计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(8):327−337 [doi: 10.15888/j.cnki.csa.008620]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

多设备间任务依赖的最佳卸载决策和资源分配①

胡　恒, 金凤林, 谢　钧, 刘　莹

(陆军工程大学 指挥控制工程学院, 南京 210007)
通信作者: 金凤林, E-mail: fljin@sina.com

摘　要: 考虑了多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, 其中多个无线设备的最终输出作为另一个设备上某个子任务的输入. 为了最小化无线设备的能耗和任务完成时间的加权和, 研究了最优的资源分配 (卸载发射功率和本地 CPU 频率) 和任务卸载决策问题. 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, 运用凸优化方法求出该问题的解. 然后基于一次爬升策略提出了一种低复杂度线性搜索算法, 该算法可以在线性时间内获得最佳卸载决策. 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.

关键词: 移动边缘计算; 移动云计算; 计算卸载; 卸载决策; D2D 技术

引用格式: 胡恒,金凤林,谢钧,刘莹.多设备间任务依赖的最佳卸载决策和资源分配.计算机系统应用,2022,31(8):327–337. http://www.c-s-a.org.cn/1003-3254/8620.html

Optimal Offloading Decision and Resource Allocation for Task Dependencies among Multiple Devices

HU Heng, JIN Feng-Lin, XIE Jun, LIU Ying

(Command & Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China)

Abstract: The collaboration network of mobile edge computing (MEC) and device-to-device (D2D) technology takes into consideration multiple devices, where the final output of multiple wireless devices is used as the input of a subtask on another device. The optimal resource allocation (offloading transmit power and local CPU frequency) and task offloading decisions are studied to minimize the weighted sum of the energy consumption of wireless devices and the task completion time. First, given an offloading decision, the closed expression of offloading transmit power and local CPU frequency are derived, and the convex optimization method is used to find the solution to the problem. Then, on the basis of the one-climb policy, a low-complexity linear search algorithm is proposed, which can obtain the best offloading decision in linear time. Numerical results show that the performance of this strategy is significantly better than that of other representative benchmark tests.

Key words: mobile edge computing (MEC); mobile cloud computing; computation offloading; offloading decision; device-to-device (D2D) technology

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

① 收稿时间: 2021-10-30; 修改时间: 2021-12-02; 采用时间: 2021-12-20; csa 在线出版时间: 2022-06-17

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对其进行研究. 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, 最小化了系统的总能耗. 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. 同时, 在优化卸载决策方面文献 [14] 提出的算法复杂度仍然很高, 本文提出的线性搜索算法更进一步的优化了卸载决策, 并且本文将场景扩展到多设备. 因此, 本文在文献 [14] 的基础上进行了进一步的优化研究, 联合优化了设备任务完成时延和能耗.

本文在第 1 节中对系统进行了分析, 建立了通信、计算和任务依赖模型. 在第 2 节中运用凸优化方法得到最佳资源分配. 在第 3 节中提出了一种降低复杂度的线性搜索算法来优化卸载决策. 最后通过仿真实验进行性能评估.

1 系统模型

1.1 MEC 与 D2D 协作网络系统模型

本文考虑了多个设备的 MEC 与 D2D 协作网络. 如图 1 所示, MEC 与 D2D 协作网络系统模型. 其中 d_s 和 d_2 分别表示设备 S、设备 2 与 MEC 服务器之间的距离, 其中, $S=1,3,4,\cdots,s$, 表示设备 1, 设备 3, …, 设备 s, $d_{(S,2)}$表示设备$1,3,4,\cdots,s$和设备 2 之间的距离.

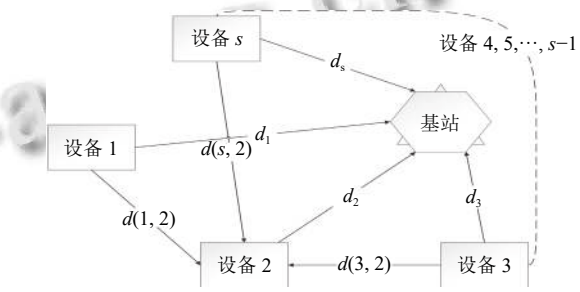


图 1 MEC 与 D2D 协作网络系统模型

设备 S 和设备 2 分别具有m_s和n个要执行的任务, 其中m_s表示设备S的第m个任务. 将任务之间的依赖关系建模为顺序图, 对每个设备引入两个虚拟任务, 0 为设备的输入任务, m_s+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S第m_s个任务的输出和设备 2 第 k–1 个任务的输出.

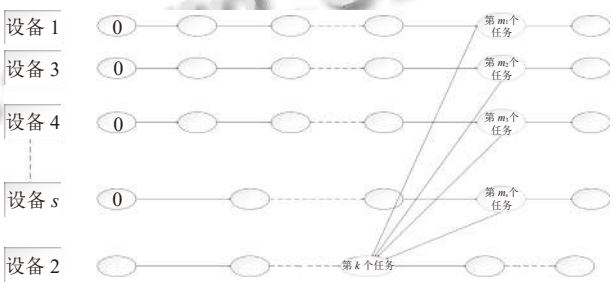


图 2 不同设备间调用图模型

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$j=S$或 2, $j=S$表示设备$S=1,3,4,\cdots,s$, $j=2$表示设备 2, i表示设备上的任务, $i=0,1,2,\cdots,m_s+1$或$i=0,1,2,\cdots,n+1$. $L_{i,j}$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S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S和设备 2 第i个任务的输入等于上一个任务的输出: $I_{i,j}=O_{i-1,j}$. 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S第m_s个任务的输出: $I_{k,2}=O_{k-1,2}+O_{m_s,s}$. 本文规定第 0 个和最后一个虚拟任务只能在本地执行. 同时, $a_{i,j}=\{0,1\}$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

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:

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

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程.

下载速率为:

$$R_{i,j}^{d}=W\log_2\left(1+\frac{p_0h_{i,j}}{\sigma^2}\right) \tag{2}$$

其中, p_0表示边缘服务器的固定发射功率.

