

文章编号: 1006-3080(2025)02-0250-10

DOI: 10.14135/j.cnki.1006-3080.20240514002

云边场景下基于合作博弈的数据上传优化

钟传江¹, 虞慧群^{1,2}, 范贵生^{1,2}

(1. 华东理工大学计算机科学与工程系, 上海 200237; 2. 上海智慧能源工程技术研究中心, 上海 201103)

摘要: 现有基于边缘计算的移动群智感知场景很少考虑边缘服务器之间的协作, 而多边缘服务器协作面临边缘服务缓存、边缘设备资源约束和大规模边缘用户任务卸载的挑战。针对以上挑战, 本文提出了一种面向多边缘服务器合作博弈和离散粒子群(MCG+DPSO)优化的计算卸载算法。该算法首先通过多边缘服务器之间的合作博弈(MCG)得到边缘用户任务在边缘服务器之间的中继初始方案。然后, 将得到的初始方案作为离散粒子群算法(DPSO)的初始解。最后, 通过 DPSO 算法得到最优解, 实现用户任务和边缘服务器的匹配, 从而最大化地降低移动感知平台数据上传的服务延迟和服务成本。通过在真实数据集上进行大量对比实验, 结果表明与云策略、边缘间不通讯策略、随机策略、合作博弈策略、DPSO 算法和差分进化算法相比, MCG+DPSO 算法可以降低 3.2%~56.0% 的服务成本和 3.6%~24.5% 的服务延迟。

关键词: 边缘计算; 合作博弈; 移动群智感知; 离散粒子群优化; 任务卸载

中图分类号: TP393

文献标志码: A

移动众包(MCS)是一种被广泛关注的传感范式^[1], 它以相对低成本的方式利用了大量移动资源, 用于执行大规模感知任务。传统移动众包平台采用集中式架构, 依赖云服务器来分发任务和收集结果。但是, 随着接入平台的用户数量不断增长, 导致集中式移动众包感知平台在计算、存储和传输方面的性能不断降低。对于大规模数据感知任务, 需要将大量的任务数据上传到云服务器, 这会消耗大量的带宽资源, 增加网络负载^[2]。而对于需要即时反馈的任务, 因为数据需要上传到云服务器并等待处理, 这会导致过高的延迟, 从而使传统的移动众包平台可能无法满足要求^[3]。此外, 边缘设备之间存在负载均衡的情况, 其中一些边缘设备可能承载过多任务而超载, 而其他一些边缘设备则可能因为缺乏任务而使边缘服务器资源处于空闲状态。通过协作, 边缘设备可以共同应对任务激增和资源约束等挑战, 更好地满足不断增长的用户需求^[4]。

近年来, 虽然已经有大量学者做了深入的研究

工作, 然而在联合考虑以下场景时还有待深入研究:

(1) 边缘服务器协作。Li 等^[5]同时考虑了数据感知、数据处理和数据传输, 提出了一个单边缘服务器场景下解决多维资源约束的 MCS 框架, 但是当边缘用户数量增加时, 单个边缘服务器由于计算和存储等资源限制难以满足所有边缘用户的服务要求, 很可能出现延迟增加和任务执行失败的情况。Chen 等^[6]提出了一个包括边缘层的 3 层 MCS 架构, 并采用两阶段博弈最大限度地提高社会福利, 但边缘服务器之间不协作导致存在服务不均衡的问题。Luan 等^[2]在云服务器和移动用户之间部署一组移动边缘服务器, 通过解决边缘服务器的选址问题来最大化地降低边缘服务器的设施成本和边缘用户的服务成本。但是没有考虑边缘服务器相互协作的场景, 导致携带相同类型的不同用户只能移动到特定边缘服务器才能上传数据。

(2) 边缘服务器缓存问题。部分工作假设所有服务都提前在边缘服务器上进行了部署, 也就是假

收稿日期: 2024-05-14

基金项目: 国家自然科学基金(62372174, 62276097)

作者简介: 钟传江(1996—), 男, 硕士生, 研究方向为云计算。E-mail: 2694767623@qq.com

通信联系人: 虞慧群, E-mail: yhq@ecust.edu.cn

引用本文: 钟传江, 虞慧群, 范贵生. 云边场景下基于合作博弈的数据上传优化[J]. 华东理工大学学报(自然科学版), 2025, 51(2): 250-259.

Citation: ZHONG Chuanjiang, YU Huiqun, FAN Guisheng. Optimization of Data Upload Based on Cooperative Game Theory in Cloud Edge Scenarios[J]. Journal of East China University of Science and Technology, 2025, 51(2): 250-259.

设边缘服务器拥有无限的存储能力。但实际上, 边缘服务器只能响应提前缓存的服务, 如果要使用没有缓存的服务, 需要从云服务器上下载, 这会造成大量的延迟。Fu 等^[7]考虑边缘场景中边缘用户可以把自己的任务卸载到不同的边缘服务器, 减少了被传输到云服务器的任务数量, 降低了移动群智感知的服务成本和整体延迟, 但是潜在假设了边缘服务器提前缓存了所有任务处理服务。桑永宣等^[8]研究了边缘计算场景中具有缓存机制的任务卸载优化问题, 考虑了边缘服务器的资源限制情况, 从而只能缓存部分任务处理服务到边缘服务器。

与以上工作不同的是, 本文同时考虑了多边缘服务器之间协作和缓存约束对实现数据高效上传的影响。针对多边缘设备协作中数据上传的延迟和成本优化问题, 提出了一种基于多人合作博弈论方法和离散粒子群优化的计算卸载算法, 有效降低了服务延迟和服务成本。

1 问题形式化

1.1 基于边缘计算的感知框架

本文主要研究基于边缘计算环境下的数据上传, 边缘计算环境由移动设备层、边缘服务器层和云服务器层组成^[9]。在移动设备层, 由数据感知工人提交一系列任务数据, 每一个任务数据对应于一个服务类型。当边缘服务器节点接收到工人发出的数据后, 只有服务器本地已经缓存了对应的服务才能处理该请求, 即, 运行数据处理服务需要预先缓存相应的服务程序^[10]。图 1 示出了基于边缘计算的感知框架。如图 1 所示, 云服务器可以执行所有类型的数据

处理任务, 而边缘服务器由于缓存资源的限制, 只能提前缓存部分类型的任务。工作人员 1 被分配了两种类型的数据采集任务, 在任务完成后, 将数据上传到边缘服务器 A, 但是边缘服务器 A 无法执行工作人员 1 的所有任务, 因此边缘服务器 A 可以把任务中继到附近的边缘节点或者云服务器。

