Cai *et al. J Wireless Com Network* (2021) 2021:133
https://doi.org/10.1186/s13638-021-02008-z

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Latency optimization for D2D-enabled parallel mobile edge computing in cellular networks

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

*Correspondence: caiy@njupt.edu.cn
Jiangsu Key Laboratory of Wireless Communications, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

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 sub-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 [10–15], and balancing the two objectives [16, 17]. Ji and Guo [4] proposed an energy-effective resource allocation strategy based on wireless transmission energy in a two-user MEC system. In [5], aiming at controlling local resources and selecting computing modes, an energy-saving computing framework was proposed. Furthermore, a execution delay parallel computing method at tiered computing nodes in the single-user system was 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. Ren et al. [13] minimized system latency by jointly allocating communication resources and computing resources in a multi-user MEC system. Besides, to deal with the limitations of computing resources and wireless access bandwidth, computation partitioning and resource allocation were integrated with the MEC proposed in [14].

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 [18–22]. 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, [22] 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 were 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. In [18], the authors investigated a D2D-enabled MEC offloading system and aimed to reduce the system delay by integrating D2D communication technique into the MEC system with interference. However, a device cannot partially offload its computation task to the edge server and the corresponding proximal device, simultaneously. In the offloading part, [18] adopted the strategy of serial processing of computation tasks, and the delay of offloading process is the sum of transmission delay and calculation delay. In this paper, we also consider a D2D-enabled MEC offloading system to optimize the system delay. However, this paper considers three different scenarios based on the location where the data are processed, i.e., the full offloading scenario where the raw data are calculated only at the edge server, the partial offloading where raw data are 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 neighbor MU and the edge server, respectively.

Here are the main contributions 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 neighboring 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 sub-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 Sect. 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 Sect. 6, and the paper is concluded in Sect. 7.

2 System model

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 them and we refer to them as helpers. We denote the set of the idle MUs as $\bar{H} = \{H_1, H_2, \ldots, H_N\}$. It is assumed that each MU can establish a D2D link with at most a helper. When $N \geq K$, each MU can be assigned a helper and we denote the set of K helpers as $\mathcal{H} = \{H_1, H_2, ..., H_K\}$. When $N < K$, for

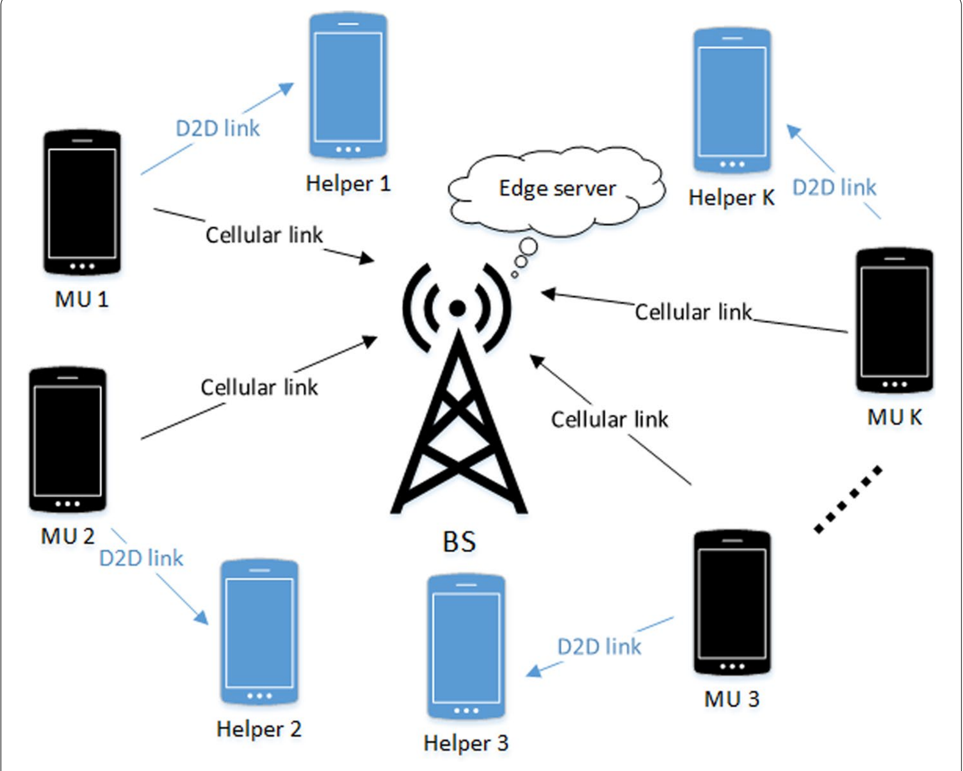

Fig. 1 An MEC-D2D system

the convenience of analysis , we still assume there are K helpers and the last $K-N$ helpers, i.e., $\{H_{N+1}, H_{N+2}, ...H_K\}$, are the virtual MUs with zero-capacity D2D links and zero-computing power. 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 capacities of $H_n (1 \leq n \leq K)$ and the edge server are denoted as V_n^{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

We assume that U_k and H_n establish a D2D link. $p_{k,n}^{\text{d2d}}$ indicates the transmit power of U_k and $h_{k,n}^{\text{d2d}}$ denotes the Rayleigh channel gain. If H_n does not actually exist, the transmit power and channel gain are both zero, i.e., $p_{k,n}^{\text{d2d}} = 0$ and $h_{k,n}^{\text{d2d}} = 0$. According to Shannon's formula, the achievable rate of the D2D link between U_k and H_n is given as

$$r_{k,n}^{\text{d2d}} = B_0 \log_2 \left(1 + \frac{p_{k,n}^{\text{d2d}} \left| h_{k,n}^{\text{d2d}} \right|^2}{N_0} \right), \tag{1}$$

where B_0 denotes the bandwidth for D2D communication, and N_0 represents the background noise power.

