研究与开发

云边端协同下多用户细粒度任务卸载调度策略

谢满德，黄竹芳，孙浩
（浙江工商大学信息与电子工程学院，浙江 杭州 310018）

摘　要：为了解决当前处理多用户应用程序效率低下、密集网络资源利用率低及系统花费成本高等问题，提出了一种云边端协同下多用户细粒度任务卸载调度方法。该方法联合考虑了时延、能耗和服务器租用成本，先划分应用程序任务并设计子任务优先级，然后提出了多用户子任务调度方案，设计了一种改进的模拟退火粒子群算法求解最小系统总成本，从而实现最佳卸载决策。实验结果表明，所提方法相较于粒子群和模拟退火二元粒子群等其他方法，分别降低了至少12.28%和7.42%的总成本。

关键词：边缘计算；任务卸载；云边端协同；多用户；细粒度调度
中图分类号：TP393
文献标志码：A
doi: 10.11959/j.issn.1000−0801.2024086

Multi-user fine-grained task offloading scheduling strategy under cloud-edge-end collaboration

XIE Mande, HUANG Zhufang, SUN Hao
School of Information and Electronic Engineering, Zhejiang Gongshang University, Hangzhou 310018, China

Abstract: To solve the current problems of inefficiency, low utilization of intensive network resources, and high system cost in handling multi-user applications, a multi-user fine-grained task offloading scheduling approach under cloud-edge-end collaboration was proposed. Latency, energy consumption, and server rental costs were jointly considered. Application tasks were firstly divided and subtask priorities were designed. Then, a multi-user subtask scheduling scheme was proposed and an improved simulated annealing particle swarm algorithm was designed to minimize the total system cost to achieve the optimal offloading decision. Experimental results show that the proposed method reduces the total cost by at least 12.28% and 7.42% compared to other methods such as particle swarm and simulated annealing binary particle swarm, respectively.

Key words: edge computing, task offloading, cloud-edge-end collaboration, multi-user, fine-grained scheduling

收稿日期：2023−11−21；**修回日期：**2024−02−21
通信作者：孙浩，sunhao@zjgsu.edu.cn
基金项目：国家自然科学基金资助项目（No.61972352）；浙江省重点研发计划项目（No.2024C01025，No.2023C01220）；浙江省教育厅科研项目（No.Y202353408）
Foundation Items: The National Natural Science Foundation of China (No.61972352), the Key Research and Development Program of Zhejiang Province (No. 2024C01025, No. 2023C01220), the Scientific Research Fund of Zhejiang Provincial Education Department (No. Y202353408)

0 引言

随着无线通信技术和智能终端设备的广泛应用，越来越多的计算密集型应用程序出现，如增强现实、交互式游戏、语音识别等，这些应用程序需要消耗大量计算资源和能量。然而，用户设备的计算能力和电池容量是有限的，难以满足这些应用的计算需求[1]。因此，边缘计算已成为解决这一问题的重要技术，并在工业界和学术界受到了广泛的关注。边缘计算在靠近用户的网络边缘侧，部署具有较强计算能力和存储资源的基站或接入点，它允许用户将计算任务卸载到具有更强计算能力的网络边缘，从而解决用户设备资源受限问题，并降低用户设备时延和能耗，提升用户体验[2]。

边缘节点能够处理来自终端设备的部分任务，然而其计算资源是有限的。因此，我们需要充分利用云端的丰富资源，并合理地分配终端设备生成的任务，以满足用户的需求并降低整个系统的开销。此外，应用程序通常由多个具有依赖关系的子任务组成，这使得任务卸载问题更为复杂。所以，如何充分利用云边端计算资源并解决多用户细粒度任务卸载问题是一个新的挑战。

现有关于任务卸载的研究中，根据卸载应用的场景大致可以分成3类：（1）单服务器单用户场景。文献[3]考虑了单个移动设备到一个边缘服务器的场景，提出了一种基于离散二进制粒子群优化算法的系统负载联合优化方法。此外，文献[4-8]也是在单用户单服务器场景中研究任务的最优卸载决策。（2）单服务器多用户场景。如文献[9-12]，其中，文献[9]考虑了在单个移动边缘计算（mobile edge computing，MEC）服务器环境中设计一种多用户、多任务的任务卸载调度方法。（3）多用户多边缘服务器协作场景。文献[13]提出了基于量子微粒群优化方法，将任务有效地调度到多边缘服务器上协作处理，文献[14-16]同样考虑了多用户多边缘服务器任务卸载场景。这些研究大多将任务卸载到特定的边缘服务器执行或者仅仅利用边缘网络的资源，未充分利用密集边缘网络和远端云中的资源，导致设备资源利用率低。因此，在研究任务卸载时需要考虑云边端协作的计算场景。

在任务模型方面，一类研究主要考虑任务的粗粒度范畴，很少考虑任务之间的依赖关系，将任务作为一个整体放在本地或卸载到服务器执行，如文献[17-19]考虑的就是这种任务模型；另一类研究考虑了任务依赖关系，如文献[20]研究了顺序任务模型，考虑任务间的依赖性及多个用户间的资源竞争，提出了一种用于长期顺序任务的多用户计算卸载框架，文献[21]提出的模型要求子任务必须挨个执行，考虑的任务依赖模型也是较为简单的顺序依赖模型，文献[22]则考虑了一个应用程序依赖任务卸载模型。然而，所有这些研究没有考虑到多个复杂并行依赖任务模型的卸载问题。因此，对于多用户复杂依赖关系任务的研究有待进一步探索。

在卸载策略方面，研究者通常采用优化算法和人工智能方法来求解任务卸载问题。文献[23]针对依赖性任务模型提出了基于遗传算法的卸载调度策略，文献[24]提出了一种用于能耗优化的多资源计算卸载粒子群优化任务调度方法。这些传统的优化方法虽然可以得到一定程度的最优解，但容易陷入局部最优。当前也有一些研究[25-29]采用人工智能方法（如深度强化学习（deep reinforcement learning，DRL））来解决问题，文献[25]提出了基于深度强化学习方法将总计算时延最小化问题表述为0-1整数规划问题，并分解为子问题和顶层问题，同时利用DRL和优化技术以获得近似最优卸载方案。文献[26]提出了一种基于深度神经网络（deep neural network，DNN）的自适应负载平衡方法，将DNN按神经网络层粒度分割，计算密集型神经网络层被卸载

到云端执行，其他简单的神经网络层则在当地处理，该方法可以在不同计算节点之间自适应地分配任务，实现了高效计算。文献[29]提出了一种基于深度强化学习的云边协同移动计算卸载方法，该方法根据每个边缘云的差异设置共享经验池的优先级，选择最有效的经验样本以完成更好的学习和训练，通过云边协作优化网络参数以实现更快的最优卸载决策。人工智能方法在很多应用场景中获得了出色的效果，但是深度神经网络需要更大规模的数据集和昂贵的计算成本，而且很难部署到边缘设备，因此，大多数深度学习模型在云端部署，但纯云推理限制了深度学习（deep learning，DL）服务的普遍部署，它不能保证实时业务的延迟需求[30]。即使DeepWear[27]使用有向无环图（directed acyclic graph，DAG）来表示深度学习模型，通过修剪和分组简化复杂的DAG来实现选择最佳卸载分区的线性复杂性解决方案，但是，将DL模型的一部分上传到边缘节点仍然会严重延迟整个DL计算的卸载过程。因此，在卸载方法研究上还有很大的探索空间。

