文章编号: 1000-5641(2021)06-0088-12

面向移动边缘计算的联合计算卸载和资源分配策略研究

黄冬晴[1], 俞黎阳[1], 陈 珏[2], 魏同权[1]

(1. 华东师范大学 计算机科学与技术学院, 上海 200062;
2. 上海工程技术大学 电子电气工程学院, 上海 201620)

摘要: 随着无人驾驶、在线游戏、虚拟现实等低延迟应用的大量涌现, 传统集中式的移动云计算范式越来越难以满足此类用户服务质量的需求. 为弥补云计算的不足, 移动边缘计算应运而生. 移动边缘计算通过计算卸载, 将计算任务迁移到网络边缘服务器来为用户提供计算和存储资源. 然而, 现有大部分工作仅考虑了延迟或能耗的单目标性能优化, 未考虑延迟和能耗的均衡优化. 为减少任务延迟和设备能耗, 提出了一种面向多用户的联合计算卸载和资源分配策略. 该策略首先利用拉格朗日乘子法获得给定卸载决策的最佳计算资源分配; 然后, 提出一个基于贪心算法的计算卸载算法获得最佳卸载决策; 最后, 通过不断迭代得到最终解. 实验结果表明, 与基准算法相比, 所提算法最高可以降低 40% 的系统成本.

关键词: 移动边缘计算; 计算卸载; 资源分配; 拉格朗日乘子法; 贪心算法
中图分类号: TP391 **文献标志码**: A **DOI**: 10.3969/j.issn.1000-5641.2021.06.010

Research on joint computation offloading and resource allocation strategy for mobile edge computing

HUANG Dongqing[1], YU Liyang[1], CHEN Jue[2], WEI Tongquan[1]

(1. *School of Computer Science and Technology, East China Normal University, Shanghai* 200062, *China*;
2. *School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai* 201620, *China*)

Abstract: With the emergence of low-latency applications such as driverless cars, online gaming, and virtual reality, it is becoming increasingly difficult to meet users' demands for service quality using the traditional centralized mobile cloud computing model. In order to make up for the shortages of cloud computing, mobile edge computing came into being, which provides users with computing and storage resources by migrating computing tasks to network edge servers through computation offloading. However, most of the existing work processes only consider single-objective performance optimization of delay or energy consumption, and do not consider the balanced optimization of delay and energy consumption. Therefore, in order to reduce task delay and equipment energy consumption, a multi-user joint computation offloading and resource allocation strategy is proposed. In this strategy, the Lagrange multiplier method is used to obtain the optimal allocation of computing resources for a given offloading decision. Then, a computation offloading algorithm based on a greedy algorithm is proposed to obtain the optimal offloading decision; the

收稿日期: 2020-06-22
基金项目: 上海市科学技术委员会项目 (19YF1418300)
通信作者: 俞黎阳, 男, 副教授, 硕士生导师, 研究方向为智慧城市、云计算等. E-mail: lyyu@cs.ecnu.edu.cn

final solution is obtained through continuous iteration. Experimental results show that, compared with the benchmark algorithm, the proposed algorithm can reduce system costs by up to 40%.

Keywords: mobile edge computing; computation offloading; resource allocation; Lagrange multiplier method; greedy algorithm

0　引　　言

随着移动互联网和 5G 通信技术[1] 的不断发展, 移动设备上需要处理越来越多的延迟敏感性应用, 例如无人驾驶、在线游戏、虚拟现实和增强现实等[2]. 但是, 由于计算和存储资源的限制, 移动设备运行这些应用会造成很高的延迟和能耗. 为解决该问题, 移动云计算 (Mobile Cloud Computing, MCC) 诞生了, 它将用户部分计算任务通过上行链路卸载到云端服务器, 减少了任务的执行时间. 然而在传统 MCC 中, 集中式部署的云服务器与移动设备距离较远, 两者数据传输过程中会占用大量网络带宽导致网络拥塞, 从而增加了通信延迟和能量消耗. 因此, 传统的 MCC 范式越来越难以满足用户服务质量的需求.

针对 MCC 中存在的问题, 欧洲标准协会提出了移动边缘计算 (Mobile Edge Computing, MEC)[3]. 在 MEC 系统中, 具有计算和存储资源的边缘服务器被部署在更靠近用户的网络边缘, 移动设备便能以更低的通信延迟进行任务卸载. MEC 的优势包括低延迟、可靠的服务交付、高效的网络运营等. 随着移动用户数目的增加, 边缘服务器因其计算资源受限而影响了任务执行延迟和能耗. 因此, 计算卸载和资源分配策略成为实现系统高效卸载的关键.

针对多用户 MEC 系统资源受限的场景, 有些学者关注最大程度减少任务时延: Zhang 等[4] 研究了基于时分多址的 MEC 系统中计算和通信资源分配问题, 设计了一个次梯度算法以最大程度减少设备延迟; Alam 等[5] 提出了一种基于深度 Q 学习的自主计算卸载框架, 用于处理系统中资源分配和移动性问题; Paymard 等[6] 提出了一种高效可感知传输的任务调度和资源分配算法, 联合优化上行/下行子载波、传输功率、计算资源分配及任务调度; Liu 等[7] 考虑任务随机性和计算强度, 设计了一个卸载框架, 并通过分析边缘服务器排队模型提出了基于 Lyapunov 优化的动态资源分配算法. 这些工作通常针对时延敏感型应用, 虽然都在一定程度上实现以极低的延迟进行任务卸载, 但是忽略了系统中设备的能量消耗.