2.1.2 Cellular communication

It is assumed that all MUs in the system can establish cellular links with the edge cloud. MUs offload computation tasks in a certain order. In other words, only one MU communicates with the edge cloud at a time. Let B_1 denotes the bandwidth for cellular communication, p_k^{edge} is the transmit power of U_k, and h_k^{edge} denotes the Rayleigh channel gain between U_k and the BS. According to Shannon's formula, the achievable rate of the cellular link of U_k is given as

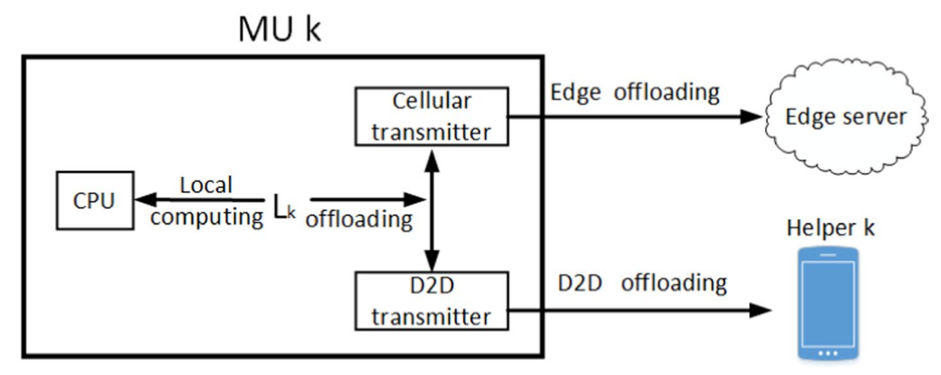

Fig. 2 Computing procedure

$$r_k^{\text{edge}} = B_1 \log_2 \left(1 + \frac{p_k^{\text{edge}} \left|h_k^{\text{edge}}\right|^2}{N_1}\right), \tag{2}$$

where N_1 denotes the background noise power for cellular communication.

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 local 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

The time division multiple access scheme is adopted in D2D communication. A time frame is divided into K time slots, corresponding to K MUs. U_k can only communicate with H_n in the corresponding time slot. For convenience, we normalize the time slot as $t_k (t_k \in [0, 1])$. Therefore, the average transmission rate between U_k and H_n in a time frame can be expressed as $R_{k,n} = t_k r_{k,n}^{\text{d2d}}$. The transmission delay is denoted as $D_{k,n}^{\text{d2d,t}} = \frac{L_k}{R_{k,n}} = \frac{L_k}{t_k r_{k,n}^{\text{d2d}}}$, and the computing delay of H_n is given as $D_{k,n}^{\text{d2d,c}} = \frac{L_k}{V_n^{\text{help}}}$. We assume each helper starts processing data only after the whole data are received, thus the D2D offloading delay for U_k can be expressed as

$$D_k^{\text{d2d}} = D_{k,n}^{\text{d2d,t}} + D_{k,n}^{\text{d2d,c}}. \tag{5}$$

Similar to Sect. 2.2.1, we define the system D2D offloading delay as the maximum of the D2D offloading delay of the MUs, i.e.,

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

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 (calculate) its task until the task of the last MU has been transmitted (calculated) completely. Besides, each MU's task can be calculated only after the data reception is finished. For example, U_1 first offloads tasks to the edge server, and the edge server performs calculations immediately after the task is transmitted. At the same time, U_2 starts to offload the task. The subsequent MUs carry out transmission and calculation in turn. If the task offloaded to the edge cloud has been computed and the next user's task has not yet been transmitted, the server is in a waiting state. Then, the total latency consumed to executed the first 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 $D_k^{\text{tran}} = \frac{L_k}{r_k^{\text{edge}}}$ 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 Sub-optimal solution to the full offloading scenario

In this section, we analyze the latency-minimization problem for edge cloud computing model and give specific optimization problems. Then, we propose an heuristic algorithm to solve the problem.

3.1 Problem formulation

In the full offloading scenario, all MUs' tasks are offloaded to the edge server for processing. According to Sect.2.2.3, the delay of MEC offloading includes three parts: the calculation delay of the edge server, the waiting delay of the edge server, and the transmission delay of the first MU. The computation resource of edge server is fixed, and the computation delay is determined when the MUs' computation task is generated. However, the waiting delay of the edge server and the transmission delay of the first MU are related to the task offloading order of MUs. We introduce a vector $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \ldots, \varphi_K]^{\mathrm{T}} \in \mathbb{R}^{K\times 1}$ which determines the transmission (calculation) order of the MUs, i.e., φ_k means that

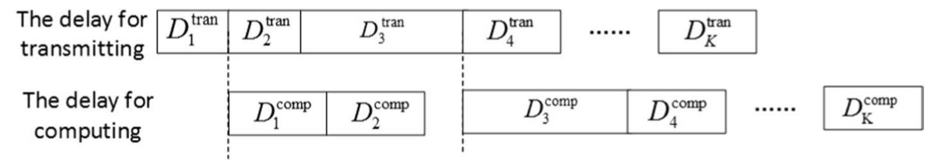

Fig. 3 MEC offloading

the task of the φ_kth MU is offloaded at the kth position. The total delay of this scenario can be obtained by (7) and (9) and is expressed as:

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

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

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

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

where (11b) denotes the value range of the index vector.

