物联网边缘计算场景下基于优先级任务的卸载决策优化

朱思峰，胡家铭，杨诚瑞，蔡江昊

（天津城建大学 计算机与信息工程学院，天津 300384）

摘 要：针对物联网应用场景下，依照最大容忍时延等标量信息划分任务卸载优先级的方式难以满足紧急任务处理需求的问题，为确保关键程度最高的紧急任务优先处理，本文提出基于任务关键程度划分优先级的方法。针对优先级任务卸载决策问题展开了研究，考虑了边缘服务器任务处理程序缓存，以最小化综合时延、社会损失率、负载失衡度为优化目标，建立了多目标优化任务卸载决策问题模型，提出了一种改进的多目标灰狼优化算法求解问题。该算法引入了灰狼个体尽力而为进化策略、基于改进差分进化算子的外部存档生成策略、加权最值法最优解保存策略以提升算法性能。仿真实验表明：本文提出的方法能有效降低综合时延和社会损失率，优化边缘服务器间负载均衡，确保紧急任务优先处理，且其性能均较其他方法表现优异。

关键词：物联网；边缘计算；任务卸载决策；优先级任务；多目标灰狼优化算法

中图分类号：TP393.1；TN929 **文献标志码**：A **文章编号**：1671-5497(2024)11-3338-13

DOI：10.13229/j.cnki.jdxbgxb.20230047

Optimization of offloading decision based on priority task in edge computing scenes of internet of things

ZHU Si-feng，HU Jia-ming，YANG Cheng-rui，CAI Jiang-hao

（*School of Computer and Information Engineering*，*Tianjin Chengjian University*，*Tianjin* 300384，*China*）

Abstract：In the application scenario of the Internet of Things，it is difficult to meet the processing needs of emergency tasks by prioritizing task offloading based on scalar information such as maximum tolerance delay. The most critical task is called an emergency task. To ensure that emergency tasks are prioritized，this paper proposes a method of prioritizing tasks based on their criticality，and conducts research on the decision-making problem of priority task offloading，taking into account the caching of edge server task handlers，with the optimization objectives of minimizing comprehensive delays，social loss rate，and load imbalance degree. A multi-objective optimization task offloading decision problem model was established，and an improved multi-objective grey wolf optimizer was proposed to solve the problem. This algorithm introduces the best effort evolution strategy of grey wolf individuals，an external archive generation strategy based on improved differential evolution operator，and a weighted maximum method optimal solution preservation strategy to improve algorithm performance. Simulation experiments show that the algorithm

收稿日期：2023-01-16.

基金项目：国家自然科学基金项目(61972456)；天津市自然科学基金重点项目(22JCZDJC00600).

作者简介：朱思峰(1975-)，男，教授，博士.研究方向：边缘计算，人工智能算法及应用等.E-mail：zhusifeng@163.com

proposed in this paper can effectively reduce the comprehensive delay and social loss rate, optimize load balancing between edge servers, ensure priority processing of emergency tasks, and its algorithm performance is superior to other algorithm schemes.

Key words: Internet of things; edge computing; task offloading decision; priority task; multi-objective grey wolf optimizer

0　引　言

随着物联网与信息通信技术的快速发展[1],视频图像处理、增强现实、即时通信等高资源需求型和时延敏感型的物联网应用服务逐渐增多,物联网设备因其自身计算与存储资源有限,难以满足此类应用的任务处理需求[2]。云计算允许物联网设备将任务发送到云端进行处理,以使用其丰富的计算存储资源。但在车联网、远程医疗、城市应急管理等应用场景下产生的待处理任务通常需要紧急处理,将任务上传至云端进行处理会带来不可避免的高额传输时延,这使得云计算模式在此场景下很难适用[3,4]。边缘计算技术通常将任务卸载到网络边缘的服务器设备上,以提供就近的服务,实现计算与存储资源的下沉,更好地满足某些物联网应用场景下的任务处理需求[3]。

目前已有学者对物联网应用场景下的计算卸载策略进行了较为深入的研究,时延、负载均衡是常用的评价指标。其中,任务卸载处理时延是一项重要标准。任务分配的不均衡会使某些服务器过载或欠载,导致服务器资源利用率下降[5]。文献[6]研究了边缘计算场景下密集型任务的计算卸载决策问题,利用改进的粒子群优化算法求解卸载问题,有效降低了任务处理时延。文献[7]在物联网应用场景下建立了边缘计算资源分配模型,提出了基于理想相似性订单偏好与多准则决策技术获得最优卸载策略的方法,实现对时延和负载均衡的优化。为满足边缘计算场景下的计算密集型和时延敏感型任务处理需求,文献[8]提出了一种面向多边缘设备协作的任务卸载和服务缓存联合优化机制,采用基于偏好的双边匹配算法,实现了对系统总时延和负载均衡的优化。在满足时延和负载均衡需求的情况下,任务卸载成功率[9]也是衡量计算卸载决策优劣性常用的评价指标。文献[10]研究了物联网场景下的边缘计算卸载决策问题,综合考虑了时延、能耗、任务成功率3个优化目标,并应用深度强化学习算法求解计算卸载问题,提升了用户体验。

为提高任务处理效率,诸多学者对物联网边缘计算应用场景下的任务卸载策略进行了研究,且多数研究以任务请求时间的先后顺序为其分配计算资源并处理,而忽略了任务处理的紧迫程度与重要性是有区分度的事实[11]。为此,许多学者考虑了任务间的优先级限制,对基于优先级任务的计算卸载策略展开了研究。现有研究多数以最大容忍时延、任务计算量、任务数据量等标量信息作为任务优先级划分标准[12-14]。其中,文献[12]根据任务最大容忍时延的不同为任务划分优先级,最大容忍时延小的任务优先级高,高优先级任务优先处理,并提出了一种DPTO(Delay-dependent priority-aware task offloading)任务卸载策略,最大限度地减少了任务处理总时延。文献[13]综合考虑了任务最大容忍时延、任务计算量和数据量对任务卸载优先级的影响,并采用层次分析法划分任务优先级,最小化任务处理时延。文献[14]根据任务最大容忍时延的不同将任务分为时延敏感型和时延容忍型任务,降低了任务排队时延。

上述文献分别从任务卸载策略优化和任务优先级划分两个方面对现有研究进行阐述,可以看出目前对边缘计算任务卸载策略与根据任务标量信息划分任务优先级的研究较为成熟。但在某些物联网应用场景下的任务卸载与任务优先级划分标准仍存在研究盲点。例如,在紧急医疗、车辆控制、城市安全预警等场景中,紧急任务的卸载和处理优先级应该比车载娱乐、增强现实等非紧急任务的卸载和处理优先级高[4,15]。针对上述问题,本文将考虑构建基于边缘服务器和终端设备的两层任务卸载模型,按照任务关键程度划分任务优先级,保证紧急任务优先处理,并采用多目标优化算法求解基于优先级任务的卸载决策问题。本文的主要贡献如下:

(1)设计一种基于改进的多目标灰狼优化算法(Improved multi-objective grey wolf optimizer,

IMOGWO),求解基于优先级任务的卸载决策问题,实现对综合时延、社会损失率和负载失衡度3个目标的优化。

(2)提出一种基于任务关键程度划分任务优先级的思想,建立端边模式的计算卸载网络模型,保证紧急任务优先处理。

(3)将本文提出的IMOGWO算法方案与MPSO/D算法方案[17]、MOGWO/D算法方案[18]、MO-NSGA算法方案[14]、Random方案和LOCAL方案进行了详细的对比实验,仿真实验验证了理论分析,论证了算法改进策略的有效性,证明了本文所提方案的优越性。

