移动边缘计算任务切分与最优卸载算法设计

路静，李晗琳，高林
（哈尔滨工业大学（深圳）电子与信息工程学院，广东 深圳 518055）

摘　要：移动边缘计算（MEC, mobile edge computing）作为将计算基础设施从远程云数据中心推向边缘设备的新架构模式，为满足物联网（IoT, Internet of things）应用时延敏感、计算密集等需求提供了新方案。针对可切分任务在多用户多 MEC 服务器系统中的任务卸载与调度问题进行研究，每个用户任务均可切分为多个相互关联的子任务，且子任务均可在本地执行或被卸载到某 MEC 服务器执行，系统通过对子任务的卸载和调度决策来提高网络性能。使用用户体验（QoE, quality of experience）和用户间公平性来表征网络性能，将优化问题建模为一个可切分任务卸载和调度（J-DTOS, joint dependent task offloading and scheduling）优化问题。该问题是一个 NP-hard 非线性混合整数规划问题，因此，所提方案进一步通过引入中间变量重新构造了原问题，并基于此提出了一个近似最优解。仿真结果表明，所提的卸载和调度策略能显著提高系统的性能。

关键词：物联网；移动边缘计算；可切分任务卸载
中图分类号：TN929.5
文献标识码：A
doi: 10.11959/j.issn.2096−3750.2020.00199

Design of task dividing and offloading algorithm in mobile edge computing

LU Jing, LI Hanlin, GAO Lin
School of Electronic and Information Engineering, Harbin Institute of Technology(Shenzhen), Shenzhen 518055, China

Abstract: Mobile edge computing (MEC) emerges as a new paradigm that pushes the computing infrastructure from the remote cloud data center to the edge equipments. It provides a new solution to meet the delay sensitive and computing intensive requirements of Internet of things (IoT). In this work, the problem of tasks offloading and scheduling in the multi-user and multi-server MEC system was considered. Specifically, each user had a task-dependent application and the tasks could be either executed locally or remotely according to the dependence. Thus, the network performance was improved by unloading and scheduling the sub tasks. Quality of experience (QoE) and fairness between users were used to characterize the network performance, and the optimization problem was modeled as a joint dependent task offloading and scheduling (J-DTOS) problem. The J-DTOS problem was a non-linear mixed integer programming, which was NP-hard in general. The original problem was reformulated by introducing intermediate variables and proposing a near-optimal solution. Simulation results show that the proposed offloading and scheduling design can significantly improve the performance of the system.

Key words: Internet of things, mobile edge computing, dependent task offloading

1　引言

随着 IoT 的快速发展，IoT 移动设备承载了更多计算密集型实时应用，如人脸识别、在线游戏、增强现实/虚拟现实（AR/VR, augmented reality/virtual reality）服务等。然而，移动设备因其资

收稿日期：2020−06−28；**修回日期：**2020−09−26
基金项目：国家自然科学基金资助项目（No. 61972113）
Foundation Item: The National Natural Science Foundation of China (No.61972113)

源的有限性，通常无法在不降低用户实时性体验的前提下高效地执行计算密集型任务。此外，移动云计算（MCC, mobile cloud computing）作为传统的任务计算卸载模式，其用户与计算中心之间存在长距离的数据传输，这将导致严重的回程拥塞，最终会使用户的 QoE 降低。在此情况下，MEC 将计算量卸载到网络边缘设备，为满足时延敏感、计算密集型移动端应用的需求提供了新思路[1]。

在 MEC 中，移动用户可以将计算任务卸载到附近承载相应应用程序的边缘服务器（部署在基站或网络接入点）上执行，在执行完成卸载的任务后，MEC 服务器会将执行结果反馈给移动用户。此外，不同 MEC 服务器可以通过回程链路连接将它们接收的任务彼此迁移，以平衡每个 MEC 服务器的工作负载。

关于边缘计算已有很多优秀的研究成果，其中大部分研究主要关注任务卸载[2-3]和调度[4]问题。如文献[2-3]主要关注任务卸载中的通信和缓存问题，没有考虑任务卸载后的调度问题；文献[4]重点研究了可打断任务的在线调度问题，没有考虑用户侧的任务卸载问题。本文的研究重点是联合任务卸载和调度问题，对这方面的研究目前较少[5-11]，具体从以下维度做划分。

1) 首先，本文将考虑任务的可切分性，以往研究中的任务切分模型通常可分为 3 种类型，具体如下。

① 0-1 卸载：用户需要将任务全部卸载到服务器或全部在本地执行[5]。

② 部分卸载：用户可以在不受任何约束的情况下粗略地按比例卸载应用程序的一部分[6]。

③ 可切分关联任务卸载：将任务可切分为多个相互关联的子任务，用户可以选择卸载应用程序中的每个子任务，且各个子任务的卸载和执行顺序受子任务依赖关系图的约束[7-11]。

显然，可切分关联任务模型更符合真实场景下的任务卸载需要。因而，本文将构建一个通用的子任务依赖模型，用于可切分关联任务的建模。本文所提模型具备一定的通用性，可概括顺序依赖、并行依赖、随机分层依赖和随机扇入/扇出依赖等现有研究常用的任务切分方式[7]。

2) 其次，在可切分关联任务卸载调度的研究中，服务器和用户的数量会对系统产生显著的影响。具体来说，文献[7-8]考虑了由单个用户和远程云服务器组成的系统；文献[9]将单个用户的计算中央处理器（CPU, central processing unit）模型从单核扩展到多核；文献[10]考虑了具有多个用户和一个远程服务器的系统，但该系统只考虑单个服务器；文献[11]研究了具有一个用户和多个服务器的系统，没有考虑多个用户的场景。以上研究均没有考虑最具普适性的多用户多服务器系统中的任务卸载问题，本文对此问题进行了研究，并将其建模为一个 J-DTOS 问题。在这个问题中，需要解决的关键问题包括如下方面。

