设备间任务依赖的最佳卸载决策和资源分配

胡 恒[1],金凤林[1*],谢 钧[1],俞 璐[2],黄科瑾[3],孟繁伦[4],杨 涛[1]
(1. 陆军工程大学 指挥控制工程学院,江苏 南京 210007;
2. 陆军工程大学 通信工程学院,江苏 南京 210007;
3. 31121 部队,江苏 南京 210018;
4. 61096 部队,江苏 南京 210007)

摘 要:对于不同设备之间具有任务依赖性的问题,考虑了两个设备的移动边缘计算(Mobile Edge Computing,MEC)与端对端(Device-to-Device,D2D)技术协作网络,其中一个无线设备的最终输出作为另一个设备上某个任务的输入。在此任务依赖模型下,为了最小化无线设备的能耗和任务完成时间的加权和,研究了最佳的资源分配(卸载发射功率和本地 CPU 频率)和任务卸载决策问题。为了解决该问题,将原问题分解为给定任务卸载决策的资源分配问题和优化与资源分配问题相对应的任务卸载问题。首先给定卸载决策,推导出卸载发射功率和本地 CPU 频率的闭合表达式,运用凸优化方法求出该问题的解。然后证明最优卸载决策遵循一次爬升策略,在此基础上提出了一种降低复杂度的在线任务卸载算法,该算法可以在多项式时间内获得最优卸载决策。数值结果表明,该策略的性能明显优于其他有代表性的基准测试,同时 MEC 与 D2D 协作可以显着提高系统的性能。

关键词:移动边缘计算;D2D 技术;计算卸载技术;卸载决策;资源分配
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2022)08-0082-07
doi:10.3969/j.issn.1673-629X.2022.08.014

Optimal Offloading Decision and Resource Allocation of Inter-devices Task Dependency

HU Heng[1],JIN Feng-lin[1*],XIE Jun[1],YU Lu[2],HUANG Ke-jin[3],MENG Fan-lun[4],YANG Tao[1]
(1. School of Command & Control Engineering,Army Engineering University of PLA,Nanjing 210007,China;
2. School of Communication Engineering,Army Engineering University of PLA,Nanjing 210007,China;
3. 31121 Troops,Nanjing 210018,China;
4. 61096 Troops,Nanjing 210007,China)

Abstract:For the problem of task dependence between different devices,the mobile edge computing (MEC) and device-to-device (D2D) technology cooperation network of two devices are considered. The final output of one wireless device is the input of a task on another device. In this task-dependent model,in order to minimize the weighted sum of energy consumption of wireless devices and task completion time,the optimal resource allocation (offloading transmitting power and local CPU frequency) and task offloading decision-making are studied. In order to solve this problem,the original problem is decomposed into a resource allocation problem for a given task offloading decision and a task offloading problem corresponding to the optimization and resource allocation problem. Firstly,given the offloading decision,the closed-form expressions of the offloading transmission power and the local CPU frequency are derived,and the convex optimization method is used to solve the problem. Then it is proved that the optimal offloading decision follows a climbing strategy,and on this basis,an online task offloading algorithm with reduced complexity is proposed,which can obtain the optimal offloading decision in polynomial time. The numerical results show that the performance of this strategy is significantly better than that of other representative benchmark tests,and the collaboration between MEC and D2D can significantly improve the performance of the system.

Key words:mobile edge computing;D2D technology;computation offloading technology;offloading decision-making;resource allocation

收稿日期:2021-07-29 **修回日期**:2021-11-30

作者简介:胡 恒(1991-),男,硕士研究生,研究方向为移动边缘计算;通讯作者:金凤林(1972-),男,CCF 会员(E200026696M),硕导,副教授,研究方向为计算机网络、卫星通信。

0　引　言

随着物联网技术和 5G 技术的发展,产生的各种新型应用,对网络的时延和带宽提出了更高的要求。但由于终端设备的计算和存储能力的限制,无法处理和存储如此庞大的数据。因此移动边缘计算[1]应运而生,MEC 能有效解决时延长、能耗高和数据不安全等问题。其中计算卸载技术[2]作为 MEC 的关键技术之一,通过合理的卸载决策和资源分配策略将终端设备上运行的任务卸载到边缘服务器,能够减少任务完成时延和设备的能耗,提高设备性能。而通过将 D2D 技术与 MEC 结合,可以更进一步降低数据传输时延和能耗。

对于计算卸载技术,目前已经出现了很多研究成果。文献[3]提出了一种低复杂度的启发式算法来最小化共享频谱中的任务执行延迟。文献[4]考虑了具有辅助节点的 MEC 系统,联合优化了用户和辅助节点的计算和通信资源分配,使总能耗降至最低。文献[5]基于一种低复杂度的算法来降低设备能耗。文献[6]使用线性规划的方法解决卸载决策、延迟和能耗联合优化问题。文献[7]为了最大程度地减少任务等待时间和能耗,提出了一种启发式算法,该算法保证了子任务之间的依赖性并提高了任务效率。文献[8]提出了一种基于 Lyapunov 优化的低复杂度的动态计算卸载在线算法,获得了最优的卸载策略。文献[9]考虑了包含多个忙碌智能设备和多个空闲智能设备的 D2D 与 MEC 协作系统,基于块坐标下降法和凸优化技术的两阶段迭代算法解决卸载决策和资源分配的联合优化问题,获得最佳卸载策略和资源分配策略,最小化系统的总能耗。文献[10]建立了一个具有通信资源和计算资源约束的混合整数非线性问题,开发了一种优化算法来解决此问题。文献[11-13]同样考虑了在 MEC 环境下引入 D2D 技术进行协作,来考虑任务的卸载情况。

