Pricing-Based Partial Computation Offloading in Mobile Edge Computing

Lanhui Li(✉) and Tiejun Lv

Beijing University of Posts and Telecommunications, Beijing 100876, China
{lilanhui,lvtiejun}@bupt.edu.cn

Abstract. For mobile devices (MDs) and Internet of Things (IoT) devices with limited computing capacity and battery, offloading part of tasks to the mobile edge computing (MEC) server is attractive. In this paper, we propose a joint partial computation offloading and pricing scheme in a multi-user MEC system. Firstly, we establish MD's cost model and MEC server's revenue model in terms of money. Secondly, we investigate MD's cost minimization partial offloading strategy to jointly control MD's task allocation, local CPU frequency and the amount of computational resource blocks (CRBs) requested. Finally, we formulate the revenue maximization problem for MEC server with limited computing capacity, a heuristic algorithm is proposed for MEC server to find the optimal service price. Numerical results verify the effectiveness of our proposed scheme in cost saving and pricing.

Keywords: Mobile edge computing · Partial computation offloading · MEC server · Pricing scheme

1 Introduction

Recent years, with the rapid development of mobile computing and communication techniques, more and more computation-demanding and latency-sensitive mobile applications are appearing, such as face recognition, argument reality and natural language processing [1]. However, due to the finite storage and computing capacity, mobile devices (MDs) are still not able to handle these tasks locally [2].

To meet these challenges, mobile edge computing (MEC), which works at the close proximity of MDs has been advocated [3,4]. MEC offloading has been recognized as a promising paradigm to enhance the computing capacity of MDs. For energy and computing capacity constrained MDs, offloading tasks to adjacent MEC server is effective in reducing latency, saving energy and extending the battery lifetime. Comparing with mobile cloud computing (MCC), MEC can reduce transmitting delay and alleviate network burden.

Recent years, many researches have investigated the computation offloading problem in mobile edge networks. The authors of [5] presented an energy-efficient computation offloading scheme in 5G heterogeneous networks. In [6], the authors

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 312, pp. 3–14, 2020.
https://doi.org/10.1007/978-3-030-41114-5_1

investigated the cost and latency trade-off in the process of mobile code offloading. In [7], the authors studied the economic performance in the D2D assisted data offloading network. In [8], the authors proposed a distributed full computation offloading and resource allocation mechanism in heterogeneous networks, however, the service price has not been optimized in this paper. The authors of [9,10] investigated partial computation offloading in MEC system. Although there have been many studies working on the energy efficiency and resource allocation in MEC system, the economic performance has seldom been investigated. Moreover, none of these works jointly optimized the revenue of MEC server and the cost of MDs with partial computation offloading. As partial computation offloading offers MDs more flexibility in making offloading strategies, we adopt partial offloading in this paper.

As money expenditure is more intuitive for MDs and MEC server, we aim to investigate the economic performance of partial computation offloading in this paper. The main contributions of this paper are summarized as follows:

- Firstly, we establish MD's cost model in terms of money and formulate its computation offloading problem. In the problem, MDs optimize its cost by jointly adjusting local execution ratio, CPU speed and computational resource blocks (CRBs) rent from MEC server. The minimization problem is subjected to the delay and computing capacity limits.
- Secondly, we formulate the revenue maximization problem of MEC server and propose a heuristic optimal pricing algorithm. In the pricing algorithm, we take the limitation of CRBs in MEC server into consideration.
- Finally, we conduct simulations to verify the proposed MEC offloading scheme. The relationships between CRB renting price, number of CRBs rented by MDs and the revenue of MEC server are analyzed. Simulation results show that our pricing algorithm converges quickly. We also compare our proposed cost minimization partial computation offloading scheme with other two computation schemes, simulation results indicate that our proposed offloading mechanism is more effective in cost saving.