在卸载决策优化目标方面，当前研究中任务卸载目标主要集中在减少时延、降低能耗，最小化时延和能耗的加权和。文献[31]考虑到5G场景中毫秒级延迟的需求，以最小化所有设备任务执行延迟总和，文献[32]利用非正交多路访问传输效率高的优势，设计了基于非正交多址接入（non-orthogonal multiple access，NOMA）的计算卸载方案以降低完成所有计算任务产生的总体延迟。文献[33]提出了一种基于遗传的自适应粒子群算法来求解能耗最小化的卸载决策，文献[34]则提出了一种基于麻雀搜索算法来求解在满足任务时延要求的情况下，最小化系统能耗的卸载问题。文献[35]提出了一个在计算资源限制下以最小化延迟和能耗的加权和为目标的优化问题。实际上，将任务卸载至云端或边缘端执行，是需要花费额外的成本来租用这些服务器的。上述研究大多没有综合时延、能耗和服务器租用成本的性能权衡。此外，任务截止时间也是任务卸载中需要考虑的问题，现有关于截止时间的研究中，大多将截止时间设置为定值或一定范围内的随机值，没有根据具体应用程序任务特性来设置合适的截止时间，环境适应性较差。因此，在制定卸载决策时需联合考虑时延、能耗和服务器租用成本，同时也要保证任务在截止时间之前完成，从而为任务选择更加合理的卸载位置，提升系统的总体性能。

本文针对密集网络资源利用率低、多用户细粒度任务卸载问题的复杂性及优化目标的局限性等问题，研究了在云边端多重资源协同下的多用户细粒度任务卸载调度问题。在联合考虑时延、能耗和云配置成本下构建细粒度任务卸载模型，提出了基于云边端的多用户细粒度任务卸载调度策略。与其他方法相比，本文所设计的方法在保证用户基本需求的同时，有效地降低了系统的总成本。本文的主要贡献如下。

（1）构建了一个在延迟约束下优化能耗和服务器租用成本的模型，并提出了一种云边端协同下多用户细粒度任务卸载调度方法（multi-user fine-grained task offloading scheduling approach under cloud-edge-end collaboration，MUFGTOS-CEE），该方法在充分利用云边端多重计算资源的同时，有效地权衡了时延、能耗和服务器租用成本性能。

（2）为实现最小系统总成本的目标，设计了基于改进的模拟退火粒子群优化（improved simulated annealing particle swarm optimization，ISAPSO）算法来为所有用户设备上的子任务制定最佳卸载决策，实验结果表明，所提方法能较大幅度降低系统总成本。

（3）通过仿真实验，将本文方法与完全本地执行、云端完全卸载、多资源计算和卸载粒子群优化调度方法以及模拟退火二元粒子群优化方法

进行了对比，验证了本文方法的有效性。

1 系统模型与问题描述

本节首先描述了网络模型、任务依赖模型、时间模型、能耗模型和服务器成本模型；然后描述了云边端协同下多用户细粒度任务卸载问题；最后将其构建成以最小化总成本为目标的优化函数。本文涉及的主要符号与相关描述见表1。

表1 主要符号与相关描述

符号	描述
N	用户终端设备集
M	边缘服务器集
V	任务集
P	终端、边缘和云端设备编号集
$W_{n,v}$	终端设备n上子任务v的计算量
F_P	设备P的计算能力
$\mathrm{CT}_{n,v}^{P}$	设备P执行用户设备n上子任务v所需时间
$D_{n,v'v}$	设备n上子任务v'与任务v之间的数据传输量
r_{e}	边缘端与终端间的传输速率
r_{c}	云端与设备间的传输速率
$C_{n,v'v}$	设备n上子任务v'与任务v之间的数据传输时间
$D_{n,l}$	应用程序最终结果l传回终端n的数据量
$C_{n,l}$	传输应用程序最终结果l到终端n的所需时间
$\mathrm{AT}_{n,v}$	用户设备n上子任务v的开始时间
$\mathrm{FT}_{n,v}$	用户设备n上子任务v的完成时间
FT_n	用户设备n上应用程序完成时间
TU_P	设备P被使用的总时长
e_P	设备P的单位时间能耗率
TP_P	设备P的发射功率
EC_P	设备P的计算能耗消耗量
ET_P	设备P传输数据能量消耗量
TE	系统总能耗
EC_{P_n}	用户设备本地能耗消耗量
c	服务器单位时间使用成本
$C_{P'}$	服务器P'租用成本
C_{S}	购买服务器总成本
$T_{n,\ \mathrm{deadline}}$	用户设备n上应用程序的最大容忍时间
T_{P_n}	应用程序本地不卸载执行完成时间
C	系统总成本

1.1 网络模型

本文构建的多用户细粒度任务卸载网络模型如图1所示。假定网络中包含N个用户设备（$N=\{1,2,\cdots,N\}$）、M个边缘服务器（$M=\{1,2,\cdots,M\}$）和一个远端云服务器。每个用户设备上都有一个应用程序级任务需要完成，每个应用程序又由一系列具有依赖关系的子任务组成。在云边端协作模式下处理所有用户产生的任务，即用户的任务可以卸载至任意边缘端设备或远端云处执行，也可直接在本地进行处理。当存在依赖关系的子任务不在同一个节点上执行时，各节点之间需要进行数据传输，传输前继任务的计算结果到执行后继任务的节点上，作为执行后继任务的输入数据。本文模型中云端、边缘端，以及用户终端设备之间都可以进行数据传输。

1.2 任务依赖模型

通常情况下，用户设备上应用程序的不同任务之间存在一定的依赖关系。对于每个用户设备上的应用程序级任务，都可将其划分为多个具有依赖关系的子任务。本文采用有向无环图$G_n=(V_n,E_n)$来描述任务之间的依赖关系，其中，用户设备n有V个待处理子任务$V=\{1,2,\cdots,V\}$，E_n表示子任务之间的关系即有向边，用户任务示例如图2所示。图2展示了在线购物平台的订单处理系统的典型任务拓扑，用圆圈表示一个子任务，用有向边表示子任务之间的依赖关系。图中用户首先进行下单操作，当用户下单后，系统会同时进行库存检查和支付处理两个任务，它们之间没有依赖关系，可以并行执行，库存检查任务用于检查商品库存是否充足，支付处理任务用于处理用户的支付信息。物流处理与库存检查和支付处理两个子任务之间均存在依赖关系，物流处理任务的执行需要库存检查和支付处理后的输出数据作为输入数据，即只有在库存检查和支付处理完成，且库存检查和支付处理完成后的计算结果被传输到执行物流处理任务的节点之后，物流处理任务才能开始执行。