1.2 任务与资源模型

在基于边缘计算的 MCS 数据收集场景中, 假定移动边缘计算网络中有 e 个边缘服务器, 为 w 个工人 (即终端用户) 提供服务。工人 i 表示第 i 个终端用户, 其中 $i \in W, W = \{1, 2, \dots, i, \dots, w\}$ 。移动感知平台把一组任务 t 分配给工人, 任务 $j \in T, T = \{1, 2, \dots, j, \dots, t\}$ 。边缘服务器 $k \in E, E = \{1, 2, \dots, k, \dots, e\}$ 。每个边缘设备配置了一个服务器, 边缘服务器 k 可提供的最大计算量表示为 $f_{\max}^k$ 。该场景可以提供 s 种数据处理服务。服务类型 $l \in S, S = \{1, 2, \dots, l, \dots, s\}$ 。 t 个任务在 w 个工人中的分配情况用矩阵 $\mathbf{WT}$ 表示, $\mathbf{WT} = (\mathbf{WT}_{ij})_{w \times t}$ 。如果 $\mathbf{WT}_{ij} = 1$, 表示第 j 个任务分配给了第 i 个工人; 相反, 如果 $\mathbf{WT}_{ij} = 0$, 表示第 j 个任务没有分配给第 i 个工人。一个工人可以接受多个任务, 但一个任务只能分配给一个工人。 e 个边缘服务器缓存的服务类型用矩阵 $\mathbf{ES}$ 表示, $\mathbf{ES} = (\mathbf{ES}_{kl})_{e \times s}$ 。如果 $\mathbf{ES}_{kl} = 1$, 表示第 l 种数据处理任务提前缓存在第 k 个边缘服务器中; 相反, 如果 $\mathbf{ES}_{kl} = 0$, 表示第 l 种数据处理任务没有提前缓存在第 k 个边缘服务器中。每个任务 $\mathbf{WT}_{ij} = \{d_{ij}, r_{ij}, s_{ij}\}$, 其中 d_{ij} 表示任务的数据大小, r_{ij} 表示任务的计算需求, s_{ij} 表示任务的类别。变量 $a_{ky, \mathbf{WT}_{ij}} \in \{0, 1\}$, 其中 $y \in \{0, 1, 2, \dots, y, \dots, e\}$ 表示云服务器和边缘服务器的编号, 云服务器的编号为 0。 $a_{ky, \mathbf{WT}_{ij}}$ 用于记录任务

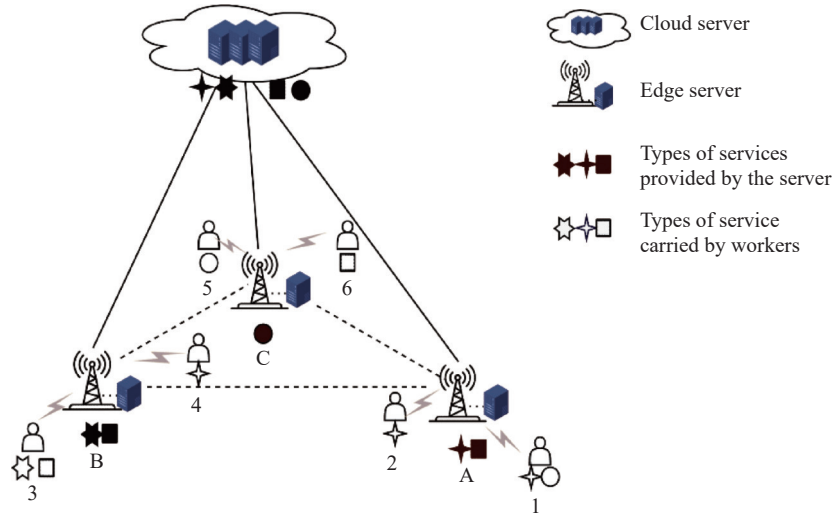


图 1 基于边缘计算的感知框架

Fig. 1 Crowdsourcing sensing framework based on edge computing

WT_{ij} 上传到关联服务器 k 后, 最终在云服务器还是边缘服务器被处理。

当任务 WT_{ij} 在关联服务器执行时, 任务的服务延迟 EL_{ij}^{re} 包括任务从工人上传到关联边缘服务器的时延和关联服务器处理任务的处理时延。

$$EL_{ij}^{re} = \frac{d_{ij}}{v_{ik}} + \frac{r_{ij}}{f_{ij}^k} \quad (1)$$

其中, f_{ij}^k 表示边缘服务器 k 分配给任务 WT_{ij} 的计算资源。 v_{ik} 表示工人 i 和关联边缘服务器 k 之间的上行传输速率, 根据香农公式定义如下:

$$v_{ik} = \omega \log_2 \left(1 + \frac{p \times \lambda_{ik}}{p^2} \right) \quad (2)$$

其中, ω 是工人 i 和关联服务器 k 之间的传输带宽, p 是工人 i 的设备传输功率, p^2 是环境的噪声功率, λ_{ik} 是工人 i 和关联服务器之间的信道增益。

在关联边缘服务器上执行时的服务成本 EC_{ij} 主要包括两个部分: (1) 用户 i 传输数据到关联服务器的传输成本; (2) 关联边缘服务器处理用户请求的处理成本。

$$EC_{ij}^{re} = \beta_1 \times d_{ij} + \beta_2 \times r_{ij} \quad (3)$$

其中, β_1 表示用户 i 传输数据到关联边缘服务器的单位传输成本, β_2 表示边缘服务器处理任务的单位处理成本。

当任务 WT_{ij} 被中继到协作服务器 y 执行时, 任务的服务延迟 EL_{ij}^{coy} 包括任务从工人上传到关联边缘服务器的时延, 关联服务器到协作服务器的传输时延以及协作服务器处理任务的处理时延。

$$EL_{ij}^{coy} = \frac{d_{ij}}{v_{ij}} + \frac{d_{ij}}{v_{ky}} + \frac{r_{ij}}{f_{ij}^y} \quad (4)$$