此外, 有学者重点关注进一步节省系统中设备的能量消耗, 以最大程度延长系统设备的使用寿命: Guo 等[8] 考虑到高干扰、多路访问和有限资源, 设计了一种基于遗传算法的次优卸载算法, 以实现卸载决策、无线和计算资源分配的联合优化; Li 等[9] 基于 Lyapunov 优化框架, 提出了一种在线能耗最小化的卸载算法, 来实现最佳能量和数据传输时间分配、任务卸载比例、传输功率以及设备计算频率; Qian 等[10] 通过共同优化资源分配和连续干扰消除排序来减少能耗, 即先基于 Karush-Kuhn-Tucher (KKT) 条件和梯度下降法获得最佳的传输功率和计算资源分配, 然后提出基于禁忌搜索的排序算法; Zeng 等[11] 针对单小区中正交多址和非正交多址场景提出了近似最优的解决方案, 以实现传输功率和计算资源最佳联合分配, 同时也证明了非正交多址优于正交多址方案. 这些工作通常针对能量消耗型应用, 一般都是在满足任务计算时延约束下最大程度降低系统的能耗, 但是以增加计算时延的开销为代价.

以上研究工作都倾向于从延迟或能耗单目标性能进行系统优化, 未实现系统延迟和能耗的均衡优化. 在某些情况下, 本文希望用更少的时间及更低的能耗完成系统所有计算任务, 从而让服务器可以更早地完成计算任务, 为更多的用户提供服务, 同时延长系统寿命. 因此, 本文在考虑 MEC 系统场

景下任务卸载中延迟和能耗两个因素之间的权衡的基础上, 提出了一种面向多用户的联合计算卸载和资源分配策略.

本文主要贡献概述如下.

(1) 考虑多用户单小区 MEC 系统场景, 建立了本地计算模型和移动边缘计算模型. 每个任务计算成本被表述为时延和能耗加权和, 在有限的信道和计算资源约束下, 优化问题被建模为一个混合整数非线性规划 (Mixed Integer Nonlinear Programming, MINILP) 问题.

(2) 提出了一种面向多用户的联合计算卸载和资源分配策略. 针对资源分配子问题, 采用拉格朗日乘子法获得最佳计算资源分配. 针对计算卸载子问题, 提出一种基于贪心算法的计算卸载算法. 最后, 通过不断迭代获得最优卸载决策和计算资源分配.

(3) 通过仿真实验, 验证了本文所提算法的有效性. 与本地执行算法、全卸载算法、分支定界算法以及带汉明距离终止的动态规划卸载算法进行比较, 本文所提算法优于其他算法, 且系统成本最高可以降低 40%.

1 系统架构和计算模型

在本章中, 首先对系统架构进行详细介绍, 然后给出任务计算模型, 包括本地计算和移动边缘计算.

1.1 系统架构

如图 1 所示, 考虑 1 个多用户单小区 MEC 系统, 有 N 个移动设备 (Mobile Device, MD), 每个移动设备都被视为用户, 用户通过无线信道的方式连接附近基站 (Base Station, BS). 假设在某一时刻每个用户仅产生 1 个计算任务, 对于第 $i(i=1,2,\cdots,N)$ 个移动设备, 定义 $R_i \triangleq \{B_i, C_i\}$ 表示其计算任务要求, 其中 B_i 表示计算任务的数据大小, 即移动设备传输到边缘服务器的数据量, 包括系统设置、程序代码和输入参数; C_i 表示任务所需的计算资源量, 即完成计算任务需要的 CPU 周期数. 电信运营商在基站附近部署有 MEC 服务器, 其具有一定的存储容量和计算能力, 可以存储用户的输入任务并提供计算服务. 表 1 总结了本文所使用的关键符号及其含义.

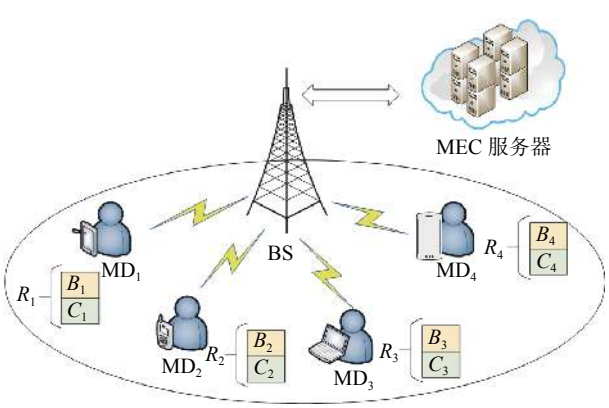

图 1 多用户单小区 MEC 系统架构

Fig. 1 Multi-user single-cell MEC system architecture

由于 MEC 服务器为多个移动用户提供服务, 计算任务可以选择卸载到 MEC 服务器或本地设备执行. 定义用户 i 的二进制卸载决策变量为 $a_i \in \{0,1\}$, 其中 $a_i=1$ 表示其计算任务将决定通过无线信道卸载到 MEC 服务器, $a_i=0$ 表示任务在本地设备上进行计算. 因此, $A=\{a_1,a_2,\cdots,a_N\}$ 为所有用户卸载决策集合.

