DOI:10.3969/j.issn.1003-5060.2025.03.006

混合线性依赖和独立子任务的卸载与服务缓存计算资源分配

金 晨, 王青山
(合肥工业大学 数学学院,安徽 合肥 230601)

摘 要:文章在设备到设备(device-to-device,D2D)辅助的单边缘服务器多用户移动设备场景下,研究混合线性依赖和独立子任务的卸载与服务缓存计算能力分配问题,目标为最小化能耗与时延的加权和。首先将该问题模型化为非凸问题,接着提出混合任务两层优化(mixed task two-layer optimization,MTTLO)算法。该算法第 1 层通过固定各个设备或边缘服务器上计算能力的分配,将混合线性依赖和独立子任务的非凸问题使用部分固定最优成本卸载算法得到优先级,并求出任务卸载策略;第 2 层使用卡罗需-库恩-塔克(Karush-Kuhn-Tucker,KKT)条件求出固定任务卸载策略下设备或边缘服务器服务缓存的计算能力的封闭解。实验结果表明, MTTLO 算法优于其他基准算法,能够有效减少系统的能耗与时延加权和,验证了算法的有效性。

关键词:移动边缘计算;任务卸载;服务缓存;线性依赖;资源分配
中图分类号:TN929.538 **文献标志码**:A **文章编号**:1003-5060(2025)03-0327-09

Offloading and service cache computing resource allocation for mixed linearly dependent and independent subtasks

JIN Chen, WANG Qingshan
(School of Mathematics, Hefei University of Technology, Hefei 230601, China)

Abstract: In device-to-device(D2D) assisted single-edge server multi-user mobile device scenario, this paper studies the offloading and service cache computing power allocation for mixed linearly dependent and independent subtasks, aiming to minimize the weight sum of energy consumption and delay. Firstly, the problem is modeled as a non-convex problem, and then the mixed task two-layer optimization(MTTLO) algorithm is proposed. In the first layer of MTTLO algorithm, the computing power allocation on each device or edge server is fixed, the non-convex problem of mixed linearly dependent and independent subtasks is given priority by using the partial fixed optimal cost offloading algorithm, and the task offloading strategy is obtained. The second layer uses the Karush-Kuhn-Tucker(KKT) conditions to find the closed solution of computing power of device or edge server service cache under fixed task offloading strategy. Simulation results show that the proposed MTTLO algorithm is superior to other benchmark algorithms, and can effectively reduce the weight sum of system energy consumption and delay, which verifies the effectiveness of the algorithm.

Key words: mobile edge computing(MEC); task offloading; service cache; linear dependence; resource allocation

收稿日期:2023-04-14;**修回日期**:2023-05-30
基金项目:安徽省自然科学基金资助项目(2208085MF165)
作者简介:金 晨(1998—),男,安徽金寨人,合肥工业大学硕士生;
王青山(1975—),男,安徽合肥人,博士,合肥工业大学教授,博士生导师,通信作者,E-mail:qswang@hfut.edu.cn.

0 引 言

随着5G和移动互联网的快速发展,移动设备上存在大量的应用程序需要进行处理,而移动设备由于其自身服务缓存有限,难以单独完成延迟敏感型和计算密集型的任务。云服务器拥有众多的服务缓存,移动设备可以将数据上传到云服务器进行处理,然而随着大量数据卸载到云计算以及网络的复杂性,云计算往往面临着网络拥塞问题,从而导致延迟,无法满足用户的需求[1],于是移动边缘计算(mobile edge computing,MEC)应运而生。边缘计算[2]不同于云计算,其部署在移动设备附近,可提供一定的计算资源,能够有效地满足计算密集型和延迟敏感型任务需求。

为了缓解无线接入点(access point,AP)的网络拥塞以及更好地利用同一AP覆盖范围内的其他空闲协助设备的计算资源,设备到设备[3](device-to-device,D2D)是一个很好的解决方案。当用户需要将其任务卸载给周围其他用户时,D2D无需通过AP卸载,2个移动设备可以直接进行通信。此时,可以在缓解边缘服务器压力的同时也进一步降低系统的能耗和时延。

在MEC的场景下,文献[4]在终端多边缘设备下研究了多跳计算部分卸载问题,考虑了路径约束,一个应用程序包含多个独立子任务,一部分可以在本地执行,另一部分在边缘设备上执行,为实现应用程序时延最小化,提出了联合部分卸载和流调度启发式算法;文献[5]在双用户单边缘的场景下,分层优化任务卸载决策和移动设备的发射功率和计算频率,使用罗需-库恩-塔克(Karush-Kuhn-Tucker,KKT)条件获得了功率和计算频率的封闭表达,基于一次爬升策略提出了一种基于吉布斯抽样算法来获取任务卸载决策的最优解;文献[6]研究了多边缘服务器多用户下可分解为依赖性子任务的任务卸载问题和移动设备资源分配问题,假设每个子任务在本地设备或具有有限容量的边缘服务器上串行处理且具有截止时间限制,提出了一种去中心的卸载算法;文献[7]考虑了间断情况下故障恢复的依赖性子任务的卸载和资源分配问题,目标是最优化每个子任务的时延、能耗和带宽费用,对于子任务在本地执行受时延约束情况下通过KKT条件获取最优的本地计算频率,考虑传输速率和传输功率的双层优化获得对应最优解;文献[8]基于多部分卸载模式的多服务器系统下的协同卸载问题,考虑了信道分配、多部分任务卸载以及基站协作,针对最小化平均延迟的目标,提出了基于遗传和深层确定性策略梯度的计算卸载方案。

在D2D的MEC场景下,文献[9]研究了任务部分卸载和资源分配问题,考虑了3种卸载模式,提出一种以最小化任务请求用户的时延和能耗的启发式算法;文献[10]研究了基于D2D的移动设备、单边缘服务器和云服务器3层部分卸载和资源分配问题,运用拉格朗日乘子法和博弈论提出了资源分配和任务卸载策略;文献[11]考虑了服务缓存确定的边缘服务器下多用户多依赖子任务的任务卸载问题,为实现依赖子任务最大完成时间的最小化,将任务卸载公式转化为不考虑服务器内部任务顺序的凸优化问题,使用贪心算法给出最后的任务卸载策略;文献[12]研究了单个宏单元基站、多个小单元基站及多个移动设备下的大规模MEC场景下,一个可分解为多个子任务的任务卸载问题以及路径规划和小单元基站的计算资源分配问题,为了最小化能耗和时延,提出了基于交替方向乘子法(alternating direction method of multipliers,ADMM)分块的优化算法。文献[13]研究了请求者终端可部分卸载任务的实时卸载和资源分配问题,为了最大化队列任务完成数量、最小化能耗和成本,采用李雅普诺夫(Lyapunov)漂移加惩罚优化方法和支持向量机(support vector machines,SVM)求解;文献[14]基于D2D的单用户多跳任务独立子任务卸载问题,将移动节点进行分层,按层将任务卸载到移动节点或本地执行,提出了2种基于学习理论的算法以最大化网络效用。