3.2 Sub-optimal solution

There will be $K!$ offloading sequences for K users. If we use the exhaustive method to search for the optimal solution of **P1**, the algorithm time complexity is $O(K!)$ which is significantly high. In this section, 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. Then, we divide all MUs into two parts: $\{U_{\varphi_1}, U_{\varphi_2}, \ldots, U_{\varphi_{k_1}}\}$, and $\{U_{\varphi_{(k_1+1)}}, U_{\varphi_{(k_1+2)}}, \ldots, U_{\varphi_K}\}$. Ensure that the waiting delay of the edge server is zero when the first part of MUs transmit(calculate) the tasks.

Lemma 1 *If $D_{\varphi_{k+1}}^{\text{tran}} < D_{\varphi_k}^{\text{comp}}$ is satisfied for $\{U_{\varphi_1}, U_{\varphi_2}, \ldots, U_{\varphi_{k_1}}\}$, the server has no idle period when calculating the tasks of these MUs.*

1 *Proof*

See "Appendix A". □

Algorithm 1 Heuristic algorithm for full offloading.

```
1:  Sort the MUs in descending order according to $L_k$.
2:  Set $k = 1$ and $n = 0$.
3:  while $k < K$ do
4:      if $D_{\varphi_{k+1}}^{\text{tran}} > D_{\varphi_k}^{\text{comp}}$ then
5:          $n = k, temp = \varphi_{k+1}$.
6:          Do
7:          $\varphi_{n+1} = \varphi_{n+2}$.
8:          $n = n + 1$.
9:          Until $n = K - 2$
10:         $\varphi_K = temp$.
11:     else
12:         Update the calculation delay of k MU tasks and edge server waiting delay by (7) and (8).
13:         Update $k = k + 1$.
14:     end if
15: end while
16: Return $D_K^{\text{edge}} + D_{\varphi_1}^{\text{tran}}, \varphi$
```

When processing the tasks of the first k_1 MUs, some tasks will be cached on the edge server. Therefore, when processing subsequent MUs, even if $D_{\varphi_{k+1}}^{\text{tran}} < D_{\varphi_k}^{\text{comp}}$ is not met, the server *may not be in a waiting state but continue to calculate the previously cached computation tasks.*

The result of Algorithm 1 is not the optimal solution of **P1**, but the sub-optimal solution. The optimal solution and the sub-optimal solution are denoted as D_{*}^{edge} and D_{**}^{edge}, respectively. We consider an ideal situation: The edge server is not in a waiting situation during the entire process of computing all MU tasks. The system delay in this case is expressed as $D_{\text{ide}}^{\text{edge}} = \sum_{i=1}^{K} D_{\varphi_i}^{\text{comp}} + D_{\varphi_1}^{\text{tran}}$. Obviously, $D_{\text{ide}}^{\text{edge}} \leq D_{*}^{\text{edge}} \leq D_{**}^{\text{edge}}$. The simulation parameters of Fig. 4 are the same as in Sect. 6. From Fig. 4 we can clearly see that $D_{\text{ide}}^{\text{edge}}$ and D_{**}^{edge} are not much different. D_{*}^{edge} is between $D_{\text{ide}}^{\text{edge}}$ and D_{**}^{edge}, so it is reasonable to replace the optimal solution with a suboptimal solution. As mentioned above, the time complexity of finding the optimal solution of **P1** is $O(K!)$, and the time complexity of Algorithm 1 is $O(K^2)$. Thus, our proposed algorithm can greatly reduce the time complexity with little performance loss, as shown in Fig. 4.

4 Sub-optimal solution to the partial offloading scenario

In the above sections, we have analyzed the local computing model and the edge cloud computing model, and we have discussed the performance of the full offloading scenario. In this section, we study the performance of partial offloading scenario. By utilizing the computing resources in both MUs and edge cloud, the system delay can be shorter than that of the full offloading scenario. In the following, we first formulate the latency-minimization problem and then give out the solution.

4.1 Problem formulation

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. According to 2.2.1, 2.2.3, and 3.1, the local delay and the offloading delay in the partial offloading scenario are given as

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

and

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

respectively. Then the system delay minimization problem is formulated as

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

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

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

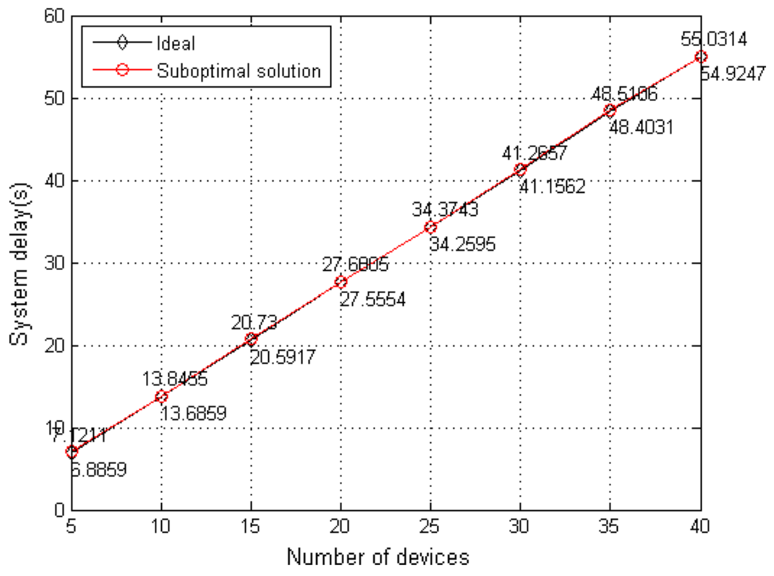

Fig. 4 Comparison of solutions to P1

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

4.2 Sub-optimal solution

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

Lemma 2 *Given the computation and communication resources and the amount of data, the total system delay cannot be minimal as long as some computing or communication resources are idle.*

