doi:10.3969/j.issn.1001-4616.2024.01.014

边端协同环境中的任务卸载和资源分配方法

张俊娜,赵　豪,李天泽,赵晓焱,王亚丽

(河南师范大学计算机与信息工程学院,河南 新乡 453007)

[**摘要**]　将终端任务卸载至边缘计算环境弥补了云计算距离较远而产生较大延迟的缺陷,同时还降低了设备能耗. 但从资源方面来讲,边缘服务器的各类资源并不像云服务器那么充足,因此,任务卸载和资源分配的联合优化成为边缘计算的研究热点之一. 已有的任务卸载和资源分配联合优化研究通常假设任务卸载至单个边缘服务器,默认每个终端设备产生一个任务,即使有研究多服务器的,也通常忽略服务器间的负载均衡. 为此,本文在一个多边缘服务器多用户多任务的边端系统中,提出了一种权衡时延、能耗和负载均衡指标(即效益)的任务卸载和资源分配方法,其通过优化任务卸载决策、服务器计算资源分配和终端设备发射功率,实现任务卸载效益最大化. 最后,为了验证所提方法的有效性,进行了充分的对比实验. 实验结果表明,与对比方法相比,所提出的方法在提升卸载效益和实现服务器间负载均衡方面有良好的性能.

[**关键词**]　边缘计算,任务卸载,资源分配,负载均衡,强化学习
[**中图分类号**]TP181　[**文献标志码**]A　[**文章编号**]1001-4616(2024)01-0121-12

Joint Task Offloading and Resource Allocation Method Based on Multi-Objective Optimization

Zhang Junna, Zhao Hao, Li Tianze, Zhao Xiaoyan, Wang Yali

(College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China)

Abstract: Offloading terminal tasks to the edge computing environment makes up for the large delay caused by the distance of cloud computing, and reduces the power consumption of equipment. But in terms of resources, all kinds of resources of edge server are not as sufficient as that of cloud server. Therefore, the joint optimization of task offloading and resource allocation becomes one of the research hotspots of edge computing. Existing studies on joint optimization of task offloading and resource allocation generally assume that tasks are offloading to a single edge server; By default, each terminal device generates one task. Even when multiple servers are considered, load balancing between servers is often ignored. Therefore, in a multi-edge server, multi-user and multi-task edge system, this paper proposes a task offloading and resource allocation method that balances delay, energy consumption and load balancing index (i.e., benefit). By optimizing task offloading decision, server computing resource allocation and terminal device transmission power, the benefit of task offloading can be maximized. In order to verify the effectiveness of the proposed method, a full comparison experiment is carried out. The experimental results show that, compared with the comparison method, the proposed method has good performance in improving the unloading efficiency and realizing the load balancing between servers.

Key words: edge computing, task offloading, resource allocation, load balancing, reinforcement learning

边缘计算将资源下沉到网络边缘,把边缘服务器部署在蜂窝基站附近或通用计算平台的本地接入点上,为用户提供低时延和高带宽的网络服务. 边缘服务器支持终端设备将其任务卸载执行,并为其分配资源,解决终端设备因计算资源和电池容量有限而存在的不足,同时降低任务执行的时延和能耗[1].

现有许多学者对任务卸载和资源分配的联合优化进行了研究[2-5],但也存在一些局限性:

边端系统中只考虑单个服务器. 张永棠[6]在一个多用户边缘系统中,通过优化计算卸载策略和资源分配策略,使延迟总成本和能耗最小. 邝祝芳等[7]研究了移动边缘计算中多用户多任务卸载,提出了一种双层算法,在上层使用基于贪心策略的流水车间调度算法解决任务卸载决策和资源调度问题,在下层使用

收稿日期:2023-05-31.
基金项目:国家自然科学基金项目(62072159)、河南省科技攻关项目(232102211061、222102210011).
通讯作者:张俊娜,博士,副教授,研究方向:边缘计算、神经网络和服务计算. E-mail:jnzhang@htu.edu.cn

强化学习方法优化服务器资源分配问题. 吴学文等[8]在一个云边协同的多用户系统中,提出了一种基于博弈论的资源分配和任务卸载方案,通过优化计算资源分配、上行功率分配和任务卸载决策,使时延、能耗和计算成本最小化. 熊兵等[9]在一个动态的边缘计算系统环境中,结合射频无线电传播方案,提出一种基于深度强化学习的任务卸载与资源分配联合优化方法. 上述文献在边缘层均只考虑单个服务器,而在现实场景中,人群密集处往往需要部署大量服务器,以满足多个用户的任务卸载需求.

边端系统进行任务卸载和资源分配时,只优化时延或能耗. Zhou 等[3]为最大限度地降低超密集物联网系统的能耗,平衡网络负载,提出了一种改进的层次自适应搜索算法进行求解. Xu 等[10]研究基于 NOMA 异构边缘网络中的任务卸载和资源分配问题,并将问题解耦为卸载决策和资源分配两个子问题,通过优化任务卸载决策、本地 CPU 频率调度、功率控制、计算资源和子通道资源分配,有效降低用户能耗. 田贤忠等[11]研究多边缘服务器网络模型,为了充分利用边缘服务器的资源,提出了基于量子粒子群算法的边缘节点卸载算法和基于启发式算法的边缘节点负载均衡算法,将热点区域的数据以多跳的方式卸载到邻近的边缘服务器. 上述文献只考虑单一目标优化,不能满足对任务执行时延和终端设备能耗的共同需求.

忽略负载均衡. Zhang 等[12]研究了支持边缘计算的多单元无线网络,其中每个基站都配备了边缘服务器;Yang 等[13]考虑了多服务器边缘计算网络,并将卸载决策问题表述为多类分类问题,将计算资源分配问题表述为回归问题;在此基础上,设计并训练了基于多任务学习的前馈神经网络模型,共同优化卸载决策和计算资源分配. Zhou 等[14]研究边缘服务器协作场景下的联合任务卸载和资源分配问题. 从用户体验角度定义物联网设备间的资源公平,在考虑系统效率和公平性的前提下,提出一种两级算法. Wang 等[15]研究多服务器场景,构造一个联合任务卸载、功率分配和资源分配的优化问题,并提出一种高效的多目标进化算法. 虽然上述文献考虑了多服务器和多目标优化,但没有考虑服务器间的负载均衡问题,可能会导致单个服务器过载,其他服务器空闲的情况[12,16].