设备S与设备 2 之间的传输速率为:

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

其中, $p'_{m_s+1,1}$和$h'_{m_s+1,1}$分别表示设备S第m_s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:

$$h_{i,j}=g\left(\frac{3\times10^8}{4\pi F_c d_r}\right)^{pl} \tag{4}$$

其中, g表示天线增益, F_c表示载波频率, d_r表示距离, $d_r=d_S$或d_2或$d_{(S,2)}$, 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:

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

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所消耗的能耗:

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

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:

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

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:

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

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

将任务i从设备传输到服务器的传输能耗:

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

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:

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

将任务从设备S传输到设备 2 的传输时间:

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

其中, $p'_{m_s+1,s}=\frac{\sigma^2}{h'_{m_s+1,s}}\left(2^{\frac{Om_s,s}{Wt_{m_s+1,s}^{s,2}}}-1\right)$.

将任务从设备S传输到设备 2 的传输能耗:

$$e_{i,j}^{s,2}=p'_{m_s+1,s}t_{m_s+1,s}^{s,2}=\frac{\sigma^2t_{m_s+1,s}^{s,2}}{h'_{m_s+1,s}}\left(2^{\frac{Om_s,s}{Wt_{m_s+1,s}^{s,2}}}-1\right) \tag{12}$$

1.4 依赖模型

由于设备S的第m_s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S第m_s个任务和设备 2 第k个任务的位置关系. 通过综合分析, 设备S的第m_s个任务和设备 2 的第k个任务的位置关系都有 4 种: 当设备S第m_s个任务和设备 2 第k个任务都在本地执行时, 第k个任务的输入依赖于第m_s个任务的输出, 由于两设备都支持 D2D 技术, 所以设备之间可以直接通信, 设备S不必将第m_s个任务的输出先传递到边缘服务器, 然后再又服务器传递到设备 2, 节省了时间和能耗, 此时只有设备间的传输时间和能耗; 当设备S第m_s个任务和设备 2 第k个任务都在服务器上执行时, 此时设备没有传输时间和能耗的损耗; 当第m_s个任务在本地执行时, 当第k个任务在服务器上执行时, 由于第k个任务的

输入依赖于第m_s个任务的输出, 此时设备S需要将第m_s个任务的输出传输到服务器, 有上传时延和能耗; 当第m_s个任务在边缘服务器执行, 第k个任务在本地执行时, 此时设备S需要将第m_s个任务的输出从服务器传输到设备 2, 此时需要下载时延.

1.5 问题公式化

由以上分析可知, 设备S的任务完成时间为:

$$T_s=\sum_{i=1}^{m_s}\left[(1-a_{i,s})t_{i,s}^{l}+a_{i,s}t_{i,s}^{c}\right]+\sum_{i=1}^{m_s+1}\left[a_{i,s}(1-a_{i-1,s})t_{i,s}^{u}+a_{i-1,s}(1-a_{i,s})t_{i,s}^{d}\right] \tag{13}$$

由于设备间任务的依赖关系, 在式 (14) 的最后一行关系式中描述了设备间任务的依赖关系. 当设备S第m_s个任务和设备 2 的第k个任务都在本地执行时, 则设备之间需要传输能耗$e_{m_s+1,s}^{s,2}$; 当设备S第m_s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, 则需要上传能耗$e_{m_s+1}^{u}$. 因此设备 s 的能耗为:

$$E_s=\sum_{i=1}^{m_s}\left[(1-a_{i,s})e_{i,s}^{l}+a_{i,s}(1-a_{i-1,s})e_{i,s}^{u}\right]+(1-a_{m_s,s})(1-a_{k,2})e_{m_s+1,s}^{s,2}+a_{k,2}(1-a_{m_s,s})e_{m_s+1}^{u} \tag{14}$$

由于设备 2 的第k个任务输入依赖于设备S的第m_s个任务和设备 2 第k–1 任务的输出, 当设备S第m_s个任务和设备 2 的第k个任务都在本地执行时, 设备间需要传输时间:$t_{m_s+1,1}^{s,2}$; 当设备S第m_s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$O_{m_s,s}/R_{k,2}^{d}$; 当设备S第m_s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$t_{m_s+1,s}^{u}$. 因此设备 2 的第 k 个任务等待设备 S 的第 m_s 个任务的输出时间为:

$$\begin{aligned}T_s^{\text{wait}}=&\sum_{i=1}^{m_s}\left[(1-a_{i,s})t_{i,s}^{l}+a_{i,s}t_{i,s}^{c}\right]+\sum_{i=1}^{m_s}\left[a_{i,s}(1-a_{i-1,s})t_{i,s}^{u}+a_{i-1,s}(1-a_{i,s})t_{i,s}^{d}\right]\\&+(1-a_{m_s,s})(1-a_{k,2})t_{m_s+1,s}^{s,2}+a_{k,2}(1-a_{m_s,s})t_{m_s+1,s}^{u}+a_{m_s,s}(1-a_{k,2})\frac{O_{m_s,s}}{R_{k,2}^{d}}\end{aligned} \tag{15}$$

等待设备 2 第k–1 个任务的输出时间:

$$T_2^{\text{wait}}=\sum_{i=1}^{k-1}\left[(1-a_{i,2})t_{i,2}^{l}+a_{i,2}t_{i,2}^{c}\right]+\sum_{i=1}^{k}\left[a_{i,2}(1-a_{i-1,2})t_{i,2}^{u}+a_{i-1,2}(1-a_{i,2})t_{i,2}^{d}\right] \tag{16}$$

令$t=\max\left\{T_1^{\text{wait}},T_2^{\text{wait}},\cdots,T_s^{\text{wait}}\right\}$.

