Cache-Enhanced Task Offloading for eIoT with Mobile Edge Computing

Zhifeng Li[1], Jie Bai[1], Haonan Zhang[4], Wei Bai[2,3(✉)], Yongmin Cao[1], Liwen Wu[1], Jianying Dong[1], and Yanshan Deng[1]

[1] State Grid Jibei Tangshan Power Supply Company, Tangshan 063000, China
[2] Electric Power Intelligent Sensing Technology and Application State Grid Corporation Joint Laboratory, Beijing, China
baiwei@geiri.sgcc.com.cn
[3] Global Energy Interconnection Research Institute Co., Ltd., Beijing 102209, China
[4] Beijing University of Posts and Telecommunications, Beijing, China

Abstract. With the continuous development and improvement of 5G networks, many emerging technology architectures have been introduced to support 5G service requirements. As one of them, mobile edge computing can meet the exponentially increasing computing requirements, and with its advantages of being more efficient, smarter, and more flexible, it can be well adapted to smart grid scenarios. However, most of the existing research contents of the eIoT focus on the research of computing offloading and content caching separately, ignoring the problem of reusability of some computing results. This paper considers the certain content caching capabilities of the MEC system itself, and aims to design a cache-enhanced MEC eIoT. The model includes offloading, calculating and backhaul for uncached task and downloading of cached content. On the other hand, the problem of task diversity and inspection robot mobility is fully analyzed. Subsequently, we studied the impact of caching capabilities on computing power to get the best MEC server parameter information. Based on the above research, this paper proposes a cache enhanced offload strategy and a collaborative scheduling algorithm to optimize the total delay of all tasks of the inspection robot in the eIoT. Simulation results show that the strategy can effectively reduce the computational offloading latency.

Keywords: eIoT · Collaborative caching · MEC · Computing offloading

1 Introduction

eIoT technology has developed rapidly in recent years, and a variety of inspection robot applications have emerged. This trend is expected to continue in the coming years [1]. With the advent of the Internet of Things (IoT) era, intelligent inspection robot terminals must become an indispensable part of it. Communication technologies that provide lower latency and more reliable services are an effective support for eIoT and play a

S. W. Loke et al. (Eds.): MONAMI 2020, LNICST 338, pp. 67–78, 2020.
https://doi.org/10.1007/978-3-030-64002-6_5

significant role in facilitating the commercialization of eIoT. In order to meet the computing needs of emerging applications, the introduction of MEC that provide auxiliary computing is a widely accepted model of current research institutions [2]. However, the computing power, coverage, and energy costs of current MEC servers are still being continually optimized, posing a challenge for MEC-enabled eIoT. Therefore, we propose a cache-enhanced MEC computing offloading strategy to address the challenges in the eIoT [3].

MEC is characterized by location awareness, delay sensitivity and mobile support [4]. The design of MEC needs to comprehensively consider the joint optimization of the location, communication quality and computing resources between distributed mobile devices and MEC servers [5, 6]. A variety of task processing performance can be improved through a federated design with the caching [7]. Recent research has made a lot of efforts and achievements in achieving a more reasonable eIoT structure. For instance, in [8], the author designed an integrated model for computing offloading, content caching, and resource allocation, using MEC's computing power and storage resources to reduce task offloading latency. In [4], the author proposes a distributed content distribution network based on MECs, and analyzes the distribution characteristics of service devices and terminals in the system to develop a grouping-based and hierarchical caching strategy for higher energy efficiency. The author of [9] considered the problem of user mobility, and extended the edge cache to MEC to implement more flexible context-aware cache decision, which proved that the scheme has higher throughput than the traditional scheme. The work in [10] proposed a new scheme where mobile edge computing resources are utilized for enhancing edge caching capability in 5G network, and introduced a concept of eIoT cache cloud to further utilize the cache resources of smart inspection robot to improve cache utilization. A novel theoretical framework that weighs the computational, caching, and communication resources of mobile edge networks reveals the fundamental benefits of computing and caching in content delivery eIoT in [3].