为了解决上述局限性,本文在多边缘服务器多用户多任务边缘环境中,提出了一种基于差分进化和近端策略优化(differential evolution and proximal policy optimization,DEPPO)的任务卸载和资源分配方法. 并通过优化任务卸载决策、服务器计算资源分配和终端设备发射功率,实现任务卸载效益最大化. 本文主要贡献如下:

(1)在边缘服务器资源有限的约束下,建立了网络模型、时延能耗模型、负载模型和效益模型,采用线性加权的方法将对时延、能耗和负载均衡的多目标优化转化为对效益函数的单目标优化问题;

(2)针对资源分配问题,通过对边缘服务器和不同的资源类型进行编码,采用差分进化算法求得资源分配决策,从而确定任务的最优资源分配方案;

(3)针对卸载问题,在资源和负载约束下,通过强化学习算法探索最优的卸载决策,使任务卸载效益最大化;

(4)基于真实数据,将本文方法与其他方法进行充分实验对比. 实验结果表明,本文方法具有更好的性能.

1　系统模型和问题描述

1.1　网络模型

本文考虑一个多边缘服务器、多用户、多任务的边缘计算卸载场景. 如图 1 所示,该区域内有若干基站和若干终端设备,部分基站配备边缘服务器用于给计算能力弱的边缘设备提供计算服务,只要在该基站覆盖范围内的用户都可通过基站将其计算任务卸载到边缘服务器上执行. 基站还可以充当任务转发器,该基站覆盖范围内终端用户所产生的任务通过该基站转发,卸载到其他基站所配备的边缘服务器上执行. 每个终

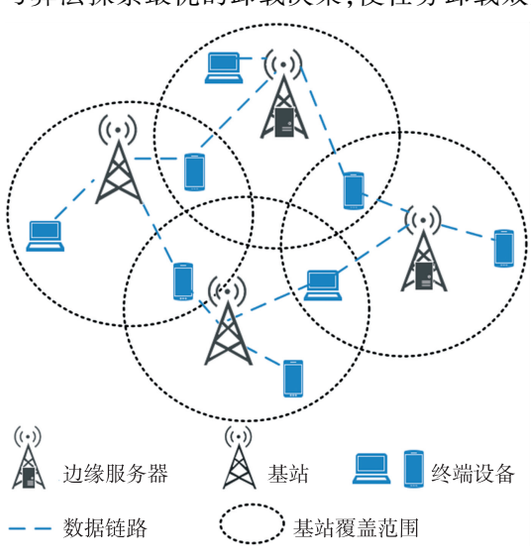


图 1　网络模型

Fig. 1　Network model

端设备都有若干个独立任务,这些任务可以在本地执行,也可以卸载到任一边缘服务器,由边缘服务器处理并将计算结果回传到终端设备.

假设该区域内存在 m 个边缘服务器,用集合 S 表示为 $S=\{s_1,s_2,\cdots,s_m\}$;其中 s_i 指第 i 个边缘服务器且该服务器的属性表示为 $s_i=(f_i^s,f_{i,r}^s,m_i^s,m_{i,r}^s,p_i^s)$,$f_i^s$ 表示边缘服务器 i 最大的 CPU 频率,$f_{i,r}^s$ 表示边缘服务器 i 当前可用的 CPU 频率,m_i^s 表示边缘服务器 i 的最大存储容量,$m_{i,r}^s$ 表示边缘服务器 i 当前可用的存储容量,p_i^s 表示边缘服务器 i 的计算功率. 有 n 个用户,用集合 U 表示为 $U=\{u_1,u_2,\cdots,u_n\}$. 其中 u_i 指第 i 个终端设备且该终端设备的属性表示为 $u_i=(f_i^u,m_i^u,m_{i,r}^u)$,$f_i^u$ 表示该终端设备 i 的 CPU 频率,m_i^u 表示终端设备 i 的存储空间,$m_{i,r}^u$ 表示终端设备 i 当前可用的存储空间. 假设终端设备 i 产生 k 个任务,用 t_i 表示为 $t_i=\{t_i^1,t_i^2,\cdots,t_i^k\}$,其中 t_i^k 表示终端设备 i 产生的第 k 个任务,任务特征可以表示为 $t_i^k=(d_i^k,c_i^k)$,c_i^k 表示计算该任务所需要的 CPU 周期数,一般以 CPU 频率(Hz)为单位,d_i^k 表示任务数据量大小,一般以比特(bit)为单位;最后将所有终端设备产生的任务用集合 T 表示为 $T=\{t_1,t_2,\cdots,t_n\}$,并用表 1 总结了在本文中使用的主要符号.

表 1　主要符号

Table 1　Key symbols

符号	意义	符号	意义
S	边缘节点集合	U	终端设备集合
T	所有任务集合	t_i	终端设备产生的任务集合
d_i^k	终端产生的任务的数据量	c_i^k	终端产生的任务所需计算周期数
f_i^u	终端设备的计算资源	m_i^u	终端设备的存储空间
$m_{i,r}^u$	终端设备的可用存储空间	f_i^s	边缘服务器的计算资源
$f_{i,r}^s$	边缘服务器的可用计算资源	m_i^s	边缘服务器的最大存储空间
$m_{i,r}^s$	边缘服务器的可用存储空间	p_i^{com}	边缘服务器的计算功率
B	带宽	$t_{i,k}^{\mathrm{wait}}$	任务等待时间
$h_{i,m}$	信道增益	N_0	噪音

1.2　时延能耗模型

如 2.1 所述,本文考虑任务在本地执行或边缘服务器执行,但在边缘服务器执行分为两种情况. 因此,本节先介绍任务在本地执行的时延和能耗模型,然后介绍任务卸载到边缘服务器时两种不同情况的时延和能耗模型.

(1)本文考虑任务在本地的执行模式为串行模式. 假设终端设备 i 产生的第 k 个任务 t_i^k 在本地执行,则终端设备 i 把所有的计算资源都用于执行该任务. 则本地计算时延可表示为:

$$t_{i,k}^{\mathrm{com}}=\frac{c_i^k}{f_i}, \tag{1}$$

其中 f_i 为终端设备 i 的计算能力,c_i^k 为任务 t_i^k 所需的计算周期数.

若任务 t_i^k 不是第一个在终端设备 i 上执行的任务,则需要等待在它之前卸载的任务执行完才能执行. 假设 t_i^j 在 t_i^k 之前卸载在本地执行,则 t_i^k 的等待时间可以表示为:

$$t_{i,k}^{wait}=\sum t_{i,j}^{\mathrm{com}},\quad t_i^j\in t_i, \tag{2}$$

其中 $t_{i,j}^{\mathrm{com}}$ 为任务 t_i^j 在本地的计算时延,t_i 为终端设备 i 产生的任务集合.

