Journal of System Simulation

Volume 34 | Issue 7 Article 9

7-20-2022

Game-Based Resource Allocation and Task Offloading Scheme in Collaborative Cloud-Edge Computing System

Xuewen Wu
School of Computer and Information, Hohai University, Nanjing 211100, China;, hhuwxw@hhu.edu.com

Jingxian Liao
School of Computer and Information, Hohai University, Nanjing 211100, China;

Follow this and additional works at: https://dc-china-simulation.researchcommons.org/journal

 Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, Numerical Analysis and Scientific Computing Commons, Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Science Commons

This Paper is brought to you for free and open access by Journal of System Simulation. It has been accepted for inclusion in Journal of System Simulation by an authorized editor of Journal of System Simulation.

Game-Based Resource Allocation and Task Offloading Scheme in Collaborative Cloud-Edge Computing System

Abstract

Abstract: Considering the delay, energy consumption and computing resource cost, the utility maximization problem in collaborative cloud-edge system is constructed, and divided into three subproblems: computing resource allocation, uplink power allocation and task offloading strategy. *A game-based resource allocation and task offloading(GRATO) scheme is proposed to solve those subproblems. The optimal solution of computing resource allocation is obtained by using convex optimization conditions; a low complexity uplink power allocation method is designed to reduce wireless interfere; a game-based distributed task offloading algorithm (GDTOA) is proposed to optimize the task offloading strategy.* Simulation results show that the performance of GRATO is better than other schemes on delay and energy consumption, and it can sense the priority of users, resulting in higher utility and lower latency for emergency users..

Keywords

edge computing, resource allocation, computation task offloading, game, utility maximization

Recommended Citation

Xuewen Wu, Jingxian Liao. Game-Based Resource Allocation and Task Offloading Scheme in Collaborative Cloud-Edge Computing System[J]. Journal of System Simulation, 2022, 34(7): 1468-1481.

This paper is available in Journal of System Simulation: https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9

第 34 卷第 7 期 系统仿真学报© Vol. 34 No. 7
2022 年 7 月 Journal of System Simulation Jul. 2022

云边协同系统中基于博弈论的资源分配与任务卸载方案

吴学文，廖婧贤
(河海大学 计算机与信息学院，江苏 南京 211100)

摘要：综合考虑时延、能耗和计算资源成本，构建云边协同系统中的效用最大化问题，并将其分解为计算资源分配、上行功率分配和任务卸载策略三个子问题。*提出一种基于博弈论的资源分配和任务卸载方案（game-based resource allocation and task offloading， GRATO）*以分别解决上述子问题。*利用凸优化条件求得计算资源分配最优解；设计一种低复杂度的上行功率分配方法用于降低无线干扰；针对任务卸载策略优化问题，提出一种基于博弈论的分布式任务卸载算法（game-based distributed task offloading algorithm， GDTOA）*。仿真结果表明，GRATO 方案在时延和能耗方面的性能优于其他方案，还可以感知用户的优先级，使紧急用户具有更高的效用和更低的时延。

关键词：边缘计算；资源分配；计算任务卸载；博弈；效用最大化
中图分类号：TP301.6；TP391　　文献标志码：A　　文章编号：1004-731X(2022)07-1468-14
DOI: 10.16182/j.issn1004731x.joss.21-0077

Game-Based Resource Allocation and Task Offloading Scheme in Collaborative Cloud-Edge Computing System

Wu Xuewen, *Liao Jingxian*
(School of Computer and Information, Hohai University, Nanjing 211100, China)

Abstract: Considering the delay, energy consumption and computing resource cost, the utility maximization problem in collaborative cloud-edge system is constructed, and divided into three subproblems: computing resource allocation, uplink power allocation and task offloading strategy. *A game-based resource allocation and task offloading(GRATO) scheme is proposed to solve those subproblems. The optimal solution of computing resource allocation is obtained by using convex optimization conditions; a low complexity uplink power allocation method is designed to reduce wireless interfere; a game-based distributed task offloading algorithm (GDTOA) is proposed to optimize the task offloading strategy.* Simulation results show that the performance of GRATO is better than other schemes on delay and energy consumption, and it can sense the priority of users, resulting in higher utility and lower latency for emergency users..

Keywords: edge computing; resource allocation; computation task offloading; game; utility maximization

引言

随着5G通信和物联网技术的发展，越来越多的移动应用对实时通信和密集计算提出严格的要求[1]，如超低延迟和高可靠性。在传统的网络中，终端设备通常将数据传输到具有丰富计算资源的集中云处理。然而，移动数据的爆炸性增长给云和无线链路带来了沉重的负担，无线通信系统的性能将受到影响而急剧恶化[2]。为了补充云计算能力的不足，提高用户的体验质量(quality of experience, QoE)，移动边缘计算(mobile edge computing, MEC)[3]的范式被提出，并得到广泛关

收稿日期：2021-01-27　　修回日期：2021-04-30
第一作者：吴学文(1962-)，男，博士，副教授，研究方向为复杂网络与边缘计算。E-mail：hhuwxw@hhu.edu.com

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

注。通过在蜂窝基站或本地无线接入点上部署边缘服务器，MEC将减少端到端时延，减轻回程网络的负担[4]。

用户设备与MEC服务器之间需要通信，任务卸载在时延和能耗方面会产生额外的开销。在具有大量卸载用户的系统中，MEC服务器有限的计算资源可能会增加任务的执行时延[5]。因此，通过优化卸载决策和资源分配来降低能耗或时延成本是实现高效计算卸载的关键。以往的工作多通过单独优化卸载决策[6]、通信资源分配[7]或计算资源分配[8]来解决该问题。对于该问题的研究集中在联合资源分配和任务卸载决策优化方面。在优化目标的选择上有两种方案，一是把最小化时延作为优化目标[9]，二是同时考虑降低能耗[10]。文献[11]考虑了多用户系统中的联合任务卸载和资源优化。文献[12]考虑时延和能耗影响，将联合任务卸载和资源分配问题分解为两个子问题并分别优化，以最大化系统效用。文献[13]研究三种不同的计算模型，提出了一种最优的联合通信和计算资源分配算法。

博弈论可以有效地解决多个理性参与者对目标的决策问题。Gu等[14]针对联合无线和计算资源分配问题，提出了基于匹配博弈的学生–项目分配博弈方法。Ma等[15]设计了一种面向服务的资源分配方案，并提出了涉及用户、边缘节点和服务供应商的三方循环博弈。Chen等[16]针对MEC的多种类资源分配问题，提出了一个基于Stackelberg博弈的框架，以最大化每个MEC和用户的效用。然而，这些工作研究的仅是用户和服务供应商之间的资源分配问题，并未考虑用户的计算卸载策略优化。

除了优化资源分配，也有部分工作利用博弈论方法研究计算卸载问题。文献[17]提出了一种移动云计算中实现高效计算卸载的博弈论方法，旨在最小化每个用户所受到的无线干扰。然而该文献仅以无线干扰作为指标，未考虑其他影响用户性能的因素。Hu等[18]提出了一种基于博弈论的计算卸载算法，包括任务卸载策略和用户设备的传输功率控制，该算法的目标主要是最大化卸载到MEC服务器的设备数量，并未进行实际的计算资源分配。Long等[19]提出了一种基于博弈论框架的多目标计算资源分配与计算卸载方案，然而未考虑能耗和用户受到的无线干扰等因素。

由以上分析可知，目前基于博弈论的边缘计算研究中存在以下不足：① 大多没有考虑同时利用MEC和云计算资源；② 大多没有同时使用时延、能耗和计算成本作为评价指标；③ 联合资源分配和任务卸载决策研究大部分是静态的，卸载决策后未同步更新资源分配；④ 未考虑用户对时延或能耗的敏感程度和用户自身的紧急程度。

本文针对云边协同系统，将时延、能耗和计算成本作为指标，构成使系统效用最大化的联合资源分配和任务卸载问题。当卸载决策更新后，用户的资源状况发生变化，将同步更新资源分配。本文的贡献包括：① 针对效用最大化问题，提出一种基于博弈论的云边协同资源分配和任务卸载方案(game-based resource allocation and task offloading, GRATO)，包括计算资源分配、上行功率分配和任务卸载策略优化；② 针对任务卸载策略优化问题，提出一种基于博弈论的分布式任务卸载算法(game-based distributed task offloading algorithm, GDTOA)；③ 为用户设置优先级系数，为高优先级用户分配更多资源，使其具有更低的时延和更高的效用；④ 将其他资源分配与计算卸载方案作为GRATO的对比方案，在不同的参数下进行仿真实验，结果表明GRATO方案可以取得良好的性能。