The rest of the paper is organized as follows. Section 2 presents our system model and computation model. In Sect. 3, we propose the cost minimization partial offloading scheme of MDs together with the optimal pricing scheme of MEC server. The simulation results are presented and analyzed in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 System Model

As depicted in Fig. 1, we consider $\mathfrak{I} = \{1, 2, \ldots, I\}$ MDs and one MEC server in the system. We assume MDs have tasks need to be executed. The MEC server which is assumed to be deployed between the remote cloud server and MDs, generally near the small-cell base stations. MEC server's computing resource can be divided into multiple CRBs. The computing capacity of each CRB is f^{c} and the maximum number of CRBs is Q. If there are available CRBs at the MEC

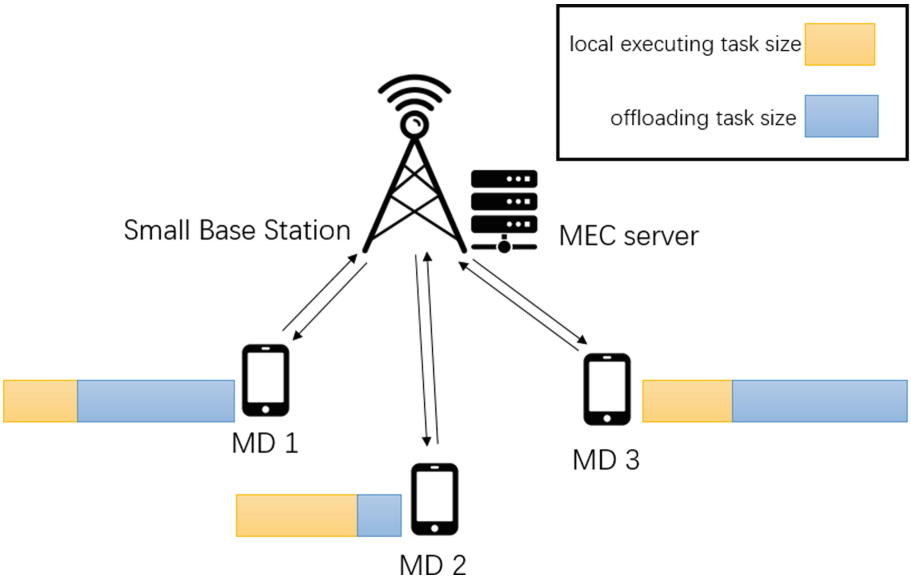

Fig. 1. System model

server, MDs can rent CRBs to compute its task. Note that the MEC server can handle different MDs' tasks independently. Thus, if errors happen in some tasks, these errors would not influence other tasks processed by the MEC server.

2.1 Computation Model

In this sub-section, we describe the local and offloading computation model of MD. We denote MD i' task as $T_i = \{m_i, c_i, t_i^{\max}\}$. Where m_i is the task size, c_i represents the number of CPU cycles required per bit; and $m_i c_i$ is the total CPU cycles needed by task T_i. $t_i^{\max}$ denotes the time limitation of task T_i, which means the task should be finished before $t_i^{\max}$. In this paper, we assume that MD's task can be divided. We define $\alpha_i\,(0 \leq \alpha_i \leq 1)$ as the ratio of the task that MD i computes locally. Thus, $1 - \alpha_i$ is the ratio of task that MD i offloads to the MEC server.

Local Computation Model. Same with [11], the energy consumed per CPU cycle of MD i is denoted as $P = k_i f_i^2$, where k_i is a parameter related to the structure of device, f_i denotes the local CPU speed of MD i. With DVS technology, MD can reduce energy consumption by adjusting f_i. According to the task model described above, the time and energy consumption of local computing is

$$t_i^{\text{local}} = \frac{\alpha_i m_i c_i}{f_i}, \tag{1}$$

$$e_i^{\text{local}} = \alpha_i m_i c_i k_i f_i^2. \tag{2}$$

MEC Offloading Model. The uplink transmitting rate of MD i can be defined as

$$r_i = B_i \log_2(1 + \frac{p_i d_i^{-v} |h_i|^2}{N_0}), \tag{3}$$

where B_i is the channel bandwidth between MD i and MEC server, d_i is the distance between MD and MEC server, h_i is the channel fading coefficient of the link and p_i is the transmitting power of MD i. Besides, v represents the path loss coefficient and N_0 is the background noise. As the output data size is generally very small, we do not consider the delay of receiving output data, like [12,13]. When the offloading data size of MD i is $(1 - \alpha_i)m_i$ bits, the consumed time and energy of transmitting can be written as