表 1　符号表

Tab. 1　Symbol table

符号	含义
N	小区内移动设备/用户的总数
B_i	用户 i 的计算任务数据大小
C_i	用户 i 的计算任务完成所需的计算资源量
a_{i}	二进制卸载决策变量
s_i	任务的计算资源分配变量
A	所有任务卸载决策集合
S	计算资源分配集合
K	基站最多可以服务的用户数
B	系统带宽
W	用户带宽
$S_{\max}$	MEC服务器上可用的最大计算资源量
r_i	用户 i 的上行链路传输速率
g_i	用户 i 与BS之间的信道增益
σ^2	高斯信道噪声的方差
$f_{\mathrm{l},i}$	本地设备CPU计算能力
ε_i	用户设备能耗系数
$T_{\mathrm{l},i}$	用户 i 的本地计算时延
$E_{\mathrm{l},i}$	用户 i 的本地计算能耗
p_i	用户设备传输功率
$p_{\mathrm{o},i}$	用户设备空闲状态下电路功率
$T_{\mathrm{e},i}$	任务卸载到MEC服务器的时延
$E_{\mathrm{e},i}$	任务卸载到MEC服务器的能耗
$\beta_{\mathrm{t},i}$	用户 i 的任务时延偏好参数
$\beta_{\mathrm{e},i}$	用户 i 的任务能耗偏好参数

对于小区中不同用户传输, 采用正交频分多址方案. 整个频谱被均匀的划分为 K 个子信道, 将每个任务分配给 1 个子信道, 确保用户上行链路传输信道的正交性. 定义 B 为无线信道的系统带宽, W 为用户带宽. 因此, BS 最多可以服务的用户数为

$$K = \frac{B}{W}. \tag{1}$$

每个移动用户通过正交频分复用信道接入小区基站, 根据香农公式可知, 移动用户 i 的上行链路传输速率为

$$r_i = W\log_2\left(1 + \frac{p_i g_i}{\sigma^2}\right), \tag{2}$$

其中, $\frac{p_i g_i}{\sigma^2}$ 为信噪比, p_i 表示用户 i 的传输功率, g_i 表示用户 i 与 BS 之间的信道增益, σ^2 为高斯信道噪

声的方差.

1.2 计算模型

计算任务的处理可分为本地计算和移动边缘计算两种方式, 下面分别介绍这两种计算方式在时延和能耗方面的计算开销.

(1) 本地计算

对于本地计算方式, 每个用户在本地设备执行其计算任务. 由表 1 可知, C_i 表示用户任务的计算需求, 即完成计算任务所需的总 CPU 周期数. 定义 $f_{\mathrm{l},i}$ 表示用户 i 的本地设备 CPU 计算能力, 即每秒 CPU 周期数. 用户 i 的任务在本地计算所需的时延为

$$T_{\mathrm{l},i} = \frac{C_i}{f_{1,i}}. \tag{3}$$

相应地, 本地计算所需的能耗为

$$E_{1,i} = \varepsilon_i C_i, \tag{4}$$

其中, ε_i 是能耗系数, 表示用户 i 的本地设备每个 CPU 周期消耗的能量, 是用户 CPU 计算能力的超线性函数[12-13]. 设置能耗模型为 $\varepsilon_i = k(f_{1,i})^r$, 其中 k 是芯片的能耗系数, 这里取 $k = 10^{-27}, r = 2$.

(2) 移动边缘计算

对于移动边缘计算方式, 每个用户将根据卸载决策通过无线信道将任务卸载到 MEC 服务器上. 任务卸载过程分为上传任务、处理任务和下载任务 3 个部分. 任务上传时延为 $T_{\mathrm{t},i} = \frac{B_i}{r_i}$, 处理时延为 $T_{\mathrm{c},i} = \frac{C_i}{s_i}$. 由于任务处理后的数据大小远小于任务执行之前的数据大小, 并且下行链路传输速率也远远高于上行链路传输速率, 因此下载任务的时延和能耗一般忽略不计. 根据用户上行链路传输速率 r_i, 任务卸载到 MEC 服务器所需的时延为

$$T_{\mathrm{e},i} = T_{\mathrm{t},i} + T_{\mathrm{c},i} = \frac{B_i}{r_i} + \frac{C_i}{s_i}, \tag{5}$$

其中, s_i 表示 MEC 服务器分配给任务 i 的计算资源量. 定义 $S_{\max}$ 表示 MEC 服务器上可用的最大计算资源量, 即服务器 CPU 计算能力, 且满足 $\sum_{i=1}^{N} a_i s_i \leqslant S_{\max}$.

相应地, 任务卸载到 MEC 服务器所需的能耗为

$$E_{\mathrm{e},i} = p_i T_{\mathrm{t},i} + p_{\mathrm{o},i} T_{\mathrm{c},i} = p_i \frac{B_i}{r_i} + p_{\mathrm{o},i} \frac{C_i}{s_i}, \tag{6}$$

其中, p_i 和 $p_{\mathrm{o},i}$ 分别是用户 i 设备的传输功率大小和空闲状态下电路功率大小.

2 问题描述

在 MEC 系统中, 用户的服务质量由完成任务的延迟和能耗决定. 定义 $\beta_{\mathrm{t},i}$ 和 $\beta_{\mathrm{e},i}$ 分别表示第 i 个用户任务执行时延和能耗的偏好参数, 满足 $\beta_{\mathrm{t},i}, \beta_{\mathrm{e},i} \in [0,1]$, $\beta_{\mathrm{t},i} + \beta_{\mathrm{e},i} = 1$, 可以根据任务完成时间的需求和剩余电池寿命来进行设置. 例如, 在线游戏、高清视频等时延要求较高的应用可以增加 $\beta_{\mathrm{t},i}$ 并减少 $\beta_{\mathrm{e},i}$ 从而最大限度减少延迟; 而当设备电量较低时, 可以减少 $\beta_{\mathrm{t},i}$ 并增加 $\beta_{\mathrm{e},i}$ 来节省设备能耗. 考虑到不同用户的偏好, 定义用户 i 的计算总成本为时延和能耗的加权和, 则可获得本地计算和移动边缘计算下的总成本, 分别为

$$C_{1,i} = \beta_{\mathrm{t},i} T_{1,i} + \beta_{\mathrm{e},i} E_{1,i}, \tag{7}$$

$$C_{\mathrm{e},i} = \beta_{\mathrm{t},i} T_{\mathrm{e},i} + \beta_{\mathrm{e},i} E_{\mathrm{e},i}. \tag{8}$$