1 *Proof*

See "Appendix B". □

It can be inferred from Lemma 2 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

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

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

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

Lemma 3 *If and only if* $D^{\mathrm{loc}} = D^{\mathrm{edge}}$, *the sub-optimal solution of* **P2** *is obtained.*

1 ***Proof***

See "Appendix C". □

According to Lemma 3, 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 $D^{\text{loc}}_{(0)}$ and $D^{\text{edge}}_{(0)}$ according to (16) and Algorithm 1, respectively.
5: while $\left| D^{\text{loc}}_{(n)} - D^{\text{edge}}_{(n)} \right| > \epsilon$ do
6:         Recalculate $D^{\text{loc}}_{(n)}$ and $D^{\text{edge}}_{(n)}$ according to (16) and Algorithm 1, respectively.
7:         Update $n = n + 1$, $h = h + s$
8: end while
9: Return $D^{\text{loc}}_{(n)}$.
```

With the increase of h, the local calculation delay linearly increases, while the edge offloading delay linearly decreases. Therefore, Algorithm 2 must converge. In addition, the complexity of Algorithm 2 mainly depends on the number of iterations, which is determined by the iteration step s. In the iterative process, D^{edge} is calculated by Algorithm 1, so the time complexity of Algorithm 2 is $O(\frac{K^2}{s})$. Since Algorithm 1 can only obtain the sub-optimal solution of **P1**, the result obtained by Algorithm 2 is the sub-optimal solution of **P2**.

5 Sub-optimal solution to the D2D-enabled partial offloading scenario

In the previous section, we studied the performance of partial offloading scenario. On this basis, 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. The system delay is further reduced by utilizing the computation resource in all MUs, helpers, and edge cloud. In the following, we first formulate the latency-minimization problem which is not convex and it is difficult to solve it. We decompose it into two problems to solve separately and then get the solution of the original problem.

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

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

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

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

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

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

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

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

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

It is not easy to find the sub-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 3.

Lemma 4 *If and only if* $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$, *the sub-optimal solution to* **P3** *is obtained.*

1 ***Proof***

See "Appendix D". □

Although D^{loc} and D^{d2d} depend on $\boldsymbol{\alpha}$, according to (17) and (19), we can see both D^{loc} and D^{d2d} are linearly inversely related to $\boldsymbol{\alpha}$. We start by solving over $\boldsymbol{\beta}$ and $\boldsymbol{t}$ for a fixed $\boldsymbol{\alpha}$

and $\boldsymbol{\varphi}$, the magnitude relation of the obtained result i.e., D^{loc^*} and D^{d2d^*} do not change any more when $\boldsymbol{\alpha}$ changes. Note that the optimal $\boldsymbol{\beta}$ and $\boldsymbol{t}$ do not depend on $\boldsymbol{\alpha}$ and $\boldsymbol{\varphi}$. In other words, the two variable sets $\{\boldsymbol{\alpha}, \boldsymbol{\varphi}\}$ and $\{\boldsymbol{\beta}, \boldsymbol{t}\}$ are independent of each other. Hence the proposed splitting into subproblems **P4** and **P5** do not suffer any optimality loss compared to **P3**. Two subproblems **P4** and **P5** are as follows:

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

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

and

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

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

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

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

Lemma 5 *The sub-optimal solution obtained by* **P4** *and* **P5** *is equivalent to the sub-optimal solution of* **P3**.

1 ***Proof***

See "Appendix E". □

5.3 Sub-optimal solution

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

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