为了准确描述任务在终端设备执行的能耗,本文引用了广泛采用的能耗模型 $\varepsilon=kf^2c$[17],其中 k 是与 CPU 架构相关的能耗参数. 因此任务在本地计算的能耗表示为:

$$\mathrm{e}_{i,k}^{l}=k(f_i)^2c_i^k. \tag{3}$$

(2)终端设备将其任务卸载到边缘服务器执行会产生以下两种卸载情况:一种是产生该任务的终端设备在执行该任务的边缘服务器覆盖范围内;另一种情况是产生该任务的终端设备不在执行该任务的边缘服务器覆盖范围内,该情况下任务在上传过程中需要借助基站进行数据转发,基站在转发数据时不进行保存.

①终端设备在边缘服务器覆盖范围内,任务卸载时延主要包含以下几方面:一是任务从终端设备上传到边缘服务器产生的时延;二是任务在边缘服务器上的处理时延;三是结果回传时延,但由于结果的数据量相对于上传数据量较小,所以一般不考虑回传时延.

假设对单个基站覆盖范围内的所有终端设备,采用正交频分多址技术[14]作为上行链路的接入方案,并为每个任务分配一个带宽 B 的子载波. 根据香农公式[17],将任务 t_i^k 卸载至边缘服务器 m 的上传速率可以表示为:

$$R_{i,k}^m = B\log_2\left(1+\frac{p_{i,k}^{\text{tran}}h_{i,m}}{N_0}\right), \tag{4}$$

其中 B 是信道带宽,$p_{i,k}^{\text{tran}}$ 是终端设备 i 的传输功率,$h_{i,m}$ 是终端设备 i 和边缘服务器 m 之间的信道增益,N_0 表示噪声功率. 结合所求传输速率,将任务卸载到边缘服务器 m 所需的时延表示为:

$$t_{i,k}^{\text{up}} = \frac{d_i^k}{R_{i,k}^m}, \tag{5}$$

其中 d_i^k 表示任务 t_i^k 的数据量. 上传过程中,终端设备 i 产生的能耗为:

$$\mathrm{e}_{i,k}^{\text{up}} = t_{i,k}^{\text{up}}p_{i,k}^{\text{tran}}. \tag{6}$$

本文考虑边缘服务器的工作方式为并行模式,当任务卸载到边缘服务器时,边缘服务器为其分配一定的计算资源来执行该任务,因此任务 t_i^k 在边缘服务器上的计算时延为:

$$t_{i,k}^{\text{exe}} = \frac{c_i^k}{f_{i,k}^m}, \tag{7}$$

其中 c_i^k 为任务 t_i^k 所需的计算周期数,$f_{i,k}^m$ 为边缘服务器 m 分配给任务 t_i^k 的计算资源. 当该任务在边缘服务器执行完,边缘服务器 m 产生的能耗为:

$$\mathrm{e}_{i,k}^m = \frac{c_i^k}{f_{i,k}^m}\cdot P_m^{\text{com}}, \tag{8}$$

其中 $f_{i,k}^m$ 为边缘服务器 m 分配给任务 t_i^k 的计算资源,P_m^{com} 为边缘服务器 m 的计算功率.

②终端设备不在边缘服务器覆盖范围内,任务的卸载延迟除了第一种卸载情况所包含的三种时延外,还有任务借助基站转发所产生的转发延迟. 假设任务 t_i^k 卸载到边缘服务器 m' 执行,任务上传过程中需要经过基站 m(产生任务 t_i^k 的终端设备在基站 m 的覆盖范围内)进行转发. 在此过程中产生的转发时延可表示为:

$$t_{i,k}^{m'} = \frac{d_i^k}{R_m^{m'}}, \tag{9}$$

其中 $R_m^{m'}$ 表示转发基站 m 与边缘服务器 m' 之间的发送速率.

1.3　负载模型

在边缘计算环境中,任务卸载不均衡会造成通信链路堵塞,边缘层资源不能充分利用等不足. 为了提高资源利用率,本文考虑边缘服务器间的负载均衡问题,首先通过卸载到边缘服务器的数据量定义服务器的负载率:

$$l^s = \frac{m_i^s - m_{i,r}^s}{m_i^s}, \tag{10}$$

其中 m_i^s 表示边缘服务器 i 的最大存储容量,$m_{i,r}^s$ 表示边缘服务器 i 的当前可用存储容量.

然后将各服务器负载率的标准差作为衡量负载均衡的指标,该指标越小,说明边缘服务器间的负载越相近,可以避免单个边缘服务器负载过高的情况,负载均衡指标定义如下:

$$\sigma^s = \sqrt{\frac{\sum_{i=1}^{m}(l_i^s - \overline{l_i^s})^2}{m}}, \tag{11}$$

其中 $\overline{l_i^s}$ 为各边缘节点资源使用率的平均数:$\overline{l_i^s} = \dfrac{\sum_{i=1}^{m} l_i^s}{m}$.

1.4 卸载效益

本文综合考虑了任务的执行时延、能耗和服务器负载均衡,通过定义任务的效益函数来衡量任务执行的满意程度. 任务在边缘服务器上执行的时间和能耗相对于本地执行的改善分别用$(1-T_S/T_L)$和$(1-E_S/E_L)$来表示. 当$T_S<T_L$时,效益为正,且T_S与T_L的差值越大,效益越大;当$T_L<T_S$时,效益为负,且T_L与T_S的差值越大,效益越小;要想使效益最大化,应当将任务卸载到执行时间小于在本地执行时间的边缘服务器上. 任务在边缘服务器执行时间相对于本地执行时间越小,该任务的卸载效益越大,与本文目标一致. 类似地,将卸载能耗与本地能耗比较,能耗的相对改善程度越高,任务卸载效益越高. 而对于边缘服务器负载情况,可直接用负载均衡指标来衡量. 综上,任务卸载到边缘服务器的效益函数可以表示为:

$$I_{i,k}=\alpha(1-T_S/T_L)+\beta(1-E_S/E_L)-\gamma\sigma^s, \tag{12}$$

其中T_L、E_L为任务在本地执行的时间和能耗,T_S、E_S为任务在边缘服务器执行的时延和能耗,σ^S为边缘节点的负载均衡指标. α、β、γ为偏好因子,分别表示系统对时延、能耗和负载均衡的偏好程度,满足α、β、$\gamma\in[0,1]$,$\alpha+\beta+\gamma=1$.