$$t_i^{\text{trans}} = \frac{(1 - \alpha_i)m_i}{r_i}, \tag{4}$$

$$e_i^{\text{trans}} = \frac{(1 - \alpha_i)p_i m_i}{r_i}. \tag{5}$$

The computation time in MEC server can be given by

$$t_i^{\text{mec}} = \frac{(1 - \alpha_i)m_i c_i}{q_i f^{\text{c}}}, \tag{6}$$

where q_i is the number of CRBs MD i rents from MEC server, and f^{c} (in cycles/s) is the computing capacity of each CRB. Thus, the time consumption of computation offloading can be represented as

$$t_i^{\text{off}} = t_i^{\text{mec}} + t_i^{\text{trans}} = \frac{(1 - \alpha_i)m_i c_i}{q_i f^{\text{c}}} + \frac{(1 - \alpha_i)m_i}{r_i}. \tag{7}$$

In partial computation offloading, MD and MEC server cope with the task simultaneously. The total time and energy consumption of completing task T_i can be denoted as

$$t_i^{\text{total}} = \max\{\frac{\alpha_i m_i c_i}{f_i}, \frac{(1 - \alpha_i)m_i c_i}{q_i f^{\text{c}}} + \frac{(1 - \alpha_i)m_i}{r_i}\}. \tag{8}$$

$$e_i^{\text{total}} = \alpha_i m_i c_i k_i f_i^2 + \frac{(1 - \alpha_i)p_i m_i}{r_i}. \tag{9}$$

3 Computation Offloading and Pricing Scheme

3.1 Cost Minimization Computation Offloading for MDs

The cost function of MD i is defined as MD's payment for renting CRBs plus the consumed energy. Here, the consumed energy is transformed into monetary

cost with λ_i, which represents the value of energy of MD i. Therefore, the cost function can be expressed as

$$C_i = \lambda_i e_i^{\text{total}} + sq_i t_i^{\text{mec}}, \tag{10}$$

where s (in cents/s) is the price per second for renting a CRB. Substituting (6) and (9) into (10), the cost function can be rewritten as

$$C_i(\alpha_i, f_i) = \lambda_i(\alpha_i m_i c_i k_i f_i^2 + \frac{p_i m_i (1-\alpha_i)}{r_i}) + \frac{(1-\alpha_i) m_i c_i s}{f^{\text{c}}}. \tag{11}$$

From (11), it is clear that once the service price s is given, C_i is affected by the local executing ratio α_i and frequency f_i. Therefore, α_i and f_i should be optimized. Thus, MD's cost minimization problem can be established as

$$\min_{\alpha_i, f_i} \quad C_i \tag{12a}$$

$$\text{s.t.} \quad t_i^{\text{total}} \le t_i^{\max}, \tag{12b}$$

$$0 \le \alpha_i \le 1, \tag{12c}$$

$$0 \le f_i \le f_i^{\max}, \tag{12d}$$

where $f_i^{\max}$ is the largest computing speed of MD i. In the above problem, condition (12b) ensures the task can be finished within the deadline; (12c) limits the domain of α_i; and (12d) is the maximum local computation frequency constraint.

Based on (12b) and (12d), we have

$$\alpha_i \le \frac{f_i^{\max} t_i^{\max}}{m_i c_i} \triangleq \alpha_i^{\text{u}}, \tag{13}$$

Thus, we have

$$\alpha_i^{\max} = \min\{\alpha_i^{\text{u}}, 1\}, \tag{14}$$

Therefore, the new range of α_i is

$$0 \le \alpha_i \le \alpha_i^{\max}. \tag{15}$$

From (11), we can know that $C_i(\alpha_i, f_i)$ increases monotonically with f_i. According to (12b), we have $t_i^{\text{local}} \le t_i^{\max}, t_i^{\text{off}} \le t_i^{\max}$ from which we can obtain $f_i \ge \frac{\alpha_i m_i c_i}{t_i^{\max}}$ and $q_i \ge \frac{m_i c_i r_i (1-\alpha_i)}{t_i^{\max} r_i f^{\text{c}} - (1-\alpha_i) m_i f^{\text{mec}}}$. Therefore, the optimal f_i is denoted as

