Latency Optimization for D2D-Enabled Parallel Mobile Edge Computing in Cellular Networks

Liang Ran , Yan Cai*, Jun Zhang , and Hongbo Zhu ,

Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China.

Email: Liang Ran - 1018010102@njupt.edu.cn; Yan Cai*- caiy@njupt.edu.cn; Jun Zhang - zhangjun@njupt.edu.cn; Hongbo Zhu - zhuhb@njupt.edu.cn;

*Corresponding author

Abstract

Edge offloading, including offloading to edge base stations (BS) via cellular links and to idle mobile users (MUs) via device-to-device (D2D) links, has played a vital role in achieving ultra-low latency characteristics in 5G wireless networks. This paper studies an offloading method of parallel communication and computation to minimize the delay in multi-user systems. Three different scenarios are explored, i.e., full offloading, partial offloading, and D2D-enabled partial offloading. In the full offloading scenario, we find a serving order for the MUs. Then, we jointly optimize the serving order and task segment in the partial offloading scenario. For the D2D-enabled partial offloading scenario, we decompose the problem into two subproblems and then find the optimal solution based on the results of the two subproblems. Finally, the simulation results demonstrate that the offloading method of parallel communication and computing can significantly reduce the system delay, and the D2D-enabled partial offloading can further reduce the latency.

Keywords: Latency, edge offloading, device-to-device (D2D) communication, task segment.

1 Introduction

With the proliferation of Internet of Things (IoT) devices and limited by computation resource of mobile users (MUs) as well as communication resources, traditional cloud computing cannot meet the requirements of ultra-low latency in

5G networks [1]. Recently, mobile edge computing (MEC) has been regarded as a promising means to cope with the challenge [2, 3], as it can effectively relieve network congestion and decrease service latency. Specifically, the MUs offload intensive computation tasks to the proximate edge servers with more computing resource for execution. Low transmission delay can be achieved because of the short distances between the MUs and edge servers.

There have been lots of works on MEC offloading problems from the perspectives of improving energy efficiency [4–9], shortening system delay [10–16], and balancing the two objectives [17, 18]. A theoretical framework of energy-optimal with joint communication and computation resource allocation has been provided in [4]. In [5], aiming at controlling local resources and selecting computing modes, an energy-saving computing framework is proposed. Furthermore, a execution delay parallel computing method at tiered computing nodes in the single-user system is proposed in [10]. In [11], under the transmit power constraint, a calculation task scheduling method based on Markov decision process has been derived in a single-user MEC offloading system to minimize the average system delay. In [12], in order to reduce the system latency, the authors developed a strategy with joint computation offloading, content caching, and resource allocation. [14] minimizes system latency by jointly allocating communication resources and computing resources in a multi-user MEC system. Besides, to deal with solve the limitation of computing resources and wireless access bandwidth, computation partitioning and resource allocation was integrated with the MEC was proposed in [15].

In addition to offloading to the edge servers, we can also take full advantage of the computation resource of the idle MUs and offload parts of the computing tasks to the idle MUs via D2D communication to relieve the pressure on the edge servers. Actually, the introduction of D2D offloading into the MEC system has also been investigated in many works, such as [19–23]. In order to design energy-efficient co-computing policies, the non-causal CPU-state information has been exploited in [19]. By introducing D2D communication into the MEC system, the computation capacity of the cellular networks was effectively improved [20]. Moreover, [23] has developed a new mobile task offloading framework in which mobile devices share the communication and computation resources to improve energy efficiency.

Most of the aforementioned works are done from the perspective of computation capacity maximization or energy efficiency maximization. However, minimizing the end-to-end latency is also a critical objective for 5G wireless networks. This paper investigates a D2D-enabled MEC offloading system and aims to reduce the system delay by integrating D2D communication technique into the MEC system. This paper considers three different scenarios based on the location where the data is processed, i.e., the full offloading scenario where the raw data is calculated only at the edge server, the partial offloading where raw data is computed on both the local equipment and the edge server, and the D2D-enabled partial offloading with one part for local computing and the rest two parts are offloaded to the

neighbour MU and the edge server, respectively.

Here are the main contribution of this paper:

1) We consider the MEC offloading scenarios where parallel communication and computation is performed. The offloading includes offloading to the edge servers via cellular communication and offloading to the neighbouring idle MUs via D2D communication. Three different scenarios are considered, i.e., the full offloading, the partial scenario, and the D2D-enabled partial offloading scenarios.

2) For the full offloading scenario, we give a heuristic solution to finding the serving order of the MUs. Then, we jointly optimize the serving order and task segment in the partial offloading scenario. For the D2D-enabled partial offloading scenario, we decompose the problem into two subproblems and then find the optimal solution based on the results of the two subproblems.

3) The simulation results show that when the computing resources of the edge server are insufficient, or the local computing resources are insufficient, or the wireless channel capacity is small, all three cases show better delay performance. In particular, D2D-enabled partial offloading can greatly reduce system latency.

The remainder of this paper is organized as follows. The system model and three different execution models are introduced in Section 2. Sections 3, 4, and 5 investigate the full offloading, the partial offloading, and D2D-enabled partial offloading scenarios, respectively. Simulation results are presented in Section 6 and the paper is concluded in Section 7.