对于所有用户 $i \in N$, 计算系统总成本, 总成本为

$$C_{\text{total}}(A,S) = \sum_{i=1}^{N} (1-\alpha_i) C_{1,i} + \alpha_i C_{\mathrm{e},i}. \tag{9}$$

为提高用户任务卸载效率, 本文主要是通过优化卸载决策和计算资源分配来最小化任务总成本. 优化问题 P1 建模如下.

$$\begin{cases} \mathrm{P1}: \min\limits_{A,S} C_{\text{total}}(A,S), \\ \text{约束条件}: \\ \mathrm{C1}: \sum_{i=1}^{N} a_i \leqslant K, \\ \mathrm{C2}: a_i \in \{0,1\},\ i = 1,2,\cdots,N, \\ \mathrm{C3}: \sum_{i=1}^{N} a_i s_i \leqslant S_{\max}, \\ \mathrm{C4}: 0 \leqslant s_i \leqslant S_{\max},\ i = 1,2,\cdots,N. \end{cases} \tag{10}$$

式 (10) 中, $A = \{a_1, a_2, \cdots, a_i, \cdots, a_N\}$是所有任务卸载决策集合; $S = \{s_1, s_2, \cdots, s_i, \cdots, s_N\}$是计算资源分配集合; 约束 C1 控制信道容量, BS 最多可以同时服务 K 个用户; 约束 C2 确保计算卸载决策是二进制变量; 约束 C3 表示 MEC 服务器 CPU 计算资源量约束; 最后约束 C4 限制资源分配范围.

由于存在整数变量 A 以及线性变量 S, 问题 P1 为 1 个 MINILP 问题, 被证明是 NP-hard 的. 随着用户数量的急剧增加, 直接解决此问题非常困难, 因此必须找到一种有效且简化的解决方案.

3　计算卸载和资源分配策略

观察到问题 P1 中约束 C1 和约束 C2 用于约束整数变量 A, 约束 C3 和约束 C4 用于约束线性变量 S, 彼此是相互分离的, 因此本文考虑将优化问题拆分为两个子问题: 计算资源分配子问题及计算卸载子问题, 最后, 通过迭代获得最优的卸载决策和资源分配结果.

3.1　基于拉格朗日乘子法的计算资源分配

因为 MEC 服务器仅为确定卸载的用户任务分配计算资源, 给定 U 为卸载用户集合, 即 $U = \{i|\alpha_i = 1\}$. 首先, 制定计算资源分配子问题 P2, 为

$$\begin{cases} \mathrm{P2}: \min\limits_{S} C_{\text{total}}(S) = \sum\limits_{i \in U} \beta_{\mathrm{t},i} \left(\frac{B_i}{r_i} + \frac{C_i}{s_i}\right) + \beta_{\mathrm{e},i} \left(p_i \frac{B_i}{r_i} + p_{\mathrm{o},i} \frac{C_i}{s_i}\right), \\ \text{约束条件}: \\ \mathrm{C3}: \sum\limits_{i \in U} s_i \leqslant S_{\max}, \\ \mathrm{C4}: 0 \leqslant s_i \leqslant S_{\max},\ i \in U. \end{cases} \tag{11}$$

计算目标函数的二阶导数, 可得

$$\frac{\partial^2 C_{\text{total}}(S)}{\partial s_i \partial s_j} = \begin{cases} \dfrac{2\left(\beta_{\mathrm{t},i} + \beta_{\mathrm{e},i} p_{\mathrm{o},i}\right) C_i}{s_i^3} \geqslant 0, & i = j, \\ 0, & \text{其他}. \end{cases} \tag{12}$$

黑塞矩阵 (Hessian Matrix) 中参数都是正数, 因此目标函数 $C_{\text{total}}(S)$ 的黑塞矩阵是正定的; 而 $C_{\text{total}}(S)$ 的域是凸的, 这是一个凸优化问题, 用拉格朗日乘子法进行求解.

通过引入拉格朗日乘子 λ, 建立问题 P2 的拉格朗日函数

$$L(S,\lambda)=C_{\text{total}}(S)+\lambda\left(\sum_{i\in U}s_i-S_{\max}\right). \tag{13}$$

通过 KKT 条件, 有

$$\frac{\partial L(S,\lambda)}{\partial s_i}=0, \tag{14}$$

$$\sum_{i\in U}s_i-S_{\max}=0. \tag{15}$$

通过求解上述公式, 得到最优资源分配 s_i^* 为

$$s_i^*=\frac{\sqrt{(\beta_{\text{t},i}+\beta_{\text{e},i}p_{\text{o},i})C_i}}{\sum_{i\epsilon U}\sqrt{(\beta_{\text{t},i}+\beta_{\text{e},i}p_{\text{o},i})C_i}}S_{\max}. \tag{16}$$

将式 (16) 代入 P2, 得到目标函数最优值, 为

$$\begin{aligned}C_{\text{total}}(S^*)&=\sum_{i\in U}\beta_{\text{t},i}\left(\frac{B_i}{r_i}+\frac{C_i}{s_i^*}\right)+\beta_{\text{e},i}\left(p_i\frac{B_i}{r_i}+p_{\text{o},i}\frac{C_i}{s_i^*}\right)\\&=\frac{1}{S}\left(\sum_{i\in U}\sqrt{(\beta_{\text{t},i}+\beta_{\text{e},i}p_{\text{o},i})C_i}\right)^2+\sum_{i\in U}(\beta_{\text{t},i}+\beta_{\text{e},i}p_i)\frac{B_i}{r_i}.\end{aligned} \tag{17}$$