多边缘服务器之间通过本地局域网连接^[11]。 v_{ky} 表示边缘服务器 k 与边缘服务器 y 之间的传输速率。

本文忽略边缘服务器之间的传输成本, 即不考虑边缘服务器之间因为传输产生的费用, 则任务被中继到协作服务器的服务成本为 EC_{ij}^{coy} 。

$$EL_{ij}^{coy} = \beta_1 \times d_{ij} + \beta_2 \times r_{ij} \quad (5)$$

当工人上传的任务不能被关联边缘服务器和协作边缘服务器提供服务时, 任务会被关联服务器直接中继到云服务器执行。该过程由 3 个部分组成: (1) 工人上传任务到边缘服务器; (2) 边缘服务器将任务发送到云平台; (3) 云平台处理任务。因此, 任务在云平台处理时的延迟为 CL_{ij}^{cl} 。

$$CL_{ij}^{cl} = \frac{d_{ij}}{v_{ij}} + tc_{ij} + \frac{r_{ij}}{f_{ij}^{cl}} \quad (6)$$

其中, tc_{ij} 表示把任务 WT_{ij} 从边缘服务器传输到云

服务的传输延迟, 在本文中被假定为定值; f_{ij}^{cl} 表示云服务器为每个任务分配的计算资源。

当任务无法在边缘服务器处理而被中继到云服务器处理时, 此时的服务成本表示为 CC_{ij}^{cl} 。

$$CC_{ij}^{cl} = \beta_1 \times d_{ij} + \beta_3 \times d_{ij} + \beta_4 \times r_{ij} \quad (7)$$

其中, β_3 表示任务从边缘服务器上传到云服务器的单位传输成本, β_4 表示云服务的单位处理成本。

1.3 问题定义

每个任务 WT_{ij} 有 3 种被处理的情况: 在关联边缘服务器被处理、在协作边缘服务器被处理或者在云服务器被处理。因此任务的服务成本 SC_{ij} 和服务延迟 SL_{ij} 分别表示为:

$$SC_{ij} = ES_{k,S_{ij}} a_{kk,WT_{ij}} EC_{ij}^{re} + \sum_{y \in E \setminus \{k\}} ES_{y,S_{ij}} a_{ky,WT_{ij}} EC_{ij}^{coy} + a_{k0,WT_{ij}} CC_{ij}^{cl} \quad (8)$$

$$SL_{ij} = ES_{k,S_{ij}} a_{kk,WT_{ij}} EL_{ij}^{re} + \sum_{y \in E \setminus \{k\}} ES_{y,S_{ij}} a_{ky,WT_{ij}} EL_{ij}^{coy} + a_{k0,WT_{ij}} CL_{ij}^{cl} \quad (9)$$

在考虑服务延迟和服务成本时, 定义任务 WT_{ij} 在关联边缘服务器、合作边缘服务器和云服务器 3 种处理决策下的增益函数, 以衡量在云服务器处理时相对改进情况。在文献 [12] 中, 请求的结束时间相比于本地处理的改进用 $(t_i' - t_i^m)/t_i'$ 来表示。同样地, 将服务成本与任务在云服务器上处理的服务成本 CC_{ij}^{cl} 进行比较, 服务成本比云上处理的成本小得越多, 获得的增益也就越高; 将服务延迟与任务在云服务器上处理的服务延迟 CL_{ij}^{cl} 比较, 服务延迟比云上处理的延迟小得越多, 获得的增益也就越高。

$$G_{ij} = \delta_1 \frac{CC_{ij}^{cl} - SC_{ij}}{CC_{ij}^{cl}} + \delta_2 \frac{CL_{ij}^{cl} - SL_{ij}}{CL_{ij}^{cl}} \quad (10)$$

其中, δ_1 、 δ_2 分别表示用户对服务成本和服务延迟影响的偏好程度; $\delta_1 + \delta_2 = 1$, 表示服务成本和服务延迟在优化问题中的权重。

本文主要研究如何使收集的数据以最优的方式上传到服务器并被处理。主要的方法是最小化处理所有任务的服务成本和服务延迟。该优化问题可以通过以下公式描述, 即如何获得整体最大的增益。

$$\max \sum_{y \in E} \sum_{WT_{ij} \in B_y} G_{ij} \quad (11)$$

$$\text{s.t.} \sum_{y \in E \cup \{0\}} a_{ky,WT_{ij}} = 1, \forall k \in E, WT_{ij} \in B_y \quad (12)$$

$$\sum_{WT_{ij} \in B_y} f_{ij}^y \leq f_y^{\max}, y \in E \quad (13)$$

$$a_{ky,WT_{ij}} \in \{0, 1\}, \forall k \in E, y \in E \cup \{0\}, WT_{ij} \in B_y \quad (14)$$

$$\delta_1 + \delta_2 = 1 \quad (15)$$

其中, B_y 表示服务器 y 接收到的任务集合。约束 (12) 表示每个任务只能被关联边缘服务器、协作边缘服务器或云服务器中的一个服务器处理。约束 (13) 表示边缘服务器分配给任务的资源总和不能超过其自身的最大资源限制。约束 (14) 表示边缘服务器 k 关联的用户 i 上传的任务 WT_{ij} 是否在边缘服务器 y 或者云服务器处理。约束 (15) 表示权重, 反映的是服务成本和服务延迟在优化目标函数中的权重。

2 求解算法

2.1 多边缘服务器合作博弈

由于关联的边缘服务器缓存的任务类型和硬件资源限制, 无法处理所有接收的数据处理请求, 从而需要把任务中继到协作边缘服务器或者云服务器。本文使用合作博弈来求解效用最大的初始中继方案。其中每个任务对不同的边缘服务器具有不同的效用。如图 1 所示, 边缘服务器 A 对于工人 2 上传的这一类任务具有最高的偏好和较高效用, 因为这一类任务不用被中继到协作边缘服务器或云服务器, 可以直接被关联边缘服务器 A 执行。如果边缘服务器 C 把工人 5 的任务中继到边缘服务器 A, 边缘服务器 A 对此类任务具有最低偏好和较低效用, 因为边缘服务器 A 无法执行此类任务, 需要把此类任务继续中继到云服务器。在合作博弈中, 每一个边缘服务器就是一个玩家, 表示为 $\{1, 2, \dots, y, \dots, e\}$, 边缘用户上传的任务请求就是玩家互相博弈的商品, 每一个玩家的目的是让自己的增益最大。定义变量 $b_{jy} \in \{0, 1\}$, 记录边缘服务器接收到的任务信息, 初始情况由边缘服务器与边缘用户的位置决定。

2.1.1 任务对边缘服务器的效用 在多边缘服务器合作博弈中, 依据任务的输入数据大小、任务的计算需求量、任务的类型和边缘服务器的缓存类型来定义一个任务对某个边缘服务器的效用。边缘服务器更愿意接收效用更高的任务。根据以上分析, 本文把任务 WT_{ij} 对边缘服务器 y 的效用 u_j^y 定义如下:

$$u_j^y = (\tau \times d_{ij} + a_{ky, S_{ij}} \times r_{ij}) \times P_{ij}^y \quad (16)$$