① 子任务应在本地执行还是被卸载到 MEC 服务器执行。

② 如果卸载，应将子任务卸载到哪个服务器执行。

③ 子任务应该在什么时间开始执行。

针对以上问题，本文构造了一个混合整数优化问题，并将其转化为可由 MATLAB 的 CVX 软件工具包求解的形式。本文的主要贡献概括如下。

① 研究了可切分任务在边缘计算中的任务卸载与调度问题，通过对异构任务中的子任务（具有不同的拓扑图、时延敏感度和完成期限）进行卸载调度来提高用户体验质量。

② 在边缘计算卸载调度中考虑用户公平性，引入了新的系统目标，促使所有用户的卸载工作负载比例彼此接近。

③ 问题分析与重构，将任务卸载与调度问题建模为一个非线性混合整数优化问题。通过引入多个中间变量，将原 NP-hard 问题转化为线性混合整数规划问题。

④ 任务切分普适性，使用随机生成的任务关联拓扑图表示可切分任务，仿真结果表明，与本地计算和 0-1 卸载方案相比，本文方案的计算卸载设计使 QoE 得到显著提高。

2　系统模型

2.1　网络模型

网络模型如图 1 所示，本文考虑一个包含 u 个用户 $U=\{1,\cdots,u\}$ 和 p 个 MEC 服务器 $P=\{1,\cdots,p\}$ 的 MEC 系统。其中，每个用户 $u\in U$ 持有一个任务 n_u。每个任务 n_u 都可以被拆分成 i 个有相互承接关系的子任务 $I_{n_u}=\{1_{n_u},\cdots,i_{n_u}\}$，每个子任务 i_{n_u} 都可以在本地或者被卸载到某个 MEC 服务器上执行。移动用户可以和 MEC 服务器建立无线连接，通过无线链路将子任务从移动端卸载到 MEC 服务器，服

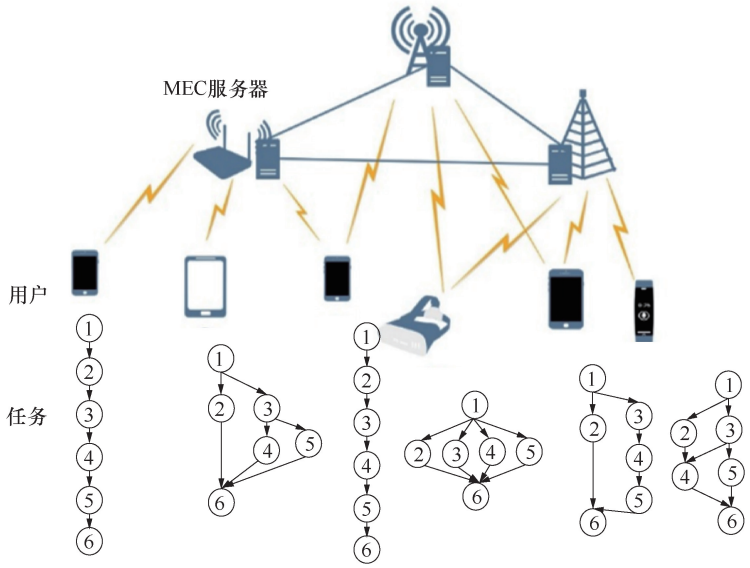


图 1　网络模型

务器可以将计算结果传回移动端。此外，任意两个 MEC 服务器之间通过有线连接进行数据传输。

2.2　任务模型

任一用户 $u \in U$ 持有一个任务 n_u，如果单个用户持有多个任务，可以将用户拆分为多个虚拟的用户，每个用户只持有一个任务。所有任务均可拆分成 I 个有相互承接关系的子任务，其承接关系可以用一个有向无环图（DAG, directed acyclic graph）表示，不同任务的子任务数量相同，承接关系不同。子任务 $i_{n_u} \in I_{n_u}$ 的计算量可以表示为 $D_{n_u,i}$，即 CPU 执行子任务需要的总转数，DAG 表示的任务模型如图 2 所示。

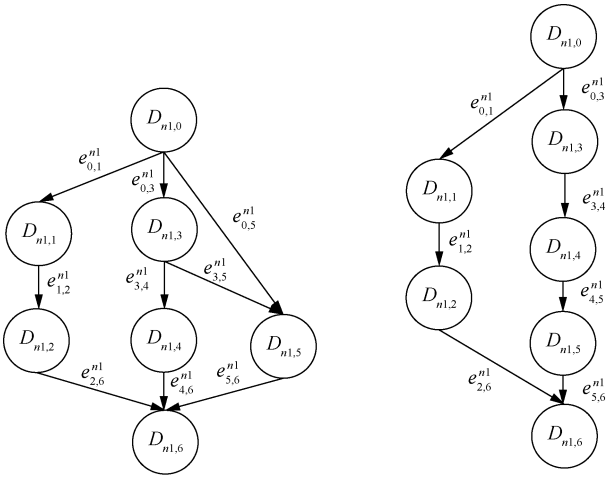


图 2　DAG 表示的任务模型

子任务间的承接关系用 $o_{j,i}^{n_u}$ 刻画。如果用户 u 的子任务 i_{n_u} 承接在子任务 j_{n_u} 之后，则 $o_{j,i}^{n_u}=1$，表示子任务 i_{n_u} 只有在子任务 j_{n_u} 执行完成并且接收其执行结果后才能开始执行；反之，如果 $o_{j,i}^{n_u}=0$，则表示两个子任务之间没有执行顺序的限制。