3.2 基于贪心算法的计算卸载算法

通过上面的讨论, 已经为给定用户卸载集合 U 获得了最优资源分配的封闭形式解, 并计算得了最优的 $C_{\text{total}}(S^*)$. 在此基础上, 原始优化问题转换为

$$\begin{cases}\min\limits_{A} C_{\text{total}}(A),\\ \text{约束条件}:\\ \text{C1}:\sum_{i=1}^{N}a_i\leqslant K,\\ \text{C2}:a_i\in\{0,1\},\ i=1,2,\cdots,N.\end{cases} \tag{18}$$

可以看出这是关于 A 的 0-1 规划问题, 也是 NP 完全问题. 解决该问题的简单方法是通过分支定界 (Branch and Bound, BB) 算法, 在遍历所有卸载决策情况下, 算法计算复杂度为 $O\left(2^N\right)$. 本文提出了一种基于贪心算法的计算卸载算法, 该算法基本思想是将问题变量先进行松弛, 即先求解该松弛问题, 再把变量修正到 $\{0,1\}$ 上, 得到问题的解.

步骤一: 采用变量松弛法将约束条件 C2 中的二元变量 α_i 进行松弛, 问题可以等价地替换为问题 P3, 即

$$\begin{cases}\text{P3}:\min\limits_{A} C_{\text{total}}(A),\\ \text{约束条件}:\ \sum_{i=1}^{N}a_i\leqslant K,\\ \sum_{i=1}^{N}a_i(1-a_i)=0,\\ \alpha_i\in[0,1],\ i=1,2,\cdots,N.\end{cases} \tag{19}$$

在问题 P3 的约束条件中, 根据变量 α_i 的离散特点, 引入了一个有关于 α_i 的非线性等式约束来代替变量 α_i 的 0-1 离散限制, 将原问题转化为 1 个 $[0,1]$ 区间上的等价非线性约束优化问题. 文献 [14] 通

过数学归纳法和反证法对两个问题进行了等价性证明. 考虑到求解非线性约束优化问题的困难, 采用混合罚函数法进行求解, 根据罚函数的思想, 建立新的定义域内的无约束目标函数

$$G\left(A,r_k\right)=C_{\text{total}}\left(A\right)-r_k\ln\left(K-\sum_{i=1}^{N}a_i\right)+\frac{1}{r_k}\sum_{i=1}^{N}\left[\alpha_i\left(1-\alpha_i\right)\right]^2, \tag{20}$$

其中, r_k 为 $G\left(A,r_k\right)$ 的惩罚因子, 通过牛顿法求解上面无约束问题; $\ln\left(K-\sum_{i=1}^{N}a_i\right)$ 相当于内点罚函数的作用, 能够限制搜索点一直在不等式约束确定的区域; $\sum_{i=1}^{N}\left[\alpha_i\left(1-\alpha_i\right)\right]^2$ 相当于外点罚函数的作用, 能够迫使搜索点靠近等式约束.

步骤二: 由于混合罚函数法得到的解违反了问题 P3 的最后一个约束条件, 不是该问题的可行解, 卸载决策变量还需要将其修正到 $\{0,1\}$ 上. 算法 1 为基于贪心算法的计算卸载算法. 如算法 1 所示, 首先定义 MEC 服务器剩余计算资源量为 S_{remain}, 其初始值为最大计算资源 $S_{\max}$ (行 1); 其次利用混合罚函数法求解问题 P3, 经过 k 次迭代后获得的解为 $A^k=\left\{a_i^k|i=1,2,\cdots,N\right\}$ (行 2); 然后通过不断迭代将所有卸载决策变量进行修正. 在每次迭代过程中, 不断寻找 A^k 中最大卸载决策变量 a_i^k 的任务 i (行 4), 因为卸载决策变量 a_i^k 越大, 意味着第 i 个任务进行卸载的可能性就越大. 然后, 将该任务的计算资源 s_i 与 MEC 服务器剩余计算资源量 S_{remain} 进行比较, 若满足计算容量限制, 则将该任务卸载决策 a_i 置为 1, 同时更新 MEC 服务器剩余计算资源量, 否则 a_i 置为 0 (行 5—行 10), 当完成决策变量 a_i 修正后, 就将该变量从 A^k 中删除 (行 11). 最后, 不断重复上述过程直到 A^k 中卸载变量全部修正完成, 即 $A^k=\varnothing$, 则迭代结束, 得到最终的任务决策变量 A.

算法 1 基于贪心算法的计算卸载算法

输入: 用户数 N; 资源分配 $S=\{s_i|i=1,2,\cdots,N\}$; $S_{\max}$

输出: 卸载决策 A

1: 初始化 $S_{\text{remain}}=S_{\max}$
2: 利用混合罚函数法求解问题 P3 得到 $A^k=\left\{a_i^k\middle|i=1,2,\cdots,N\right\}$
3: **While** $A^k\neq\varnothing$ **do**
4: 寻找 A^k 中最大 a_i^k 的任务 i, 即 $i=\text{argmax}\left\{a_i^k\middle|a_i^k\in A^k,i=1,2,\cdots,N\right\}$
5: **If** $S_{\text{remain}}\geqslant s_i$ **then**
6: $a_i=1$
7: $S_{\text{remain}}=S_{\max}-s_i$
8: **Else**
9: $a_i=0$
10: **End if**
11: 将 a_i^k 从 A^k 中移除, 即 $A^k=A^k\backslash\{a_i^k\}$
12: **End while**

3.3 联合优化策略总结

联合计算卸载和资源分配策略通过相互迭代两个子问题的解来获得最佳卸载决策和资源分配. 算法 2 为联合优化算法. 如算法 2 所示, 初始化阶段定义初始成本 $C_{\text{total,old}}$ 以及最小误差 ε (行 1), 然后进入迭代阶段, 每次迭代过程中都先通过拉格朗日乘子法求解资源分配子问题, 获得初始卸载决策 A^0 下的计算资源分配结果 S^t, 并且通过算法 1 求解计算卸载子问题, 获得卸载决策结果 A^t (行 3—行

4). 根据公式 (9) 重新计算成本, 若前后成本差值满足误差条件, 将第 t 次迭代得到的卸载决策和资源分配向量作为最佳解, 算法停止 (行 6—行 10), 否则保存成本 (行 11) 进入下次迭代.