The D2D offload delay consists of two parts: the D2D-link transmission delay and the helper processing delay. The time division multiple access scheme is adopted in D2D communication, and $\boldsymbol{t}$ represents the normalized length of each time slot. It is assumed that the D2D communication time of MUs can be allocated arbitrarily. In order to enable all helpers to complete computing tasks at the same time, we adopt an on-demand distribution strategy. If the helper has sufficient computing resources, we will allocate more communication resources t_k to the D2D link; otherwise, we allocate less communication resources t_k to the D2D link. Specifically, we make that the ratio of the communication rate of all D2D links to the computing resources of the device is equal and a certain value, i.e.,

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

which leads to

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

and

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

If the D2D communication resource $\boldsymbol{t}$ is not allocated by (26), the computational resources of some helpers are bound to be in an idle state. According to Lemma 1, the system delay is definitely not minimal. Therefore, on-demand distribution strategy (25) does not result in optimality loss compared to the optimization problem **P3**. From (18) and (25), the helper processing delay can be expressed as

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

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

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

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

whose sub-optimal solution is given by the following lemma.

Lemma 6 *The sub-optimal solution of* **P6** *is given by*

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

1 ***Proof***

See "Appendix F". □

Based on Lemma 6, 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 sub-optimal solution D^{sys^*} of **P5** can be obtained. And then, it can be seen from Lemma 5 that D^{sys^*} is also the sub-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. 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., $p_k^{\mathrm{edge}} = p_{k,n}^{\mathrm{d2d}} = 24$ dBm. The data size L_k of U_k follows a uniform distribution within $L_k \in [10, 100]$ Mbits. The computing capacity V_k^{loc} of U_k and V_n^{help} of H_n follow a uniform distribution within $V_k^{\mathrm{loc}} \in [0.5, 2]$ Mbps and $V_n^{\mathrm{help}} \in [0, 2]$ Mbps, respectively. 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 [13] for comparison. In this model, the cellular communication resource and the computing resource of the edge server are allocated to each MU. 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 [13]: In this scenario, all MUs' tasks are offloaded to the edge server for execution.
(2) Existing partial offloading in [13]: 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.

Figure 5 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 proposed full offloading, we can observe that proposed 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. In the existing full offloading model, MUs first transfer computing tasks to the edge cloud, and then the edge cloud server starts computing. Therefore, the system delay is the sum of the cellular link transmission delay and the edge cloud server calculation delay. However, in the proposed

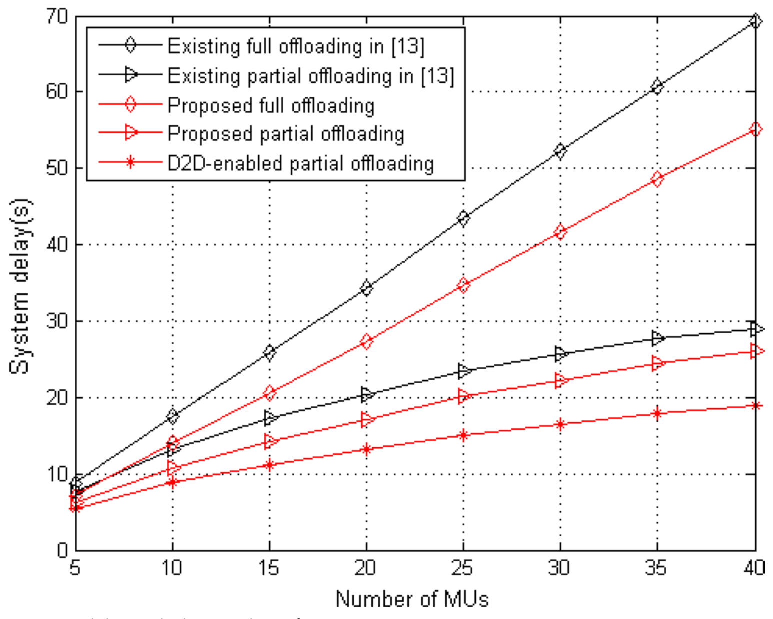

Fig. 5 The system delay with the number of MUs

full offloading model, MUs transmission computing task and the edge cloud processing computing task are performed at the same time. Here, the edge cloud computing delay is greater than MUs transmission delay so the system delay is almost equal to the computing delay. We can get the same conclusion by comparing the curves of existing partial offloading and proposed partial offloading . And then, the performance of D2D-enabled partial offloading is the best among the five scenarios. It is intuitive that when idle helpers computing resources are utilized, the delay of the system must decrease. When there are more 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. 6, 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 and 2.5 Mbps. From this figure, the advantage of proposed partial offloading is more prominent when the local computing resources are scarce compared to existing partial offloading. The reason can be explained as follows. When there are fewer local computing resources, more data will be offloaded 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 is from 0.75 to 2.5 Mbps, and D2D-enabled partial offloading model shows better performance than other four models.

Figure 7 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.

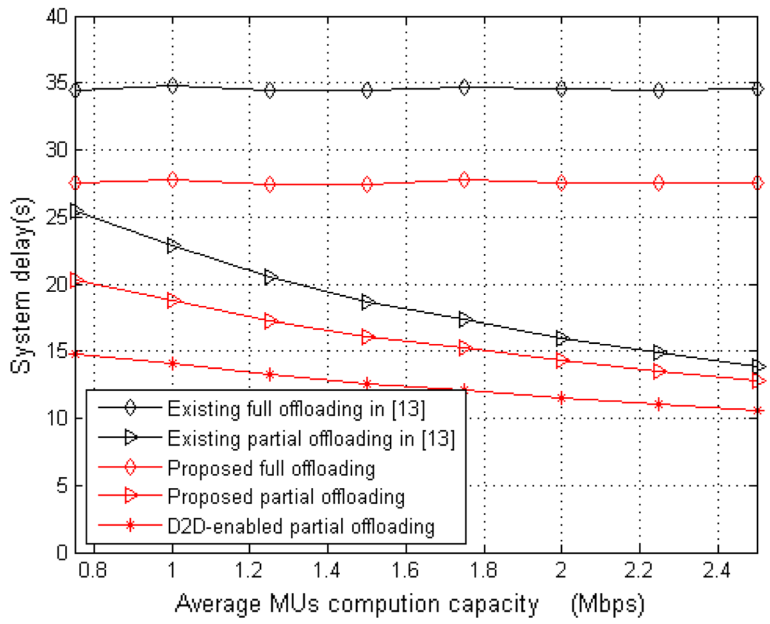