子任务 i_{n_u}、j_{n_u} 间的数据传输量为 $e_{j,i}^{n_u}$：如果两个子任务之间有承接关系，那么当这两个子任务的执行位置不同时，相应的数据 $e_{j,i}^{n_u}$ 需要在前一个子任务完成计算后从其执行位置传输到后一个子任务的执行位置上，当传输完成后，后一个子任务才能开始执行。

为了方便优化问题的描述，在任务的承接关系的开始和结尾各插入一个虚拟的子任务 $i_{n_u}^0$ 和 $i_{n_u}^{I+1}$，其计算量为零，并且必须在本地执行。其通信数据量为原任务的最初输入数据量和最终输出数据量，即开头的虚拟子任务与原任务中的第一批任务之间存在数据传输，结尾的虚拟子任务和最后一批任务之间存在数据传输。

每个任务都有起始时间与截止时间。用户 u 的任务 n_u 在 $T_{n_u}^{\mathrm{st}}$ 时产生，在 $T_{n_u}^{\mathrm{ft}}$ 时截止。

2.3　计算模型

利用 CPU 转速来刻画本地设备和 MEC 服务器的计算频率。本地用户 $u \in U$ 的 CPU 转速表示为 f_u，则用户 u 的任务 n_u 的子任务 i_{n_u} 在本地执行所需要的时间为 $\dfrac{D_{n_u,i}}{f_u}$。此外，本地执行任务的功率即计算消耗的单位时间能耗为 c_u^{cpt}。

与本地计算模型相似，MEC 服务器 $p \in P$ 的

CPU 转速为 f_p，则子任务 i_{n_u} 在 MEC 服务器 p 上执行所需要的时间为 $\frac{D_{n_u,i}}{f_p}$。由于移动端使用电池时对能耗更敏感，因此本文主要考虑本地用户端的能耗，不考虑 MEC 服务器端的能耗。

为了刻画任务在本地执行或被卸载到某个服务器上执行的情况，使用 $p \in \{0\} \cup P$ 表示任务执行的地点。$p=0$ 表示任务在本地执行，$p \in P$ 表示任务在 MEC 服务器 p 上执行，则整合后用户 u 的子任务 i_{n_u} 在 MEC 服务器 p 上执行所需要的时间可以表示为

$$T_{u,i,p}^{\mathrm{EXE}} = \begin{cases} \dfrac{D_{n_u,i}}{f_u}, p=0 \\ \dfrac{D_{n_u,i}}{f_p}, p \in P \end{cases} \tag{1}$$

2.4　传输模型

在移动用户与MEC服务器的数据传输过程中，用户 $u \in U$ 到 MEC 服务器 $p \in P$ 的传输速率为 $r_{u,p}$，无线传输的功率为 $c_{u,p}^{\mathrm{trs}}$。根据香农公式可得传输速率为

$$r_{u,p} = \omega \log\log\left(1 + \frac{h_{u,p} c_{u,p}^{\mathrm{trs}}}{\sigma^2}\right) \tag{2}$$

其中，ω 为无线带宽，$h_{u,p}$ 为信道增益。假设用户与 MEC 服务器之间的无线连接稳定，如果用户 u 的任务 n_u 的子任务 j_{n_u} 在本地执行，子任务 i_{n_u} 在 MEC 服务器 p 上执行，则其数据传输时间为 $\frac{e_{j,i}^{n_u}}{r_{u,p}}$。

在 MEC 各服务器之间的数据传输过程中，因为是有线传输，所以可以简单地将两个 MEC 服务器 q、$p(q,p \in P)$ 之间的数据传输速率表示为 $r_{q,p}$，则用户 u 的关联子任务 j_{n_u} 与 i_{n_u} 之间的传输时间为 $\frac{e_{j,i}^{n_u}}{r_{q,p}}$。

将本地与服务器、服务器与服务器之间的数据传输情况刻画为：如果两个关联子任务在同一位置执行，则它们之间的数据传输量和传输时间为零；如果两个关联子任务在不同位置执行，则它们之间需要进行数据传输。关联子任务 j_{n_u}、i_{n_u} 从 MEC 服务器 q 到 MEC 服务器 p 的数据传输时间可以表示为

$$T_{u,j,i,q,p}^{\mathrm{TRS}} = \begin{cases} 0, q=p \\ \dfrac{e_{j,i}^{n_u}}{r_{u,p}}, q=0 \text{或} p=0 \\ \dfrac{e_{j,i}^{n_u}}{r_{q,p}}, q,p \in P \end{cases} \tag{3}$$

3　子任务卸载与调度问题构建

3.1　决策变量

首先，定义子任务卸载决策变量 $x_{u,i,p} \in \{0,1\}$，表示每个用户的每个子任务执行的位置，如果用户 $u \in U$ 的子任务 $i_{n_u} \in I_{n_u}$ 在 MEC 服务器 $p \in P \cup \{0\}$ 上执行，则 $x_{u,i,p}=1$；否则 $x_{u,i,p}=0$。每个子任务只能选择在一个位置执行，其中，$p=0$ 表示子任务在移动设备本地执行，$p \in P$ 表示子任务被卸载到服务器端执行。其中，$x_{u,i,p} \in \{0,1\}$，$\forall u \in U$，$i \in I_{n_u}$，$p \in P \cup \{0\}$。