以上研究有的只考虑应用程序的整体卸载,有的只单独考虑独立子任务的划分形式或任务依赖的划分形式。事实上,由于应用程序所需要的服务不同,2种划分形式共存,造成整个系统中设备或边缘服务器上子任务执行顺序更加复杂。此外,现有关于设备或边缘服务器服务缓存的计算能力资源是固定分配假设,造成了计算资源的浪费,从而导致任务计算时延较大,本文则根据任务卸载策略对服务缓存进行动态分配。

本文在D2D辅助单边缘服务器多移动设备场景下,研究了任务卸载和服务缓存计算能力资源分配的混合线性依赖和独立子任务的新问题,将该问题建模成优化问题,并提出混合任务两层优化(mixed task two-layer optimization,MTTLO)算法求解优化问题。MTTLO算法使用部分

固定最优成本卸载算法得到任务卸载策略，根据KKT 条件求出用户移动设备或边缘服务器计算能力资源分配策略的封闭解。

1　系统模型

本文考虑存在空闲协助设备和用户移动设备的单边缘服务器场景，系统模型如图 1 所示。该场景由 M 个用户移动设备、$(M-K)$个空闲协助设备和 1 个边缘服务器组成，边缘服务器部署在AP 附近，忽略两者的通信时间。用户移动设备定义为 $N_{\mathrm{u}}=\{1,2,\cdots,K\}$，空闲协助设备和边缘服务器组成协助节点 $N_{\mathrm{h}}=\{K+1,\cdots,M,M+1\}$，其中，$M+1$ 表示边缘服务器。用户移动设备应用程序划分的子任务或本地执行，或卸载到边缘服务器，或通过 D2D 将任务卸载到空闲协助设备上。空闲协助设备间 D2D 连接，空闲协助设备与边缘服务器无线连接。

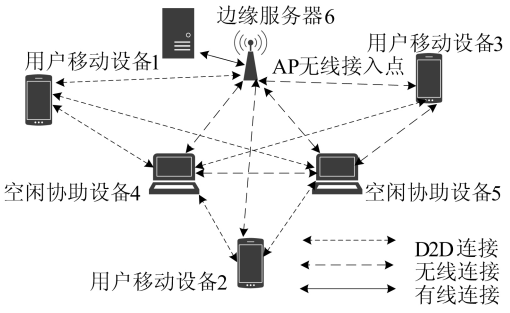

图 1　系统模型

1.1　任务模型

假设用户移动设备集合 $N_{\mathrm{u}}=N_{\mathrm{u,d}}\cup N_{\mathrm{u,ind}}$，其中：$N_{\mathrm{u,d}}$表示应用程序只划分为线性依赖子任务的用户移动设备集合；$N_{\mathrm{u,ind}}$表示应用程序只划分为独立子任务的用户移动设备集合。用户移动设备 $i\in N_{\mathrm{u}}$ 有一个应用程序 v_i 需要计算。若 $i\in N_{\mathrm{u,ind}}$，则其产生的应用程序分解为 N_i 个独立子任务，即 $v_i=\{v_{i,1},v_{i,2},\cdots,v_{i,N_i}\}$，$v_i$ 中的每个子任务相互独立没有依赖性；若 $i\in N_{\mathrm{u,d}}$，则其产生的应用程序分解为 N_i 线性依赖子任务，即 $v_i=\{v_{i,0},\cdots,v_{i,N_i},v_{i,N_i+1}\}$，其中：$v_{i,0}$与 v_{i,N_i+1}为虚拟子任务，均在本地执行且没有执行时间，$v_{i,j}(0\leqslant j\leqslant N_i)$与 $v_{i,j+1}$构成直接依赖关系，$v_{i,j+1}$需等待 $v_{i,j}$执行完成后并将结果数据传输至 $v_{i,j+1}$所在设备上后 $v_{i,j+1}$才能执行，即 $v_{i,j}$的输出为 $v_{i,j+1}$的输入。对于 $i\in N_{\mathrm{u}}$，用二元组$\{d_{i,j},w_{i,j}\}$表示 $v_{i,j}$，其中：$d_{i,j}$表示 $v_{i,j}$的数据大小；$w_{i,j}$表示 $v_{i,j}$每 1 bit 数据量所需要的 CPU 周期。系统中服务缓存集合为 $S_{\mathrm{F}}=\{1,2,\cdots,J\}$，$F(v_{i,j})\in S_{\mathrm{F}}$ 表示 $v_{i,j}$所需要的服务缓存。记需要服务缓存 $F(v_{i,j})$计算的所有子任务集合为 $V_{i,j}=\{v_{m,n}\mid F(v_{m,n})=F(v_{i,j}),m\in N_{\mathrm{u}},1\leqslant n\leqslant N_m\}$，其数量为$|V_{i,j}|$。假设需要服务缓存 $s\in S_{\mathrm{F}}$ 来计算的所有子任务集合定义为$V(s)=\{v_{m,n}\mid F(v_{m,n})=s,s\in S_{\mathrm{F}},m\in N_{\mathrm{u}},1\leqslant n\leqslant N_m\}$，其数量为$|V(s)|$。对于同一个应用程序中独立子任务所需要的服务缓存通常相同。

1.2　计算模型

假设用户移动设备或协助节点上每个服务缓存同一时间只能计算一个任务，每个用户移动设备和协助节点配置服务缓存数量和类型有区别，任务只能在与之对应的服务缓存的本地设备或协助节点上进行计算，用户移动设备或协助节点$k\in N_{\mathrm{u}}\cup N_{\mathrm{h}}$ 上配置的服务缓存数量为 $N_{\mathrm{F}k}$，配置的服务缓存为 $F_k=\{F_{k,1},F_{k,2},\cdots,F_{k,N_{\mathrm{F}k}}\}$，其中 $F_{k,m}\in S_{\mathrm{F}},1\leqslant m\leqslant N_{\mathrm{F}k}$。用 $f_k^{F(v_{i,j})}$ 表示用户移动设备或协助节点 k 上服务缓存 $F(v_{i,j})$的计算能力。用户移动设备和空闲协助设备配置的服务缓存数量有限，而边缘服务器上存在服务缓存。