$$f_i^*(\alpha_i) = \frac{\alpha_i m_i c_i}{t_i^{\max}}. \tag{16}$$

The amount of CRBs requested by MD i is written as

$$q_i^* = \left\lceil \frac{m_i c_i r_i (1-\alpha_i)}{t_i^{\max} r_i f^{\mathrm{c}} - (1-\alpha_i) m_i f^{\mathrm{c}}} \right\rceil. \tag{17}$$

Taking (15) and (16) into (12a)–(12d), we can transform the problem into

$$\min_{\alpha_i} \quad C_i, \tag{18a}$$

$$\text{s.t.}\, 0 \le \alpha_i \le \alpha_i^{\max}, \tag{18b}$$

where

$$C_i(\alpha_i) = \frac{k_i \lambda_i \alpha_i^3 m_i^3 c_i^3}{(t_i^{\max})^2} + \frac{\lambda_i p_i m_i (1-\alpha_i)}{r_i} + \frac{(1-\alpha_i) m_i c_i s}{f^{\mathrm{c}}}. \tag{19}$$

The first and second-order derivatives of C_i with respect to α_i can be denoted as

$$\frac{\partial C_i(\alpha_i)}{\partial \alpha_i} = \frac{3 k_i \lambda_i \alpha_i^2 m_i^3 c_i^3}{(t_i^{\max})^2} - \frac{\lambda_i p_i m_i}{r_i} - \frac{m_i c_i s}{f^{\mathrm{CRB}}}, \tag{20}$$

$$\frac{\partial C_i^2(\alpha_i)}{\partial \alpha_i^2} = \frac{6 k_i \lambda_i \alpha_i m_i^3 c_i^3}{(t_i^{\max})^2} \ge 0. \tag{21}$$

Let

$$\lim_{\alpha_i \to 0} \frac{\partial C_i(\alpha_i)}{\partial \alpha_i} = -\frac{\lambda_i p_i m_i}{r_i} - \frac{m_i c_i s}{f^{\mathrm{c}}} < 0, \tag{22}$$

$$\lim_{\alpha_i \to \alpha_i^{\max}} \frac{\partial C_i(\alpha_i)}{\partial \alpha_i} = \frac{3 k_i \lambda_i (\alpha_i^{\max})^2 m_i^3 c_i^3}{(t_i^{\max})^2} - \frac{\lambda_i p_i m_i}{r_i} - \frac{m_i c_i s}{f^{\mathrm{c}}}. \tag{23}$$

When (23) is smaller than zero, C_i is a decreasing function of α_i, the optimal local execution ratio is

$$\alpha_i^* = \alpha_i^{\max}. \tag{24}$$

If (23) is greater than zero, the cost function of MD i is a strict convex function of α_i, then

$$\alpha_i^* = \sqrt{\frac{\lambda_i p_i m_i f^{\mathrm{c}} (t_i^{\max})^2 + m_i c_i r_i s (t_i^{\max})^2}{3 r_i k_i \lambda_i m_i^3 c_i^3 f^{\mathrm{c}}}}. \tag{25}$$

Therefore, $C_i(\alpha_i)$ must have a minimum value when $0 \le \alpha_i \le \alpha_i^{\max}$. Now, we have obtained the optimal local execution ratio α_i^*. By taking it into (16) and (17), the local computing speed f_i^* and the number of CRBs MD should request q_i^* can be obtained.

3.2 Optimal Pricing Scheme for MEC Server

In the MEC offloading system, earn profit from computation offloading service is the motivation of MEC server. Thus, pricing strategy is very important to MEC server. We aim to design a dynamic pricing strategy for MEC server, where MEC server charges MDs according to the number of CRBs requested and the length of time occupied. Take the service cost into consideration, MEC server's revenue function can be given by

$$R(s) = (s - \varepsilon) \sum_{i=1}^{I} q_i t_i^{\text{mec}}. \tag{26}$$

where ε (in cents/s) is the cost to maintain CRB's service per second. Then, the revenue maximization problem is established as

$$\max_{s} \quad R \tag{27a}$$

$$\text{s.t. } 0 \leq \sum_{i=1}^{I} q_i \leq Q, \tag{27b}$$

$$s \geq 0, \tag{27c}$$

where Q is the maximum number of CRBs that MEC server can offer. In this problem, (27b) limits the total requested CRBs would not be more than Q, and (27c) guarantees the price is positive.