然而以上文献都仅考虑了单个设备上任务的卸载情况。对于不同设备之间任务具有依赖性的研究非常少,几乎没有。实际上,在不同设备上执行的任务通常也是具有相关依赖性的。文献[14]虽然考虑了 MEC 系统环境下两个不同设备之间的任务相关性,但没有考虑与 D2D 技术进行协作来优化系统性能。虽然业内也在 MEC 环境中引入了 D2D 技术来优化系统性能,但是对于 MEC 与 D2D 技术协作网络环境中不同设备任务之间任务具有相关性的研究,就作者目前所知,还没有相关文献对此方面进行研究。因此该文针对此方面,同时在文献[14]的基础上进行了研究,联合优化了设备任务完成时延和能耗,在任务执行时延和设备能耗之间进行权衡,寻找一个平衡点使得系统性能最优。

1　系统模型

1.1　网络模型

考虑了两个设备的 MEC 与 D2D 协作网络。图 1 为 MEC 与 D2D 协作网络系统模型,其中 d_1 和 d_2 分别表示设备 1、设备 2 与 MEC 服务器之间的距离, d_3 表示设备 1 和设备 2 之间的距离。

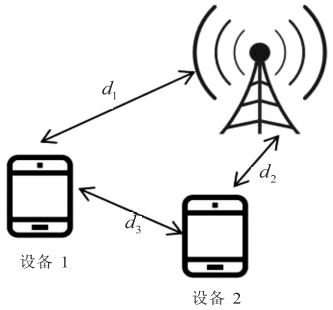


图 1　MEC 与 D2D 协作网络系统模型

设备 1 和设备 2 分别具有 m 和 n 个要执行的一系列任务,并将任务之间的依赖关系建模为顺序图,同时对每个设备引入两个虚拟任务,0 为设备的输入任务, $m+1$ 和 $n+1$ 为输出任务,如图 2 所示。其中设备 2 的第 k 个子任务的输入依赖于设备 1 第 m 个任务的输出和设备 2 第 $k-1$ 个任务的输出。

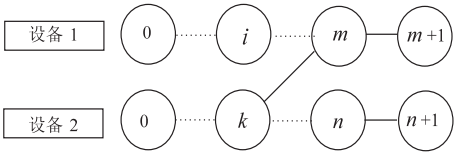


图 2　不同设备间调用图模型

该文采用 $\{L_{i,j}, I_{i,j}, O_{i,j}\}$ 来描述每个设备的每个任务,其中 $j=1$ 表示设备 1, $j=2$ 表示设备 2。i 表示设备上的任务, $i=0,1,\cdots,m+1$ 或 $0,1,\cdots,n+1$。$L_{i,j}$ 表示任务的计算量,完成任务总共需要的 CPU 周期, $I_{i,j}$ 表示计算任务的输入数据的大小, $O_{i,j}$ 表示任务的输出数据的大小。对于设备 1 和设备 2 第 0 个和最后一个虚拟任务计算量为 0。同时对于两个设备而言,第 0 个任务输入数据的大小为 0,最后一个任务输出数据的大小为 0。对于设备 1 和设备 2 第 i 个任务的输入等于上一个任务的输出: $I_{i-1,j}=O_{i-1,j}$, $j=\{1,2\}$, $i=1,2,\cdots,m,m+1$ 或 $1,\cdots,k-1,k+1,\cdots,n+1$。特别的是,对于设备 2 的第 k 个任务的输入依赖于设备 2 的第 $k-1$ 个任务和设备 1 第 m 个任务的输出: $I_{i,2}=O_{i-1,2}+O_{m,1}$。该文规定第 0 个和最后一个虚拟任务只能在本地执行。同时,用 $a_{i,j}=\{0,1\}$ 表示卸载决策: $a_{i,j}=0$ 表示在本地执行, $a_{i,j}=1$ 表示在服务器执行。

1.2　通信模型

首先介绍了系统的通信模型。对于每个设备分配

带宽相等的正交信道,用 W 表示,卸载和上传设备之间互不干扰。

上传速率为:

$$R_{i,j}^{u} = W\log_2(1 + \frac{p_{i,j}\,h_{i,j}}{\sigma^2}) \tag{1}$$

其中,$p_{i,j}$ 表示设备 j 的发射功率,$h_{i,j}$ 表示设备 j 到边缘服务器的信道增益,σ^2 表示设备的噪声功率。

下载速率为:

$$R_{i,j}^{d} = W\log_2(1 + \frac{p_0\,h_{i,j}}{\sigma^2}) \tag{2}$$

其中,p_0 表示边缘服务器的固定发射功率。

设备 1 与设备 2 之间的传输速率为(设备 1 第 m 个任务的输出与设备 2 的第 k 个任务有关):

$$R^{1,2} = W\log_2(1 + \frac{p'_{m+1,1}\,h'_{m+1,1}}{\sigma^2}) \tag{3}$$

其中,$p'_{m+1,1}$ 和 $h'_{m+1,1}$ 分别表示设备 1 第 m 个任务的输出与设备 2 的第 k 个任务都在本地执行时,设备 1 的发射功率和此时设备间的信道增益。

