Joint Design of Caching and Offloading for Mobile Edge Computing in 5G Heterogenous Networks

Jianxiong Xiao[1,2,3], Jing Liu[1], Xiang Chen[2,3(✉)], Yuhan Dong[4], and Terng-Yin Hsu[5]

[1] College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China
[2] School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China
chenxiang@mail.sysu.edu.cn
[3] Research Institute of Tsinghua University in Shenzhen (RITS), Shenzhen 518057, China
[4] Shenzhen International Graduate School, Tsinghua University, Shenzhen, China
[5] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

Abstract. With the rapid development of the information and communication technology, more and more computation-intensive and also latency-sensitive applications have been emerging. Mobile edge computing (MEC) has been regarded as one of the most significant technologies to satisfy the rigorous requirements of these applications. However, the efficient allocation of MEC resources in multiple dimensions remains an obstacle to further develop MEC in 5G and beyond 5G heterogeneous networks. In this paper, we formulate a joint optimization problem of caching and offloading for MEC systems to minimize the total latency of computation tasks. Following a divide and conquer idea, we first propose a caching scheme oriented to computation tasks with a comprehensive consideration in popularity, data size and computation complexity. Then, we design a caching-aware offloading strategy to make more efficient allocation of communication and computation resources based on the caching situation at the MEC server. Simulation results verify that the proposed joint design can achieve lower total latency comparing to the basic or separate schemes.

Keywords: edge computing · active caching · task offloading

1 Introduction

The recent years have witnessed the explosive growth of the emerging mobile applications like interactive games, virtual reality and natural language process-

The work is supported partly by the State's Key Project of Research and Development Plan under Grants 2019YFE0196400, and partly by the Shenzhen Natural Science Foundation under Grant JCYJ20200109143016563.

K. Rabie et al. (Eds.): IoTaaS 2022, LNICST 506, pp. 94–105, 2023.
https://doi.org/10.1007/978-3-031-37139-4_9

ing. Mobile cloud computing (MCC) has played a significant role to facilitate the development of these applications, as it allows the users to upload some computation-intensive tasks to it. As such, even a mobile device with limited capabilities can offer fascinating and high-quality services to its user. Nevertheless, the long physical distance between the users and the MCC server can produce a noticeable transmission delay and greatly infect the experience of some time-sensitive services. Moreover, the transmissions might require multiple forwards or relays, which further brings uncertain delay fluctuation [1]. Mobile edge computing (MEC) is novel paradigm proposed to tackle this problem. The MEC server is usually equipped to a cellular base station that is much closer to the clients. A shorter distance brings a smaller latency and hence, the MEC sever is capable to provide nearly real-time response to the mobile users.

However, comparing with the MCC server, the MEC server only possesses limited storage and computation resources in most cases. Therefore, the management of these resources is crucial to take advantage of the MEC. Task offloading has been an issue of great concern since the MEC is proposed. It is mostly about how to allocate computation resources to a great number of clients, with the objective to satisfy the different requirements on latency from various users. On the other hand, active caching has also been a trending topic all the time. It involves how to selectively store some popular contents like audios, videos or movies, so that the clients no longer need to download them from the remote cloud. Recently, some researchers have argued that the active caching can also be a vital approach to help further reducing the total delay of some computation-intensive tasks [2]. This is because, users within a certain range might share the same computing tasks. Caching the results of these tasks enables the MEC server to respond without any computing latency, and meanwhile prevents from wasting computation resources to duplicated calculation. In fact, offloading and caching can be further associated together to acquire more latency reduction.

MEC is a key component of 5G and beyond 5G (B5G) networks. In 5G and B5G cellular systems, there will be various kinds of base stations (BS) and they can be roughly categorized as micro base station (MBS) and small base station (SBS). This multi-station circumstance actually forms a heterogenous network. As such, a client might be served by several BSs simultaneously. This makes the MEC offloading further couple with the radio resource allocation between different BSs. Therefore, the offloading-caching problem mentioned above will further extend to the joint optimization on the utilization of communication, computation and cache resources in 5G/B5G heterogenous networks. Although there has been a great body of research on offloading and caching separately, this kind of collaborative problem still need further investigation.