At present, for popular AR and multimedia transformation applications, there is a problem of repeated calculation of computing tasks [10]. Similarly, in the scenario of a service eIoT, there are always numerous identical computing tasks on the same short-distance road segment, and the same task key value is offloaded to the MEC server by different inspection robot. After a complex operation, the same result is fed back to the task requester. When the resource requirements of these high-frequency identical tasks are superimposed, it becomes a cost that cannot be ignored [7]. In this paper, we describe the MEC-assisted task calculation and unloading process in the eIoT. Using the storage capacity of the MEC server to cache the high hit rate calculation results can reduce the resource consumption of the repetitive operation. In order to avoid the high price brought by strengthening the server configuration, the interaction between cache capacity and computing task volume is studied. The cache-enhanced computing offload strategy and collaborative scheduling algorithm is implemented in the eIoT proposed in this paper, which reduces the computational latency of multi-tasking and provides a better experience for intelligent inspection robot.

The rest of the paper is organized as follows, Sect. 2 discusses in details our system model. Section 3 presents our cache-enhanced offloading strategy and collaborative scheduling algorithm, while Sect. 4 presents our performance evaluation. Then we conclude the paper and in Sect. 5.

2 System Model

In this section, we will model the proposed network. A cache-enhanced MEC system that provides services to inspection robot terminals, as shown in Fig. 1.

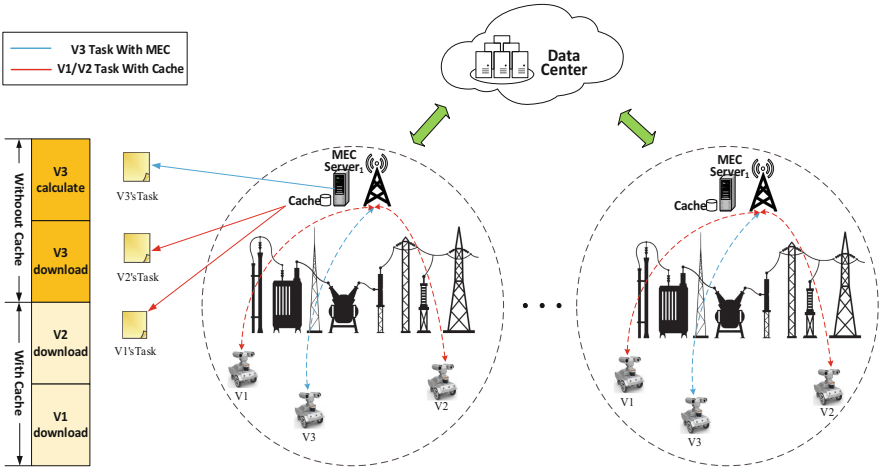

Fig. 1. System model of cache-enhanced vehicular networks

2.1 Transmission Model

Most of the existing research on MEC ignore the backhaul latency in returning calculation results. However, as the number of tasks increases, a series of task types with large amount of backhaul data (such as AR/VR, video transcoding, etc.) appear. Therefore, our transmission model considers not only the upload delay but also the delay of return result [11]. Thanks to the further promotion of 5G communication technology, it can also better support the information transmission of the eIoT.

In the transmission model, a mode in which a base station and a MEC server are deployed in cooperation is used, and the combination is recorded as Access Point (AP). Furthermore, based on the existing 5G version specification, the inspection robot can communicate with AP [1].

During the one-way road patrol, the distance between the robot and AP can be obtained according to the following formula

$$d_{on}(t) = \sqrt{r^2 + \left(\frac{L}{2} - vt_{on}\right)^2} \tag{1}$$

where r is the horizontal distance between the road and the base station inspected by the inspection robot, L is the coverage distance of the AP and the speed of the robot is v. t_{on} is the duration of the robot within the current AP coverage, which can be obtained by

$$t_{on} = \frac{L}{v} \tag{2}$$

In addition, during the change of d_{on}, the data rate R_{V2I} of the task offloading and the backhaul calculation result can be derived as

$$R_{V2I}(t) = B_{V2I} \log_2\left(1 + \frac{P_t d_{on}^{-\delta}(t) h^2}{N_0}\right) \tag{3}$$

Let B_{V2I} and δ denote the channel bandwidth and path loss exponent respectively. The transmission model takes both the uplink and the downlink, so the transmission power P_t and the channel fading coefficient h should be considered in two cases [9].