信道增益为:

$$h_{i,j} = g\,(\frac{3 * 10^8}{4\pi F_c\,d_r})^{\mathrm{pl}} \tag{4}$$

其中,g 表示天线增益,F_c 表示载波频率,d_r 表示距离,pl 表示路径损耗指数。

1.3 计算模型

下面给出了执行和传输任务的时间和能耗。

本地执行任务 i 的完成时间为:

$$t_{i,j}^{l} = \frac{L_{i,j}}{f_{i,j}} \tag{5}$$

其中,$f_{i,j}$ 表示执行任务 i 的 CPU 计算能力。

本地完成任务 i 所损失的能耗为:

$$e_{i,j}^{l} = k\,L_{i,j}f_{i,j}^{2} \tag{6}$$

其中,k 是取决于芯片架构的有效开关电容参数,是固定常数。

服务器执行任务 i 的完成时间为:

$$t_{i,j}^{c} = \frac{L_{i,j}}{f_0} \tag{7}$$

其中,f_0 表示服务器的恒定 CPU 频率。

将任务 i 从设备传输到服务器的上传时间为:

$$t_{i,j}^{u} = \frac{O_{i-1,j}}{R_{i,j}^{u}} \tag{8}$$

其中,$p_{i,j} = \dfrac{\sigma^2}{h_{i,j}}(2^{\frac{O_{i-1,j}}{Wt_{i,j}^{u}}} - 1)$。

将任务 i 从设备传输到服务器的传输能耗为:

$$\mathrm{e}_{i,j}^{u} = p_{i,j}\,t_{i,j}^{u} = \frac{\sigma^2\,t_{i,j}^{u}}{h_{i,j}}(2^{\frac{O_{i-1,j}}{Wt_{i,j}^{u}}} - 1) \tag{9}$$

由文献[15]可知式(9)是关于 $t_{i,j}^{u}$ 的凸函数。

从服务器返回到设备的下载时间为:

$$t_{i,j}^{d} = \frac{O_{i-1,j}}{R_{i,j}^{d}} \tag{10}$$

将任务从设备 1 传输到设备 2 的传输时间为:

$$t_{m+1,1}^{1,2} = \frac{O_{m,1}}{R^{1,2}} \tag{11}$$

其中,$p'_{m+1,1} = \dfrac{\sigma^2}{h'_{m+1,1}}(2^{\frac{O_{m,1}}{Wt_{m+1,1}^{1,2}}} - 1)$。

将任务从设备 1 传输到设备 2 的传输能耗为:

$$\mathrm{e}_{i,j}^{1,2} = p'_{m+1,1}\,t_{m+1,1}^{1,2} = \frac{\sigma^2\,t_{m+1,1}^{1,2}}{h'_{m+1,1}}(2^{\frac{O_{m,1}}{Wt_{m+1,1}^{1,2}}} - 1) \tag{12}$$

1.4 依赖模型

由于设备 1 的第 m 个任务的输出作为设备 2 第 k 个任务的输入,所以要综合考虑设备 1 第 m 个任务和设备 2 第 k 个任务的位置关系:当 m 任务和 k 任务都在本地执行时,由于两设备都支持 D2D 技术,所以设备之间可以直接通信,有传输时延和能耗;当 m 任务和 k 任务都在边缘服务器上执行时,此时没有时间和能耗的损耗;当 m 任务在本地执行,k 任务在边缘服务器执行时,此时需要上传时延和能耗;当 m 任务在边缘服务器执行,k 任务在本地执行时,此时需要下载时延。

1.5 问题公式化

由以上分析可知,设备 1 的任务完成时间为:

$$\begin{aligned} T_1 = &\sum_{i=1}^{m}\left[(1 - a_{i,1})\,t_{i,1}^{l} + a_{i,1}\,t_{i,1}^{c}\right] + \\ &\sum_{i=1}^{m+1}\left[a_{i,1}(1 - a_{i-1,1})\,t_{i,1}^{u} + a_{i-1,1}\,(1 - a_{i,1})t_{i,1}^{d}\right] \end{aligned} \tag{13}$$