In this paper, we focus on the joint design of caching and offloading for MEC in 5G heterogenous networks. Unlike caching content-data like videos, we notice that the caching of computation results should no only take the popularity and data size but also the computational complexity into consideration. Therefore, we design a comprehensive caching strategy based on all these factors. In addition, the caching policy will infect the latency gain of task offloading. Therefore, we

further design a offloading method with awareness of the caching situation at the MEC server. Our contributions can be summarized as:

- We formulate a latency optimization problem that involves caching, offloading and base station cooperation. Following a divide and conquer idea, we propose a jointly designed solution to achieve better latency performance.
- We propose a novel caching policy oriented to computation tasks, which simulta-neously considers the popularity, result size and computation complexity of tasks to save more time.
- We design an offloading scheme that takes into account both the caching decisions of the MEC server and the channel conditions between the users and the BSs when allocating computational and bandwidth resources. Through properly prioritizing the clients tasks, the scheme can provide more latency gain.

The rest of this paper is organized as follows: We provide a brief review of the related works in Sect. 2 and present the system model and problem formulation in Sect. 3. The proposed methods are elaborated in Sect. 4 and the numerical results are reported in Sect. 5. Section 6 concludes this work.

2 Related Works

MEC is a promising technology to improve the capabilities of the mobile cellular systems. Recently, some studies have tried to apply MEC to 5G heterogeneous networks. For instance, an energy-efficient computational offloading algorithm via allocating channel resources between the MBS and SBS are proposed in [3]. Besides, some works focus on combing the caching and offloading functions of the MEC. The authors in [4] construct a large dataset to train neural networks that can predict the future popularity of tasks and make better caching decisions. In [5], the authors investigate a computational offloading and data-content caching problem in NOMA-MEC systems and reduce the total completion latency by jointly optimizing the task offloading decision and data-content caching strategy.

In [6], the authors propose an active caching method for computational results but still lack comprehensive consideration on allocating the MEC resources in multiple dimensions. In [7], the authors consider the joint optimization of caching and computation, but the MEC is not deployed in heterogeneous networks and thus the allocation of communication resources between multiple BSs is not involved. In [8], the authors consider the collaborative offloading between the MEC and MCC servers and propose a cloud-edge-end algorithm to optimize the allocation of communicating, computing and caching resources and achieve lower latency. Nonetheless, this work still lacks considerations on the result size and computing complexity of tasks.

In summary, the existing works have investigated the joint design problem in different perspective, yet the optimization of MEC resource allocation based on comprehensive concerns of the popularity, result size and computation complexity of user tasks is still underexplored.

3 Problem Formulation

3.1 System Model

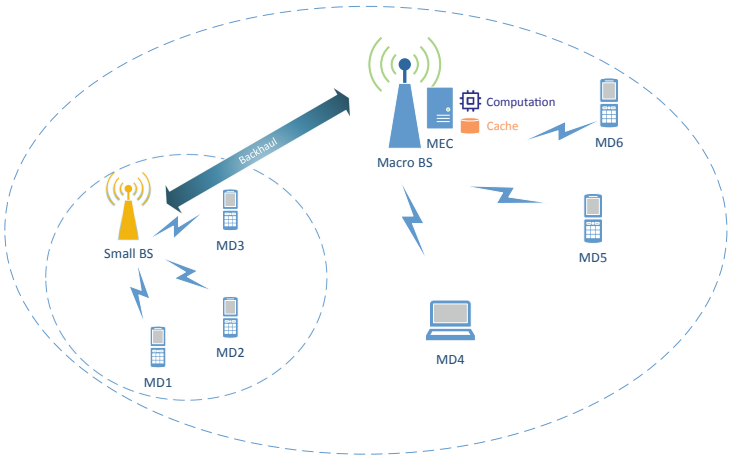

Fig. 1. The scenario of joint offloading and caching in heterogeneous networks.