1 系统架构与效用函数

1.1 系统模型

云边协同系统模型如图1所示，包含1个云计算服务器、1个MEC服务器和N个用户。其中，MEC服务器配有一个基站(base station, BS)。用户

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

Wu and Liao: Game-Based Resource Allocation and Task Offloading Scheme in Coll

集合表示为$N=\{1,2,\cdots i,\cdots,N\}$，用户$i\in N$。每个用户$i$有一个计算任务$\tau_i$等待执行。$\tau_i$的特征表示为参数元组$\{C_i,D_i,T_i\}$，其中$C_i$为计算负载(以CPU周期数为单位)，即完成任务$\tau_i$所需的计算量，$D_i$为用户传输到服务器的输入数据量(以单位：比特)，$T_i$为时延要求，即任务的最大可容忍完成时间。

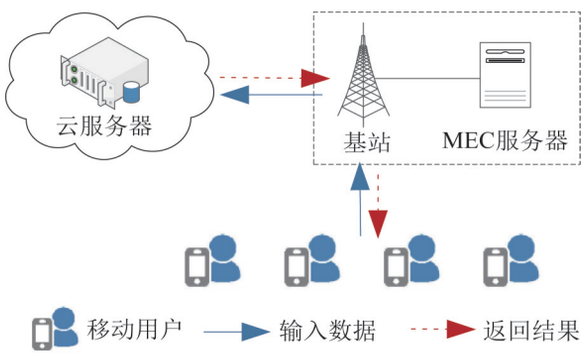


图1 系统模型
Fig. 1 System Model

计算任务可以在本地处理，也可以通过BS卸载到MEC或云上完成。卸载策略集合表示为$S=\{s_i|s_i\in\{s_i^l,s_i^m,s_i^c\},s_i^j\in\{0,1\},i\in N,j\in A\}$，其中$A=\{l,m,c\}$为用户可选的卸载决策集，$l$，$m$和$c$分别为本地计算、卸载到MEC和卸载到云三种策略。如果选择本地执行，则$s_i^j=s_i^l=1$，如果选择卸载到MEC服务器上执行，则$s_i^j=s_i^m=1$，如果选择卸载到云服务器处理，$s_i^j=s_i^c=1$。

1.2 不同卸载决策下的时延与能耗

1.2.1 本地处理模型

当用户i在本地处理其计算任务时，处理时间取决于其自身的计算资源。用f_i^l表示用户设备i的CPU计算能力(以周期数/秒为单位)，则本地计算时间表示为

$$t_i^l=\frac{C_i}{f_i^l} \tag{1}$$

为了计算用户设备在本地执行任务时的能耗，使用广泛采用的能耗模型$\varepsilon=\kappa f^2$[17]，其中κ为取决于芯片结构的能量系数，f为CPU频率。因此，在本地执行任务时，用户i的能耗表示为

$$E_i^l=\kappa(f_i^l)^2C_i \tag{2}$$

1.2.2 MEC处理模型

当用户i将计算任务τ_i卸载到MEC服务器时，其时延包括上行链路传输时间、MEC服务器执行任务的时间和将输出结果从MEC传回用户的时间。参考文献[12]，由于输出结果的大小一般比输入小得多，将接收结果的时间忽略不计。

将非正交多址(non-orthogonal multiple access, NOMA)作为上行链路中的多址接入方案，以满足5G的连接要求[18]。NOMA技术允许多个用户设备在一个信道下共享相同的资源，功率域NOMA利用发射机的叠加编码和接收机的连续干扰消除来检测信号。假设BS的信道带宽为B，σ^2为噪声功率，h_i为用户设备i与BS之间的信道增益，p_i为用户的上行传输功率，I_i为信道内其他用户对用户i造成的干扰，则上行传输数据速率r_i表示为

$$r_i=B\cdot\log_2\left(1+\frac{p_ih_i}{I_i+\sigma^2}\right) \tag{3}$$

假设MEC分配给用户i的计算资源为f_i^m，则MEC处理模型中用户i的总时延t_i^m为MEC执行时延t_i^{mc}和上行传输时延t_i^{up}之和，即

$$t_i^m=t_i^{mc}+t_i^{up}=\frac{C_i}{f_i^m}+\frac{D_i}{r_i} \tag{4}$$

用户i用于上行传输的能耗为$E_i=p_it_i^{up}/\xi_i$，其中ζ_i是用户i的功率放大器效率[12]。不失一般性地假设$\zeta_i=1$，用户i的上行传输能耗可以简化为

$$E_i^m=p_it_i^{up}=\frac{p_iD_i}{r_i} \tag{5}$$

1.2.3 云处理模型

如果用户i将计算任务卸载到远程云，则必须同时考虑将输入数据从BS发送到云的上行传输时间和将输出结果从云发回BS的下行传输时间。假设云服务器分配给用户i的计算资源为f_i^c，则云处理模型中用户i的总时延t_i^c表示为

Published by Journal of System Simulation, 2022

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

$$t_i^c = t_i^{cc} + t_i^{up} + t_i^{bc} = \frac{C_i}{f_i^c} + \frac{D_i}{r_i} + (D_i + D_i^0)v \quad (6)$$

式中：t_i^{cc}为云执行任务的时间；t_i^{up}为用户数据从本地发送到BS的上行传输时间；t_i^{bc}为BS和云之间的传输时间；v为从BS到云的单位数据传输时延；D_i^0为任务经过云处理后返回结果的数据量。

类似于MEC处理模型，云处理模型的能耗为

$$E_i^c = p_i t_i^{up} = \frac{p_i D_i}{r_i} \quad (7)$$

任务无论是卸载到MEC服务器，还是卸载到云，都要经过从用户到BS的这段上行链路。在分析信道内干扰时，采用这两种卸载策略的用户应一起考虑。将卸载到MEC服务器的用户集合记作N_m，卸载到云的用户集合记作N_c，采取卸载而非本地处理策略的用户集合记作N_{off}。

对于用户i来说，在同一个信道内来自其他用户造成的干扰表示为$I_i = \sum_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k$[17]，将其代入式(3)中，则上行数据传输速率表示为

$$r_i = B \cdot \text{lb}\left(1 + \frac{p_i h_i}{\sum_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k + \sigma^2}\right) \quad (8)$$

1.3 不同卸载策略下的效用函数

考虑任务处理时延、任务处理能耗和计算资源的成本这三个指标，定义用户i在三种不同卸载策略下的效用函数，以衡量用户的满意程度。在文献[11]中，任务的完成时间和能耗相对于本地处理的改善可以分别用$(t_i^l - t_i^m)/t_i^l$和$(E_i^l - E_i^m)/E_i^l$来表示。类似地，将任务时延与时延要求T_i比较，剩余时间越多效用越高，卸载能耗与本地处理的能耗比较，能耗的相对改善程度越高效用越高。用户i在卸载到MEC服务器和云服务器两种策略下的效用函数可以分别表示为

$$u_i^m = \beta_i^t \cdot \frac{T_i - t_i^m}{T_i} + \beta_i^e \cdot \frac{E_i^l - E_i^m}{E_i^l} - (1 - \beta_i^t - \beta_i^e) c_m f_i^m \quad (9)$$

$$u_i^c = \beta_i^t \cdot \frac{T_i - t_i^c}{T_i} + \beta_i^e \cdot \frac{E_i^l - E_i^c}{E_i^l} - (1 - \beta_i^t - \beta_i^e) c_c f_i^c \quad (10)$$

式中：β_i^t，β_i^e和$(1 - \beta_i^t - \beta_i^e)$为偏好因子，分别表示用户$i$对时延、能耗和计算资源成本影响的偏好程度，满足$\beta_i^t \in [0,1]$，$\beta_i^e \in [0,1]$，$\beta_i^t + \beta_i^e \leqslant 1$。$c_m$和$c_c$分别为MEC服务器和云服务器上的单位计算资源成本。