由于设备间任务的依赖关系,在公式(14)的最后一行描述了设备间任务的依赖关系。当设备 1 第 m 个任务和设备 2 的第 k 个任务都在本地执行时,则需要传输能耗 $e_{m+1,1}^{1,2}$;当设备 1 第 m 个任务在本地执行,而设备 2 的第 k 个任务在服务器上执行时,则需要传输能耗 $e_{m+1,1}^{u}$。因此设备 1 的能耗为:

$$\begin{aligned} E_1 = &\sum_{i=1}^{m}\left[(1 - a_{i,1})\,e_{i,1}^{l} + a_{i,1}\,(1 - a_{i-1,1})e_{i,1}^{u}\right] + \\ &(1 - a_{m,1})(1 - a_{k,2})\,e_{m+1,1}^{1,2} + \\ &a_{k,2}(1 - a_{m,1})\,e_{m+1,1}^{u} \end{aligned} \tag{14}$$

由于设备 2 的第 k 个任务输入依赖于设备 1 的第 m 个任务和设备 2 第 $k-1$ 任务的输出,当设备 1 第 m 个任务和设备 2 的第 k 个任务都在本地执行时,需要传输时间 $t_{m+1,1}^{1,2}$;当设备 1 第 m 个任务在服务器执行,而设备 2 的第 k 个任务在本地执行时,则需要下载时间 $O_{m,1}/R_{k,2}^{d}$;当设备 1 第 m 个任务在本地执行,而设备 2 的第 k 个任务在服务器上执行时,则需要上传时

间 $t^u_{m+1,1}$。因此设备 2 的第 k 个任务等待设备 1 的第 m 个任务的输出时间为:

$$T_1^{\text{wait}} = \sum_{i=1}^{m}\left[(1-a_{i,1})\,t^l_{i,1} + a_{i,1}\,t^c_{i,1}\right] + \sum_{i=1}^{m}\left[a_{i,1}(1-a_{i-1,1})\,t^u_{i,1} + a_{i-1,1}(1-a_{i,1})t^d_{i,1}\right] + (1-a_{m,1})(1-a_{k,2})\,t^{1,2}_{m+1,1} + a_{k,2}(1-a_{m,1})\,t^u_{m+1,1} + a_{m,1}(1-a_{k,2})\,\frac{O_{m,1}}{R^d_{k,2}} \tag{15}$$

等待设备 2 第 $k-1$ 个任务的输出时间为:

$$T_2^{\text{wait}} = \sum_{i=1}^{k-1}\left[(1-a_{i,2})\,t^l_{i,2} + a_{i,2}\,t^c_{i,2}\right] + \sum_{i=1}^{k}\left[a_{i,2}(1-a_{i-1,2})\,t^u_{i,2} + a_{i-1,2}\,(1-a_{i,2})t^d_{i,2}\right] \tag{16}$$

令 $t = \max\{T_1^{\text{wait}}, T_2^{\text{wait}}\}$

设备 2 的任务完成时间为:

$$T_2 = \max\{T_1^{\text{wait}}, T_2^{\text{wait}}\} + \sum_{i=k}^{n}\left[(1-a_{i,2})\,t^l_{i,2} + a_{i,2}\,t^c_{i,2}\right] + \sum_{i=k+1}^{n+1}\left[a_{i,2}(1-a_{i-1,2})\,t^u_{i,2} + a_{i-1,2}\,(1-a_{i,2})t^d_{i,2}\right] \tag{17}$$

设备 2 消耗的能耗为:

$$E_2 = \sum_{i=1}^{n}\left[(1-a_{i,2})\,e^l_{i,2} + a_{i,2}\,(1-a_{i-1,2})e^u_{i,2}\right] \tag{18}$$

两个设备总性能指标为:

$$\eta = \eta_1 + \eta_2 = \beta_1^T\,T_1 + \beta_1^E\,E_1 + \beta_2^T\,T_2 + \beta_2^E\,T_2 \tag{19}$$

其中,β_1^T,β_1^E,β_2^T,β_2^E 分别表示设备的能耗和时间的权重,且权重之间存在以下关系:

$$0<\beta_1^T<1, 0<\beta_1^E<1, 0<\beta_2^T<1, 0<\beta_2^E<1$$
$$\beta_1^E = 1-\beta_1^T, \quad \beta_2^E = 1-\beta_2^T \tag{20}$$

所以问题公式化为以下公式:

$$\text{p(1)} \min_{(\{a_{i,j}\},\{p_{i,j}\},\{f_{i,j}\})} \eta_1 + \eta_2$$
$$\text{s.t.}$$
$$0 \leqslant p_{i,j} \leqslant P_{\text{peak}}$$
$$0 \leqslant p'_{m+1,1} \leqslant P_{\text{peak}}$$
$$0 \leqslant f_{i,j} \leqslant f_{\text{peak}} \tag{21}$$
$$a_{i,j} \in \{0,1\}, \forall i,j$$