然后，定义子任务执行时间决策变量 $t_{u,i}$。$t_{u,i}$ 表示子任务 i_{n_u} 开始执行的时刻，即子任务在 $t_{u,i}$ 开始执行，并持续占用 CPU 直到执行完成。即 $t_{u,i} \in R$，$\forall u \in U$，$i \in I_{n_u}$。

基于上述卸载调度决策和执行时间决策，可以将每个子任务的实际执行时长和两个子任务间的实际数据传输时长刻画出来。用户 u 的子任务 i_{n_u} 的执行总时长可以表示为

$$T_{u,i}^{\mathrm{EXE}} = \sum_{p=0}^{P} x_{u,i,p} T_{u,i,p}^{\mathrm{EXE}}, \forall u \in U, \forall i \in I_{n_u} \tag{4}$$

任务的执行是独占 CPU 且连续不可打断的，任务执行结束的时间可以表示为

$$T_{u,i}^{\mathrm{FIN}} = t_{u,i} + T_{u,i}^{\mathrm{EXE}}, \forall u \in U, \forall i \in I_{n_u} \tag{5}$$

此外，如果两个子任务在不同位置执行，则它们之间需要存在数据传输。用户 u 的子任务 j_{n_u} 到子任务 i_{n_u} 的传输时间可以表示为

$$T_{u,j,i}^{\mathrm{TRS}} = \sum_{q=0}^{P} \sum_{p=0}^{P} x_{u,j,q} x_{u,i,p} T_{u,j,i,q,p}^{\mathrm{TRS}}, \quad \forall u \in U, \forall j,i \in I_{n_u} \tag{6}$$

3.2　限制条件

任务卸载和调度问题的限制条件主要分为 4 个部分：1) 每个任务的各个子任务的执行顺序必须符合任务 DAG 的要求；2) 本地和服务器端在同一时

间只能处理一个任务，即任务独自占有 CPU；3) 整个任务的执行时间要符合开始时间和截止时间的要求；4) 每个子任务只能在同一个位置执行。下面分别表述上述 4 个限制条件。

1) 在任务承接关系 DAG 中，对于同一个任务 n_u 中的两个子任务 j_{n_u} 和 i_{n_u}，如果 $o_{j,i}^{n_u}=1$，即两个子任务关联，则 i_{n_u} 必须在 j_{n_u} 完成全部计算且将结果数据全部传输到 i_{n_u} 的执行位置后才能开始执行，满足条件为

$$t_{u,i} > o_{j,i}^{n_u}\left(T_{u,j}^{\mathrm{FIN}} + T_{u,j,i}^{\mathrm{TRS}}\right), \forall u \in U, \forall j,i \in I_{n_u} \tag{7}$$

2) 单个 CPU 在同一时间只能处理一个任务，所以对于调度到同一个 CPU 上的不同子任务，一个子任务必须在前一个子任务执行完毕释放 CPU 后才能开始执行。进一步地，将其分为两种情况考虑，即本地执行和 MEC 服务器执行。

当某一用户 u 的两个子任务 j_{n_u} 和 i_{n_u} 均在本地或同一个 MEC 服务器上执行时，由于无法确定先执行哪一个子任务，分两种情况讨论。如果 j_{n_u} 在 i_{n_u} 之前执行，即 $t_{u,j} < t_{u,i}$，则两个子任务开始执行时间的差值需要大于 j_{n_u} 的执行时长，即

$$t_{u,i} - t_{u,j} > T_{u,j}^{\mathrm{EXE}}, \forall u \in U, \forall j,i \in I_{n_u} \tag{8}$$

如果 i_{n_u} 在 j_{n_u} 之前执行，即 $t_{u,i} < t_{u,j}$，则两个子任务开始执行时间的差值需要大于 i_{n_u} 的执行时长，即

$$t_{u,j} - t_{u,i} > T_{u,i}^{\mathrm{EXE}}, \forall u \in U, \forall j,i \in I_{n_u} \tag{9}$$

类似地，当两个用户 v、u 的两个子任务 j_{n_v} 和 i_{n_u} 均在同一个 MEC 服务器上执行时，存在两种情况。如果 j_{n_v} 在 i_{n_u} 之前执行，即 $t_{v,j} < t_{u,i}$，则两个子任务开始执行时间的差值需要大于 j_{n_v} 的执行时长，即

$$t_{u,i} - t_{v,j} > T_{v,j}^{\mathrm{EXE}}, \forall v,u \in U, \forall j \in I_{n_v}, \forall i \in I_{n_u} \tag{10}$$

如果 i_{n_u} 在 j_{n_v} 之前执行，即 $t_{u,i} < t_{v,j}$，则两个任务开始执行时间的差值需要大于 i_{n_u} 的执行时长，即

$$t_{v,j} - t_{u,i} > T_{u,i}^{\mathrm{EXE}}, \forall v,u \in U, \forall j \in I_{n_v}, \forall i \in I_{n_u} \tag{11}$$

3) 对于每个任务来说，其执行时间受开始时间和截止时间的限制，可通过限定第一个虚拟子任务和最后一个虚拟子任务的执行时间，来实现任务的时间限制。第一个虚拟子任务需要在任务开始时间之后才能开始执行，即 $t_{u,0} > T_{n_u}^{\mathrm{st}}$，$\forall u \in U$；最后一个虚拟子任务需要在任务截止时间之前执行完毕，即 $T_{u,I+1}^{\mathrm{FIN}} < T_{n_u}^{\mathrm{ft}}$，$\forall u \in U$。

