IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 7, 2017, accepted January 13, 2018, date of publication March 1, 2018, date of current version March 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2805798

Energy Efficient Task Caching and Offloading for Mobile Edge Computing

YIXUE HAO[1], MIN CHEN[1,2], (Senior Member), LONG HU[1], M. SHAMIM HOSSAIN[3], (Senior Member), AND AHMED GHONEIM[4]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan 430074, China
[3]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[4]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia, and also with the Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Menoufia 32721, Egypt

Corresponding authors: Long Hu (hulong@hust.edu.cn) and M. Shamim Hossain (mshossain@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research, King Saud University, Riyadh, Saudi Arabia, through the research group under Project RGP 229.

ABSTRACT While augment reality applications are becoming popular, more and more data-hungry and computation-intensive tasks are delay-sensitive. Mobile edge computing is expected to an effective solution to meet the low latency demand. In contrast to previous work on mobile edge computing, which mainly focus on computation offloading, this paper introduces a new concept of task caching. Task caching refers to the caching of completed task application and their related data in edge cloud. Then, we investigate the problem of joint optimization of task caching and offloading on edge cloud with the computing and storage resource constraint. We formulate this problem as mixed integer programming which is hard to solve. To solve the problem, we propose efficient algorithm, called task caching and offloading (TCO), based on alternating iterative algorithm. Finally, the simulation experimental results show that our proposed TCO algorithm outperforms others in terms of less energy cost.

INDEX TERMS Caching, computation offloading, mobile edge computing, energy efficient.

I. INTRODUCTION

Nowadays, with the rapid development of wireless technology and Internet of things, more and more mobile devices, such as smart phones, wearable devices, have different wireless network access requirements for bandwidth and computation. In the future, mobile devices will become more intelligent, and applications deployed on them will require extensive computing power and persistent data access [1]. However, the development of these new applications and services is limited by the finite computing power and battery life of these devices. If data-hungry and computing-intensive tasks can be offloaded to the cloud to execute, it can overcome the shortcomings of the lack of computing power of the mobile devices. However, when mobile devices are connected to the cloud through a wireless network, a relatively long delay occurs, which is not suitable for delay-sensitive tasks [2].

In recent years, mobile edge computing provides users with short delay and high performance computing services by deploying computing nodes or servers at the edge of the network, to meet users' requirements for delay-sensitive tasks [3], [4]. There are two main advantages of using the edge cloud: (i) compared with local computing [5], mobile edge computing can overcome the limited computing power of mobile devices; (ii) in contrast to the remote cloud computing [6], although the edge cloud has a small geographical scope and limited resource capacity, it can avoid large delay may be caused by offloading the task content to the remote cloud. Therefore, mobile edge computing exhibits a better tradeoff between delay-sensitivity and computing-intensive tasks.

To the best of our knowledge, the previous work on mobile edge computing is mainly focused on the following two aspects.

- Content offloading. This is also known as edge caching. In edge caching, content provider can cache popular content on edge cloud to reduce the delay and energy consumption of user requesting content [7], [8]. For this problem, the researchers have proposed various caching strategies, considering e.g., content distribution [9], user mobility [10].
- Task offloading. Its main design issues are where, when, what, how to offload user's tasks from the mobile device to the edge cloud, to reduce computing delay

and save energy. Various works have studied different task offloading policy, considering e.g., multi-user [11], multi-server [12].

Unfortunately, the main concern of content offloading is the storage capacity of edge cloud, they do not consider the computing and storage capabilities of the edge clouds at the same time. Furthermore, for the task offloading, in the above work, it is assumed that the edge cloud has enough hardware and software resources to support computing tasks. In fact, this assumption is impractical, because edge cloud computing power is limited, which cannot support all of the computing tasks.

Thus, in this paper, we consider more practical and energy efficient mobile edge computing. We first introduce the new concept of task caching. The task caching refers to caching task application and their related data. Since the task need both storage and computation, thus, the task caching consider both computing and storage constraint of edge cloud. Then, we propose the problem of joint optimization of task caching and offloading on edge cloud. Our goal is to achieve energy efficient of mobile device while meet the user's demand for delay by designing optimal task caching and offloading scheme. However, the scheme faces the following three challenges:

- **Energy efficient task caching problem:** Considering the heterogeneity of task such as task demand among users, the data size and the required computation capacity of task (e.g., the augment reality services and online natural language processing have different storage and CPU requirements) and the computing and storage constraints of edge cloud, how to give a energy efficient task caching strategy is challenging problem.
- **Energy efficient task offloading problem:** In the execution of a task, computing task can be processed on edge cloud or locally. However, when the task is cached in the edge cloud, it may not need to be processed locally. Therefore, it is challenging to make a energy efficient task offloading decision with task caching.
- **How to solve the joint optimization problem:** When computing and storage resources at edge cloud is limited, how to give task caching and offloading problem and solve this problem is challenging issue. This is because energy efficient task caching and offloading scheme needs careful coordination.