设备 2 的任务完成时间为:

$$T_2=\max\left\{T_1^{\text{wait}},T_2^{\text{wait}},\cdots,T_s^{\text{wait}}\right\}+\sum_{i=k}^{n}\left[(1-a_{i,2})t_{i,2}^{l}+a_{i,2}t_{i,2}^{c}\right]+\sum_{i=k+1}^{n+1}\left[a_{i,2}(1-a_{i-1,2})t_{i,2}^{u}+a_{i-1,2}(1-a_{i,2})t_{i,2}^{d}\right] \tag{17}$$

设备 2 消耗的能耗为:

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

设备总性能指标为:

$$\begin{aligned}\eta\left(\{a_{i,j}\},\{p_{i,j}\},\{f_{i,j}\}\right)&=\eta_1+\eta_2+\cdots+\eta_s\\&=\beta_1^{T}T_1+\beta_1^{E}E_1+\beta_2^{T}T_2\\&\quad+\beta_2^{E}T_2+\cdots+\beta_s^{T}T_s+\beta_s^{E}E_s\end{aligned} \tag{19}$$

其中, $\beta_1^{T},\beta_1^{E},\beta_2^{T},\beta_2^{E},\cdots,\beta_s^{T},\beta_s^{E}$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:

$0<\beta_1^{T}<1,\ 0<\beta_1^{E}<1,\ 0<\beta_2^{T}<1,\ 0<\beta_2^{E}<1,\cdots,0<\beta_s^{T}<1,\ 0<\beta_s^{E}<1,\ \beta_s^{E}=1-\beta_s^{T}$

所以问题公式化为式 (20):

$$P(1)\quad\min_{(\{a_{i,j}\},\{p_{i,j}\},\{f_{i,j}\})}\eta\left(\{a_{i,j}\},\{p_{i,j}\},\{f_{i,j}\}\right)$$

$$\text{s.t.}\begin{cases}0\leqslant p_{i,j}\leqslant P_{\text{peak}},\\0\leqslant p'_{m_s+1,s}\leqslant P_{\text{peak}},\\0\leqslant f_{i,j}\leqslant f_{\text{peak}},\\a_{i,j}\in\{0,1\},\forall i,j\end{cases} \tag{20}$$

其中, P_{peak} 表示设备发射功率的峰值, f_{peak} 表示设备 CPU 计算能力的峰值. 由式 (5), 式 (8), 式 (11) 可知$f_{i,j},p_{i,j},p'_{m_s+1,s}$与$f_{i,j},p_{i,j},p'_{m_s+1,s}$, 所以式 (20) 可以转

化为:

$$P(2)\quad \min_{\left(\{a_{i,j}\},\{t_{i,j}^{l}\},\{t_{i,j}^{u}\},\{t_{m_s+1,s}^{s,2}\},t\right)} \eta\left(\{a_{i,j}\},\{t_{i,j}^{l}\},\{t_{i,j}^{u}\},\{t_{m_s+1,s}^{s,2}\},t\right)$$

$$\text{s.t.}\begin{cases} T_1^{\text{wait}} \leqslant t, T_2^{\text{wait}} \leqslant t, \cdots, T_s^{\text{wait}} \leqslant t \\ \dfrac{L_{i,j}}{f_{\text{peak}}} \leqslant t_{i,j}^{l}, \\ \dfrac{O_{i-1,j}}{W\log_2\left(1+\dfrac{p_{\text{peak}}h_{i,j}}{\sigma^2}\right)} \leqslant t_{i,j}^{u}, \\ \dfrac{O_{m_s,s}}{W\log_2\left(1+\dfrac{p_{\text{peak}}h'_{m_s+1,s}}{\sigma^2}\right)} \leqslant t_{m_s+1,s}^{s,2}, \\ a_{i,j}\in\{0,1\}, \forall i,j \end{cases} \tag{21}$$

由于含有二进制变量$a_{i,j}$, 所以问题是非凸的, 但当固定$a_{i,j}$时, 原来的非凸问题就会转化为关于$t_{i,j}^{l}, t_{i,j}^{u}, t_{m_s+1,s}^{s,2}$的凸问题.

2 固定卸载决策的最佳资源分配

假设固定$a_{i,j}$, 则非凸问题$P(2)$转化为关于$t_{i,j}^{l}, t_{i,j}^{u}, t_{m_s+1,s}^{s,2}$的凸问题$P(3)$:

$$P(3)\quad \min_{\left(\{t_{i,j}^{l}\},\{t_{i,j}^{u}\},\{t_{m_s+1,s}^{s,2}\},t\right)} \eta\left(\{t_{i,j}^{l}\},\{t_{i,j}^{u}\},\{t_{m_s+1,s}^{s,2}\},t\right)$$

$$\text{s.t.}\begin{cases} T_1^{\text{wait}} \leqslant t, T_2^{\text{wait}} \leqslant t, \cdots, T_s^{\text{wait}} \leqslant t \\ \dfrac{L_{i,j}}{f_{\text{peak}}} \leqslant t_{i,j}^{l}, \\ \dfrac{O_{i-1,j}}{W\log_2\left(1+\dfrac{p_{\text{peak}}h_{i,j}}{\sigma^2}\right)} \leqslant t_{i,j}^{u}, \\ \dfrac{O_{m_s,s}}{W\log_2\left(1+\dfrac{p_{\text{peak}}h'_{m_s+1,s}}{\sigma^2}\right)} \leqslant t_{m_s+1,s}^{s,2}, \end{cases} \tag{22}$$

问题$P(3)$的拉格朗日表示为:

$$\begin{aligned} &L\left(\{t_{i,j}^{u}\},\{t_{i,j}^{l}\},\{t_{m_s+1,s}^{s,2}\},t,\lambda_1,\lambda_2,\cdots,\lambda_s\right) \\ &=\eta_1+\eta_2+\cdots+\eta_s+\lambda_1\left(T_1^{\text{wait}}-t\right) \\ &\quad+\lambda_2\left(T_2^{\text{wait}}-t\right)+\cdots+\lambda_s\left(T_s^{\text{wait}}-t\right) \end{aligned} \tag{23}$$

