$p$. 相对的，将所有边缘设备$K=\{U_1, \cdots, U_i, \cdots, U_K\}$的所有本地任务存入集合$L=\{L_1, \cdots, L_i, \cdots, L_K\}$，本地集合中的任意子集$L_i$中共有$Nl$个任务，记为$L_i=[L_{i,1}, L_{i,2}, \cdots, L_{i,Nl}]$. 此外，$N=Ns+Nl$.

所有的边缘设备随机均匀分布于基站旁边，距离服从均匀分布$L\sim Unif([l_{\min}, l_{\max}])$，服务器的总频率为$F$，分配给边缘设备$U_i$的频率为$f_{i,ser}$，用户$U_i$的本地CPU频率为$f_{i,user}$.

移动设备(边缘设备)$U_i$的任务$T_{i,j}$在边缘服务器的执行时间表示为$t^{\text{dispose}}_{(i,j),S}$：

$$t^{\text{dispose}}_{(i,j),S}=\frac{C_{i,j}}{f_{i,ser}} \tag{3}$$

任务$T_{i,j}$的本地执行时间表示为$t^{\text{dispose}}_{(i,j),L}$：

$$t^{\text{dispose}}_{(i,j),L}=\frac{C_{i,j}}{f_{i,user}} \tag{4}$$

任务$T_{i,j}$的卸载传输时间表示为$t^{\text{trans}}_{i,j}$：

$$t^{\text{trans}}_{i,j}=\frac{D_{i,j}}{R_{i,j}} \tag{5}$$

任务$T_{i,j}$的卸载传输耗能为$e_{(i,j),S}$：

$$e_{(i,j),S}=t^{\text{trans}}_{i,j}p \tag{6}$$

任务$T_{i,j}$的本地执行耗能为$e_{(i,j),L}$：

$$e_{(i,j),L}=\delta_{i,L}f^2_{i,\text{user}}C_{i,j} \tag{7}$$

其中，$\delta_{i,L}$为边缘设备$U_i$有效电容系数，取决于CPU的芯片架构.

边缘设备$U_i$的耗能$e_i$为卸载耗能加上本地执行耗能：

$$e_i=\sum_{j=1}^{N}\left(e_{(i,j),S}+e_{(i,j),L}\right) \tag{8}$$

系统的总消耗为每个用户最后一个任务的完成延迟与该边缘设备的耗能加权和：

$$E_i=max\left\{t^{\text{comp}}_{(i,Ns),S}, t^{\text{comp}}_{(i,Nl),L}\right\}+e_i \tag{9}$$

其中$t^{\text{comp}}_{(i,Ns),S}$表示排序后所有卸载任务的完成时间，$Ns$表示所有卸载执行任务数. $t^{\text{comp}}_{(i,Nl),L}$表示所有本地任务的完成时间，$Nl$表示本地执行任务数. $\sum_{i=1}^{K}\sum_{j=1}^{N}(e_{(i,j),S}+e_{(i,j),L})$表示边缘服务器执行所有任务的总耗能.

本文所描述的数学模型是一个有约束的联合优化问题，在服务器资源有限的条件下，优化卸载决策、卸载调度和服务器资源分配. 因此，模型的优化目标式可表示为

$$P1:\min_{S,L,X,f_{ser}}\sum_{i=1}^{K}E_i \tag{10a}$$

s. t.

$$\sum_{i=1}^{K}f_{i,ser}\leqslant F \tag{10b}$$

$$f_{i,ser}\leqslant F, \forall i\in\Upsilon \tag{10c}$$

$$x_{i,j}\in\{0,1\},\quad \forall T_{i,j}\in G \tag{10d}$$

$$S_i=\left[S_{i,1}, S_{i,2}, \cdots, S_{i,Ns}\right], S_{i,j}\neq S_{i,k}, k\neq j, \forall S_{i,j}, S_{i,k}\in G_i \tag{10e}$$

本文优化的变量包括任务卸载决策、任务卸载调度、边缘服务器CPU频率分配.

目标式(10a)为系统延迟和移动用户耗能的加权和.

约束(10b)表示服务器分配频率不能超过最大总频率.

约束(10c)表示边缘设备分得的频率不能超过最大总频率.

约束(10d)表示任务只能在本地或者服务器端

执行.

约束(10e)表示任务的卸载顺序.

问题(10a)可以通过寻找最优卸载决策和计算卸载的资源分配来解决.然而卸载决策向量$\boldsymbol{X}$是二元变量的可行集并且目标函数是非凸问题.此外,当任务数增加时,问题(10a)的求解难度将会呈指数规模增加,是NP难题.本文结合传统优化方法和强化学习法来进行联合求解,用针对多用户基于流水车间调度原理的部分卸载决策与调度算法解决移动设备上的卸载决策和卸载调度问题,用强化学习方法解决服务器端的资源分配问题.

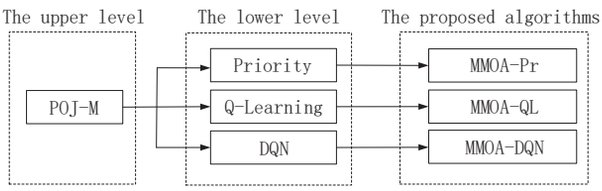
## 3 强化学习优化算法

由于P1是一个混合整数的非凸问题,为了解决这个问题,本文提出了一个多用户多任务卸载调度和资源分配算法框架(Multi-user Multi-task Offloading Scheduling and Power allocation algorithm,MMOA),并且在此基础之上提出了基于优先级、基于Q-Learning、基于深度强化学习的求解算法.MMOA算法的基本思想是结合流水车间调度算法和强化学习算法,在给出边缘服务器CPU频率分配的情况,基于流水车间调度算法求解任务卸载决策和卸载调度子问题,在任务卸载决策和卸载调度顺序给定的情况,基于强化学习方法求解边缘服务器的CPU频率分配子问题,通过双层反复迭代调用最终求得问题的近似最优解.

MMOA算法是一个双层迭代算法框架,上层在边缘服务器分配CPU频率$f_{ser}=\{f_{1,ser},\cdots,f_{K,ser}\}$确定的情况下,利用基于贪心策略的流水车间调度算法(POJ-M)解决任务卸载决策和卸载调度子问题.下层采用强化学习方法进行边缘服务器CPU频率分配.为了说明和提高本文提出的基于深度强化学习算法的优势,在3.1中,首先提出一个基于优先级的资源分配算法(MMOA-Pr).在3.2中,然后提出一个基于Q-Learning算法的强化学习资源分配算法(MMOA-QL).最后在3.3中,提出一个基于DQN的强化学习资源分配算法(MMOA-DQN).图2为所提出算法之间的关系,图3为DQN算法的算法流程.