1 系统模型

物联网边缘计算应用场景下的端边模式任务卸载系统模型如图1所示。

假设系统中有M个边缘服务器,记作$E=\{e_1,e_2,\cdots,e_j,\cdots,e_M\}$,其中$1\leqslant j\leqslant M$。1个协调服务器用于负载均衡控制。有$N$个终端设备,记作$U=\{u_1,u_2,\cdots,u_i,\cdots,u_N\}$,其中$1\leqslant i\leqslant N$。在$t$时间段内,每个$u_i$产生一个待处理的任务,系统中的任务记作$S=\{s_1,s_2,\cdots,s_i,\cdots,s_N\}$。参照文献[19]的描述,本文将任务划分为数据部分和处理程序部分,且部分任务处理程序能在边缘服务器缓存,以提高资源利用率。任务$s_i=\{d_i^{\text{up}},d_i^{\text{down}},c_i,t_i^{\text{endure}},pr_i,\theta_i,DP_i\}$,其中$d_i^{\text{up}}$为$s_i$数据部分的上传数据量,$d_i^{\text{down}}$为$s_i$的回传数据量,$c_i$为$s_i$所需计算量,$t_i^{\text{endure}}$为$s_i$的最大容忍时延,$pr_i$为$s_i$的优先级。$\theta_i$为处理$s_i$所需的任务处理程序,$\theta_i\in Z$,$Z$为所有任务处理程序名称的集合,可表示为$Z=\{z_1,z_2,z_3,\cdots\}$。$DP_i$为$s_i$的任务处理程序$\theta_i$的数据量。

每个终端设备上的待处理任务可选择在本地执行或卸载至边缘服务器执行,决策向量$\boldsymbol{X}=\{x_1,x_2,\cdots,x_i,\cdots,x_N\}$。决策变量$x_i$定义为:

$$x_i=\begin{cases}j,s_i\text{卸载到}e_j\text{执行}\\0,s_i\text{在本地执行}\end{cases}\tag{1}$$

其中:当$x_i=0$时表示s_i在本地执行;当$x_i=j$时表示s_i卸载到e_j执行。

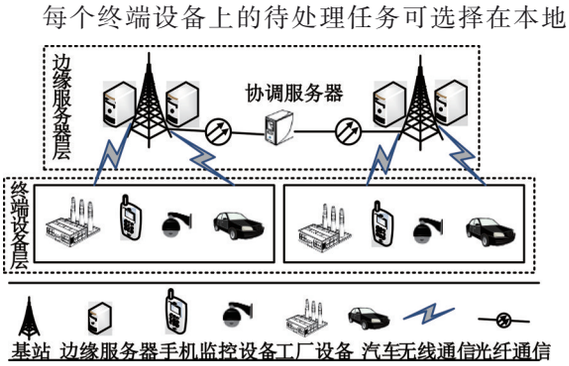

图1 系统模型

Fig. 1 System model

1.1 任务优先级分类

本文根据任务的关键程度,将其分为紧急任务、一般紧急任务和非紧急任务。这3类任务对应的优先级分别为高优先级任务(PR_1)、中优先级任务(PR_2)和低优先级任务(PR_3)。任务优先级作为任务的一个属性存在。

规则:3种优先级别任务均可选择在本地执行或卸载到边缘服务器执行,但执行计算卸载决策时,PR_1类任务相较于PR_2类任务与PR_3类任务具有优先传输和优先计算的权利。PR_2类任务不如PR_1类任务紧急,但有一定的处理紧迫性,相较于PR_3类任务在计算卸载时优先处理。PR_3类任务无处理紧迫性,不需要被紧急处理。例如,文献[4]中考虑紧急医疗任务和紧急预警任务应该比一般任务的处理优先级高。

1.2 任务调度

如图2所示,假设边缘服务器均内置任务分类器,根据任务s_i的优先级为其分配队列。具有PR_1、PR_2、PR_3 3种不同优先级的任务应分别进入Q_1(高优先级任务队列)、Q_2(中优先级任务队列)、Q_3(低优先级任务队列)队列。

基本假设:本文规定边缘服务器上每次仅能运行一个任务,当有新任务到来时,根据其优先级插入对应队列。此外,当边缘服务器需要从队列中读取任务时,优先从高优先级队列读取;若高优先级队列为空,则从中优先级队列读取;若中优先级队列为空,则从低优先级队列读取。

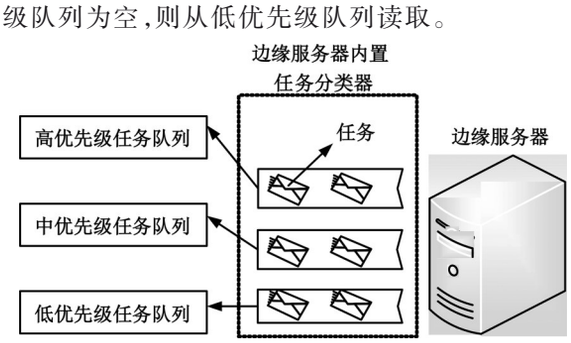

图2 边缘服务器内置任务分类器

Fig. 2 Edge server built-in task classifier

1.3　缓存设定

边缘服务器具备存储功能,用于缓存任务处理程序。本文将任务分为数据部分与任务处理程序部分,边缘服务器上需要同时具备任务数据与任务处理程序才可完成任务计算,任务处理程序可复用。如果 e_j 未缓存任务处理程序 θ_i,但收到了应用任务处理程序 θ_i 的任务,则 u_i 需要向 e_j 同时传输数据和任务处理程序,任务处理完成后 e_j 缓存任务处理程序 θ_i;如果 e_j 已缓存 θ_i,则进行任务卸载时仅需上传任务的数据部分。e_j 定期清除缓存中不常用的任务处理程序。若同时存在两个使用相同任务处理程序的任务都卸载至 e_j,且 e_j 未缓存该任务处理程序,则仅需一个终端传输任务处理程序。

1.4　时延模型

s_i 本地执行的计算时延 T_i^{local} 表示为:

$$T_i^{\text{local}}=\frac{c_i}{f_i^{\text{u}}} \tag{2}$$

式中:f_i^{u} 为终端设备 u_i 的计算能力。

然后考虑 s_i 卸载至 e_j 过程中的时延情况。其中包括 s_i 上传的传输时延、在任务分类器内的排队时延、在 e_j 上的处理时延,任务处理结果回传至 u_i 的回传时延。此外,不同任务的处理优先级不同,在任务卸载阶段表现为任务所获传输带宽和传输优先级不同,从而影响任务上传速率和上传顺序,接下来讨论不同优先级任务的传输带宽和传输优先级。

考虑任务优先级、是否携带处理程序卸载及最大容忍时延三方面对任务卸载的影响,将任务卸载到合适的边缘服务器。根据香农公式,终端设备 u_i 卸载任务到 e_j 的上传速率(单位:bps)R_{ij}^{tran} 为:

$$R_{ij}^{\text{tran}}=B_i\log_2\left(1+\frac{P_i^{\text{up}}\hbar}{\sigma^2}\right) \tag{3}$$

式中:P_i^{up} 为 u_i 的发射功率;$\hbar$ 为信道增益;σ^2 为高斯白噪声;B_i 为 u_i 获得的信道带宽。结合任务优先级,并应用正交频分复用(OFDM)技术,系统总带宽 B 根据信道带宽分配规则划分成 m 个子频带实现数据传输[20],为确保数据传输带宽划分的公平性,信道带宽分配规则如下。