用户i进行本地处理的效用函数可以表示为

$$u_i^l = \beta_i^t \cdot \frac{T_i - t_i^c}{T_i} \quad (11)$$

在不同的卸载策略下，用户i的效用函数为

$$u_i = s_i^j u_i^j, i \in N, j \in \{l,m,c\} \quad (12)$$

2 问题构成

2.1 联合资源分配和任务卸载问题

将系统的效用函数定义为所有用户效用值的加权和，表示为

$$J(S,P,F) = \sum_{i \in N} \lambda_i u_i \quad (13)$$

式中：卸载策略$S = \{S^l, S^m, S^c\}$，上行链路功率分配$P = \{p_i | 0 < p_i \leqslant p^{\max}, i \in N_{\text{off}}\}$，计算资源分配$F = \{f_i^m | 0 < f_i^m \leqslant F, i \in N_{\text{m}}\}$。

$\lambda_i \in (0,1]$是用户$i(i \in N)$的优先级系数，由设备类型、用户的紧急/重要程度和计算任务的临界度决定λ_i。例如，涉及到人体健康和公共安全的设备应优先考虑。对于医疗健康应用，如果用户出现了紧急情况(如心肌梗塞、老人跌倒等)，也应为其设置更高的λ_i。对于如何确定用户的紧急程度，目前已有比较成熟的研究[20]，不在本文的讨论范围之内。假设已经获取到用户i的紧急/重要程度q_i，则可以得到优先级系数λ_i为

$$\lambda_i = \begin{cases} 1, & \text{if } q_i \geqslant q_{\text{th}} \\ \dfrac{q_i}{\sum_{i \in N} q_i}, & \text{if } q_i < q_{\text{th}} \end{cases} \quad (14)$$

式中：q_{th}为用户类型的优先级阈值，当$q_i \geqslant q_{\text{th}}$时，$\lambda_i = 1$，说明此用户非常紧急，此时这种用户

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

Wu and Liao: Game-Based Resource Allocation and Task Offloading Scheme in Coll

类型应占据绝对优先级，只要是该用户发送的任务数据，就应该立即被执行。

将联合资源分配和任务卸载问题描述为一个系统效用最大化问题，表示为

$$\begin{aligned}&\max_{S,P,F}\ J(S,P,F)\\&\text{s.t. } C1:\ 0\leqslant f_i^m\leqslant F,\forall i\in N_{\mathrm{m}}\\&\quad C2:\ \sum_{i\in N_{\mathrm{m}}} f_i^m\leqslant F,\forall i\in N_{\mathrm{m}}\\&\quad C3:\ s_i^l+s_i^m+s_i^c\leqslant 1,\forall i\in N\\&\quad C4:\ s_i^j=\{0,1\},i\in N,j\in\{l,m,c\}\\&\quad C5:\ 0<p_i\leqslant p^{\max},\forall i\in N_{\mathrm{off}}\end{aligned}\tag{15}$$

式中：$C1$是MEC服务器中用户i可用的计算资源分配的约束，$C2$是MEC服务器总体计算资源的约束。$C3$和$C4$是卸载策略约束，对于每个计算任务，一次只能选择一个卸载策略。$C5$表示上行链路功率的约束。

2.2 问题分解

式(15)是一个混合非线性规划(mixed integer nonlinear programming, MINLP)问题，非凸且NP-hard。利用Tammer分解方法[21]，暂时固定二元变量s_i^j，原问题可以分解为具有分离目标和约束的多个子问题。将问题(15)重写为

$$\begin{aligned}&\max_{S}\ (\max_{P,F} J(S,P,F)),\\&\text{s.t.}\quad C1-C5\end{aligned}\tag{16}$$

卸载决策S的约束条件C3和C4与资源分配的约束条件C1，C2和C5是互相解耦的。解决问题(16)相当于解决以下任务卸载策略问题

$$\begin{aligned}&\max_{S}\ (J^*(S))\\&\text{s.t.}\quad C3,C4\end{aligned}\tag{17}$$

式中：$J^*(S)$为资源分配问题的最优化函数

$$\begin{aligned}&J^*(S)=\max_{P,F} J(S,P,F)\\&\text{s.t.}\quad C1,C2,C5\end{aligned}\tag{18}$$

3 联合资源分配和任务卸载优化方案

本节提出一个基于博弈论的资源分配和任务卸载方案(GRATO)。针对选择卸载的用户i, $\forall i\in N_{\mathrm{off}}$，解决其资源分配问题。根据给定的满足约束$C3$和$C4$的任务卸载策略$S$，并利用效用函数的表达式$u_i^m$和$u_i^c$，得到卸载到MEC和云的用户集效用，分别为$J_m(S,P,F)$和$J_c(S,P,F)$。

将卸载到MEC服务器的用户集效用表示为

$$J_m(S,P,F)=\sum_{i\in N_m}\lambda_i(\beta_i^t+\beta_i^e)-V_m(S,P,F)\tag{19}$$

式中：$V_m(S,P,F)$为卸载到MEC服务器的用户的总开销，表示为

$$\begin{aligned}V_m(S,P,F)=\sum_{i\in N_{\mathrm{m}}}\lambda_i[&\frac{\beta_i^t t_i^m}{T_i}+\frac{\beta_i^e E_i^m}{E_i^l}+\\&(1-\beta_i^t-\beta_i^e)c_{\mathrm{m}}f_i^m]\end{aligned}\tag{20}$$

将式(4)~(5)代入式(20)中，得到

$$\begin{aligned}V_m=\sum_{i\in N_{\mathrm{m}}}\lambda_i[&\frac{D_i}{r_i}(\frac{\beta_i^t}{T_i}+\frac{\beta_i^e}{E_i^l}\cdot p_i)+\frac{\beta_i^t}{T_i}\cdot\frac{C_i}{f_i^m}+\\&(1-\beta_i^t-\beta_i^e)c_m f_i^m]\end{aligned}\tag{21}$$

类似地，卸载到云的用户总开销$V_c(S,P,F)$为

$$\begin{aligned}V_{\mathrm{c}}(S,P,F)=\sum_{i\in N_{\mathrm{c}}}\lambda_i\{&\frac{D_i}{r_i}(\frac{\beta_i^t}{T_i}+\frac{\beta_i^e}{E_i^l}\cdot p_i)+\\&\frac{\beta_i^t}{T_i}[\frac{C_i}{f_i^c}+(D_i+D_i^0)l_i]+\\&(1-\beta_i^t-\beta_i^e)c_c f_i^c\}\end{aligned}\tag{22}$$

式(19)中，$\lambda_{\mathrm{i}}(\beta_{\mathrm{i}}^t+\beta_{\mathrm{i}}^e)$对于特定的卸载策略是常量。因此，最大化卸载效用$J_j(S,P,F),j\in\{m,c\}$等价于最小化卸载开销$V_j(S,P,F),j\in\{m,c\}$，即

$$\begin{aligned}&\min_{P,F} V_j(S,P,F),j\in\{m,c\}\\&\text{s.t. } C1,C2,C5\end{aligned}\tag{23}$$

观察发现，式(23)中的问题具有可分离的结构，即上行功率分配p_i和计算资源分配$f_i^m(f_i^c)$的优化目标和约束可以相互解耦。因此，可以将问题(23)分解为计算资源分配问题和上行功率分配问题。

3.1 计算资源分配

3.1.1 MEC计算资源分配

当用户将其计算任务卸载到MEC服务器时，计算资源分配问题表示为

Published by Journal of System Simulation, 2022

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

$$\min_{F} \sum_{i \in N_m} \lambda_i [\frac{\beta_i^t}{T_i} \cdot \frac{C_i}{f_i^m} + (1 - \beta_i^t - \beta_i^e) c_m f_i^m]$$
$$s.t. \quad C1: 0 \leqslant f_i^m \leqslant F, \forall i \in N_m \tag{24}$$
$$C2: \sum_{i \in N_m} f_i^m \leqslant F, \forall i \in N_m$$

将问题(24)中的最小化目标函数记作$\Lambda(S,F)$，得到$\Lambda(S,F)$关于f_i^m的一阶导数和二阶导数

$$\frac{\partial \Lambda(S,F)}{\partial f_i^m} = -\frac{\lambda_i C_i}{(f_i^m)^2} \cdot \frac{\beta_i^t}{T_i} + (1 - \beta_i^t - \beta_i^e) c_m, \forall i \in N_m$$
$$\frac{\partial^2 \Lambda(S,F)}{\partial (f_i^m)^2} = \frac{2\lambda_i C_i}{(f_i^m)^3} \cdot \frac{\beta_i^t}{T_i} > 0, \forall i \in N_m \tag{25}$$