As shown in Fig. 1, we consider a 5G heterogeneous network that consists of an MBS, an SBS and N mobile devices (MD) denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$ in this paper. The MBS can cover all the users while the SBS only serves those who are within a smaller range around it. The MEC server is only equipped at the MBS, and all the tasks offloaded to the SBS will then be transferred to the MBS through a backhaul link. Suppose that for a period of time, each user only generate a computation-intensive task. The one generated by the i-th user is referred as the i-th task, which is described by a triple

$$T_i = \{d_i^{in}, c_i, d_i^{out}\},$$

where d_i^{in}, d_i^{out} respectively denote the size of the input data and output result, and c_i represents the number of CPU cycles needed to calculate this task.

The MEC server has a certain amount of cache and computation resources. Based on these resources, on the one hand, the MEC server can accelerate the computation of the tasks that are uploaded to it. On the other hand, the MEC server can actively cache the results of some popular tasks to further reduce the latency of task offloading. According to the offloading approach and whether the results have been cached, there are five schemes available to obtain the result of a computation task, which are: (a) Local computing; (b) MBS offloading and MEC computing; (c) MBS offloading and MEC caching; (c) SBS offloading and MEC computing; (e) SBS offloading and MEC caching. Obviously, the latency of processing a task depends on which scheme is used. However, as the resources

at the MEC server are limited, it is impossible to cache or offload all the tasks. Therefore, finding a efficient way to allocate these resources and finally reduce the overall latency of the users is important. Yet before diving into the optimization of resource utilization, we first introduce the caching, computing and communicating models used in this work and analyze how they infect the latency.

3.2 Lantency Analysis

First, we use a binary variable y_i to indicate whether the MEC server has cached the result of the i-th task, which is given by

$$y_i = \begin{cases} 1 & \text{no cached result for the } i\text{-th task} \\ 0 & \text{otherwise.} \end{cases}$$

If a cached task is uploaded to the MEC server, it consumes no computing resource and the result can be returned immediately without computing delay. In exchange, the caching of the i-th task occupies a storage space of d_i^{out} bits.

Next, we move to the task offloading. We only consider binary offloading in this work, and thus there are three offloading choices for a task: the choice 1 is not offloading, the choice 2 is offloading via the MBS and the choice 3 is offloading via the SBS. We use a variable $\alpha_{i,j}$ to indicate the offloading choice of the i-th task, which is defined as

$$\alpha_{i,j} = \begin{cases} 1 & \text{the } i\text{-th task adopts choice } j \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $\alpha_{i,3}$ can only be 0 for those that can not access the SBS.

For the users that choose the scheme (a) i.e., local computing, the total delay is only the computation delay. Suppose the CPU frequency of the i-th mobile device is f_i^l, then the latency of using scheme (a) to calculate the i-th task is

$$t_i^l = \frac{c_i}{f_i^l}. \tag{1}$$

If a task is offloaded to the MEC server via the MBS or SBS, it costs time to upload the inputs of this task at first. In this work, we suppose the orthogonal frequency-division multiple access (OFDMA) is adopted by both the MBS and SBS for uplink transmission. The whole frequency band is divided into K orthogonal sub-channels, and each with a bandwidth of W. The MBS and SBS share these sub-channels without overlapping to avoid interference. Assume that k_i sub-channels are allocated to the i-th user. For those that offload tasks via the MSB, the transmission delay of uploading the task is

$$t_i^{m,t} = \frac{d_i^{in}}{r_i^m}.$$

Here, r_i^m is the uplink data rate from the i-th user to the MBS, given by

$$r_i^m = k_i W \log_2(1 + \frac{p_i g_i^m}{\sigma^2}),$$

where p_i is the transmission power of the i-th mobile device, g_i^m is the channel gain between the user and the MBS, and σ^2 is the noise power. Similarly, if the user selects to offload its task via the SBS, the transmission delay is

$$t_i^{s,t} = \frac{d_i^{in}}{r_i^s}, \quad r_i^s = k_i W \log_2(1 + \frac{p_i g_i^s}{\sigma^2}).$$

where g_i^s is the channel gain between the user and the SBS.