算法 2 联合优化算法

输入: 任务数 N; $R_i=\{B_i,C_i\}$; A^0; S^0

输出: 最佳卸载决策 A^*; 资源分配 S^*; 系统成本 C_{total}

1: 初始化 $C_{\text{total,old}}=0$, $\varepsilon=$ 1e-6
2: **For** $t=1$ **to do**
3: 根据初始 A^0 通过拉格朗日乘子法获得资源分配 S^t
4: 通过算法 1 获得卸载决策 A^t
5: 计算成本 C_{total}
6: **If** $|C_{\text{total}}-C_{\text{total,old}}|\leqslant\varepsilon$ **then**
7: $A^*=A^t$
8: $S^*=S^t$
9: **Break**
10: **End if**
11: $C_{\text{total,old}}=C_{\text{total}}$
12: **End for**

用户任务数目设置为 N, 该算法首先获得计算资源分配需要经过 N 次计算, 计算复杂度为 $O(N)$, 然后通过算法 1 获得卸载决策, 计算复杂度为 $O(KN+N)$, 其中 K 是使混合罚函数算法达到收敛的迭代次数, 最后经过通过 T 次迭代获得最佳卸载决策和最佳资源分配, 算法的总计算复杂度为 $O(KNT+2NT)$, 该算法相比传统分支定界算法的计算复杂度更低.

4 实验仿真和性能评估

本章采用 Python 语言进行仿真实验, 通过设置相应仿真参数来评估本文所提算法的性能.

4.1 实验设置

在仿真场景中, MEC 服务器位于 BS 附近, 网络中有 15 到 35 个随机分布的用户移动设备, BS 覆盖范围为 1 000 m. 设置系统带宽 $B=20$ MHz, 用户传输带宽 $W=1$ MHz, 因此 $K=20$, 通信参数遵循第三代合作伙伴计划规范[15]. 设置无线信道增益模型为 $g_i=127+30\times\log_{10}d$, 其中 d 为用户与 MEC 服务器之间的距离; 高斯信道噪声 $\sigma^2=2\times10^{-13}$ W; 移动设备 CPU 计算能力 $f_{\text{l},i}=1$ GHz/s; MEC 服务器最大计算资源量 $S_{\max}=20$ GHz/s, 移动设备传输功率 $p_i=0.5$ W, 以及空闲状态下功率 $p_{\text{o},i}=0.1$ W; 用户偏好参数为 $\beta_{\text{t},i}=\beta_{\text{e},i}=0.5$. 与文献 [16] 工作类似, 假设任务数据通过概率分布 (即, 正态分布和均匀分布) 生成: 任务数据大小 B_i 和计算资源量 C_i 由正态分布生成时, $B_i\sim N(500,100)$ kB, $C_i\sim N(1\,000,100)$ megacycles; 任务数据大小 B_i 和计算资源量 C_i 由均匀分布生成时, $B_i\sim U(400,600)$ kB, $C_i\sim U(800,1\,200)$ megacycles. 此外, 初始用户卸载集合为全部用户, 卸载决策集合 A 初始化为全 1 以及计算资源分配 S 初始化为全 0.

仿真实验结果取 100 次实验的平均值, 分别从移动设备数目, 任务计算资源量, MEC 服务器计算资源量, 用户偏好参数几个方面对本文所提算法进行评估和分析. 同时, 将本文所提算法与本地执行、全卸载算法、BB 卸载算法以及带汉明距离终止的动态规划 (Dynamic Programming with Hamming

Distance Termination, DPH) 卸载算法[17] 进行对比实验. 注意当任务数超过 BS 容纳的最大数量时, 所有系统带宽均已被占用, 移动用户默认选择本地执行其计算任务, MEC 服务器中资源分配均由拉格朗日乘子法给出.

4.2　性能评估

图 2 显示了移动设备数量对系统成本 (时延和能耗的加权和) 的影响, 移动设备数量 (N) 从 15 增加到 35. 由图 2 可以看到, 所有算法成本均随移动设备数量的增加而增加, 其中有两条重叠的曲线, 表明本文所提算法以及 BB 算法的系统成本几乎相等, 因此, 本文所提算法可以收敛到由 BB 算法得出的最优解, 并且相比本地执行算法、全卸载算法和 DPH 算法, 本文所提算法的系统成本最低. 在图 2(a) 和图 2(b) 中, 本文所提算法相比本地执行算法最高可以减少 40% 的系统成本.