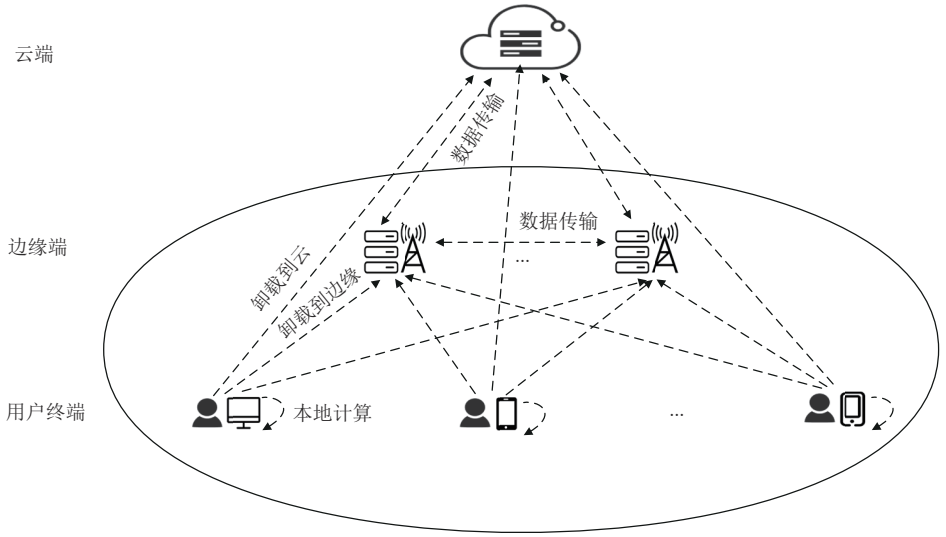

图1　多用户细粒度任务卸载网络模型

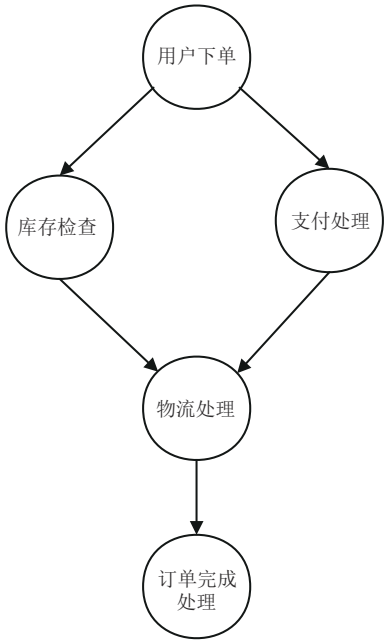

图2　用户任务示例

1.3　时间模型

设定应用程序初始子任务的开始时刻为0，用户设备$n\in N$，n上子任务v在第P个设备上的计算时延定义如式（1）所示。

$$\mathrm{CT}_{n,v}^{P}=\frac{W_{n,v}}{F_P} \tag{1}$$

其中，$P=\{0,1,\cdots,M+1\}$表示用户设备、边缘服务器集和云端服务器的编号，$W_{n,v}$为任务v的计算量，即计算任务v的CPU周期数，F_P表示设备P的计算能力（FLOPS）。

设备n上两个具有依赖关系的子任务之间数据传输时间计算方法如式（2）所示。

$$C_{n,v'v}=\begin{cases}0, P_{n,v'}=P_{n,v}\\ \dfrac{D_{n,v'v}}{r_{\mathrm{e}}}, P_{n,v'}\notin P_{\mathrm{c}}\text{且}P_{n,v}\notin P_{\mathrm{c}}\\ \dfrac{D_{n,v'v}}{r_{\mathrm{c}}}, P_{n,v'}\in P_{\mathrm{c}}\text{或}P_{n,v}\in P_{\mathrm{c}}\end{cases} \tag{2}$$

其中，$P_{n,v'}$和$P_{n,v}$分别表示子任务v'和v的执行位置，P_{c}表示在云端处理。当两个子任务不在同一设备上执行时，其传输时间为任务数据量与执行任务设备间的数据传输速率的比值，否则为0。

应用程序的最后子任务执行完成后需将结果传回用户终端，最终结果传回终端的传输时间计算方法如式（3）所示。

$$C_{n,l}=\begin{cases}0, P_{n,l}=P_n\\ \dfrac{D_{n,l}}{r_{\mathrm{e}}}, P_{n,l}\in P_{\mathrm{e}}\\ \dfrac{D_{n,l}}{r_{\mathrm{c}}}, P_{n,l}\in P_{\mathrm{c}}\end{cases} \tag{3}$$

其中，$P_{n,l}$表示处理用户设备n上最后子任务所在的设备编号，P_n、P_{e}和P_{c}分别表示用户设备编号集、边缘设备编号集和云端设备编号集。

子任务v需要在其直接前继子任务执行完成并将结果传到计算任务v的节点上且节点空闲时才可以开始执行，子任务v的开始时间计算方法如式（4）所示。

$$\mathrm{AT}_{n,v}=\max\left(\max_{v'\in\mathrm{pred}(v)}\left(C_{n,v'v}+\mathrm{FT}_{n,v'}\right),T_{\mathrm{idle}}^{P}\right) \quad (4)$$

其中，pred(v)表示任务v直接前继子任务集，T_{idle}^{P}表示处理器P的就绪时间。

子任务v的完成时间表示为任务在节点P上的开始执行时间与其计算时间之和，计算方法如式（5）所示。

$$\mathrm{FT}_{n,v}=\mathrm{AT}_{n,v}+\mathrm{CT}_{n,v}^{P} \quad (5)$$

用户设备n上整个应用程序的完成时间表示为最大子任务完成时间与计算结果回传时间之和，计算方法如式（6）所示。

$$\mathrm{FT}_{n}=\max_{v\in V}\left(\mathrm{FT}_{n,v}\right)+C_{n,l} \quad (6)$$

1.4 能耗模型

设备P被使用的总时长表示为该设备上需要处理的所有任务的计算时长之和，计算方法如式（7）所示。

$$\mathrm{TU}_{P}=\sum_{n=1}^{N}\sum_{v=1}^{V}\mathrm{CT}_{n,v}^{P} \quad (7)$$

设备P计算任务消耗的能耗表示为设备P被使用总时长与单位时间能耗率的乘积，计算方法如式（8）所示。

$$\mathrm{EC}_{P}=\mathrm{TU}_{P}\times e_{P} \quad (8)$$

设备P传输数据消耗的能耗表示为设备发射功率与数据传输时间的乘积，计算方法如式（9）所示。

$$\mathrm{ET}_{P}=\mathrm{TP}_{P}\times C_{n,v'v} \quad (9)$$

处理多用户应用程序系统花费的总能耗表示为所有设备的计算能耗与传输能耗之和，计算方法如式（10）所示。

$$\mathrm{TE}=\sum_{n=1}^{N}\mathrm{EC}_{P_n}+\sum_{P=1}^{M+1}\mathrm{EC}_{P}+\sum_{n=1}^{N}\sum_{v=1}^{V}\mathrm{ET}_{P} \quad (10)$$

1.5 服务器成本模型

在考虑设备完成任务产生的时延和能耗之外，还应考虑租用服务器的总成本。

服务器P'（$P'\in P_{\mathrm{e}}\cup P_{\mathrm{c}}$）的使用成本定义为服务器被占用时间与租用服务器单位时间成本的乘积，具体计算方法如式（11）所示。

$$C_{P'}=\mathrm{TU}_{P'}\times c \quad (11)$$

租用服务器总成本的计算方法如式（12）所示。

$$C_{\mathrm{S}}=\sum_{P'=1}^{M+1}C_{P'} \quad (12)$$

1.6 问题描述

本文研究了在云边端协同下，联合时延、能耗和服务器租用成本的多用户细粒度任务卸载问题，目标是为每个任务选择合适的卸载位置，使系统总成本最小。

权衡时延、能耗及租用服务器开销的多用户应用程序处理总成本的定义如式（13）所示。

$$C=\alpha\sum_{n=1}^{N}\mathrm{FT}_{n}+\beta\mathrm{TE}+\left(1-\alpha-\beta\right)C_{\mathrm{S}} \quad (13)$$

其中，α、β和$1-\alpha-\beta$分别表示时延、能耗和服务器成本在系统成本中的比重，α，$\beta\in[0,1]$，由于每个节点的权值可能不同，所以其具体参数值可以根据应用关注点通过实验来选择[29]。