其中, τ 是常数, 用于使任务的计算需求量和数据大小在同一个量级; P_{ij}^y 表示边缘服务器 y 对任务 WT_{ij} 的偏好。

边缘服务器 y 的效用总和 U_y 等于在边缘服务器 y 上处理的任务效用和边缘服务器 y 中继到云服务器上的任务效用总和。

$$U_y = \sum_{j=1}^I b_{jy} \times u_j^y \quad (17)$$

所以, 边缘服务器 y 的增益变化被定义为 $F_y = U_y - U_y^0$, 其中 U_y^0 是指边缘服务器未参加合作博弈时的效用。合作博弈的目标是找到一个策略 S^* , 它可以实现最大的纳什乘积, 公式如下:

$$S^* = (s_1^*, s_2^*, \dots, s_e^*) \leftarrow \operatorname{argmax} \prod_{y=1}^e (U_y - U_y^0) = \operatorname{argmax} \prod_{y=1}^e \left(\sum_{j=1}^I b_{jy} \times u_j^y - U_y^0 \right) \quad (18)$$

确定多玩家合作博弈的纳什均衡解是一个复杂的问题, 没有一个固定的解决方案。在本文中使用文献 [13] 中提出的一种基于空间距离的几何近似方法来求解多边缘服务器合作博弈的纳什均衡解。

2.1.2 空间距离 任务 WT_{ij} 对于边缘服务器 y 的空间距离被定义为 u_j^y 的倒数, 并进行 $[0, 1]$ 上的归一化, 即对边缘服务器效用越高的任务对边缘服务器的空间距离越短, 表达式如下:

$$sd_j^y = \frac{1}{u_j^y} / \sum_{y=1}^e \frac{1}{u_j^y} \quad (19)$$

2.1.3 效用空间距离乘积 效用空间距离乘积 $usdp_j^y$ 即任务 WT_{ij} 对边缘服务 y 的效用 u_j^y 乘以空间距离 sd_j^y 。

在文献中作者采用类似于杠杆原理来获得不同向量乘积之和的平衡点, 效用空间距离乘积由以下公式确定:

$$usdp_j^y = 1 / \sum_{y=1}^e \frac{1}{u_j^y} \quad (20)$$

这相当于一个力矩。确定的平衡点可以使效用距离乘积向量均匀地围绕平衡点分布。该向量由以下公式给出:

$$BP_y = \frac{1}{e} \times \sum_{j=1}^I usdp_j^y \quad (21)$$

2.1.4 任务优先级排序 对于 T 个被分配的任务 WT_{ij} , 基于与边缘服务器 y 的归一化距离 sd_j^y 进行排序。边缘服务器 y 排序好的距离效用乘积序列表示为 Φ_y 。边缘服务器需要最大化自己的增益并保持距离效用乘积的平衡。确定纳什均衡策略的方法是对排好序的向量 Φ_y 从小到大进行累加, 直到和大于或等于达到平衡点时 y 的分量和 BP_y 。表示如下:

$$\sigma_y = \left\{ x \mid \sum_{y=1}^x \Phi_{y,y} \leq BP_y \right\} \quad (22)$$

任务集合 σ_y 即为最终需要被边缘服务器 y 处理或者上传到云服务器的任务集。最后根据合作博弈得到的策略 σ_y 和约束(12)~(15)确定最终的任务处理决策。

图 2 示出了两玩家博弈确定的空间距离乘积与平衡点,即两个边缘服务器玩家博弈 6 个任务时的情景,其中任务 2、5、4 对边缘服务器 1 的效用与距离乘积的和,约等于任务 1、3、6 对边服务器 2 的效用与距离乘积的和,从而使分配达到平衡。平衡点左侧的任务 2、5、4 被分配给 1 号边缘服务器;平衡点右侧的任务 1、3、6 被分配给 2 号边缘服务器。

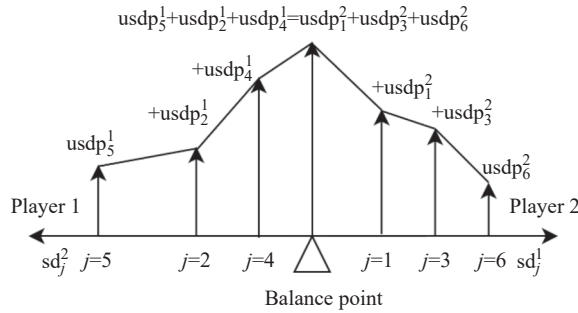


图 2 两玩家博弈确定空间距离乘积与平衡点

Fig. 2 Determining balance points and the product of spatial distances in a two-player game

算法 1 描述了基于合作博弈的任务处理决策的求解过程。

算法 1: MCG

输入: 任务总数 t 、边缘服务器总数 e

输出: 任务决策向量 T_{MCG}

1. for $y=1; y \leq e; y++$ do
2. for $j=1; j \leq t; j++$ do
3. 计算效用 u_j^y , 根据等式(16);
4. end
5. end
6. for $y=1; y \leq e; y++$ do
7. for $j=1; j \leq t; j++$ do
8. 计算空间距离 sd_j^y , 根据等式(19);
9. 计算效用空间距离乘积 $usdp_j^y$, 根据等式(20);
10. end
11. end
12. for $y=1; y \leq e; y++$ do
13. 计算达到平衡的点时边缘服务器 y 的分量 BP_y , 根据等式(21);
14. 根据任务对边缘服务器的空间距离大小进行升序排序;
15. 根据任务的升序序列更新任务的距离效用序列;
16. 计算边缘服务器 y 最优策略 σ_y , 根据等式(22);

17. end

18. 根据 $\{\sigma_1, \sigma_2, \dots, \sigma_e\}$ 更新 T_{MCG} ;