4) 每个子任务不可继续拆分，因此每个子任务只能在同一个 CPU 上完成执行。

$$\sum_{p=0}^{P} x_{u,i,p} = 1, \forall u \in U, \forall i \in I_{n_u} \tag{12}$$

3.3 优化目标

本文考虑的系统优化目标由两部分组成，包括用户的体验和多用户之间的公平性。用户的体验包括用户本地消耗的能量以及任务完成的时间；多用户之间的公平性利用卸载到 MEC 服务器的计算量比重来衡量。

1) 用户的本地能量消耗分为本地计算能量消耗和将子任务卸载到 MEC 服务器端的传输能量消耗两部分。其中，用户端 u 在本地执行任务时的总能耗可以表示为

$$E_u^{\mathrm{EXE}} = \sum_{i=1}^{I} x_{u,i,0} T_{u,i,0}^{\mathrm{EXE}} c_u^{\mathrm{cpt}}, \forall u \in U \tag{13}$$

用户端 u 将子任务从本地卸载到 MEC 服务器端消耗的总传输能量可以表示为

$$E_u^{\mathrm{TRANS}} = \sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{p=1}^{P} x_{u,i,0} x_{u,j,p} T_{u,j,i,0,p}^{\mathrm{TRS}} c_{u,p}^{\mathrm{trs}}, \forall u \in U \tag{14}$$

则用户端 u 在本地计算和卸载传输消耗的总能耗为

$$E_u = E_u^{\mathrm{EXE}} + E_u^{\mathrm{TRANS}}, \forall u \in U \tag{15}$$

2) 在考量任务的完成时延时，因为考虑整个系统中不同任务对时延的要求不同，可将任务分成 3 种，包括时延敏感任务、一般敏感任务、不敏感任务，并给这 3 种任务的完成时延赋予不用的权重 γ_u，对时延越敏感的任务其权重越高。对于用户端 u，可以把关于时延的代价函数表示为

$$C_u = \gamma_u T_{u,I+1}^{\mathrm{FIN}}, \forall u \in U \tag{16}$$

3) 由于各用户任务的差异性，某些本地能耗很高或者对时延敏感的用户会倾向于将所有任务卸载到 MEC 服务器，导致某些用户失去卸载的机会，无法从系统中获取收益。因此，本文引入表示公平性的效用函数，用来表示留在本地的计算量占任务总计算量的比重，如式(17)所示。

$$F_u=\left(\frac{W_u^{\text{total}}-W_u^{\text{off}}}{W_u^{\text{total}}}\right)^2,\forall u\in U \tag{17}$$

其中，$W_u^{\text{total}}=\sum_{i=1}^{I}\sum_{p=1}^{P}x_{u,i,p}D_{n_u,i}$ 表示卸载到 MEC 服务器的总计算量，$W_u^{\text{total}}=\sum_{i=1}^{I}D_{n_u,i}$ 表示用户所有子任务的总计算量。通过最小化所有用户的 F_u 之和，可以使多个用户的卸载比重更接近。

3.4　问题描述

在明确限制条件和优化目标后，可以将问题描述成一个 J-DTOS 优化问题，用 δ_E、δ_C、δ_F 分别表示能量消耗、任务时延和公平性所占的权重。系统通过调度中心集中做出决策，明确每个子任务的执行位置和执行时间，降低各个用户的能量消耗和任务时延，提高用户之间的公平性。具体来说，将 J-DTOS 问题建模成优化问题 P1。

$$\text{P1}:\min_{x_{u,i,p},t_{u,i}}\min_{x_{u,i,p},t_{u,i}}\sum_{u=1}^{U}\delta_E E_u+\delta_C C_u+\delta_F F_u \tag{18}$$

s.t.式(1)、(2)、(6)、(7)、(8)、(9)、(10)

4　优化问题分析与求解

上述优化问题 P1 的求解主要面临两个困难：1) 限制条件中的传输时间 $T_{u,j,i}^{\text{TRS}}$ 和目标函数中的传输能量 E_u^{TRANS} 的表达式中存在决策变量 $x_{u,j,q}$ 与 $x_{u,i,p}$ 的相乘，这部分会引入非线性项；2) 限制条件中存在对任务执行先后顺序的非线性判断。本文通过引入中间变量和转化限制条件来对上述非线性部分进行处理。

4.1　传输时间与传输能量

对于传输时间 $T_{u,j,i}^{\text{TRS}}$ 和传输能量 E_u^{TRANS} 中的决策变量 $x_{u,j,q}$ 与 $x_{u,i,p}$ 的相乘，通过引入一组新的中间变量 $z_{u,j,i,q,p}$ 来代替 $x_{u,j,q}$ 与 $x_{u,i,p}$ 的相乘项，可以将传输时间和传输能量中的决策变量相乘项转化为线性的，通过如式(19)所示的线性约束条件实现。

$$\begin{cases}z_{u,j,i,q,p}\in\{0,1\}\\ z_{u,j,i,q,p}\leqslant x_{u,j,q}\\ z_{u,j,i,q,p}\leqslant x_{u,i,p}\\ z_{u,j,i,q,p}\geqslant x_{u,j,q}+x_{u,i,p}-1\end{cases} \tag{19}$$

引入这组中间变量 $z_{u,j,i,q,p}$ 后，可以将传输时间和传输能量分别转换成线性形式，如式(20)所示。

$$T_{u,j,i}^{\text{TRS}}=\sum_{q=0}^{P}\sum_{p=0}^{P}z_{u,j,i,q,p}T_{u,j,i,q,p}^{\text{TRS}},\forall u\in U,\forall j,i\in I_{n_u} \tag{20}$$

$$E_u^{\text{TRANS}}=\sum_{i=1}^{I}\sum_{j=1}^{I}\sum_{p=1}^{P}z_{u,j,i,0,p}T_{u,j,i,0,p}^{\text{TRS}}c_{u,p}^{\text{trs}},\forall u\in U \tag{21}$$