In this paper, we study the less investigated problem of task caching for mobile edge computing, and propose a joint optimization of task caching and task offloading (TCO) scheme in order to minimize the total energy cost by the mobiles device. In summary, the main contributions of this paper include:

- We introduces a new concept of task caching. Task caching refers to the caching of completed task application and their related data in edge cloud. In terms of deployment scheme of task caching, the experiment shows that task caching relates to the task popularity and size of contents, as well as the required computation capacity of tasks.
- We propose the joint task caching and offloading problem to minimize energy cost of mobile device while meet user's delay requirement, which included the task caching decision (i.e., decide which task should be cached) and task offloading scheduling (i.e., decide how much task to offload) optimization problem. We formulate this problem as mixed integer nonlinear optimization problem which is hard to solve.
- In order to solve this problem, we design an efficient scheme, called TCO (task caching and offloading) to solve the joint optimization problem. The scheme includes two sub-problem: a convex sub-problem (i.e., task offloading problem) and 0-1 programming (i.e., task caching problem). The simulation experiment shows that energy cost of mobile device can be decreased significantly by deploying the task caching and offloading strategy reasonably.

The paper is organized as follows. The system model which include the scenario description, communication model, computation model, task caching model and problem formulation in Section II. The problem solution is given in Section III. Section IV provide the simulation results. Finally, Section V concludes the paper.

II. SYSTEM MODEL

In this section, we will propose the system model in order to minimize the total energy consumed by the mobile device while meet the user's delay requirement. The system model as shown in Fig. 1, we use virtual reality (VR) as a typical application scenario.

The mobile device has five virtual reality application task (e.g., rendering scenes, recognizing and tracking objects) need to be processed, namely T1, T2, T3, T4 and T5. Each task has different number of requests (i.e., task popularity), data size and computation capacity requirement. For example, the scene rendering task have high popularity, the task of tracking objects has larger data size and needs lots of data transmissions and the task of recognizing objects needs higher computing resource. Though task caching and offloading can reduce the energy consumption of mobile device while meet the user's demand for delay, how to design the optimal task caching and offloading strategy is still a challenging problem when considering the heterogeneity of the task and the limited resources of the edge cloud. Thus, in this paper, we will deal with the following two problems:

- Which tasks to cache: it refers to the decision whether to cache the computing tasks on edge cloud or not.
- How many tasks to offload: it refers to the decision how much task should be processed locally and how much tasks should be processed on edge cloud.

In the following, we introduce the system model in detail.

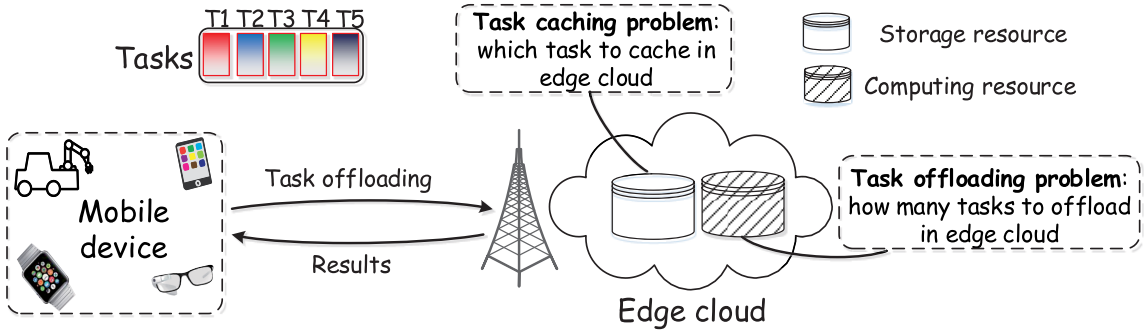

FIGURE 1. Illustration of task caching and offloading for mobile edge computing.

A. SCENARIO DESCRIPTION

We consider a edge computing ecosystem which includes multiple mobile devices and an edge cloud. Let assume the edge computing system consists of N mobile device, K computation task. we denote the set of device and task by $\mathcal{N} = \{1, 2, \cdots, N\}, \mathcal{K} = \{1, 2, \cdots, K\}$, respectively. In this paper, we assume that the number of users is more than the number of task ($N \geq K$), this is because some computing tasks have higher popularity (e.g., scenes rendering task in VR). Thus, those tasks would be repeatedly requested and processed for many times. Thus, we assume that any mobile device only request task once while different users can request a same task based on their preferences. We also define $u_{n,k}$ as user n requests task k. In addition, mobile devices can communicate with edge cloud through wireless channel. Edge cloud is a small data center with computing and storage resources, whereas computing resources provides task processing for mobile devices and storage resource provides for task content and processing code.

For heterogeneous computing tasks, we adopt three parameter model the computation task. For computing task $u_{n,k}$, we define $u_{n,k} = \{\omega_k, s_k, D_n\}$, where ω_k (in CPU cycles per bit) is the amount of computing resource required for the task $u_{n,k}$, i.e., the total number of CPU cycles needed to complete the task, and s_k (in bits) is the data size of computation task $u_{n,k}$, i.e., the amount of data content (e.g., the processing code and parameter(s)) to be delivered toward edge cloud. Finally, D_n represents the completion deadline for user n. For instance, in the video decoding case, ω_k is the computing resource needed for video decoding, s_k is the video data size, and D_n is the completion deadline. Furthermore, since the computing and storage of edge cloud is limited, we assume that the cache size and computing capacity of edge cloud is c_e and c_s, respectively.