1.5 问题描述

本文在边缘服务器计算资源和存储资源有限的情况下,提出了一种权衡时延、能耗和负载均衡指标的任务卸载和资源分配方法. 通过优化任务卸载决策、服务器计算资源分配和终端设备发射功率,将边缘系统中任务卸载所产生的效益最大化,基于上述分析,可以将问题形式化为:

$$\begin{aligned}
&\max_{X,F,P}\sum_{i=1}^{n}\sum_{k=1}^{q}I_{i,k}\\
&\text{s.t. } C1:x_i^k\in\{0,1,\cdots,m\},\forall m\in S,(i,k)\in T\\
&\quad C2:f_{i,k}^m>0,\forall m\in S,(i,k)\in T\\
&\quad C3:\sum_{x_i^k=m}f_{i,k}^m<=f^m,\forall m\in S,(i,k)\in T\\
&\quad C4:0<p_{i,k}^{\text{tran}}<=P_{\max}^{\text{tran}},\forall(i,k)\in T\\
&\quad C5:d_i^k<m_{i,r}^m,\forall m\in S,(i,k)\in T\\
&\quad C6:0<\sigma^s,\forall m\in S
\end{aligned}\tag{13}$$

Ma 等[18]研究了多服务器多终端设备边缘环境下的任务卸载和资源分配问题,并证明该问题为MINLP 问题,将其分解为任务卸载和资源分配两个子问题求解. 与该参考文献研究问题相比,本文研究问题的优化目标增加了边缘服务器负载均衡指标,因此本文研究问题也属于 MINLP 问题,借鉴该参考文献研究方法,可将本文研究问题分解后采用不同的算法求解.

2 算法描述

为了解决本文研究问题,本节结合启发式算法和强化学习算法,提出一种基于差分进化和近端策略优化的任务卸载和资源分配方法(differential evolution and proximal policy optimization,DEPPO). 该方法分为两层,上层采用差分进化算法求解资源分配,下层采用近端策略优化算法求解任务卸载决策;该算法的执行过程主要分为 3 步:1)初始化一个随机卸载决策,然后在给定卸载决策的情况下采用差分进化算法求解资源分配方案. 2)结合上层求解的资源分配,采用近端策略优化算法求解任务的卸载决策. 3)经过不断迭代上下层算法,求解最优的任务卸载决策和资源分配.

2.1 基于差分进化的资源分配算法

差分进化是一种高效的全局优化算法,其进化流程与遗传算法类似. 但遗传算法在变异时产生的新基因可能与种群中已有个体基因重合,优化后期会陷入局部最优解;而差分进化算法能保证变异时产生的新基因与原有基因有一定距离,避免陷入局部最优. 因此,本节将采用差分进化算法求解任务的资源分配方案.

(1)问题编码

假设种群中有m个个体,表示为$X=\{x_1,x_2,\cdots,x_m\}$,其中x_i表示个体i基因向量. 为了将任务和两种资源类型映射到种群个体上,设每个个体有两条染色体f和p,即个体的基因向量x_i表示为$x_i=\{f_i,p_i\}$;同

时每条染色体上有多个基因,即 $f_i=\{f_i^1,f_i^2,\cdots,f_i^n\}$,$p_i=\{p_i^1,p_i^2,\cdots,p_i^n\}$. 将上述个体向量、染色体、基因体对应到本文环境中,个体向量 x_i 代表一个资源分配解,染色体 f_i 和 p_i 分别代表计算资源分配解和通信资源分配解,f_i^j 和 p_i^j 分别代表第 j 个任务的计算资源分配值和通信资源分配值.

(2)初始化

随机卸载决策初始化:现已知边缘服务器的编号为[1,2,3,⋯,m],则单个任务的卸载决策取值范围为[0,1,2,3,⋯,m],若取值为 0,表示该任务在本地终端执行,若取值非 0 整数,则表示该任务卸载到对应编号的边缘服务器上执行. 同时,为每个边缘服务器设置一个队列,将卸载到边缘服务器的任务都添加到对应队列中;以便判断边缘服务器的剩余存储空间能否满足任务需求(即约束条件 $C5$). 综上,个体初始化的卸载决策可表示为 $x=\{x_1,x_2,\cdots,x_n\}$,$x_i\in[0,m]$ 满足式(13)中的约束条件 $C1$、$C5$.

种群初始化:种群初始化主要工作是对个体的基因位点进行赋值. 对于个体 f 染色体上的基因位点,其值应大于 0,小于其对应任务卸载的边缘服务器剩余计算资源,以满足约束条件 $C2$、$C3$;对于个体 p 染色体上的基因位点,其值满足终端发射功率范围,以满足约束条件 $C4$. 另外,用 $x_i(g)$、$v_i(g)$ 和 $c_i(g)$ 表示个体的原有基因、变异基因和交叉基因,以区分进化过程中不同时间的基因信息.

(3)进化过程

变异:在差分进化算法中,要为每个个体生成一个变异基因. 该基因表示不同时间段,个体原有基因发生变异而产生的结果. 在实验过程中,当个体发生变异时,随机选出三个同代个体,然后对个体基因做如下操作:

$$V_i^{\beta_i^j/p_i^j}(g)=X_a^{\beta_i^j/p_i^j}(g)+F\cdot(X_b^{\beta_i^j/p_i^j}(g)-X_c^{\beta_i^j/p_i^j}(g)),\tag{14}$$

其中 g 表示第 g 代个体;a、b、c 表示不同个体的编号;F 为缩放比例因子. $X_i^f(g)$ 表示第 g 代的个体 i 的原有基因,$V_i^f(g)$ 表示第 g 代的个体 i 生成的变异基因. 另外,在变异过程中需要对生成的变异基因进行检测,看是否满足实验环境的要求,即 $V_i^f(g)$ 中每个分量的取值都应大于 0,满足约束条件 $C2$,$V_i^p(g)$ 中每个分量的取值都应满足约束条件 $C4$. 对于不满足条件的变异操作,就随机生成满足条件的值.

交叉:交叉是指让个体原有基因与其变异基因在一定概率下进行替换,具体的交叉位置随机确定. 交叉后要检测交叉基因是否满足环境约束,在本实验中,要求卸载到同一边缘服务器上的任务所分配的计算资源总和不能超出该边缘服务器的最大计算资源,以满足式(13)中的约束条件 $C3$;另外要求设备发射功率在一定范围内,以满足式(13)中的约束条件 $C4$,由于在种群初始化和变异过程中都添加了此约束,经过交叉操作后的基因位点值仍满足条件. 交叉操作可表示为:

$$C_i^{\beta_i^j/p_i^j}(g)=\begin{cases}X_i^{\beta_i^j/p_i^j}(g), & \text{random}(0,1)<CR \text{ or randint}(0,m)=d,\\ V_i^{\beta_i^j/p_i^j}(g), & \text{else},\end{cases}\tag{15}$$

其中 CR 为交叉概率;d 为随机整数,保证在交叉操作过程中至少有一个基因是来自变异操作产生的基因.

