Computation Offloading for Multi-Access Mobile Edge Computing in Ultra-Dense Networks

Hongzhi Guo, Jiajia Liu, and Jie Zhang

The authors study the MECO problem in UDN and propose a heuristic greedy offloading scheme as their solution. Extensive numerical results and comparisons demonstrate the necessity for and superior performance of conducting computation offloading over multiple MEC servers.

Abstract

The ultra-dense network (UDN) is envisioned to be an enabling and highly promising technology to enhance spatial multiplexing and network capacity in future 5G networks. Moreover, to address the conflict between computation-intensive applications and resource-constrained IoT mobile devices (MDs), multi-access mobile edge computing (MA-MEC), which provides the IoT MDs with cloud capabilities at the edge of radio access networks, has been proposed. UDN and MA-MEC are regarded as two distinct but complementary enabling technologies for 5G IoT applications. Over the past several years, lots of research on mobile edge computation offloading (MECO) — the key technique in MA-MEC — has emerged. However, it is noticed that all these works focused on the single-tier base station scenario and computation offloading between the MD and the MEC server connected to the macro base station, and few works can be found on the problem of computation offloading for MA-MEC in UDN (i.e., a multi-user ultra-dense MEC server scenario). Toward this end, we study in this article the MECO problem in UDN and propose a heuristic greedy offloading scheme as our solution. Extensive numerical results and comparisons demonstrate the necessity for and superior performance of conducting computation offloading over multiple MEC servers.

Introduction

To meet the demands of large traffic volume, massive connection density, and high peak data rate in next-generation wireless communication networks, that is, fifth generation (5G), lots of new techniques have been proposed in recent years. In particular, these techniques include coordinated multipoint (CoMP), massive multiple-input multiple-output (MIMO), millimeter-wave (mmWave) communications, non-orthogonal multiple access (NOMA) [1, 2], the ultra-dense network (UDN), and so on. Among them, UDN deploys a large number of short-range and low-power small cells (SCs) over the coverage area of macro base stations (MBSs), and it is envisioned to be an enabling and highly promising technology to provide users with high throughput and flexible access services in the 5G era [3].

Moreover, with the development of the Internet of Things (IoT), new applications, including remote e-health, intelligent transportation, the Internet of Vehicles, and so on, have emerged and attracted lots of attention. Note that most of these IoT applications have stringent requirements on computation resources, resulting in high energy consumption by IoT mobile devices (MDs). However, limited by their physical size, IoT MDs are always resource-constrained and have limited computation capabilities and battery life. The conflict between the computation-intensive applications and the resource-constrained lightweight IoT MDs brings an unprecedented challenge for future IoT development.

To address this issue, mobile edge computing (MEC) was first proposed by a European Telecommunications Standards Institute Industry Specification Group (ETSI ISG) in December 2014 as a potential solution, with the objective to integrate cloud computing functionalities into radio access networks efficiently and seamlessly. By offloading computation-intensive tasks to MEC servers in close proximity to users, MEC can not only reduce the traffic bottlenecks in the core and backhaul networks, but also reduce both the processing delay and the energy consumption at user equipments (UEs). Thus, MEC has been envisioned to be an enabling and highly promising technology in 5G networks once it was put forward. Later, in September 2016, MEC ISG dropped "Mobile" from MEC and renamed it "Multi-Access Edge Computing" so as to support more access modes including WiFi and wired access [4].

Furthermore, with the rapid development of IoT technology, the growing demands for ultra-low latency, massive connectivity, and high reliability of the massive IoT applications have yielded a critical issue in MEC. To address this issue, multi-access mobile edge computing (MA-MEC), which exploits integration of recent radio access technologies, including Long Term Evolution (LTE)/LTE-Advanced (LTE-A), WiFi, 5G, and so on, to enhance the access capacity of the IoT MDs, has been proposed and considered to be a potential solution. This evolution of MEC toward MA-MEC facilitates the development of UDN, and thus, UDN and MA-MEC are regarded as two distinct but complementary technologies in 5G networks.

Digital Object Identifier:
10.1109/MCOM.2018.1701069

The authors are with Xidian University;

Over the past several years there has been much research on mobile edge computation offloading (MECO) [5], which is a key technique in MEC/MA-MEC. For example, Zhang *et al.* [5] formulated the MECO problem in 5G heterogeneous networks as an energy-efficient optimization problem that aims to minimize the system energy consumption with given processing latency constraints, and presented a three-stage energy-efficient computation offloading scheme by adopting the type classification and the priority assignment for the MDs. Moreover, Zheng *et al.* [6] investigated the problem of multi-user MECO with dynamic user status and wireless channels, and proposed a multi-agent stochastic learning computation offloading scheme as their solution.