B. COMMUNICATION MODEL

We introduce the communication model and give the uplink data rate when mobile device offloads task on edge cloud. Let H_n as the channel gain between the user n and edge cloud. It is assumed that the user does not move much during task offloading. Thus, we consider H_n is a constant. Denote P_n as the transmission power of mobile device of user n. Then, the uplink data rate of user n can be obtained as follows:

$$r_n = B \log_2 \left(1 + \frac{P_n H_n}{\sigma^2}\right) \tag{1}$$

where σ^2 denotes the noise power, and B represents the channel bandwidth.

In this paper, we do not consider downlink transmission delay and packer loss. This is because data size after task processing is generally smaller than it before processing [11], and downlink rate from edge cloud to mobile device is higher than uplink rate from mobile device to edge cloud.

C. COMPUTATION MODEL

In this subsection, we introduces the computation offloading model. In this paper, we assume that computing task is divisible, which means that task can be divided into two or more parts. Given video streaming analytics (e.g., objective recognition) as an example, a large video file with lots of frames can be divided into multiple video clips through video segmentation. Thus, some of video clips can be process at edge cloud while the others can be handled locally. We will introduce in detail.

1) DELAY AND ENERGY COST OF LOCAL COMPUTING

For local task computing, we define f_n^l as a CPU computing capability of mobile user n. Thus, the local execution time of task $u_{n,k}$ can be expressed as follows:

$$T_{n,k}^l = \frac{\omega_k}{f_n^l} \tag{2}$$

According to work [13], the energy consumption per computing cycle is $\varepsilon = \kappa f^2$, and thus the energy cost of local computation for the computing task can be obtained as follows:

$$E_{n,k}^l = \kappa (f_n^l)^2 \omega_k \tag{3}$$

where κ is the energy coefficient, which depends on the chip architecture. In this paper, according to the work in [14], we set $\kappa = 10^{-25}$.

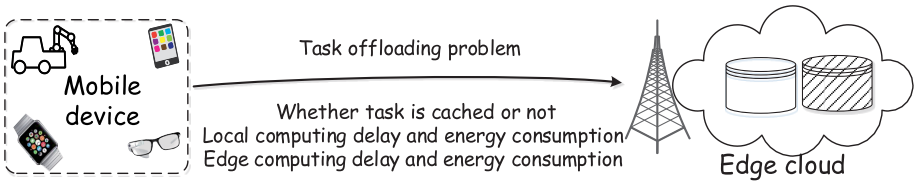

FIGURE 2. Illustration of task offloading problem for mobile edge computing. A user offloads computation task to edge cloud based on whether the task is cached or not, local computing delay and energy consumption, edge computing delay and energy consumption.

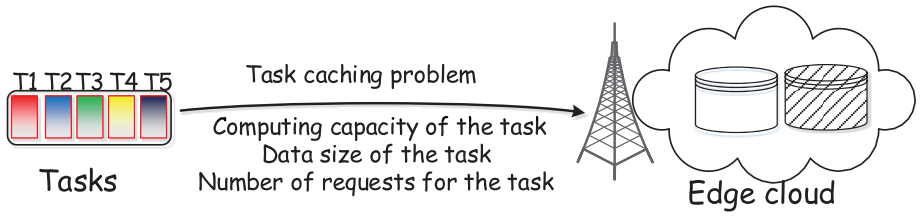

FIGURE 3. Illustration of task caching problem for mobile edge computing. The decision of task caching is based on computing capacity of the task, data size of the task and number of requests for the task.

2) DELAY AND ENERGY COST OF MOBILE EDGE CLOUD COMPUTING

For task computing on edge cloud, we define f_n^c as the computational resource of edge cloud allocated to user n. In this case, the task duration consists of time consumed by two procedures: (i) time consumed when mobile devices offloads the task (i.e., task transmission duration $T_{n,k}^{tra}$), (ii) time consumed when computation task are processed on the edge cloud (i.e., task process duration $T_{n,k}^{pro}$). Therefore, we can obtained the task duration of task $u_{n,k}$ on edge cloud as follows:

$$T_{n,k}^{c} = T_{n,k}^{tra} + T_{n,k}^{pro} = \frac{s_k}{r_n} + \frac{\omega_k}{f_n^c} \tag{4}$$

In this paper, we neglect the time delay for edge cloud sends the task outcome back to the mobile device, this is because the size of task result is much smaller than that of task input data.

If task is offloaded to edge cloud, the energy consumption of mobile device is only calculated by the communication cost for offloading the task content to the edge cloud. Thus, the transmission energy cost of mobile device for sending s_k bits of the task into the edge cloud as shown below:

$$E_{n,k}^{C} = P_n T_{n,k}^{tra} = P_n \frac{s_k}{r_n} \tag{5}$$

D. TASK CACHING MODEL

In this subsection, we will introduce the task caching model. The task caching refers to the caching of completed task application and their related data in edge cloud. On the other word, a relative dependent "resource container" (e.g., virtual machine with associated task data contents) reside in the edge cloud. The process of task caching is as follows. Mobile device first requests the computing task that needs to be offloaded. If the task is caching on the edge cloud, then the edge cloud informs the mobile device that task exists on edge cloud. Thus, mobile device does not have to offload the computing task to edge cloud. Finally, when edge cloud finishes task processing, it transmits the result to mobile device. By this fashion, a user does not need to offload the same task to the edge cloud when it is cached. Thus, the energy cost of mobile device and the delay of task offloading can be reduced by task caching.