2 Methods/Experimental

As illustrated in Fig. 1, we consider a D2D-enabled MEC system which consists of one BS equipped with an edge server and K mobile users (MUs) denoted by $\mathcal{U} = \{U_1, U_2, \ldots, U_K\}$. The MU k can be characterized by a tuple $\{L_k, V_k^{\text{loc}}\}$ where L_k (in bits) represents the size of its task and V_k^{loc} (in bits/s) is its local computing capacity. Due to the limited local computing resource, each MU can offload its task to the edge server for processing through a cellular link. Besides, there are some idle MUs which are willing to process the tasks offloaded to themselves and we refer to them as helpers. For simplicity, we assume that the number of the helpers is the same as that of the MUs waiting for service and these helpers are denoted by a set $\mathcal{H} = \{H_1, H_2, \ldots, H_K\}$. We also suppose that each MU can establish a D2D communicate channel with one and only one helper at the same time. As shown in Fig. 2, the task L_k can be divided into three parts for local computing, edge server processing, and helper processing, respectively. It is assumed that the delay for task segmenting can be ignored. The computing capacity of H_k and the edge server are denoted as V_k^{help} and V^{edge}, respectively.

2.1 Communication Models

There are two kinds of wireless communication: Cellular communication and D2D communication. We assume that D2D communication and cellular communication use different frequency bands and the time division multiple access scheme is adopted in D2D communication. Thus, there is no inter-user interference in the considered system.

2.1.1 D2D Communication

The achievable rate of the D2D link between U_k and H_k is given as

$$r_k^{\text{d2d}} = B_0 \log_2 \left(1 + \frac{p_k^{\text{d2d}} \left|h_k^{\text{d2d}}\right|^2}{N_0}\right), \tag{1}$$

where B_0 denotes the bandwidth, p_k^{d2d} indicates the transmit power of U_k for D2D communication, N_0 represents the background noise power, and h_k^{d2d} denotes the Rayleigh channel gain.

2.1.2 Cellular Communication

The achievable rate of the cellular link of U_k is given as

$$r_k^{\text{edge}} = B_1 \log_2 \left(1 + \frac{p_k^{\text{edge}} \left|h_k^{\text{edge}}\right|^2}{N_0}\right), \tag{2}$$

where B_1 denotes the bandwidth for cellular communication, p_k^{edge} is the transmit power, and h_k^{edge} denotes the Rayleigh channel gain between U_k and the BS.

2.2 Executing Models

The task of each MU can be processed locally or offloaded to the helper and the edge server. Thus, there are two kinds of offloading: D2D offloading and MEC offloading.

2.2.1 Local Computation

When U_k processes its task locally, the related computation delay is given as

$$D_k^{\text{loc}} = \frac{L_k}{V_k^{\text{loc}}}. \tag{3}$$

We define the system delay as the maximum of the local computation delay of the MUs, i.e.,

$$D^{\text{loc}} = \max \left(D_1^{\text{loc}}, D_2^{\text{loc}}, ..., D_K^{\text{loc}}\right). \tag{4}$$

2.2.2 D2D Offloading

We assume each helper starts processing data only after the whole data is received, thus the D2D offloading delay for U_k can be expressed as

$$D_k^{\text{d2d}} = D_k^{\text{d2d,t}} + D_k^{\text{d2d,c}}, \tag{5}$$

where $D_k^{\text{d2d,t}} = \frac{L_k}{t_k r_k^{\text{d2d}}}$ is the transmission delay and $D_k^{\text{d2d,c}} = \frac{L_k}{V_k^{\text{help}}}$ is the computing delay. $t_k \in [0, 1]$ is the normalized slot length of U_k.

The total delay of D2D offloading is given as

$$D^{\text{d2d}} = \max\left(D_1^{\text{d2d}}, D_2^{\text{d2d}}, ..., D_K^{\text{d2d}}\right). \tag{6}$$

2.2.3 MEC Offloading

We assume that the transmission and computation tasks are implemented in two sequences in parallel, as shown in Fig. 3. Specifically, an MU cannot transmit (execute) its task until the task of the last MU has been transmitted (executed) completely. Besides, each MU's task can be processed only after the data reception is finished. Then, the total latency consumed to executed K MUs' tasks by edge server is computed as

$$D_K^{\text{edge}} = \sum_{i=1}^{K} D_i^{\text{comp}} + \sum_{i=1}^{K-1} D_i^{\text{wait}}, \tag{7}$$

where $D_k^{\text{comp}} = \frac{L_k}{V^{\text{edge}}}$ is the computing delay and D_k^{wait} represents the waiting time of the edge server before computing the data of U_{k+1} and is given as

$$D_k^{\text{wait}} = \max\left(\sum_{i=2}^{k+1} D_i^{\text{tran}} - D_k^{\text{edge}}, 0\right), \tag{8}$$

where $D_k^{\text{tran}} = \frac{L_k}{r_k^{edge}}$ is the transmission delay. Finally, based on (7) and (8), the total delay of the MEC offloading can be expressed as

$$D^{\text{edge}} = D_K^{\text{edge}} + D_1^{\text{tran}}. \tag{9}$$

In the following, depending on the place where the tasks are performed, we will consider three different scenarios: Full offloading, partial offloading, and D2D-enabled partial offloading.