选择:差分进化算法在每代(即每次迭代)都会选出一个最优个体,与种群当前保存的历代最优个体比较,适应值小的个体留下;因此,用变量 *global_best_solution* 和 *global_best_fitness* 保存当前最优个体的资源分配决策和适应值. 在选择过程中,采用贪婪策略. 选择过程如下:

$$\begin{aligned}global\_best\_solution&=\begin{cases}C_i(g), if(F(C_i(g))<F(global\_best\_solution))\\ global\_best\_solution, else,\end{cases}\\ global\_best\_fitness&=\begin{cases}F(C_i(g)), if(F(C_i(g))<F(global\_best\_solution))\\ global\_best\_fitness, else.\end{cases}\end{aligned}\tag{16}$$

(4)适应度函数

在差分进化算法中,通过适应值衡量个体对环境的适应度. 结合本文研究问题,在边缘服务器资源约束下,权衡时延、能耗和负载均衡指标构建任务效益函数,最终目的是最大化效益,而差分进化算法本身在随机寻找最小值,因此定义差分进化算法的适应度函数与本文的目标函数负相关:

$$F(I_{i,k})=-\sum_{i=1}^{n}\sum_{k=1}^{q}(\alpha(1-T_S/T_L)+\beta(1-E_S/E_L)-\gamma\sigma^s).\tag{17}$$

算法 1　基于 DE 算法的资源分配

输入:随机生成的任务卸载决策 X(满足式(13)中的约束条件 $C1$、$C5$),种群大小 M,问题解的维度 D,迭代次数 T,功率最小值 $P_{\min}$、最大值 $P_{\max}$,边缘节点编号[0,7]

输出:资源分配方案 global_best_solution,其中包括分配给任务的计算资源 F,传输功率 P

1.初始化种群. (f 染色体上的基因位点值满足约束条件 $C2$、$C3$,p 染色体上的基因值满足约束 $C4$)

2.while $t<T$ do

3.　for $i=1$ to M do

4.　for $j=1$ to D do

5.　根据式(14),为每个个体生成变异基因. 并检测变异基因是否满足式(13)中的约束 $C2$、$C4$.

6.　根据式(15),对每个个体进行交叉操作,并检测生成的交叉基因是否满足式(13)中的约束 $C3$.

7.　end for

8.　if $F(C_i(g))<F(X_i(g))$

9.　$X_i(g)\leftarrow C_i(g)$,更新该个体的基因和适应值.

10.　if $F(C_i(g))<F(global\_best\_solution)$

11.　$global\_best\_solution\leftarrow C_i(g)$

12.　end for

2.2　基于近端策略优化的任务卸载算法

在上一节求解的资源分配方案确定后,本节将采用近端策略优化算法求解任务卸载决策. 近端策略优化算法是一种深度强化学习算法,通过智能体与环境交互产生数据,并用该数据训练智能体的网络模型,使智能体获得更大的回报. 下文先对状态、动作、奖励做了定义,然后描述了算法的实现过程.

(1)结合本文研究问题,对环境状态、智能体动作和奖励函数的定义如下:

状态:在本文中将某一时刻边缘服务器的计算、存储资源和卸载任务的数据量、所需计算周期定义为一个状态,表示为:$s=\{f_{i,r}^1,f_{i,r}^2,\cdots,m_{i,r}^1,m_{i,r}^2,\cdots,d_i^k,c_i^k\}$. 其中 $f_{i,r}^i$ 表示边缘节点 m 空闲的计算资源,$m_{i,r}^j$ 表示边缘节点 m 空闲的存储空间,d_i^k 表示计算任务 t_i^k 的数据大小,c_i^k 表示执行计算任务 t_i^k 所需的计算资源.

动作:即单个任务的卸载决策,本文对服务器进行编号[0,m],其中表示终端,表示编号为的边缘服务器,所以动作 $a\in[0,m]$,表示卸载到哪执行,同时满足式(13)的约束条件 $C1$、$C5$.

奖励函数:在强化学习中,智能体每执行一个动作会收到环境的一个反馈(即奖励),该奖励在一定程度上反应了智能体在某个状态下执行该动作的好坏. 本文在定义奖励时,权衡了时延、能耗和负载均衡指标;将卸载到本地执行的任务奖励设置为 0;卸载到服务器执行的任务奖励表示为:

$$R(t_i^k)=(\alpha(1-T_S/T_L)+\beta(1-E_S/E_L)-\gamma\sigma^s). \tag{18}$$

根据奖励函数,任务卸载到边缘服务器执行的时延、能耗和负载均衡指标相对于本地越小,智能体获得的奖励就越大,负载均衡越好. 因此,通过奖励函数鼓励智能体将任务卸载到能获得更大效益的边缘服务器.

(2)结合实验过程,使用近端策略优化算法求解任务卸载决策的具体过程可描述为:

1)将环境状态 s_1 输入到策略网络,输出各动作的概率分布,并通过随机采样确定智能体动作 a_1;再将动作 a_1 作用于环境,环境转移到下一个状态 s_2,并返回奖励 r_1. 循环该过程,直到收集一定量的[$(s_1,a_1,r_1),\cdots,(s',a',r')$].

2)将 1)中最后一步的环境状态 s'输入到评价网络,计算状态价值 $V(s')$;并计算每一步的奖励 R,得到:

$$R=[R1,R2,\cdots,R']. \tag{19}$$

3)将 1)中存储的(s_i,a_i,r_i)输入到评价网络,得到每一步的状态值 $V(s_i)$. 计算优势函数:

$$A^{\theta k}(s_t,a_t)=R(s_t,a_t)-V(s_t,a_t) \tag{20}$$

4)计算评价网络的损失函数 $critic\_loss=MES(A)$,并将 $critic\_loss$ 进行反向传播,用于更新评价网络.

5)将 1)中存储的所有(s_i,a_i,r_i)分别输入到行为策略网络和目标策略网络. 得到相应状态下每个动作对应的概率值 $pb1$ 和 $pb2$,则重要性权重 $IW=pb1/pb2$;

6)计算策略网络的损失函数 *actor_loss*,即近端比率裁剪损失,并进行反向传播,更新目标策略网络;

7)循环 5)~6),到一定步数后结束循环,把目标策略网络的权重加载到行为策略网络;

8)循环 1)~7),直到满足迭代结束条件,结束训练.