### 3.1 优先级算法MMOA-Pr

为了提高本文提出的基于DQN的强化学习资源分配算法MMOA-DQN的性能,以及减少MMOA-DQN算法的搜索空间,并为接下来的卸载




POJ-M: Partial Offloading decision and scheduling based on Johnson algorithm for Multi-user

MMOA-Pr: Multi-user Multi-task Offloading Scheduling and Power allocation algorithm based on Priority

MMOA-QL: Multi-user Multi-task Offloading Scheduling and Power allocation algorithm based on Q-Learning

MMOA-DQN: Multi-user Multi-task Offloading Scheduling and Power allocation algorithm based on DQN

图2 所提出算法之间的关系

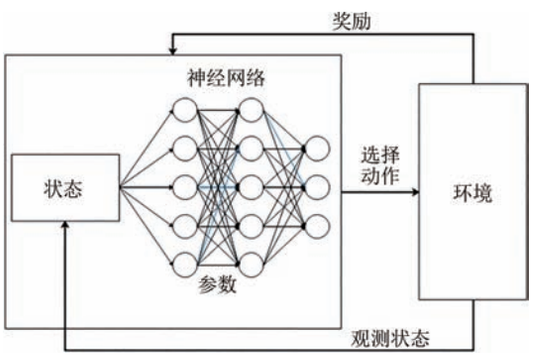



图3 带有神经网络的DQN算法流程

决策和卸载调度提供一个初始服务器分配频率,同时为了体现MMOA-QL、MMOA-DQN算法的优势,我们首先提出一个优先级资源分配算法(MMOA-Pr),其基本思想是基于优先级给各边缘设备分配CPU频率资源.

MMOA-Pr算法求解服务器分配给边缘设备的CPU频率$f_{ser,base}=\{f_{1,ser},\cdots,f_{i,ser},\cdots,f_{K,ser}\}$的优先级算法步骤如下:

1)计算边缘设备$U_i$的资源占系统总资源的比例$f_{i,ratio}$:

$$f_{i,ratio}=\frac{f_{i,user}}{\sum_{i=1}^{K}f_{i,user}} \tag{11}$$

$f_{i,ratio}$表示边缘设备$U_i$的资源充盈度,越小则本地处理能力越弱,其任务需要卸载到MEC进行处理的程度越大.

2)计算边缘设备$U_i$的本地执行时延占系统总时延的相对比例$t_{i,ratio}$:

$$t_{i,ratio}=1-\frac{\sum_{j=1}^{N}t_{(i,j),L}^{\text{dispose}}}{\sum_{i=1}^{K}\sum_{j=1}^{N}t_{(i,j),L}^{\text{dispose}}} \tag{12}$$

$t_{i,ratio}$表示边缘设备$U_i$的任务紧急程度,越大则

任务越紧急，其任务需要卸载到MEC进行处理的程度越高.

3）计算边缘设备$U_i$的频率分配权重$\bar{\eta}_i$：

$$\bar{\eta}_i = t_{i,ratio} \Big/ f_{i,ratio} \tag{13}$$

$\bar{\eta}_i$分子越大，表示任务越紧急，需要卸载到MEC进行处理的程度越大；分母越小，表示本地处理能力越弱，需要卸载到MEC进行处理的程度越大.故，$\bar{\eta}_i$越大，边缘设备$U_i$需要卸载任务到MEC进行处理的程度越大，所分配的服务器资源越多.

4）对权重$\bar{\eta}_i$进行归一化：

$$\eta_i = \frac{\bar{\eta}_i}{\sum_{k=1}^{K} \bar{\eta}_k} \tag{14}$$

5）边缘服务器分配给移动设备的CP频率$f_{i,ser}$：

$$f_{i,ser} = \eta_i \times F \tag{15}$$

MMOA-Pr算法如表1所示：

**表1　优先级算法MMOA-Pr**

**算法1.**　优先级算法MMOA-Pr:

输入：用户数$K$，移动设备$U_i$的所有任务集合$G_i$，移动设备$U_i$的CPU频率$f_{i,user}$，边缘服务器分配给移动设备的CPU频率$f_{i,ser}$，服务器CPU总频率$F$

输出：分配频率$f_{ser,base}=\{f_{1,ser},\cdots,f_{i,ser},\cdots,f_{K,ser}\}$

1：根据式(11)计算边缘设备$U_i$的资源占系统总资源的比例$f_{i,ratio}$

2：根据式(12)计算边缘设备$U_i$的本地执行时延占系统总时延的相对比例$t_{i,ratio}$

3：根据式(13)计算边缘设备$U_i$的频率分配权重$\bar{\eta}_i$

4：通过式(14)对权重$\bar{\eta}_i$进行归一化

5：通过式(15)计算边缘服务器分配给边缘设备的频率$f_{i,ser}$

### 3.2　卸载决策与调度算法POJ-M

在上层，服务器分配频率$f_{ser}=\{f_{1,ser},\cdots,f_{K,ser}\}$确定的情况下，利用基于贪心策略的流水车间调度算法（POJ-M）求解任务卸载决策和卸载调度问题.

POJ-M算法的基本思想是：基于流水车间作业调度理论求解边缘设备$U_i$的任务卸载决策、以及卸载任务调度顺序，其步骤如下：

1）对边缘设备$U_i$的所有任务$T_{i,j}$进行分类，通过比较卸载传输时间$t_{i,j}^{\text{trans}}$和边缘服务器执行时间$t_{(i,j),S}^{\text{dispose}}$，将卸载传输时间小于边缘服务器执行时间的任务加入数组$P_i$，$P_i=\{T_{i,j}|t_{i,j}^{\text{trans}}<t_{(i,j),S}^{\text{dispose}},\ T_{i,j}\in G_i\}$，将$P_i$中所有任务根据卸载传输时间$t_{i,j}^{\text{trans}}$升序排列.将卸载传输时间大于或等于边缘服务器执行时间的任务加入数组$Q_i$，$Q_i=\{T_{i,j}|t_{i,j}^{\text{trans}}\geqslant t_{(i,j),S}^{\text{dispose}},\ T_{i,j}\in G_i\}$.将$Q_i$中所有任务根据边缘服务器执行时间$t_{(i,j),S}^{\text{dispose}}$降序排列.将数组$Q_i$加到数组$P_i$后面得到新的任务顺序$\sigma_i=[P_i\ Q_i]$.

2）设数组$P_i$和数组$Q_i$的初始下标值分别为$hp=1$和$hq=1$，从数组$P_i$中取出$P_i[hp]$放入卸载任务集合$S_i$，任务$P_i[hp]$的卸载决策变量$x_{i,hp}=1$，$hp=hp+1$.从数组$Q_i$中取出$Q_i[hq]$放入本地任务集合$L_i$，任务$Q_i[hq]$的卸载决策变量$x_{i,hq}=0$，$hq=hq+1$.