其中,P_{peak} 表示设备的发射功率峰值,f_{peak} 表示设备 CPU 计算能力的峰值。由公式(5)、(8)、(11)可知 $f_{i,j}, p_{i,j}, p'_{m+1,1}$ 与 $t^l_{i,j}, t^u_{i,j}, t^{1,2}_{m+1,1}$ 一一对应,所以 p(1) 公式可以转化为:

$$\text{p(2)} \min_{(\{a_{i,j}\},\{t^l_{i,j}\},\{t^u_{i,j}\},\{t^{1,2}_{m+1,1}\},t)} \eta_1 + \eta_2$$
$$\text{s.t.}$$
$$T_1^{\text{wait}} \leqslant t, \quad T_2^{\text{wait}} \leqslant t, \quad \frac{L_{i,j}}{f_{\text{peak}}} \leqslant t^l_{i,j}$$
$$\frac{O_{i-1,j}}{W\log_2(1+\frac{p_{\text{peak}}\,h_{i,j}}{\sigma^2})} \leqslant t^u_{i,j}$$
$$\frac{O_{m,1}}{W\log_2(1+\frac{p_{\text{peak}}\,h'_{m+1,1}}{\sigma^2})} \leqslant t^{1,2}_{m+1,1} \tag{22}$$
$$a_{i,j} \in \{0,1\}, \forall i,j$$

由于含有二进制变量 $a_{i,j}$,所以 p(2) 问题是非凸的,但当 $a_{i,j}$ 固定时,原来的非凸问题就会转化为关于 $t^l_{i,j}, t^u_{i,j}, t^{1,2}_{m+1,1}$ 的凸问题。

2　固定卸载决策的最佳资源分配

假设固定 $a_{i,j}$,则 p(2) 非凸问题转化为关于 $t^l_{i,j}$, $t^u_{i,j}, t^{1,2}_{m+1,1}$ 的凸问题 p(3)。

问题 p(3) 的拉格朗日表示为:

$$L(\{t^u_{i,j}\},\{t^l_{i,j}\},\{t^{1,2}_{m+1,1}\},t,\lambda,\mu) = \eta_1+\eta_2+\lambda(T_1^{\text{wait}}-t)+\mu(T_2^{\text{wait}}-t) \tag{23}$$

其中,λ 和 μ 都是不小于零的表示与相应约束相关联的对偶变量,λ^* 和 μ^* 表示最佳对偶变量,则能推导出每个设备的最佳 CPU 频率和发射功率的闭合表达式如下:

$$\begin{cases} f^*_{i,1} = \begin{cases} f_{\text{peak}}, \text{如果} \beta_1^T + \lambda^* > 2k\beta_1^E f_{peak}^3 \\ \sqrt[3]{\dfrac{\beta_1^T+\lambda^*}{2k\beta_1^E}}, \text{否则} \end{cases}, \text{当 } i \in \{1,\cdots,m\} \text{ 时} \\ f^*_{i,2} = \begin{cases} \begin{cases} f_{\text{peak}}, \text{如果} \mu^* > 2k\beta_2^E f_{peak}^3 \\ \sqrt[3]{\dfrac{\mu^*}{2k\beta_2^E}}, \text{否则} \end{cases}, \text{当 } i \in \{1,\cdots,k-1\} \text{ 时} \\ \begin{cases} f_{\text{peak}}, \text{如果} \beta_2^T > 2k\beta_2^E f_{peak}^3 \\ \sqrt[3]{\dfrac{\beta_2^T}{2k\beta_2^E}}, \text{否则} \end{cases}, \text{当 } i \in \{k,\cdots,n\} \text{ 时} \end{cases} \end{cases} \tag{24}$$

$$\begin{cases} p_{i,1}^{*}=\begin{cases} P_{\text{peak}},\text{如果 } h_{i,1}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_1}{-W(-A_1e^{-A_1})}-1\right] \\ \dfrac{\sigma^2}{h_{i,1}}\left[\dfrac{B_1}{W(B_1e^{-1})}-1\right],\text{否则} \end{cases} ,\text{当 } i\in\{1,\cdots,m\} \text{ 时} \\ p_{i,1}^{*}=\begin{cases} P_{\text{peak}},\text{如果} h_{i,1}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_2}{-W(-A_2e^{-A_2})}-1\right] \\ \dfrac{\sigma^2}{h_{i,1}}\left[\dfrac{B_2}{W(B_2e^{-1})}-1\right],\text{否则} \end{cases} ,\text{当 } i=m+1 \text{ 时} \\ \text{或者} p_{i,1}^{*}=\begin{cases} P_{\text{peak}},\text{如果} h_{i,1}^{'}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_2}{-W(-A_2e^{-A_2})}-1\right] \\ \dfrac{\sigma^2}{h_{i,1}^{'}}\left[\dfrac{B_2^{'}}{W(B_2^{'}e^{-1})}-1\right],\text{否则} \end{cases} ,\text{当 } i=m+1 \text{ 时} \end{cases}$$

$$\begin{cases} p_{i,2}^{*}=\begin{cases} P_{\text{peak}},\text{如果 } h_{i,2}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_3}{-W(-A_3e^{-A_3})}-1\right] \\ \dfrac{\sigma^2}{h_{i,2}}\left[\dfrac{B_3}{W(B_3e^{-1})}-1\right],\text{否则} \end{cases} \\ \quad \text{当 } i\in\{1,\cdots,k\} \text{ 时} \\ p_{i,2}^{*}=\begin{cases} P_{\text{peak}},\text{如果 } h_{i,2}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_4}{-W(-A_4e^{-A_4})}-1\right] \\ \dfrac{\sigma^2}{h_{i,2}}\left[\dfrac{B_4}{W(B_4e^{-1})}-1\right],\text{否则} \end{cases} \\ \quad \text{当 } i\in\{k+1,\cdots,n\} \text{ 时} \end{cases} \tag{25}$$