假设在 t 时间段内有 m 个终端设备产生 n($m>n$)个 PR_1 类型任务、$m-n$ 个 PR_2 或 PR_3 类型任务需要卸载到边缘服务器,其中 n 个产生 PR_1 类型任务的终端设备将优先占用系统全部带宽 B 进行数据传输,并按照式(4)划分成 n 个子信道;另外 $m-n$ 个产生 PR_2 或 PR_3 类型任务的终端设备将等待 PR_1 类型任务传输完成后再根据式(4)划分为 $m-n$ 个子信道。

$$B_i=B\times\frac{d_i^{\text{up}}}{\sum_{i=1}^{Y}d_i^{\text{up}}} \tag{4}$$

式中:Y 为按照任务优先级信道划分规则在 Δt 时刻需要上传到边缘服务器的任务总数,$Y=n$ 或 $Y=m-n$。

s_i 卸载到 e_j 的传输时延 T_{ij}^{tran} 为:

$$T_{ij}^{\text{tran}}=\begin{cases}\dfrac{\text{Data}_i}{R_{ij}^{\text{tran}}}, & PR_1\in s_i\\ T^{\text{tran\_PR}_1}+\dfrac{\text{Data}_i}{R_{ij}^{\text{tran}}}, & (PR_2\text{ 或 }PR_3)\in s_i\end{cases} \tag{5}$$

式中:$T^{\text{tran\_PR}_1}$ 为 PR_1 类型任务的传输时延;Data_i 为 s_i 卸载到 e_j 的总数据大小,若 e_j 已缓存 s_i 的处理程序,则 $\text{Data}_i=d_i^{\text{up}}$,若 e_j 上未缓存 s_i 所需任务处理程序,则 $\text{Data}_i=d_i^{\text{up}}+DP_i$。

s_i 的排队时延 T_i^{qu} 为:

$$T_i^{\text{qu}}=\begin{cases}T_0^{\text{pro}}+\sum_{k=1}^{K}T_k^{\text{pro}}, & PR_1\in s_i\\ T_0^{\text{pro}}+\sum_{k=1}^{K}T_k^{\text{pro}}+\sum_{r=1}^{R}T_r^{\text{pro}}, & PR_2\in s_i\\ T_0^{\text{pro}}+\sum_{k=1}^{K}T_k^{\text{pro}}+\sum_{r=1}^{R}T_r^{\text{pro}}+\sum_{l=1}^{L}T_l^{\text{pro}}, & PR_3\in s_i\end{cases} \tag{6}$$

式中:T_0^{pro} 为 t 时间段内 e_j 上正在执行的任务剩余处理时延;$T_k^{\text{pro}}(1\leqslant k\leqslant K)$ 为 PR_1 类型任务在 e_j 上的处理时延;k 为 PR_1 类型任务在 Q_1 中所处排队位置;K 表示 Q_1 中一共有多少任务排队;$T_r^{\text{pro}}(1\leqslant r\leqslant R)$ 为 PR_2 类型任务在 e_j 上的处理时延;r 为 PR_2 类型任务在 Q_2 中所处排队位置,R 表示 Q_2 中一共有多少任务排队;$T_l^{\text{pro}}(1\leqslant l\leqslant L)$ 为 PR_3 类型任务在 e_j 上的处理时延;l 为 PR_3 类型任务在 Q_3 中所处排队位置;L 表示 Q_3 中一共有多少任务排队。

s_i 出队推送至 e_j 的 CPU 计算前先估算卸载总时延。s_i 在 e_j 上的预估处理时延 $T_{ij}^{\text{pro\_ev}}$ 为:

$$T_{ij}^{\text{pro\_ev}}=\frac{c_i}{f_j^{\text{e}}} \tag{7}$$

式中:f_j^{e} 为 e_j 的计算能力。

任务预估回传时延 $T_{ji}^{\text{down\_ev}}$ 为:

$$T_{ji}^{\text{down\_ev}}=\frac{d_i^{\text{down}}}{R_{ji}^{\text{down}}} \tag{8}$$

式(8)中的任务回传速率 R_{ji}^{down} 为:

$$R_{ji}^{\text{down}}=B\log_2\left(1+\frac{P_{ji}^{\text{down}}\hbar}{\sigma^2}\right) \tag{9}$$

式中:P_{ji}^{down} 为回传功率。

综合式(3)~(9),估算 s_i 计算卸载总时延 $T_i^{\text{total\_ev}}$ 为:

$$T_i^{\text{total\_ev}}=T_{ij}^{\text{tran}}+T_i^{\text{qu}}+T_{ij}^{\text{pro\_ev}}+T_{ji}^{\text{down\_ev}} \tag{10}$$

若 $T_i^{\text{total\_ev}}$ 不超过 s_i 的最大容忍时延 t_i^{endure},则判断任务可按时交付处理,CPU正常处理任务;若 $T_i^{\text{total\_ev}}$ 超过 s_i 的最大容忍时延 t_i^{endure},则判断任务无法交付,将任务移出CPU,不再计算。由此可得卸载到 e_j 的 s_i 处理时延 T_{ij}^{pro} 为:

$$T_{ij}^{\text{pro}}=\begin{cases}T_{ij}^{\text{pro\_ev}} & ,T_i^{\text{total\_ev}}\leqslant t_i^{\text{endure}}\\ 0 & ,T_i^{\text{total\_ev}}>t_i^{\text{endure}}\end{cases} \tag{11}$$

s_i 从 e_j 的回传时延 T_{ji}^{down} 为:

$$T_{ji}^{\text{down}}=\begin{cases}T_{ji}^{\text{down\_ev}} & ,T_i^{\text{total\_ev}}\leqslant t_i^{\text{endure}}\\ 0 & ,T_i^{\text{total\_ev}}>t_i^{\text{endure}}\end{cases} \tag{12}$$

u_i 产生的 s_i 可选择在本地处理,或卸载到 e_j 上处理,但 s_i 的计算卸载总时延不能超过 t_i^{endure}。s_i 本地计算时,总卸载时延仅取决于 s_i 在 u_i 上的处理时延,即式(2)。然而,u_i 将 s_i 卸载至边缘服务器 e_j 上处理的总时延受到任务优先级、e_j 上是否缓存 s_i 的处理程序和 t_i^{endure} 的影响,e_j 上 s_i 的卸载时间 T_{ij}^{offload} 为:

$$T_{ij}^{\text{offload}}=T_{ij}^{\text{tran}}+T_i^{\text{qu}}+T_{ij}^{\text{pro}}+T_{ji}^{\text{down}} \tag{13}$$

s_i 的总处理时间 T_i^{total} 为:

$$T_i^{\text{total}}=\begin{cases}T_i^{\text{local}} & ,x_i=0\\ T_{ij}^{\text{offload}} & ,x_i\neq 0\end{cases} \tag{14}$$

模型的综合时延:本文将考虑 N 个任务中的最大任务处理时延 $T^{\max}$ 与 N 个任务的总处理时延 T^{tw} 的加权时延,权重系数分别为 w_1 和 w_2,综合时延 T^{syn} 为:

$$\begin{cases}T^{\max}=\max\left(T_i^{\text{total}}\right)\\ T^{\text{ut}}=\sum\limits_{i=1}^{N}T_i^{\text{local}},x_i=0\\ T^{\text{et}}=\sum\limits_{i=1}^{N}\sum\limits_{j=1}^{M}T_{ij}^{\text{offload}},x_i\neq 0\\ T^{\text{tw}}=\dfrac{T^{\text{ut}}+T^{\text{et}}}{N}\\ T^{\text{syn}}=w_1\times T^{\max}+w_2\times T^{\text{tw}}\end{cases} \tag{15}$$

式中:T^{ut} 为 N 个任务中选择在本地处理的全部任务处理总时延;T^{et} 为 N 个任务中选择在卸载至边缘服务器处理的全部任务总处理时延。

1.5 社会损失率模型

现实中完成紧急任务对社会的贡献远大于非紧急任务。本文规定具有不同优先级的任务在 t_i^{endure} 内处理完成所获社会收益不同,PR_1、PR_2、PR_3 类型任务处理完成所获社会收益分别由 BE_1、BE_2、BE_3 表示,且 BE_1、BE_2、BE_3 之间满足 $BE_1>BE_2>BE_3$ 的关系。而社会损失率越低,总体的社会收益也就越大。社会损失率 SLR 可表示为:

$$SLR=1-\frac{BE_1\times ct_{\text{PR}_1}^{\text{succ}}+BE_2\times ct_{\text{PR}_2}^{\text{succ}}+BE_3\times ct_{\text{PR}_3}^{\text{succ}}}{BE_1\times ct_{\text{PR}_1}^{\text{total}}+BE_2\times ct_{\text{PR}_2}^{\text{total}}+BE_3\times ct_{\text{PR}_3}^{\text{total}}} \tag{16}$$