3 Optimal Solution to the Full Offloading Scenario

In the full offloading scenario, all MUs' tasks are offloaded to the edge server for processing.

3.1 Problem Formulation

There are only two delays involved here: the latency for cellular link transmission and the computational latency for the edge server. The total delay of this scenario can be obtained by 2.2.3 and is expressed as:

$$D^{\text{edge}} = \sum_{i=1}^{K} D_i^{\text{comp}} + \sum_{i=1}^{K-1} D_i^{\text{wait}} + D_{\varphi_1}^{\text{tran}}, \tag{10}$$

where φ_1 denotes the index and $D_{\varphi_1}^{\text{tran}}$ denotes the first transmission delay.

Our goal is to minimize the total system latency for all MUs, i.e.,

$$\textbf{P1:} \quad \min_{\boldsymbol{\varphi}} D^{\text{edge}} \tag{11a}$$

$$\text{s.t.} \ \ \varphi_k \in \{1, 2, \ldots, K\}, \tag{11b}$$

where $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \ldots, \varphi_K]^T \in \mathbb{R}^{K \times 1}$ denotes the index vector without replacement which determines the transmission (processing) order of the MUs.

3.2 Optimal Solution

We propose a heuristic algorithm for **P1**, as shown in Algorithm 1. In Algorithm 1, the MU with the shortest transmission delay is chosen as the first one for offloading. Since the processing time of each MU's task at the edge server is certain, minimizing the total system delay is equivalent to minimizing the waiting delay. The delay of the edge server computing current MU's task is greater than the transmission delay of the next MU, ensuring that the server has no idle period.

Algorithm 1 Heuristic algorithm for full offloading.

Sort the MUs in descending order according to L_k.
Set $k = 1$ and $n = 0$.
while $k < K$ **do**
  if $D_{\varphi_{k+1}}^{\text{tran}} > D_{\varphi_k}^{\text{comp}}$ **then**
    $n = k, temp = \varphi_{k+1}$.
    Do
    $\varphi_{n+1} = \varphi_{n+2}$.
    $n = n + 1$.
    Until $n = K - 2$
    $\varphi_K = temp$.
  else
    Update the delay of edge computation by (7).
    Update the waiting delay of edge server by (8).
    Update $k = k + 1$.
  end if
end while
Return D_K^{edge} + $D_{\varphi_1}^{\text{tran}}$, $\boldsymbol{\varphi}$

4 Optimal Solution to the Partial Offloading Scenario

In the partial offloading scenario, each MU's task L_k is partitioned into two parts with a split parameter λ_k: One with $\lambda_k L_k$ bits remains for local computing and the other with $(1-\lambda_k)L_k$ bits is offloaded to the edge server for processing.

4.1 Problem Formulation

According to 2.2.1 and 2.2.3, the local delay and the offloading delay in the partial offloading scenario are given as

$$D^{\text{loc}} = \max\left(\frac{\lambda_1 L_1}{V_1^{\text{loc}}}, \frac{\lambda_2 L_2}{V_2^{\text{loc}}}, ..., \frac{\lambda_K L_K}{V_K^{\text{loc}}}\right), \tag{12}$$

and

$$D^{\text{edge}} = \sum_{i=1}^{K} \frac{(1-\lambda_k) L_k}{V^{\text{edge}}} + \sum_{i=1}^{K-1} D_i^{\text{wait}} + D_{\varphi_1}^{\text{tran}}, \tag{13}$$

respectively. Then the system delay minimization problem is formulated as

$$\textbf{P2:} \quad \min_{\boldsymbol{\lambda},\boldsymbol{\varphi}} \max\left\{D^{\text{loc}}, D^{\text{edge}}\right\} \tag{14a}$$

$$\text{s.t.} \quad 0 \le \lambda_k \le 1, \tag{14b}$$

$$(11b), \tag{14c}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_K]^T$.

4.2 Optimal Solution

Before presenting the proposed algorithm, we first introduce some lemmas.

Lemma 1 *Given the computation and communication resources and the amount of data, the total system delay can not be minimal as long as some computing or communication resources are idle.*

Proof: See Appendix A.

It can be inferred from Lemma 1 that all MUs should finish their local computation at the same time. Thus, the splitting ratio of each MU's task is proportional to its local computing resource. We introduce a variable h and the division ratio λ_k is represented as

$$\lambda_k = \min\left(h\frac{V_k^{\text{loc}} \sum_{k=1}^{K} L_k}{L_k \sum_{k=1}^{K} V_k^{\text{loc}}}, 1\right), h > 0. \tag{15}$$

Then, according to (12) the local calculation delay can be denoted as

$$D^{\text{loc}} = h\frac{\sum_{k=1}^{K} L_k}{\sum_{k=1}^{K} V_k^{\text{loc}}}. \tag{16}$$

Lemma 2 *If and only if $D^{loc} = D^{edge}$, the optimal solution of **P2** is obtained.*

Proof: See Appendix B.

According to Lemma 2, we need to find an optimal h^* to make $D^{\text{loc}} = D^{\text{edge}}$. Thus, Algorithm 2 is proposed.

Algorithm 2 Proposed algorithm for partial offloading.