其中, $\lambda_1,\lambda_2,\cdots,\lambda_s$都是不小于零的表示与相应约束相关联的对偶变量, $\lambda_1^*,\lambda_2^*,\cdots,\lambda_s^*$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:

$$\begin{cases} f_{i,s}^{*}=\begin{cases} f_{\text{peak}}, \text{如果}\beta_s^T+\lambda_s^*>2k\beta_s^E f_{\text{peak}}^3 \\ \sqrt[3]{\dfrac{\beta_s^T+\lambda_s^*}{2k\beta_s^E}}, \text{否则} \end{cases} & \text{当}i\in\{1,\cdots,m_s\}\text{时} \\ f_{i,2}^{*}=\begin{cases} \begin{cases} f_{\text{peak}}, \text{如果}\lambda_2^*>2k\beta_2^E f_{\text{peak}}^3 \\ \sqrt[3]{\dfrac{\lambda_2^*}{2k\beta_2^E}}, \text{否则} \end{cases} & \text{当}i\in\{1,\cdots,k-1\}\text{时} \\ \begin{cases} f_{\text{peak}}, \text{如果}\beta_2^T>2k\beta_2^E f_{\text{peak}}^3 \\ \sqrt[3]{\dfrac{\beta_2^T}{2k\beta_2^E}}, \text{否则} \end{cases} & \text{当}i\in\{k,\cdots,n\}\text{时} \end{cases} \end{cases} \tag{24}$$

$$
\begin{cases}
p_{i,s}^{*}=\begin{cases}P_{\text{peak}}, \text{如果} h_{i,s}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_1}{-W(-A_1 e^{-A_1})}-1\right] \\ \dfrac{\sigma^2}{h_{i,s}}\left[\dfrac{B_1}{W(B_1 e^{-1})}-1\right], \text{否则}\end{cases} & \text{当} i\in\{1,\cdots,m_s\}\text{时} \\
p_{i,s}^{*}=\begin{cases}P_{\text{peak}}, \text{如果} h_{i,s}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_2}{-W(-A_2 e^{-A_2})}-1\right] \\ \dfrac{\sigma^2}{h_{i,s}}\left[\dfrac{B_2}{W(B_2 e^{-1})}-1\right], \text{否则}\end{cases} & \\
\text{或者 } p_{i,s}^{*}=\begin{cases}P_{\text{peak}}, \text{如果} h'_{i,s}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_2}{-W(-A_2 e^{-A_2})}-1\right] \\ \dfrac{\sigma^2}{h'_{i,s}}\left[\dfrac{B'_2}{W(B'_2 e^{-1})}-1\right], \text{否则}\end{cases} & \text{当} i=m_s+1\text{时} \\
p_{i,2}^{*}=\begin{cases}P_{\text{peak}}, \text{如果} h_{i,2}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_3}{-W(-A_3 e^{-A_3})}-1\right] \\ \dfrac{\sigma^2}{h_{i,2}}\left[\dfrac{B_3}{W(B_3 e^{-1})}-1\right], \text{否则}\end{cases} & \text{当} i\in\{1,\cdots,k\}\text{时} \\
p_{i,2}^{*}=\begin{cases}P_{\text{peak}}, \text{如果} h_{i,2}<\dfrac{\sigma^2}{P_{\text{peak}}}\left[\dfrac{A_4}{-W(-A_4 e^{-A_4})}-1\right] \\ \dfrac{\sigma^2}{h_{i,2}}\left[\dfrac{B_4}{W(B_4 e^{-1})}-1\right], \text{否则}\end{cases} & \text{当} i\in\{k+1,\cdots,n\}\text{时}
\end{cases} \tag{25}
$$

其中, $A_1=1+\dfrac{\beta_s^T+\lambda_s{}^*}{\beta_s^E P_{\text{peak}}}$, $B_1=\dfrac{h_{i,s}(\beta_s^T+\lambda_s{}^*)}{\beta_s^E\sigma^2}-1$

$$A_2=1+\frac{\lambda_s{}^*}{\beta_s^E P_{\text{peak}}}, B_2=\frac{h_{i,s}\lambda_s{}^*}{\beta_s^E\sigma^2}-1, B'_2=\frac{h'_{i,s}\lambda_s{}^*}{\beta_s^E\sigma^2}-1$$

$$A_3=1+\frac{\lambda_2{}^*}{\beta_2^E P_{\text{peak}}}, B_3=\frac{h_{i,2}\lambda_2{}^*}{\beta_2^E\sigma^2}-1$$

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

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

算法 1. 固定卸载决策下求解最佳资源分配算法

1. 输入: 给定大于 0 的λ_s, λ_2, 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_s^{wait}, T_2^{wait}和目标值
7. t=max$\{T_s^{\text{wait}},T_2^{\text{wait}}\}$
8. 根据梯度下降算法, 分别更新λ_s的值
9. 如果$|T_s^{\text{wait}}-T_2^{\text{wait}}|<pre$:
10. 退出循环
11. End

3 优化卸载决策

在第 2 节中, 给定卸载决策, 可以得出最佳资源分配, 需要搜索$2^{m_1\times n\times m_3\times\cdots\times m_s}$次卸载决策, 然后从中选择最小目标的卸载决策. 但是随着任务任务的增多, 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线性搜索算法, 可以在线性时间内获得最佳卸载决策.

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: 由于文章篇幅限制, 并且有许多文献都有证明了一爬策略, 这里就不再证明.

3.2 基于一爬策略的线性搜索算法

在本小节中, 借鉴了文献 [16] 的思想, 同时基于一次爬升策略提出了一种低复杂度的线性搜索算法, 来优化卸载决策. 首先根据式 (27) 证明所有卸载点共享最小加权和点, 然后在找到最小加权和点的基础上寻找入口任务, 只需将两点之间的任务包括两点, 都卸载到服务器, 如图 3 所示.

图 3 一次爬升策略

公式如下, 设备有 m 个任务:

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

$$\eta_i=\begin{cases}\min\limits_{1\leqslant o\leqslant m}\{\eta(i,o)\}, \text{如果}\ \ i=1\\ \eta_i-\beta^E e_{i-2,i-1}^u+\beta^E\left(e_{i-1}^l+e_{i-1,i}^u\right)\\ -\beta^T\left(t_{i-2,i-1}^u+t_{i-1}^c\right)+\beta^T\left(t_{i-1,i}^u+t_{i-1}^l\right), \text{否则}\end{cases} \tag{27}$$

其中, η_i表示任务i到最小加权和点o的最佳性能, $\eta(i,o)$表示任务从入口任务i到出口任务o的加权和. e^u: 传输能耗, e^l: 本地能耗, t^u: 上传时间, t^c: 在服务器上的执行时间, t^d: 下载时间, t^l: 本地执行时间, β^E和β^T分别代表权重.