Nevertheless, we note that most available MECO research focused on the single-tier BS scenario, and the computation offloading strategies were very simple. Specifically, the MDs either execute the computation tasks locally at their own CPUs, or offload parts of the computationally intensive tasks to the MEC server connected to the MBS. If the MBS is congested due to the large number of IoT MDs and mobile applications, the quality of service (QoS) and quality of experience (QoE) drop dramatically. In fact, we can deploy some lightweight MEC servers at the SCs in UDN, and offload parts of the computation tasks to the MEC servers connected to the SCs so as to lessen the burden at the MBS. However, to the best of our knowledge, few works can be found on performing computation offloading for MA-MEC in UDN (i.e., a multi-user ultra-dense MEC server scenario). Toward this end, we provide this article to demonstrate the key features of MA-MEC in UDN, and investigate the problem of computation offloading for MA-MEC in UDN through a case study in a multi-user ultra-dense MEC server scenario.

In particular, we first analyze the computation overhead via adopting local computing at the MD and MECO, respectively, and then present a definition for the problem of computation offloading for MA-MEC in UDN. After that, a heuristic greedy offloading scheme is proposed as our solution, where the computation resources at the MD, and the MEC servers connected to the MBS and the SCs are utilized collaboratively. Numerical results corroborate the necessity for deploying more MEC servers in UDN, and the superior performance of conducting computation offloading among multiple MEC servers over multi-user single-MEC scenarios. Finally, we conclude the article in the last section.

Multi-Access Mobile Edge Computing in Ultra-Dense Networks

Generally, UDNs can be defined as networks with a large number of low-power SCs with small coverage installed inside a single MBS, as illustrated in Fig. 1. These SCs, whose number is greater than or at least equal to the number of UEs, are usually deployed in hotspots like airports, metro/train stations, markets, and residential areas by the operators/customers. Depending on their functionality and category, these SCs can be connected to the network via fiber, cable, or microwave links, or share the same wireless channels with the MBS [3]. By adopting the large number of densely deployed SCs, UDNs can dramatically improve network capacity, network coverage, spectrum efficiency, and data rates, and thus the user experience can be enhanced significantly [7]. Moreover, due to the much shorter distance between the UEs and the cells in UDNs, the scenario in which multiple SCs serve a single user becomes possible, which shifts the network service mode from cell-centric to user-centric. Furthermore, with the aid of the dual connectivity technique developed by the Third Generation Partnership Project (3GPP) or techniques like CoMP and massive MIMO, developed for 5G networks, a UE can simultaneously connect to an MBS and one or more SCs so as to further improve network throughput, spectrum efficiency, and data rates.

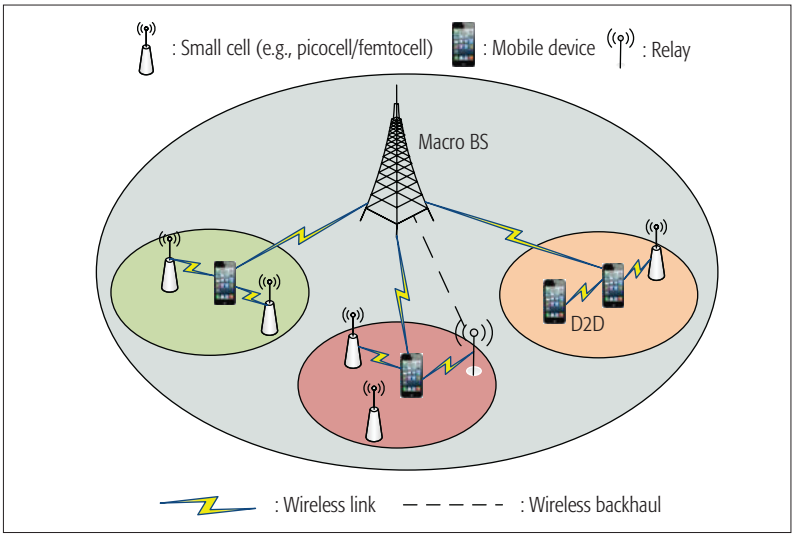

Figure 1. A generic network architecture for UDN.

Although the development of UDNs still faces many challenges, such as the bottleneck of backhaul capacity, increased interference and energy consumption, and increased delay, failures, and rates caused by unnecessary and frequent handovers, researchers have done lots of work recently, and a number of potential solutions like architecture changes in 5G networks, software-defined networking (SDN), interference and mobility management, hybrid fiber-wireless (FiWi) networks, and NOMA have been proposed [8–10]. Besides, as a potential key enabling technology in 5G networks, the integration of UDN with other 5G enabling technologies, such as mmWave, device-to-device (D2D), and MEC/MA-MEC, has gradually become the focus of some researchers recently [11, 12]. Among them, UDN and MA-MEC are regarded as two distinct but complementary technologies in 5G IoT applications. However, the success of MA-MEC in UDN still faces several challenges, including optimal computation/storage/radio resource allocation in UDNs, cell association caused by frequent handovers, and so on, among which the problem of efficient computation offloading among multiple MEC servers, as the key technique in MA-MEC, is particularly important.