Subsequently, the MEC server will compute the result of the task if there is no cached result for it, i.e., $y_i = 1$. In this work, we assume that the MEC server can calculate multiple tasks simultaneously by distributing its computing capabilities. The total CPU frequency of the MEC server is referred to as F while the fraction allocated to the i-th task, if necessary, is denoted as v_i. As such, the computation delay of the task is

$$t_i^c = \frac{c_i}{v_i F}.$$

As the transmission power of the BS is usually much more adequate, and the size of the result is also much smaller than that of the input, the communication delay of returning the result is neglected in this work. Overall, for the users that choose scheme (b) and (c), the total latency is

$$t_i^m = t_i^{m,t} + y_i \cdot t_i^c. \tag{2}$$

For SBS offloading, there is a extra delay to transfer the computation tasks from the SBS to the MBS via the backhaul link. We suppose that the transfer delay is proportional to the data size with a coefficient φ, i.e.,

$$t_i^b = \varphi d_i^{in}.$$

Therefore, for the users that use scheme (d) and (e), the total latency is

$$t_i^s = t_i^{s,t} + t_i^b + y_i \cdot t_i^c. \tag{3}$$

3.3 Optimization Objective

Based on the analysis above, we can establish the optimization objective of this work as

$$\begin{aligned}
\underset{\{y_i, \alpha_{i,j}, k_i, v_i\}}{\text{minimize}} \quad & \sum_{i \in \mathcal{N}} \left(\alpha_{i,1} \cdot t_i^l + \alpha_{i,2} \cdot t_i^m + \alpha_{i,3} \cdot t_i^s\right) \\
\text{subject to} \quad & C1: y_i, \alpha_{i,j} \in \{0,1\}, \quad \forall i \in \mathcal{N}, j \in \{1,2,3\}, \\
& C2: \sum\nolimits_{i \in \mathcal{N}} (1 - y_i) \cdot d_i^{out} \leq M, \\
& C3: \sum\nolimits_{i \in \mathcal{N}} k_i \leq K, \\
& C4: \sum\nolimits_{i \in \mathcal{N}} v_i \leq 1, \\
& C5: \sum\nolimits_{j=1}^{3} \alpha_{i,j} = 1, \quad \forall i \in \mathcal{N}
\end{aligned} \tag{4}$$

which is to minimize the total latency of the system. Thereinto, C2 to C4 are constraints with respect to the resources of the MEC server or BSs. C2 restrains the space to cache the selected tasks is smaller than a given maximum M. C3 limits the number of sub-channels used by the MBS and SBS not to exceed the given amount. C4 states that the summation of the computation capabilities arranged to different tasks should be under the total capability of the MEC server. C5 ensures that each user chooses and only chooses one offloading scheme.

In the problem (4), there are binary variables $\alpha_{i,j}, y_j, k_i$, continous variables $v_i \in [0, 1]$ and the objective is comprised of non-linear terms derived from multiplying these undetermined variables. Actually, this optimization problem is a mixed integer nonlinear programming (MINLP) problem [9], which is non-convex and difficult to be solved directly. Therefore, we adopt a divide and conquer strategy to cope with it, which will be detailed in the next section.

4 Proposed Method

4.1 Caching Scheme Oriented to Computation Tasks

Since the storage resources are limited at the MEC server, it is impossible to cache all the task results. We assume that all the users generate tasks in an identical pattern and the generating probability of each task is different. It is intuitive that the MEC server should preferentially cache those tasks with higher generating probability. This is greatly reasonable for data-content caching. However, the caching of computation results has another significant factor that should be taken into consideration, which is the computation complexity. A task with relatively high generating probability and small space cost is not necessarily a good choice to be cached. If the computation complexity is quite low, the user are likely not to upload this task and the caching space is wasted. Therefore, we design a new policy that is more dedicated to caching computation tasks.