High-speed mobility is an important feature in eIoT, which causes d_{on} changing faster. Therefore, the average transmission rate $\overline{R_{V2I}}$ is used to measure the uplink and downlink data rate of the inspection robot terminal and it can be given by

$$\overline{R_{V2I}} = \frac{\int_0^{t_{on}} R_{V2I}(t)dt}{t_{on}} \tag{4}$$

2.2 Computation Model

Typically, a task can be modeled as a profile with three parameters including d_c, d_{up}, $t\max$. The three parameters represent the task input-data size, computation intensity, and maximum allowable delay respectively. d_c and d_{up} are linear as similar to much existing research,

$$d_c = \alpha d_{up} \tag{5}$$

The task complexity index α is usually greater than 1. When some latency-sensitive or computation-intensive tasks are generated, the requester may not finish these tasks on time due to limited processing capacity of the on-board unit, i.e.,

$$t_{local} < t\max \tag{6}$$

the local calculation latency t_{local} exceeds maximum tolerance latency [9]. Thus, we introduced a MEC server system equipped with a multi-core CPU that can provide auxiliary calculations for multiple inspection robot terminals simultaneously. Since the computing resources of the MEC are usually subject to payment, the inspection robot terminal also has to obtain a specified CPU cycles by paying an unequal fee.

2.3 Caching Model

The MEC server has a certain storage capacity as a computing device deployed to the edge of the network. Moreover, as part of the AP, the MEC server can be directly connected to a dedicated caching device [13]. Based on its cache capability, the AP stores some task results. When the requester hits the content stored in the cache, the calculation result can be directly obtained. Let c denote the cache action [4, 13].

$$c_n \in \{0, 1\}, n \in \mathcal{N} \tag{7}$$

here, $c_n = 1$ indicates the action of the task n hitting the cached content, and $c_n = 0$ is the opposite. Under the limited cache capacity,

$$\mathrm{C} \leq \sum_{n \in \mathcal{N}} c_n d_{n,dl} \tag{8}$$

C and $d_{n,dl}$ represent the total caching amount at the current AP and the calculation result data amount of task n, respectively.

3 Cache-Enhanced Offloading Scheme and Collaborative Scheduling Algorithm

Based on the model discussed in Sect. 2, this section designs the cache-enhanced computational offloading scheme. The scheme calls for increased utilization of MEC. In addition, caching reusable results can increase the scalability of the system and reduce resource consumption, latency, and cost. Then we propose a collaborative scheduling algorithm to guide the task requester to do joint computing offloading.

3.1 Cache-Enhanced Offloading Scheme

First, the cache-enhanced offloading scheme is applicable to scenarios where multi-robot terminals carry multitasking. When robot terminal i meets the conditions described in Eq. (7), we record i as a task requester, $i \in \mathbb{SV}$ $\mathbb{SV} = \{1, \ldots, i, \ldots, SV\}$. The requester needs the MEC-assisted computing task $n \in \mathcal{N}, \mathcal{N} \triangleq \{1, 2, \ldots N\}$. In order to describe the key information of the task n, we denote $T_n = \left\{d_{n,up}, d_{n,dl}, c_n, t_{i,max}\right\}$ [5]. Indeed, a requester may carry multiple different tasks. The set and number of vehicles who need to execute task n at the random system task state X,

$$SV_n(X) \triangleq \sum_{i \in \mathbb{SV}} I[X_i = n] \tag{9}$$

where $I[\cdot]$ denotes the indicator function. Next, the requester i starts the operation of offloading the task n.

$$\overline{R_{i,n,up}} = \frac{\int_0^{t_{on}} R_{i,n,up}(t)dt}{t_{on}} \tag{10}$$

$$t_{i,n,up} = \frac{d_{i,n,up}}{\overline{R_{i,n,up}}} \tag{11}$$

$R_{i,n,up}(t)$ represents the instantaneous transmission rate of the n-type task offloading process on the terminal i. The transmission power of the robot terminal is P_t and the offloading average rate $\overline{R_{i,n,up}}$ can be derived according to (4)–(5). The latency of this process is recorded as $t_{i,n,up}$. Then, the AP determines whether the result has been cached according to the relevant key values of the received task. If $c_n = 0$, the MEC server provides the operating frequency $f_{m,i,n}$ to the requester for calculation. After the time $t_{i,n,ex}$, the result is transmitted back to the requester through the AP [12]. Under the influence of α, the operation delay $t_{i,n,ex}$ and the backhaul delay $t_{i,n,dl}$ can be presented as