1: **Initialize**
2: Initialize the variable $h = 0$ and the tolerance factor $\epsilon > 0$.
3: Set the iteration step $s = 0.01$ and the iteration index $n = 0$.
4: Calculate $D_{(0)}^{\text{loc}}$ and $D_{(0)}^{\text{edge}}$ according to (16) and Algorithm 1, respectively.
5: **while** $\left|D_{(n)}^{\text{loc}} - D_{(n)}^{\text{edge}}\right| > \epsilon$ **do**
6: Recalculate $D_{(n)}^{\text{loc}}$ and $D_{(n)}^{\text{edge}}$ according to (16) and Algorithm 1, respectively.
7: Update $n = n + 1$, $h = h + s$
8: **end while**
9: **Return** $D_{(n)}^{\text{loc}}$.

5 Optimal Solution to the D2D-Enabled Partial Offloading Scenario

In this section, we will study the latency-optimization problem of introducing D2D communication into the MEC system. Each MU's task is divided into three parts, one part remains local, and the other two parts are offloaded to its corresponding helper and the edge server, respectively.

5.1 Problem Formulation

We first introduce two parameters α_k and β_k for U_k. Then, the amount of data that is offloaded to edge server is denoted as $\alpha_k L_k$. We also denote by $(1-\alpha_k)\,\beta_k L_k$ the amount of data that is processed locally and $(1-\alpha_k)\,(1-\beta_k)L_k$ the amount of data that is offloaded to the corresponding helper, respectively. Then, according to 2.2.1, 2.2.2 and 2.2.3, the delays of each part in this scenario can be expressed as

$$D^{\text{loc}} = \max\left(\frac{(1-\alpha_1)\,\beta_1 L_1}{V_1^{\text{loc}}}, \frac{(1-\alpha_2)\,\beta_2 L_2}{V_2^{\text{loc}}}, ..., \frac{(1-\alpha_K)\,\beta_K L_K}{V_K^{\text{loc}}}\right), \tag{17}$$

$$D_k^{\text{d2d}} = \frac{(1-\alpha_k)\,(1-\beta_k)\,L_k}{t_k r_k^{\text{d2d}}} + \frac{(1-\alpha_k)\,(1-\beta_k)\,L_k}{V_k^{\text{help}}}, \tag{18}$$

$$D^{\text{d2d}} = \max\left(D_1^{\text{d2d}}, D_2^{\text{d2d}}, ..., D_K^{\text{d2d}}\right), \tag{19}$$

and

$$D^{\text{edge}} = \sum_{i=1}^{K} \frac{\alpha_k L_k}{V^{\text{edge}}} + \sum_{i=1}^{K-1} D_i^{\text{wait}} + D_{\varphi_1}^{\text{tran}}, \tag{20}$$

respectively. Finally, the delay minimization problem under D2D-enabled partial offloading is formulated as

$$\textbf{P3:} \min_{\boldsymbol{\alpha},\boldsymbol{\beta},\mathbf{t},\boldsymbol{\varphi}} \max\left(D^{\text{loc}}, D^{\text{d2d}}, D^{\text{edge}}\right) \tag{21a}$$

$$\text{s.t.} \sum_{k=1}^{K} t_k \leq 1, \tag{21b}$$

$$0 \leq t_k \leq 1, \tag{21c}$$

$$0 \leq \alpha_k \leq 1, \tag{21d}$$

$$0 \leq \beta_k \leq 1, \tag{21e}$$

$$\varphi_k \in \{1, 2, \ldots, K\}. \tag{21f}$$

It is not easy to find the optimal solution to **P3** due to its non-convexity.

5.2 Problem Decomposition

In order to solve **P3**, we first introduce a lemma based on Lemma 2.

Lemma 3 *If and only if $D^{loc} = D^{d2d} = D^{edge}$, the optimal solution to **P3** is obtained.*

Proof: See Appendix C.

Notice that the two variable sets $\{\boldsymbol{\alpha}, \boldsymbol{\varphi}\}$ and $\{\boldsymbol{\beta}, \boldsymbol{t}\}$ are independent of each other. Thus, we can split **P3** into two sub-problems:

$$\textbf{P4:} \min_{\boldsymbol{\beta},\boldsymbol{t}} \max\left(D^{\text{loc}}, D^{\text{d2d}}\right) \tag{22a}$$

$$\text{s.t. (21b), (21c), and (21e)}, \tag{22b}$$

and

$$\textbf{P5:} \min_{\boldsymbol{\alpha},\boldsymbol{\varphi}} \max\left(D^{\text{l,d*}}, D^{\text{edge}}\right) \tag{23a}$$

$$\text{s.t. (21d) and (21f)}, \tag{23b}$$

where $D^{\text{l,d*}}$ represents the optimal solution of **P4**

By analyzing the intrinsic relationship between the two subproblems, we further have the following lemma.

Lemma 4 *The optimal solutions to **P4** and **P5** are equivalent to the optimal solution to **P3**.*

Proof: See Appendix D.