We suppose that there are totally L possible tasks, and the generating probabilities of these tasks follow the Zipf's distribution, which is commonly used in the literature [10]. The generating probability of the l-th task is given by

$$P_l = \frac{1/l^z}{\sum_{l=1}^{L} 1/l^z},$$

where z is the Zipf exponent. We suppose that the system has worked for a long time. Hence, the MEC is capable to build a Zipf's distribution accurate enough, according to the historical data. Then, we propose a novel metric for the MEC server to determine the caching strategy, which is defined as

$$T_l = \frac{P_l \cdot c_l}{d_l^{out}}. \tag{5}$$

It measures how much computation cost is expected to reduce per cache bit. Larger T_l means higher efficiency. Therefore, using (5) as a criterion, the MEC

server can first sort the L tasks into an ordered list $\{T_{x_1}, T_{x_2}, \ldots, T_{x_L}\}$, where $T_{x_1} \geq T_{x_2}$, and then cache the first X tasks. Here, X is the largest value that satisfies

$$\sum_{l=1}^{X} d_{x_l}^{out} \leq M.$$

4.2 Caching-Aware Offloading Algorithm

Once the MEC server has determined the caching scheme, we can remove the constraint C2 from the problem (4). And the next questions are which users should offload their tasks and how should they offload. To answer these two questions, we propose a two-step solution, which takes the caching decision into consideration to achieve better performance.

For the i-th task, we can first calculate the local-computing delay t_i^l with (1). However, the MBS-offloading and SBS-offloading delays are not available at this time, as the allocation of communication and computation resources is undecided yet. Hence, we alternatively define a concept of "full-resource delay" t_i^f to determine which tasks are more deserved to be offloaded. We suppose that all the bandwidth and computation resources are allocated to the i-th task, that is, $k_i = K$ and $v_i = 1$. Based on this, we can get the "full-resource" versions of the MBS-offloading and SBS-offloading delays according to (2) and (3), which are referred to as $t_i^{m,f}$ and $t_i^{s,f}$, respectively. For those users that are within both the coverage of the SBS and MBS, we define the latency gain as

$$\Delta D_i = t_i^l - \min(t_i^{m,f}, t_i^{s,f}),$$

while for those that are only covered by the MBS, the latency gain is given by

$$\Delta D_i = t_i^l - t_i^{m,f}.$$

Next, we sort the tasks into $\mathcal{N}_{sort} = \{n_1, n_2, \ldots, n_N\}$ in a descending order, where $D_{n_i} \geq D_{n_j}, \forall N > i > j > 0$. Then, we sequently determine the offloading and resource allocating for each task with a greedy strategy.

We denote the set of users choosing local computing as $\mathcal{N}_l$ while the sets of those offloading tasks via the MBS and the SBS are denoted as $\mathcal{N}_m$ and $\mathcal{N}_s$, respectively. Then the optimization objective in problem (4) can be equivalently transformed to

$$\underset{\{k_i, v_i\}}{\text{minimize}}\ L(\mathcal{N}_l, \mathcal{N}_m, \mathcal{N}_s) = \sum_{i \in \mathcal{N}_l} t_i^l + \sum_{i \in \mathcal{N}_m} t_i^m + \sum_{i \in \mathcal{N}_s} t_i^s. \tag{6}$$

The sets $\mathcal{N}_m, \mathcal{N}_s$ are initialized as empty sets while $\mathcal{N}_l$ is set to be $\mathcal{N}_{sort}$. Here, we take the first user in $\mathcal{N}_{sort}$ to illustrate how to update the user sets: If the n_1-th user can only access the MBS, then it is directly moved to $\mathcal{N}_m$. That is, we update

$$\mathcal{N}_m \leftarrow \mathcal{N}_m \cup \{n_1\}.$$

If the user are in the coverage of both the MBS and SBS, then we update

$$\begin{cases} \mathcal{N}_m \leftarrow \mathcal{N}_m \cup \{n_1\} & \text{if } t_i^{m,f} < t_i^{s,f} \\ \mathcal{N}_s \leftarrow \mathcal{N}_s \cup \{n_1\} & \text{otherwise.} \end{cases}$$

In both cases, we then exclude n_1 from $\mathcal{N}_l$ as

$$\mathcal{N}_l \leftarrow \mathcal{N}_l \backslash \{n_1\}.$$

Given the sets $\mathcal{N}_l, \mathcal{N}_m, \mathcal{N}_s$, the constraints and objective functions are convex functions, and the feasible domains of k_i and v_i are convex sets. The problem (6) becomes a convex optimization problem [11]. Therefore, we can obatin the current minimal total delay

$$T_1 = \underset{\{k_i, v_i\}}{\text{minimize}}\ L(\mathcal{N}_l, \mathcal{N}_m, \mathcal{N}_s)$$

by using the CVX toolkit to determine $\{k_i, v_i\}$. After that, we can move $n_2, n_3, \ldots$ to $\mathcal{N}_m$ or $\mathcal{N}_l$ until $T_i < T_j$, which mean no more latency gain can be obtain through task offloading.