However, in the task caching strategy, there are still challenges related to computing task caching: (i) although caching capacity and computing capability of edge cloud are better than those of a mobile device, edge cloud is still not able to cache and support all types of computing tasks; (ii) comparing to content caching, task caching not only needs to consider task popularity, but also needs to consider data size and computing resource required for the task. Therefore, the design of task caching strategy is a challenging issue. In this paper, we give the task duration and energy consumption in edge cloud under circumstance of task caching or not. For task caching, the task duration is simplified into the task processing delay $T_{n,k}^{pro}$. The major energy consumption happens in edge cloud while there is no energy cost for mobile device. For the case without task caching, the delay is $T_{n,k}^{c}$, and energy consumption of mobile device is $E_{n,k}^{c}$.

E. PROBLEM FORMULATION

For task caching problem, we define the integer caching decision variable, $x_k \in \{0, 1\}$ that indicates whether task k is cached at edge cloud ($x_k = 1$) or not ($x_k = 0$). Therefore, the task caching strategy can be represented as follows: $\mathbf{x} = (x_1, x_2, \cdots, x_K)$. For task offloading problem, we define the decision variable $\alpha_n \in [0, 1]$. When $\alpha_n = 1$, the user n task is processed locally; when $\alpha_n = 0$, the user n task is offloaded to edge cloud; when $\alpha_n \in (0, 1)$, the part α_n of the user n task is processed locally, and the part $1 - \alpha_n$ is offloaded to edge cloud. Thus, the task offloading policy can be represented as follows: $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$. Fig. 2 and Fig. 3 illustrate the task offloading and caching problem for mobile edge computing.

According to above discuss, considering task caching, local and mobile edge computing, the total task duration of user n task k, $u_{n,k}$ can be obtained as follows:

$$T_{n,k} = x_k \frac{\omega_k}{f_n^c} + (1 - x_k)\left[\alpha_n T_{n,k}^l + (1 - \alpha_n) T_{n,k}^c\right] \quad (6)$$

Correspondingly, the energy consumption of mobile device can be calculated as

$$E_{n,k} = (1 - x_k)\left[\alpha_n E_{n,k}^l + (1 - \alpha_n) E_{n,k}^c\right] \quad (7)$$

Our goal is to minimize the energy cost of mobile device while guaranteeing the quality of service. In this paper, we assume that a specific user n knows task k when the user requests the task, i.e., the number of requests for the task is known. Thus, the problem can be expressed as follows:

$$\begin{aligned} &\underset{\mathbf{x},\alpha}{\text{minimize}} \sum_{n=1}^{N} E_{n,k} \\ &\text{subject to } C1: \sum_{k=1}^{K} x_k s_k \le c_e \\ &\qquad C2: \sum_{n=1}^{N} \alpha_n f_n^c \le c_s \\ &\qquad C3: T_{n,k} \le D_n, \forall n \in \mathcal{N}, \forall k \in \mathcal{K} \\ &\qquad C4: x_k \in \{0, 1\}, \forall k \in \mathcal{K}, \\ &\qquad C5: \alpha_n \in [0, 1], \forall n \in \mathcal{N} \end{aligned} \quad (8)$$

where the objective function computes the minimal energy consumed by the mobile device through deployment of task caching and offloading. The first constraint condition (C1) is that the data size of cached computing task cannot exceed the edge cloud caching capacity. Constraint (C2) ensures the total required computation resources for the offloaded task should not exceed the computation capacity of the edge cloud. The next constraint (C3) shows that the task execution for user n is successful before the required deadline. The constraint (C4) ensures that the task caching decision variable is a binary variable. Finally, constraint (C5) show that the task is separable.

III. ENERGY EFFICIENT TASK CACHING AND OFFLOADING

In this section, we will give the solutions to above optimal problems, which are related with both aspects of task caching and offloading. Here, we use alternative optimization techniques and consider the following two sub-problem, (i) Task offloading problem: when the strategy of task caching is given, i.e, $\mathbf{x} = \mathbf{x^0}$, original problem becomes convex optimization problem about α. Then, we can obtain the optimal solution α^*. (ii) Task caching problem: when α^* is fixed, this sub-problem is transferred to 0-1 integer programming problem by the use of branch and bound algorithm, the optimal solution can be obtained.

A. ENERGY EFFICIENT TASK OFFLOADING

In this subsection, we will give the energy efficient task offloading scheme.

Theorem 1: Given $\mathbf{x} = \mathbf{x^0}$, the original optimization problem in (8) with respect to α is a convex optimization problem.

Proof: Given $\mathbf{x} = \mathbf{x^0}$, the objective function becomes function of α. Since the $x_k T_{n,k}^{pro}$ in the objective function is independent of α, Thus, we can denote the objective function with respect to α as $f(\alpha)$, and can be calculated as follows:

$$f(\alpha) = \sum_{n=1}^{N} (1 - x_k^0)\left[\alpha_n E_{n,k}^l + (1 - \alpha_n) E_{n,k}^c\right] \quad (9)$$

Furthermore, the optimization problem in (8) with respect to α can be written as follows:

$$\begin{aligned} &\underset{\alpha}{\text{minimize}}\ f(\alpha) \\ &\text{subject to} \sum_{n=1}^{N} \alpha_n f_n^c \le c_s \\ &\qquad T_{n,k} \le D_n, \forall n \in \mathcal{N}, \forall k \in \mathcal{K} \\ &\qquad \alpha_n \in [0, 1], \forall n \in \mathcal{N} \end{aligned} \quad (10)$$

Since the objective function is linear and the constrain conditions are also linear, the optimization problem is convex problem. □

Since the optimization problem (10) is convex problem, therefore, we can obtain the optimal task offloading strategy α^* using well-studied optimization techniques such as interior point method.