3）计算集合$L_i$中新加入的第一个任务$k0=1$的完成时间$t_{(i,k0),L}^{\text{comp}}$，计算集合$S_i$中新加入的第一个任务$k1=1$的完成时间$t_{(i,k1),S}^{\text{comp}}$，分别如式(16)和式(17)所示.

$$t_{(i,k0),L}^{\text{comp}} = t_{(i,k0),L}^{\text{dispose}} \qquad k0=1 \tag{16}$$

$$t_{(i,k1),S}^{\text{comp}} = t_{(i,k1),S}^{\text{trans}} + t_{(i,k1),S}^{\text{dispose}} \qquad k1=1 \tag{17}$$

4）比较$t_{(i,k0),L}^{\text{comp}}$、$t_{(i,k1),S}^{\text{comp}}$的大小，若$t_{(i,k0),L}^{\text{comp}}<t_{(i,k1),S}^{\text{comp}}$，说明本地任务集$L_i$中新加入的任务$k0$先执行完，则执行步骤i），否则执行步骤ii），以下两步循环执行直至跳出循环：

i）从数组$Q_i$中反复取任务$Q_i[hq]$放入本地集合$L_i$，任务$Q_i[hq]$的卸载决策变量$x_{i,hq}=0$，$hq=hq+1$，$k0=k0+1$，根据式(18)计算新加入的任务$k0$的完成时间，比较$t_{(i,k0),L}^{\text{comp}}$与$t_{(i,k1),S}^{\text{comp}}$，如果$t_{(i,k0),L}^{\text{comp}}$小于$t_{(i,k1),S}^{\text{comp}}$，并且$Q_i$中还有任务，则继续执行步骤i）；如果$t_{(i,k0),L}^{\text{comp}}$大于$t_{(i,k1),S}^{\text{comp}}$，并且$Q_i$中还有任务，则执行步骤ii）；如果$t_{(i,k0),L}^{\text{comp}}$小于$t_{(i,k1),S}^{\text{comp}}$，并且$Q_i$中没有任务了，则说明$Q_i$中任务被取完且$L_i$中所有任务完成时间仍小于集合$S_i$中所有任务的完成时间，则执行步骤5），并将$QN$标志位置1，表示$Q_i$集合被提前分配完且$P_i$集合有剩余.

ii）从数组$P$中反复取任务$P_i[hp]$放入卸载集合$S_i$，任务$P_i[hp]$的卸载决策变量$x_{i,hp}=1$，$hp=hp+1$，$k1=k1+1$，根据式(19)计算新加入的任务$k1$的完成时间，比较$t_{(i,k0),L}^{\text{comp}}$与$t_{(i,k1),S}^{\text{comp}}$，如果$t_{(i,k1),S}^{\text{comp}}$小于$t_{(i,k0),L}^{\text{comp}}$，并且$P_i$中还有任务，则继续执行步骤ii）；如果$t_{(i,k1),S}^{\text{comp}}$大于$t_{(i,k0),L}^{\text{comp}}$，并且$P_i$中还有任务，则执行步骤i）；如果$t_{(i,k1),S}^{\text{comp}}$小于$t_{(i,k0),L}^{\text{comp}}$，并且$P_i$中没有任务了，则说明$P_i$中任务被取完且$S_i$中所有任务完成时间仍小于集合$L_i$中所有任务的完成时间，则执行步骤5），并将$PN$标志位置1，表示$P_i$集合被提前分配完成且$Q_i$集合有剩余.

$$t_{(i,k0),L}^{\text{comp}} = \sum_{k=1}^{k0} t_{(i,k),L}^{\text{dispose}},\ k0>1 \tag{18}$$

$$t_{(i,k1),S}^{\text{comp}} = \max\{t_{(i,k1),S}^{\text{ready}}, t_{(i,k1-1),S}^{\text{comp}}\} + t_{(i,k1),S}^{\text{dispose}},\ k1>1 \tag{19}$$

其中，

$$t^{\text{ready}}_{(i,k1),S}=\sum_{k=1}^{k1}t^{\text{trans}}_{(i,k),S} \tag{20}$$

5) 检测标志位$PN$、$QN$. 若$QN$=1，则集合$P_i$中任务仍有剩余，将集合$P_i$中所有任务存入集合$M$；若$PN$=1，则集合$Q_i$中任务仍有剩余，将集合$Q_i$中所有任务存入集合$M$；

6) 取出集合$M$中的任务，根据公式(18)、(19)分别求出若该任务加入集合$L_i$、集合$S_i$中的完成时间$t^{\text{comp}}_{(i,k0),L}$、$t^{\text{comp}}_{(i,k1),S}$.

7) 比较两者大小，若$t^{\text{comp}}_{(i,k0),L}<t^{\text{comp}}_{(i,k1),S}$，则将任务加入本地集合$L_i$，否则将任务加入卸载集合$S_i$.

8) 反复执行步骤6)到步骤7)，直至$M$中任务被取完为止. 算法步骤如表2所示：

**表2 POJ-M算法**

**算法2.** 多用户基于流水车间作业调度的卸载调度方法POJ-M

输入：边缘设备$U_i$的所有任务集合$G_i$，边缘设备$U_i$的CPU频率$f_{i,user}$，服务器分配给边缘设备$U_i$的CPU频率$f_{i,ser}$.

输出：卸载任务集合$S_i=\{S_{i,1},S_{i,2},\cdots,S_{i,Ns}\}$，本地任务集合$L_i=\{L_{i,1},L_{i,2},\cdots,L_{i,Nl}\}$，卸载决策向量$X_i=\{x_{i,1},x_{i,2},\cdots,x_{i,K}\}$.

初始化：分类所有任务$T_{i,j}$

1. IF $t^{\text{trans}}_{i,j}<t^{\text{dispose}}_{(i,j),S}$
   任务$T_{i,j}$加入数组$P_i$，将$P_i$中所有任务根据卸载传输时间$t^{\text{trans}}_{i,j}$升序排列
   ELSE
   任务$T_{i,j}$加入数组$Q_i$，将$Q_i$中所有任务根据边缘服务器执行时间$t^{\text{dispose}}_{(i,j),S}$降序排列.
   将数组$Q_i$加到数组$P_i$后面得到新的任务顺序$\sigma_i=[P_i\ Q_i]$
2. 将数组$P_i$和数组$Q_i$的初始下标值分别为$hp$=1和$hq$=1，从数组$P_i$中取出$P_i[hp]$放入卸载任务集合$S_i$，任务$P_i[hp]$的卸载决策变量$x_{i,hp}$=1，$hp=hp+1$.
3. 从数组$Q_i$中取出$Q_i[hq]$放入本地任务集合$L_i$，任务$Q_i[hq]$的卸载决策变量$x_{i,hq}$=0，$hq=hq+1$.
4. 通过式(16)(17)计算集合$L_i$中新加入的第一个任务k0=1的完成时间$t^{\text{comp}}_{(i,k0),L}$与计算集合$S_i$中新加入的第一个任务k1=1的完成时间$t^{\text{comp}}_{(i,k1),S}$.
5. DO
   WHILE $t^{\text{comp}}_{(i,k0),L}\geqslant t^{\text{comp}}_{(i,k1),S}$ and $hq\leqslant|Q_i|$
   $hq=hq+1$, $k0=k0+1$, $x_{i,hq}=0$
   根据式(18)计算新加入的任务$k0$的完成时间
   END WHILE
   WHILE $t^{\text{comp}}_{(i,k0),L}\leqslant t^{\text{comp}}_{(i,k1),S}$ and $hp\leqslant|P_i|$
   $hp=hp+1$, $k1=k1+1$, $x_{i,hp}=1$
   根据式子(19)计算新加入的任务$k1$的完成时间
   END WHILE