任务 $v_{i,j}$在配置有对应的服务缓存 $F(v_{i,j})$的用户移动设备或协助节点 $k(k\in N_{\mathrm{u}}\cup N_{\mathrm{h}})$上计算时间为：

$$t_k^{i,j}=\frac{d_{i,j}w_{i,j}}{f_k^{F(v_{i,j})}} \tag{1}$$

子任务 $v_{i,j}$在本地移动设备上计算能耗为：

$$E_i^{i,j}=d_{i,j}w_{i,j}\kappa_i(f_i^{F(v_{i,j})})^2 \tag{2}$$

其中，κ_i 为本地移动设备 i 上与芯片结构相关的有效开关电容。

1.3　传输模型

连接边缘服务器的 AP 使用正交频分复用，将带宽平均分配。一部分带宽用于移动设备和边缘服务器的通信以及空闲设备和边缘服务器的通信，另一部分用于移动设备和空闲协助设备 D2D 通信。用户移动设备间不进行 D2D 通信。子任务 $v_{i,j}$从协助节点 m 传输到其他协助节点 k 的传输时间为：

$$t_{m,k}^{i,j}=\frac{d_{i,j}}{B\mathrm{lb}\left(1+\frac{p_m h_{m,k}}{BN_0}\right)} \tag{3}$$

其中：B 为信道带宽；p_m 为协助节点 m 的传输功率；N_0 为信道的噪声功率谱密度；$h_{m,k}$为协助节点 m 与协助节点 k 之间的信道增益，$h_{m,k}=G[3\times 10^8/(4\pi F_{\mathrm{c}}d_{m,k})]^3$，$G=4.11$，$F_{\mathrm{c}}=915$ MHz，$d_{m,k}$

为协助节点 m 与协助节点 k 之间的距离。

同理可以得到本地设备 i 到协助节点 k 的传输时间 $t_{i,k}^{i,j}$ 与协助节点 k 到本地设备 i 的传输时间 $t_{k,i}^{i,j}$。

对于独立子任务 $v_{i,j}(i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i)$，单独传输至协助节点 k 的传输时间为 $t_{i,k}^{i,j}$，本文假设用户移动设备 $i\in N_{\mathrm{u,ind}}$ 需要将传输至某个协助节点上所有子任务传输完成后才开始该子任务在该协助节点上的计算。因此，应用程序 v_i 存在多个独立子任务均要传输到该协助节点时，每个独立子任务的实际传输时间为这些独立子任务单独传输至该协助节点时间之和。

子任务 $v_{i,j}$ 从本地设备 i 传输到协助节点 k 的传输能耗为：

$$E_{i,k}^{i,j}=p_i t_{i,k}^{i,j} \tag{4}$$

1.4 问题形式化

记 $S_{i,j}=\{k\mid F(v_{i,j})\in F_k,k\in\{i\}\bigcup N_{\mathrm{h}}\}$ 为存在子任务 $v_{i,j}$ 对应的服务缓存 $F(v_{i,j})$ 的本地设备和协助节点。用 $x_{k,s,r}^{i,j}$ 表示 $v_{i,j}$ 的任务卸载策略。当且仅当子任务 $v_{i,j}$ 卸载到本地设备或协助节点 $k(k\in S_{i,j})$ 上服务缓存 $s(s=F(v_{i,j}))$ 的第 r 个位置时 $x_{k,s,r}^{i,j}=1$；否则 $x_{k,s,r}^{i,j}=0$。

由于 1 个子任务只能卸载到 1 个本地设备或协助节点上，对应需要满足的约束为：

$$\sum_{k\in S_{i,j}}\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}=1,\quad i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i \tag{5}$$

1 个任务最多在 1 个本地设备或协助节点服务缓存上占据 1 个位置：

$$\sum_{i=1}^{K}\sum_{j=1}^{N_i}x_{k,s,r}^{i,j}\leqslant 1,k\in S_{i,j},\quad s\in F_k,1\leqslant r\leqslant|V(s)| \tag{6}$$

服务缓存上 1 个位置被占用之前，任务不能分配到后续的位置：

$$\sum_{i=1}^{K}\sum_{j=1}^{N_i}x_{k,F(v_{i,j}),r}^{i,j}-\sum_{i=1}^{K}\sum_{j=1}^{N_i}x_{k,F(v_{i,j}),r-1}^{i,j}\leqslant 0,\quad k\in S_{i,j},2\leqslant r\leqslant|V_{i,j}| \tag{7}$$

应用程序 $v_i(i\in N_{\mathrm{u,ind}})$ 的时延 t_i 不小于该应用程序划分的独立子任务的最晚完成时间，即

$$t_{\mathrm{s}}^{i,j}+\sum_{k\in S_{i,j}}\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}t_k^{i,j}\leqslant t_i,\quad i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i \tag{8}$$

其中，$t_{\mathrm{s}}^{i,j}$ 为子任务 $v_{i,j}$ 的开始计算时间。由于独立子任务输出结果通常很小，忽略下行传输时间。

独立子任务 $v_{i,j}(i\in N_{\mathrm{u,ind}})$ 开始计算时间满足：

$$\sum_{m=1}^{N_i}\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,m}t_{i,k}^{i,j}-C\Big(1-\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}\Big)\leqslant t_{\mathrm{s}}^{i,j},\quad i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i,k\in S_{i,j},k\neq i \tag{9}$$

其中，C 为较大的常数。当 $v_{i,j}$ 卸载到协助节点 k 上时，即 $\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}=1$，需要满足式(9)约束，$v_{i,j}$ 传输时间不小于 v_i 传输至该协助节点 k 上所有子任务传输时间之和；反之该约束不起作用。

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}})$ 的开始时刻一定晚于前 1 个子任务 $v_{i,j-1}$ 的完成时刻与 $v_{i,j}$ 传输时间之和，因此有：

$$t_{\mathrm{s}}^{i,j-1}+\sum_{k\in S_{i,j-1}}\sum_{r=1}^{|V_{i,j-1}|}x_{k,F(v_{i,j-1}),r}^{i,j-1}t_k^{i,j-1}+\sum_{k\in S_{i,j-1}}\sum_{r=1}^{|V_{i,j-1}|}\sum_{l\in S_{i,j},l\neq k}\sum_{s=1}^{|V_{i,j}|}x_{k,F(v_{i,j-1}),r}^{i,j-1}x_{l,F(v_{i,j}),s}^{i,j}t_{k,l}^{i,j}\leqslant t_{\mathrm{s}}^{i,j},\quad i\in N_{\mathrm{u,d}},2\leqslant j\leqslant N_i \tag{10}$$

当 $j=1$ 时，式(10)为：

$$\sum_{l\in S_{i,j},l\neq i}\sum_{s=1}^{|V_{i,j}|}x_{l,F(v_{i,j}),s}^{i,j}t_{i,l}^{i,j}\leqslant t_{\mathrm{s}}^{i,j}。$$