19. return T_{MCG}

2.2 离散粒子群算法

2.2.1 离散粒子群编码与解码方式 DPSO 算法采用整数编码方式, 编码内容记录的是任务最终被中继到的边缘服务器或者云服务器的信息。粒子的维度数量是所有的任务数量, 每个维度对应一个任务。首先对云服务器和边缘服务器进行编号, 云服务器编号为 0, e 个边缘服务器依次编号为 $\{1, 2, \dots, e\}$, 整个系统中云服务器和边缘服务器缓存的 s 个任务类型, 依次被编号为 $\{1, 2, \dots, s\}$ 。由于某一个任务并不是所有边缘服务器都能提供服务, 所以需要对特定的任务类型进行重新索引。如图 3 显示了 1 个云服务器和 3 个边缘服务器缓存了 5 种服务类型时的编码与解码情况。图片最上方编码的是 6 个任务, 编号依次为 1~6, 对应的任务类型分别是类型 2、5、1、2、4、1。粒子携带的信息表示每个任务需要被中继到云服务器还是边缘服务器。以粒子左边第一维和第二维的信息为例, 它分别表示任务编号为 1、类型为 2 的任务需要被中继到云服务器; 任务编码为 2、任务类型为 5 的任务需要被中继到 3 号边缘服务器执行。给定一个粒子的位置, 通过图 3 编码和解码方案可以将任务和服务器之间的对应关系的映射求解出来。然后, 服务器根据自己的资源限制决定如何处理分配的任务。

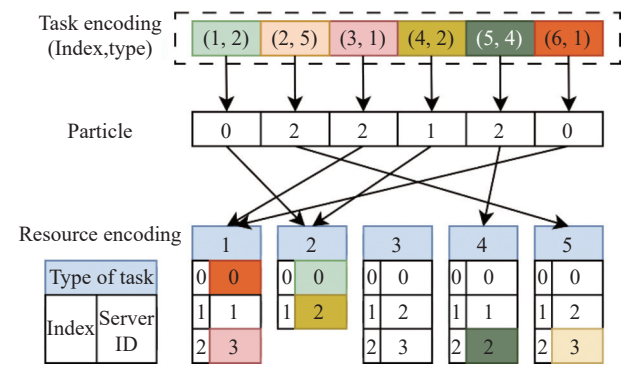


图 3 粒子编码与解码

Fig. 3 Particle encoding and decoding

算法 2: DPSO 算法 2 总结了基于 DPSO 的带有缓存机制的卸载策略的过程, 算法中 T_{MCG}' 是对 T_{MCG} 进行了编码, $F(*)$ 表示的是公式(11)中的优化目标函数。输入: T_{MCG}' , 迭代次数 P , 粒子数 N , 适应度函数 $F(*)$

输出: P^*

1. $gbest^0 = T_{MCG}'$;
2. 随机初始化粒子的位置 x_n^0 和速度 v_n^0 ;

```

3. 初始化  $\text{pbest}_n^0$ ;
4. for  $p=1; p \leq P; p++$  do
5.   for  $n=1; n \leq N; n++$  do
6.     for  $m=1; m \leq M; m++$  do
7.       do
8.         更新  $v_{n,m}^p$ , 根据等式(23);
9.         更新  $x_{n,m}^p$ , 根据等式(24);
10.        while  $x_{n,m}^p$  超出索引范围;
11.      end
12.      if  $F(x_n^p) > F(\text{pbest}_n)$  then
13.         $\text{pbest}_n = x_n^p$ ;
14.      end
15.      if  $F(\text{pbest}_n) > F(\text{gbest})$  then
16.         $\text{gbest} = \text{pbest}_n$ ;
17.      end
18.    end
19.  end
20.  $P^* = \text{gbest}$ ;
21. return  $P^*$ 

```

2.2.2 粒子迭代 DPSO 以一组随机粒子和 MCG 算法的输出进行初始化, 其中一个随机粒子代表一个随机解。然后, 进行 P 次迭代求解最优解。在这个过程中, 用 N 来表示粒子的总数。在第 p 次迭代中, 粒子 n 通过寻找两个极值, 即个体最佳位置 pbest_n 和群体最佳位置 gbest , 来更新其速度 v_n^p 和位置 x_n^p 。这里, pbest_n 是粒子在迭代中所探索到的最佳位置, 而 gbest 则是全局范围内的所有粒子找到的最佳位置。依据 pbest_n 和 gbest , 粒子每个维度根据如下公

式更新自身的速度 $v_{n,m}^p$ 。

$$v_{n,m}^p = \omega v_{n,m}^{p-1} + c_1 \text{rand}(0, 1) (\text{pbest}_{n,m} - x_{n,m}^{p-1}) + c_2 \text{rand}(0, 1) (\text{gbest}_m - x_{n,m}^{p-1}) \quad (23)$$

其中, ω 表示惯性常数。加速因子 c_1 和 c_2 分别代表对个体经验 pbest_n 和群体经验 gbest 的权衡。本文设 $c_1 = 2$, $c_2 = 2$, 以在实际应用中取得平衡的效果。因为粒子每个维度都是一个整数, 所以速度也是一个整数。粒子的维度值通过以下公式更新, 根据任务类型判断 $v_{n,m}^p$ 是否超出索引最小值和最大值, 若超出范围则需重算公式(23)、(24)的值, P 次迭代后, 全局最佳位置 gbest^p 将用作 DPSO 算法求得的最优解。

$$x_{n,m}^p = x_{n,m}^{p-1} + \lfloor v_{n,m}^p \rfloor \quad (24)$$

总之, 采用 DPSO 优化 MCG 算法得到次优解, 以达到服务成本和延迟最小化。

3 模拟实验

3.1 参数设置

本文采用了澳大利亚墨尔本市中央商务区 EUA^[14] 数据集。在每个实验场景中随机抽取 E 个边缘服务器和 W 个边缘用户用于模拟基于边缘计算的数据感知场景。每个用户可以执行多项数据感知任务, 完成任务后直接上传到最近的边缘服务器, 然后由边缘服务器对任务做进一步处理或中继。为了评估 MCG+DPSO 和其他基线的性质, 在 AMD R7、3.2 GHz、32 GB RAM 的笔记本电脑上进行了模拟实验, 确定了表 1 定义的实验参数及数值^[15]。

表 1 实验参数
Table 1 Experimental parameters

Transmission power for end user/W	Transmission latency from the edge device to the cloud/ (s · task ⁻¹)	Bandwidth for edge device/MHz	Calculated requirements of the task/ (GHz · task ⁻¹)	Noise power/(dB · m)	Size of input data for the task/(MB · task ⁻¹)
0.5	5	20	[0.3—0.6]	100	[0.3—0.5]
Communication range of edge devices/m	Computing resources in the remote cloud/ (GHz · task ⁻¹)	Transmission cost from end device to edge device/ (MB · task ⁻¹)	Transmission cost from edge device to cloud/ (MB · task ⁻¹)	Processing cost for edge devices/(GHz · task ⁻¹)	Remote cloud processing costs/(MB · task ⁻¹)
1000	5	0.01	0.02	0.001	0.005