$$t_{i,n,ex} = \frac{\alpha d_{i,n}}{f_{m,i,n}} \tag{12}$$

$$t_{i,n,dl} = \frac{d_{i,n,dl}}{\overline{R_{i,n,dl}}} \tag{13}$$

On the other hand, when $c_n = 1$, the MEC server does not perform repetitive computing on the current task, which is beneficial to provide more idle resources for uncached tasks [3]. In summary, the total latency of computing offloading can be expressed as

$$t_{i,n} = (1 - c_n)t_{i,n,up} + t_{i,n,dl} + t_{i,n,ex} \tag{14}$$

In this paper, how to get task popularity and cache high hit rate content is not our focus since caching is a technology to enhance MEC computing offloading. This scheme ensures that the MEC server has sufficient computing power, and reduces the delay for part of robot terminals. When considering the task requester set $\mathbb{SV}$, the problem of optimizing the total latency can be formulated as

$$\mathbf{P1}: \min_{C} \sum_{i\in\mathbb{SV}} \sum_{n\in\mathcal{N}} t_{i,n}$$
$$\text{s.t.} \begin{cases} C1: \mathrm{C} \le \sum\limits_{n\in\mathcal{N}} c_n d_{n,dl} \\ C2: t_{i,n,local} < t_{i,n,max} \\ C3: \sum\limits_{n\in\mathcal{N}} \left((1-c_n)t_{i,n,up} + t_{i,n,dl} + t_{i,n,ex}\right) \le t_{i,n,max}, \\ \forall i \in \mathbb{SV} \end{cases} \tag{15}$$

Specifically, $C1$ indicates that the cached result cannot contain all task types, which is determined by the actual conditions. $C2$ is a prerequisite for task offloading. If the cache-enhanced scheme still could not guarantee that the task is completed within the maximum tolerance delay, other calculation methods need to be considered, and $C3$ ensures that the situation does not occur. By solving the problem **P1**, we optimize the total latency.

3.2 Collaborative Scheduling Algorithm

Task offloading should take into account the impact of pricing strategies. The number of computing resources provided by the MEC server for the robot terminal is positively correlated with the pricing [10]. As the price of the payment increases, it will inevitably gain a stronger computing power and a lower processing delay. According to the different needs of users, we can set the peak amount of payment for it. In other words, when the computing resources are in short supply, the unit resource price of the MEC server rises, and the resources required by some users to complete the current task reach the upper limit of their ability to pay. Such users choose to sacrifice the calculation delay to reduce the tariff or use other auxiliary. Way (such as D2D calculation). Such users choose to sacrifice the calculation latency to reduce the tariff or use other auxiliary methods (Such as D2D scheme).

Our collaborative scheduling algorithm can jointly consider the utility value and the delay tolerance parameter to improve the availability of the cache enhanced offloading strategy. Specifically, the utility value U is used to describe the relationship between resources and costs can be given by

$$U_{i,n} = \varphi\left(f_{m,i,n}\right) - \omega(P_n, f_{m.i.n}) \tag{16}$$

where $U_{i,n}$ is a utility function [17]. For task n, the MEC server provides the corresponding $f_{m,i,n}$ computation capacity. Task requester always expect a higher QoE, while $f_{m,i,n}$ is directly related to it.

$$\begin{aligned}\varphi\left(f_{m,i,n}\right) &= \beta\left(t_{i,n,local} - t_{i,n,up} - t_{i,n,dl} - t_{i,n,ex}\right) \\ &+ \gamma\left(ln\left(\rho f_{m,i,n} + \in\right)\right) \\ &= \beta\left(\frac{d_{i,n}}{f_i} - \frac{d_{i,n,up} + d_{i,n,dl}}{\overline{R_{V2I}}} - \frac{d_{i,n}}{f_{m,i,n}}\right) \\ &+ \gamma\left(ln\left(\rho f_{m,i,n} + \in\right)\right)\end{aligned} \tag{17}$$

In (17), we use the processing latency to describe the total utility value $\varphi\left(f_{m,i,n}\right)$. ρ is a utility coefficient that saves unit delay, and γ is a utility coefficient that saves computational resources. $\omega(P_n, f_{m.i.n})$ represents the payment for MEC server, where the parameter P_n is the price of the unit resource.