So far, we have set up a complete procedure to cope with the collaborative optimization of caching and offloading in a heterogeneous network. And we will verify the benefits of this algorithm in the next section.

5 Numerical Result

5.1 Simulation Setup

In the simulation environment, the MBS covers a radius of 500 m, and the SBS is located 150 m away from the MBS with a covering radius of 150 m. The latency coefficient φ of transmission via backhaul link between the MBS and SBS is set to 0.0001 s/KB. The total bandwidth is 20 MHz and evenly divided into $K = 20$ sub-channels. The MEC server has a CPU frequency F of 40 GHz and it can only cache the results of half of the possible tasks. The number of users is 30, and 60% of these users are randomly located at the whole area while the rest 40% are set to be randomly located within the coverage of the SBS. The input size d_i^{in} (KB) for Sect. 5.2 and 5.3 is set to follow uniform distributions $U[1000, 5000]$ and $U[500, 1000]$, respectively. For each task, the CPU cycle c_i (GHz) follows a uniform distribution $U[1, 2]$ and the result size d_i^{out} (KB) follows a normal distribution $\mathcal{N}(50, 80)$ for all simulations.

5.2 Benefits of the Proposed Caching Scheme

We have proposed a caching scheme oriented to computation tasks in Sect. 4.1, which particularly takes the output size and computation complexity of tasks into consideration. In this section, we evaluate the benefits of this comprehensive consideration by comparing it with two basic schemes: Basic Scheme 1 randomly

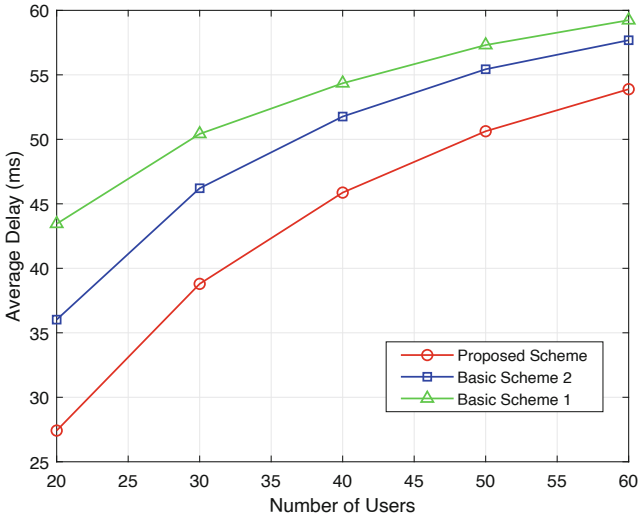

Fig. 2. Comparing the delay of the proposed caching scheme versus two basic schemes

caches a proportion of the tasks under the limitation of caching space. Basic Scheme 2 only considers the popularity of the tasks, and thus caches a certain number of tasks with the highest generating probability [12]. All of them use the offloading strategy proposed in Sect. 4.2. We use the average latency of all the users as metric and the results are presented in Fig. 2. It can be seen that for all the three schemes, the latency increases when more users are served. However, no matter how many users are given, the proposed method provides a lower latency than the other two basic schemes.

5.3 Benefits of the Proposed Offloading Scheme

We have proposed a caching-aware offloading scheme in Sect. 4.2, which takes the caching results into consideration to determine which tasks should be uploaded and how many communication and computation resources should be allocated to them. To show the advantages of the proposed strategy, we compares it with two basic schemes: Basic Scheme 1 has no cache space and thus provides an upper bound, while the Basic Scheme 2 determines the offloading policy regardless of the caching decision at the MEC server. The proposed scheme and the Basic Scheme 2 both use the caching method proposed in Sect. 4.1. The results are reported in Fig. 3, which reveal that the proposed method always outperforms the other two basic schemes under different configurations. Moreover, it brings higher latency gain when F is relatively small, which indicates that the caching-aware offloading method can play a important role when the MEC is not powerful enough.