4.2　CPU 独占

针对任务执行先后顺序的非线性判断问题，通过引入一个中间变量 $b_{j,i}$ 来表示两个子任务 j_{n_v} 和 i_{n_u} 的执行顺序。当 j_{n_v} 先执行时，$b_{j,i}=1$；当 i_{n_u} 先执行时，$b_{j,i}=0$。

$$b_{j,i}=\begin{cases}1,t_{u,i}-t_{v,j}>0\\ 0,t_{v,j}-t_{u,i}>0\end{cases} \tag{22}$$

通过进一步转化将 $b_{j,i}$ 表示成线性形式，如式(23)所示。

$$\begin{cases}b_{j,i}\in\{0,1\}\\ \dfrac{t_{u,i}-t_{v,j}}{C}\leqslant b_{j,i}\leqslant\dfrac{t_{u,i}-t_{v,j}}{C}+1\end{cases} \tag{23}$$

其中，C 是一个足够大的数，足以保证 $\dfrac{\left|t_{u,i}-t_{v,j}\right|}{C}<1$。由此可以将式(7)表示为如式(24)所示的形式。

$$t_{u,i}-t_{u,j}+B\left(2-x_{u,j,p}-x_{u,i,p}\right)\geqslant b_{j,i}T_{u,j}^{\text{EXE}}-A\left(1-b_{j,i}\right),\quad\forall u\in U,\forall j,i\in I_{n_u},\forall p\in P\cup\{0\} \tag{24}$$

$$t_{u,j}-t_{u,i}+B\left(2-x_{u,j,p}-x_{u,i,p}\right)\geqslant\left(1-b_{j,i}\right)T_{u,i}^{\text{EXE}}-Ab_{j,i},\quad\forall u\in U,\forall j,i\in I_{n_u},\forall p\in P\cup\{0\} \tag{25}$$

式(24)、式(25)两个限制条件限制了同一用户 u 的两个子任务在本地或 MEC 服务器执行时 CPU 独占。其中 B 是一个足够大的数，保证了上述限制只有两个子任务在同一个 CPU 执行时才生效；A 也是一个足够大的数，保证了当 j_{n_u} 先执行时，前一个限制条件有效；当 i_{n_u} 先执行时，后一个限制条件有效，可表示为

$$t_{u,i}-t_{v,j}+B\left(2-x_{v,j,p}-x_{u,i,p}\right)\geqslant b_{j,i}T_{v,j}^{\text{EXE}}-A\left(1-b_{j,i}\right),\quad\forall v,u\in U,\forall j\in I_{n_v},\forall i\in I_{n_u},\forall p\in P \tag{26}$$

$$t_{v,j}-t_{u,i}+B\left(2-x_{v,j,p}-x_{u,i,p}\right)\geqslant\left(1-b_{j,i}\right)T_{u,i}^{\text{EXE}}-Ab_{j,i},\quad\forall v,u\in U,\forall j\in I_{n_v},\forall i\in I_{n_u},\forall p\in P \tag{27}$$

式(26)、式(27)两个限制条件限制了不同用户 v、u 的两个子任务在MEC服务器执行时的CPU独

占。其中 B 是一个足够大的数，保证了上述限制只有两个子任务在同一个 CPU 执行时才生效；A 也是一个足够大的数，保证了当 j_{n_v} 先执行时，前一个限制条件有效；当 i_{n_u} 先执行时，后一个限制条件有效。

在完成对非线性变量的转换后，可以将优化问题 P1 转化成线性规划问题，能使用标准求解方式求解。将转化后的问题表示为 P2。

$$\mathrm{P2}:\min_{\substack{x_{u,i,p},t_{u,i},\\ z_{u,j,i,q,p},b_{j,i}}}\ \min_{\substack{x_{u,i,p},t_{u,i},\\ z_{u,j,i,q,p},b_{j,i}}}\ \sum_{u=1}^{U}\delta_E E_u+\delta_C C_u+\delta_F F_u$$

$$\text{s.t.式(1)、(2)、(6)、(8)、(9)、(10)、(16)、(19)、(20)} \quad (28)$$

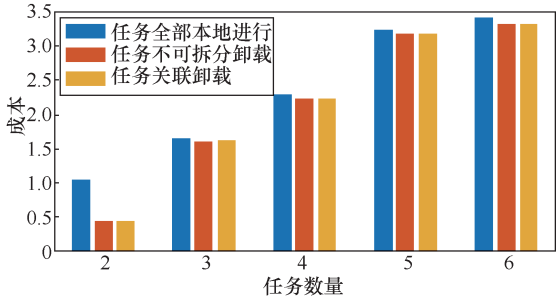
混合整数线性规划问题 P2 已有许多成熟的求解方法，可以使用分支定界法对其求解，本文不再详细论述求解过程。