UNTIL $hq=|Q|$ or $hp=|P|$

IF $hq=|Q|$
$QN$=1

IF $hp=|P|$
$PN$=1

6. IF $QN$=1
   将集合$P_i$中所有任务存入集合$M$
   IF $PN$=1
   将集合$Q_i$中所有任务存入集合$M$
7. FOR 1:|M|
   1) 根据公式(18)、(19)分别求出若该任务加入集合$L$、集合$S$中的完成时间$t^{\text{comp}}_{(i,k0),L}$、$t^{\text{comp}}_{(i,k1),S}$
   2) IF $t^{\text{comp}}_{(i,k0),L}<t^{\text{comp}}_{(i,k1),S}$
   将任务$T_{i,j}$加入本地集合$L_i$
   ELSE
   将任务$T_{i,j}$加入卸载集合$S_i$

### 3.3 卸载决策与资源分配

在下层，任务卸载决策和卸载任务调度顺序确定的情况下，基于深度强化学习求解边缘服务器CPU频率分配. 为了说明和提高本文提出的基于深度强化学习算法的优势，首先提出基于Q-Learning算法的强化学习资源分配算法MMOA-QL，然后提出基于DQN的深度强化学习资源分配算法MMOA-DQN.

资源分配是卸载调度的依据，本节提出的MMOA-QL和MMOA-DQN采用强化学习的方式进行资源分配. 强化学习的方法有几个关键要素，即状态、动作、奖励，具体到本文的模型，其说明如下：

1) 状态：状态包含当前系统所有的计算资源占用情况、任务卸载调度情况、系统执行情况、服务器资源分配情况等情况. 虽然计算所有任务所需的CPU周期数和系统可分配资源是不变的，但是由于MEC服务器资源分配不同与每个用户所包含的任务大小以及处理难度不一致，这导致了任务卸载到MEC服务器上的完成时延、处理该任务的服务器频率不同. 因此在状态中，任务需要包含卸载到MEC服务器的延迟和服务器资源分配的情况. 因此，系统状态由两个部分组成：

$$s=(tc,ac) \tag{21}$$

我们将$tc$定义为整个系统的消耗，$tc=E$，由式(10a)可得. $ac$为MEC服务器的可用计算容量：

$$ac=F-\sum_{i}^{K}f_{i,ser} \tag{22}$$

2) 动作：对于每个移动设备而言，服务器可为其分配不大于最大服务器频率的服务器资源协助处理任务，由于频率分配是一个连续的范围，且每个移动设备可卸载多个任务到MEC服务器，因此动作空间很大. 在这里，我们根据服务器所分配的频率采用针对多用户基于流水车间调度理论的卸载决策

与调度算法(POJ-M)进行卸载决策和调度，从而大大压缩了神经网络的动作空间.此时，在我们的系统模型中，动作$a$只需要考虑服务器分配频率即可：$f=[f_{1,ser},f_{2,ser},\cdots,f_{K,ser}]$.

3) 奖励：优化目标是最小化延迟和能耗.在强化学习中，状态、动作、回报是一个不断循环的三元组，每个状态下产生的动作都有不同的奖励，这样不断循环直到到达结束态或者最大迭代次数为止.对于每一步，智能体在执行一个动作$a$后，都会在一定的状态$s$下，得到一个奖励$r$.一般来说，奖励应该与目标相关.因为，我们的优化问题的目标是获得最小的延迟和能耗，而强化学习的目标是获得最大奖励回报，因此奖励的价值应该与延迟和能耗的大小呈负相关，我们将奖励定义为归一化的如公式(23)所示：

$$r=1-\frac{tc}{tc_{\text{fulllocal}}} \tag{23}$$

其中，$tc_{\text{fulllocal}}$为完全本地的延迟和能耗，计算公式如式(24)所示：

$$tc_{\text{fulllocal}}=\sum_{i=1}^{K}\sum_{j=1}^{N}\left(t_{(i,j),L}^{\text{dispose}}+e_{(i,j),L}\right) \tag{24}$$

为了说明基于DQN的强化学习资源分配算法MMOA-DQN的优势，本节首先提出基于Q-Leaning的强化学习资源分配算法MMOA-QL算法，然后提出基于DQN的强化学习资源分配算法MMOA-DQN.基于Q-Leaning求解所有边缘设备$\mathrm{r}=\{U_1,U_2,\cdots,U_K\}$的服务器资源分配$f_{ser,best}=\{f_{1,ser},\ldots,f_{K,ser}\}$的求解步骤如下：

1) 初始化$Q$表，即生成一个$(n,n,K,n)$维的矩阵.由式(24)计算完全本地系统消耗$tc_{\text{fulllocal}}$.

2) 反复执行2)～3)步骤共Num次.

随机生成一个初始动作$a$，代入式(22)，获得初始系统状态$s=(tc,ac)$.将$(s,a)$存入Q表，存入方式如下：

$$(Q_{index1},Q_{index2},Q_i,Q_{index3})=\left(\frac{tc}{tc_{\text{fulllocal}}},\frac{ac}{F},i,f_{i,ser}\right) \tag{25}$$

$$index1=\left[\frac{tc}{tc_{\text{fulllocal}}}\times n\right] \tag{26}$$

$$index2=\left[\frac{ac}{F}\times n\right] \tag{27}$$

$$index3=\left[\frac{f_{i,ser}}{F}\times n\right] \tag{28}$$

其中，$n$为$index1$、$index2$、$index3$中各元素所能取到的最大值.

3) 反复执行i)到v)共C次：

i) 以$\epsilon$的概率输出一个满足约束(10b)、(10c)的随机动作$a$，或以1-$\epsilon$的概率输出Q表中最大值所对应的动作$a'=\arg\max_{\pi}Q^{*}(s,a)$.

ii) 将动作$a'$代入POJ-M算法进行调度排序，将调度结果$S$和动作$a'$代入式(22)得到$s'=(tc',ac')$.将$tc'$代入式(23)(26)得到$r$、$index1$，将$ac'$、$a'$代入式(27)、(28)，求得$index2$、$index3$.

iii) 若$\sum index3>n$，则结束当前循环.否则，记录最大的$r$ 值所对应的动作$a_{best}=a'$.

iv) 将$s'=(tc',ac')$、$a$、$a'$、$r$、$\gamma$、$\alpha$、$s=(tc,ac)$代入式(25)、(29)，以更新$Q$表.$Q$表更新公式如式(29)所示：

$$Q(s,a)=Q(s,a)+\alpha\left[r+\gamma max_{a'}Q(s',a')-Q(s,a)\right] \tag{29}$$

其中，$\gamma$为奖励性衰变系数，$\alpha$为学习率.

v) 状态迭代：$s=s'$

4) 将$a_{best}$代入POJ-M算法进行调度排序，将调度结果$S$和$a_{best}$代入式(10a)计算得到P1.