对于每个实验场景, 更改其中一个参数值, 保持其余参数不变, 并进行一组实验以评估该方案的性能, 总共进行了 4 组实验, 如表 2 所示, 其中, #1 更改边缘服务器数量, 其余参数值保持不变; #2 更改边缘

用户数量; #3 更改边缘服务器缓存比例; #4 更改边缘服务器最大计算资源数量。

6 种策略如下: (1) 云策略(Cloud): 在该策略中, 工人把任务上传到关联边缘服务器后, 所有任务都

表 2 实验设置
Table 2 Experimental settings

Scenario	Edge server	Edge user	Cache ratio	Computing resource
#1	5, 10, ..., 25	100	0.2	10
#2	20	20, 40, ..., 100	0.2	10
#3	20	100	0.05, 0.10, ..., 0.20	10
#4	20	100	0.2	2, 4, ..., 14

被中继到云服务器执行。(2)边缘间不通信(ENC):该策略假设边缘服务器之间不进行合作,如果边缘服务器不能处理工人的请求,直接把任务请求中继到云服务器。(3)随机策略(Random):该策略允许为每个任务随机选择一个云服务器或者边缘服务器(缓存了该任务类型)。(4)MCG^[16]:该策略仅通过边缘服务器之间的合作博弈获得任务决策策略。(5)DPSO^[17]:该策略通过离散粒子群优化算法对候选的工作者集合进行优化,进一步提高了以平台为中心的移动感知激励效用。(6)贪婪差分进化算法(DE)^[18]:该策略基于差分进化算法来优化任务卸载

和资源分配。

3.2 实验结果与分析

在 4 组实验场景中,将 MCG+DPSO 与其他 6 种策略进行了比较,性能评估指标包括服务成本、服务延迟。在这个实验中 $\sigma_1=\sigma_2=0.5$, 粒子群和种群数量均等于 50。图 4~图 7 示出了实验结果。

图 4 示出了实验场景#1 的实验结果,检验了不同数量的边缘服务器对 MCS 数据上传服务成本、服务延迟的影响。如图 4 所示,可以看出 MCG+DPSO 算法在这 2 个方面都不同程度地优于其他 6 个比较策略。MCG+DPSO 的性能分别比 Cloud、ENC、Random、MCG、DE 和 DPSO 算法在服务成本上高出 56.0%、46.6%、33.8%、16.0%、13.7% 和 4.5%, 同时服务延迟上分别高出 27.3%、16.7%、11.0%、5.4%、4.2% 和 3.6%。由于云策略所有任务都在云中执行,服务成本和服务延迟保持不变。而其他 6 种策略,随着边缘服务器数量的增加,服务成本和服务延迟整体呈降低趋势。因为随着边缘服务器数量的增加,它可以为边缘数据感知工人提供更多的计算资源,可以缓存更多类型的任务,降低被上传到云服务器的任务

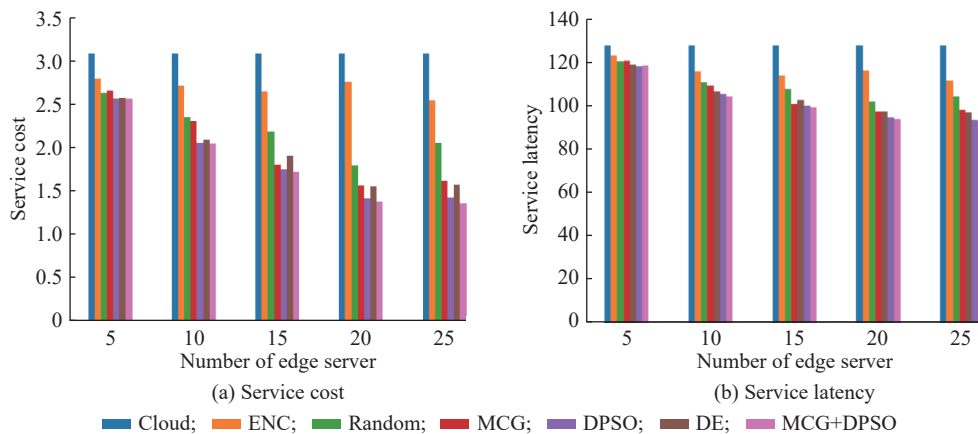


图 4 改变边缘服务器数量

Fig. 4 Adjusting edge server quantity

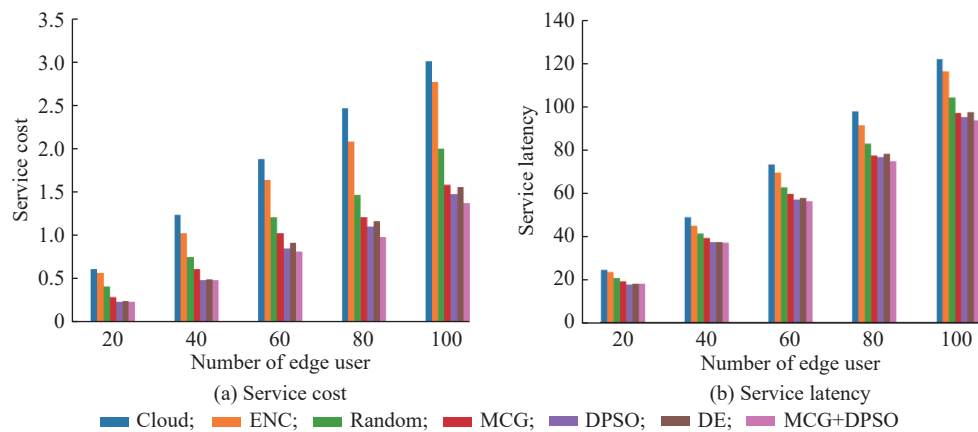


图 5 改变边缘用户数

Fig. 5 Adjusting edge user quantity

数量,从而降低服务成本和服务延迟。从实验结果可以看出,由于云策略只能把任务上传到云服务器执行,而边缘间不通讯策略任务不能在边缘服务器之间互相协作执行,所以这两个策略整体的性能都要劣于其他策略,这也说明在 MCS 架构中引入边缘层且使边缘服务器之间互相协同是很有意义的。

图 5 示出了实验场景#2 的实验结果,验证了不同数量的边缘用户对 MCS 数据上传服务成本、服务延的影响。从图 5 可以看出, MCG+DPSO 相较于其他比较策略表现更好。MCG+DPSO 的性能分别比 Cloud、ENC、Random、MCG、DE 和 DPSO 算法在服务成本上高出 54.6%、50.7%、31.5%、13.2%、11.9% 和 6.7%, 同时在服务延迟上分别高出 23.4%、19.5%、10.2%、4.5%、4.2% 和 3.6%。如图 5(a) 和 5(b) 所示, 所有策略的服务成本和服务延迟均随着边缘用户数的增加而增加。因为当边缘服务器数量一定时,边缘服务器的缓存资源和计算资源是有限的,当边缘用户增加会导致需要上传的任务数据急剧上升,导