B. ENERGY EFFICIENT TASK CACHING

According to the above discussion, we have obtained the optimal task offloading scheme α^*. In this subsection, we will give the best task caching scheme. When $\alpha = \alpha^*$, the objective function becomes function of $\mathbf{x}$ and the objective function can be expressed as follows:

$$g(\mathbf{x}) = \sum_{n=1}^{N} (1 - x_k)\left[\alpha_n^* E_{n,k}^l + (1 - \alpha_n^*) E_{n,k}^c\right] \quad (11)$$

Furthermore, the optimization problem in (8) with respect to $\mathbf{x}$ can be expressed as follows:

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}}\ g(\mathbf{x}) \\ &\text{subject to} \sum_{n=1}^{N} \alpha_n f_n^c \le c_s \\ &\qquad T_{n,k} \le D_n, \forall n \in \mathcal{N}, \forall k \in \mathcal{K} \\ &\qquad x_k \in \{0, 1\}, \forall k \in \mathcal{K}, \end{aligned}$$

Thus, the objective function is transformed into 0-1 linear programming problem with respect to $\mathbf{x}$. By the use of branch and bound algorithm, the solution can be calculated. Thus, the linear iterative algorithm can be utilized and obtain the approximate optimal solution. The optimal result represents the task caching and task offloading strategy of edge cloud.

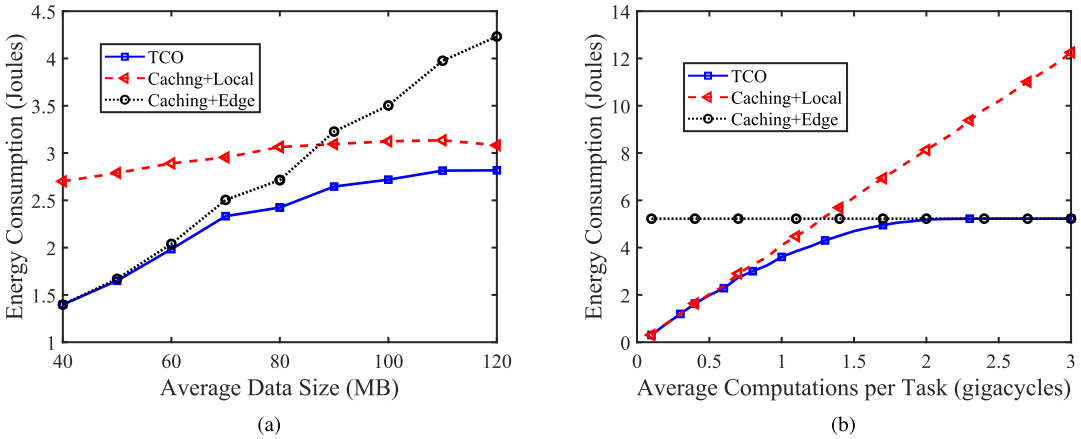

FIGURE 4. Effect of task offloading. (a) Energy consumption over different data size of computation task; (b) Energy consumption over different required computation capacity of task. The default setting is $n = 100$, $c_e = 500$ MB, ω follows normal distribution with an average of 0.8 gigacycles per task, s follows uniform distribution with an average of 100 MB.

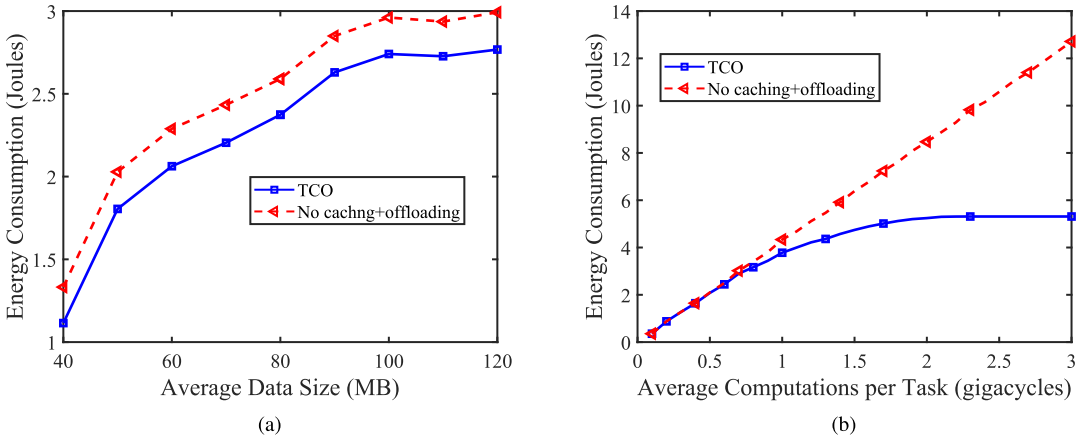

FIGURE 5. Effect of task caching or not. (a) Energy consumption over different data size of computation task; (b) Energy consumption over different required computation capacity of task. The default setting is $n = 100$, $c_e = 500$ MB, ω follows normal distribution with an average of 0.8 gigacycles per task, s follows uniform distribution with an average of 100 MB.