MMOA-QL算法如表3所示：

**表3　MMOA-QL算法**

**算法3.**　基于*Q-learning*的服务器资源分配算法：

输入：用户数$K$，边缘设备的CPU频率$f_{i,user}$，服务器CPU总频率$F$，迭代轮数Num，最大步长C，状态特征维度n，奖励性衰变系数$\gamma$，学习率$\alpha$

输出：最佳动作$a_{best}=(f_{1,ser}\cdots,f_{K,ser})$，目标值P1

初始化：Q表

1. 生成一个$(n,n,K,n)$维的矩阵.由式子(24)计算完全本地系统消耗$tc_{fulllocal}$

2. FOR 1: NUM

　1). 随机生成初始动作a，由式(22)求得初始系统状态$s=(tc,ac)$

　2). 由式子(25)(26)(27)(28)将$(s,a)$存入Q表

　　FOR 1:C

　　1). 随机生成一个数$\eta$，$\eta\in(0,1)$

　　IF $\eta<\epsilon$

　　生成一个满足约束(10b)(10c)的随机动作$a$

　　ELSE

　　选择Q表中最大值所对应的动作$a'=\arg\max_{\pi}Q^{*}(s,a)$

　3). 将$a'$代入POJ-M算法进行调度排序，并将调度结果$S$和$a'$代入式(10a)与(22)得到$s'=(tc',ac')$

　4). 将$tc'$代入式(23)(26)得到$r$，$index1$，

　5). 将$a'$、$ac'$代入式(27)(28)得到$index2$、$index3$.

　6). IF $\sum index3>n$

BREAK

记录最大的$r$值所对应的动作$a_{best}=a'$

7). 更新Q表

8). 状态迭代:$s=s'$

接下来,提出一个基于DQN的强化学习资源分配算法MMOA-DQN,其基本思想是在MMOA-QL算法的基础上,采用全连接神经网络代替Q表进一步优化算法性能.基于DQN求解所有边缘设备$r=\{U_1,U_2,\cdots,U_K\}$的服务器资源分配$f_{ser,best}=\{f_{1,ser},\cdots,f_{K,ser}\}$的求解步骤如下:

1)以$\varepsilon$的概率生成一个随机动作$a=\{(f_{1,ser},\cdots f_{i,ser},\cdots,f_{K,ser})\,|0\leqslant f_{i,ser}\leqslant 2f_{i,base}\}$,或以$1-\varepsilon$的概率将状态$s=(tc,ac)$输入目标网络$Q'$,预测出动作$a$.输出动作$a$对应的输出层神经元索引$a_{id}$,$step=step+1$.计算$a_{id}$的方法如下:

i)对于随机动作$a=(f_{1,ser},\cdots f_{i,ser},\cdots,f_{K,ser})$,首先生成一个数组$F_{i,list}$,$F_{i,list}$为范围在$(0,2f_{i,base})$之间共sec个数的等差数列,sec为候选动作数.依次将$a$中的频率$f_{i,ser}$插入$F_{i,list}$,并对$F_{i,list}$就地升序排列,$f_{i,ser}$在$F_{i,list}$中的顺序即为$f_{i,ser}$在$F_{i,list}$中的索引$a_{i,id}$,故$f_{i,ser}$对应输出层神经元的索引为$(i-1)\times\text{sec}+a_{i,id}$.

ii)对于神经网络预测动作$a=(f_{1,ser},\cdots f_{i,ser},\cdots,f_{K,ser})$,直接输出动作$a$对应的输出层神经元索引即可.

计算整个系统的消耗tc,由式(10a)可得.$ac$为MEC服务器的可用计算容量,由式(22)可得.

$\varepsilon$的计算方式如下:

$$\varepsilon=\varepsilon_{end}+(\varepsilon_{init}-\varepsilon_{end})*e^{-step/\varepsilon_{const}} \tag{30}$$

其中,$\varepsilon_{end}$为收敛随机率,$\varepsilon_{init}$为初始随机率,$\varepsilon_{const}$为随机率常数.

2)根据动作$a$计算下一个状态$s'=(tc',ac')$.若$ac'<0$则标志位$end=$False否则$end=$True,奖励$r$,将$(s,s',r,a_{id},end)$存入$SumTree$,并进行状态迭代:$s=s'$.其中,奖励$r$的计算公式如下:

$$r=1-\frac{tc}{tc_{\text{fulllocal}}} \tag{31}$$

3)如果$tc<tc_{best}$,则$tc_{best}=tc$,$f_{ser,best}=a$.

4)判断$step>V$,若是则进入下一步,若否返回1)

5)从$SumTree$中抽取$m$个样本训练神经网络$Q$,并更新每个样本的优先级,样本优先级更新方式如下:

$$p_y=loss_y+0.0001 \quad y\in V \tag{32}$$

其中,$loss_y$为样本y的损失值.

6)判断$step\%C=0$,若是则进入步骤8),否则进入步骤9).

7)同步当前网络$Q$与目标网络$Q'$的权重:$w'=w$.

8)判断$end=True\ or\ step\%T=0$,若是则$epoch=epoch+1$并进入步骤5,否则返回步骤2).

其算法步骤如表4所示:

**表4 MMOA-DQN算法**

**算法4.** 基于DQN的求解所有边缘设备的服务器资源分配算法:

输入:迭代步长$T$,采样权重系数$\beta$,奖励性衰变系数$\gamma$,探索率$\epsilon$,当前网络$Q$,目标网络$Q'$,参数更新频率$C$,批量梯度下降的样本数$m$,$SumTree$的叶子节点数$V$

输出:服务器资源分配:$f_{ser,best}=\{f_{1,ser},\cdots,f_{K,ser}\}$

1. 随机生成一个$\eta\in(0,1)$
   IF $\eta<\epsilon$
   生成一个随机动作
   $a=\{(f_{1,ser},\cdots f_{i,ser},\cdots,f_{K,ser})\,|0\leqslant f_{i,ser}\leqslant 2f_{i,base}\}$
   ELSE
   将状态$s=(tc,ac)$输入目标网络$Q'$,预测出动作$a$.输出动作$a$对应的输出层神经元索引$a_{id,i}$.
2. 根据动作$a$计算下一个状态$s'=(Val,ac)$
   IF $ac<0$
   $end=$True
   ELSE
   $end=$False
3. 由式(31)计算奖励$r$,将$(s,s',r,a_{id},end)$存入$SumTree$,并进行状态迭代:$s=s'$
4. IF $tc<tc_{best}$
   $tc_{best}=tc$,$f_{ser,best}=a$
5. IF $step<=V$ 返回步骤1
6. 从$SumTree$中抽取$m$个样本训练神经网络$Q$,并由式(32)更新每个样本的优先级
7. IF $step\%C=0$
   同步当前网络$Q$与目标网络$Q'$的权重:$w'=w$
8. IF $end=$True $or\ step\%T=0$
   $step=step+1$,并回到步骤6
   ELSE
   返回步骤2

## 4 实验结果与分析

### 4.1 实验参数设置

本实验的关键参数默认值设置如表5所示:

**表 5　仿真参数设置**

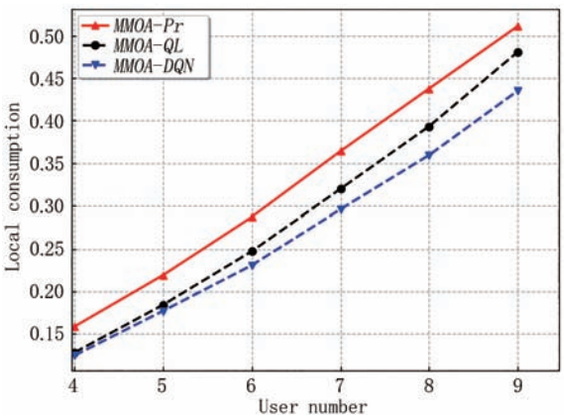
| 符号 | 值 | 定义 |
|---|---|---|
| $K$ | 5 | 边缘设备数 |
| $N$ | 40 | 每个边缘设备包含的任务数 |
| $(L_{\min},L_{\max})$ | (100, 200) | 边缘设备和服务器之间距离范围(m) |
| $L_0$ | 1 | 相对距离(m) |
| $\gamma$ | 0.9 | 折扣率 |
| $\varepsilon_{init}$ | 0.8 | 初始随机率 |
| $\varepsilon_{end}$ | 0.1 | 最后随机率 |
| $\varepsilon_{\text{const}}$ | 500 | 随机率常数 |
| $C$ | 100 | 每轮最大步长 |
| $M$ | 100 | 算法最大循环轮数 |
| $h$ | 512 | 隐藏层神经元个数 |
| $p$ | (10, 100) | 卸载传输功率范围*(mw)* |
| $N_0$ | −174 | 环境高斯白噪声功率密度*(dBm/Hz)* |
| $lr$ | $5\times10^{-4}$ | 学习率 |
| $wd$ | $5\times10^{-3}$ | 梯度衰减系数 |
| $T$ | 100 | 硬更新频率 |
| $m$ | 64 | 训练随机样本数 |
| $W$ | $10^7$ | 传输总带宽(Hz) |
| $F$ | 7 | 边缘服务器总频率*(GHz)* |
| $\delta_{i,L}$ | $(4.13,66.16)\times10^{-27}$ | 边缘设备$U_i$有效电容系数范围 |
| sec | 64 | 每个用户对应的输出神经元个数 |
| $tc_{best}$ | 100 | 初始目标值 |
| $V$ | 256 | *Sumtree*叶子节点数 |
| $f_{i,user}$ | (0.05,2) | 边缘设备CPU频率范围*(GHz)* |
| $\beta$ | 0.4 | *Sumtree*采样权重系数 |

### 4.2　实验结果与分析

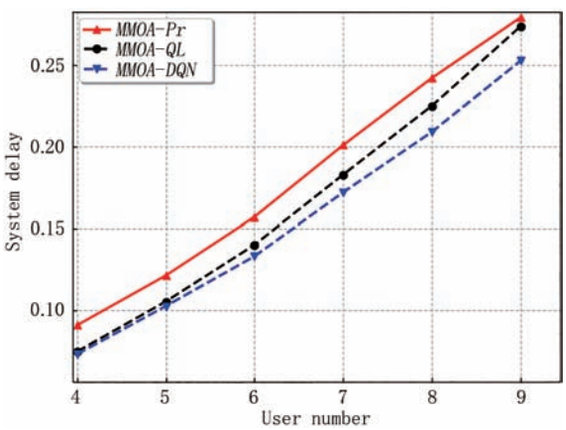
在本文中提出了三种算法MMOA-Pr，MMOA-QL和MMOA-DQN，根据实验结果分析所提出三种算法的优化性能.

图4中比较了本地消耗和系统延迟随用户数的变化情况.设置服务器频率$F$=7 GHz，用户频率$f_{i,user}$=0.5 GHz，任务数$N$=40.从图4(a)和图4(b)中可以看出，随着用户数的增加，MMOA-DQN算法始终保持最好的性能.MMOA-DQN算法相对于MMOA-QL算法而言，使用了深度神经网络代替$Q$表，从而使搜索范围更大更精确；而MMOA-QL算法中，$Q$表的搜索范围有限，故效果相对不如MMOA-DQN算法.而相对前两种强化算法而言，MMOA-Pr算法仅进行了一次卸载决策与卸载调度，故优化效果不如前两种算法明显.此外，随着边缘设备数增加而系统资源有限，本地消耗和系统延迟也随着用户数增加而增加.

图5比较了服务器频率变化时的本地能量消耗和系统延迟的变化情况.设置用户数$K$=5，任务数




(a)　不同用户数下本地消耗

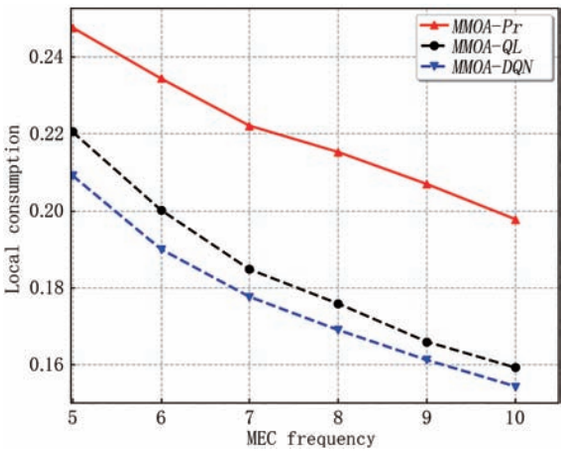



(b)　不同用户数下系统延迟

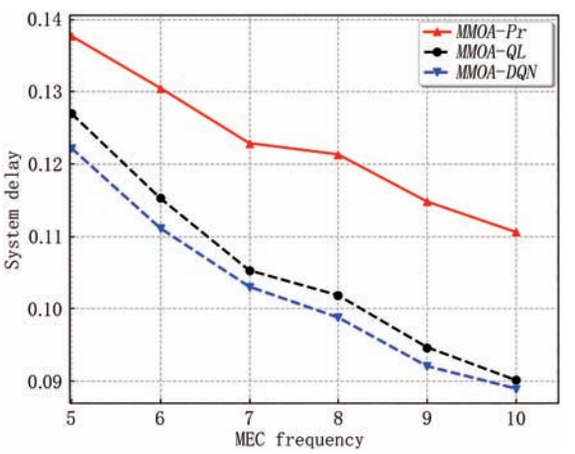
图4　不同用户数对系统性能的影响

$N$=40时，用户频率$f_{i,user}$=0.5 GHz.随着服务器总频率的增加，系统延迟和本地消耗均降低.MMOA-DQN算法由于采用了深度神经网络，始终保持最好的性能，在降低本地消耗的同时显著降低系统延迟.MMOA-DQN算法相对于MMOA-QL算法而言，使用了深度神经网络代替$Q$表，从而使搜索范围更大更精确，故效果也更好；而MMOA-QL算法中，由于$Q$表的大小有限从而搜索范围也有限，故在系统消耗和延迟优化方面优化效果相对MMOA-DQN算法较弱.而相对前两种强化算法而言，MMOA-Pr算法仅进行了一次卸载决策与卸载调度，故优化效果不如前两种算法明显.随着MEC服务器的CPU频率增加，三个算法的本地消耗和系统延迟均逐渐降低，是因为随着服务器总频率的增加，服务器为每个用户分配了更大的频率从而可以卸载更多的任务到服务器执行，此时系统延迟也随之逐渐降低.