定理 2. 所有卸载点共享最小加权和点o.

证明: 假设当任务$i=j$时最小加权和点为o, $1\leqslant j\leqslant o\leqslant m$, 那么$j+1$的最小加权和点也是$o$.

当$i=j$, 卸载任务j时:

$$\eta(j,j)=\beta^E\left[\sum_{h=1}^{j-1}e_h^l+e_{j-1,j}^u+\sum_{k=j+1}^{m}e_k^l\right]+\beta^T\left[\sum_{h=1}^{j-1}t_h^l+t_{j-1,j}^u+t_j^c+t_{j,j+1}^d+\sum_{k=j+1}^{m}t_k^l\right] \tag{28}$$

卸载任务j, $j+1$时:

$$\eta(j,j+1)=\beta^E\left[\sum_{h=1}^{j-1}e_h^l+e_{j-1,j}^u+\sum_{k=j+2}^{m}e_k^l\right]+\beta^T\left[\sum_{h=1}^{j-1}t_h^l+t_{j-1,j}^u+t_j^c+t_{j+1}^c+t_{j+1,j+2}^d+\sum_{k=j+2}^{M}t_k^l\right] \tag{29}$$

卸载任务j, $j+1$, $j+2$, ⋯, o时:

$$\eta(j,o)=\beta^E\left[\sum_{h=1}^{j-1}e_h^l+e_{j-1,j}^u+\sum_{k=o+1}^{m}e_k^l\right]+\beta^T\left[\sum_{h=1}^{j-1}t_h^l+t_{j-1,j}^u+t_j^c+\cdots+t_o^c+t_{o,o+1}^d+\sum_{k=o+1}^{M}t_k^l\right] \tag{30}$$

卸载任务j, $j+1$, ⋯, m时:

$$\eta(j,m)=\beta^E\left[\sum_{h=1}^{j-1}e_h^l+e_{j-1,j}^u\right]+\beta^T\left[\sum_{h=1}^{j-1}t_i^l+t_{j-1,j}^u+t_j^c+\cdots+t_m^c+t_{m,m+1}^d\right] \tag{31}$$

因为当$i=j$时, 最小加权和点是o, 所以:

$$\begin{aligned}&\eta(j,o)\leqslant\eta(j,j)\Rightarrow\\ &\beta^T[t_{j+1}^c+\cdots+t_o^c+t_{o,o+1}^d]\leqslant\beta^E[e_{j+1}^l+\cdots+e_o^l]\\ &+\beta^T[t_{j,j+1}^d+t_{j+1}^l+\cdots+t_o^l]\\ &\eta(j,o)\leqslant\eta(j,j+1)\Rightarrow\\ &\beta^T[t_{j+2}^c+\cdots+t_o^c+t_{o,o+1}^d]\leqslant\beta^E[e_{j+2}^l+\cdots+e_o^l]\\ &+\beta^T[t_{j+1,j+2}^d+t_{j+2}^l+\cdots+t_o^l]\\ &\eta(j,o)\leqslant\eta(j,j+2)\Rightarrow\\ &\beta^T[t_{j+3}^c+\cdots+t_o^c+t_{o,o+1}^d]\leqslant\beta^E[e_{j+3}^l+\cdots+e_o^l]\\ &+\beta^T[t_{j+2,j+3}^d+t_{j+3}^l+\cdots+t_o^l]\\ &\cdots\\ &\eta(j,o)\leqslant\eta(j,o+1)\Rightarrow\\ &\beta^E[e_{o+1}^l]+\beta^T[t_{o,o+1}^d+t_{o+1}^l]\leqslant\beta^T[t_{o+1}^c+t_{o+1,o+2}^d]\\ &\eta(j,o)\leqslant\eta(j,o+2)\Rightarrow\\ &\beta^E[e_{o+1}^l+e_{o+2}^l]+\beta^T[t_{o,o+1}^d+t_{o+1}^l+t_{o+2}^l]\leqslant\\ &\beta^T[t_{o+1}^c+t_{o+2}^c+t_{o+2,o+3}^d]\\ &\cdots\\ &\eta(j,o)\leqslant\eta(j,m)\Rightarrow\\ &\beta^E[e_{o+1}^l+\cdots+e_m^l]+\beta^T[t_{o,o+1}^d+t_{o+1}^l+\cdots+t_m^l]\leqslant\\ &\beta^T[t_{o+1}^c+\cdots+t_m^c+t_{m,m+1}^d]\end{aligned} \tag{32}$$

当$i=j+1$, 卸载任务$j+1$时:

$$\eta(j+1,j+1)=\beta^E\left[\sum_{h=1}^{j}e_h^l+e_{j,j+1}^u+\sum_{k=j+2}^{m}e_k^l\right]+\beta^T\left[\sum_{h=1}^{j}t_h^l+t_{j,j+1}^u+t_{j+1}^c+t_{j+1,j+2}^d+\sum_{k=j+2}^{m}t_k^l\right] \tag{33}$$

卸载任务$j+1$, $j+2$, ⋯, o时:

$$\eta(j+1,o)=\beta^E\left[\sum_{h=1}^{j}e_h^l+e_{j,j+1}^u+\sum_{k=o+1}^{m}e_k^l\right]+\beta^T\left[\sum_{h=1}^{j}t_h^l+t_{j,j+1}^u+t_{j+1}^c+\cdots+t_o^c+t_{o,o+1}^d+\sum_{k=o+1}^{m}t_k^l\right] \tag{34}$$

卸载任务$j+1$, $j+2$, ⋯, m时:

$$\eta(j+1,m)=\beta^E\left[\sum_{h=1}^{j}e_h^l+e_{j,j+1}^u\right]+\beta^T\left[\sum_{h=1}^{j}t_h^l+t_{j,j+1}^u+t_{j+1}^c+\cdots+t_m^c+t_{m,m+1}^d\right] \tag{35}$$