From (17) and (25), we can know that q_i^* is a decreasing function of s. It's obvious that when the service price s increases, MDs tend to request less CRBs from MEC server. As the number of CRBs is limited, we need to find the lowest price $s^{\min}$ that guarantees the requested CRBs would not be more than Q. In addition, we set $s^{\max} = 10 * \varepsilon$.

Inspired by the simulate anneal algorithm, we propose the following Optimal Pricing Algorithm. In the following algorithm, U and $U^{\min}$ are the initial and lowest temperature while g is the temperature decreasing factor. This algorithm runs while $U > U^{\min}$ holds. Firstly, the lowest price is got by step 2–7, where ω is a constant used to control the price. In step 8 and 9, MEC server obtains a random initial price s_1 ($s^{\min} < s_1 < s^{\max}$) and calculate the corresponding revenue according to MDs' offloading decisions. In step 11–13, we get the next price by $s_{n+1} = s_n + random$ and calculate R_{n+1}. In step 14–18, we compare R_n and R_{n+1}, if the revenue is improved, we get the new price s_{n+1} directly. If not improved, s_{n+1} can be used as new price if $P = e^{-\tau/U} > random$, where $\tau = R(n) - R(n+1)$. With this step, we can jump out of the local optimal price. As the current temperature is cooled by $U = g * U$ in each iteration, the iteration will end when $U < U^{\min}$.

Algorithm 1. Optimal Pricing Algorithm for MEC Server

input: $s^{\max}$, Q, f^c, U, $U^{\min}$, g, $n = 1$, ε
Set the initial service price as $s_0 = 0$ and announce vehicles
Calculate $\sum_{i=1}^{I} q_i$ according to MDs' reply
while $\sum_{i=1}^{I} q_i > Q$
    Increase the service price by $s = s + \omega$
end while
$s^{\min} \leftarrow s$
Get the initial service price s_1 and announce MDs, $s^* \leftarrow s_1$
Calculate R_1 according to MDs' replies
while $U > U^{\min}$
    Moves randomly to get a new service price $s_{n+1} = s_n + random$
    MDs calculate q_i^* based on s_{n+1} and reply to the server
    Calculate R_{n+1} by (26)
    if $R_{n+1} > R_n$
        $s^* \leftarrow s_{n+1}$
    else if $P > random$
        $s^* \leftarrow s_{n+1}$
    end if
    Announce s^* to MDs
    $U = g * U$
    $n = n + 1$
end while
output: s^*

4 Simulations

In this section, we present the simulation results of our computation offloading mechanism. We firstly list the simulation parameters used, then we discuss the experiment results.

4.1 Simulation Settings

We list the main parameter settings in Table 1.

4.2 Simulation Results

In this subsection, the simulation results of our proposed scheme is presented.

Figure 2 describes the relationship between the service price s and the number of CRBs required by MDs. When s increases from 0 cents/s to 3 cents/s, the number of CRBs rented by MDs decreases obviously. When s rises to an unacceptable value, MDs do not require any CRBs at all, which implies $\alpha_i = 1$.