式中:$ct_{\text{PR}_1}^{\text{total}}$、$ct_{\text{PR}_2}^{\text{total}}$、$ct_{\text{PR}_3}^{\text{total}}$ 分别为 t 时间段内所有终端设备产生的 PR_1、PR_2、PR_3 优先级类型的任务总数;$ct_{\text{PR}_1}^{\text{succ}}$、$ct_{\text{PR}_2}^{\text{succ}}$、$ct_{\text{PR}_3}^{\text{succ}}$ 分别为 t 时间段内所有终端设备产生的 PR_1、PR_2、PR_3 优先级类型任务在 t_i^{endure} 内成功处理的数量。

1.6 负载失衡度模型

在 t 时间段内,不同终端设备产生的 s_i 可选择本地处理,或卸载至不同的 e_j 上进行处理。为实现在 t 时间段内各边缘服务器接收的总任务量均衡,本文建立了负载失衡度模型。负载失衡度越小,表示各边缘服务器负载越均衡。e_j 接收到的任务队列表示为 $Q=\{s_1,s_2,\cdots,s_i,\cdots,s_H\}$,其中 H 表示 e_j 接收到的总任务数。e_j 接收到的任务需要的总计算量 CQ_j^{e} 表示为 $\sum\limits_{i=1}^{H}s_i\times c_i$。基于 CQ_j^{e},系统中各 e_j 接收到的任务需要的平均计算量 Ave 可表示为:

$$Ave=\frac{\sum\limits_{j=1}^{M}CQ_j^{\text{e}}}{M} \tag{17}$$

负载失衡度 LID 可表示为:

$$LID=\frac{\sum\limits_{j=1}^{M}\left|CQ_j^{\text{e}}-Ave\right|}{M} \tag{18}$$

1.7 问题模型

本文将物联网应用场景下基于优先级任务的卸载决策问题视为一个带有约束的多目标卸载决策优化问题,可表示为:

$$\min T^{\text{syn}},\min SLR,\min LID \tag{19}$$

$$\text{s.t. } 1\leqslant i\leqslant N \tag{20}$$

$$1\leqslant j\leqslant M \tag{21}$$

$$pr_i=\{PR_1,PR_2,PR_3\} \tag{22}$$

$$T_i^{\text{total}}\leqslant t_i^{\text{endure}} \tag{23}$$

$$w_1+w_2=1 \tag{24}$$

其中:式(20)表示模型中有N个终端设备;式(21)表示模型中有M个边缘服务器;式(22)中的pr表示任务s_i存在3种属性的优先级,分别为PR_1、PR_2、PR_3;式(23)表示任务s_i的处理时延T_i^{total}应小于或等于其最大容忍时延t_i^{endure};式(24)中的w_1和w_2分别表示任务s_i最大时延和所有任务平均时延的权重,权重和等于1。

该问题以最小化综合时延、社会损失率和负载失衡度为优化目标,以提高紧急任务卸载成功率,降低社会损失率,保证边缘服务器的均衡负载。

2　基于IMOGWO的卸载决策方案

多目标灰狼优化算法[21](Multi-objective grey wolf optimizer,MOGWO)中,每个灰狼个体代表卸载决策问题的一个候选解,灰狼个体的一个基因位代表一个任务的卸载策略,采用适应度函数评估候选解的优劣性。接下来将介绍编码方式、算法改进策略和IMOGWO算法流程等内容。

2.1　编码与初始化

分析问题特征,并选择合适的编码方式是算法解决问题的第一步。如图3所示,本文所提问题主要针对3个相互耦合目标进行组合优化,对任务的卸载目标服务器进行决策。本文采用十进制编码,问题的候选解由$X=\{x_1,x_2,\cdots,x_i,\cdots,x_N\}$表示,其中$x_i$表示第$i$个终端设备上的任务卸载决策,当$x_i=0$时表示任务$s_i$在本地执行,当$x_i=j$时表示任务$s_i$卸载到$e_j$上执行。本文采用随机初始化方式生成初始种群。

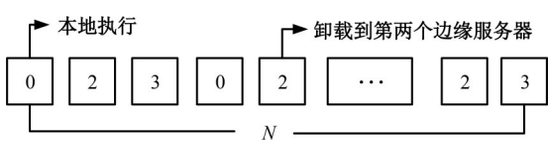

图3　个体编码

Fig. 3　Individual coding

2.2　适应度评价函数

本文从综合时延、社会损失率和负载失衡度三个方面对适应度进行评价,灰狼个体的适应度评价函数为式(19),灰狼个体的适应度评价函数约束条件分别为式(20)~(24)。

2.3　基本灰狼算法

灰狼优化算法是一种模仿灰狼群觅食行为的群智能优化算法,按照适应度值将灰狼个体划分为4种,即灰狼个体最优解α、第二最优解β、第三最优解δ和其他候选解ω。灰狼包围猎物过程的数学模型如下:

$$D=|\boldsymbol{C}\times X_p-X_i| \tag{25}$$

$$X_{i+1}=X_p-\boldsymbol{A}\times D \tag{26}$$

$$\boldsymbol{A}=2a\times\boldsymbol{r}_1-a \tag{27}$$

$$\boldsymbol{C}=2\times\boldsymbol{r}_2 \tag{28}$$

式中:D为灰狼个体与猎物之间距离;$\boldsymbol{C}$为系数向量;X_p为猎物的位置;X_i为灰狼个体当前位置。式(26)中的X_{i+1}为灰狼个体的新位置;$\boldsymbol{A}$为系数向量。式(27)中的a为收敛因子,随评价次数的增加从2到0线性递减。式(27)(28)中的$\boldsymbol{r}_1$、$\boldsymbol{r}_2$为[0,1]之间的随机向量。

对猎物进行包围后,由α、β、δ狼引导,剩余灰狼(ω狼)更新其各自的位置。该过程ω狼位置更新表达式为:

$$X_{i+1}=\frac{(X_\alpha-\boldsymbol{A}_1\times D_\alpha)+(X_\beta-\boldsymbol{A}_2\times D_\beta)+(X_\delta-\boldsymbol{A}_3\times D_\delta)}{3} \tag{29}$$

式中:$\boldsymbol{A}_1$、$\boldsymbol{A}_2$、$\boldsymbol{A}_3$均为系数向量;X_α、X_β、X_δ分别为α、β以及δ狼当前的位置;D_α、D_β、D_δ分别为狼群中其他个体与α、β以及δ狼之间的距离。

本文在原有MOGWO研究的基础上,提出一种改进的多目标灰狼优化算法(IMOGWO)。为提升MOGWO算法的收敛性,提出了灰狼个体尽力而为进化策略;应用改进的差分进化算子对外部存档进行优化,保证种群的多样性;为避免保存大量的极端解,利用加权最值法对外部存档进行更新。

2.4　灰狼个体进化策略

为提升灰狼个体的收敛速度,本文借鉴文献[22]提出的个体进化策略,并对该策略进行改进,将其应用于灰狼个体进化过程。本文提出的灰狼个体进化策略具体流程为:灰狼个体$X(i)$代表一种计算卸载决策方案,首先应用式(19)的适应度评价函数计算$X(i)$的适应度$F(X(i))$,然后利用式(25)~(29)对$X(i)$实施进化操作,进化后的灰狼个体记为$X^*(i)$,并计算其适应度函数