5　仿真结果与分析

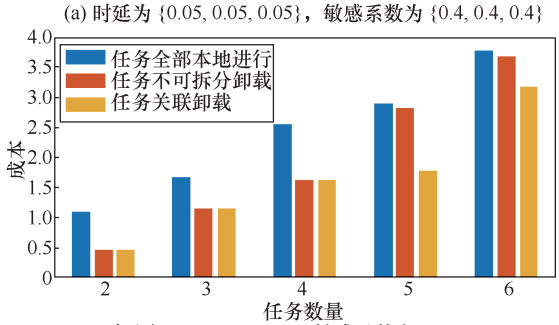
考虑由两个 MEC 服务器、3 个移动用户构成的仿真场景，移动用户持有对时延敏感任务、对时延一般敏感任务或对时延不敏感任务。MEC 服务器之间的传输时延在 1.1～0.9 ns/bit 随机产生。移动用户与 MEC 服务器之间的无线传输速率由以下配置生成：带宽为 5 MHz，噪声功率为 30 dBm，每个用户与 MEC 服务器之间的信道增益为 2～4 的随机数，用户的传输功率在 0.9～1.1 随机产生。此外，每个用户的本地计算功率在 3.9～4.1 随机产生，本地和 MEC 服务器的计算频率分别分散在 60～61 GHz 和 10～11 GHz。针对用户的任务模型，使用每轮仿真中重新随机生成的 DAG 表示，两个子任务间关联的概率固定为 0.2，每个子任务的计算量在 50～300 KB，其中所需计算量为 2 000 CPU cycles/bit；每两个关联子任务之间的传输数据量在 200～500 KB。

5.1　不同时延敏感度情况下任务复杂度对性能的影响

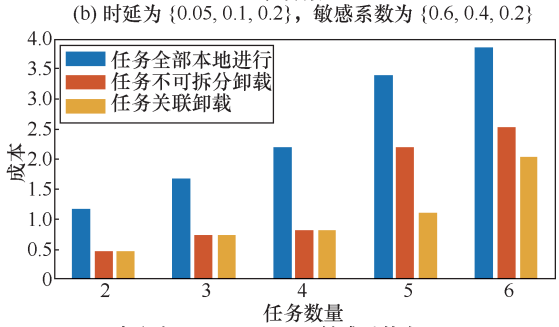
首先比较 3 种不同卸载方案，即任务全部本地执行、任务不可拆分卸载，以及本文考虑的任务关联卸载，分别对不同子任务数量和时延要求下三者的性能进行对比。为了方便观察，设置表示公平性代价的系数 $\delta_F=0$，只观察移动用户时延和能耗的变化。固定参数 $\delta_E=1$ 且 $\delta_C=1$，所有任务的时延敏感度之和 $\sum_{u=1}^{U}\gamma_u=1.2$。不同时延情况下任务复杂度对性能的影响如图 3 所示，其中图 3(a)中 3 种任务的最大时延 $T_{n_u}^{\mathrm{ft}}$ 均设为 0.05 s，时延的敏感系数 γ_u 均设为 0.4；图 3(b)中 3 种任务的时延 $T_{n_u}^{\mathrm{ft}}$ 分别设为 0.05 s、0.1 s、0.2 s，时延敏感系数 γ_u 分别设为 0.6、0.4、0.2；图 3(c)中 3 种任务的时延 $T_{n_u}^{\mathrm{ft}}$ 分别设为 0.05 s、0.2 s、0.4 s，时延敏感系数 γ_u 分别设为 0.8、0.3、0.1；图 3(d)中 3 种任务的最大时延 $T_{n_u}^{\mathrm{ft}}$ 均设为 0.4 s，时延的敏感系数 γ_u 均设为 0.4。子任务的个数均从 2 变化到 6。



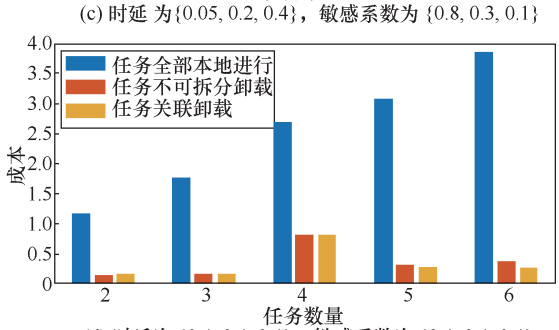
(a) 时延为 {0.05, 0.05, 0.05}，敏感系数为 {0.4, 0.4, 0.4}



(b) 时延为 {0.05, 0.1, 0.2}，敏感系数为 {0.6, 0.4, 0.2}



(c) 时延 为{0.05, 0.2, 0.4}，敏感系数为 {0.8, 0.3, 0.1}



(d) 时延为 {0.4, 0.4, 0.4}，敏感系数为 {0.4, 0.4, 0.4}

图 3　不同时延情况下任务复杂度对性能的影响

从图 3 可以看出，在时延敏感度相同的情况下，

子任务越多，任务复杂度越高，任务卸载的灵活性越高，所以关联任务卸载相对于其他两种卸载方式有更明显的性能优势，关联任务的卸载最多可以比本地执行节省能量约 58%，最多比全部卸载或全部本地执行的策略节省能量约 50%。此外，在多个任务的时延要求差别大的情况下，时延要求差别越大，关联任务卸载的性能越好。在多个任务的时延要求差别很小的情况下，因为时间紧张，系统会尽量使用本地的计算资源执行任务，而将少量并行的任务卸载到服务器端执行，所以节省能量效果不明显；而在时延要求较低的情况下，系统会尽量将可串联执行的任务都卸载到服务器端，或者直接将所有任务卸载到服务器端，所以系统性能会与任务不可拆分卸载的性能相近。