当$i=j$, 最小加权和点为o, 仔细观察可以得出当$i=j+1$时, 最小加权和点也是o, 即:

$$\begin{aligned}&\eta(j+1,o)\leqslant\eta(j+1,j+1)\Rightarrow\\&\beta^T[t_{j+2}^c+\cdots+t_o^c+t_{o,o+1}^d]\leqslant\beta^E[e_{j+2}^l+\cdots+e_o^l]\\&+\beta^T[t_{j+1,j+2}^d+t_{j+2}^l+\cdots+t_o^l]\\&\eta(j+1,o)\leqslant\eta(j+1,j+2)\Rightarrow\\&\beta^T[t_{j+3}^c+\cdots+t_o^c+t_{o,o+1}^d]\leqslant\beta^E[e_{j+3}^l+\cdots+e_o^l]\\&+\beta^T[t_{j+2,j+3}^d+t_{j+3}^l+\cdots+t_o^l]\\&\cdots\\&\eta(j+1,o)\leqslant\eta(j+1,o+1)\Rightarrow\\&\beta^E[e_{o+1}^l]+\beta^T[t_{o,o+1}^d+t_{o+1}^l]\leqslant\beta^T[t_{o+1}^c+t_{o+1,o+2}^d]\\&\cdots\\&\eta(j+1,o)\leqslant\eta(j+1,M)\Rightarrow\\&\beta^E[e_{o+1}^l+\cdots+e_m^l]+\beta^T[t_{o,o+1}^d+t_{o+1}^l+\cdots+t_m^l]\leqslant\\&\beta^T[t_{o+1}^c+\cdots+t_m^c+t_{m,m+1}^d]\end{aligned} \tag{36}$$

从而得出所有卸载点共享最小加权和点o, 证毕.

从定理 2 可知, 只需遍历i=1 就可以找到最小加权和点o, 同时得到此时的最佳性能为$\eta_1=\eta(1,o)$, 然后根据η_i与η_{i-1}的关系, 求出$\eta_1,\cdots,\eta_o$, 通过式 (37) 找到入口任务h.

$$\eta_h=\max\{\eta_1,\cdots,\eta_o\} \tag{37}$$

基于定理 1 和定理 2, 可以通过提出的线性搜索算法快速地找到最小加权和点o和入口任务h. 算法 2 给出了寻找最优的入口任务h和最佳性能点o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$2^{m_1\times n\times m_3\times\cdots\times m_s}$个卸载决策. 对于每个设备, 根据算法 2 首先遍历m个任务寻找到最佳性能点o, 然后再根据每个卸载点的关系, 求出每个卸载点到最佳性能点o的值, 选出最小的作为卸载点, 算法的复杂度为O(m). 其他设备类似. 所以只需O($m_1\times n\times m_3\times\cdots\times m_s$)的复杂度就可以找到最优的卸载策略. 随着任务数的增多, 基于单爬策略的线性搜索算法的性能远远优于暴力搜索算法, 复杂度更低, 算法 2 描述如下.

算法 2. 低复杂度线性搜索算法

1. 输入: 根据式 (33) 计算$\eta(1,1)$令$\eta_{\min}=\eta(1,1)$, o=1
2. 输出: h, o

3. For $i\leftarrow$1 to m:
4. 根据算法 1 计算出最优f^*和p^*
5. 然后分别计算设备能耗和时间
6. 根据能耗、时间和式 (26) 计算$\eta(1,i)$
7. If $\eta(1,i)<\eta_{\min}$:
8. $\eta_{\min}=\eta(1,i)$
$o=i$
9. End
10. 找到i=1 时的最小加权和点o
11. $\eta_1=\eta_{\min}$
12. For $j\leftarrow$2 to o:
13. 根据η_j与η_{j-1}的关系公式计算出η_j的值
14. End
15. 根据公式$\eta_h=\max\{\eta_1,\cdots,\eta_o\}$选出入口任务$h$

4 实验评估

在这一节, 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 (时延和能耗的加权和最小), 用 ETC 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, 实验数据值的大小参考了文献 [14] 进行了设置.

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 1、设备 2 和设备 3 的实际任务数量分别设置为 3、4 和 5, 每个设备引入了两个虚拟任务. 服务器的发射功率p_0固定为 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), $L_{i,3}$=[0, 65.5, 86.4, 158.6, 96.6, 0](Mcycles).

根据第 3 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, $r=1,2,3,(1,2),(3,2)$对于设备 1、设备 2 和设备 3 到服务器的距离d_1、d_2、d_3的值别为 100 m、100 m 和 100 m.设备 1 到设备 2 和设备 3 到设备 2 之间的距离$d_{(1,2)}$、$d_{(3,2)}$的值分别为 10 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, β_3^T, β_3^E分别设置为 0.05, 0.95, 0.1, 0.9, 0.02, 0.98.

在图 5 中, 首先进行了本文所提的算法和一爬策略枚举搜索算法的比较. 通过图 5 可以观察到, 随着任务数的增多获得最佳卸载策略所花费的时间, 本文所提算法的性能要优于一爬枚举搜索算法. 在相同的任务数下, 本文所提的算法获得最优卸载策略的时间明显小于一爬策略枚举搜索算法明.