5.3 Optimal Solution

According to the result in Section 4.2, the local computing delay of the system is expressed as

$$D^{\text{loc}} = h\frac{\sum_{k=1}^{K}(1-\alpha_k)L_k}{\sum_{k=1}^{K}V_k^{\text{loc}}}, h > 0. \tag{24}$$

The D2D offload delay consists of two parts: The D2D-link transmission delay and the helper processing delay. We consider that the D2D transmission resource t_k can be allocated according to the computing resource of the corresponding helper. Let t_k satisfy the equation below,

$$\frac{t_1 r_1^{\text{d2d}}}{V_1^{\text{help}}} = \frac{t_2 r_2^{\text{d2d}}}{V_2^{\text{help}}} = ... = \frac{t_K r_K^{\text{d2d}}}{V_K^{\text{help}}} = g, g > 0, \tag{25}$$

which leads to

$$t_k = \frac{gV_k^{\text{help}}}{r_k^{\text{d2d}}}, \tag{26}$$

and

$$g = \frac{1}{\sum_{k=1}^{K}\frac{V_k^{\text{help}}}{r_k^{\text{d2d}}}}. \tag{27}$$

From (18) and (25), the helper processing delay can be expressed as

$$D_k^{\text{d2d}} = \frac{(1-\alpha_k)(1-\beta_k)L_k}{gV_k^{\text{help}}} + \frac{(1-\alpha_k)(1-\beta_k)L_k}{V_k^{\text{help}}} = \frac{(1-\alpha_k)(1-\beta_k)L_k}{\frac{g}{g+1}V_k^{\text{help}}}, \tag{28}$$

where $\frac{g}{g+1}V_k^{\text{help}}$ is the combined rate of D2D transmission and helper processing.

Similar to (24), the D2D offloading delay of this model is expressed as

$$D^{\text{d2d}} = f\frac{\sum_{k=1}^{K}(1-\alpha_k)L_k}{\frac{g}{g+1}\sum_{k=1}^{K}V_k^{\text{help}}}, f > 0. \tag{29}$$

Then, **P4** can be rewritten as

$$\textbf{P6:} \min_{\boldsymbol{\beta},\boldsymbol{t}} \max\left(h\frac{\sum_{k=1}^{K}(1-\alpha_k)L_k}{\sum_{k=1}^{K}V_k^{\text{loc}}}, f\frac{\sum_{k=1}^{K}(1-\alpha_k)L_k}{\frac{g}{g+1}\sum_{k=1}^{K}V_k^{\text{help}}}\right) \tag{30a}$$

$$\text{s.t. (21b), (21c) and (21e),} \tag{30b}$$

whose optimal solution is given by the following lemma.

Lemma 5 *The optimal solution of **P6** is given by*

$$\beta_k = \frac{V_k^{loc}}{V_k^{loc} + \frac{g}{g+1} V_k^{help}},\ t_k = \frac{V_k^{help}}{r_k^{d2d} \sum_{k=1}^{K} \frac{V_k^{help}}{r_k^{d2d}}}. \tag{31}$$

Proof: See Appendix E.

Base on Lemma 5, we can derive the closed-form solutions to **P4** and **P6**, as

$$D^{\mathrm{l,d}^*} = r^* \frac{\sum_{k=1}^{K} (1-\alpha_k) L_k}{\sum_{k=1}^{K} \left(V_k^{\mathrm{loc}} + \frac{g}{g+1} V_k^{\mathrm{help}} \right)}, \tag{32}$$

where $r^* = h^* + f^*$.

After getting $D^{\mathrm{l,d}^*}$, **P5** is equivalent to **P2**. According to Algorithm 2, the optimal solution D^{sys^*} of **P5** can be obtained. And then, it can be seen from Lemma 4 that D^{sys^*} is also the optimal solution of **P3**.

6 Results and Discussion

In this section, we provide numerical results to verify the superiority of system delay of three different scenarios based on parallel communication and computational offload strategies. The maximum coverage radius of the BS is set to 300 m. The location of the MUs in the system is random. Each MU can establish a wireless communication channel with the immediate helper. The radius of the longest D2D link is 50 m. The channel gains of all links follow i.i.d. Rayleigh distribution of unit variance. The bandwidth of the D2D link and the bandwidth of the cellular link are $B_0 = 5$ MHz and $B_1 = 10$ MHz, respectively. Each MU has the same transmit power, i.e., $p_k^{\mathrm{edge}} = p_k^{\mathrm{d2d}} = 24$ dBm. The data size L_k of U_k follows a uniform distribution within $L_k \in [10, 100]$ Mbits and the computing capacity V_k^{loc} (V_k^{help}) of U_k (H_k) follows a uniform distribution within $V_k^{\mathrm{loc}}, V_k^{\mathrm{help}} \in [0.5, 2]$ Mbps. Besides, the computing capacity of the edge server V^{edge} is set as 40 Mbps.

In the simulation, we introduced the MEC model of resource allocation [14] for comparison. In this model, the cellular communication resource and the computing resource of the edge server are allocated to each MU. And after the task transmission of all MUs is completed, these data are processed on the edge server at the same time. Here are two scenarios.

1) Existing full offloading: In this scenario, all MUs' tasks are offloaded to the edge server for execution.

2) Existing partial offloading: Each MU's task is partitioned into two parts: One with remains for local computing and the other is offloaded to the edge server processing.