$F(X^*(i))$的值,当且仅当$F(X(i))\succ F(X^*(i))$才停止进化$X(i)$,否则将循环上述步骤,不断进化$X(i)$。本文将提出的灰狼个体进化策略称为灰狼个体尽力而为进化策略。

2.5 外部存档生成与更新策略

运用灰狼个体进化策略生成NP个灰狼个体$\boldsymbol{X}^*(i)$,并根据非支配规则选出非支配个体进入外部存档。此外,为增加外部存档中解的多样性,扩大算法搜索空间,本文借鉴文献[23]提出的差分进化算子改进策略,对外部存档实施进化操作,以增加外部存档内个体间的交流机会,有效避免种群因收敛速度过快而出现陷入局部最优解的问题。

灰狼个体$X^*(i)$的变异操作可表示为:

$$X(i+1)=X^*(i)+Fc(g)(X^1-X^2) \tag{30}$$

式中:X^1、X^2为$X^*(i)$邻域内两个不同的灰狼个体;g为种群迭代次数;$Fc(g)$为第g次迭代的变异概率;$X(i+1)$为变异后新生成的个体。

交叉操作可表示为:

$$X(i+1)_j=\begin{cases}X_j(i), & r\leqslant CR(g)\ \text{或}\ j=j_{\text{rand}}\\ X_j^*(i), & \text{其他}\end{cases} \tag{31}$$

式中:$X_j(i)$为交叉后的新个体第j维基因;$X_j^*(i)$为变异个体第j维基因;$X(i+1)_j$为变异后新个体第j维基因;r为[0,1]内的随机数;$CR(g)$为第g次迭代的交叉概率;j_{rand}为从[1,M]中随机选择的整数;M为变量维度,变量维度为边缘服务器的数量。

本文对$Fc(g)$和$CR(g)$两个参数利用自适应参数控制方法进行改进,限于篇幅原因,具体改进操作参见文献[23]。

2.6 最优解保存策略

为降低最优解保存阶段的计算复杂度,并避免保存大量的极端解(某一维度的目标上适应度值很大,在其他维度的目标上适应度值很小的候选解),确保保存的解在所有维度的目标上都相对最优,有利于算子寻优。灰狼群与外部存档均使用基于分解的加权最值法保存最优解,加权最值法通过计算加权后的目标值,取其中的最大值作为最终目标值,可表示为:

$$o_i=\max(\lambda_{i,1}\times o_1,\cdots,\lambda_{i,j}\times o_j) \tag{32}$$

式中:o_i为个体$X(i)$在权重向量$\boldsymbol{\lambda}_i$下的最终目标值;$(o_1,\cdots,o_j)$为个体$X(i)$在j维目标的目标值,$j=\{1,\cdots,M\}$;$(\lambda_{i,1},\cdots,\lambda_{i,j})$为权重向量$\boldsymbol{\lambda}_i$在$j$维目标的权重。

2.7 IMOGWO算法框架

本文提出了一种改进的多目标灰狼优化算法(IMOGWO)求解基于优先级任务的卸载决策问题,并给出了IMOGWO算法的伪代码。首先,随机初始化灰狼种群和外部存档,应用标准的灰狼算子和个体进化停止准侧对NP个灰狼个体进行操作,然后对于新生成的NP个灰狼个体根据非支配准则选出非支配个体进入外部存档,并对外部存档进行进化与更新操作,当满足算法停止条件时结束。

假设函数最大评价次数为MF,任务数为N,边缘服务器数为M。算法共分为三部分:第一部分为灰狼个体进化策略,第二部分为外部存档更新策略,第三部分为最优解保存策略。在时间复杂度最坏情况下,灰狼群更新无法停止,此时第一部分的时间复杂度为$O(MF\times N)$,第二部分未进行进化操作,第三部分的时间复杂度为$O(M\times MF\times NP)$。综上所述,IMOGWO最坏情况下总时间复杂度为$O(MF(N+M\times NP))$。

算法:IMOGWO算法伪代码

输入:终端设备集合|U|,边缘服务器集合|E|,任务集合|S|,种群规模NP,最大评价次数MF

输出:外部存档ARC*

开始

(1)初始化终端设备、边缘服务器和任务特征信息

(2)随机初始化灰狼种群GP

(3)初始化外部存档ARC=GP

(4)**while** mg≤MF **do** // mg为当前函数评价次数

(5)**if** i≤NP **do**

(6)对灰狼种群GP中第i个灰狼个体X(i)应用灰狼新个体进化策略得到$X^*(i)$,并使$X^*(i)$进入种群GP*

(7)i=i+1

(8)**else**

(9)对新生成的灰狼种群GP^*应用非支配排序,得到非支配个体,进入外部存档ARC

(10)使用最优解保存策略更新外部存档ARC

(11)使用外部存档更新策略产生新的外部存

档ARC*

(12)使用最优解保存策略更新外部存档ARC*

(13)**end if**

(14)**end while**

结束

3　仿真分析

本文给出了6种算法方案求解计算卸载问题,分别为MPSO/D算法方案:新型基于分解的多目标粒子群算法;MOGWO/D算法方案:基于分解的多目标灰狼算法;MO-NSGA算法方案:文献[18]研究了物联网应用场景下的任务卸载和资源调度问题,考虑了任务请求优先级,优先级代表不同任务的重要性,并提出基于i-NSGA-Ⅱ的多目标优化算法(MO-NSGA)求解问题;Random方案:终端设备产生的任务随机决策为本地执行和卸载至边缘服务器执行;LOCAL方案:系统中终端设备产生的任务全部本地执行;本文提出的IMOGWO算法方案。

为公平起见,除LOCAL方案外,所有算法方案的种群规模NP=100,函数最大评价次数MF=50 000。算法参数设定均参照原论文,所有算法方案均在基于MATLAB的PlatEMO平台运行[24]。

3.1　参数设置

仿真实验中,终端设备数量(或任务数量)N设为50、100、200、400、600个,边缘服务器数量M设为2、4、6、8、10个。终端设备与边缘服务器之间的系统总带宽B=20 MHz,θ设为10种。每个任务都有优先级属性,其中pr_i={1,2,3}依次表示高、中、低3种优先级。此外,在t时间段内,边缘服务器中的任务分类器中可能存在待处理任务,本文假设队列Q_1、Q_2、Q_3中已有待任务的排队时延T_i^{qu}为[0,1]内的随机数。其他参数设置如表1所示,其中Rand(a,b)表示a到b之间的随机数。

表1 参数设置

Table 1　Parameter setting

参数	描述	取值
f_i^u	u_i的计算能力	Rand(0.3,0.31)GHz
d_i^{up}	s_i数据部分的上传数据量	Rand(10,30)MB
DP_i	s_i的处理程序数据量	Rand(200,400)MB
d_i^{down}	s_i的回传数据量	Rand(1,20)MB
t_i^{endure}	s_i的最大容忍时延	Rand(4,6)s
pr_i	s_i的优先级	{1,2,3}
BE_1	PR_1任务的社会收益	Rand(18,20)
BE_2	PR_2任务的社会收益	Rand(7,9)
BE_3	PR_3任务的社会收益	Rand(1,3)
f_j^e	e_j的计算能力	Rand(10,20)GHz
T_0^{pro}	e_j上的剩余任务处理时延	Rand(0,0.2)s

3.2　IMOGWO算法性能测试

为证明本文提出的算法性能优异,考虑对比MPSO/D、MOGWO/D、MO-NSGA和IMOGWO算法,求解不同任务数量和不同边缘服务器数量的卸载决策问题。每个多目标优化算法均独立运行30次。采用HV[25]指标评价算法性能,HV常用于评价多目标优化算法的收敛性和多样性,HV的值越大,表示算法的性能越佳。另外,本文应用Wilcoxon秩和检验验证4种算法的性能,显著性水平设为0.05。符号"+""−""≈"分别代表其他3种算法方案显著优于、显著劣于、无明显差异于IMOGWO算法方案,粗体表示算法获得的HV的最佳平均值Mean。