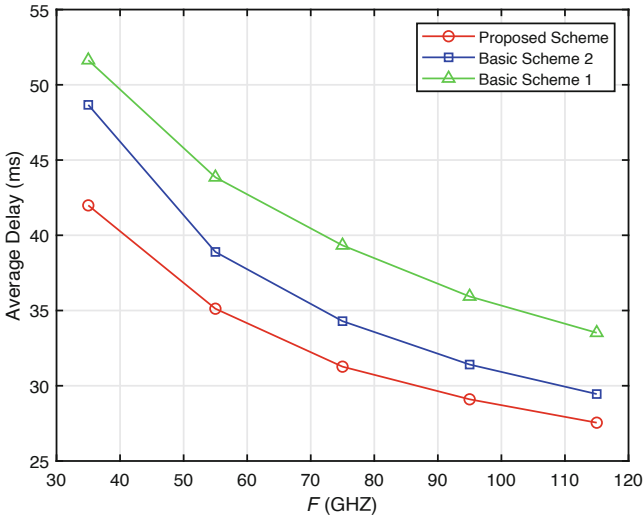

Fig. 3. Comparing the delay of the proposed offloading scheme versus two basic schemes

6 Conclusion

In this paper, aiming at optimizing the resource utilization in 5G heterogeneous networks, we propose a joint design of caching and offloading for MEC systems. We analyze the latency of different caching and offloading schemes and formulate an optimization problem to minimize the total latency of users. With a divide and conquer idea, we first design a caching scheme oriented to computation tasks with comprehensive consideration of popularity, data size and computation complexity. And then, we propose a caching-aware offloading method to make more reasonable and efficient allocations on communication and computation resources. The simulation results verify that the proposed schemes can bring benefits in various situations.

References

1. Meng, S., Wang, Y., Miao, Z., Sun, K.: Joint optimization of wireless bandwidth and computing resource in cloudlet-based mobile cloud computing environment. Peer-to-Peer Networking Appl. **11**(3), 462–472 (2018)
2. Hu, Y.C., Patel, M., Sabella, D., Sprecher, N., Young, V.: Mobile edge computing-a key technology towards 5G. ETSI White Paper **11**(11), 1–16 (2015)
3. Zhang, K., et al.: Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access **4**, 5896–5907 (2016)
4. Chen, B., Liu, L., Sun, M., Ma, H.: IoTCache: toward data-driven network caching for internet of things. IEEE Internet Things J. **6**(6), 10064–10076 (2019)

5. Huynh, L.N., Pham, Q.V., Nguyen, T.D., Hossain, M.D., Shin, Y.R., Huh, E.N.: Joint computational offloading and data-content caching in NOMA-MEC networks. IEEE Access **9**, 12943–12954 (2021)
6. Elbamby, M.S., Bennis, M., Saad, W.: Proactive edge computing in latency-constrained fog networks. In: 2017 European Conference on Networks and Communications (EuCNC), pp. 1–6. IEEE (2017)
7. Li, S., Li, B., Zhao, W.: Joint optimization of caching and computation in multi-server NOMA-MEC system via reinforcement learning. IEEE Access **8**, 112762–112771 (2020)
8. Zhao, H., Wang, Y., Sun, R.: Task proactive caching based computation offloading and resource allocation in mobile-edge computing systems. In: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 232–237. IEEE (2018)
9. Chen, X., Jiao, L., Li, W., Fu, X.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans. Network. **24**(5), 2795–2808 (2015)
10. Golrezaei, N., Dimakis, A.G., Molisch, A.F.: Scaling behavior for device-to-device communications with distributed caching. IEEE Trans. Inf. Theory **60**(7), 4286–4298 (2014)
11. Borst, S., Gupta, V., Walid, A.: Distributed caching algorithms for content distribution networks. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)
12. Suyu, Z., Jiang, L.: Cloud edge and collaborative offloading strategy based on active caching. J. Comput. Eng. Des. **42**(8), 2124–2129 (2021)