5.2　任务复杂度以及公平性权重对系统公平性的影响

本文使用业界常用的 Jain's fairness 系数来衡量整个系统对多用户的公平性，Jain's fairness 系数越接近 1，多用户之间的差别越小。采用 3 种任务的时延 $T_{n_u}^{\mathrm{ft}}$ 分别为 0.05 s、0.2 s、0.4 s，时延系数 γ_u 分别为 0.8、0.3、0.1 的设置，$\delta_E=1$ 且 $\delta_C=1$。任务复杂度对系统公平性的影响如图 4 所示，公平性权重对系统公平性的影响如图 5 所示。在图 4 中，固定公平性权重 $\delta_F=10$，观察子任务数量从 2 变化到 6 时系统公平性的变化。可以看到，随着子任务数量的增加，任务模型更复杂，卸载灵活性更高，系统的公平性权重从 0.3 增加到接近 1。在图 5 中，固定子任务数量 $I=3$，可以看到，随着公平性权重的增加，系统的公平性权重也从 0.3 增加到接近 0.8。

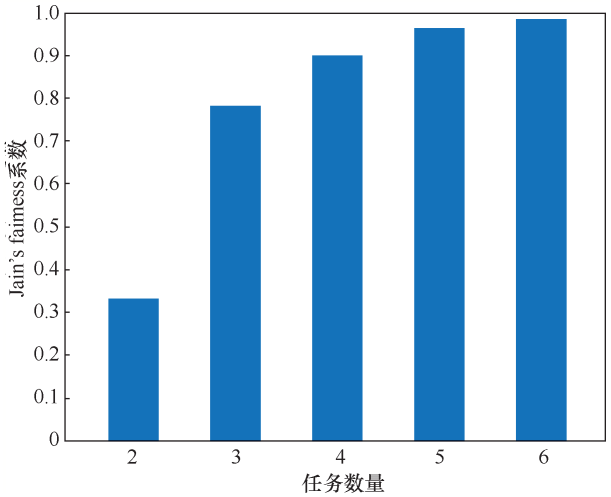


图 4　任务复杂度对系统公平性的影响

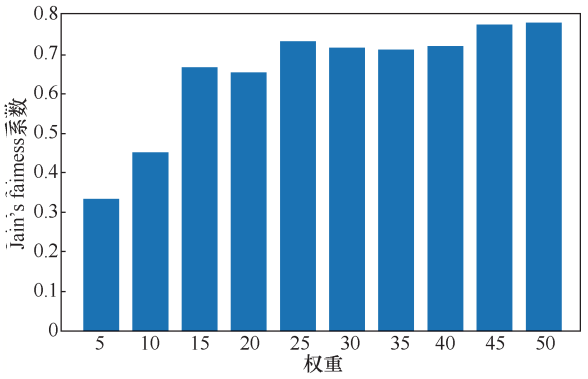


图 5　公平性权重对系统公平性的影响

6　结束语

本文研究了多用户多服务器系统中的可切分任务卸载和调度问题，当任务可在时间限制内完成时，联合优化系统的 3 个目标（即能量、时延和公平性）构建了 J-DTOS 问题。通过对问题中的非线性部分进行转化，将原 NP-hard 问题转化为线性混合整数规划问题并求解。最后通过仿真评估本文所提方案的性能，相比本地执行方案和 0-1 卸载方案，本文所提方案可显著提高用户 QoE 和公平性。本文主要考虑了集中式子任务卸载和调度方案，在后续研究中，将进一步考虑分布式卸载和调度场景，研究自私用户的多用户博弈问题。

参考文献：

[1] MAO Y, YOU C, ZHANG J, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[2] WANG F, XU J, DING Z G. Multi-antenna NOMA for computation offloading in multiuser mobile edge computing systems[J]. IEEE Transactions on Communications, 2017, 67(3): 2450-2463.

[3] LIU M, YU F R, TENG Y, et al. Joint computation offloading and content caching for wireless blockchain networks[C]//2018 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway: IEEE Press, 2018: 517-522.

[4] TAN H, HAN Z, LI X Y, et al. Online job dispatching and scheduling in edge-clouds[C]//2017 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2017: 1-9.

[5] 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.

[6] WANG Y T, SHENG M, WANG X J, et al. Mobile-edge computing: partial computation offloading using dynamic voltage scaling[J]. IEEE Transactions on Communications, 2016, 64(10): 4268-4282.

[7] MAHMOODI S E, UMA R N, SUBBALAKSHMI K P. Optimal joint scheduling and cloud offloading for mobile applications[J]. IEEE Transactions on Cloud Computing, 2019, 7(2): 301-313.

[8] HUANG D, WANG P, NIYATO D. A dynamic offloading algorithm for mobile computing[J]. IEEE Transactions on Wireless Communications, 2012, 11(6): 1991-1995.

[9] GENG Y L, YANG Y , CAO G H. Energy-efficient computation offloading for multicore-based mobile devices[C]//2018 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 46-54.
[10] SUNDAR S, LIANG B. Offloading dependent tasks with communication delay and deadline constraint[C]//2018 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2018: 37-45.
[11] GUO G T, LIU J D, YANG Y Y, et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing[J]. IEEE Transactions on Mobile Computing, 2019, 18(2): 319-333.

[作者简介]

路静（1996− ），女，哈尔滨工业大学（深圳）电子与信息工程学院硕士生，主要研究方向为移动边缘计算中的任务卸载策略。

李晗琳（1996− ），女，哈尔滨工业大学（深圳）电子与信息工程学院硕士生，主要研究方向为移动边缘计算中的资源部署、任务卸载、经济模型分析等。

高林（1980− ），男，博士，哈尔滨工业大学（深圳）电子与信息工程学院副教授，主要研究方向为移动边缘计算、群智计算、群体智能、博弈论、强化学习等。