其中，

$$A_1=1+\frac{\beta_1^T+\lambda^*}{\beta_1^E P_{\text{peak}}},B_1=\frac{h_{i,1}(\beta_1^T+\lambda^*)}{\beta_1^E\sigma^2}-1$$

$$A_2=1+\frac{\lambda^*}{\beta_1^E P_{\text{peak}}},B_2=\frac{h_{i,1}\lambda^*}{\beta_1^E\sigma^2}-1,B_2^{'}=\frac{h_{i,1}^{'}\lambda^*}{\beta_1^E\sigma^2}-1$$

$$A_3=1+\frac{\mu^*}{\beta_2^E P_{\text{peak}}},B_3=\frac{h_{i,2}\mu^*}{\beta_2^E\sigma^2}-1$$

$$A_4=1+\frac{\beta_2^T}{\beta_2^E P_{\text{peak}}},B_4=\frac{h_{i,2}\beta_2^T}{\beta_2^E\sigma^2}-1$$

$W(x)$ 表示 LambertW 函数，应用梯度下降法迭代更新 λ 和 μ，直到满足一定的停止标准。该方法的伪代码如算法 1 所示。由于 P(3) 是一个固定 $a_{i,j}$ 的凸问题，梯度下降法保证收敛。

算法 1：最佳资源分配算法。

1 输入：给定大于 0 的 λ 和 μ、步长 α 和精度值 pre

2 输出：p^*，f^*

3 while True：

4 根据最佳 CPU 频率的闭合表达式(24)计算 $f_{i,j}$

5 根据最佳发射功率的闭合表达式(25)计算 $p_{i,j}$

6 根据 $f_{i,j}$ 和 $p_{i,j}$ 分别计算 T_1^{wait}、T_2^{wait} 和目标值

7 $t=\max\{T_1^{\text{wait}},T_2^{\text{wait}}\}$

8 根据梯度下降算法，分别更新 λ 和 μ 的值

9 如果 $|T_1^{\text{wait}}-T_2^{\text{wait}}|<\text{pre}$：

10 退出循环

11 End

3 优化卸载决策

在第 2 小节中，给定卸载决策，可以得出最佳资源分配，需要搜索 $2^{(m+n)}$ 次卸载决策，然后从中选择最小目标的卸载决策。但是随着任务 m 或 n 的增多，这种遍历搜索方法是行不通的，复杂度会很高。基于此提出了一种降低复杂度的在线任务卸载算法，可以在多项式时间内获得最优卸载决策。本节首先证明最优卸载决策具有连续性，然后在此基础上提出了一种降低复杂度的优化算法。

3.1 一次爬升策略

定理 1：假设 $f_0>f_{\text{peak}}$，则最优卸载决策具有连续性(如果有任务要卸载)，即对于最优决策，如果设备有任务需要卸载到边缘服务器，那么在整个任务的执行过程中，任务卸载到边缘服务器只会发生一次，它被称为一次爬升策略。

证明：略。

3.2 基于一次爬升策略的在线搜索算法

基于一次爬升策略，提出了低复杂度的在线任务卸载算法。通过此算法不必暴力遍历所有可行的卸载决策，只需在每个设备上寻找满足使得设备能耗和时间加权和最小这个指标要求的两个任务即可。如图 3 所示，找到满足指标要求的两个任务 i 和 j，然后只需将 i 和 j，以及中间的任务都卸载到边缘服务器即可。

图 3 一次爬升策略

因此只需寻找这样两个任务，使得设备的能耗和时间的加权和 $\eta(i,j)$ 最小。这里设定任务 i^* 为入口任务，任务 j^* 为退出任务，$i=1,2,\cdots,m,j=1,2,\cdots,$

n ,设备的能耗和时间加权和公式如下:

$$\eta(i,j)=\beta^{E}\Big[\sum_{h=1}^{i-1}e_h^l+e_{i-1,i}^u+\sum_{k=j+1}^{m}e_k^l\Big]+\beta^{T}\Big[\sum_{h=1}^{i-1}t_h^l+t_{i-1,i}^u+\sum_{k=i}^{j}t_k^c+t_{j,j+1}^d+\sum_{k=j+1}^{m}t_k^l\Big] \tag{26}$$

其中, e^u 为传输能耗, e^l 为本地能耗, t^u 为传输时间, t^c 为在服务器上的执行时间, t^d 为下载时间, t^l 为本地执行时间, β^E 和 β^T 分别为能耗和时延权重, $\eta(i,j)$ 为加权和, $\eta_{\min}$ 为最小加权和。