$\Lambda(S,F)$关于f_i^m的二阶导数恒大于零，且约束条件$C1$和$C2$均为凸，因此式(24)是一个凸优化问题，满足 slater 条件，可利用 KKT(karush kuhn-tucker)条件求最优计算资源分配F^*。

将式(24)表述为部分拉格朗日函数

$$L(f_i^m, \eta) = \sum_{i \in N_m} \lambda_i [\frac{\beta_i^t}{T_i} \cdot \frac{C_i}{f_i^m} + (1 - \beta_i^t - \beta_i^e) c_m f_i^m] + \eta (\sum_{i \in N_m} f_i^m - F) \tag{26}$$

式中：η为与计算资源约束相关的拉格朗日乘子，满足$\eta \geqslant 0$。

然后求解$L(f_i^m, \eta)$关于f_i^m的一阶导数，有

$$\frac{\partial L(f_i^m, \eta)}{\partial f_i^m} = -\frac{\beta_i^t}{T_i} \cdot \frac{\lambda_i C_i}{(f_i^m)^2} + (1 - \beta_i^t - \beta_i^e) c_m + \eta \tag{27}$$

令式(27)等于 0，得到 MEC 上计算资源分配的最佳解为

$$f_i^{m*} = \sqrt{\frac{\beta_i^t}{T_i} \cdot \frac{\lambda_i C_i}{\eta^* + (1 - \beta_i^t - \beta_i^e) c_m}}, \forall i \in N_m \tag{28}$$

由于f_i^{m*}与拉格朗日乘子η^*耦合，采用梯度下降法迭代更新η直到满足计算资源约束条件$\sum_{i \in N_m} f_i^m \leqslant F$，得到最优计算资源分配解$f_i^{m*}$，过程如算法 1 所示。

算法 1: MEC 最优计算资源分配

Initialization: very small tolerance $\varepsilon > 0$， $\eta = \eta^{\max}$

while $f_i^{m*} - f_i^m > \varepsilon$ do

set $f_i^m = f_i^{m*}$

compute f_i^m according to substitute η into (28).

If $\sum_{i \in N_m} f_i^m < F$, update $(\eta) = \eta - \Delta\eta$

end while

Optimal computation resource allocation scheme can be derived by substituting η into (28).

Output: $F = \{ f_i^{m*} | 0 < f_i^{m*} \leqslant F, i \in N_m \}$

3.1.2 云计算资源分配

尽管云服务器总是有足够的计算资源，但应考虑降低计算资源成本，也需要优化资源分配。

对效用函数u_i^c求解关于f_i^c的一阶导数和二阶导数，有

$$\frac{\partial u_i^c}{\partial f_i^c} = -\frac{\beta_i^t}{T_i} \cdot \frac{\lambda_i C_i}{(f_i^c)^2} + (1 - \beta_i^t - \beta_i^e) c_c, \forall i \in N_c$$
$$\frac{\partial^2 u_i^c}{\partial (f_i^c)^2} = \frac{2\beta_i^t}{T_i} \cdot \frac{\lambda_i C_i}{(f_i^c)^3} > 0, \forall i \in N_c \tag{29}$$

效用函数u_i^c关于f_i^c的二阶导数大于零，因此u_i^c是凸函数。利用一阶最优性条件$\partial u_i^c / \partial f_i^c = 0$，得到云计算资源分配的最优解为

$$f_i^{c*} = \sqrt{\frac{\beta_i^t}{T_i} \cdot \frac{\lambda_i C_i}{(1 - \beta_i^t - \beta_i^e) c_c}}, \forall i \in N_c \tag{30}$$

3.2 上行功率分配

卸载用户$i, \forall i \in N_{\text{off}}$关于$p_i$的开销最小化表达式为

$$\min_{P} \sum_{i \in N_{\text{off}}} \lambda_i [\frac{\beta_i^t}{T_i} \cdot \frac{D_i}{B \cdot \text{lb}(1 + \frac{p_i h_i}{\sum_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k + \sigma^2})} + \frac{\beta_i^e}{E_i^l} \cdot \frac{D_i p_i}{B \cdot \text{lb}(1 + \frac{p_i h_i}{\sum_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k + \sigma^2})}] \tag{31}$$

式(31)非凸，且复杂度很高。因为用户发射功率的变化不仅会改变自身的数据速率，还会干扰信道中其他用户的数据速率。

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

Wu and Liao: Game-Based Resource Allocation and Task Offloading Scheme in Coll

为了降低无线干扰，本节提出了一种低复杂度的离散化方法来快速分配用户的发射功率，在满足用户i时延要求的前提下最小化能耗。

设置一系列功率值$P \triangleq [p^1, p^2, \cdots, p^L]$作为发射功率的选项，满足$p^{\max} = p^1 > p^2 > \cdots > p^L > 0$，其中$L$为选项的个数，$l$为$1 \leqslant l \leqslant L$的整数。

卸载到MEC和云的任务应分别满足时延要求$t_i^m \leqslant T_i$和$t_i^c \leqslant T_i$，表示为

$$t_i^m = \frac{C_i}{f_i^m} + \frac{D_i}{B \cdot \log_2(1 + \dfrac{p_i h_i}{\sum\limits_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k + \sigma^2})} \leqslant T_i \tag{32}$$

$$t_i^c = \frac{C_i}{f_i^c} + \frac{D_i}{B \cdot \log_2(1 + \dfrac{p_i h_i}{\sum\limits_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k + \sigma^2})} + (D_i + D_i^0)v \leqslant T_i \tag{33}$$

式中：f_i^m和f_i^c为3.1节中求得的计算资源分配最优解。

用户i的功率p_i与其他用户的功率p_k耦合，式(32)和(33)依旧非凸。解决办法是为干扰$I_i = \sum\limits_{k \in N_{\text{off}} \setminus \{i\}} p_k h_k$找到一个近似值，用$p^{\max}$代替$p_k$，得到$I_i$的上界，表示为

$$I_i^a \triangleq \sum_{k \in N_{\text{off}} \setminus \{i\}} p^{\max} h_k \tag{34}$$

由于实际的I_i一定满足$I_i \leqslant I_i^a$，将所有用户的干扰都假设为最大值，得出的功率阈值一定满足任务时延要求，所以将I_i^a作为近似值是合理的。

根据时延约束，分别求出卸载到MEC和云的用户的发射功率阈值$p_i^{m'}$和$p_i^{c'}$，有

$$p_i^m \geqslant (2^{\frac{D_i}{B(T_i - \frac{C_i}{f_i^m})}} - 1)\left(\frac{I_i^a + \sigma^2}{h_i}\right) = p_i^{m'}, \forall i \in N_m \tag{35}$$

$$p_i^c \geqslant (2^{\frac{D_i}{B[T_i - \frac{C_i}{f_i^c} - (D_i + D_i^0)l]}} - 1)\left(\frac{I_i^a + \sigma^2}{h_i}\right) = p_i^{c'}, \forall i \in N_c \tag{36}$$

如果用户i卸载到MEC服务器，则$p_i' = p_i^{m'}$，否则，$p_i' = p_i^{c'}$。如果$p_i' > p^{\max}$，任务将在本地处理，令$s_i = s_i^l = 1$。如果$p' > p^{l+1}$且$p' \leqslant p^l$，则为用户i分配的功率为$p_i^* = p^l$。

3.3 任务卸载策略

本节利用博弈论方法解决任务卸载策略问题。将任务卸载策略博弈定义为$G = \{N, (S_i)_{i \in N}, (u_i)_{i \in N}\}$，其中$N$表示参与者，即所有具有计算任务的用户，$S_i$是用户$i$的卸载策略集合。用户$i$的效用函数是$u(s_i, s_{-i})$，其中$s_{-i} = (s_1, \cdots, s_{i-1}, s_{i+1}, \cdots, s_N)$，表示除了用户$i$以外的其他所有用户的卸载策略集合。

每个用户选择一个有利的卸载策略以最大化自身的效用，即

$$\max_{s_i} u(s_i, s_{-i}) = s_i^l u_i^l + s_i^m u_i^m + s_i^c u_i^c \tag{37}$$

然后，引入纳什均衡(nash equilibrium, NE)和势博弈来解决任务卸载策略博弈G。

定义1：对于卸载策略$S^* = \{S^{l*}, S^{m*}, S^{c*}\}$，如果对于$\forall i \in N$，都有

$$u(s_i^*, s_{-i}^*) \geqslant u(s_i, s_{-i}^*), s_i \in (s_i^l, s_i^m, s_i^c) \tag{38}$$