Fig. 4 depicts the minimum delay in the five scenarios varies with the number of MUs in the system. By comparing the curves of existing full offloading and full offloading, we can observe that full offloading performs better than

existing full offloading. Moreover, as the number of MUs in the system increases, the performance gap between the two scenarios becomes more obvious. The system delay of the existing full offloading model is the sum of the cellular link transmission delay and the edge server calculation delay. However, the system delay of full offloading is almost equal to the computational delay. We can get the same conclusion by comparing the curves of existing partial offloading and partial offloading. And then, the performance of D2D-enabled partial offloading is best among the five scenarios. When there are many MUs in the system, the performance advantages of D2D-enabled partial offloading are more obvious, which indicates that D2D-enabled partial offloading model is suitable for user-intensive scenarios.

In Fig 5, the minimum system delay is the ordinate and the abscissa is the average computing resources of MUs. In this simulation, there were 20 MUs in the system, whose average local computing resources varied between 0.75 Mbps and 2.5 Mbps. From this figure, the advantage of partial offloading is more prominent when the local computing resources are scarce compared to existing partial offloading. The reason can be explained as follow. When there are fewer local computing resources, more data will be downloaded to the edge server. Existing partial offloading is subject to limited communication resources, resulting in large delay. However, the system delay of the partial offloading is less affected by the transmission delay. Therefore, our proposed partial offloading model is suitable for the case of sensor networks with weak computing power. Then, the average local computation resources of all MUs from 0.75 Mbps to 2.5 Mbps, D2D-enabled partial offloading model shows better performance than other four models.

Fig. 6 illustrates how the minimum system delay varies with the computing resource of the edge server. It is not difficult to find that the system delays of five different scenarios decrease with the increase of calculating resource of the edge server. Next, the performance gap between existing full offloading and full offloading becomes more evident with the increased of the calculating resource of the edge server. We have the similar observation by comparison of the existing partial offloading and partial offloading. Since the helpers' computing resource is utilized in the D2D-enabled partial offloading, the latency of such systems can be kept to a small level when the calculating resources of the edge server are scarce.

Fig. 7 shows the minimum system delay versus the radius of the BS. The BS radius is taken from 100m to 450m and we assume that there are 20 MUs in the system. It is not difficult to find that the system delays of existing full offloading and existing partial offloading are positively correlated with the BS radius. However, the system delay of three scenarios of parallel communication and computation offloading are approximately invariant. The reason is that, the increase in the radius of the BS affects the transmission capacity of the cellular link, thereby increasing the delay of transmitting data over the cellular link. We proposing the three scenarios are also applicable when the wireless channel is poor.

Fig. 8 shows the impact of the average size of MU data on the minimum system delay in five different scenarios.

The the average size of MUs' data is taken from 55 Mbits to 90 Mbits and we assume that there are 20 MUs in the system. From this figure, compared with the existing full offloading and existing partial offloading models, the corresponding full offloading and partial offloading have great performance advantages. And D2D-enabled partial offloading further shortens the latency of the system in processing data. So the three scenarios we propose can also perform well in networks with high-density computing tasks.

7 Conclusion

This paper proposes a strategy of parallel communication and computation for MEC offloading to improve the quality of service. Three scenarios, namely full offloading, partial offloading, and D2D-enabled partial offloading, are studied and compared. Finally, the numerical results validate the performance of the proposed algorithms.

Appendices

A. Proof of Lemma 1

The total data in the system is represented as $L^{\text{sys}} = \sum_{k=1}^{K} L_k$. The local computing resource of all the MUs and the computing resource of all the helpers are represented as $V^{\text{loc}} = \sum_{k=1}^{K} V_k^{\text{loc}}$ and $V^{\text{help}} = \sum_{k=1}^{K} V_k^{\text{help}}$, respectively. The sum of the achievable transmission rates of all the D2D links established in the system is expressed as $r^{\text{d2d}} = \sum_{k=1}^{K} r_k^{\text{d2d}}$. Then the rate of the helpers process data is

$$R^{\text{help}} = \frac{1}{\frac{1}{r^{\text{d2d}}} + \frac{1}{V^{\text{help}}}} = \frac{r^{\text{d2d}} V^{\text{help}}}{r^{\text{d2d}} + V^{\text{help}}}. \tag{33}$$

The system delay can be expressed as

$$D^{\text{sys}} = \frac{L^{\text{sys}}}{V^{\text{loc}} + R^{\text{help}} + V^{\text{edge}}}. \tag{34}$$

During the tasks is calculated, if some resources are idle, the resources participating in the calculation will be smaller than the resources existing in the system. This can be expressed as follows

$$V^{\text{loc'}} + R^{\text{help'}} + V^{\text{edge'}} < V^{\text{loc}} + R^{\text{help}} + V^{\text{edge}}. \tag{35}$$

Then we have

$$D^{\text{sys'}} = \frac{L^{\text{sys}}}{V^{\text{loc'}} + R^{\text{help'}} + V^{\text{edge'}}} > D^{\text{sys}}. \tag{36}$$

Thus, Lemma 1 is proved.

B. Proof of Lemma 2

Adequacy: Proof using the counter-evidence method.

Assuming $D^{\text{loc}} \neq D^{\text{edge}}$, the following discussion will be conducted.

1) When $D^{\text{loc}} > D^{\text{edge}}$, we have $D^{\text{loc}} - D^{\text{edge}} > 0$. In this case, the total system delay is determined by D^{loc} (i.e., $D^{\text{sys}} = D^{\text{loc}}$). The helpers will be idle for the $\left(D^{\text{loc}} - D^{\text{edge}}\right)$. According to Lemma 1, this situation is not an optimal solution.