算法 2 给出了寻找最优的入口任务 i^* 和退出任务 j^* 的算法,初始时,任务全部在本地执行。因此只需要枚举满足一次爬政策的卸载决策,而不必遍历所有 2^{m+n} 个卸载决策,即只需在每个设备上寻找满足使得设备能耗和时间加权和最小这个指标要求的两个任务即可。对于设备 1,根据算法 2 只需搜索 $m*(1+m)/2$ 个任务组合。类似的,设备 2 也只需要搜索 $n*(1+n)/2$ 个任务组合。因此搜索总数[$m*(1+m)/2*n*(1+n)/2$],即 $O(m^2*n^2)$ 。随着 m 或 n 的增大,基于单爬策略的在线搜索算法的复杂度明显低于暴力搜索算法。

算法 2:在线搜索算法。

1 输入:给定 β^E , β^T 值
2 输出: $\eta_{\min}$, i^* , j^*
3 根据加权和公式计算 $\eta(1,1)$,令 $\eta_{\min}=\eta(1,1)$, $i^*=1$, $j^*=1$
4 For　i <----1 to m :
5 For　j <--- i to m :
6 计算出最优 f^* 和 p^* ,然后计算设备的能耗和完成时间
7 根据能耗、时间和加权和公式计算设备的加权和 $\eta(i,j)$
8　　If $\eta(i,j)<\eta_{\min}$:
9 $\eta_{\min}=\eta(i,j)$
10 $i^*=i$
11 $j^*=j$
12　End
13 End

4　实验评估

在这一小节,将进行数值模拟来评估所提的卸载算法,关注的性能指标是两设备的总性能,用 ET 表示,其中每个设备的性能用设备完成任务所消耗的能耗和时间的加权和表示。为了方便与文献[14]算法进行比较,实验数据值的大小参考文献[14]进行了设置。接下来,将会用最优卸载算法和文献[14]中算法进行对比分析。

图 4 为每个设备的任务调用图,每个节点代表一个任务,节点权值为完成此任务的计算量,边权值表示输入和输出数据的大小。这里将设备的任务数量设置为 $m=3$ 和 $n=5$,服务器的发射功率固定 p_0 为 1 W,每个设备的发射功率峰值 p_{peak} 为 0.1 W,边缘服务器的 CPU 频率 f_c 和每个设备的 CPU 频率峰值 f_{peak} 分别为 10^{10} 和 10^8(cycles/s),噪声功率 σ^2 为 10^{-7} W, k 是取决于芯片架构的有效开关电容参数,是固定常数,这里设置为 10^{-26}。此外设置带宽 W 为 2 MHz,对于每个设备上任务的计算量大小设置为 $L_{i,1}$ =[0,65.5,40,3,96.6,0](Mcycles), $L_{i,2}$ =[0,70.8,95.3,86.4,18.6,158.6,0](Mcycles)。

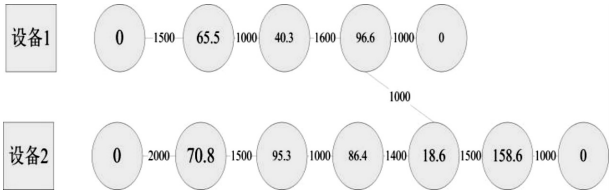


图 4　实验模型和参数

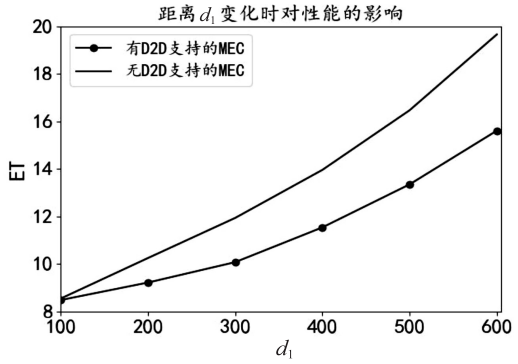
根据前面小节列出的信道增益公式,公式中的一些参数设置为: $g=4.11$ 表示天线增益, $F_c=915$ MHz 表示载波频率, d_r 表示距离,对于设备到服务器的距离和两个设备间的距离 d_1 、 d_2 和 d_3 的值分别为 300 m、300 m 和 10 m, pl=3 表示路径损耗指数。这里将每个设备的时间和能耗权重 β_1^T , β_1^E , β_2^T , β_2^E 分别设置为 0.085,0.915,0.1,0.9。

接下将与文献[14]算法下的系统性能做比较。在此次实验中由于文献[14]与本实验的最优卸载决策:设备 1 的第 m 个任务和设备 2 的第 k 个任务都是在边缘服务器执行,所以为了更好地对比最优算法和文献[14]中的算法,这里采用了相同的次优卸载决策,此决策中设备 1 的第 m 个任务和设备 2 的第 k 个任务都是在本地执行,这样能更直观地观察有无 D2D 技术支持系统性能的优劣。