优化目标可表示为：

$$\min_{P}C \quad \text{s.t. } \mathrm{FT}_{n}\leqslant T_{n,\mathrm{deadline}} \quad (14)$$

为不同用户应用程序设定不同的容忍时间，本文将其设定为完全本地不卸载执行策略下执行时间的λ倍，计算方法如式（15）和式（16）所示。

$$T_{n,\mathrm{deadline}}=\lambda T_{P_n} \quad (15)$$

$$T_{P_n}=\sum_{v=1}^{V}\mathrm{CT}_{n,v}^{P_n} \quad (16)$$

2 基于云边端多用户细粒度任务卸载调度方法

针对第1节所述问题，本文提出了一种

基于云边端多用户细粒度任务卸载调度方法（MUFGTOS-CEE），以实现在综合考虑时延、能耗和服务器成本的情况下，为所有用户应用程序上的每个子任务选择最佳卸载位置，使得系统总成本最小。首先，鉴于子任务的调度顺序影响系统总延迟，基于DAG任务，计算所有子任务优先级，以实现有依赖关系的子任务有序执行，无依赖关系的子任务可并行执行，提高任务处理效率，有效降低系统延迟；其次，按照优先级从高到低依次调度子任务，并基于改进的模拟退火粒子群算法求解子任务的最佳卸载位置。多用户细粒度任务卸载调度算法流程如图3所示。

2.1　子任务优先级计算

对于给定的DAG任务，需计算每个用户设备n上的子任务v的优先级，计算方法如式（17）所示。

$$S_{n,v}=\mathrm{MT}_{n,v}+\max_{v'\in \mathrm{succ}(v)}\left(S_{n,v'}+\mathrm{MC}_{n,vv'}\right) \tag{17}$$

其中，succ(v)表示子任务v的直接后继子任务集，$\mathrm{MT}_{n,v}$表示子任务v的平均执行时间，其计算方法如式（18）所示。$\mathrm{MC}_{n,vv'}$表示子任务v与后继任务v'间的平均数据传输时间，其计算方法如式（19）所示。

$$\mathrm{MT}_{n,v}=\frac{\sum_{P=0}^{M+1}\mathrm{CT}_{n,v}^{P}}{M+2} \tag{18}$$

$$\mathrm{MC}_{n,vv'}=\frac{\frac{D_{n,vv'}}{r_{\mathrm{e}}}+\frac{D_{n,vv'}}{r_{\mathrm{c}}}}{2} \tag{19}$$

$S_{n,v}$大于$S_{n,v'}$，v'的执行需要v的输出数据，因此只要优先调度S值大的子任务，就可以保证有依赖关系的子任务有序执行，无依赖关系的子任务并行执行，提高效率，有效减少处理任务延迟。

对于不同用户应用程序上的子任务，应优先卸载具有较高计算开销和本地能耗的子任务，以此缩短延迟并节省更多的本地能耗。因此，本文进一步定义子任务的优先级如式（20）所示。

$$R_{n,v}=\gamma S_{n,v}+\left(1-\gamma\right)E_{n,v} \tag{20}$$

其中，$E_{n,v}$表示设备n上处理子任务v的能量消耗量，γ可以根据实际需求自定义，$E_{n,v}$的计算如式（21）所示。

$$E_{n,v}=\mathrm{CT}_{n,v}^{P_n}\times e_{P_n} \tag{21}$$

2.2　多用户细粒度任务卸载调度方法

本文设计的MUFGTOS-CEE实现了所有用户应用程序子任务的卸载调度，该方法为所有用户子任务做出最优卸载决策，从而优化处理多用户应用程序产生的总成本。该方法的基本步骤如下。

步骤1　计算子任务的优先级并排序；

步骤2　对于每个用户应用程序，选择当前子任务排序序列中最高优先级的未被调度的子任务，形成候选调度集；

步骤3　选择调度集中具有最高优先级的子任务，并基于改进的模拟退火粒子群算法得出最优卸载位置，将该子任务调度到对应的最佳位置；

步骤4　重复此过程直至所有子任务计算完成。

MUFGTOS-CEE方法如算法1所示。

算法1　MUFGTOS-CEE方法

输入：设备集P，任务数量V，任务拓扑图$G_n=(V_n,E_n)$，每个子任务输出数据量$D_{n,v}$，数据传输速率r，能耗率e_P，子任务在不同设备上的执行时间$\mathrm{CT}_{n,v}^{P}$