子任务 $v_{i,j}(i\in N_{\mathrm{u}})$，$v_{m,n}$ 在本地设备或协助节点 k 上依次执行时需要满足以下约束：

$$t_{\mathrm{s}}^{m,n}+\sum_{k\in S_{m,n}}t_k^{m,n}-C(2-x_{k,F(v_{i,j}),r}^{i,j}-x_{k,F(v_{i,j}),r-1}^{m,n})\leqslant t_{\mathrm{s}}^{i,j},\quad i,m\in N_{\mathrm{u}},1\leqslant j\leqslant N_i,1\leqslant n\leqslant N_m,\quad k\in S_{i,j},2\leqslant r\leqslant|V_{i,j}| \tag{11}$$

式(11)表明 $v_{i,j}$ 在本地设备或协助节点 k 的服务缓存 $F(v_{i,j})$ 上第 r 位置计算，且 $v_{m,n}$ 在其前面第 $r-1$ 位置计算，即 $x_{k,F(v_{i,j}),r}^{i,j}=x_{k,F(v_{i,j}),r-1}^{m,n}=1$，可得 $v_{i,j}$ 的开始时刻应不小于 $v_{m,n}$ 的开始时刻与计算时间之和；反之该约束不起作用。

独立子任务 d 的传输与计算能耗之和为：

$$E_{i,j}=\sum_{k\in S_{i,j},k\neq i}\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}E_{i,k}^{i,j}+\sum_{r=1}^{|V_{i,j}|}x_{i,F(v_{i,j}),r}^{i,j}E_i^{i,j} \tag{12}$$

应用程序 $v_i(i\in N_{\mathrm{u,d}})$ 的时延为：

$$t_i=t_{\mathrm{s}}^{i,N_i}+\sum_{k\in S_{i,N_i}}\sum_{r=1}^{|V_{i,N_i}|}x_{k,F(v_{i,N_i}),r}^{i,N_i}t_k^{i,N_i}+\sum_{k\in S_{i,N_i},k\neq i}\sum_{r=1}^{|V_{i,N_i}|}x_{k,F(v_{i,N_i}),r}^{i,N_i}t_{k,i}^{i,N_i+1} \tag{13}$$

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},2\leqslant j\leqslant N_i)$ 的传输与计算能耗之和为:

$$E_{i,j}=\sum_{k\in S_{i,j},k\neq i}\sum_{s=1}^{|V_{i,j}||V_{i,j-1}|}\sum_{r=1}^{x_{i,F(v_{i,j-1}),r}^{i,j-1}x_{k,F(v_{i,j}),s}^{i,j}E_{i,k}^{i,j}}+\sum_{r=1}^{|V_{i,j}|}x_{i,F(v_{i,j}),r}^{i,j}E_i^{i,j}\tag{14}$$

当 $j=1$ 时,$E_{i,j}=\sum_{k\in S_{i,j},k\neq i}\sum_{s=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),s}^{i,j}E_{i,k}^{i,j}+\sum_{r=1}^{|V_{i,j}|}x_{i,F(v_{i,j}),r}^{i,j}E_i^{i,j}$。

应用程序 $v_i(i\in N_{\mathrm{u}})$ 的能耗为:

$$E_i=\sum_{j=1}^{N_i}E_{i,j}\tag{15}$$

本文将系统成本定义为所有应用程序的时延和能耗的加权之和为:

$$c=\theta\sum_{i=1}^{K}t_i+(1-\theta)\sum_{i=1}^{K}E_i\tag{16}$$

其中,权重 $0\leqslant\theta\leqslant1$。

当用户移动设备和协助节点上的服务缓存计算资源给定时,最小化系统成本的优化问题 P1:可以表述为:

$$\begin{aligned}&\min_{\boldsymbol{x},\boldsymbol{t}}\quad c;\\&\text{s.t.}\quad\text{式}(5)\sim(16);\\&x_{k,F(v_{i,j}),r}^{i,j}\in\{0,1\},i\in N_{\mathrm{u}},\\&1\leqslant j\leqslant N_i,1\leqslant r\leqslant|V_{i,j}|;\\&x_{k,s,r}^{i,j}=0,i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i,\\&k\in S_{i,j},s\neq F(v_{i,j});\\&x_{k,s,r}^{i,j}=0,i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i,k\notin S_{i,j};\\&t_{\mathrm{s}}^{i,j}\geqslant0,i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i\end{aligned}\tag{17}$$

其中:$\boldsymbol{x}$ 为所有子任务的任务卸载策略;$\boldsymbol{t}$ 为所有子任务的开始时间和应用程序的完成时间组成的向量。因为 $x_{k,F(v_{i,j}),r}^{i,j}$ 为整数,所以 P1 为非凸优化问题。

2　混合任务两层优化算法

为了解决上述非凸优化问题 P1,本节提出了 MTTLO 算法,该算法第 1 层使用部分固定最优成本卸载算法,依据目标函数合理设置子任务成本寻找到当前最优的卸载位置;第 2 层则根据 KKT 条件求出服务缓存计算资源分配的封闭解。

2.1　部分固定最优成本卸载算法

部分固定最优成本卸载算法首先确定各个子任务的优先级。部分固定是指对于独立子任务的应用程序提前确定卸载位置得到传输时间,为此考虑最小化应用程序 $v_i(i\in N_{\mathrm{u,ind}})$ 成本的 P2 问题:

$$\begin{aligned}&\min_{y_i,t_i}\quad\theta t_i+(1-\theta)E_i;\\&\text{s.t.}\quad\sum_{k\in S_{i,j}}y_{k,F(v_{i,j})}^{i,j}=1,\ 1\leqslant j\leqslant N_i;\\&\sum_{j=1}^{N_i}(t_{i,k}^{i,j}+t_k^{i,j})y_{k,F(v_{i,j})}^{i,j}\leqslant t_i,k\in S_{i,j};\\&0\leqslant y_{k,F(v_{i,j})}^{i,j}\leqslant1,\ 1\leqslant j\leqslant N_i,k\in S_{i,j};\\&y_{k,s}^{i,j}=0,\ 1\leqslant j\leqslant N_i,k\in S_{i,j},s\neq F(v_{i,j});\\&y_{k,s}^{i,j}=0,\ 1\leqslant j\leqslant N_i,k\notin S_{i,j}\end{aligned}\tag{18}$$