2) When $D^{\text{loc}} < D^{\text{edge}}$, we have $D^{\text{edge}} - D^{\text{loc}} > 0$. In this case, the total system delay is determined by D^{edge} (i.e., $D^{\text{sys}} = D^{\text{edge}}$). The local computing resources of the MUs will be idle for the $\left(D^{\text{edge}} - D^{\text{loc}}\right)$. According to Lemma 1, this situation is not the optimal solution.

So, when $D^{\text{loc}} = D^{\text{edge}}$, the optimal solution of **P2** will be got.

Necessity:

According to the inverse lemma of Lemma 1, the **P2** obtains the optimal solution, and no resources in the system are in an idle state during the raw data is calculated. In other words, the local and helper are doing the calculations at the same time. That is $D^{\text{loc}} = D^{\text{edge}}$.

In summary, lemma 2 is proved.

C. Proof of Lemma 3

Adequacy: Proof using the counter-evidence method.

Assuming $D^{\text{loc}} \neq D^{\text{d2d}} \neq D^{\text{edge}}$, the following discussion will be conducted.

1) When $D^{\text{loc}} > D^{\text{d2d}} > D^{\text{edge}}$, we have $D^{\text{loc}} - D^{\text{d2d}} > 0$, $D^{\text{loc}} - D^{\text{edge}} > 0$. In this case, the total system delay is determined by D^{loc} (i.e., $D^{\text{sys}} = D^{\text{loc}}$). The helpers and the edge cloud will be idle for the $\left(D^{\text{loc}} - D^{\text{d2d}}\right)$ and $\left(D^{\text{loc}} - D^{\text{edge}}\right)$, respectively. According to Lemma 1, this situation is not an optimal solution.

2) Same as 1), when $D^{\text{loc}} > D^{\text{edge}} > D^{\text{d2d}}$, this situation is not the optimal solution.

3) Same as 1), when $D^{\text{edge}} > D^{\text{loc}} > D^{\text{d2d}}$, this situation is not the optimal solution.

4) Same as 1), when $D^{\text{d2d}} > D^{\text{loc}} > D^{\text{edge}}$, this situation is not the optimal solution.

5) Same as 1), when $D^{\text{edge}} > D^{\text{d2d}} > D^{\text{loc}}$, this situation is not the optimal solution.

6) Same as 1), when $D^{\text{d2d}} > D^{\text{edge}} > D^{\text{loc}}$, this situation is not the optimal solution.

So, when $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$, the optimal solution of P3 will be got.

Necessity:

According to the inverse lemma of Lemma 1, the **P3** obtains the optimal solution, and no resources in the system are in an idle state before the raw data is calculated. In other words, the local, helper and edge server are doing the

calculations at the same time. That is $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$.

In summary, Lemma 3 is proved.

D. Proof of Lemma 4

According to Lemma 3, the optimal solution to problem **P3** is obtained if and only if $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$. And $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$ is equivalent to $\left(D^{\text{loc}} = D^{\text{d2d}}\right) = D^{\text{edge}}$. In other words, we can first make $D^{\text{loc}} = D^{\text{d2d}}$ under constraints. According to Lemma 2, it is equivalent to solving $\min\limits_{\boldsymbol{\beta},\boldsymbol{t}} \max\left(D^{\text{loc}}, D^{\text{d2d}}\right)$ to obtain $D^{\text{l,d*}}$. Then let $D^{\text{l,d*}} = D^{\text{edge}}$ under the constraint condition, i.e., solve $\min\limits_{\boldsymbol{\alpha},\boldsymbol{\varphi}} \max\left(D^{\text{l,d*}}, D^{\text{edge}}\right)$, and get $D^{\text{sys*}}$. $D^{\text{sys*}}$ guarantees $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$, so $D^{\text{sys*}}$ is the optimal solution of **P3**.

E. Proof of Lemma 5

From (26) and (27), t_k can be obtained directly, denoted as

$$t_k = \frac{V_k^{\text{help}}}{r_k^{\text{d2d}} \sum_{k=1}^{K} \frac{V_k^{\text{help}}}{r_k^{\text{d2d}}}}. \tag{37}$$

It can be seen from Lemma 2 that the optimal solution of **P6** is obtained when $D^{\text{loc}} = D^{\text{d2d}}$, i.e.,

$$h^* \frac{\sum_{k=1}^{K} (1-\alpha_k) L_k}{\sum_{k=1}^{K} V_k^{\text{loc}}} = f^* \frac{\sum_{k=1}^{K} (1-\alpha_k) L_k}{\frac{g}{g+1} \sum_{k=1}^{K} V_k^{\text{help}}}. \tag{38}$$

The ratio of the raw data for local computing is

$$(1-\alpha_k)\beta_k^* = \min\left(h^* \frac{V_k^{\text{loc}} \sum_{k=1}^{K} (1-\alpha_k) L_k}{(1-\alpha_k) L_k \sum_{k=1}^{K} V_k^{\text{loc}}}, (1-\alpha_k)\right). \tag{39}$$

The ratio of the raw data for D2D offloading is

$$(1-\alpha_k)(1-\beta_k^*) = \min\left(f^* \frac{\frac{g}{g+1} V_k^{\text{help}} \sum_{k=1}^{K} (1-\alpha_k) L_k}{(1-\alpha_k) L_k \sum_{k=1}^{K} \frac{g}{g+1} V_k^{\text{help}}}, (1-\alpha_k)\right). \tag{40}$$

There are three cases of raw data segmentation for local computing and D2D computing:

1) $\beta_k = 1$, in this case, all data is placed locally.