致有大量的服务因为没有分配到边缘服务器资源而被中继到远端云服务器执行,从而导致任务的服务成本和服务延迟变差。在 MCG+DPSO 算法中,随着用户数的增加,在服务延迟和服务成本上都优于其他比较策略。云策略是直接云中执行所有任务,因此服务成本和服务延迟最大。其中边缘间不通讯和随机策略性能较差,因为边缘间不通讯策略没有边缘服务器协作的优势,而随机策略则很难找到适合卸载的边缘服务器。

图 6 示出了实验场景#3 的实验结果,验证了边缘服务器的服务缓存比例对 MCS 数据上传服务成本、服务延迟影响。如图 6 所示,本文提出的 MCG+DPSO 策略表现优于比较策略。MCG+DPSO 的性能分别比 Cloud、ENC、Random、MCG、DE 和 DPSO 算法在服务成本上高出 55.6%、50.4%、23.0%、13.0%、11.6% 和 5.1%, 同时在服务延迟上分别高出 23.2%、19.4%、7.8%、4.4%、4.2% 和 3.3%。如图 6(b) 和 6(c) 所示,随着边缘服务缓存率的增加,服务成本和服务

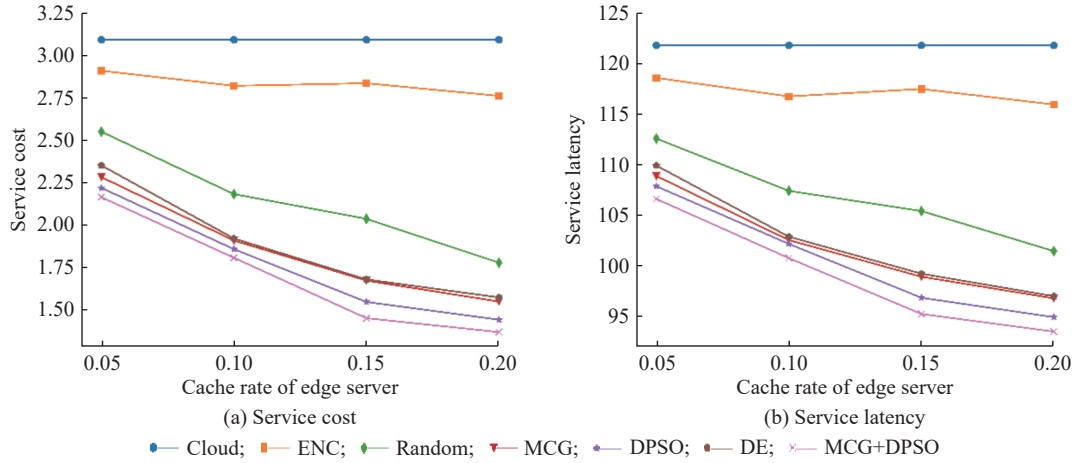


图 6 改变边缘服务缓存率

Fig. 6 Changing edge service cache rate

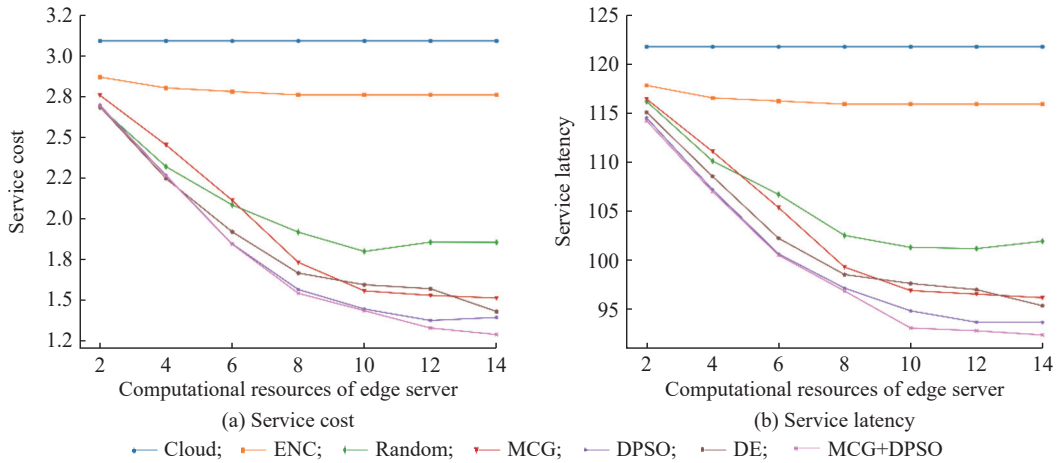


图 7 改变边缘服务器计算资源数

Fig. 7 Changing edge server computational resources count

延迟整体上都有明显下降。因为边缘服务器可以为更多类型的任务提供服务,从而有更多的任务在边缘服务器上执行,减少了被上传到边缘服务器的任务数量,从而降低了总体服务成本和服务延迟。

图 7 示出了实验场景#4 的实验结果,验证了边缘服务器计算资源的变化对 MCS 数据上传服务成本、服务延迟的影响。如图 7 所示, MCG+DPSO 策略与其他 6 种策略相比较表现出了一定的优势。MCG+DPSO 的性能分别比 Cloud、ENC、Random、MCG、DE 和 DPSO 算法在服务成本上高出 58.2%、53.2%、30.4%、14.7%、9.8% 和 7.5%, 同时在服务延迟上分别高出 24.1%、20.2%、9.4%、5.6%、5.5% 和 4.1%。如图 7(a) 和 7(b) 所示, 在图中的 7 种策略中, 除云策略以外, 随着边缘服务器计算资源的增加, 服务成本和服务延迟呈下降趋势。因为增加边缘服务器的计算资源使更多任务可以在边缘服务器中执行, 从而降低数据收集任务的服务成本和服务延迟。