图6比较用户频率变化时的本地能量消耗和系统延迟的变化情况.设置用户数$K$=5，服务器频率$F$=7 GHz，任务数$N$=40.此时，用户频率较低，大量任务被卸载到了服务器，因此本地消耗主要是执行延迟.

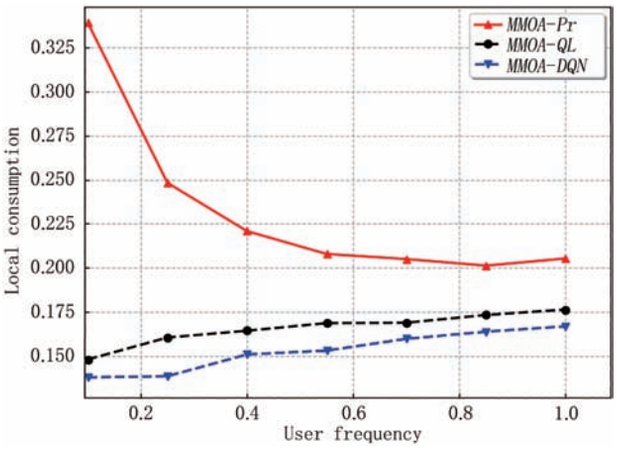



(a) 不同服务器频率下本地消耗

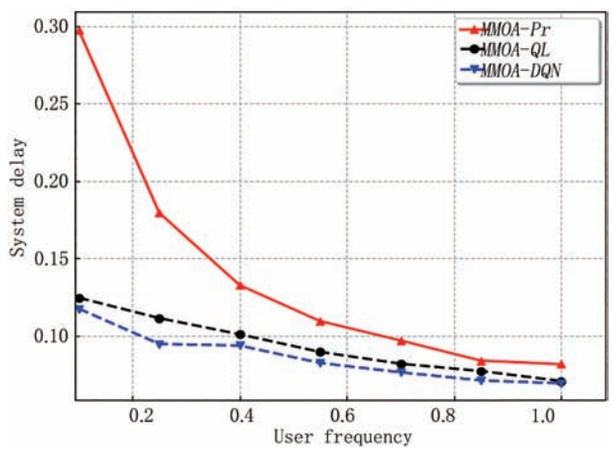



(b) 不同服务器频率下系统延迟

图5 不同服务器频率对系统性能的影响




(a) 不同用户频率下本地消耗




(b) 不同用户频率下系统延迟

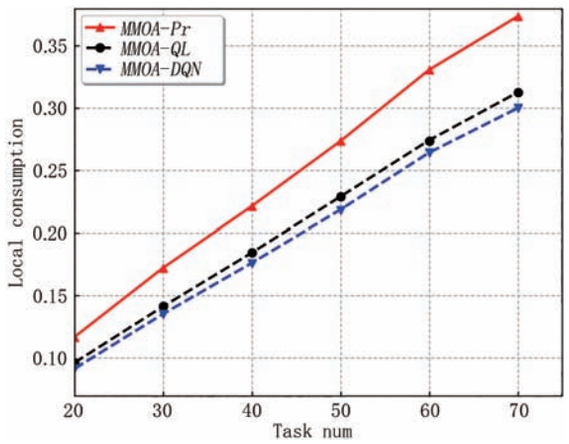
图6 不同用户频率对系统性能的影响

MMOA-Pr算法仅进行了一次调度，系统延迟较大. MMOA-QL和MMOA-DQN算法经过多次调度和频率分配，大幅度降低了系统延迟，因此当用户频率较低时系统延迟较低. 随着用户频率增加，本地任务持续增加，系统延迟降低、本地耗能增加，MMOA-DQN和MMOA-QL的本地消耗开始增加. 由于预分类只有一次调度，随着用户频率增加，边缘设备有效电容系数较小从而系统延迟降低速度大于能量增加速度，故本地消耗持续降低. MMOA-DQN算法相对于MMOA-QL算法而言，使用了深度神经网络代替$Q$表，从而使搜索范围更大更精确；而相对前两种强化算法而言，MMOA-Pr算法仅进行了一次卸载决策与卸载调度，故优化效果不如前两种算法明显.

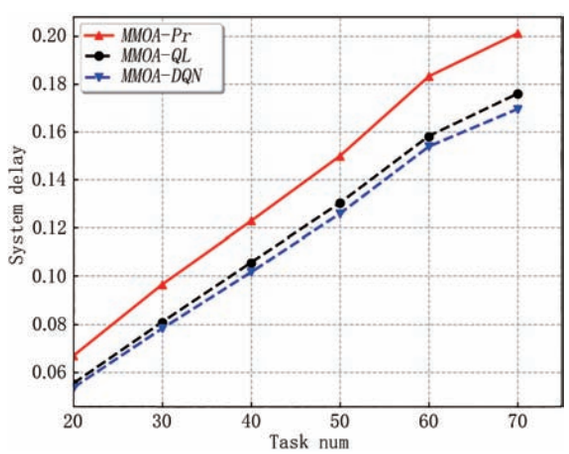
图7比较了任务数变化时的本地能量消耗和系统延迟的变化情况. 设置用户数$K$=5，用户频率$f_{i,user}$=0.5 GHz，服务器频率$F$=7 GHz. MMOA-DQN算法相对于MMOA-QL算法而言，使用了深度神经网络代替$Q$表，从而使搜索范围更大更精确. 而相对前两种强化算法而言，MMOA-Pr算法仅进行了一次卸载决策与卸载调度，故优化效果不如前两种算法明显. 随着任务数增加而服务器可分配资源有限，故本地消耗和系统延迟逐渐增加，但MMOA-DQN算法始终保持了最好的性能.

## 5 总 结

本文研究了多用户多任务移动边缘计算中的任务卸载调度与服务器资源分配问题. 在多用户MEC系统中，我们建模了一个以边缘设备低耗能、低延迟为目标的任务卸载调度和服务器频率分配问题. 由于这是一个非凸混合整数优化问题，为了最大限度地减少执行延迟和能量消耗，提出了一种基于深度强化学习的优化算法. 实验仿真分析了各变量在变化过程中对系统耗能和系统延迟的影响. 大量的实验也验证了所提出的MMOA-DQN算法在不同网络环境下都能有效提高系统能量效率.