IV. PERFORMANCE EVALUATION

A. EXPERIMENT SETUP

In this section, we consider a edge computing ecosystem which includes edge cloud and mobile devices, where mobile devices have computation-intensive and data-intensive tasks. In addition, we assume that edge cloud is deployed near to the wireless access point. Thus, mobile devices can offload task to edge cloud through wireless channel. We also assume that transmission bandwidth B and transmitting power P_n of mobile device are 20 MHz and 0.5 W, respectively, and the corresponding noise power $\sigma^2 = 2 \times 10^{-13}$. The wireless channel gain H_n is modeled as $H_n = 127+30\times\log d$, where d is the distance between user n and edge cloud.

For user task $u_{n,k}$, we assume that required computing capacity ω_k and data size s_k are generated by a probability distribution(i.e., normal distribution and uniform distribution) [4]. For the task popularity, we assume that the number of task requests follows the Zipf distribution. Furthermore, we assume that the computing capabilities of edge cloud and mobile device are 25 GHz and 1 GHz, respectively.

B. EFFECTIVE TASK CACHING AND OFFLOADING EVALUATION

1) EFFECT OF TASK OFFLOADING

To evaluate the offloading, we compare the TCO with optimal task caching and following task offloading strategy.

- Caching+Local: First, the computation tasks are cached according to the caching policy proposed in this paper, and secondly, the tasks that are not cached will only be handled locally rather than being offloaded to edge cloud for processing.
- Caching+Edge: The not cached tasks are offloaded to edge cloud for processing relative to Caching+Local.

From Fig. 4, we can see that the energy consumption of proposed TCO is the lowest, i.e., reasonably deploying caching placement and task offloading can effectively reduce the energy cost of mobile device. From the Fig. 4(a), we can also see that the difference of energy consumption between TCO and caching+edge is little when the data size of tasks is small. Also, the difference between TCO and caching+local

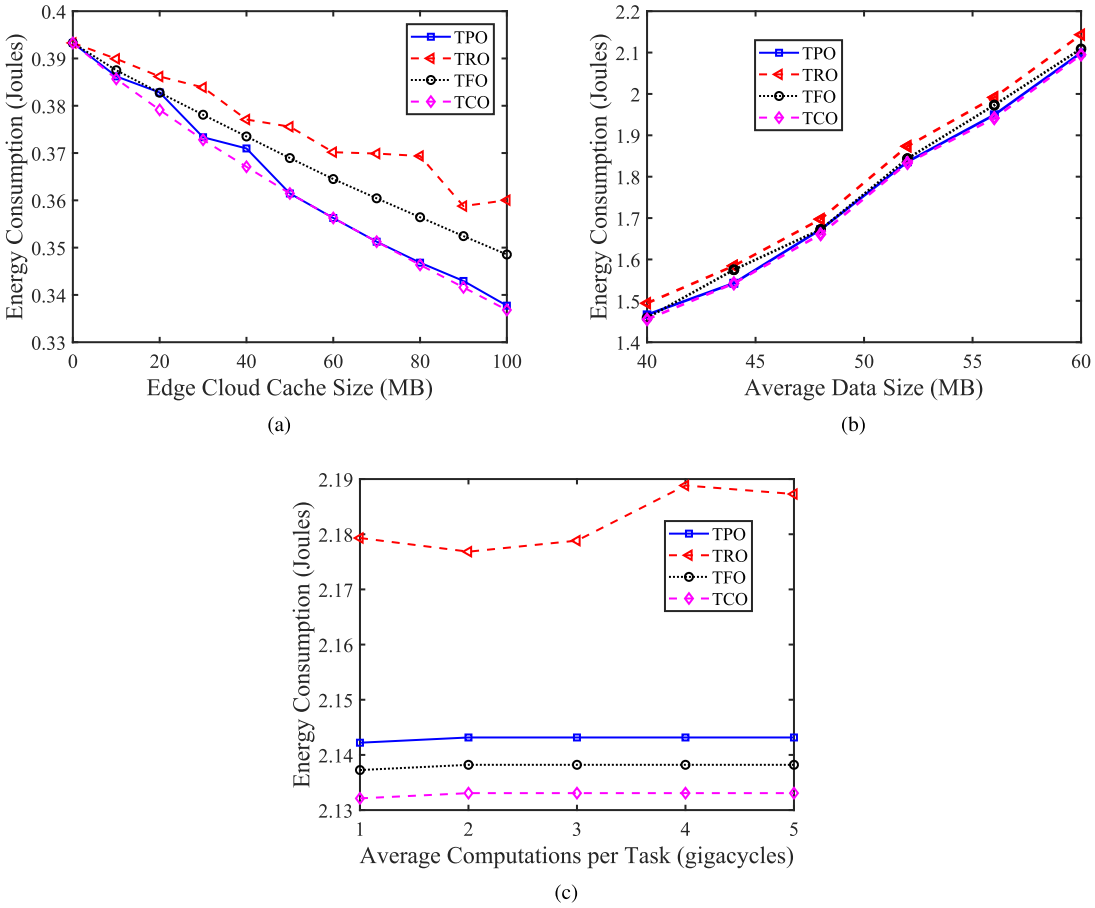

FIGURE 6. Effect of task caching. Energy consumption achieved by TPO (task popular caching and offloading), TRO (task random caching and offloading), TFO (task femtocaching and offloding) and TCOfor various value of (a) the edge cloud capacity, (b) the average data size of per task and (c) the average computations per task. The default setting is $n = 100$, $c_e = 100$ MB, ω follows normal distribution with an average of 2 gigacycles per task, s follows uniform distribution with an average of 50 MB.