输出：所有用户应用程序各子任务的卸载决策和系统总成本

（1）　根据式（17）计算子任务优先级$S_{n,v}$；

（2）　**while**存在未调度子任务

（3）　从每个G_n中选择具有最高优先级$S_{n,v}$的未调度子任务，形成一个候选调度集D；

（4）　根据式（20）计算优先级$R_{n,v}$，从D中按照优先级$R_{n,v}$高低依次调度子任务v；

（5）　使用算法2计算得出子任务v的最佳卸载位置；

（6）　将子任务卸载至最佳位置；

（7）　将该子任务标记为已调度子任务；

（8）　**end while**

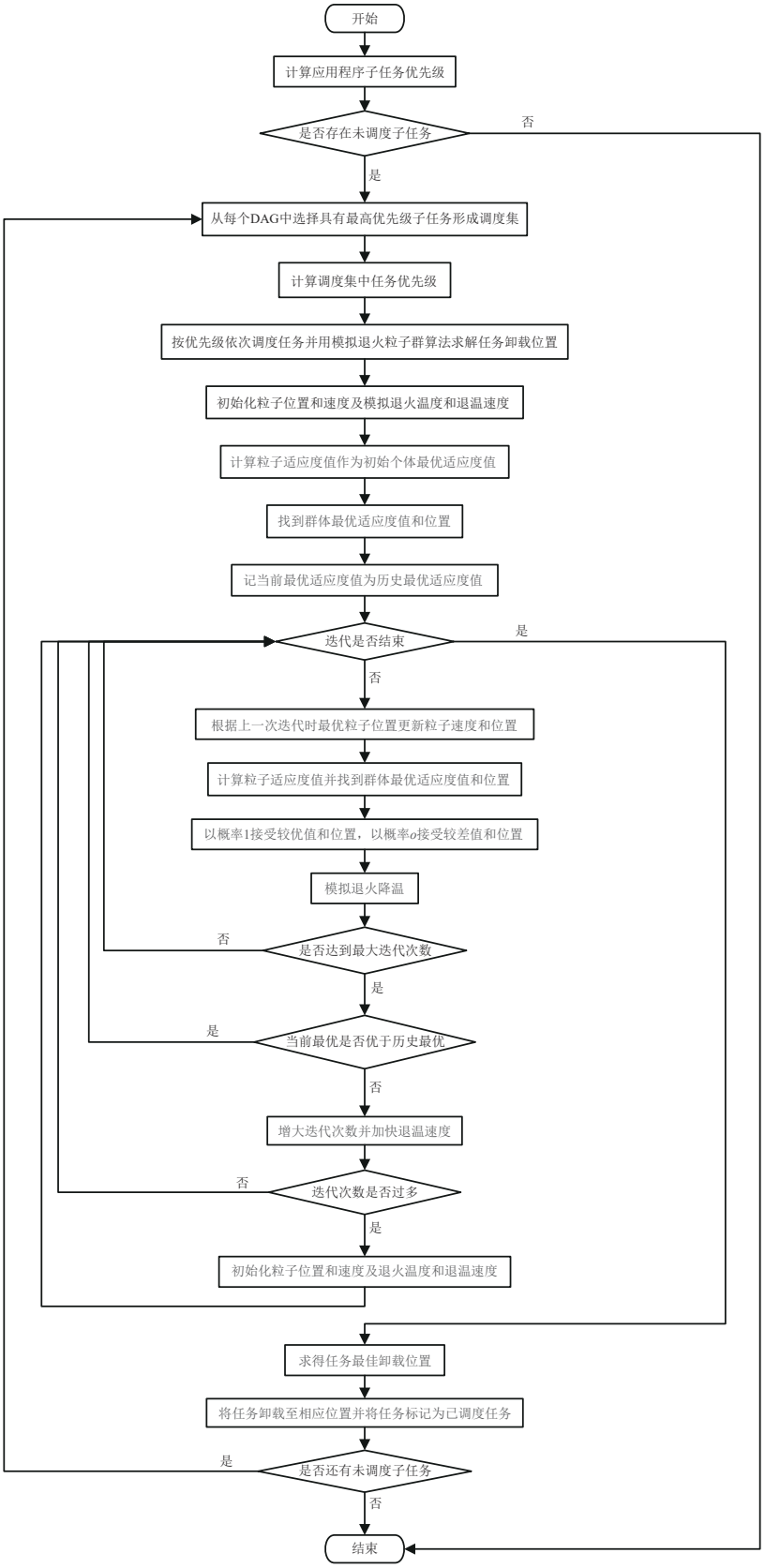

图3　多用户细粒度任务卸载调度算法流程

算法2给出了基于改进的模拟退火粒子群算法ISAPSO。针对粒子群容易局部收敛的缺点，将模拟退火融合到粒子群算法之中，并加入历史最优值的判断，判断算法是否陷入了局部最优解。如果在达到最大迭代次数时，当前最优解差于历史最优解，就增加迭代次数，从而扩大其搜索范围，使其跳出局部最优。与此同时，当长时间都没有跳出局部最优时，重新随机初始化粒子群，从而提高算法的收敛精度和准确性，以此高效求解云边端协同下联合时延、能耗和服务器成本的子任务最优卸载决策。ISAPSO算法具体流程如算法2所示。

算法2　ISAPSO算法

输入：最大迭代次数MaxNum，种群大小ParticleSize，数据传输速率r，服务器能耗率e_P，服务器使用价格c

输出：联合时延、能耗和服务器使用成本的最优卸载方案Best和与之对应的适应度值BestValue

（1）随机初始化种群中各粒子的位置x和速度v；

（2）计算粒子适应度值f并初始化粒子个体最优适应度值p_BestValue和位置x_p；

（3）找到群体最优适应度值g_BestValue和位置x_{g_Best}；

（4）历史最优BestValue=g_BestValue；

（5）初始化退火温度$T=\max(f)-\min(f)$和退温速度s=0.9；

（6）　**while** T==0

（7）　重新初始化粒子群算法；

（8）　重新初始化退火温度T；

（9）　**end while**

（10）　t=1；

（11）　**while** t!=MaxNum

（12）　**for** k=1:ParticleSize

（13）　根据最优粒子位置更新其速度：$v_k^{t+1}=w\times v_k^t+c_1\times\text{rand}(0,1)\times(x_p-x_k^t)+c_2\times\text{rand}(0,1)\times(x_{g\_Best}-x_k^t)$；

（14）　更新粒子位置：$x_k^{t+1}=x_k^t+v_k^{t+1}$；

（15）　计算适应度值f；

（16）　比较f与p_BestValue大小，以概率1接受较优值和位置，以概率o（$o=\mathrm{e}^{\frac{-|f-p\_\text{BestValue}|}{T}}$）接受较差值和位置；

（17）　找到群体最优适应度值g_BestValue1与位置x_{g_Best1}；

（18）　比较g_BestValue1与g_BestValue大小，以概率1接受较优值和位置，以概率$o=\mathrm{e}^{\frac{-|g_\text{BestValue1}-g_\text{BestValue}|}{T}}$接受较差值和位置；

（19）　降温$T=s\times T$；

（20）　$t=t+1$；

（21）　**if** t==MaxNum && g_BestValue>BestValue

（22）　迭代结果差于历史最优，增大迭代次数重新迭代：MaxNum=t+MaxNum；

（23）　$T=T/s$；

（24）　$s=s\times0.9$加快退温速度；

（25）　**if** s<0.5

（26）　迭代次数过多且未得到最优值，重新初始化迭代；

（27）　重新随机初始化粒子位置x和速度v；

（28）　计算粒子适应度值f；

（29）　重新初始化退火初始温度T；

（30）　**end if**

（31）　**end if**

（32）　**if** g_BestValue<BestValue

（33）　BestValue=g_BestValue；

（34）　**end if**

（35）　**end while**

3 仿真验证与结果分析

3.1 实验设置

本文通过MATLAB R2018b仿真软件搭建仿真环境，仿真平台参数为：内存8 GB，CPU型号Intel(R) Core(TM) i5-6200U，主频2.3 GHz，显卡Intel(R) HD Graphics 520。本节模拟了一个云边端协同计算系统，包括1个云端服务器、2个边缘服务器和50个终端用户设备。仿真参数设定见表2。

表2 仿真参数设定

参数	数值
终端设备数N	50
边缘服务器数M	2
任务数V	[30,180]
边缘端与终端间传输速率r_e/(MB·s^{-1})	25
云端与设备间传输速率r_c/(MB·s^{-1})	5
任务计算量$W_{n,v}$/megacycles	[10,60]
任务数据量$D_{n,v}$/MB	[0.02,4.00]
终端设备计算能力F_{P_n}/GHz	1
边缘端设备最大计算能力F_{P_e}/GHz	5
云端服务器最大计算能力F_{P_c}/GHz	10
终端设备发射功率TP_{P_n}/W	0.1
边缘服务器发射功率TP_{P_e}/W	0.6
云服务器发射功率TP_{P_c}/W	3
终端设备单位时间能耗率e_{P_n}/(J·s^{-1})	0.6
边缘服务器单位时间能耗率e_{P_e}/(J·s^{-1})	0.8
云端服务器单位时间能耗率e_{P_c}/(J·s^{-1})	1.6
边缘服务器单位时间订购成本c_{P_e}/(分·s^{-1})	0.8
云端服务器单位时间订购成本c_{P_c}/(分·s^{-1})	1.2

本节分别设计了如下4种方案。

（1）完全本地计算（fully local computing，FLC）策略：所有用户设备上的任务都放在本地处理。

（2）云端完全卸载（cloud fully offloading，CFO）策略：所有用户设备上的任务都卸载到云端处理。

（3）能量优化多资源计算和卸载粒子群优化任务调度方法[24]（energy-optimized multi-resource computing and offloading particle swarm optimization task scheduling，EMO）：用粒子群算法来解决多资源任务卸载调度问题，在考虑响应时间约束下，充分降低用户设备的能耗。

（4）模拟退火二元粒子群优化方法[17]（simulated annealing-binary particle swarm optimization，SA-BPSO）：将模拟退火的Metropolis准则引入粒子群算法中，以有效解决任务卸载问题。