Table 1. Simulation settings

Parameter	Description	Value
I	Number of MDs	3–6
m_i	Size of task	200 kB–1300 KB
c_i	Cycles needed per bit by task	600–1000 cycles/bit
$t_i^{\max}$	Latency constraint of task	5–10 s
$f_i^{\max}$	Maximum computing capacity of MD i	0.5–1 Gigacycles/s
p_i	Transmitting power of MD i	0.1–0.3 W
k	Local energy consumption coefficient	1×10^{-24}
λ_i	Energy-Money coefficient of MD i	1.5–2.4 cents/W
f^{c}	Computing capacity of CRB	0.1 Gigacycles/s
Q	Total number of CRBs	80
ε	MEC server's cost	0.3–1 cents/s

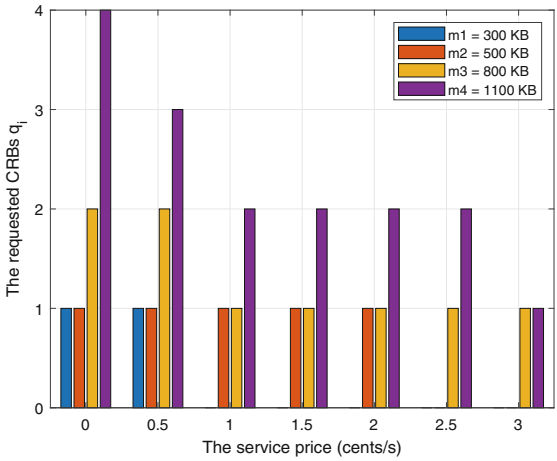

Fig. 2. The requested CRBs q_i versus service price s

The relationship between price s and revenue of MEC server is presented in Fig. 3. When $s = 0$ cents/s, the revenue of MEC server is not positive, which indicates the MEC server is in a loss state. Obviously, R rises with s in the beginning, after reaching the maximum value, R starts to decline with s. It's obvious that MEC server with higher service cost has relatively low revenue. For MEC server, reducing the service cost ε is an effective way to gain more profit.

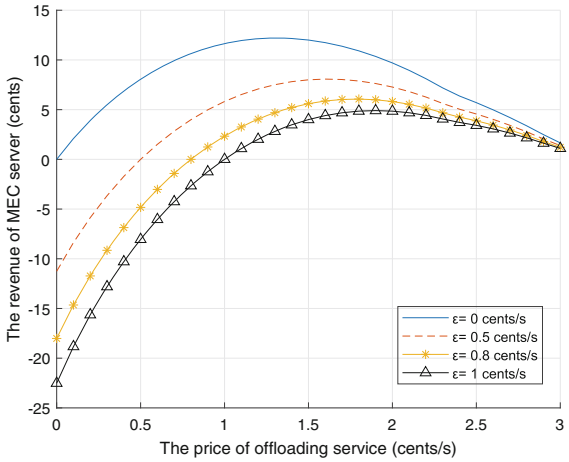

Fig. 3. The revenue R versus service price s

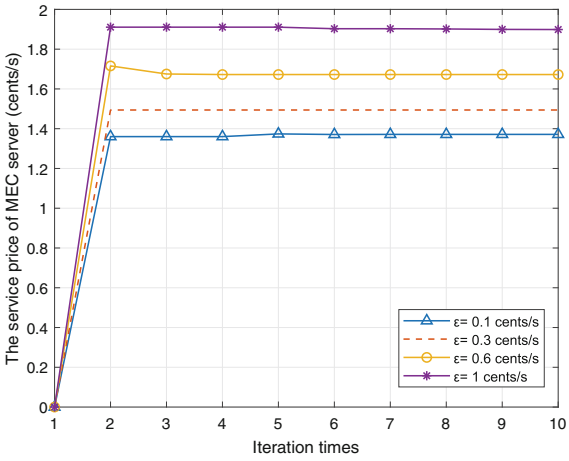

Fig. 4. Service price s versus iteration times

Figure 4 shows the service price got by the proposed Optimal Pricing Algorithm. As shown in the picture, the algorithm converges very fast, which implies that our proposed algorithm is very effective in pricing. When providing service to the same MDs, MEC server with higher cost has a higher service price correspondingly, which verifies the trend in Fig. 3.

In Fig. 5, we compare our computation scheme with other two schemes. (1) Local computation: MD computes the task locally and configures its local CPU speed to save cost. (2) Full computation offloading: MD chooses either local execution or full computation offloading according to the cost.