其中:$y_{k,F(v_{i,j})}^{i,j}=\sum_{r=1}^{|V_{i,j}|}x_{k,F(v_{i,j}),r}^{i,j}$;$y_i$ 为 $v_i(i\in N_{\mathrm{u,ind}})$ 的卸载策略;t_i 为该应用程序的完成时间。通过求解上述线性优化问题,可以得到 $v_{i,j}$ 的对应卸载策略 $y_{k,F(v_{i,j})}^{i,j}$,将其视为概率,取概率最大对应的本地设备或协助节点作为卸载位置,即 $\tilde{k}_{i,j}=\underset{k\in S_{i,j}}{\operatorname{argmax}}\,y_{k,F(v_{i,j})}^{i,j}$,然后令 $y_{\tilde{k}_{i,j},F(v_{i,j})}^{i,j}=1$,当每个子任务卸载位置确定后,得到 $v_{i,j}(i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i)$ 实际传输到协助节点 $\tilde{k}_{i,j}$ 的时间为 $\sum_{m=1}^{N_i}y_{\tilde{k}_{i,j},F(v_{i,j})}^{i,m}t_{i,\tilde{k}_{i,j}}^{i,m}$。

将独立子任务 $v_{i,j}(i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i)$ 的优先级设置为:

$$r(i,j)=\theta\sum_{j=1}^{N_i}y_{\tilde{k}_{i,j},F(v_{i,j})}^{i,j}t_{\tilde{k}_{i,j}}^{i,j}+(1-\theta)E_{i,\tilde{k}_{i,j}}^{i,j}\tag{19}$$

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的平均传输时间定义为:

$$\bar{t}_{\mathrm{tran}}^{i,j}=\frac{1}{|S_{i,j-1}||S_{i,j}|}\sum_{k\in S_{i,j-1}}\sum_{l\in S_{i,j}}t_{k,l}^{i,j}\tag{20}$$

其中:$|S_{i,j}|$ 为拥有 $F(v_{i,j})$ 服务缓存的本地设备和协助节点的数量;当 $k=l$ 时,$t_{k,l}^{i,j}=0$。

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的平均计算时间为:

$$\bar{t}_{\mathrm{c}}^{i,j}=\frac{1}{|S_{i,j}|}\sum_{k\in S_{i,j}}t_k^{i,j}\tag{21}$$

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的平均传输能耗定义为:

$$\bar{E}_{\mathrm{tran}}^{i,j}=\frac{1}{|S_{i,j-1}||S_{i,j}|}\sum_{k\in S_{i,j-1}}\sum_{l\in S_{i,j}}E_{k,l}^{i,j}\tag{22}$$

当 $k=l$ 时,$E_{k,l}^{i,j}=0$。本文只考虑移动设备的传输能耗,因此当 $k\neq i$ 时,$E_{k,l}^{i,j}=0$。

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的平均计算能耗为:

$$\bar{E}_{\mathrm{c}}^{i,j}=\frac{1}{|S_{i,j}|}\sum_{k\in S_{i,j}}E_k^{i,j}\tag{23}$$

本文只考虑移动设备的能耗,因此当 $k\neq i$

时,$E_k^{i,j}=0$。

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的优先级可以由下式递推得到:

$$r(i,j)=r(i,j-1)+\theta(t_{\mathrm{c}}^{i,j-1}+t_{\mathrm{tran}}^{i,j})+(1-\theta)(\bar{E}_{\mathrm{c}}^{i,j-1}+\bar{E}_{\mathrm{tran}}^{i,j}) \tag{24}$$

其中,$r(i,0)=t_{\mathrm{c}}^{i,0}=\bar{E}_{\mathrm{c}}^{i,0}=0$。

将优先级按照从小到大排序得到任务卸载列表 L,根据列表先后顺序执行贪心算法。其思想是,对于独立子任务直接卸载到由式(18)恢复出的二进制解对应的本地设备或协助节点,而对于线性依赖子任务每次选择从当前子任务的前驱所处卸载位置到使得当前子任务成本最小的存在对应服务缓存的本地设备或协助节点。

当前线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的最优卸载位置为:

$$\tilde{k}_{i,j}=\underset{k\in S_{i,j}}{\operatorname{argmin}}\{\theta(t_k^{i,j}+\max\{t_{l,k}^{i,j},t_{k,F(v_{i,j})}^{\mathrm{avail}}\})+(1-\theta)(E_k^{i,j}+E_{l,k}^{i,j})\} \tag{25}$$

其中:l 表示子任务 $v_{i,j-1}$ 的卸载位置;$t_{k,F(v_{i,j})}^{\mathrm{avail}}$ 表示本地设备或协助节点 k 上服务缓存 $F(v_{i,j})$ 的空闲时间。

记 $\boldsymbol{T}_{\mathrm{s}}$ 为包含所有子任务计算开始时间的矩阵,矩阵元素 $t_{\mathrm{s}}^{i,j}$ 表示对应的子任务 $v_{i,j}$ 计算开始时间;记 $\boldsymbol{T}_{\mathrm{f}}$ 为包含所有子任务计算完成时间的矩阵,矩阵元素 $t_{\mathrm{f}}^{i,j}$ 为对应子任务 $v_{i,j}$ 的计算完成时间。

线性依赖子任务 $v_{i,j}(i\in N_{\mathrm{u,d}},1\leqslant j\leqslant N_i)$ 的计算开始时间 $t_{\mathrm{s}}^{i,j}$ 为:

$$t_{\mathrm{s}}^{i,j}=\max\{t_{\mathrm{f}}^{i,j-1}+t_{l,k}^{i,j},t_{k,F(v_{i,j})}^{\mathrm{avail}}\} \tag{26}$$

独立子任务 $v_{i,j}(i\in N_{\mathrm{u,ind}},1\leqslant j\leqslant N_i)$ 的计算开始时间为 $t_{\mathrm{s}}^{i,j}$:

$$t_{\mathrm{s}}^{i,j}=\max\Big\{\sum_{m=1}^{N_i}y_{k,j,F(v_{i,j})}^{i,m}\,t_{i,k_{i,j}}^{i,m},t_{k_{i,j},F(v_{i,j})}^{\mathrm{avail}}\Big\} \tag{27}$$

其中,$\tilde{k}_{i,j}$ 为式(18)恢复出 $y_{k,F(v_{i,j})}^{i,j}$ 二进制解对应的卸载位置。

子任务 $v_{i,j}(i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i)$ 的计算完成时间为 $t_{\mathrm{f}}^{i,j}$:

$$t_{\mathrm{f}}^{i,j}=t_{\mathrm{s}}^{i,j}+t_k^{i,j} \tag{28}$$

其中,当 $v_{i,j}$ 为线性依赖子任务时,k 为根据式(25)选择的卸载位置;当 $v_{i,j}$ 为独立子任务时,$k=\tilde{k}_{i,j}$。

部分固定最优成本卸载算法描述如下:

算法 1 部分固定最优成本卸载算法

输入:$N_{\mathrm{u}},N_{\mathrm{h}},p_k,f_k^{F(v_{i,j})},d_{i,j},w_{i,j},\kappa_k,B,N_0,h_{i,k}$

输出:任务卸载策略 x,总能耗 E,应用程序时延 t_i,系统成本 c,线性依赖子任务卸载位置 X

初始化 $r_{k,s}=0,t_{k,s}^{\mathrm{avail}}=0,E=0,X=0,x_{k,s,r}^{i,j}=0$

$X(i,0)=i$

根据式(18)求出 $y_{k,F(v_{i,j})}^{i,j}$,根据式(19)~(24)求出 r

将 r 从小到大排序得到任务卸载列表 L

当 $L\neq\varnothing$ 时,选择 L 第 1 个子任务 $v_{i,j}$

若 $i\in N_{\mathrm{u,ind}}$,则 $k=\underset{k\in S_{i,j}}{\operatorname{argmax}}\,y_{k,F(v_{i,j})}^{i,j}$

否则获取该子任务的前驱卸载位置 $l=X(i,j-1)$,k 为根据式(25)获得的卸载位置

$L=L\backslash\{v_{i,j}\}$,$r_{k,F(v_{i,j})}=r_{k,F(v_{i,j})}+1$,$x_{k,F(v_{i,j})}^{i,j}$,$r_{k,F(v_{i,j})}=1$,$X(i,j)=k$,根据式(26)或式(27)计算得到 $t_{\mathrm{s}}^{i,j}$,根据式(28)计算得到 $t_{\mathrm{f}}^{i,j}$,$t_{k,F(v_{i,j})}^{\mathrm{avail}}=t_{\mathrm{f}}^{i,j}$

若 $k=i$,则 $E=E+E_i^{i,j}$;若 $X(i,j-1)=i$ 且 $k\neq i$,则 $E=E+E_{i,k}^{i,j}$

若 $i\in N_{\mathrm{u,d}}$,则 $t_i=t_{\mathrm{f}}^{i,N_i+1}$;若 $i\in N_{\mathrm{u,ind}}$,则 $t_i=\max\limits_{1\leqslant j\leqslant N_i}t_{\mathrm{f}}^{i,j}$

$c=\theta\sum\limits_{i=1}^{M}t_i+(1-\theta)E$。

2.2 服务缓存计算资源分配

为了进一步减少系统计算时间,需要通过用户设备和协助节点多服务缓存资源的动态分配来实现。对于给定的任务卸载决策,考虑以下凸优化问题 P3:

$$\begin{aligned}&\min_{f}\sum_{k\in N_{\mathrm{u}}\cup N_{\mathrm{h}}}\sum_{i=1}^{K}\sum_{j=1}^{N_i}\sum_{s\in F_k}\sum_{r=1}^{|V(s)|}x_{k,s,r}^{i,j}\frac{d_{i,j}w_{i,j}}{f_k^s}\\&\text{s.t.}\quad\sum_{s\in F_k}f_k^s=f_k^{\max},\forall k\in N_{\mathrm{u}}\cup N_{\mathrm{h}}\end{aligned} \tag{29}$$

其中:f 为包含所有用户移动设备和协助节点的所有服务缓存的计算能力;$f_k^{\max}$ 为用户移动设备或协助节点 k 上移动设备或协助节点的最大计算能力;f_k^s 为用户移动设备或协助节点 k 上服务缓存 s 分配到的计算能力。当用户移动设备或协助节点 k 上服务缓存数量 $N_{\mathrm{F}k}$ 超过 1 时才能进行服务缓存的计算能力分配。

由于用户移动设备、协助节点上计算资源分配互不干扰,上述问题可分解单个用户移动或协助节点上最小化计算时间的凸优化问题。对于用户移动设备或协助节点 k,最小化计算时间的优化问题对应的拉格朗日函数为:

$$L(f_k,\lambda_k)=\sum_{i=1}^{K}\sum_{j=1}^{N_i}\sum_{s\in F_k}\sum_{r=1}^{|V(s)|}x_{k,s,r}^{i,j}\frac{d_{i,j}w_{i,j}}{f_k^s}+\lambda_k\Big(\sum_{s\in F_k}f_k^s-f_k^{\max}\Big) \tag{30}$$

其中:f_k 为用户移动设备或协助节点 k 上所有服务缓存计算能力;λ_k 为拉格朗日乘子。

由 KKT 条件可得:

$$\frac{\partial L}{\partial f_k^s}=-\sum_{i=1}^{K}\sum_{j=1}^{N_i}\sum_{r=1}^{|V(s)|}x_{k,s,r}^{i,j}\frac{d_{i,j}w_{i,j}}{(f_k^s)^2}+\lambda_k=0 \tag{31}$$

$$\frac{\partial L}{\partial \lambda_k}=\sum_{s\in F_k}f_k^s-f_k^{\max}=0 \tag{32}$$

从而求得 f_k^s 的封闭解如下：

$$f_k^s=\frac{\sqrt{\sum_{i=1}^{K}\sum_{j=1}^{N_i}\sum_{r=1}^{|V(s)|}x_{k,s,r}^{i,j}d_{i,j}w_{i,j}}}{\sum_{s\in F_k}\sqrt{\sum_{i=1}^{K}\sum_{j=1}^{N_i}\sum_{r=1}^{|V(s)|}x_{k,s,r}^{i,j}d_{i,j}w_{i,j}}}f_k^{\max} \tag{33}$$