则称卸载策略S^*是博弈G的NE。NE是一个使系统稳定的状态，在NE点的任何用户都不能通过单方面改变其策略来进一步增加其效用。

定义2：如果博弈G承认一个势函数$\phi(s)$，当每个用户的卸载策略从s_i单方面地变为s_i'，且$s_{-i} \in \prod_{j \neq i} S_j$，$s_i, s_i' \in S$时，可以得到以下关系

$$u(s_i, s_{-i}) - u(s_i', s_{-i}) = \phi(s_i, s_{-i}) - \phi(s_i', s_{-i}), \forall i \in N \tag{39}$$

则博弈G是一个精确势博弈。每个具有有限策略集的精确势博弈都具有纳什均衡，并具备有限改进性(finite improvement property, FIP)，即任何异步更优响应的更新过程都必须是有限的，并导致NE。

引理1：博弈G是一个精确势博弈，其势函数$\phi(s)$如式(40)所示，并且总是收敛于NE，具有FIP。

Published by Journal of System Simulation, 2022

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

$$\phi(s)=s_i^l\sum_{n=1}^{N}u_n^l+(1-s_i^l)\cdot(s_i^m u_i^m+s_i^c u_i^c+\sum_{n=1,n\neq i}^{N}u_n^l),\forall i\in N \tag{40}$$

证明：对于用户 i $(\forall i\in N)$,$s_{-i}\in\prod_{j\neq i}S_j$,当用户将卸载策略从 s_i 更新到 s_i' 时，势函数 $\phi(s)$ 应满足式(40)。考虑以下三种情况：①$s_i=s_i^l=1$，$s_i'=s_i^c=1$；②$s_i=s_i^m=1$，$s_i'=s_i^l=1$；③$s_i=s_i^m=1$，$s_i'=s_i^c=1$。

(1) 用户 i 的卸载策略从本地处理改变为卸载到云服务器处理。此时有

$$\phi(s_i^l,s_{-i})-\phi(s_i^c,s_{-i})=\sum_{n=1}^{N}u_n^l-u_i^c-\sum_{n=1,n\neq i}^{N}u_n^l=u_n^l-u_i^c=u(s_i^l,s_{-i})-u(s_i^c,s_{-i})$$

(2) 用户 i 的卸载策略从卸载到MEC服务器处理改变为本地处理。此时有

$$\phi(s_i^m,s_{-i})-\phi(s_i^l,s_{-i})=u_i^m+\sum_{n=1,n\neq i}^{N}u_n^l-\sum_{n=1}^{N}u_n^l=u_n^m-u_i^l=u(s_i^m,s_{-i})-u(s_i^l,s_{-i})$$

(3) 用户 i 的卸载策略从卸载到MEC服务器改变为卸载到云服务器。此时有

$$\phi(s_i^m,s_{-i})-\phi(s_i^c,s_{-i})=u_i^m+\sum_{n=1,n\neq i}^{N}u_n^l-u_i^c-\sum_{n=1,n\neq i}^{N}u_n^l=u_n^m-u_i^c=u(s_i^m,s_{-i})-u(s_i^c,s_{-i})$$

可以看出，对于用户 i 的任何卸载决策的改变，势函数 $\phi(s)$ 总是满足式(39)。因此，博弈 G 是一个精确势博弈，总是存在NE。通过FIP得到最佳组合策略后，任何用户都没有单方面偏离的动机。

3.4 GDTOA算法

针对3.3节任务卸载策略问题，设计了博弈论的GDTOA。在最优资源分配下得到卸载策略，然后在给定的卸载策略下利用3.1和3.2节的方法优化资源分配，互相迭代这两个过程直到收敛，使系统效用最大化。

算法2给出了用户的卸载策略预处理过程。步骤1~4进行初始化，每个用户都卸载到MEC。步骤6~10先优化计算资源分配和上行功率分配，再计算每一个用户采取不同卸载策略时的效用。如果 $t_i^l>T_i$ 且 $t_i^c>T_i$，卸载到MEC服务器。否则，用户将采取使自身效用值最大的卸载策略。

算法 2: 任务卸载策略预处理

1.Initialization:

2.Userset$N=\{1,2,\cdots,N\}$,task$\tau_i=\{C_i,D_i,T_i\}$,$i\in N$

3. Each user$i,i\in$ Nset offloading strategy$s_i=s_i^m=1$

4.**End initialization**

5.**for each user**$i,i\in N$ do

6. Calculatet_i^l, $E_i^l$and$u_i^l$by (1), (2) and (11) respectively.

7. Calculate$f_i^{m^*}$and$f_i^{c^*}$by algorithm 1 and (30) respectively.

8. Update$p_i^*$by using uplink power allocation.

9. Calculatet_i^m, $E_i^m$and$u_i^m$by (4), (5) and (9) respectively.

10. Calculatet_i^c, $E_i^c$and$u_i^c$by (6), (7) and (10) respectively.

11. **if** $t_i^l>T_i$and $t_i^c>T_i$, then updates$s_i=s_i^m=1$.

12. **else**

13. **if**$u_i^m>\max(u_i^c,u_i^l)$, then updates$s_i=s_i^m=1$.

14. **else**

15. **if** $u_i^l>u_i^c$, then updates$s_i=s_i^l=1$.

16. **else**

17. updates$s_i=s_i^c=1$.

18. **end if**

19. **end if**

20. **end if**

21.**end for**

22. **Output:** Pre task offloading strategy $S_0=\{S_0^l,S_0^m,S_0^c\}$

采用GDTOA算法的目的是在计算任务执行之前，协调移动用户实现相互满意的卸载策略。算法的每次迭代包含以下两个过程：

(1) 用户效用计算：具体细节见算法2的步骤

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

Wu and Liao: Game-Based Resource Allocation and Task Offloading Scheme in Coll

6~10。在这个过程中，用户根据当前的卸载策略(s_i,s_{-i})，先优化资源分配，再计算每个用户采取不同卸载策略时对应的效用值$u(s_i^j,s_{-i}),j\in$ A。

(2) 策略更新竞争：利用G的FIP，每轮迭代

$$\Delta_i(t)\triangleq\{s_i^j:u(s_i^*,s_{-i}(t))>u(s_i(t),s_{-i}(t))\}$$

$$\triangleq\begin{cases}s_i'=s_i^1=1,\text{if}\,s_i(t)\neq(s_i^1=1)\text{ and }u(s_i^l,s_{-i}(t))>u(s_i^j,s_{-i}(t)),j\in\{m,c\}\\ s_i'=s_i^{\mathrm{m}}=1,\text{if}\,s_i(t)\neq(s_i^{\mathrm{m}}=1)\text{ and }u(s_i^m,s_{-i}(t))>u(s_i^j,s_{-i}(t)),j\in\{l,c\}\\ s_i'=s_i^{\mathrm{c}}=1,\text{if}\,s_i(t)\neq(s_i^{\mathrm{c}}=1)\text{ and }u(s_i^c,s_{-i}(t))>u(s_i^j,s_{-i}(t)),j\in\{l,m\}\\ \phi,\text{otherwise}\end{cases}$$

如果$\Delta_i(t)\neq\phi$（即用户i可以通过更新策略增加自身效用），用户i将参与此轮迭代的竞争，否则它将在此轮迭代中保持原卸载策略。用户在竞争中获胜的条件是拥有最大的效用改进，即拥有最大的$\Delta u(s_i)=u(s_i',s_{-i})-u(s_i,s_{-i})$。获胜者可以更新令自己满意的卸载策略，而其他的所有用户继续保持原卸载策略，即$s_i(t)=s_i(t-1)$，等待下一轮决策更新竞争。直到没有用户有动机改变其卸载策略时，迭代过程就会终止。具体流程如算法3所示。

算法3: GDTOA

Initialization:

User set $N=\{1,2,\cdots,N\}$,

task $\tau_i=\{C_i,D_i,T_i\}$,$i\in N$,iteration $t=0$.

Obtain initial task offloading strategy $S_0=\{S_0^l,S_0^m,S_0^c\}$

according to Algorithm 2.

$t\leftarrow t+1$

while $S(t-1)\neq S_0$ do

$S_0=S(t-1)$,set $i=1$.

while $i\leqslant N$ do

calculate$u'(i)$=$u(s_i^j,s_{-i}(t-1)),j\in\{l,m,c\}$

compute the best response$\Delta_i(t)$.