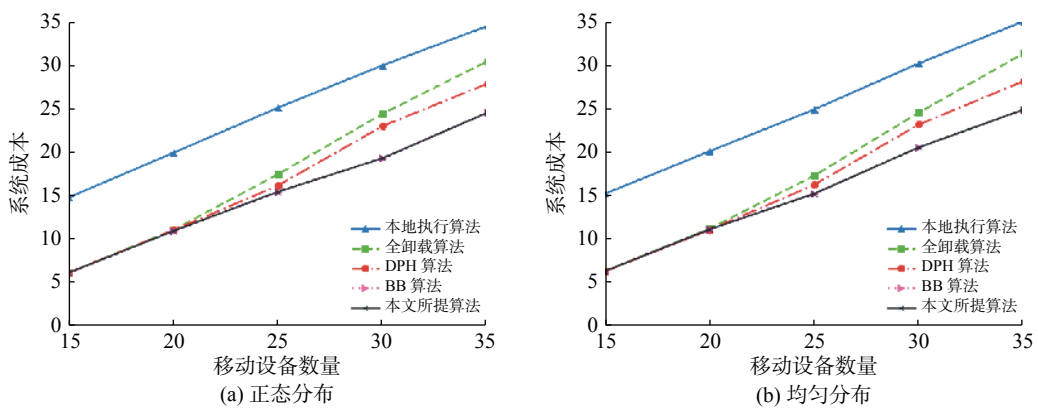

图 2　移动设备数量对系统成本的影响

Fig. 2　Impact of MD number on system cost

图 3 显示了不同任务计算资源量对系统成本的影响, 移动设备数量设置为 25. 从仿真结果来看, 随着任务计算量的增加, 算法系统成本也随之增加. 而且在所有算法中, 本文所提算法的系统成本最低. 在图 3(a) 中, 本文所提算法相比本地执行算法可以减少 40% 的系统成本; 与全卸载算法和 DPH 算法相比, 本文所提算法可以减少 10% 的系统成本, 从图 3(b) 中也得到了同样的结论.

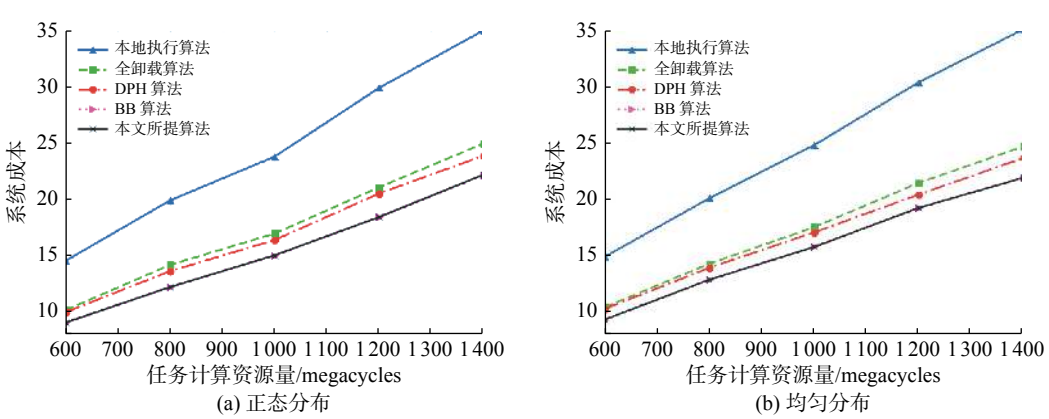

图 3　任务计算资源量对系统成本的影响

Fig. 3　Impact of task computation resources on system cost

为了分析 MEC 服务器计算资源量如何影响系统成本, 在图 4 中, 移动设备数量设置为 25, 通过改变服务器容量, 对比各卸载算法的系统成本. 从图 4(a) 和 4(b) 中可以看出, 当 MEC 服务器计算资源

量增加时, 本地执行算法的系统成本基本不变, 这是因为任务执行只是消耗了用户设备的计算资源, 与 MEC 服务器无关. 对于其他算法, MEC 服务器计算资源量越大, 则分配给每个卸载任务的计算资源量就越多, 系统成本自然降低. 特别注意到, 当 MEC 服务器计算资源量小于 14 时, 全卸载算法的系统成本最高, 这是由于服务器计算资源受限, 每个卸载任务分配到的计算资源较少, 才导致高成本. 反之, 当 MEC 服务器计算资源量大于 25 时, 服务器计算资源充足, 全卸载算法, DPH 算法, BB 算法以及所提算法的系统成本均缓慢降低, 并且最终趋近一致.

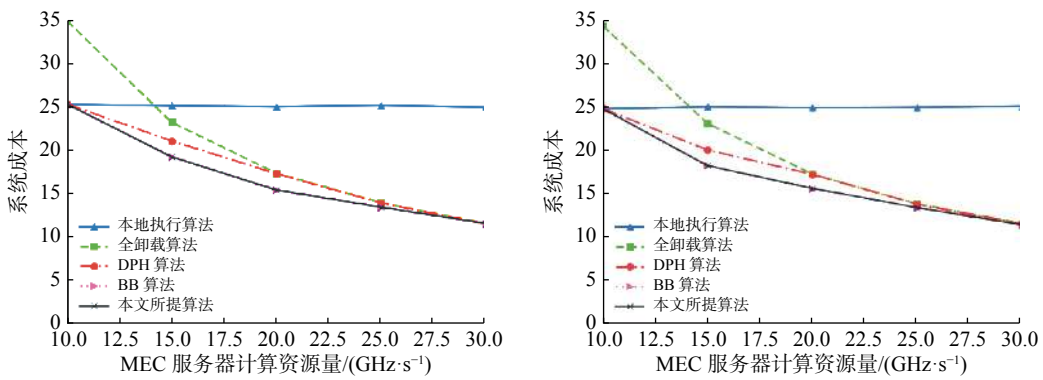

图 4 MEC 服务器计算资源量对系统成本的影响

Fig. 4 Impact of MEC server computation resources on system cost

图 5 显示了本文算法在不同时延偏好参数下任务平均时延和能耗, 同时对比了移动设备数量 (N) 为 20 和 30 的这两种情况. 设置任务数据大小和计算量都服从正态分布, 时延偏好参数 $\beta_{\mathrm{t},i}$ 从 0.1 变化到 0.9, 则能耗偏好参数为 $\beta_{\mathrm{e},i}=1-\beta_{\mathrm{t},i},\ i=1,2,\cdots,N$. 在时延偏好参数 $\beta_{\mathrm{t},i}$ 增加的情况下, 任务的平均时延降低, 代价是消耗更多的能量. 在 $N=30$ 的情况下, 用户竞争计算资源导致任务卸载的可能性降低, 因此, 与 $N=20$ 的情况相比, 任务具有更高的平均延迟和能耗.