实验参数设置如下:不同任务数的卸载决策问题中任务数量N={50,100,200,400,600},边缘服务器数量M=4,实验比较算法所获HV的平均值,实验结果如表2所示;不同边缘服务器数的卸载决策问题中边缘服务器数量M={2,4,6,8,

表2 不同任务数量下4种算法所得HV平均值

Table 2　HV obtained by four algorithm schemes under different number of tasks

方案	MPSO/D	MOGWO/D	MO-NSGA	IMOGWO
N	Mean	Mean	Mean	Mean
50	6.641 9e−1−	5.964 1e−1−	6.625 1e−1−	**6.860 7e−1**
100	5.870 4e−1−	5.019 3e−1−	5.913 2e−1−	**6.247 8e−1**
200	4.194 7e−1−	3.726 0e−1−	4.207 3e−1−	**4.794 6e−1**
400	3.081 3e−1−	2.645 5e−1−	3.197 5e−1−	**3.646 0e−1**
600	2.567 0e−1−	2.121 9e−1−	2.688 0e−1−	**3.288 6e−1**
+/−/≈	0/5/0	0/5/0	0/5/0	

表3 不同边缘服务器数量下4种算法所得HV平均值

Table 3 HV obtained by four schemes under different number of edge servers

方案	MPSO/D	MOGWO/D	MO-NSGA	IMOGWO
M	Mean	Mean	Mean	Mean
2	2.630 8e−1−	1.651 9e−1−	2.501 9e−1−	**3.195 6e−1**
4	3.057 2e−1−	2.591 9e−1−	3.222 0e−1−	**3.670 4e−1**
6	3.364 1e−1−	3.162 2e−1−	3.486 6e−1−	**3.897 2e−1**
8	3.081 3e−1−	2.645 5e−1−	3.197 5e−1−	**3.646 0e−1**
10	3.564 3e−1−	3.119 0e−1−	3.379 9e−1−	**4.120 7e−1**
+/−/≈	0/5/0	0/5/0	0/5/0	

10},任务数量N=400,实验结果如表3所示。两个问题的权重均设为$w_1=0.5$、$w_2=0.5$。

由表2和表3中的4种算法在10种卸载决策问题上所获HV的平均值可以看出,多次重复实验下的MOGWO/D算法相较于其他3种算法所获的HV平均值最低,说明其算法寻优能力最弱,易陷入局部最优解。相比之下IMOGWO算法所获HV平均值相较于其他3种算法均最优,说明IMOGWO算法的搜素精度与广度均优于其他3种算法。并印证了灰狼个体的生成与更新策略、外部存档生成与更新策略、最优解保存策略这3种算法改进策略有效提升了多目标灰狼优化算法的搜索性能。

3.3 单个目标上算法性能比较

为直观观察本文所提算法方案在单个目标上的寻优能力,本文设计了不同任务数量情况下的算法性能比较和不同边缘服务器数量情况下的算法性能比较。

3.3.1 不同任务数量情况下的算法性能比较

实验参数设置如下:任务数量N={50,100,200,400,600},边缘服务器数量M=4,权重均设为$w_1=0.5$、$w_2=0.5$,实验从综合时延、社会损失率和负载失衡度三个方面进行比较,实验结果如图4~6所示。

由图4和图5可见,随着任务数量的增加,除LOCAL方案外,其他5个方案的综合时延和社会损失率显著增加。这是因为,随着任务数量的增加,在边缘网络的计算和通信总资源基本不变的情况下,边缘服务器中总的任务处理时延随之增加,单个任务占有的通信资源也会减少,任务传输时延变大。且在任务传输和处理时延变大的情况下,由于最大容忍时延的限制,任务总的处理成功率也会随之降低,总体社会损失率变大。另外,任务数量的增加,使得卸载问题规模变大,对算法的寻优能力要求更高。由图4可知,当任务数量为400时,LOCAL方案仅较IMOGWO方案表现差。这是因为MPSO/D、MOGWO/D、MO-NSGA和Random 4种算法的寻优能力变弱,所得卸载策略变差,导致任务排队时延变大,综合时延也随之变大。由图6可见,除LOCAL方案外,随着