Obviously, the utility function of terminal i is mainly related to computation capacity allocated by RSU and $f_{m,i,n}$ is the main influencing factor [10]. According to the configuration of the MEC server, the maximum resource is F_m, so that $f_{m,i,n} \leq F_m$. Bringing (16) into (17), we derive the maximum value of the utility function. The second-order derivative of $U_{i,n}$ with respect to $f_{m,i,n}$ is

$$\frac{\partial^2 U_{i,n}}{\partial^2 f_{m.i.n}} = -2\beta\frac{1}{f_{m.i.n}^3} \tag{18}$$

Because all the values of parameters in (17) are positive, the second-order derivative of the utility function is lower than zero, namely, $\frac{\partial^2 U_{i,n}}{\partial^2 f_{m.i.n}} < 0$ [10]. It shows that the utility function has a maximum value. And the optimal solution $f^*_{m,i,n}$ can be solved by

$\frac{\partial U_{i,n}}{\partial f_{m.i.n}} = 0$. The requester i has an ideal effect value $U^*_{i,n}$ in the unloading calculation. When $U^*_{i,n} > U_{i,n}$, we have to consider other auxiliary calculation methods. The specific algorithm design is as follows. The detail steps about the proposed collaborative scheduling algorithm is described as Algorithm 1.

Table 1. Joint selection algorithm process

Algorithm 1 Joint Selection Algorithm

1: Initialization:
   a)The request robots $\mathbb{SV} = \{1, \ldots, i, \ldots, SV\}$
   b)Set caching action $c_n = 0, \mathrm{n} \in \mathcal{N}$
   c)The P_n
2: **for** i in **SV** **do**
3: **for** n in N **do**
4: Scan $d_{i,n,up}$
5: **if** $d_{i,n,up}$ has cached, **then**
6: Set $c_n = 1$;
7: **end if**
8: Calculate $U_{i,n}$ by (14)-(17)
9: **if** $U_{i,n} > U^*_i$ and $t_{i,n,local}$ satisfy constrain (6), **then**
10: Calculate $t_{i,n}$ by (14)
11: **else if**
12: Set $f_{m,i,n} = f^*_{m,i,n}$
13: Calculate $t_{i,n}$
14: **if** $t_{i,n} > t_{i.n.local}$, **then**
15: **Set** $t_{i,n} = t_{i.n.local}$
16: **end if**
17: **end if**
18: **end for**
19: **end for**
20: Output: Offloading strategy and the optimal total latency

4 Simulation Result and Discussion

In this section, the performance of the cache enhanced offloading scheme and the collaborative scheduling algorithm are illustrated by comparing the simulation results. In this simulated scenario, each AP cover a range of 200 m [16]. We set transmission power P_t of the vehicle and the AP is 1.3 W and 2 W, respectively, and data is transmitted through the wireless bandwidth of 10 M [16]. The power spectral density of additive white Gaussian noise $N_0 = -100$ dBm, and the channel fading coefficient $h = 4$ [12]. The computational complexity index $\alpha = 1.05$, $\beta = 0.6$, and $\gamma = 0.4$. Since the price P_n of the unit resource is fluctuating, we set the initial value of P_n to 0.5 according to the number of idle resources. With resources become rarer, the P_n is larger.

Compared with the commonly used MEC offloading strategy and partial offloading strategy, the simulation results show that the scheme designed in this paper obtain better performance in latency and scalability for vehicle network. At the beginning, we analyze the situation where an AP covers 40 robot terminals, and each terminal randomly generates computing tasks which size is $5 \times 10^6 bits$ [13]. We simulate the velocity of each robot randomly selected from 10 m/s to 30 m/s. By adjusting the cache capacity, it is possible to store different numbers of high hit rate calculation results, which affects the amount of resources of the required MEC. Figure 2 shows the relationship between the amount of buffer and the required resources. When the buffer reaches $4 \times 10^7 bits$, the slope of the curve is significantly reduced. At this point, the MEC needs to provide $4 \times 10^{10} cycles/s$ processing power for uncached computing tasks.