$i\leftarrow i+1$

end while

for each user*i*,$i\in N$ do

if user*i*wins in the t-th iteration ,

then update$s_i(t)=s_i'$.

else $s_i(t)=s_i(t-1)$

end for

$t\leftarrow t+1$

end while

Output: Optimal computation resource allocation F^*, uplink power allocation P^* and offloading strategy S^*

仅让一个用户更新卸载策略。能提高效用的用户以分布式的方式竞争策略更新的机会，每个用户i根据效用值大小的比较计算其最佳响应集：

4 实验结果与分析

4.1 实验设置

本文的仿真实验参考文献[12]和[22]设置无线通信和计算相关参数。云边系统包括1个云服务器、1个MEC服务器、1个BS和N个用户。N从[5,100]中随机选取，用户随机分布在距基站半径为0~200 m的覆盖范围内。上行信道增益由路径损耗模型$L(\text{dB})=140.7+36.7\lg d_{[\text{km}]}$生成，对数正态阴影标准差设为8 dB。其他部分参数如表1所示。

本文考虑将人脸识别或心率测量应用作为用户的计算任务。类似于文献[23]，输入任务数据量D_i从[100,500]kB的范围内随机选择，所需CPU周期数为1 GHz，用户i的时延要求T_i遵循[1,2]s内的均匀分布。时间和能量偏好因子分别设置为$\beta_i^t=0.6$和$\beta_i^e=0.3$。

仿真实验在装有i5-10210U CPU@1.6 GHz 2.11 GHz的Windos10 PC上运行，运行平台是JDK1.8。

Published by Journal of System Simulation, 2022

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

表1 实验参数
Table 1 Experimental parameters

参数	数值
系统带宽B/MHz	20
用户最大上行发射功率p^{max}/dBm	23
背景噪声功率σ^2/dBm	-100
从BS到云的单位数据传输时延v/μs/bit^{-1}	1
能量系数κ	5×10^{-27}
MEC服务器的CPU频率F/GHz	20
用户设备的CPU频率f_i^l/GHz	1
MEC服务器的计算资源成本/\$/GHz^{-1}	0.015
云服务器的计算资源成本/\$/GHz^{-1}	0.03

将以下方案作为GRATO的对比方案：

(1) 穷举法(exhaustive)：通过遍历3^n个决策组合以找到最优的卸载方案，时间复杂度非常高，因此只在用户数量较少时,使用此方案作为对比。

(2) 多目标计算卸载和资源分配(multi objective computation offloading and resource allocation, MCORA)方案：采用文献[19]中的卸载决策算法，计算任务可以选择本地处理或卸载到MEC。与GRATO方案相比，MCORA方案不涉及云计算处理模型。

(3) 分布式计算卸载和资源分配(distributed computation offloading and resource allocation , DCORA)方案：采用文献[22]中基于博弈论的卸载算法DCORA，不考虑能耗，资源分配和功率控制方法和GRATO方案相同。比较卸载决策算法的性能。

(4) 联合任务卸载和资源分配(joint task offloading and resource allocation, JTORA)方案：采用文献[12]中的功率控制方法，资源分配和卸载决策和GRATO方案相同。比较功率控制对能耗的影响。

(5) 计算卸载策略优化(computation offloading strategy optimization, COSO)方案：不进行计算资源分配优化，MEC计算资源平均分配，云计算资源固定，只对卸载决策进行优化(采用本文提出的GDTOA算法)。

4.2 GRATO方案的次优性和收敛性

图2比较了$N=10$时所有方案在不同计算负载下的系统效用值。可以看出，穷举法的性能最优，GRATO方案与穷举法的性能接近，并且明显优于其他方案。此外，随着计算负载增加，系统效用降低，这是因为任务所需计算量增加会造成更大的时延和能耗。GRATO方案相较于穷举法有约3%的性能损耗，相对于JTORA，MCORA，DCORA和COSO方案分别有16%，33%，7.1%和3.6%的性能提升。

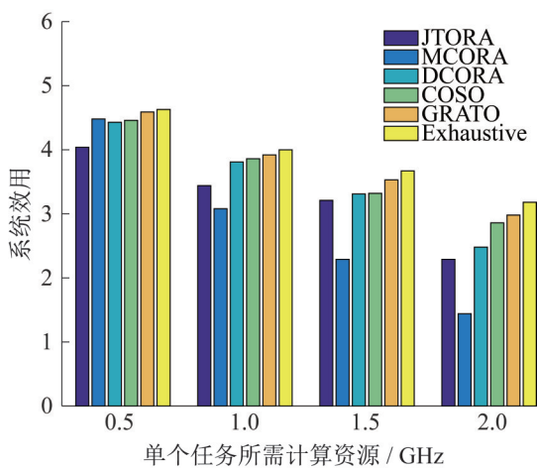


图2 系统效用的比较
Fig. 2 Comparison of system utility

各方案的程序运行时间如表2所示。穷举法的运行时间远超其他方案，当$N=10$时，穷举法的运行时间是GRATO方案的近200倍。虽然穷举法可以得到最优解，但运行时间不可接受。GRATO方案的运行时间虽然高于其他方案，但维持在毫秒级别，与边缘计算任务执行的总时间(秒级别)相比，可以忽略不计。因此，GRATO方案在性能和运行时间的综合考虑上是较为实际的次优解方案。

图3给出了$N=20$时GRATO方案在MEC服务器不同计算资源量下的收敛性。系统效用值随着迭代次数逐渐增加，最后趋于稳定。GRATO方案每一次迭代都优化资源分配和卸载策略，增加系统效用，

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

第 34 卷第 7 期 系统仿真学报 Vol. 34 No. 7
2022 年 7 月 Journal of System Simulation Jul. 2022

直到系统达到NE。此外，MEC计算资源越多，系统效用值越高，且迭代次数越少，收敛越快。

表2 各方案的运行时间
Table 2 Runtime of schemes ms

方案	$N=5$	$N=10$
MCORA	0.096±0.02	0.22±0.07
COSO	0.63±0.2	0.77±0.18
DCORA	0.71±0.18	0.95±0.32
JTORA	0.93±0.2	2.51±0.4
GRATO	3.26±0.37	11.8±2.5
Exhaustive	32.2±3.9	2 543±36

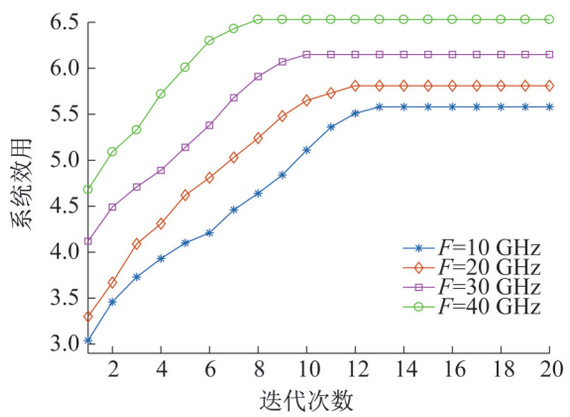


图3 GRATO算法的收敛性
Fig. 3 Convergence of the GRATO algorithm.

4.3 GRATO方案的性能

图4给出了各方案在不同用户数量下的系统效用比较。随着用户数量的增加，各方案的系统效用值增加。在所有方案中，GRATO的性能最好。这是因为GRATO方案能更好地利用MEC和云计算资源，共同优化计算资源分配、功率控制和卸载策略。MCORA方案由于没有云计算辅助，只能将任务本地处理或卸载到MEC。MEC服务器上的计算资源有限，随着用户数量的增加，分配给每个用户的计算资源减少，MCORA在本地处理的任务数量变多，因此系统效用最低。