Fig. 6 The system delay versus the average computation capacity of MUs

Next, the performance gap between existing full offloading and proposed full offloading becomes more evident with the increase of the calculating resource of the edge server. This is because when the computing resources of the edge server are sufficient, the computing delay is small, and the delay of proposed full offloading is also small. However, the delay for existing full offloading decreases slowly because it contains the transmission delay that has not changed. We have the similar observation by comparison of the existing partial offloading and proposed 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.

Figure 8 shows the minimum system delay versus the radius of the BS. The BS radius is taken from 100 to 450 m, 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 is 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 propose the three scenarios are also applicable when the wireless channel is poor.

Figure 9 shows the impact of the average size of MU data on the minimum system delay in five different scenarios. Then the average size of MUs' data is taken from 55 to 90 Mbits, and we assume that there are 20 MUs in the system. More computing tasks are generated, and MUs offload more computing tasks to edge cloud servers.

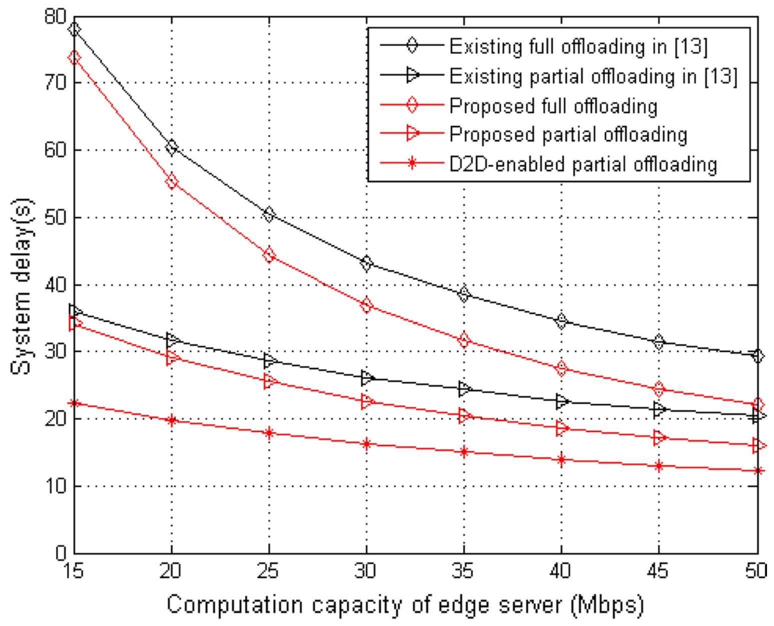

Fig. 7 The system delay with the computation capacity of the edge server with K=20

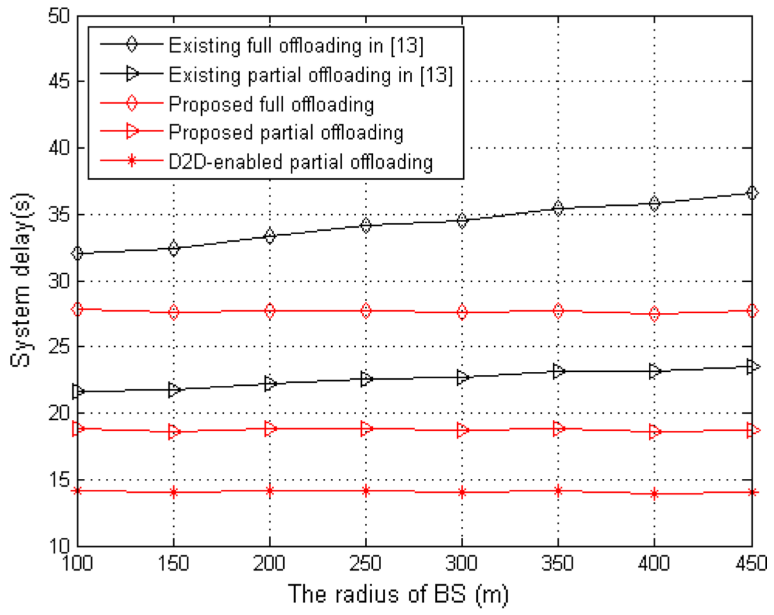

Fig. 8 The system delay versus the radius of BS

The advantages of the proposed offloading strategy of parallel transmission and calculation are more prominent. From this figure, compared with the existing full offloading and existing partial offloading models, the corresponding proposed full offloading and proposed partial offloading have great performance advantages. 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 proposed 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, were studied and compared. First, we proposed a heuristic algorithm to solve the optimization problem of the proposed full offloading. Although this algorithm could only get a suboptimal solution, the complexity of this algorithm was greatly reduced compared to the complexity of finding the optimal solution. Second, on the basis of Algorithm 1, Algorithm 2 is proposed to find the solution of the partial offloading scenario. Third, the D2D-enabled partial offloading scenario was complicate and we decomposed it into two sub-problems to solve separately. Then it was proved that the solutions of the two sub-problems could form a satisfying solution to the original problem. Finally, the numerical results validated the performance of the proposed algorithms.

In this paper, we utilized the overlay mode of D2D communication, which required a wider spectrum. Considering the limited spectrum resources, we will introduce the underlay mode of D2D communication into the parallel mobile edge computing system

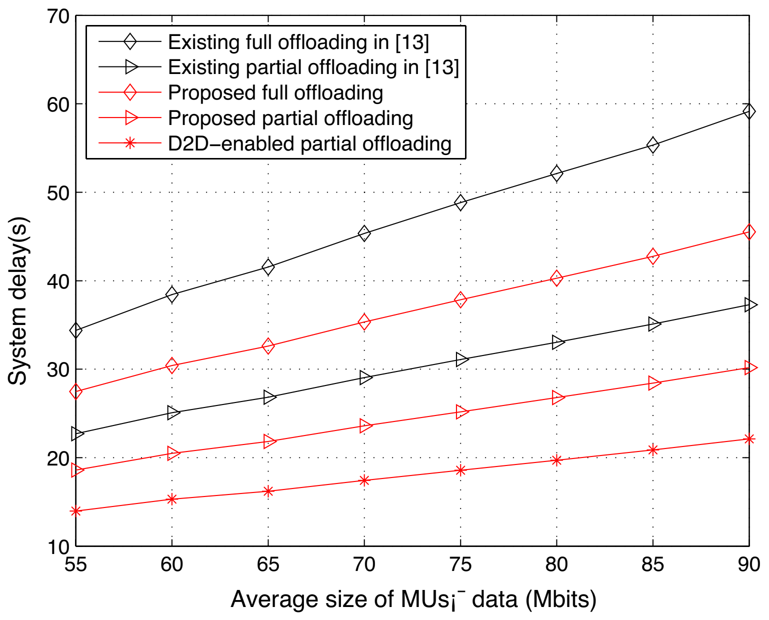

Fig. 9 The system delay versus the average size of the MUs' data

in the future work. Another significant direction for future work is to optimize energy efficiency in this multi-user system.

Appendices

A. Proof of Lemma 1

1. When $k = 1$, according to (7), $D_{\varphi_1}^{\text{edge}} = D_{\varphi_1}^{\text{comp}}$. According to (8), $D_{\varphi_1}^{\text{wait}} = \max\left(D_{\varphi_2}^{\text{tran}} - D_{\varphi_1}^{\text{edge}}, 0\right) = \max\left(D_{\varphi_2}^{\text{tran}} - D_{\varphi_1}^{\text{comp}}, 0\right)$. $D_{\varphi_2}^{\text{tran}} < D_{\varphi_1}^{\text{comp}}$ is established, so the edge server has no idle period.

2. Suppose that when $k = k_0$, the theorem holds, i.e., $D_{\varphi_{k_0}}^{\text{edge}} = \sum_{i=1}^{k_0} D_{\varphi_i}^{\text{comp}}$. According to (8), $D_{\varphi_{k_0}}^{\text{wait}} = \max\left(\sum_{i=2}^{k_0+1} D_{\varphi_i}^{\text{tran}} - D_{\varphi_{k_0}}^{\text{edge}}, 0\right) = \max\left[\sum_{i=1}^{k_0}\left(D_{\varphi_{i+1}}^{\text{tran}} - D_{\varphi_i}^{\text{comp}}\right), 0\right] = 0$. When $k = k_0 + 1$, $D_{\varphi_{k_0+1}}^{\text{wait}} = D_{\varphi_{k_0}}^{\text{wait}} + \max\left(D_{\varphi_{k_0+2}}^{\text{tran}} - D_{\varphi_{k_0+1}}^{\text{comp}}, 0\right)$. $D_{\varphi_{k_0+2}}^{\text{tran}} < D_{\varphi_{k_0+1}}^{\text{comp}}$ is established, so $D_{\varphi_{k_0+1}}^{\text{wait}} = 0$. Therefore, the server has no idle period when $k = k_0 + 1$.

Combining 1 and 2, Lemma 1 is proved.

B. Proof of Lemma 2