算法 2　基于近端策略优化的任务卸载

输入:任务计算资源、功率分配　　输出:任务卸载决策

1. for $k=1,\cdots$ do
2. 在策略 π_{θ_k} 下,收集 $\{s_t,a_t,r_t\}$,保存轨迹集
3. 计算奖励 $\bar{G}_t$
4. 基于价值函数 V_{φ_k},计算优势函数 $\hat{A}_t=\bar{G}_t-V_{\varphi_k}$
5. for $m=1,\cdots,M$ do
6. 采用 Adam 随机梯度上升最大化 PPO-Clip 的目标函数来更新策略
7. end for
8. for $b=1,\cdots,B$ do
9. 采用梯度下降法最小化均方误差来学习价值函数
10. end for
11. end for

3　实验结果与分析

3.1　实验参数设置

在实验中,所有终端设备都在基站的覆盖范围内随机分布,部分基站配备边缘服务器. 其中,每个终端设备的计算能力分布在[0.7,0.9]GHz/s,边缘服务器的计算能力为 20 GHz/s,分配给单个任务的计算能力分布在[0.9,2.1]GHz/s. 仿真实验在配备 Intel i5-7200U 的计算机上进行,其 CPU 主频为 2.5 GHz、内存为 16GB. 基于 Python 3.6 和 Pytorch 开源机器学习框架构建基于 Actor-Critic 的神经网络模型. 其中,Actor 网络的学习率为 0.000 3,Critic 网络的学习率为 0.001,奖励折扣因子为 0.9. 其他的关键参数设置如表 2 所示.

表 2　实验参数

Table 2　Experimental parameters

实验参数	数值	实验参数	数值
信道带宽	20 MHz	噪声功率	-174 dBm/Hz
信道增益	10e6	终端发射功率	[150,200]mW
终端能量系数	10e-25	终端计算能力	[0.2,1.4] GHz
边缘节点计算能力	20 GHz	边缘节点计算功率	[10,50]W
单个终端任务数	[3,8]	单任务数据大小	[100,500]KB
处理每 bit 数据所需 CPU 周期数	[500,700]	DE 种群大小	32
DE 缩放因子	0.5	DE 交叉概率	0.8
Actor 网络的学习率	0.003	Critic 网络的学习率	0.001
折扣因子	0.9	Clip 参数	0.2

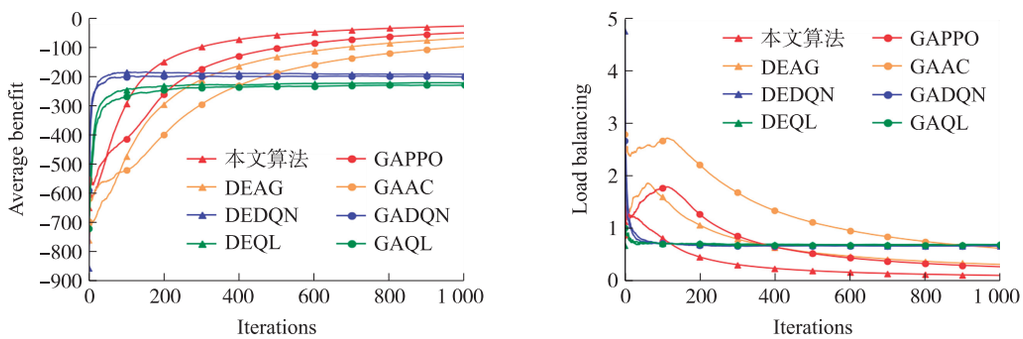
3.2　实验结果分析

为了验证本文方法的有效性,参照文献[7]的算法框架,设计以下方法与本文方法进行对比,包括:基于差分进化和演员评论家算法的任务卸载和资源分配方法(differential evolution and actor-critic, DEAC)[19],基于差分进化和 Deep Q-Network 的任务卸载和资源分配方法(differential evolution and deep Q-network,DEDQN)[20-21],基于差分进化和 Q-learning 的任务卸载和资源分配方法(differential evolution and Q-learning,DEQL)[21]. 另外,为使实验更加充分严谨,针对资源分配子问题,本文在下一小节中还采用了遗传算法求解[3],并将其与强化学习算法结合,作为求解本文研究问题的对比算法.

(1)方法性能对比

本文探讨了不同方法的收敛性和性能. 图 2 描述不同方法进行任务卸载和资源分配时的收敛性,以

及任务卸载效益和服务器负载均衡指标与迭代次数之间的关系. 其中图 2(a)中的纵坐标表示任务卸载效益,该值与延迟、能耗和负载均衡指标相关,取值越大越好. 图 2(b)中的纵坐标表示服务器负载均衡指标,其值越小越好. 可以看出,随着迭代次数的增加,八种方法都趋于收敛,其中本文方法性能最好;而且 DEPPO(本文算法)、GAPPO、DEAG、GAAC 的效果普遍要比其他四种方法的效果好,但 DEDQN、DEQL、GADQN、GAQL 前期的收敛速度更快. 因为状态数少时,后面四种方法中的 DQN、QL 只有一个神经网络,通过动作-价值来选择动作,能够快速的构建网络模型,具有一定优势;而前面四种方法中的 PPO、AC 都基于 AC 架构,具有两个神经网络,网络模型的训练需要更多的数据,收敛会慢一点;但随着数据量的增加,AC 架构的网络模型经过大量数据训练后更加准确,性能更好. 此外,实验结果表明 DE 与强化学习结合的方法普遍要比 GA 与强化学习结合的方法效果好. 因此,在接下来的实验中,我们采用 DE 与强化学习结合的方法求解本文研究问题.



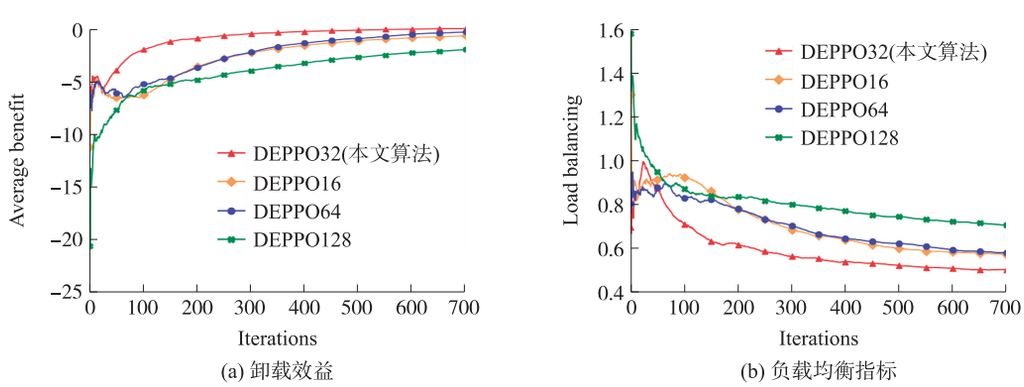
(a) 卸载效益　　(b) 负载均衡指标

图 2　不同方法对卸载效益和负载均衡指标的影响

Fig. 2　The impact of different methods on offloading efficiency and load balancing indicators

(2)网络层不同神经元个数的影响

为了提高本文方法的性能,本节探讨使用不同神经元数对边缘计算系统性能的影响. 从图 3 中可以看出,单层神经元数分别为 16、32、64、128 时,本文方法都能快速收敛;且当单层神经元数为 32 时,实验效果最佳. 图 3(a)展示了使用不同神经元个数对系统卸载效益的影响,当每个神经层含有 32 个神经元时,方法收敛速度最快,实验效果最好. 图 3(b)展示了使用不同神经元个数对服务器负载均衡指标的影响,同样地当每层神经元个数设置为 32 时,方法的收敛性和效果更佳. 因此,在接下来的实验中,我们在神经网络的每层设置 32 个神经元.