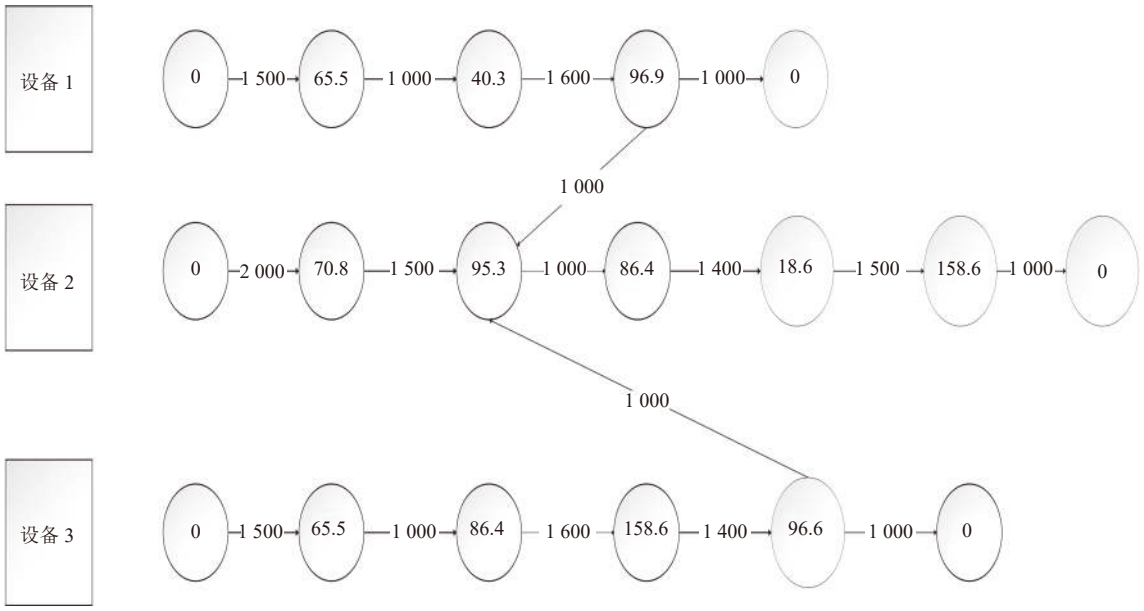


图 4 实验模型和参数

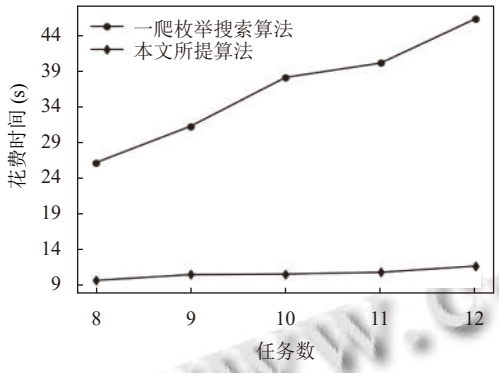


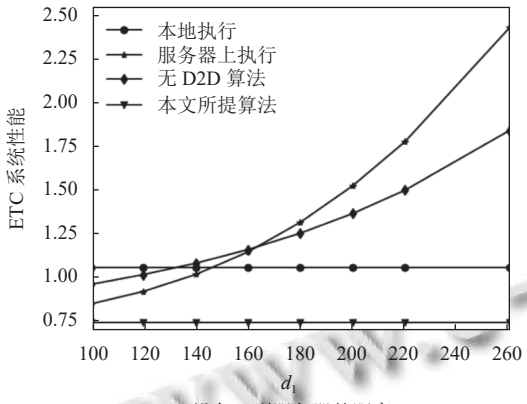
图 5 两种算法对比曲线图

接下来, 本文给出了当d_1、d_2和$d_{(1,2)}$变化时, 不同算法下设备的性能, 其中设置β_1^T=0.0 5、β_2^T=0.1 和β_3^T=0.02. 为了进行性能比较, 考虑了 3 个次优算法作为基准. 第 1 种算法被称为所有任务卸载, 其中将 3 个设备中的所有任务卸载到边缘边缘服务器上执行. 对于第 2 种算法被称为所有任务本地执行, 3 个设备中的所有任务都在本地执行. 此外, 本文将没有 D2D 技术支持称为第 3 种算法, 也算是文献 [14] 算法的一种实现, 其中每个设备最小化自己的性能 ETC.

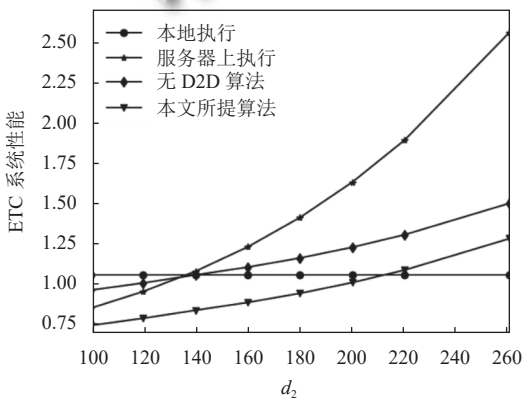
从图 6(a) 可以观察到, 当固定其他距离时, 随着距离d_1越来越大, 本文所提算法要优于另外 3 种卸载算法, 性能更好. 而任务全部在本地执行算法和本文所提算法性能都没有变化, 是因为对于任务全部在本地执行算法, 此时设备 1 的第m个任务与设备 2 的第k个任务都在本地执行, 而两个设备都支持 D2D 技术, 可以直接进行通信, 所以距离d_1增大, 对于任务全部在本地执行的性能是没有影响, 而本文所提的算法获得的最优卸载决策设备 1 的任务也都是在本地执行, 所以也不受距离d_1变化的影响. 另外两种算法, 所有任务卸载和无 D2D 技术支持算法都随着距离的增大, 性能越来越差, 因为当距离d_1增大时, 卸载任务到服务器需要花费更高的传输时延和能耗. 同时也可以观察到, 当服务器和设备距离小于 160 m 时, 全部卸载算法要优于无 D2D 技术支持算法, 因为服务器的性能更好, 设备距离服务器更近时, 将任务全部卸载到服务器, 能获得更短的时延和更低的能耗. 但不管距离如何增大, 本文所提算法都要优于其他算法.