The total data in the system are represented as $L^{\text{sys}} = \sum_{k=1}^{K} L_k$. The local computing resource of all the MUs is represented as $V^{\text{loc}} = \sum_{k=1}^{K} V_k^{\text{loc}}$. The rate of D2D offloading processing data is

$$R_k^{\text{help}} = \frac{1}{\frac{1}{r_k^{\text{d2d}}} + \frac{1}{V_k^{\text{help}}}} = \frac{r_k^{\text{d2d}} V_k^{\text{help}}}{r_k^{\text{d2d}} + V_k^{\text{help}}}. \tag{33}$$

The total rate of helpers processing data is $R^{\text{help}} = \sum_{k=1}^{K} R_k^{\text{help}}$. 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 are 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 2 is proved.

C. Proof of Lemma 3

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 edge cloud server will be idle for the $\left(D^{\text{loc}} - D^{\text{edge}}\right)$. According to Lemma 2, this situation is not a sub-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 2, this situation is not the sub-optimal solution. So, when $D^{\text{loc}} = D^{\text{edge}}$, the sub-optimal solution of **P2** will be got.

Necessity: According to the negative proposition of Lemma 2, if the **P2** obtains the sub-optimal solution, no resources in the system will be in an idle state during the raw data are calculated. In other words, local device and edge cloud server simultaneously complete the computing tasks, i.e., $D^{\text{loc}} = D^{\text{edge}}$.

In summary, Lemma 3 is proved.

D. Proof of Lemma 4

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 2, this situation is not an sub-optimal solution.
2) Same as 1), when $D^{\text{loc}} > D^{\text{edge}} > D^{\text{d2d}}$, this situation is not the sub-optimal solution.
3) Same as 1), when $D^{\text{edge}} > D^{\text{loc}} > D^{\text{d2d}}$, this situation is not the sub-optimal solution.
4) Same as 1), when $D^{\text{d2d}} > D^{\text{loc}} > D^{\text{edge}}$, this situation is not the sub-optimal solution.
5) Same as 1), when $D^{\text{edge}} > D^{\text{d2d}} > D^{\text{loc}}$, this situation is not the sub-optimal solution.
6) Same as 1), when $D^{\text{d2d}} > D^{\text{edge}} > D^{\text{loc}}$, this situation is not the sub-optimal solution.

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

Necessity:

According to the negative proposition of Lemma 2, if the **P3** obtains the sub-optimal solution, no resources in the system will be in an idle state during the raw data are calculated. In other words, the local, helper, and edge server simultaneously complete the computing tasks. That is $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$.

In summary, Lemma 4 is proved.

E. Proof of Lemma 5

According to Lemma 4, the sub-optimal solution to problem **P3** is obtained if and only if $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$. $D^{\text{loc}} = D^{\text{d2d}} = D^{\text{edge}}$ 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 constraint conditions (22b).

According to Lemma 3, it is equivalent to solving $\min_{\boldsymbol{\beta},\boldsymbol{t}} \max\left(D^{\mathrm{loc}}, D^{\mathrm{d2d}}\right)$ to obtain $D^{\mathrm{l,d}^*}$. Then let $D^{\mathrm{l,d}^*} = D^{\mathrm{edge}}$ under constraint conditions (23b), i.e., solve $\min_{\boldsymbol{\alpha},\boldsymbol{\varphi}} \max\left(D^{\mathrm{l,d}^*}, D^{\mathrm{edge}}\right)$, and get D^{sys^*}. D^{sys^*} guarantees $D^{\mathrm{loc}} = D^{\mathrm{d2d}} = D^{\mathrm{edge}}$, so D^{sys^*} is the sub-optimal solution of **P3**.

F. Proof of Lemma 6

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

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

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

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

Here the sum of the locally computed and D2D offloaded tasks is $(1-\alpha_k)L_k$. Similar to 4.2, the ratio of the raw data for local computing is

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

The combined rate of D2D offloading processing tasks is $\frac{g}{g+1} V_k^{\mathrm{help}}$. Similar to (39), the ratio of the raw data for D2D offloading is

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

When $\alpha_k = 1$, all the computation tasks of MUs are offloaded to the edge cloud server. There are no computation tasks locally, so β_k can take any value. When $\alpha_k \neq 1$, there are three cases of raw data segmentation for local computing and D2D computing:

1) $\beta_k = 1$, in this case, all data are placed locally.
2) $\beta_k = 0$, in this case, all data are offloaded to the helpers.
3) $\beta_k \neq 1$ and $\beta_k \neq 0$, in this case, the two split ratios are