is little when the data size is large. So we can conclude that under the same required computation capacity of tasks, the tasks should be processed at edge cloud when the data size of tasks is small. Conversely, if the data size is large, the tasks should be handled locally. Similarly, we can conclude from Fig. 4(b) that under the same data size of tasks, the tasks should be handled locally when the required computation capacity is relatively small, and processed at edge cloud when the computation capacity is large.

2) EFFECT OF TASK CACHING

We describe the comparison of energy consumption in terms of two cases, i.e., task not cached and task cached. As it can be seen in Fig. 5, when the task is cached, the energy consumption of mobile device is lower than that without caching. Thus, computing task caching can reduce the energy consumption. From the Fig. 5(a) and Fig. 5(b), we can also see that the bigger the task data size and computation capactiy is, the higher the energy cost is.

To evaluate the caching strategy, we compare the TCO strategy proposed in this paper with optimal task offloading and the following caching strategies.

- Task popular caching and offloading (TPO): For edge cloud, the edge cloud caches the computing task with the maximum number of requests, till reaching the caching capacity of edge cloud.
- Task random caching and offloading (TRO): The edge cloud caches the computing task randomly, till reaching the caching capacity of edge cloud.
- Task Femtocaching and offloading (TFO): The caching capacity of computing task is set as being empty at the start. Iteratively, add a task to a cache that minimum the total energy consumption, till the caching capacity of edge cloud.

From the Fig. 6(a-c), the TCO proposed in this paper is optimal, and the TRO is relatively poor. This is because the random cache fails to consider the number of task requests and also fails to consider the task computation amount and data size of computing task in the case of the computing task caching. The TPO only considers the number of task requests, rather than considering the data size and computation amount of task comprehensively. TFO considers the computation amount, data size and request of the task to a certain extent. From the Fig. 6(a), we can also see that the

larger the cache capacity of the edge cloud, the smaller the energy consumption is. This is because the capacity of the cache becomes larger, which lead to cache more tasks, thus energy consumption can be reduced. From the Fig. 6(b) and Fig. 6(c), we can get the impact of task data size on the algorithm is less than the impact of computing capacity on the algorithm.

V. CONCLUSION

In this paper, we first proposed the task caching on edge cloud, to the best of our knowledge, this is the first study of task caching for mobile edge computing. Furthermore, we investigate task caching and offloading strategy that decides which tasks should be cached and how much task should be offload. The objective is to minimize the total energy consumed by mobile device while meet the users delay requirement. We formulate this problem as an mixed integer nonlinear programming and propose efficient algorithm to solve this problem. Simulation result have shown that our proposed scheme has low energy cost compared to other schemes. For future work, we will consider multiple edge cloud task caching and offloading strategies.

REFERENCES

[1] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5G networks: Architecture and delay analysis," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 70–75, Feb. 2018.
[2] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
[3] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.
[4] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.
[5] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.
[6] M. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 1285–1293.
[7] X. Wang, H. Wang, K. Li, S. Yang, and T. Jiang, "Serendipity of sharing: Large-scale measurement and analytics for device-to-device (D2D) content sharing in mobile social networks," in *Proc. IEEE SECON*, San Diego, CA, USA, Jun. 2017, pp. 1–5, doi: 10.1109/SAHCN.2017.7964925.
[8] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, "Cognitive Internet of vehicles," *Comput. Commun.*, 2018, doi: 10.1016/j.comcom.2018.02.006.
[9] X. Wang, Y. Zhang, V. C. M. Leung, N. Guizani, and T. Jiang, "D2D big data: Content deliveries over wireless device-to-device sharing in realistic large scale mobile networks," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 32–38, Feb. 2018.
[10] M. Chen, Y. Hao, L. Hu, K. Huang, and V. Lau, "Green and mobility-aware caching in 5G networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 12, pp. 8347–8361, Dec. 2017.
[11] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[12] M. Chen, Y. Hao, M. Qiu, J. Song, D. Wu, and I. Humar, "Mobility-aware caching and computation offloading in 5G ultra-dense cellular networks," *Sensors*, vol. 16, no. 7, pp. 974–987, 2016.
[13] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2716–2720.
[14] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band Internet of Things," *IEEE Access*, vol. 5, pp. 20557–20577, 2017.

YIXUE HAO received the B.E. degree from the Henan University, China, and the Ph.D. degree in computer science from the Huazhong University of Science and Technology, China, in 2017. He is currently the Post-Doctoral Scholar with the School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include 5G network, Internet of Things, edge caching, and mobile edge computing.