总之, 云策略由于所有任务都被中继到远端云服务器, 从而导致较高的服务延迟和服务成本。云策略由于没有引入边缘服务器层显然不具有优势。边缘间不通讯策略由于边缘服务器之间不能相互合作, 导致存在严重的负载不均衡问题。并且因为单个边缘服务器缓存的服务类型有限, 导致大量的边缘任务被直接中继到了远端云服务器, 从而导致了较高的服务延迟和服务成本。单纯的 DPSO 和差分进化算法都是在随机策略的基础上生成初始粒子和种群, 然后进行进一步的优化, 因此 DPSO 和差分进化算法在各项指标上均优于随机策略。而 MCG+DPSO 算法先采用合作博弈获得次优解, 然后采用离散粒子群算法做进一步优化, 因此相对 MCG 和 DPSO 的性能都有一定程度的提高。

4 结束语

本文研究了一个基于边缘计算的 MCS 数据上传场景, 其中重点考虑了边缘服务器相互协作为边缘用户提供卸载服务, 从而降低数据上传的服务成本和服务延迟; 同时提出了基于多边缘服务器合作博弈和离散粒子群优化(MCG+DPSO)算法, 仿真模拟实验表明, 该方法优于其他基线方法。

本文只考虑了边缘计算环境下数据上传的服务成本和服务延迟的相关优化目标。在该研究领域中, 预算成本约束、安全与隐私性、资源利用率等都是重要的考虑因素, 在接下来的工作中, 将结合这些指标考虑更全面的场景去探索他们的影响和相互制

约关系, 提出能够平衡各类指标并且性能优秀的优化算法。

参考文献:

- [1] CAPPONI A, FIANDRINO C, KANTARCI B, *et al.* A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities[J]. *IEEE Communications Surveys & Tutorials*, 2019, 21(3): 2419-2465.
- [2] LUAN D M, WANG E, LIU W B, *et al.* Minimum-cost edge-server location strategy in mobile crowdsensing[J]. *IEEE Transactions on Network and Service Management*, 2021, 18(3): 3375-3388.
- [3] LIU L N, WANG L H, LU Z M, *et al.* Cost-and-quality aware data collection for edge-assisted vehicular crowdsensing[J]. *IEEE Transactions on Vehicular Technology*, 2022, 71(5): 5371-5386.
- [4] LIN R P, XIE T Z, LUO S, *et al.* Energy-efficient computation offloading in collaborative edge computing[J]. *IEEE Internet of Things Journal*, 2022, 9(21): 21305-21322.
- [5] LI X Q, FENG G, LIU Y J, *et al.* Joint sensing, communication, and computation in mobile crowdsensing enabled edge networks[J]. *IEEE Transactions on Wireless Communications*, 2022, 22(4): 2818-2832.
- [6] CHEN X, TANG C, LI Z, *et al.* A pricing approach toward incentive mechanisms for participant mobile crowdsensing in edge computing[J]. *Mobile Networks and Applications*, 2020, 25: 1220-1232.
- [7] FU Y M, ZHANG X, QIN X Q, *et al.* Data collection of multi-player cooperative game based on edge computing in mobile crowd sensing[J]. *Computer Networks*, 2023, 222: 109551.
- [8] 桑永宣, 魏江坡, 王博, 等. 具有边缘缓存机制的混合启发式任务卸载算法 [J]. *计算机工程*, 2023, 49(4): 149-158.
- [9] CHU W B, YU P J, YU Z W, *et al.* Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach[J]. *IEEE Transactions on Mobile Computing*, 2022, 22(7): 4150-4167.
- [10] XUE Z, LIU C, LIAO C L, *et al.* Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems[J]. *IEEE Transactions on Vehicular Technology*, 2023, 72(5): 6709-6722.
- [11] CHEN L X, XU J, ZHOU S. Computation peer offloading in mobile edge computing with energy budgets[C]//*IEEE Global Communications Conference*. Singapore, IEEE, 2017: 1-6.
- [12] LYU X C, TIAN H, SENGUL C, *et al.* Multiuser joint task

- offloading and resource optimization in proximate clouds[J]. *IEEE Transactions on Vehicular Technology*, 2016, 66(4): 3435-3447.
- [13] WONG K K L. A geometrical perspective for the bargaining problem[J]. *Plos One*, 2010, 5(4): e10331.
- [14] CUI G M, HE Q, XIA X Y, *et al.* OL-EUA: Online user allocation for noma-based mobile edge computing[J]. *IEEE Transactions on Mobile Computing*, 2021, 22(4): 2295-2306.
- [15] 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.
- [16] LIU Y, WEI W, XU H Y. Efficient multi-resource scheduling algorithm for hybrid cloud-based large-scale media streaming[J]. *Computers & Electrical Engineering*, 2019, 75: 123-134.
- [17] WANG Y J, GAO Y, LI Y S, *et al.* A worker-selection incentive mechanism for optimizing platform-centric mobile crowdsourcing systems[J]. *Computer Networks*, 2020, 171: 107144.
- [18] NAOURI A, WU H X, NOURI N A, *et al.* A novel framework for mobile-edge computing by optimizing task offloading[J]. *IEEE Internet of Things Journal*, 2021, 8(16): 13065-13076.

Optimization of Data Upload Based on Cooperative Game Theory in Cloud Edge Scenarios

ZHONG Chuanjiang¹, YU Huiqun^{1,2}, FAN Guisheng^{1,2}

(1. Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China; 2. Shanghai Engineer Research Center of Smart Energy, Shanghai 201103, China)

Abstract: In the existing mobile crowd intelligence sensing scenarios based on edge computing, the collaboration among edge servers is little considered. Collaboration among multiple edge servers faces challenges such as edge service caching, edge device resource constraints, and large-scale edge user task offloading. Addressing the above challenges, this paper formulates the optimization problem for service latency and service cost based on collaborative multi-edge server cooperation, and proposes a computational offloading algorithm based on multi edge server cooperative game and discrete particle swarm optimization (MCG+DPSO) optimization (MCG+DPSO). Firstly, an initial relay scheme for edge users' tasks among multiple edge servers is derived through Multi-Edge Server Cooperation Game (MCG). Then, the obtained scheme is taken as an initial solution for the Discrete Particle Swarm Optimization (DPSO) algorithm. Finally, the DPSO algorithm is used to obtain the optimal solution, achieving the matching between user tasks and edge servers, thereby maximizing the reduction of service latency and costs for data uploading on mobile sensing platforms. Through extensive comparative experiments on real datasets, it is shown that, compared with cloud strategy, edge non communication strategy, random strategy, cooperative game strategy, DPSO algorithm, and differential evolution algorithm, the proposed MCG+DPSO algorithm can reduce service costs by up to 3.2% to 56.0% and service latency by 3.6% to 24.5%.

Key words: edge computing; cooperative game theory; mobile crowd sensing; DPSO; task offloading

(责任编辑: 李娟)