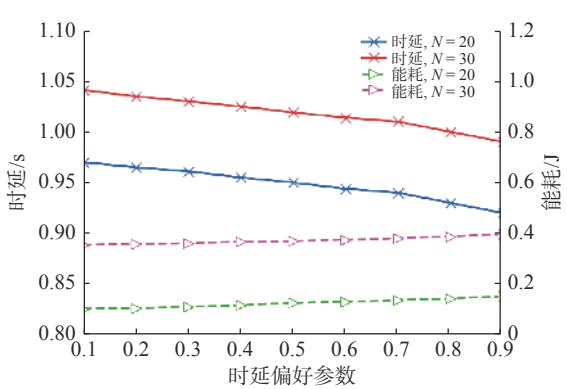

图 5 不同时延偏好参数下任务的平均时延和能耗

Fig. 5 Average delay and energy consumption of tasks with different delay preference parameters

5 结 论

本文为降低多用户 MEC 系统任务卸载的时延和能耗, 提出了一个联合计算卸载和资源分配策略. 优化问题以最小化所有用户计算时延和设备能耗的加权和为目标, 共同优化了卸载决策和计算资源分配. 优化问题被拆分为资源分配子问题和计算卸载子问题, 先通过拉格朗日乘子法求解资源分配子

问题, 然后提出一个基于贪心算法的计算卸载算法来求解计算卸载子问题, 最后给出联合优化算法并证明其具有较低的计算复杂度. 实验仿真验证了本文所提算法的有效性, 该算法与其他基准算法相比能实现更低的系统成本. 在未来, 这项工作将扩展到多服务器的移动边缘系统中, 并考虑用户设备功率、信道干扰等因素的影响.

[参　考　文　献]

[1] GOHIL A, MODI H, PATEL S K. 5G technology of mobile communication: A survey [C]// 2013 International Conference on Intelligent Systems and Signal Processing (ISSP). IEEE, 2013: 288-292.

[2] MACH P, BECVAR Z. Mobile edge computing: A survey on architecture and computation offloading [J]. IEEE Communications Surveys and Tutorials, 2017, 19(3): 1628-1656.

[3] HU Y C, PATEL M, SABELLA D, et al. Mobile Edge Computing: A Key Technology Towards 5G [M]. [S.l.]: ETSI (European Telecommunications Standards Institute) , 2015.

[4] ZHANG D Y, TANG J Z, DU W T, et al. Joint optimization of computation offloading and UL/DL resource allocation in MEC systems [C]// 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). IEEE, 2018: 6 pages. DOI: 10.1109/PIMRC.2018.8580841.

[5] ALAM M G R, HASSAN M M, UDDIN M Z, et al. Autonomic computation offloading in mobile edge for IoT applications [J]. Future Generation Computer Systems, 2019, 90: 149-157.

[6] PAYMARD P, REZVANI S, MOKARI N. Joint task scheduling and uplink/downlink radio resource allocation in PD-NOMA based mobile edge computing networks [J]. Physical Communication, 2019, 32: 160-171.

[7] LIU J H, ZHANG Q. Computation resource allocation for heterogeneous time-critical IoT services in MEC [EB/OL]. (2020-02-12)[2020-05-30]. https://arxiv.org/abs/2002.04851.

[8] GUO F X, ZHANG H L, JI H, et al. Energy efficient computation offloading for multi-access MEC enabled small cell networks [C]// 2018 IEEE International Conference on Communications Workshops (ICC Workshops). IEEE, 2018: 6 pages. DOI: 10.1109/ICCW.2018.8403701.

[9] LI C L, TANG J H, ZHANG Y, et al. Energy efficient computation offloading for nonorthogonal multiple access assisted mobile edge computing with energy harvesting devices [J]. Computer Networks, 2019, 164: 106890.

[10] QIAN L P, ZHU Z Y, YU N N, et al. Joint minimization of transmission energy and computation energy for MEC-aware NOMA NB-IoT networks [C]// 2019 IEEE Global Communications Conference (GLOBECOM). IEEE, 2019: 7 pages. DOI: 10.1109/GLOBECOM38437.2019.9013350.

[11] ZENG M, FODOR V. Energy minimization for delay constrained mobile edge computing with orthogonal and non-orthogonal multiple access [J]. Ad Hoc Networks, 2020, 98: 102060.

[12] LI J, GAO H, LYU T J, et al. Deep reinforcement learning based computation offloading and resource allocation for MEC [C]// 2018 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, 2018: 6 pages. DOI: 10.1109/WCNC.2018.8377343.

[13] HUANG C M, CHIANG M S, DAO D T, et al. V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture [J]. IEEE Access, 2018(6): 17741-17755.

[14] 隋允康, 贾志超. 0-1线性规划的连续化及其遗传算法解法 [J]. 数学的实践与认识, 2010, 40(6): 119-127.

[15] 3GPP Technical Specification Group Radio Access Network. Further advancements for E-UTRA physical layer aspects(Release 9) [R]. 3GPP TS 36.814 V9.0.0, 2010.

[16] CHEN M, HAO Y X. Task offloading for mobile edge computing in software defined ultra-dense network [J]. IEEE Journal on Selected Areas in Communications, 2018, 36(3): 587-597.

[17] SHAHZAD H, SZYMANSKI T H. A dynamic programming offloading algorithm for mobile cloud computing [C]// 2016 IEEE 9th International Conference on Cloud Computing (CLOUD). IEEE, 2016: 960-965.

(责任编辑: 李　艺)