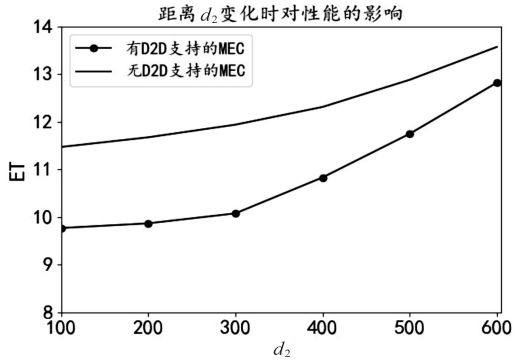
从图 5 中的(a)和(b)可以观察到,当固定 d_3 和 d_2 或固定 d_3 和 d_1 时,随着 d_1 或 d_2 距离的增大,该文提出的算法要优于文献[14]提出的算法,因为在支持 D2D 技术的 MEC 系统中,设备 1 的第 m 个任务和设备 2 的第 k 个任务都在本地执行,设备间之间可以直接通信,避免了在不支持 D2D 技术的 MEC 系统中需要先将任务卸载到边缘服务器,然后从边缘服务器下载到设备 2 上。所以可以得出支持 D2D 技术的 MEC 系统相比于文献[14]不支持 D2D 技术的 MEC 系统,节省了时间和能耗,系统性能更好

从图 5(c)可以观察到,当 d_3 增大时,文献[14]的算法不受 d_3 距离变化的影响,是因为设备 1 的第 m 个任务和设备 2 的第 k 个任务都在本地执行,设备之间不需直接通信。虽然该文提出的算法受 d_3 的影响,但是通过图(c)中曲线可以观察到,在两设备间的距离

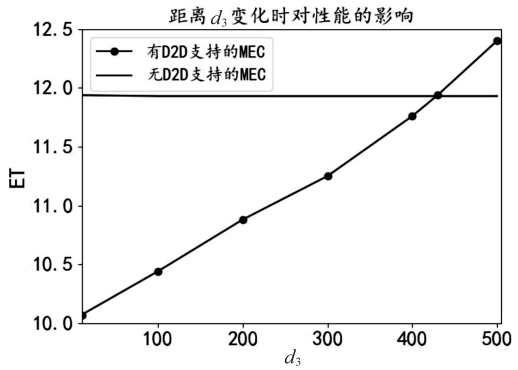
不大于设备到服务器之间的距离 400 m 时,该文所提的算法还是优于文献[14]所提的算法。而当两设备间的距离大于 400 m 时,任务卸载到边缘服务器上是较好的选择,因为边缘服务器的性能要比设备的性能好,将任务卸载到边缘服务器能节省能耗和时间。现实中也可以知道,当设备间的距离远大于设备到服务器之间的距离时,当然是卸载到服务器更好,但当设备间距离较近时,卸载到设备是更好的选择。



(a)



(b)



(c)

图 5 有 D2D 技术和无 D2D 技术系统性能对比

5 结束语

该文研究了 MEC 与 D2D 技术协作系统环境下不同设备之间具有任务依赖性的最优的资源分配和优化任务卸载决策问题。为了最小化设备的能耗和任务完成时间的加权和,首先固定卸载决策,推导出卸载发射功率和本地 CPU 频率的闭合表达式,运用凸优化方法求解出该问题的最优解,得到最佳资源分配策略。然后证明了最优卸载决策遵循一次爬升策略,在此基础上提出了一种降低复杂度的在线任务卸载算法,通过该算法获得了最优卸载决策。通过数值实验表明,提出的最优卸载策略明显优于其他算法。

参考文献:

[1] HU Y C,PATEL M,SABELLA D,et al. Mobile edge computing a key technology towards 5G[J]. ETSI White Paper,2015,11(11):1-16.

[2] FLORES H,HUI P,TARKOMA S,et al. Mobile code offloading:from concept to practice and beyond[J]. IEEE Communications Magazine,2015,53(3):80-88.

[3] LI Y,LIU Y, SALEEM U, et al. Latency minimization ford2d-enabled partial computation offloading in mobile edge computing[J]. IEEE Transactions on Vehicular Technology,2020,69(4):4472-4486.

[4] CAO H,WANG F,XU J,et al. Joint computation and communication cooperation for energy efficient mobile edge computing[J]. IEEE Internet of Things Journal,2019,6(3):4188-4200.

[5] ZHOU J,ZHANG X,ZHANG Y,et al. Energy-efficient collaborative task offloading in d2d-assisted mobile edge computing networks[C]//IEEE wireless communications and networking conference. Marrakesh,Morocco:IEEE,2019.

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

[7] HAN Y,ZHAO Z,MO J,et al. Efficient task offloading with dependency guarantees in ultra-dense edge networks[C]//IEEE global communications conference. Waikoloa, HI, USA:IEEE,2019:1-6.

[8] LI M,CHEN T,QI H,et al. D2D assisted computation offloading for mobile edge computing systems with energy harvesting[C]//20th international conference on parallel and distributed computing, applications and technologies. Gold Coast,OLD,Australia:[s. n.],2019.

[9] LI Y,LIU P,XU G,et al. Jointly optimizing helpers selection and resource allocation in D2D mobile edge computing[C]//IEEE wireless communications and networking conference. Seoul,Korea(South):IEEE,2020.

[10] HE Y,REN J,YU G,et al. Joint computation offloading and resource allocation in d2d enabled MEC networks[C]//IEEE international conference on communications. Shanghai, China:IEEE,2019.

[11] XING H,LIU L,XU J,et al. Joint task assignment andresource allocation for d2d-enabled mobile-edge computing

(下转第 95 页)