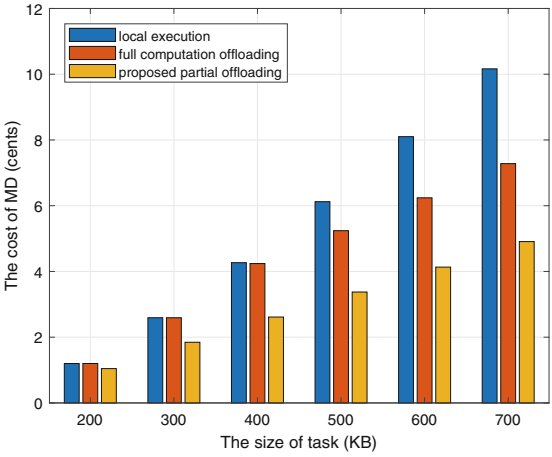

Fig. 5. Cost of MD i in different computing schemes

When the task size is not very large, the cost of the three execution mechanisms is similar. Nevertheless, the gap of cost between local execution and computation offloading becomes larger with the increase of task size. This is mainly because that the local energy is usually more valuable compared with the price of renting CRBs. It's obvious that our proposed offloading scheme keeps the lowest cost among the three schemes. Thus, we have proved the superiority of our proposed offloading strategy.

5 Conclusion

In this paper, we proposed a cost minimization offloading mechanism for MDs together with a price scheme for server. Firstly, we established the cost model of MDs. Then, the cost minimization partial offloading scheme was studied. After that, we formulated MEC server's revenue maximization problem and proposed the optimal pricing algorithm. Numerical results verified our scheme as compared to two benchmark schemes. In the future work, we will pay attention to the mobility of MDs.

Acknowledgements. This work is financially supported by the National Natural Science Foundation of China (NSFC) (Grant No. 61671072).

References

1. Huang, J., Xing, C., Wang, C.: Simultaneous wireless information and power transfer: technologies, applications, and research challenges. IEEE Commun. Mag. **55**(11), 26–32 (2017). https://doi.org/10.1109/MCOM.2017.1600806

2. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017). https://doi.org/10.1109/COMST.2017.2745201
3. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017). https://doi.org/10.1109/MC.2017.9
4. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. **19**(3), 1628–1656 (2017). https://doi.org/10.1109/COMST.2017.2682318
5. Zhang, K., et al.: Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access **4**, 5896–5907 (2016). https://doi.org/10.1109/ACCESS.2016.2597169
6. Kim, Y., Kwak, J., Chong, S.: Dual-side optimization for cost-delay tradeoff in mobile edge computing. IEEE Trans. Veh. Technol. **67**(2), 1765–1781 (2018). https://doi.org/10.1109/TVT.2017.2762423
7. Shang, B., Zhao, L., Chen, K., Chu, X.: An economic aspect of device-to-device assisted offloading in cellular networks. IEEE Trans. Wirel. Commun. **17**(4), 2289–2304 (2018). https://doi.org/10.1109/TWC.2018.2791518
8. Zhang, J., Xia, W., Yan, F., Shen, L.: Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. IEEE Access **6**, 19324–19337 (2018). https://doi.org/10.1109/ACCESS.2018.2819690
9. Ning, Z., Dong, P., Kong, X., Xia, F.: A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. IEEE Internet Things J. **6**(3), 4804–4814 (2019). https://doi.org/10.1109/JIOT.2018.2868616
10. Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J.: Mobile-edge computing: partial computation offloading using dynamic voltage scaling. IEEE Trans. Wirel. Commun. **64**(10), 4268–4282 (2016). https://doi.org/10.1109/TCOMM.2016.2599530
11. Zhang, W., Wen, Y., Guan, K., Kilper, D., Luo, H., Wu, D.O.: Energy-optimal mobile cloud computing under stochastic wireless channel. IEEE Trans. Wirel. Commun. **12**(9), 4569–4581 (2013). https://doi.org/10.1109/TWC.2013.072513.121842
12. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Netw. **24**(5), 2795–2808 (2016). https://doi.org/10.1109/TNET.2015.2487344
13. Chen, M., Hao, Y.: Task offloading for mobile edge computing in software defined ultra-dense network. IEEE J. Sel. Areas Commun. **36**(3), 587–597 (2018). https://doi.org/10.1109/JSAC.2018.2815360