将上述方案与本文所提出的MUFGTOS-CEE方案进行对比实验，从能耗、时间、服务器租用成本和系统总成本4个方面评估本文MUFGTOS-CEE策略的有效性。

3.2 实验结果与分析

3.2.1 系统能耗

系统能耗表示处理所有用户应用程序的过程中所有设备消耗的能量总和。系统能耗受到用户数量和用户子任务数量因素影响。在不同卸载策略下，用户设备数对系统能耗的影响如图4所示，从中可以看出本文方法MUFGTOS-CEE的总能耗始终是最小的。任务卸载到边缘端或云端处理时等待过程的能耗可忽略不计。本文方法相较于FLC策略，由于将大量任务卸载至具有更强计算能力的边缘及云端执行，能较快地得到任务计算结果，总体消耗的能耗相较于FLC也就会随着用户数的增大而降低，降低了约34.21%的能耗；CFO策略下的系统能耗始终是最高的，这是因为用户将所有任务完全卸载至远端云处执行，传输数据量大且传输链路长，因此总体能耗也就显著提高。由于EMO与SA-BPSO解空间较为局限，容易陷入局部最优问题，所以在图4中可以看到，本文方法相较于EMO和SA-BPSO方法分别降低了至少12.26%和7.39%的能耗，在能耗方面具有优越性。

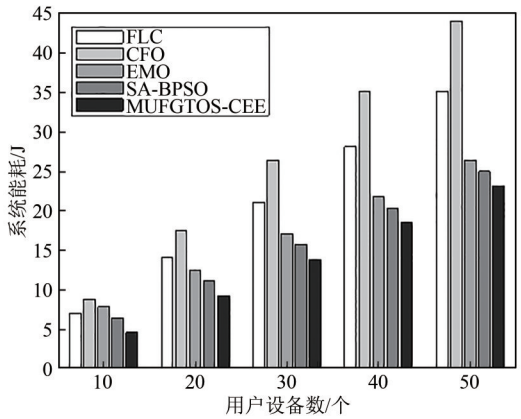

图4　用户设备数对系统能耗的影响

在不同卸载策略下，任务数对系统能耗的影响如图5所示，随着任务数目的增多，所有方案的系统能耗增大。相较于FLC，本文方法实现了有依赖关系的任务有序执行，无依赖关系任务在云边端分工协作执行，能节省大量时间。因此，随着任务数的增多，本文方法比FLC具有更明显的能耗优势。与图4结果有相似的原因，CFO的能耗始终是最高的，同时本文方法引入历史最优可以有效地避免陷入局部最优而导致精度下降的情况，相较于EMO和SA-BPSO方法能耗更低，分别降低了约13.99%和8.54%的能耗。

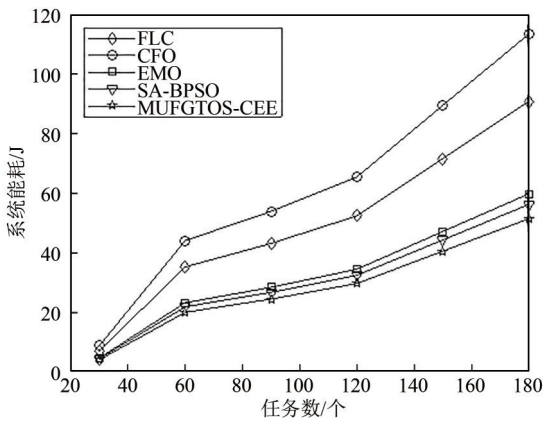

图5　任务数对系统能耗的影响

3.2.2　系统时延

系统时延表示处理所有用户设备上应用程序的总时延。用户设备数对系统时延的影响如图6所示，任务数对系统时延的影响如图7所示。从图6和图7可以看出，本文方法相较FLC具有较低的时延，由于多用户细粒度任务处理的复杂性，基于云边端协作的方法无论是用户数还是任务数的增多，相较于本地不卸载策略显著提高了任务的处理效率，系统时延也就相对较低，降低了至少60.52%。此外，相比于CFO也具有较低的时延。这是由于基于云边端协作的策略在数据量较大且传输速率受限的情况下，更多的任务会卸载至边缘端处执行，会节省大量的数据传输时延，总体时延也就较低，降低了至少21.05%。从图6和图7中仍然可以看出本文方法相比EMO和SA-BPSO方法的有效性，分别降低了至少12.26%和7.39%的时延。本文方法较适用于图像处理、视频编码等资源密集型任务的应用。

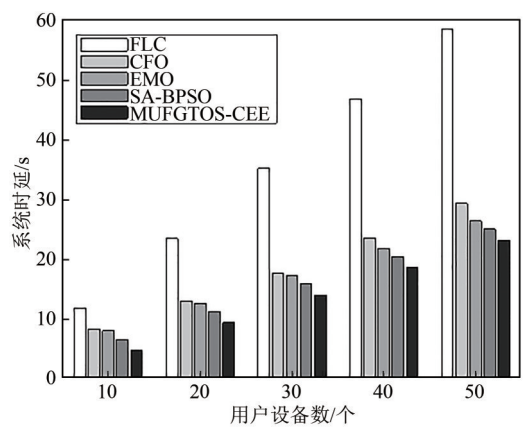

图6　用户设备数对系统时延的影响

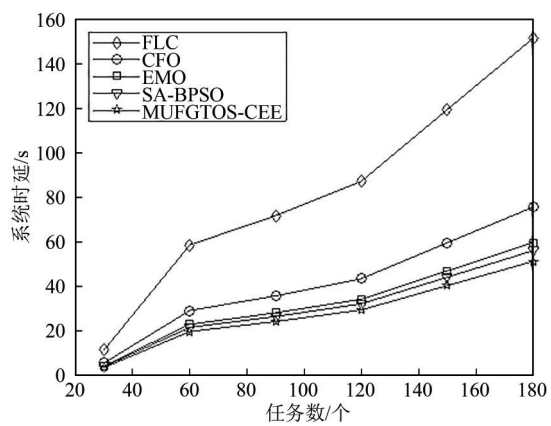

图7　任务数对系统时延的影响

3.2.3 云配置成本

云配置成本表示在处理各用户设备应用程序任务时租用服务器的开销。用户设备数对云配置成本的影响如图8所示，任务数对云配置成本的影响如图9所示。图8和图9中，除FLC以外的所有方案都会带来云配置成本，用户数和任务数的增多都会给系统带来更大的时延，因此租用服务器的时间就相对越长，相应的云配置成本也就越高。完全本地执行即FLC策略只在用户设备本地进行任务的处理，不占用服务器资源，因此，FLC策略的云配置成本始终为0。实际上，由于边缘服务器的计算性能较云端服务器的计算性能低，所以其租用成本也相对较低。本文方案将一部分任务放到本地或边缘处去执行，而不是将所有任务都放到云端执行，所以其放在云端的任务计算量较小，相应地减少了占用云端服务器资源的时间，且本地执行的任务的云配置成本为0。因此，本文方案与完全卸载到云端执行的CFO方案相比，其云配置成本始终最低，降低了约59.77%的服务器租用开销。

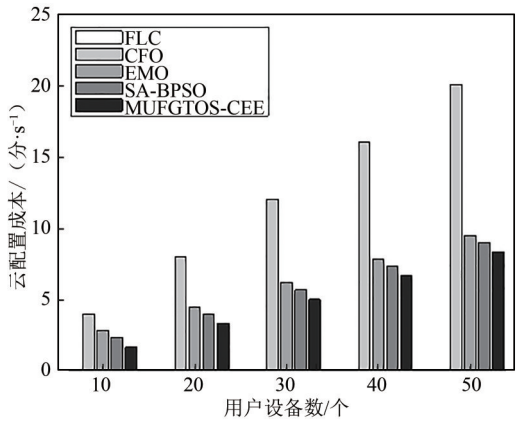

图8 用户设备数对云配置成本的影响

此外，由于多用户细粒度任务卸载的烦琐性，很难在短期内搜索获得全局最优解，而本文方法通过记录历史最优值来指导搜索方向，从而提高了算法的收敛精度和准确性，因此相比于EMO和SA-BPSO方法具有更高的性能，能更快地完成任务，占用服务器资源时间相对较少，租用服务器的成本也就较低，分别降低了至少12.27%和7.40%的租用开销。