(a) 卸载效益　　(b) 负载均衡指标

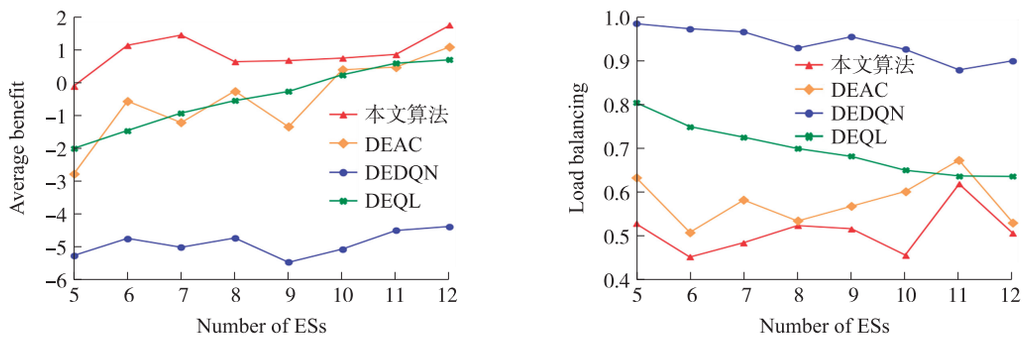
图 3　本文方法的层神经元数对卸载效益和负载均衡指标的影响

Fig. 3　The impact of the number of layer neurons in this method on offloading efficiency and load balancing indicators

(3)边缘服务器数量的影响

本节探讨了不同数量的边缘服务器对边缘计算系统性能的影响. 如图 4 所示,两个子图分别从卸载效益和负载均衡两方面展示了不同边缘服务器数量下的系统性能,随着边缘服务器数量的增加,本文方法在两项性能比较中都优于其他方法. 在图 4(a)中,本文方法对应的卸载效益随着边缘服务器数量的增加先增后减,这是因为卸载效益取决于每个任务卸载的优劣,而优劣是通过奖励函数进行定义,因此效益并

不与边缘服务器数量直接相关. 但由图 4 可以看出,本文方法对应的任务卸载效益一直处于最大值,相比于其他方法而言效果更佳. 同样地,在图 4(b)中,本文方法对应的负载均衡指标一直处于最小值,效果也都优于其他方法. 实验结果表明,本文方法相对稳定,能够在不同的边缘服务器数量下保持良好的效果.



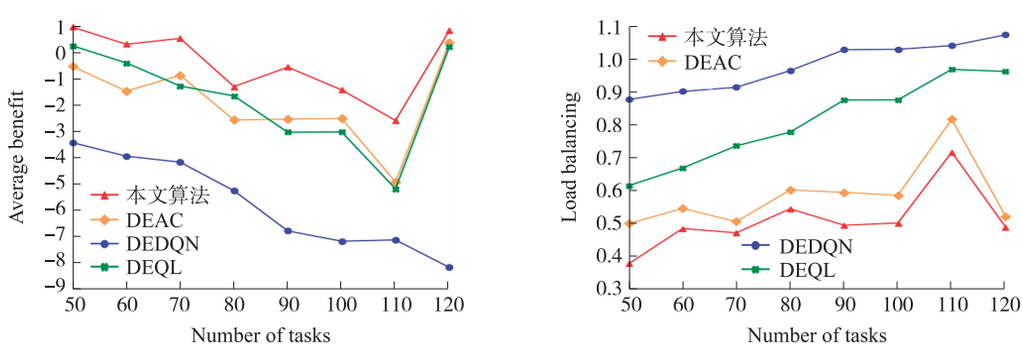
(a) 卸载效益　　(b) 负载均衡指标

图 4　不同边缘服务器数对卸载效益和负载均衡指标的影响

Fig. 4　The impact of different number of edge servers of offloading efficiency and load balancing indicators

(4)任务数量的影响

本节探讨了不同任务数量对边缘系统性能的影响. 如图 5 所示,两个子图分别展示了不同任务数对任务卸载效益和负载均衡的影响,本文方法在两项性能比较中都优于其他方法. 在图 5(a)中,本文方法获得的系统效益随着任务数的增加都保持最大,但没有随着任务数量的增加而增大. 因为实验过程中,当任务数增加时,单个终端设备产生的任务数量也随之增加,但由于任务是随机生成的,数据量大小在[100,500]KB,每个 bit 需要的计算周期为[500,600],在每次改变任务数量时,就在原有任务基础上新生成 10 个任务,这样可以保证随着任务数增加数据量和所需计算资源都增加,但任务间的数据量与所需计算资源并不一定成正比,即数据量大的任务所需的计算资源并不一定大,所以在计算系统效益时,系统效益并不会随着任务数的增加有成正比的趋势.



(a) 卸载效益　　(b) 负载均衡指标

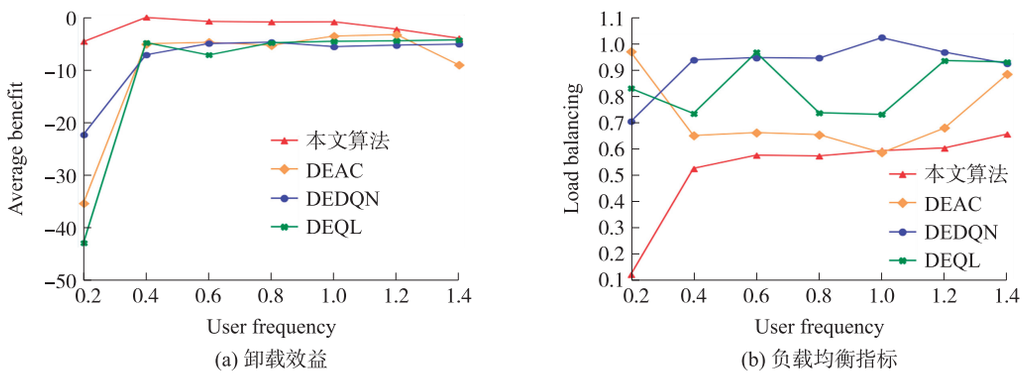
图 5　不同任务数对卸载效益和负载均衡指标的影响

Fig. 5　The impact of different number of tasks on offloading efficiency and load balancing indicators

(5)终端设备计算频率的影响

一般情况下,终端设备的计算能力只会影响在本地执行任务的时延和能耗,但本文定义的卸载效益是任务在边缘节点执行与本地执行时各指标比值的加权和,所以终端设备计算频率对本文所求卸载效益和负载均衡有较大影响.