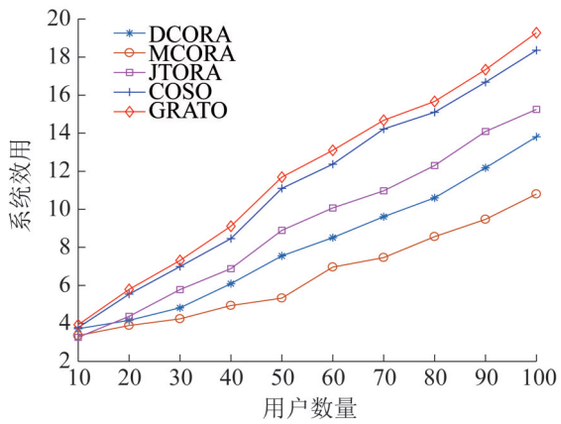


图4 不同用户数量下的系统效用
Fig. 4 System utility against different number of users

图5给出了$N=30$时各方案在不同计算负载下的平均任务时延。随着计算负载的增加，任务时延增大。在所有方案中，GRATO的时延最小，其次是COSO，这是因为COSO方案未对计算资源分配进行优化。MCORA的时延最大，因为它只能将任务卸载到资源受限的MEC服务器。类似地，DCORA的卸载决策算法也更倾向于卸载到MEC。这说明在MEC计算资源紧张时，将任务卸载到云可以显著地减少任务时延。GRATO方案有效地利用MEC和云计算资源，减少了任务时延。

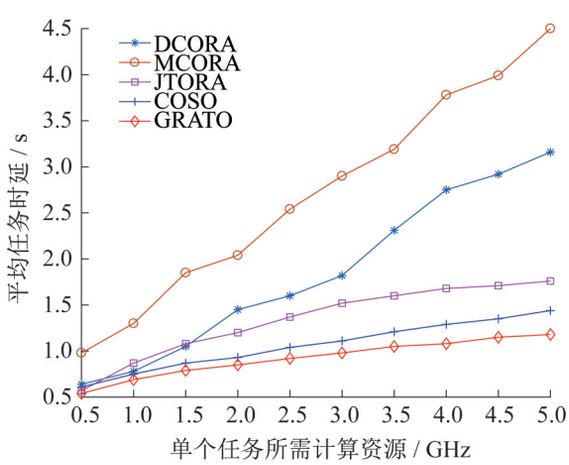


图5 不同计算负载下的平均任务时延
Fig. 5 Average task delay against computation load

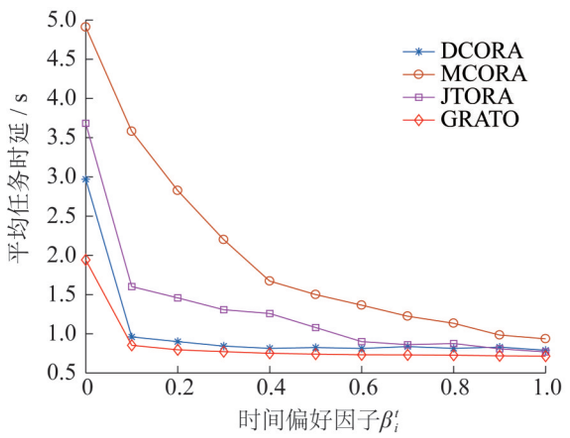
4.4 偏好因子和成本系数的影响

图6(a)给出了$N=30$时平均任务时延和时间偏好因子β_i^t的关系。β_i^t从0~1变化，相应地，$\beta_i^e=$

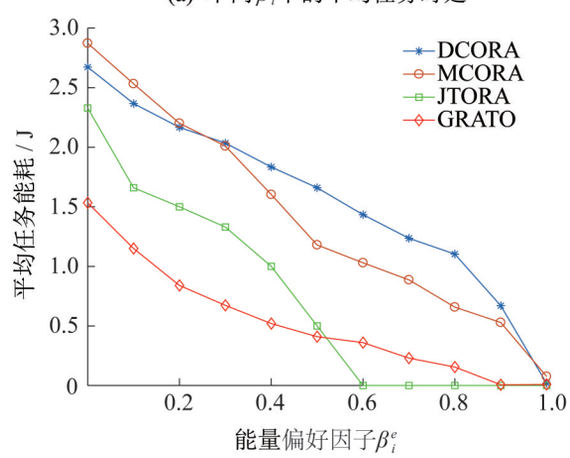
Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

$1-\beta_i^t$。所有方案的时延随着β_i^t的增大而减小。当$\beta_i^t \geqslant 0.6$后，平均时延降低到一定水平后收敛。GRATO方案的时延一直保持最小。

图6(b)给出了$N=30$时平均任务能耗和能量偏好因子β_i^e的关系。各方案的能耗随着β_i^e的增大而减小。在$\beta_i^e \leqslant 0.5$时，GRATO方案的能耗最小。而JTORA方案在$\beta_i^e=0.6$就收敛到接近0的水平，它的功率控制方法使用户功率保持在较小的范围内，更适用于对能耗要求高的场景。GRATO方案在时延和能耗之间做了一定的权衡，在保证低时延的前提下，降低用户设备的能耗。



(a) 不同β_i^t下的平均任务时延



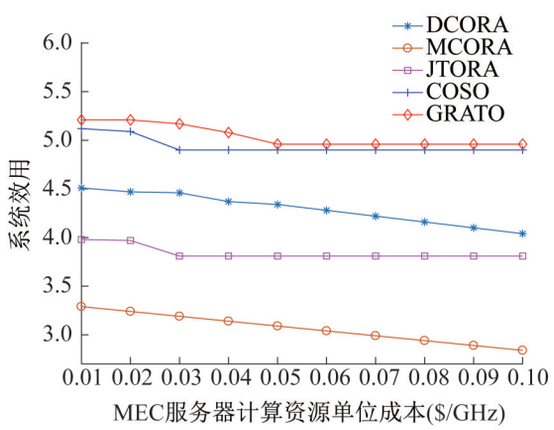
(b) 不同β_i^e下的平均任务能耗

图6 平均时延及能耗与偏好因子的关系

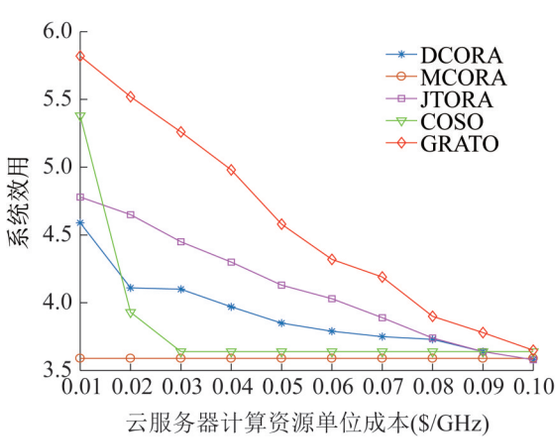
Fig. 6 Delay and energy consumption versus preference factors

图7(a)给出了$N=30$时系统效用与MEC计算资源成本之间的关系。随着成本升高，GRATO，DCORA，JTORA和COSO这几种方案的效用都是先下降再趋于稳定，这是因为随着MEC计算资源成本的增加，更多用户会选择将任务卸载到云。而MCORA方案无法将任务卸载到云，只能购买高价的MEC计算资源，导致效用越来越低。

图7(b)给出了$N=30$时系统效用与云计算资源的成本之间的关系。MCORA方案不涉及到云，效用保持不变。其他方案都随着成本升高而效用下降，最终收敛到和MCORA类似的水平。这是因为用户为了获得更高的效用，不愿意购买高成本的云计算资源，而是选择本地处理或卸载到MEC。下降最快的是COSO方案，它没有对计算资源分配进行优化，云服务器为其任务分配固定的计算资源。可见在计算资源成本增加的情况下，GRATO方案能通过优化计算资源分配获得较高的性能。



(a) 不同MEC计算资源单位成本下的系统效用



(b) 不同云计算资源单位成本下的系统效用

图7 系统效用与计算资源成本的关系

Fig. 7 System utility versus the cost of computing resource

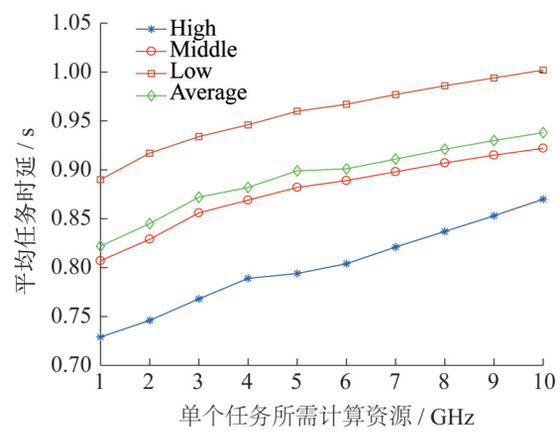
https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077

第 34 卷第 7 期 系统仿真学报 Vol. 34 No. 7
2022 年 7 月 Journal of System Simulation Jul. 2022