当没有子任务卸载到 k 上时，$f_k^s(s\in F_k)$保持之前给定的分配策略。

3　仿真结果

本文通过仿真实验对本文所提算法与其他基准算法的性能进行了评估和比较。实验结果为 100 次执行的平均结果，实验考虑一个 100 m×100 m 的正方形区域，移动设备、空闲协助设备和边缘服务器随机分布在其中，用户移动设备数量 K 为 10，只划分独立子任务的用户设备数量为 5，只划分线性依赖子任务的用户设备数量为 5，空闲协助设备数量 $M-K$ 为 3，AP 的总带宽为 0.1 GHz，移动设备的计算能力 $f_i^{\max}(i\in N_{\mathrm{u}})$范围为 1.0～4.0 GHz，空闲协助设备的计算能力 $f_i^{\max}(i\in N_{\mathrm{h}}\backslash\{M+1\})$为 6.0～8.0 GHz，边缘服务器计算能力 $f_{M+1}^{\max}$为 15.0×10^9 Hz，有效开关电容 $\kappa_i(i\in N_{\mathrm{u}})$为 10^{-28}，只划分独立依赖子任务的用户移动设备和空闲协助设备的传输功率 p_i $(i\in N_{\mathrm{u,d}}\cup N_{\mathrm{h}})$为 0.1 W，只划分为独立子任务的用户移动设备的传输功率 $p_i(i\in N_{\mathrm{u,ind}})$为 $0.1/|S_{i,1}\backslash\{i\}|$ W，边缘服务器的传输功率 p_{M+1} 为 1 W。噪声的功率谱密度 N_0 为 −174 dBm/Hz，权重 θ 设置为 0.5。系统中服务缓存数量 J 为 4，移动设备上服务缓存数量为 $N_{\mathrm{F}i}(i\in N_{\mathrm{u}})$为 1，空闲协助设备服务缓存数量 $N_{\mathrm{F}i}(i\in N_{\mathrm{h}}\backslash\{M+1\})$为 2，边缘服务器服务缓存数量为 $N_{\mathrm{F}(M+1)}$为 4，协助节点上的服务缓存计算能力初始为计算能力平均分配。应用程序分解的子任务数量 $N_i(i\in N_{\mathrm{u}})$为 5。子任务 $d_{i,j}(i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i)$的大小为 125～375 KiB，子任务每 1 bit 数据量所需要的 CPU 周期 $w_{i,j}(i\in N_{\mathrm{u}},1\leqslant j\leqslant N_i)$为 150～250 CPU周期。

用于比较的基准算法如下：

1）轮询算法。轮流访问各个设备的子任务并以贪心算法获得对应的卸载决策。对于独立子任务通过线性规划提前确定卸载位置和传输时间，当轮到可分解为独立子任务的应用程序时，取计算量最小的独立子任务作为该设备本轮卸载的子任务。

2）随机算法。所有子任务按照随机卸载顺序，保持线性依赖顺序，其中独立子任务通过线性规划提前确定卸载位置和传输时间，按照卸载顺序通过贪心算法卸载到协助节点或本地计算。

3）全卸载到边缘服务器算法。每个应用程序选出一个当前卸载子任务，其中独立子任务选择计算量最小的子任务，线性依赖子任务选择当前尚未卸载的子任务；接着对上述子任务再选择传输时间和计算时间之和最小的子任务卸载到边缘服务器上对应的服务缓存，直至所有子任务卸载完成。

4）TBTOA 算法[15]。该算法是一种基于表的卸载算法，其思想为依赖性子任务构造优先级，得到任务卸载顺序表，按照顺序选择子任务，找当前最小成本的节点进行卸载。为了使 TBTOA 能够解决本文问题，不考虑能耗约束，先卸载依赖性子任务，再卸载独立子任务。其中，依赖性子任务使用 TBTOA 求出卸载策略，独立子任务根据 2.1 节线性规划得到传输时间和最优卸载位置，对所有独立子任务按照传输时间与计算时间之和从小到大顺序卸载到对应位置。

子任务数量对 5 种算法系统成本的影响如图 2所示。

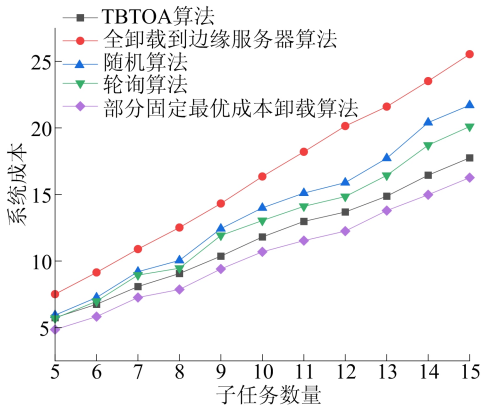

图 2　子任务数量对 5 种算法系统成本的影响

由图 2 可知，每个移动设备上的子任务数量从 5 增加到 15 时，5 种算法的系统成本也相应增加。全卸载到边缘服务器的系统成本最大，这是由于当所有任务全卸载到边缘服务器上时，计算排队时间将会增加从而影响系统成本。随机算法

可以选择空闲协助设备卸载,因此效果优于全卸载到边缘服务器。TBTOA 算法针对依赖性子任务的优化,优先级主要通过系统时延计算,无法将依赖性子任务和独立子任务优先级结合考虑,而部分固定最优成本卸载算法结合了依赖性子任务和独立子任务的优先级,每次卸载以当前子任务成本最小,与整个系统优化目标函数保持一致能够确定出合适的卸载位置。

子任务数量对应用 KKT 的 5 种算法系统成本的影响如图 3 所示。由图 3 可知,当算法结合 KKT 得到的系统成本随着子任务数量增加而增加,这是由于子任务数量增加带来的影响高于 KKT 优化计算时间带来的影响。与图 2 相比,系统成本略微降低,这是由于 KKT 条件优化更好地利用了本地设备和协助节点的计算资源,从而减少了每个子任务的计算时间。

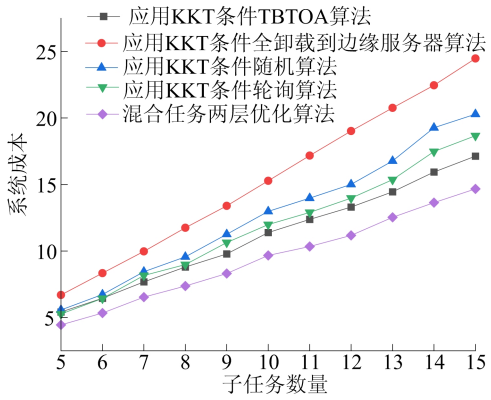

图 3 子任务数量对应用 KKT 的 5 种算法系统成本的影响

空闲协助设备数量对 5 种算法系统成本的影响如图 4 所示。由图 4 可知,全卸载到边缘服务器算法的系统成本也随着空闲协助数量增加而增大,这是由于随着空闲协助设备的数量增加,各个信道分到的带宽变小。其他算法的系统成本出现了波动,这是由于带宽变小和空闲协助设备带来新的计算资源的共同影响,当带宽影响弱于空闲协助设备的计算资源利用时,成本降低,反之上升。可以看出本文提出的算法仍然有较好的效果。尽管拥有更多卸载协助节点,但是 TBTOA 算法没有更好地考虑独立子任务和依赖子任务的卸载顺序,因此随着空闲协助设备数量的增加,TBTOA 算法系统成本增加较多。

空闲协助设备数量对应用 KKT 的 5 种算法系统成本的影响如图 5 所示。与图 4 相比,全卸载到边缘服务器算法系统成本不再随着空闲协助设备数量一直上升,反而出现了上下波动,表明 KKT 降低了其计算时间,因此 5 种算法成本都有所下降,混合两层优化算法使得系统成本处于较低水平。当空闲协助设备数量超过 7 时,TBTOA 算法效果最差,这是由于带宽变小,子任务没有卸载到合适位置,应用 KKT 条件优化效果不明显,进一步说明了将独立子任务和依赖性子任务共同考虑优先级的重要性。