如图 6 所示,两个子图分别从系统效益和负载均衡两方面展示了终端设备不同计算频率下的系统性能,在图示范围内随着终端设备计算频率的增加,本文方法得到的策略在各项性能比较中都优于其他方法. 图 6(a)为各方法对应的系统效益随着终端设备计算频率增加的变化趋势,可以看出,本文方法对应的值一直高于其他方法;图 6(b)为各方法对应的负载均衡指标随着终端设备计算频率增加的变化趋势,同样地,本文方法一直小于其他方法值. 结果表明,本文方法相比于其他算法,能够在终端设备的不同计算频率下保持稳定、良好的效果.



(a) 卸载效益　　(b) 负载均衡指标

图 6　不同终端计算频率对卸载效益和负载均衡指标的影响

Fig. 6　The impact of different terminal equipment calculation frequency on offloading efficiency and load balancing indicators

4　结论

本文研究多服务器多任务边缘环境中的任务卸载和资源分配问题. 通过联合优化任务卸载决策、计算资源分配和终端设备功率,使卸载时延、能耗以及边缘服务器间的负载均衡指标达到最佳. 首先定义了任务卸载效益,将关于时延、能耗和系统负载均衡指标的多目标优化问题转化为单目标优化;然后提出双层算法求解资源分配和卸载决策. 在卸载决策确定的情况下,使用差分进化算法求解最佳资源分配;在资源分配确定的情况下,使用近端策略优化算法求解最佳卸载决策;并通过不断迭代求解系统最优解. 最后,设计一个综合实验来验证本文方法的性能和可行性. 实验结果表明,本文方法在系统效益和服务器负载均衡方面均明显优于其他方法.

在今后的工作中,我们将延续本文工作,对以下问题进行研究. (1)本文假设终端设备产生的任务相互独立,后续将考虑任务之间的依赖关系. (2)本文考虑无相互干扰的信道传输,后续研究中将讨论干扰信道下的效果. (3)本文考虑任意两基站之间都有通信链路直接连接,后续将考虑网络节点的拓扑结构.

[参考文献]

[1] 刘伟,黄宇成,杜薇,等. 移动边缘计算中资源受限的串行任务卸载策略[J]. 软件学报,2020,31(6):1889-1908.

[2] FAN W, LI S, LIU J, et al. Joint task offloading and resource allocation for accuracy-aware machine-learning-based IIoT applications[J]. IEEE internet of things journal,2022,10(4):3305-3321.

[3] ZHOU T,YUE Y,QIN D,et al. Joint device association,resource allocation,and computation offloading in ultra-dense multidevice and multitask IoT networks[J]. IEEE internet of things journal,2022,9(19):18695-18709.

[4] TRAN T X,POMPILI D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks[J]. IEEE transactions on vehicular technology,2019,68(1):856-868.

[5] DAI P,HU K,WU X,et al. Asynchronous deep reinforcement learning for data-driven task offloading in mec-empowered vehicular networks[C]//2021 IEEE Conference on Computer Communications(INFOCOM). IEEE,Electr Network,2021:1-10.

[6] 张永棠. 一种深度强化学习的 C-RAN 动态资源分配方法[J]. 小型微型计算机系统,2021,42(1):132-136.

[7] 邝祝芳,陈清林,李林峰,等. 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[J]. 计算机学报,2022,45(4):812-824.

[8] 吴学文,廖婧贤. 云边协同系统中基于博弈论的资源分配与任务卸载方案[J]. 系统仿真学报,2022,34(7):1468-1481.

[9] 熊兵,张俊杰,黄思进,等. 多约束边环境下计算卸载与资源分配联合优化[J/OL]. 小型微型计算机系统:1-8[2022-11-17].

[10] XU C,ZHENG G,ZHAO X. Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks[J]. IEEE transactions on vehicular technology,2020,69(12):16001-16016.

[11] 田贤忠,许婷,朱娟. 一种最小化时延多边缘节点卸载均衡策略研究[J]. 小型微型计算机系统,2022,43(6):1162-1169.

[12] ZHANG W, ZHANG G, MAO S. Joint parallel offloading and load balancing for cooperative-MEC systems with delay constraints[J]. IEEE transactions on vehicular technology, 2022, 71(4): 4249-4263.

[13] YANG B, CAO X, BASSEY J, et al. Computation offloading in multi-access edge computing networks: a multi-task learning approach[J]. IEEE transactions on mobile computing, 2021, 20(9): 2745-2762.

[14] ZHOU J Y, ZHANG X L. Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing[J]. IEEE internet of things journal, 2022, 9(5): 3812-3824.

[15] WANG P, LI K, XIAO B, et al. Multi objective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing[J]. IEEE internet of things journal, 2022, 9(14): 11737-11748.

[16] LIU H, LI Y, WANG S. Request scheduling combined with load balancing in mobile-edge computing[J]. IEEE internet of things journal, 2022, 9(21): 20841-20852.

[17] CAI J, FU H, LIU Y. Multi-task multi-objective deep reinforcement learning-based computation offloading method for industrial internet of things[J]. IEEE internet of things journal, 2023, 10(2): 1848-1859.

[18] MA M, GONG C, WU L, et al. FLIRRAS: fast learning with integrated reward and reduced action space for online multitask offloading[J]. IEEE internet of things journal, 2023, 10(6): 5406-5417.

[19] HAMMAMI N, NGUYEN K K. On-policy vs. off-policy deep reinforcement learning for resource allocation in open radio access network[C]//2022 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, Austin, TX, 2022: 1461-1466.

[20] FILALI A, NOUR B, CHERKAOUI S, et al. Communication and computation O-RAN resource slicing for URLLC services using deep reinforcement learning[J]. IEEE communications standards magazine, 2023, 7(1): 66-73.

[21] ZHANG H, ZHOU H, EROL-KANTARCI M. Team learning-based resource allocation for open radio access network[C]// 2022 IEEE International Conference on Communications (ICC). IEEE, 2022: 4938-4943.

[责任编辑:杜忆忱]