2) $\beta_k = 0$, in this case, all data is offloaded to the helpers.

3) $\beta_k \neq 1$ and $\beta_k \neq 0$, in this case, the two split ratios are

$$(1-\alpha_k)\beta_k^* = h^* \frac{V_k^{\text{loc}} \sum_{k=1}^{K} (1-\alpha_k) L_k}{(1-\alpha_k) L_k \sum_{k=1}^{K} V_k^{\text{loc}}}, \tag{41}$$

and

$$(1-\alpha_k)(1-\beta_k^*) = f^* \frac{\frac{g}{g+1} V_k^{\text{help}} \sum_{k=1}^{K} (1-\alpha_k) L_k}{(1-\alpha_k) L_k \sum_{k=1}^{K} \frac{g}{g+1} V_k^{\text{help}}}, \tag{42}$$

respectively. We then have

$$(1-\alpha_k) = \frac{(h^*+f^*)\left(V_k^{\text{loc}} + \frac{g}{g+1}V_k^{\text{help}}\right)\sum_{k=1}^{K}(1-\alpha_k)L_k}{(1-\alpha_k)L_k\sum_{k=1}^{K}\left(V_k^{\text{loc}} + \frac{g}{g+1}V_k^{\text{help}}\right)}. \tag{43}$$

β_k^* can be obtained via dividing (41) by (43), i.e.,

$$\beta_k^* = \frac{V_k^{\text{loc}}}{V_k^{\text{loc}} + \frac{g}{g+1}V_k^{\text{help}}}. \tag{44}$$

It can be seen from Lemma 1 that the optimal solution of **P6** is certainly not obtained when β_k = 1 or β_k = 0. So the **P6** solution is the optimal solution $D^{\text{l,d}*}$ at $\beta_k^* = \frac{V_k^{\text{loc}}}{V_k^{\text{loc}} + \frac{g}{g+1}V_k^{\text{help}}}$.

Abbreviations

BS: Base station; MU: Mobile user; D2D: Device-to-device; IoT: Internet of Things; P1: Problem 1; P2: Problem 2; P3: Problem 3; P4: Problem 4; P5: Problem 5; P6: Problem 6.

Availability of data and materials

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the National Natural Science Foundation of China under Grant U1805262, 61871446.

Author's contribution

Conceived the study:H.B. Zhu, J. Zhang and Y. Cai. Designed and modelled: Y. Cai and L. Ran. Performance analysis:L. Ran, Y. Cai and J. Zhang. Results interpretation and manuscript writing: L. Ran, Y. Cai and J. Zhang.

Acknowledgements

The authors thank the person who provided valuable suggestions for improving the paper.

References

1. D. Evans, "The internet of things: How the next evolution of the internet is changing everything," CISCO, San Jose, CA, USA, White Paper, Apr. 2011.

2. J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

3. T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

4. W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

5. C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757–1771, May. 2016.

6. X. Xiang, C. Lin, and X. Chen, "Energy-efficient link selection and transmission scheduling in mobile cloud computing," *IEEE Wireless Communications Letters*, vol. 3, no. 2, pp. 153–156, Apr. 2014.

7. C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

8. S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

9. W. Xia, J. Zhang, T. Q. S. Quek, S. Jin, and H. Zhu, "Power minimization-based joint task scheduling and resource allocation in downlink c-ran," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7268–7280, Nov. 2018.

10. K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, "Task offloading and scheduling in fog ran: A parallel communication and computation perspective," *IEEE Wireless Communications Letters*, vol. 9, no. 2, pp. 215–218, Feb. 2020.

11. J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2016, pp. 1451–1455.

12. M. Molina, O. Muñoz, A. Pascual-Iserte, and J. Vidal, "Joint scheduling of communication and computation resources in multiuser wireless application offloading," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, Sep. 2014, pp. 1093–1098.

13. Y. Kao, B. Krishnamachari, M. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, Nov. 2017.

14. J. Ren, G. Yu, Y. Cai, and Y. He, "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506–5519, Aug. 2018.

15. L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, Jun. 2017, pp. 246–253.

16. W. Xia, T. Q. S. Quek, J. Zhang, S. Jin, and H. Zhu, "Programmable hierarchical c-ran: From task scheduling to resource allocation," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 2003–2016, Mar. 2019.

17. Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, Dec. 2016, pp. 1–6.

18. X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

19. C. You and K. Huang, "Exploiting non-causal cpu-state information for energy-efficient mobile cooperative computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4104–4117, June. 2018.

20. Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.

21. Y. Li, L. Sun, and W. Wang, "Exploring device-to-device communication for mobile cloud computing," in *Proc. IEEE International Conference on Communications (ICC)*, Jun. 2014, pp. 2239–2244.

22. W. Hu and G. Cao, "Quality-aware traffic offloading in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3182–3195, Nov. 2017.

23. L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.

Figures