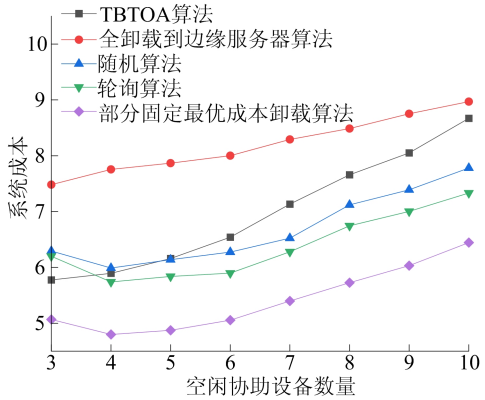

图 4 空闲协助设备数量对 5 种算法系统成本的影响

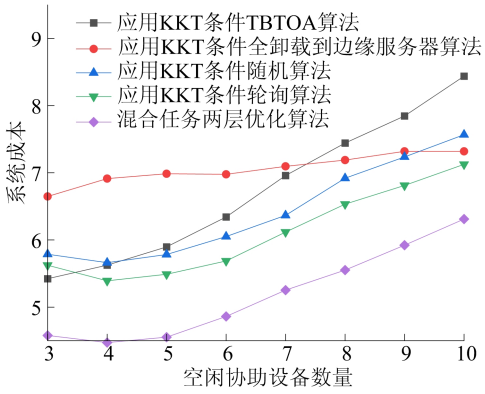

图 5 空闲协助设备数量对应用 KKT 的 5 种算法系统成本的影响

4 结 论

本文研究了 D2D 辅助单边缘服务器多移动设备下混合线性依赖和独立子任务的卸载与服务缓存计算能力资源分配问题,将其形式化为非凸优化问题,并提出了 MTTLO 算法。在给定服务缓存计算资源的情况下,首先通过部分固定最优成本卸载算法确定子任务的优先级,然后按照优先级的先后顺序使用贪心算法求出任务卸载策略。给定任务卸载策略,使用凸优化中 KKT 条件求出各个服务器上服务缓存的计算能力分配的最优解。仿真实验结果验证了所提算法的有效性。

(下转第 359 页)

[21] 陈治坤. 偏高岭土基地聚物的高温固化性能及常温固化机理研[D]. 长沙:长沙理工大学,2016.

[22] JIAO X,ZHANG Y,CHEN T. Thermal stability of a silica-rich vanadium tailing based geopolymer[J]. Construction and Building Materials,2013,38:43-47.

[23] 孔凡龙,刘泽,张俱嘉,等. 矿渣-粉煤灰基地质聚合物性能与微观结构的研究[J]. 电子显微学报,2016,35(3):229-234.

[24] LEE N K,LEE H K. Reactivity and reaction products of alkali-activated, fly ash/slag paste[J]. Construction and Building Materials,2015,81:303-312.

[25] MALDONADO-ALAMEDA A,GIRO-PALOMA J,ANDREOLA F,et al. Weathered bottom ash from municipal solid waste incineration:alkaline activation for sustainable binders[J]. Construction and Building Materials,2022,327:126983.

[26] PULIGILLA S,MONDAL P. Co-existence of aluminosilicate and calcium silicate gel characterized through selective dissolution and FTIR spectral subtraction[J]. Cement and Concrete Research,2015,70:39-49.

(责任编辑　吴　亮)

(上接第 334 页)

[参　考　文　献]

[1] LIU Y X,ZENG Z W,LIU X,et al. A novel load balancing and low response delay framework for edge-cloud network based on SDN[J]. IEEE Internet of Things Journal,2020,7(7):5922-5933.

[2] 张文杰,魏振春,徐俊逸,等. 移动边缘计算中的低能耗任务卸载决策算法[J]. 合肥工业大学学报(自然科学版),2020,43(6):770-776.

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

[4] SAHNI Y,CAO J N,YANG L,et al. Multi-hop multi-task partial computation offloading in collaborative edge computing[J]. IEEE Transactions on Parallel and Distributed Systems,2021,32(5):1133-1145.

[5] YAN J,BI S Z,ZHANG Y J,et al. Optimal task offloading and resource allocation in Mobile-Edge Computing with inter-user task dependency[J]. IEEE Transactions on Wireless Communications,2020,19(1):235-250.

[6] DING Y,LIU C B,ZHOU X,et al. A code-oriented partitioning computation offloading strategy for multiple users and multiple Mobile Edge Computing servers[J]. IEEE Transactions on Industrial Informatics,2020,16(7):4800-4810.

[7] CHEN M G,GUO S T,LIU K,et al. Robust computation offloading and resource scheduling in cloudlet-based Mobile Cloud Computing[J]. IEEE Transactions on Mobile Computing,2021,20(5):2025-2040.

[8] ZHANG H X,YANG Y J,SHANG B D,et al. Joint resource allocation and multi-part collaborative task offloading in MEC systems[J]. IEEE Transactions on Vehicular Technology,2022,71(8):8877-8890.

[9] CHAI R,LIN J L,CHEN M L,et al. Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems[J]. IEEE Systems Journal,2019,13(4):4110-4121.

[10] WANG X,HAN Y B,SHI H T,et al. Joagt:latency-oriented joint optimization of computation offloading and resource allocation in D2D-assisted MEC system[J]. IEEE Wireless Communications Letters,2022,11(9):1780-1784.

[11] ZHAO G M,XU H L,ZHAO Y M,et al. Offloading tasks with dependency and service caching in Mobile Edge Computing[J]. IEEE Transactions on Parallel and Distributed Systems,2021,32(11):2777-2792.

[12] ZHONG X X,WANG X H,YANG T T,et al. Potam:a parallel optimal task allocation mechanism for large-scale delay sensitive Mobile Edge Computing[J]. IEEE Transactions on Communications,2022,70(4):2499-2517.

[13] ABBAS N,FAWAZ W,SHARAFEDDINE S,et al. SVM-based task admission control and computation offloading using Lyapunov optimization in heterogeneous MEC network[J]. IEEE Transactions on Network and Service Management,2022,19(3):3121-3135.

[14] XIE J D,JIA Y J,WEN W L,et al. Dynamic D2D multihop offloading in multi-access edge computing from the perspective of learning theory in games[J]. IEEE Transactions on Network and Service Management,2023,20(1):305-318.

[15] LV X Y,DU H W,YE Q. TBTOA:a DAG-based task offloading scheme for Mobile Edge Computing[C]//Proceedings of the 2022 IEEE International Conference on Communications. [S. l.]:IEEE,2022:4607-4612.

(责任编辑　李　凯)