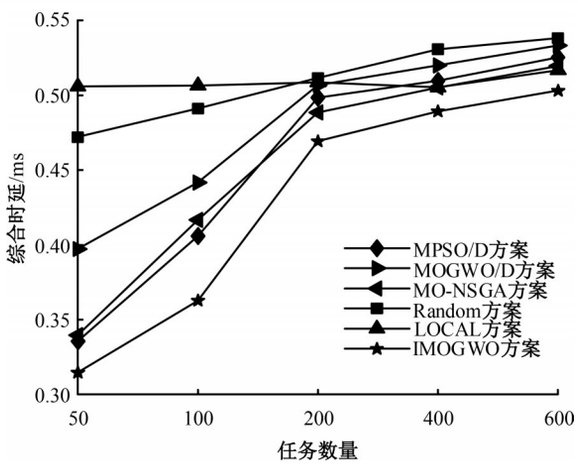

图4 不同任务数量下的综合时延对比

Fig. 4 Comparison of comprehensive delays by different number of tasks

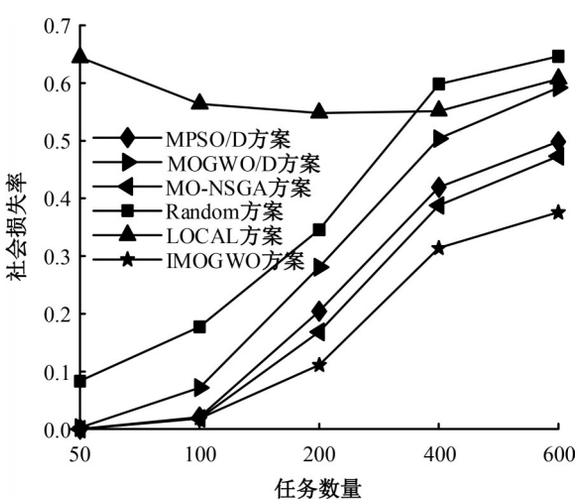

图5 不同任务数量下的社会损失率对比

Fig. 5 Comparison of social loss rate by different number of tasks

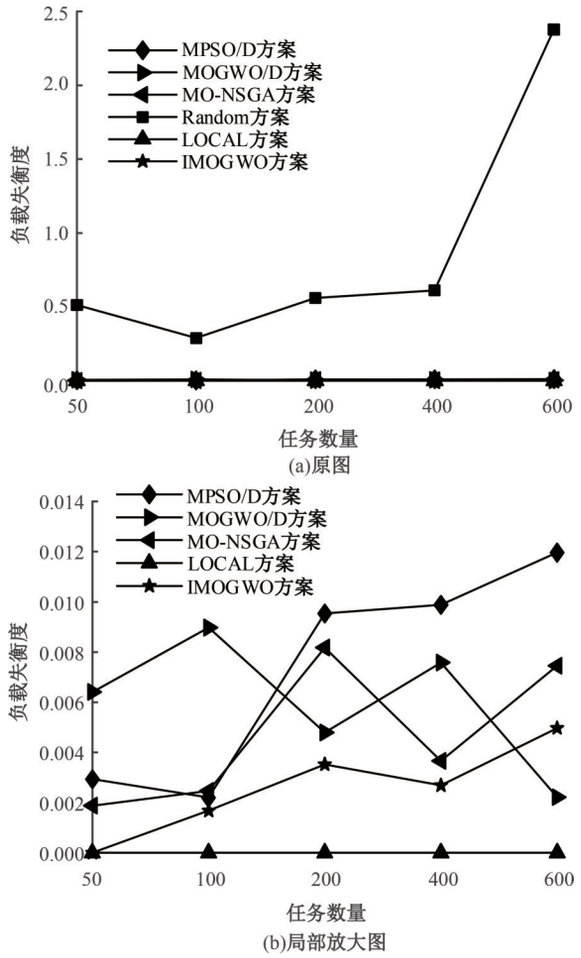

图 6　不同任务数量下的负载失衡度对比

Fig. 6　Comparison of load imbalance degree by different number of tasks

任务数量的增加,IMOGWO方案的负载失衡度在多数情况均优于其他4个方案。

综合图4~6可知,随着任务数量的增加,本文提出的IMOGWO方案整体表现均优于其他5个方案。也印证了表2所得结论的真实性。

3.3.2　不同边缘服务器数量情况下的算法性能比较

实验参数设置如下:任务数量$N=400$,边缘服务器数量$M=\{2,4,6,8,10\}$,权重均设为$w_1=0.5$、$w_2=0.5$,实验从综合时延、社会损失率和负载失衡度三个方面进行比较,实验结果如图7~9所示。

由图7和图8可见,因任务数量不变,LOCAL方案的综合时延和社会损失率均无变化。随着边缘服务器数量的增加,除LOCAL方案外,其他5个方案的综合时延和社会损失率均呈现下降趋势。这是因为,随着边缘服务器数量的增加,

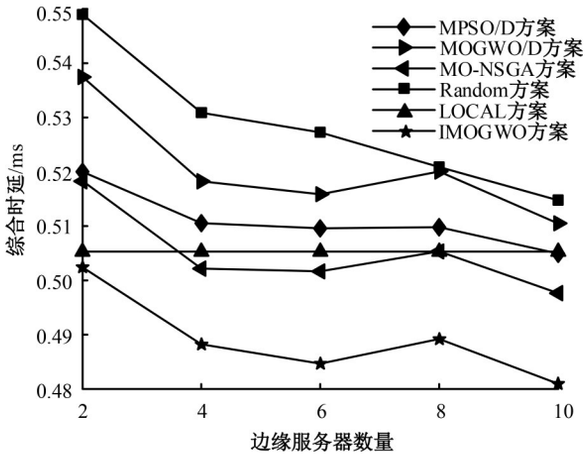

图 7　不同边缘服务器数量下的综合时延

Fig. 7　Comparison of comprehensive delays by different number of edge servers

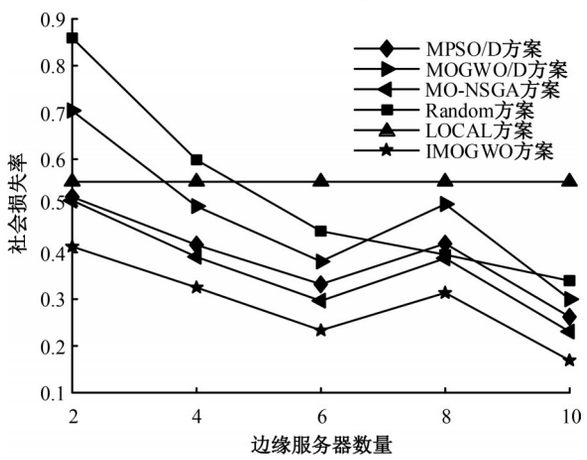

图 8　不同边缘服务器数量下的社会损失率对比

Fig. 8　Comparison of social loss rate by different number of edge servers

边缘端的计算和通信资源也随之增多,可为终端设备提供更加充足的资源,任务的传输和处理时延也随之变小,任务处理卸载成功率变大,总体社会损失率也降低。另外,IMOGWO方案的综合时延和社会损失率相较于其他5个方案均表现最优。

边缘服务器数量的增加,使得任务卸载到哪个服务器的选择增多,算法寻优能力要求变高。由图9可见,除LOCAL方案外,IMOGWO方案的负载失衡度均优于其他4个方案,且在$M=10$时优势最明显。说明IMOGWO方案会使任务在各个边缘服务器上的分配更均衡。

综合图7~9可知,随着边缘服务器数量的增加,本文提出的IMOGWO方案整体表现均优于其他5个方案,也印证了表3所得结论的真实性。

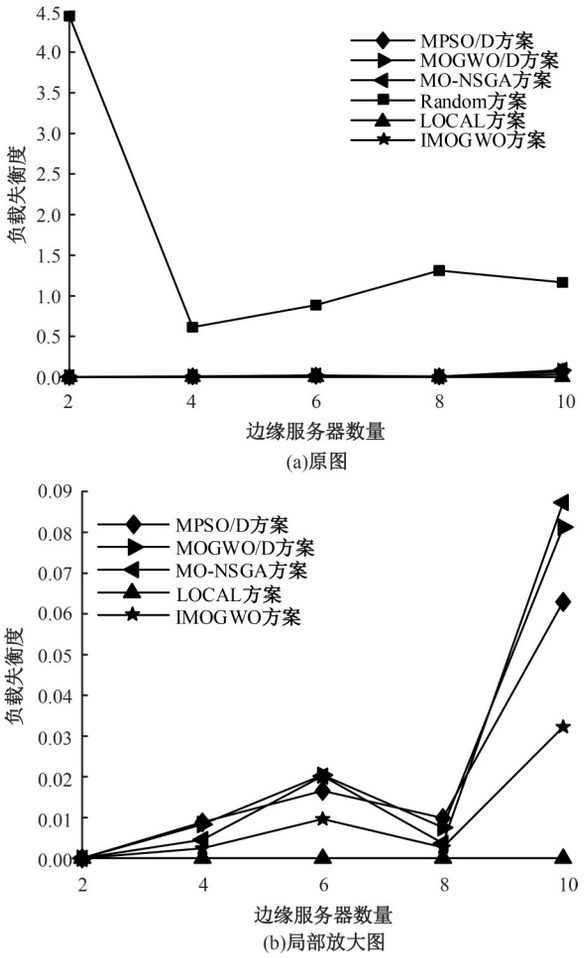

图9 不同边缘服务器数量下的负载失衡度对比

Fig. 9 Comparison of load imbalance degree by different number of edge servers

3.4 不同比例任务优先级对比

为探究本文所提任务优先级划分方案对社会损失率的影响,本节根据高、中、低3种优先级任务占总任务数的百分比不同设定了3种情况。情况1:高、中、低3种优先级任务分别占总任务数的20%、50%、30%;情况2:高、中、低3种优先级任务分别占总任务数的30%、20%、50%,情况3:高、中、低3种优先级任务占总任务数量的50%、30%、20%。3种情况的任务数量N=400,边缘服务器数量M=4,权重w_1=0.5、w_2=0.5,函数最大评价次数MF=20 000。3种情况所得社会损失率变化如图10所示。

由图10可见,随着高优先级任务占总任务量的比例逐渐增加,社会损失率逐渐降低,这与本文所提模型设定相互吻合,体现了本文提出的模型在物联网应用场景下处理紧急任务的优势。另外,IMOGWO方案在解决3种不同任务优先级比例的卸载决策问题时,均比其他5个方案获得的社会损失率低,再次印证了本文所提IMOGWO算法方案的优异性。