(a)　不同任务数下本地消耗




(b)　不同任务数下系统延迟

图7　不同任务数对系统性能的影响

**致　谢**　我们向对本文提出宝贵建议的审稿专家及参与本文内容讨论的所有老师和同学表示衷心的感谢.

## 参　考　文　献

[1] F. Spinelli and V. Mancuso, Toward Enabled Industrial Verticals in 5G: A Survey on MEC-Based Approaches to Provisioning and Flexibility. IEEE Communications Surveys & Tutorials ,2021, 23(1):596-630

[2] S. Lee, S. Lee and M. -K. Shin. Low Cost MEC Server Placement and Association in 5G Networks//Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea (South), 2019: 879-882

[3] Y. Zhuang, X. Li, JiH. and H. Zhang. Optimization of Mobile MEC Offloading with Energy Harvesting and Dynamic Voltage Scaling//Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 2019: 1-6

[4] B. Liu, C. Liu and M. Peng. Resource Allocation for Energy-Efficient MEC in NOMA-Enabled Massive IoT Networks. IEEE Journal on Selected Areas in Communications, 2021, 39 (4): 1015-1027

[5] Z. Xiaoet al. Vehicular Task Offloading via Heat-Aware MEC Cooperation Using Game-Theoretic Method. IEEE Internet of Things Journal, 2020,37(3): 2038-2052

[6] J.Wu, Z.Cao, Y.Zhang and X.Zhang. Edge-Cloud Collaborative Computation Offloading Model Based on Improved Partical Swarm Optimization in MEC//Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China, 2019: 959-962

[7] T. Wang, S. Jie, B. Jiang, S. K. Bose and G. Shen. Distributed Backup Resource Allocation in Fiber-Wireless (FiWi) Access Networks Supporting Mobile Edge Computing //Proceedings of the 2019 21st International Conference on Transparent Optical Networks (ICTON), Angers, France, 2019: 1-4

[8] Z. Ding, J. Xu, O. A. Dobre and H. V. Poor. Joint Power and Time Allocation for NOMA-MEC Offloading. IEEE Transactions on Vehicular Technology, 2019,68(6): 6207-6211

[9] Y. Yang, X. Chen, Y. Chen and Z. Li. Green-Oriented Offloading and Resource Allocation by Reinforcement Learning in MEC //Proceedings of the 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Tianjin, China, 2019: 378-382

[10] Y. Mao, J. Zhang, and K. B. Letaief, . Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems //Proceedings of the IEEE Wireless Communication Network Conference (WCNC), Berlin, Germany, 2017: 1-6

[11] N. Janatian, I. Stupia, and L. Vandendorpe. Optimal resource allocation in ultra-low power fog-computing SWIPT-based networks //Proceedings of the IEEE Wireless Communications Network Conference (WCNC), Marrakech, Morocco, 2018: 1-6

[12] Z. Ning, P. Dong, X. Kong and F. Xia. A Cooperative Partial Computation Offloading Scheme for Mobile Edge Computing Enabled Internet of Things. IEEE Internet of Things Journal, 2019,6(3): 4804-4814

[13] M.Qin, L.Chen, N.Zhao, Y.Chen, F.R.Yu and G.Wei. Power-Constrained Edge Computing With Maximum Processing Capacity for IoT Networks. IEEE Internet of Things Journal, 2019,6(3): 4330-4343

[14] P. Yang, L. Li, W. Liang, H. Zhang and Z. Ding. Latency Optimization for Multi-user NOMA-MEC Offloading Using Reinforcement Learning //Proceedings of the 2019 28th Wireless and Optical Communications Conference (WOCC), Beijing, China, 2019: 1-5

[15] Y. Zhang, X. Zhou, Y. Teng, J. Fang and W. Zheng. Resource Allocation for Multi-User MEC System: Machine Learning Approaches //Proceedings of the 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, USA, 2018: 794-799

[16] W. Zhou, L. Lin, J. Liu, D. Zhang and Y. Xie. Joint Offloading Decision and Resource Allocation for Multiuser NOMA-MEC Systems. IEEE Access, 2019,(7): 181100-181116

[17] M. Sheng, Y. Dai, J. Liu, N. Cheng, X. Shen and Q. Yang. Delay-Aware Computation Offloading in NOMA MEC Under

Differentiated Uploading Delay. IEEE Transactions on Wireless Communications, 2020,19(4): 2813-2826

[18] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu and H. Duan. Mobility-Aware Multi-User Offloading Optimization for Mobile Edge Computing. IEEE Transactions on Vehicular Technology, 2020,69(3): 3341-3356

[19] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio and G.Fortino. Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA. IEEE Access, 2020,8: 54074-54084

[20] Zhao, Q.Li, Y.Gong and K.Zhang. Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks. IEEE Transactions on Vehicular Technology, 2019,68(8): 7944-7956

[21] S. Jošilo and G. Dan. Joint Management of Wireless and Computing Resources for Computation Offloading in Mobile Edge Clouds. IEEE Transactions on Cloud Computing, 2021, 99: 1,doi: 10.1109/TCC.2019.2923768

[22] H. Yuan and M. Zhou. Profit-Maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems. IEEE Transactions on Automation Science and Engineering, 2021: 1, doi: 10.1109/TASE.2020.30009

[23] T.Zhao, S.Zhou, L.Song, Z.Jiang, X.Guo and Z.Niu. Energy-optimal and delay-bounded computation offloading in mobile edge computing with heterogeneous clouds. China Communications, 2020,17(5): 191-210

[24] Z. Kuang, L. Li, J. Gao, L. Zhao and A. Liu. Partial Offloading Scheduling and Power Allocation for Mobile Edge Computing Systems. IEEE Internet of Things Journal, 2019,6(4): 6774-6785

**KUANG Zhu-Fang**, Ph. D., Professor. His main research interests include Internet of Things, edge computing, big data analysis, artificial intelligence.

**CHEN Qing-Lin**, M. S. candidate. Her main research interest is edge computing.

**LI Lin-Feng**, M. S. Her main research interest is edge computing.

**DENG Xiao-Heng**, Ph. D., Professor. His main research interests include wireless communication and network, edge computing.

**CHEN Zhi-Gang**, Ph. D., Professor. His main research interests include network and distributed computing, edge computing.

**Background**

Mobile edge computing is an emerging technology that provides cloud and IT services to mobile users at close range. In mobile edge computing, the mobile edge server is also deployed on each base station, and the mobile edge computing platform reduces network latency by providing computing and storage capabilities to the edge devices. Mobile devices and IoT devices often utilize mobile edge computing services to perform computational offloading for computation-intensive applications, such as image processing, mobile gaming, etc. Existing domestic and foreign studies mainly consider multi-user MEC offloading with the goal of minimizing energy consumption or delay. However, in the multi-user MEC offloading system, task offloading scheduling has not been paid much attention. Essentially, joint offloading decision and offloading scheduling in multi-user MEC systems is a challenging problem.

In this paper, based on deep reinforcement learning, we propose a task offload scheduling and resource allocation algorithm for multi-user and multi-task. In the case of multi-user and multi-task offloading in mobile edge computing network, the offloading decision, task scheduling and server resource allocation are studied in order to minimize system delay, transmission energy consumption and local energy consumption. Simulation results show that the proposed method has superior performance in reducing system latency and local energy consumption.

The National Natural Science Foundation of China (No. 62072477, No. 61309027), The Hunan Provincial Natural Science Foundation of China (No. 2018JJ3888), The Scientific Research Fund of Hunan Provincial Education Department (No. 18B197), The National Key R&D Program of China (No. 2018YFB1700200), the Hunan Key Laboratory of Intelligent Logistics Technology (2019TP1015).