MIN CHEN (SM'09) was an Assistant Professor with the School of Computer Science and Engineering, Seoul National University (SNU). He was a Post-Doctoral Fellow with SNU for one and half years. He was a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, The University of British Columbia (UBC) for three years. He has been a Full Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST), since 2012. He is the Director of the Embedded and Pervasive Computing Lab, HUST. He has authored over 300 paper publications, including over 200 SCI papers, over 80 IEEE transactions/journal papers, 18 ISI highly cited papers, and eight hot papers. He has published four books: *OPNET IoT Simulation* (HUST Press, 2015), *Big Data Inspiration* (HUST Press, 2015), *5G Software Defined Networks* (HUST Press, 2016), and *Introduction to Cognitive Computing* (HUST Press, 2017), a book on big data: *Big Data Related Technologies* (2014), and a book on 5G: *Cloud Based 5G Wireless Networks* (2016) with Springer Series in computer science. His latest book (co-authored with Prof. K. Hwang), entitled *Big Data Analytics for Cloud/IoT, and Cognitive Computing* (U.K.: Wiley, 2017). His research interests include cyber physical systems, IoT sensing, 5G networks, mobile cloud computing, SDN, healthcare big data, medical cloud privacy and security, body area networks, emotion communications, and robotics. He received the Best Paper Award from QShine 2008, the IEEE ICC 2012, ICST IndustrialIoT 2016 and the IEEE IWCMC 2016, and the IEEE Communications Society Fred W. Ellersick Prize in 2017. He is the Chair of the IEEE Computer Society Special Technical Communities on Big Data. He is the Co-Chair of the IEEE ICC 2012-Communications Theory Symposium and the Co-Chair of the IEEE ICC 2013-Wireless Networks Symposium. He is the General Co-Chair of the IEEE CIT-2012, Tridentcom 2014, Mobimedia 2015, and Tridentcom 2017. He is a Keynote Speaker for CyberC 2012, Mobiquitous 2012, Cloudcomp 2015, IndustrialIoT 2016, and The 7th Brainstorming Workshop on 5G Wireless. He serves as an Editor or Associate Editor for *Information Sciences*, *Information Fusion*, and the IEEE Access. He is a Guest Editor for the IEEE Network, the IEEE Wireless communications, and the IEEE Transactions on Service Computing. His Google Scholars Citations reached over 12,000 with an h-index of 53. His top paper was cited over 1100 times.

LONG HU was a Visiting Student with the Department of Electrical and Computer Engineering, The University of British Columbia, from 2015 to 2017. He has been currently a Lecturer with the School of Computer Science and Technology, Huazhong University of Science and Technology, China, since 2017. His research interests include the Internet of Things, software-defined networking, caching, 5G, body area networks, body sensor networks, and mobile cloud computing.

M. SHAMIM HOSSAIN (SM'09) received the Ph.D. degree in electrical and computer engineering from the University of Ottawa, Ottawa, ON, Canada. He is currently a Professor with the Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. He is also an Adjunct Professor with the School of Electrical Engineering and Computer Science, University of Ottawa. He has authored and co-authored around 165 publications, including refereed IEEE/ACM/Springer/Elsevier journals, conference papers, books, and book chapters. His research interests include cloud networking, social media, Internet of Things, cloud and multimedia for healthcare, smart health, and resource provisioning for big data processing on media clouds.

Dr. Shamim is a member of the ACM and ACM SIGMM. He has served as a member of the organizing and technical committees of several international conferences and workshops. He was a recipient of a number of awards including, the Best Conference Paper Award, the 2016 *ACM Transactions on Multimedia Computing, Communications and Applications* Nicolas D. Georganas Best Paper Award, and the Research in Excellence Award from King Saud University. He has served as the co-chair, general chair, workshop chair, publication chair, and TPC for over 12 IEEE and ACM conferences and workshops. He currently serves as the Co-Chair of the first IEEE ICME Workshop on Multimedia Services and Tools for Smart-health MUST-SH 2018. He served as a Guest Editor for the IEEE TRANSACTIONS ON INFORMATION TECHNOLOGY IN BIOMEDICINE (currently JBHI), the *International Journal of Multimedia Tools and Applications* (Springer), *Cluster Computing* (Springer), *Future Generation Computer Systems* (Elsevier), *Computers and Electrical Engineering* (Elsevier), and the *International Journal of Distributed Sensor Networks*. He currently serves on the Editorial Board for the IEEE MULTIMEDIA, the IEEE ACCESS, *Computers and Electrical Engineering* (Elsevier), *Games for Health Journal*, and the *International Journal of Multimedia Tools and Applications* (Springer). He currently serves as a Lead Guest Editor for the *IEEE Communication Magazine*, the IEEE TRANSACTIONS ON CLOUD COMPUTING, the IEEE ACCESS, *Future Generation Computer Systems* (Elsevier), and *Sensors* (MDPI).

AHMED GHONEIM received the M.Sc. degree in software modeling from the University of Menoufia, Shibin Al Kawm, Egypt, in 1999, and the Ph.D. degree in software engineering from the University of Magdeburg, Magdeburg, Germany, in 2007. He is currently an Assistant Professor with the Department of Software Engineering, King Saud University, Riyadh, Saudi Arabia. His current research interests include address software evolution, service oriented engineering, software development methodologies, quality of services, net-centric computing, and human–computer interaction.