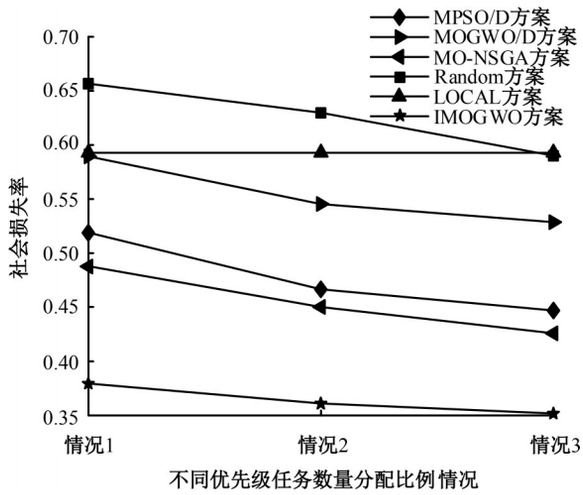

图10 3种情况所得社会损失率

Fig. 10 Social loss rate of three schemes

4 结束语

为满足物联网应用场景下紧急任务的处理需求,提出了根据任务关键程度划分任务优先级的思想,考虑了任务处理程序的卸载与缓存,设定了不同优先级任务卸载的传输带宽和任务处理顺序,构建了基于优先级任务的边缘计算卸载决策模型。在该模型的基础上,以最小化综合时延、社会损失率和负载失衡度为优化目标,构建了多目标优化卸载决策问题模型,提出了一种改进的多目标灰狼优化算法(IMOGWO)求解卸载问题。实验结果表明:本文提出的IMOGWO算法较MPSO/D、MOGWO/D和MO-NSGA算法具有更强的寻优能力,基于IMOGWO算法的卸载方案能够有效降低综合时延和社会损失率,并合理分配任务以保持各个边缘服务器的负载均衡,且性能较各对比方案均有所提升。未来的研究中,本文将研究不同应用场景下动态划分任务优先级的卸载决策模型。

参考文献:

[1] Liu P, Zhang Y F, Fu T T, et al. Intelligent mobile edge caching for popular contents in vehicular cloud toward 6G[J]. IEEE Transactions on Vehicular Technology, 2021, 70(6): 5265-5274.

[2] Sabella D, Vaillant A, Kuure P, et al. Mobile-edge

computing architecture: the role of MEC in the internet of things[J]. IEEE Consumer Electronics Magazine, 2016, 5(4): 84-91.

[3] Khan L U, Yaqoob I, Tran N H, et al. Edge-computing-enabled smart cities: a comprehensive survey[J]. IEEE Internet of Things Journal, 2020, 7(10): 10200-10232.

[4] Li M, Xiong N X, Zhang Y, et al. Priority-MECE: a mobile edge cloud ecosystem based on priority tasks offloading[J]. Mobile Networks and Applications, 2022, 27(3): 1768-1777.

[5] Xu X L, Gu R H, Dai F, et al. Multi-objective computation offloading for internet of vehicles in cloud-edge computing[J]. Wireless Networks, 2020, 26: 1611-1629.

[6] 朱思峰, 赵明阳, 柴争义. 边缘计算场景中基于粒子群优化算法的计算卸载[J]. 吉林大学学报: 工学版, 2022, 52(11): 2698-2705.
Zhu Si-feng, Zhao Ming-yang, Chai Zheng-yi. Computing offloading scheme based on particle swarm optimization algorithm in edge computing scene[J]. Journal of Jilin University (Engineering and Technology Edition), 2022, 52(11): 2698-2705.

[7] Liu Q, Mo R C, Xu X L, et al. Multi-objective resource allocation in mobile edge computing using PAES for internet of things[J]. Wireless Networks, 2020, 26(3): 1-13.

[8] 张秋平, 孙胜, 刘敏, 等. 面向多边缘设备协作的任务卸载和服务缓存在线联合优化机制[J]. 计算机研究与发展, 2021, 58(6): 1318-1339.
Zhang Qiu-ping, Sun Sheng, Liu Min, et al. Online joint optimization mechanism of task offloading and service caching for multi-edge device collaboration [J]. Journal of Computer Research and Development, 2021, 58(6): 1318-1339.

[9] 李燕君, 蒋华同, 高美惠. 基于强化学习的边缘计算网络资源在线分配方法[J]. 控制与决策, 2022, 37(11): 2880-2886.
Li Yan-jun, Jiang Hua-tong, Gao Mei-hui. Reinforcement learning-based online resource allocation for edge computing network[J]. Control and Decision, 2022, 37(11): 2880-2886.

[10] Lu H D, He X M, Du M, et al. Edge QOE: computation offloading with deep reinforcement learning for internet of things[J]. IEEE Internet of Things Journal, 2020, 7(10): 9255-9265.

[11] 韩旭. 基于优先级任务的电力物联网边缘计算任务卸载方法研究及实现[D]. 北京: 华北电力大学控制与计算机工程学院, 2022.
Han Xu. Research and implementation of edge computing task offloading method for power internet of things based on priority task[D]. Beijing: School of Control and Computer Engineering , North China Electric Power University, 2022.

[12] Adhikari M, Mukherjee M, Srirama S N. DPTO: a deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing[J]. IEEE Internet of Things Journal, 2019, 7(7): 5773-5782.

[13] 赵海涛, 朱银阳, 丁仪, 等. 车联网中基于移动边缘计算的内容感知分类卸载算法研究[J]. 电子与信息学报, 2020, 42(1): 20-27.
Zhao Hai-tao, Zhu Yin-yang, Ding Yi, et al. Research on content-aware classification offloading algorithm based on mobile edge calculation in the internet of vehicles[J]. Journal of Electronics & Information Technology, 2020, 42(1): 20-27.

[14] Hu S H, Li G H. Dynamic request scheduling optimization in mobile edge computing for IOT applications [J]. IEEE Internet of Things Journal, 2020, 7(2): 1426-1437.

[15] Lyu X C, Tian H, Jiang L, et al. Selective offloading in mobile edge computing for the green internet of things[J]. IEEE Network, 2018, 32(1): 54-60.

[16] 李智勇, 王琦, 陈一凡, 等. 车辆边缘计算环境下任务卸载研究综述[J]. 计算机学报, 2021, 44(5): 963-982.
Li Zhi-yong, Wang Qi, Chen Yi-fan, et al. A survey on task offloading research in vehicular edge computing[J]. Chinese Journal of Computers, 2021, 44(5): 963-982.

[17] Dai C, Wang Y P, Ye M. A new multi-objective particle swarm optimization algorithm based on decomposition[J]. Information Sciences, 2015, 325: 541-557.

[18] Zapotecas M S, Garcia N A, Lopez J A. Multi-objective grey wolf optimizer based on decomposition [J]. Expert Systems with Applications, 2019, 120(4): 357-371.

[19] Bi S Z, Huang L, Zhang Y J. Joint optimization of service caching placement and computation offloading in mobile edge computing systems[J]. IEEE Transac-

tions on Wireless Communications, 2020, 19(7): 4947-4963.

[20] 张德干, 李霞, 张捷, 等. 基于模拟退火机制的车辆用户移动边缘计算任务卸载新方法[J]. 电子与信息学报, 2022, 44(9): 3220-3230.
Zhang De-gan, Li Xia, Zhang Jie, et al. New method of task offloading in mobile edge computing for vehicles based on simulated annealing[J]. Journal of Electronics & Information Technology, 2022, 44(9): 3220-3230.

[21] Mirjalili S, Saremi S, Mirjalili S M, et al. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization[J]. Expert Systems with Applications, 2016, 47(5): 106-119.

[22] Liang Z P, Wang X Y, Lin Q Z, et al. A novel multi-objective co-evolutionary algorithm based on decomposition approach[J]. Applied Soft Computing, 2018, 73(12): 50-66.

[23] Wang J H, Zhang W W, Zhang J. Cooperative differential evolution with multiple populations for multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2015, 46(12): 2848-2861.

[24] Tian Y, Cheng R, Zhang X Y, et al. PlatEMO: a matlab platform for evolutionary multi-objective optimization educational forum[J]. IEEE Computational Intelligence Magazine, 2017, 12(4): 73-87.

[25] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach[J]. IEEE Transactions on Evolutionary Computation, 1999, 3(4): 257-271.