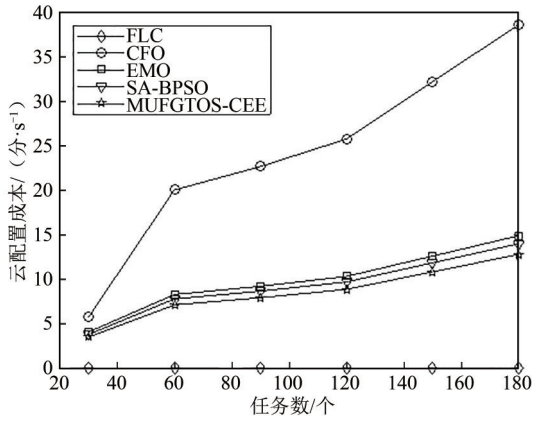

图9 任务数对云配置成本的影响

3.2.4 系统总成本

本文方法有效性的评估需综合考虑能耗、时延和云配置成本。用户设备数对系统总成本的影响如图10所示，其中，FLC策略的总成本始终最高，这是由于所有任务都在本地执行，时延远高于其他卸载方案；CFO策略的超高能耗和租用成本以及较高的时延使得其总成本相比于云边端部分卸载策略的总成本高；此外，本文方法相比EMO和SA-BPSO方法分别降低了至少12.28%和7.42%的总成本。

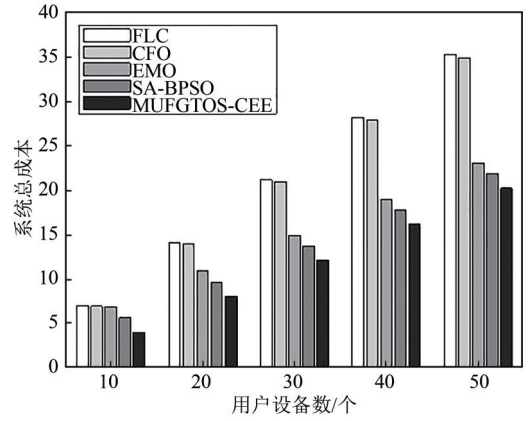

图10 用户设备数对系统总成本的影响

任务数对系统总成本的影响如图11所示，其中，FLC和CFO在任务数的影响下，由于各自在

能耗、时延和云配置成本的差距总和相近，所以其总成本不相上下；本文方法在满足用户对时延基本需求的情况下，有效降低了系统能耗和云配置成本，相比 EMO 和 SA-BPSO 方法，总成本也始终最低，分别降低了约 14.09% 和 8.61%。

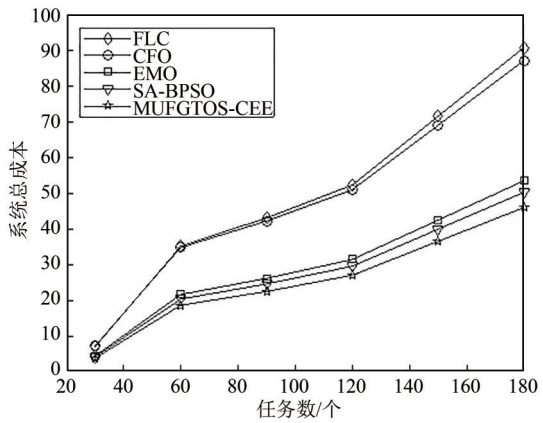

图 11　任务数对系统总成本的影响

为了更好地评估本文方法的性能，扩大用户规模，进一步对比验证用户数对系统总成本的影响。用户设备规模对系统总成本的影响如图 12 所示。用户数从 20 增大到 100，可以看出本文 MUFGTOS-CEE 方法与 FLC、CFO 方法相比具有较大的优势，且这种优势随着用户数的增多而更加明显。当用户数为 20 时，MUFGTOS-CEE 方法的系统总成本相较 FLC 和 CFO 的系统总成本分别降低了约 6.00 和 5.85；当用户数为 100 时，MUFGTOS-CEE 方法的系统总成本相较 FLC 和 CFO 的系统总成本分别降低了约 30.43 和 28.96，与用户数为 20 相比，降低了约 23 的成本。此外，本文方法相比 EMO 和 SA-BPSO 方法具有更好的性能，在用户设备数为 100 时比在用户设备数为 20 时分别降低了约 1.21 和 0.81 的总成本，总体分别降低了约 14.39% 和 8.72%。

4　结束语

本文针对密集网络资源利用率低、多用户细粒度任务卸载问题的复杂性及优化目标的局限性

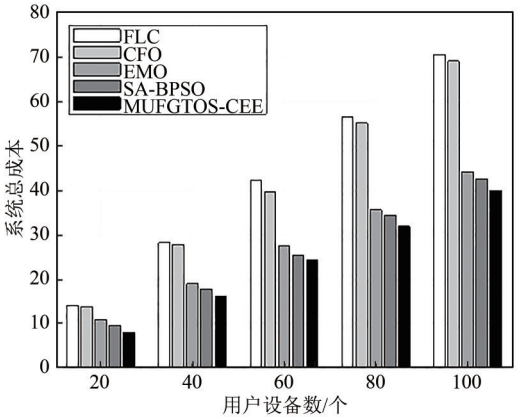

图 12　用户设备规模对系统总成本的影响

等问题，提出了一种基于云边端协同的多用户细粒度任务卸载调度方法。该方法充分利用了云边端多重计算资源，并有效权衡时延、能耗和服务器租用成本性能，将用户任务划分为多个具有依赖关系的子任务，基于子任务优先级，提出了多用户子任务调度优化方案。本文设计了一种 ISAPSO 算法求解子任务最小总成本的最佳卸载决策。仿真结果验证了本文方法的有效性，结果表明，本文方法在满足用户对应用程序执行时延的基本需求情况下，有效地降低了任务处理总能耗和云配置成本，并提高了服务器计算资源的利用率。本文只考虑了用户设备上单个应用程序的卸载问题，未来将考虑多应用程序协同卸载场景。

参考文献：

[1] LIU J, MAO Y Y, ZHANG J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT). Piscataway: IEEE Press, 2016: 1451-1455.

[2] 施巍松, 孙辉, 曹杰, 等. 边缘计算: 万物互联时代新型计算模型[J]. 计算机研究与发展, 2017, 54(5): 907-924.
SHI W S, SUN H, CAO J, et al. Edge computing—an emerging computing model for the Internet of everything era[J]. Journal of Computer Research and Development, 2017, 54(5): 907-924.

[3] TANG X, WEN Z, CHEN J L, et al. Joint optimization task offloading strategy for mobile edge computing[C]//Proceedings

of the 2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA). Piscataway: IEEE Press, 2021: 515-518.

[4] WANG F, XU J, CUI S G. Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems[J]. IEEE Transactions on Wireless Communications, 2020, 19(4): 2443-2459.

[5] MENG H, CHAO D C, GUO Q Y. Deep reinforcement learning based task offloading algorithm for mobile-edge computing systems[C]//Proceedings of the Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence. New York: ACM Press, 2019: 90-94.