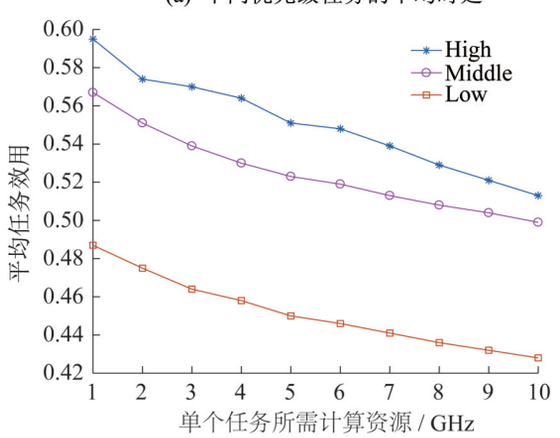
4.5 任务优先级的影响

根据优先级系数λ_i的值，将不同的任务划分为三个优先级，$0.8 < \lambda_i \leqslant 1$为高优先级，$0.3 < \lambda_i \leqslant 0.8$为中优先级，$0.1 < \lambda_i \leqslant 0.3$为低优先级。将用户数量设置为$N=30$，输入数据量固定为平均值300 kB。

图8(a)给出了三种优先级的任务在不同计算负载下的时延。随着任务计算负载的增加，三种优先级任务的时延增加，其中高优先级任务的时延最低，低优先级任务的时延最高。系统平均时延大于高优先级任务，说明高优先级任务时延的减少是以增加低优先级任务时延为代价。图8(b)给出了三种优先级的任务在不同计算负载下的平均任务效用，高优先级任务的效用值最高。这说明，GRATO方案不仅能保证较低的系统时延，还能区分任务的优先级，使紧急的用户能更迅速地完成计算任务。



(a) 不同优先级任务的平均时延



(b) 不同优先级任务的平均效用

图8 不同优先级任务的性能比较

Fig. 8 Comparison of performance of different priorities

5 结论

为了更好地综合利用云和MEC服务器的计算资源，本文针对云边协同的系统模型，构建了一个系统效用最大化问题。由于该问题是MINLP问题，很难得到最优解,于是将其分解为三个子问题，并提出了一种GRATO方案。利用KKT条件求解计算资源分配问题，然后设计了一种离散化的上行功率分配策略。针对任务卸载策略优化问题，提出了一种GDTOA算法，在多项式时间内求该问题的次优解，在保证较优性能的情况下，降低了时间复杂度。仿真结果表明，在时延、能耗和系统效用方面，GRATO方案均具有良好的性能，且优先级更高的用户具有更高的性能。本文的研究基于用户的静态场景，而实际上用户在计算卸载或资源分配时期可能动态地离开系统，后续的研究中需要考虑用户的移动性，以提升GRATO方案的性能。

参考文献：

[1] Tao Xiaoyi, Kaoru Ota, Dong mianxiong, et al. Performance Guaranteed Computation Offloading for Mobile-edge Cloud Computing[J]. IEEE Wireless Communications Letters (S2162-2337), 2017, 6(6): 774-777.

[2] Zhang Guowei, Fei Shen, Zhang Yueyue, et al. Delay Minimized Task Scheduling in Fog-enabled IoT Networks[C]// International Conference on Wireless Communications and Signal Processing (WCSP). IEEE, 2018, 10:1-6.

[3] Hu Yun Chao, Milan Patel, Dario Sabella, et al. Mobile Edge Computing—A key technology towards 5G[R]. ETSI white paper, 2015, 11(11): 1-16.

[4] Tran Tuyen X, Abolfazl Hajisami, parul pandey, et al. Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges[J]. IEEE Communications Magazine (S0163-6804), 2017, 55(4): 54-61.

[5] Lei Yang, Cao Jiannong, Cheng Hui, et al. Multi-user Computation Partitioning for Latency Sensitive Mobile Cloud Applications[J]. IEEE Transactions on Computers (S0018-9340), 2015, 64(8): 2253-2266.

[6] Cardellini Valeria, Vittoria De Nitto Personé, Valero

Journal of System Simulation, Vol. 34 [2022], Iss. 7, Art. 9

Divacerio, et al. A Game-theoretic Approach to Computation Offloading in Mobile Cloud Computing[J]. Math Program (S0025-5610), 2016, 157(2): 421-449.

[7] You Changsheng, Kaibin Huang, Chae Hyukjin, et al. Energy-efficient Resource Allocation for Mobile-edge Computation Offloading[J]. IEEE Transactions on Wireless Communications (S1536-1276), 2016, 16(3): 1397-1411.

[8] Rahimi M Reza, Nalini Venkatasubramanian, Av vasilakos, et al. Music: Mobility-aware Optimal Service Allocation in Mobile Cloud Computing[C]// IEEE Sixth International Conference on cloud computing. IEEE, 2013: 75-82.

[9] Xiao Surong, Liu Chubo, Li Kenli, et al. System Delay Optimization for Mobile Edge Computing[J]. Future Generation Computer Systems (S0167-739X), 2020 (109): 17-28.

[10] Huynh Luan NT, Quoc-Viet Pham, Xuan-Qui Pham, et al. Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach[J]. Applied Sciences (S2076-3417), 2020(10): 203.

[11] X. Lyu, H Tian, P Zhang, et al. Multi-user Joint Task Offloading and Resources Optimization in Proximate Clouds[J]. IEEE Transactions on Vehicular Technology (S0018-9545), 2017, 66(4): 3435–3447.

[12] Tran Tuyen X, Dario Pompili. Joint Task Offloading and Resource Allocation for Multi-server Mobile-edge Computing Networks[J]. IEEE Transactions on Vehicular Technology (S0018-9545), 2019, 68(1): 856-868.

[13] Ren Jinke, Yu Guanding, Cai Yunlong, et al. Latency optimization for Resource Allocation in Mobile-edge Computation Offloading[J]. IEEE Transactions on Wireless Communications (S1536-1276), 2018, 17(8): 5506-5519.

[14] Gu Yunan, Zheng Chang, Miao Pan, et al. Joint radio and computational resource allocation in IoT fog computing [J]. IEEE Transactions on Vehicular Technology (S0018-9545), 2018, 67(8): 7475-7484.

[15] Ma S, Guo S, Wang K, et al. A cyclic game for service-oriented resource allocation in edge computing[J]. IEEE Transactions on Services Computing (S1939-1374), 2020, 13(4): 723-734.

[16] Chen Yifan, Zhi Y, Yang B, et al. A Stackelberg Game Approach to Multiple Resources Allocation and Pricing in Mobile Edge Computing[J]. Future Generation Computer Systems (S0167-739X), 2020, 108: 273-287.

[17] Chen Xu. Decentralized Computation Offloading Game for Mobile Cloud Computing[J]. IEEE Transactions on Parallel and Distributed Systems (S1045-9219), 2014, 26(4): 974-983.

[18] Hu Junyan, Li kenli, Liu chubo, et al. Game-Based Task Offloading of Multiple Mobile Devices with QoS in Mobile Edge Computing Systems of Limited Computation Capacity[J]. ACM Transactions on Embedded Computing Systems (S1539-9087), 2020, 19 (4): 1-21.

[19] Long Long, Liu zichen, Zhou Yiqing, et al. Delay Optimized Computation Offloading and Resource Allocation for Mobile Edge Computing[C]// IEEE 90th Vehicular Technology Conference (VTC2019-Fall), IEEE, 2019:1-5.

[20] Misra S, Sarkar S. Priorty-based Time-slot Allocation in Wireless Body Area Networks during Medical Emergency Situations: An Evolutionany Game-Thearetic Perspective[J]. IEEE Journal of Biamed and Health Informatios, 2015, 19(2): 541-548.

[21] K. Tammer. The Application of Parametric Optimization and Imbedding to the Foundation and Realization of a Generalized Primal Decomposition Approach[J]. Mathematical research (S0138-3019), 1987, 35: 376-386.

[22] Zhao Junhui, Li Qiuping, Gong Yi, et al. Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks[J]. IEEE Transactions on Vehicular Technology (S0018-9545), 2019, 68(8): 7944-7956.

[23] Ren J, Yu G, He Y, et al. Collaborative Cloud and Edge Computing for Latency Minimization[J]. IEEE Transactions on Vehicular Technology (S0018-9545), 2019, 68(5): 5031-5044.

https://dc-china-simulation.researchcommons.org/journal/vol34/iss7/9
DOI: 10.16182/j.issn1004731x.joss.21-0077