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

and

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

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 sub-optimal solution of **P6** is certainly not obtained when $\beta_k = 1$ or $\beta_k = 0$. So the **P6** solution is the sub-optimal solution $D^{\text{l,d}^*}$ at $\beta_k^* = \frac{V_k^{\text{loc}}}{V_k^{\text{loc}} + \frac{g}{g+1} V_k^{\text{help}}}$.

Since β_k can take any value when $\alpha_k = 1$, we make $\beta_k^* = \frac{V_k^{\text{loc}}}{V_k^{\text{loc}} + \frac{g}{g+1} V_k^{\text{help}}}$. In summary, the **P6** solution is the sub-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.

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

Authors' contributions

HBZ, JZ, and YC conceived the study. YC and LR designed and modeled. LR, YC, and JZ analyzed the performance. LR, YC, and JZ interpreted the results and wrote the manuscript. All authors read and approved the final manuscript.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grant U1805262, 61871446, Jiangsu Provincial Key Research and Development Program (No. BE2020084-5).

Availability of data and materials

Not applicable.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 10 August 2020 Accepted: 23 May 2021
Published online: 07 June 2021

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 (2011)
2. J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
3. T.X. Tran, A. Hajisami, P. Pandey, D. Pompili, Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges. IEEE Commun. Mag. **55**(4), 54–61 (2017)
4. L. Ji, S. Guo, Energy-efficient cooperative resource allocation in wireless powered mobile edge computing. IEEE Internet Things J **6**(3), 4744–4754 (2019)

5. C. You, K. Huang, H. Chae, Energy efficient mobile cloud computing powered by wireless energy transfer. IEEE J. Sel. Areas Commun. **34**(5), 1757–1771 (2016)
6. Y. Pan, M. Chen, Z. Yang, N. Huang, M. Shikh-Bahaei, Energy-efficient noma-based mobile edge computing offloading. IEEE Commun. Lett. **23**(2), 310–313 (2019)
7. C. You, K. Huang, H. Chae, B. Kim, Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16**(3), 1397–1411 (2017)
8. M. Li, F.R. Yu, P. Si, Y. Zhang, Energy-efficient machine-to-machine (M2M) communications in virtualized cellular networks with mobile edge computing (MEC). IEEE Trans. Mob. Comput. **18**(7), 1541–1555 (2019)
9. W. Xia, J. Zhang, T.Q.S. Quek, S. Jin, H. Zhu, Power minimization-based joint task scheduling and resource allocation in downlink c-ran. IEEE Trans. Wirel. Commun. **17**(11), 7268–7280 (2018)
10. K. Guo, M. Sheng, T.Q.S. Quek, Z. Qiu, Task offloading and scheduling in fog ran: a parallel communication and computation perspective. IEEE Wirel. Commun. Lett. **9**(2), 215–218 (2020)
11. J. Liu, Y. Mao, J. Zhang, K.B. Letaief, Delay-optimal computation task scheduling for mobile-edge computing systems, in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455 (2016)
12. Y. Kao, B. Krishnamachari, M. Ra, F. Bai, Hermes: Latency optimal task assignment for resource-constrained mobile computing. IEEE Trans. Mob. Comput. **16**(11), 3056–3069 (2017)
13. J. Ren, G. Yu, Y. Cai, Y. He, Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **17**(8), 5506–5519 (2018)
14. L. Yang, B. Liu, J. Cao, Y. Sahni, 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)*, pp. 246–253 (2017)
15. W. Xia, T.Q.S. Quek, J. Zhang, S. Jin, H. Zhu, Programmable hierarchical c-ran: From task scheduling to resource allocation. IEEE Trans. Wirel. Commun. **18**(3), 2003–2016 (2019)
16. Y. Mao, J. Zhang, S.H. Song, K.B. Letaief, Power-delay tradeoff in multi-user mobile-edge computing systems, in *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6 (2016)
17. X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2016)
18. U. Saleem, Y. Liu, S. Jangsher, X. Tao, Y. Li, Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. IEEE Trans. Veh. Technol. **69**(4), 4472–4486 (2020)
19. C. You, K. Huang, Exploiting non-causal cpu-state information for energy-efficient mobile cooperative computing. IEEE Trans. Wirel. Commun. **17**(6), 4104–4117 (2018)
20. Y. He, J. Ren, G. Yu, Y. Cai, D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. IEEE Trans. Wirel. Commun. **18**(3), 1750–1763 (2019)
21. W. Hu, G. Cao, Quality-aware traffic offloading in wireless networks. IEEE Trans. Mob. Comput. **16**(11), 3182–3195 (2017)
22. L. Pu, X. Chen, J. Xu, X. Fu, D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration. IEEE J. Sel. Areas Commun. **34**(12), 3887–3901 (2016)

Publisher's Note