[6] 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.

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

[8] MAO Y Y, ZHANG J, LETAIEF K B. Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems[C]//Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC). Piscataway: IEEE Press, 2017: 1-6.

[9] 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.
KUANG Z F, CHEN Q L, LI L F, et al. Multi-user edge computing task offloadingscheduling and resource allocation based on deep reinforcement learning[J]. Chinese Journal of Computers, 2022, 45(4): 812-824.

[10] LIANG Z Z, LIUT Y, HUANG K B, et al. I/O interference aware multiuser computation offloading for virtualized edge computing[C]//Proceedings of the ICC 2019 - 2019 IEEE International Conference on Communications (ICC). Piscataway: IEEE Press, 2019: 1-6.

[11] BI S Z, HUANG L, ZHANG Y J A. Joint optimization of service caching placement and computation offloading in mobile edge computing systems[J]. IEEE Transactions on Wireless Communications, 2020, 19(7): 4947-4963.

[12] LU S F, LIU S, ZHU Y J, et al. A DRL-based decentralized computation offloading method: an example of an intelligent manufacturing scenario[J]. IEEE Transactions on Industrial Informatics, 2022, 19(9): 9631-9641.

[13] DONG S, XIA Y J, KAMRUZZAMAN J. Quantum particle swarm optimization for task offloading in mobile edge computing[J]. IEEE Transactions on Industrial Informatics, 2022, 19(8): 9113-9122.

[14] GAN Q, LI G X, HE W H, et al. Delay-minimization offloading scheme in multi-server MEC networks[J]. IEEE Wireless Communications Letters, 2023, 12(6): 1071-1075.

[15] LI H L, XU H T, ZHOU C C, et al. Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment[J]. IEEE Transactions on Vehicular Technology, 2020, 69(9): 10214-10226.

[16] SONG Z Y, LIU Y W, SUN X. Joint task offloading and resource allocation for NOMA-enabled multi-access mobile edge computing[J]. IEEE Transactions on Communications, 2020, 69(3): 1548-1564.

[17] CHEN X, ZHENG S. Resource allocation and task offloading strategy base on hybrid simulated annealing-binary particle swarm optimization in cloud-edge collaborative system[C]//Proceedings of the 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). Piscataway: IEEE Press, 2022: 379-383.

[18] LI X, HUANG L, WANG H, et al. An integrated optimization-learning framework for online combinatorial computation offloading in MEC networks[J]. IEEE Wireless Communications, 2022, 29(1): 170-177.

[19] CHEN G J, WU Q Q, LIU R Q, et al. IRS aided MEC systems with binary offloading: A unified framework for dynamic IRS beamforming[J]. IEEE Journal on Selected Areas in Communications, 2022, 41(2): 349-365.

[20] XU H H, ZHOU J Y, WEI W Q, et al. Multiusercomputation offloading for long-term sequential tasks in mobile edge computing environments[J]. Tsinghua Science and Technology, 2022, 28(1): 93-104.

[21] LI X, FAN R F, HU H, et al. Joint task offloading and resource allocation for cooperative mobile-edge computing under sequential task dependency[J]. IEEE Internet of Things Journal, 2022, 9(23): 24009-24029.

[22] 张俊娜, 鲍想, 陈家伟, 等. 一种联合时延和能耗的依赖性任务卸载方法[J]. 计算机研究与发展, 2023, 60(12): 2770-2782.
ZHANG J N, BAO X, CHEN J W, et al. A dependent task offloading method forjoint time delay and energy consumption[J]. Journal of Computer Research and Development, 2023, 60(12): 2770-2782.

[23] AL-HABOB A A, DOBRE O A, ARMADA A G, et al. Task scheduling for mobile edge computing using genetic algorithm and conflict graphs[J]. IEEE Transactions on Vehicular Technology, 2020, 69(8): 8805-8819.

[24] WEI Z, YU X B, ZOU L. Multi-resource computing offload strategy for energy consumption optimization in mobile edge computing[J]. Processes, 2022, 10(9): 1762.

[25] ZHENG K C, JIANG G D, LIU X Y, et al. DRL-based offloading for computation delay minimization in wireless-powered multi-access edge computing[J]. IEEE Transactions on Communications, 2023, 71(3): 1755-1770.

[26] CHEN X, LI M, ZHONG H, et al. DNNOff: offloading DNN-based intelligent IoT applications in mobile edge computing[J]. IEEE Transactions on Industrial Informatics, 2022, 18(4): 2820-2829.

[27] XU M, QIAN F, ZHU M, et al. Deepwear: adaptive local offloading for on-wearable deep learning[J]. IEEE Transactions on Mobile Computing, 2019, 19(2): 314-330.

[28] GAO H, WANG X, WEI W, et al. Com-DDPG: task offloading based on multiagent reinforcement learning for information-communication-enhanced mobile edge computing in the internet of vehicles[J]. IEEE Transactions on Vehicular Technology, 2023, 73(1): 348-361.

[29] CHEN S G, CHEN J M, MIAO Y F, et al. Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks[J], IEEE Transactions on Signal and Information Processing over Networks, 2022(8): 364-375.

[30] WANG X, HAN Y, LEUNG V C M, et al. Convergence of edge computing and deep learning: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(2): 869-904.

[31] SALEEM U, LIU Y, JANGSHER S, et al. Latency minimization for D2D-enabled partial computation offloading in mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2020, 69(4): 4472-4486.

[32] WU Y, QIAN L P, NI K, et al. Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading[J]. IEEE Journal of Selected Topics in Signal Processing, 2019, 13(3): 392-407.

[33] CHEN X, ZHANG J S, LIN B, et al. Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33(3): 683-697.

[34] 高文轩, 杨新杰. 一种针对能耗优化的车联网计算卸载方案[J]. 电信科学, 2023, 39(10): 29-40.
GAO W X, YANG X J. A computation offloading scheme for energy consumption optimization in Internet of vehicles[J]. Telecommunications Science, 2023, 39(10): 29-40.

[35] CONG Y L, XUE K, WANG C, et al. Latency-energy joint optimization for task offloading and resource allocation in mec-assisted vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2023, 72(12): 16369-16381.

[36] ALAMEDDINE H A, SHARAFEDDINE S, SEBBAH S, et al. Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(3): 668-682.

[37] 张文柱, 余静华. 移动边缘计算中基于云边端协同的任务卸载策略[J]. 计算机研究与发展, 2023, 60(2): 371-385.
ZHANG W Z, YU J H. Taskoffloading strategy in mobile edge computing based on cloud-edge-end cooperation [J]. Journal of Computer Research and Development, 2023, 60(2): 371-385.

[38] OZA P, HUDSON N, CHANTEM T, et al. Deadline-aware task offloading for vehicular edge computing networks using traffic light data[J]. ACM Transactions on Embedded Computing Systems, 2024, 23(1): 1-25.

[作者简介]

谢满德（1977– ），男，博士，浙江工商大学信息与电子工程学院（萨塞克斯人工智能学院）副院长、博士生导师，主要研究方向为边缘计算、云计算、无线传感器网络和网络安全。

黄竹芳（1999– ），女，浙江工商大学信息与电子工程学院（萨塞克斯人工智能学院）硕士生，主要研究方向为边缘计算。

孙浩（1993– ），男，博士，浙江工商大学信息与电子工程学院（萨塞克斯人工智能学院）讲师，主要研究方向为边缘计算、边缘智能。