图 6(b) 显示了距离d_2变化对设备性能的影响. 从图中可以观察到本文所提算法、全部在服务器执行算

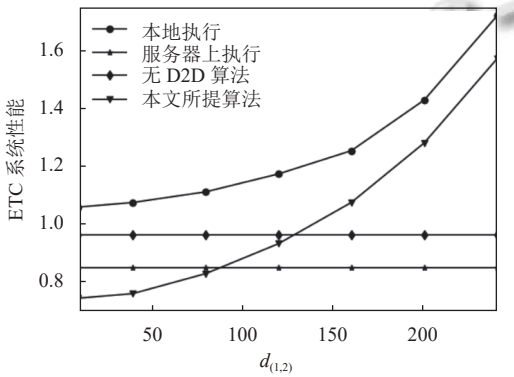
法和无 D2D 技术支持算法, 设备 ETC 性能都随着距离的增加而增加. 虽然随着距离的增加, 性能变差, 但是本文所提算法还是优于服务器执行算法和无 D2D 技术支持算法. 同时从图中可以观察到, 当距离小于 130 m 左右时, 全部卸载算法还要优于无 D2D 技术支持算法. 而全部在本地执行算法, 总的 ETC 不变



(a) 设备 1 到服务器的距离 d_1



(b) 设备 2 到服务器的距离 d_2



(c) 设备 1 与设备 2 之间的距离 $d_{(1,2)}$

图 6 与其他算法的对比

图 6(c) 显示了距离$d_{(1,2)}$变化对设备性能的影响. 对于任务全部在本地执行算法和本文所提算法有明显的影响, 因为此时设备 1 的第m个任务与设备 2 的第k个任务都在本地执行, 而两个设备都支持 D2D 技术, 两设备可以直接进行通信. 所以随着距离$d_{(1,2)}$的增大, 任务全部在本地执行算法和本文所提算法性能越来越差, 但本文所提算法还是优于全部在本地执行算法. 另一个方面也可以得出, 当两设备间距离更近时, 设备间直接通信越节省时间和能耗. 而对于全部卸载算法和无 D2D 技术支持算法, 由于本次设备 1 的第m个任务与设备 2 的第k个任务都在服务器上执行, 所以距离$d_{(1,2)}$的变化, 对于这两种算法都不受影响. 类似的当d_3和$d_{(3,2)}$变化时, 能够得到与d_1和$d_{(1,2)}$变化时类似图 6(a) 和图 6(c) 的结果, 这里不再过多叙述.

通过以上对比分析, 本文所提的算法明显优于其他基准算法, 同时与 D2D 技术协作可以显着提高系统的性能.

5 结论与展望

本文研究了 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, 来最小化设备的能耗和任务完成时间的加权和, 为了解决该问题, 提出了一种低复杂度的在线任务卸载算法, 获得了最优计算卸载策略. 最后通过数值实验表明, 本文所提的算法明显优于其他基准算法. 下一步将对多设备多服务器场景进行研究.

参考文献

1 Abbas N, Zhang Y, Taherkordi A, *et al*. Mobile edge computing: A survey. IEEE Internet of Things Journal, 2018, 5(1): 450–465. [doi: 10.1109/JIOT.2017.2750180]

2 Flores H, Hui P, Tarkoma S, *et al*. Mobile code offloading: From concept to practice and beyond. IEEE Communications Magazine, 2015, 53(3): 80–88. [doi: 10.1109/MCOM.2015.7060486]

3 Saleem U, Liu Y, Jangsher S, *et al*. Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. IEEE Transactions on Vehicular Technology, 2020, 69(4): 4472–4486. [doi: 10.1109/TVT.2020.2978027]

4 Cao XW, Wang F, Xu J, *et al*. Joint computation and communication cooperation for energy-efficient mobile edge computing. IEEE Internet of Things Journal, 2019, 6(3): 4188–4200. [doi: 10.1109/JIOT.2018.2875246]

5 Zhou JZ, Zhang X, Wang WB, *et al*. Energy-efficient collaborative task offloading in D2D-assisted mobile edge

computing networks. Proceedings of 2019 IEEE Wireless Communications and Networking Conference. Marrakesh, Morocco: IEEE, 2019. 1–6.

6 Mahmoodi SE, Uma RN, Subbalakshmi KP. Optimal joint scheduling and cloud offloading for mobile applications. IEEE Transactions on Cloud Computing, 2019, 7(2): 301–313. [doi: 10.1109/TCC.2016.2560808]

7 Han YP, Zhao ZW, Mo JW, *et al*. Efficient task offloading with dependency guarantees in ultra-dense edge networks. Proceedings of 2019 IEEE Global Communications Conference. Waikoloa: IEEE, 2019. 1–6.

8 Li ML, Chen T, Zeng JX, *et al*. D2D-assisted computation offloading for mobile edge computing systems with energy harvesting. Proceedings of the 20th International Conference on Parallel and Distributed Computing, Applications and Technologies, Gold Coast: IEEE, 2019. 90–95.

9 Li Y, Xu GC, Ge JQ, *et al*. Jointly optimizing helpers selection and resource allocation in D2D mobile edge computing. Proceedings of 2020 IEEE Wireless Communications and Networking Conference. Seoul: IEEE, 2020. 1–6.

10 He YH, Ren JK, Yu GD, *et al*. Joint computation offloading and resource allocation in D2D enabled MEC networks. Proceedings of 2019 IEEE international Conference on Communications. Shanghai: IEEE, 2019. 1–6.

11 Xing H, Liu L, Xu J, *et al*. Joint task assignment and resource allocation for D2D-enabled mobile-edge computing. IEEE Transactions on Communications, 2019, 67(6): 4193–4207. [doi: 10.1109/TCOMM.2019.2903088]

12 He YH, Ren JK, Yu GD, *et al*. D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Transactions on Wireless Communications, 2019, 18(3): 1750–1763. [doi: 10.1109/TWC.2019.2896999]

13 Saleem U, Liu Y, Jangsher S, *et al*. Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Transactions on Wireless Communications, 2021, 20(1): 360–374. [doi: 10.1109/TWC.2020.3024538]

14 Yan J, Bi SZ, Zhang YJA. Optimal offloading and resource allocation in mobile-edge computing with inter-user task dependency. Proceedings of 2018 IEEE Global Communications Conference. Abu Dhabi: IEEE, 2018. 1–8.

15 Boyd S, Vandenberghe L. Convex Optimization. Cambridge: Cambridge University Press, 2004.

16 Zhang Y, Liu H, Jiao L, *et al*. To offload or not to offload: An efficient code partition algorithm for mobile cloud computing. Proceedings of the 2012 IEEE 1st International Conference on Cloud Networking. Paris: IEEE, 2012. 80–86.

(校对责编: 孙君艳)