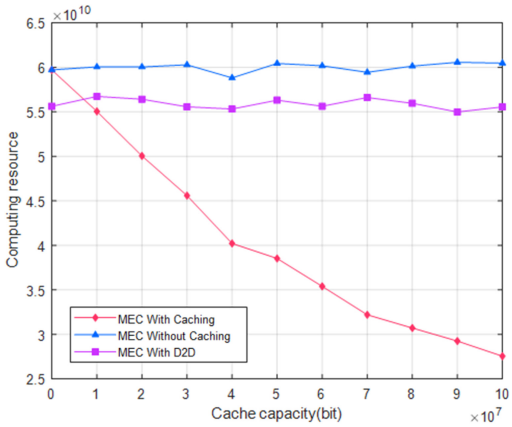

Fig. 2. The relationship between the cache capacity and the computing resource.

Figure 3 depicts the total latency of cache enhanced offloading and popular MEC offloading scheme. Similar to the results shown in Fig. 2, when the cache reaches $4 \times 10^7 bits$, the latency optimization result is quite obvious. Therefore, $C = 4 \times 10^7 bits$ is determined for subsequent research.

Figure 4 and Fig. 5 discuss optimization in multitasking situations. In Fig. 4, it describes the total delay of the three modes in the interval of task data in [0, 45M]. Due to the limited computing power of the onboard unit, the D2D mode performance is lower than other modes. With the exhaustion of MEC capabilities, the scheme of offloading some tasks to nearby idle vehicle processing yields lower latency than the scheme that only supports MEC offloading. However, the strategies and algorithms proposed in this paper can save a lot of repetitive task calculation overhead for MEC servers, so the overall performance has always been better than other strategies. Figure 5 describes the total delay in the processing of the task when the number of robot terminals is increasing.

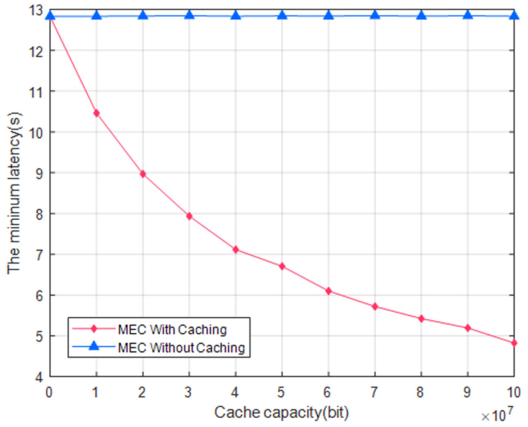

Fig. 3. The minimum latency for computing offloading of MEC with/without caching.

The simulation results are shown in Fig. 4. However, the model proposed by us can save a lot of repetitive calculation overhead for MEC servers, so the overall performance has always been better than other strategies. Figure 5 describes the total latency when the number of robot terminals continues to increase. The results obtained from this simulation are consistent with Fig. 4.

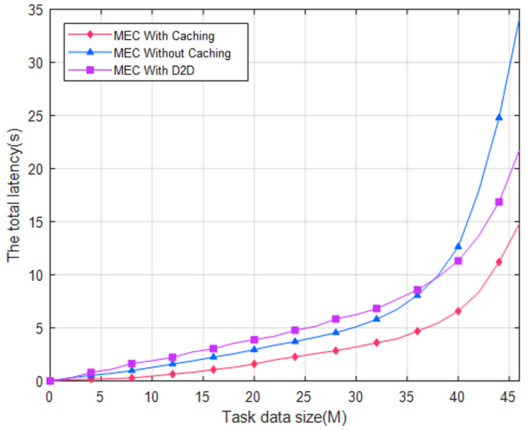

Fig. 4. The total latency of cache-enhanced offloading scheme, without caching and D2D versus the task data size.

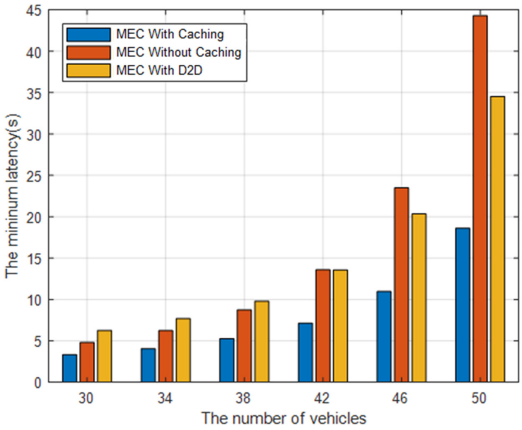

Fig. 5. The total latency of cache-enhanced offloading scheme, without caching and D2D versus the number of robots.

5 Conclusion

This paper focuses on the application of popular MEC technology in smart gird. A cache-enhanced computing task offloading scheme is proposed, which uses cache repetitive task calculation results to optimize system resources and processing latency. At the same time, a collaborative scheduling algorithm is designed to help smart inspection robots to perform optimal calculation and offloading within the limits of their respective payment capabilities. Simulation results verify the effectiveness of our proposal.

Acknowledgement. This paper is supported by Science and Technology Project of State Grid Jibei Electric Power Co., Ltd.; Research and Application of Intelligent Operation Inspection Service Based on Electric Power Wireless Private Network (5201031801CS).

References

1. Shah, V.S.: Deterministic consumer applications enabled policy Architecture of Mobile Edge Computing ecosystem. In: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–7. IEEE (2017)
2. Ren, D., Gui, X., Lu, W., An, J., Dai, H., Liang, X.: GHCC: grouping-based and hierarchical collaborative caching for mobile edge computing. In: 2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), Shanghai, pp. 1–6 (2018)
3. Wang, X., Han, Y., Wang, C., Zhao, Q., Chen, X., Chen, M.: In-Edge AI: intelligentizing mobile edge computing, caching and communication by federated learning. IEEE Netw. **33**(5), 156–165 (2019)
4. Chen, L., Dong, X., Kuang, X., et al.: Towards ubiquitous power distribution communication: multi-service access and QoS guarantees for IoT applications in smart grid. In: 2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia), pp 894–898. IEEE (2019)

5. Zhang, J., et al.: Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching. IEEE Internet Things J. **6**(3), 4283–4294 (2019)
6. Liu, X., Zhang, J., Zhang, X., Wang, W.: Mobility-aware coded probabilistic caching scheme for MEC-enabled small cell networks. IEEE Access **5**, 17824–17833 (2017)
7. Zhang, K., Leng, S., He, Y., Maharjan, S., Zhang, Y.: Cooperative content caching in 5G networks with mobile edge computing. IEEE Wireless Commun. **25**(3), 80–87 (2018)
8. Zaw, C.W., Ei, N.N., Reum Im, H.Y., Tun, Y.K., Hong, C.S.: Cost and latency tradeoff in mobile edge computing: a distributed game approach. In: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), Kyoto, Japan, pp. 1–7 (2019)
9. Vu, T.X., Lei, L., Chatzinotas, S., Ottersten, B., Trinh, A.V.: On the successful delivery probability of full-duplex-enabled mobile edge caching. IEEE Commun. Lett. **23**(6), 1016–1020 (2019)
10. Li, M., Rui, L.L., Qiu, X., et al.: Design of a service caching and task offloading mechanism in smart grid edge network. In: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 249–254. IEEE (2019)
11. Xu, X., Liu, J., Tao, X.: Mobile edge computing enhanced adaptive bitrate video delivery with joint cache and radio resource allocation. IEEE Access **5**, 16406–16415 (2017)
12. Wang, K., Li, J., Wu, J., et al.: QoS-predicted energy efficient routing for information-centric smart grid: a network calculus approach. IEEE Access **6**, 52867–52876 (2018)
13. Wang, D., Lan, Y., Zhao, T., Yin, Z., Wang, X.: On the design of computation offloading in cache-aided D2D multicast networks. IEEE Access **6**, 63426–63441 (2018)
14. Qian, Y., Li, S., Shi, L., et al.: Cache-enabled MIMO power line communications with precoding design in smart grid. IEEE Trans. Green Commun. Netw. **4**(1), 315–325 (2019)
15. Li, Y., Li, C., Wu, G., et al.: Research on high-precision time distribution mechanism of multi-source power grid based on MEC. In: 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 1–5. IEEE (2019)
16. Arun, M., Jenson, D., Nandhini, V.M., et al.: Smart grid robot exclusively designed for high power transmission lines. In: 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), pp. 652–654. IEEE (2019)
17. Disyadej, T., Promjan, J., Poochinapan, K., et al.: High voltage power line maintenance & inspection by using smart robotics. In: 2019 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), pp. 1–4. IEEE (2019)