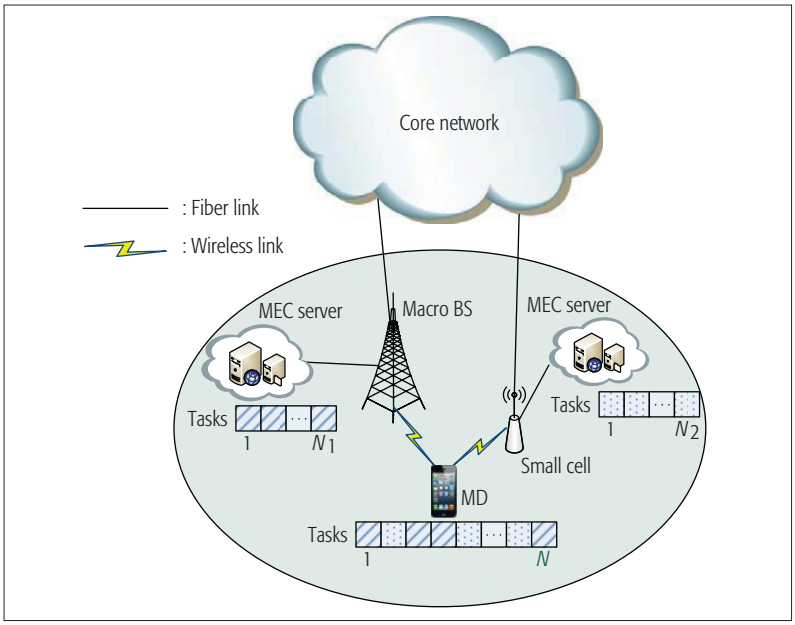

Figure 2. Computation offloading for MA-MEC in UDN.

Computation Offloading for MA-MEC in UDN

As illustrated in Fig. 2, we consider a computation offloading scenario for MA-MEC in UDN, where an MD is covered by an MBS and several SCs, and the MD can communicate with the MBS and access the Internet via the SCs simultaneously by adopting the dual connectivity technique. To provide users with better MEC services, one or more MEC servers are connected to the MBS and the SCs via dedicated fiber links. Without loss of generality, only the scenario with one MEC server connected to the MBS and the SCs is considered in this article. Moreover, we assume that there are N types of computation tasks to be processed at the MD (e.g., facial recognition, natural language processing, and interactive gaming), and the probability of type-i task being requested by the MD at a certain time is set as P_i ($0 < P_i < 1$).

With the aid of the MEC servers connected to the MBS/SC, the type-i task requested by the MD can be accomplished by adopting one of three offloading strategies: computing locally at the MD's CPU, offloading to the MEC server connected to the MBS, and offloading to the MEC server connected to the SC, and its corresponding processing delay and energy consumption can be calculated easily. For the sake of discussion, we denote the set $\{L, M, S\}$ as the offloading decisions of the computation tasks, which are local computing at the MD's CPU, offloading to the MEC server connected to the MBS, and offloading to the MEC server connected to the SC, respectively.

Computing Locally at the MD

Regarding the MD's type-i task, it can be generally described in three items including its input data size, the required CPU cycles to accomplish this task (i.e., its required computational capacity), and the maximum permissible processing delay for finishing this task. An MD can apply the methods in [13, 14] to obtain the information of the required CPU cycles for each type of computation task. If the MD decides to accomplish type-i task via computing locally at its own CPU, the processing time for finishing this task can be calculated as its required CPU cycles divided by the MD's computing ability, and the corresponding energy consumption can be further obtained by the product of its required CPU cycles and the consumed energy per CPU cycle of the MD.

Mobile Edge Computation Offloading

In order to reduce both the MD's energy consumption and the tasks' processing delay, the MD can offload parts of its tasks to the MEC server in close proximity. Different from previous research, there are two optional offloading choices for the MD in adopting dual connectivity UDNs (i.e., offloading to the MEC server connected to the MBS/SC). Let $\lambda_i^j \in \{0, 1\}$ denote the probability of type-i computation tasks choosing offloading decision j ($j \in \{L, M, S\}$), where $\lambda_i^j = 1$ means that the MD chooses offloading decision j to accomplish its type-i task, and $\lambda_i^j = 0$ otherwise. When the MD chooses to offload its type-i task to the MEC server, the processing time of the type-i task contains not only the time cost for transmitting its computation input and outcome between the MD and the MEC server, but also its execution time and queuing time at the MEC server.

In particular, the time cost for transmitting the computation input and outcome of the MD's type-i task between the MD and the MEC server can be calculated easily by adding up the wireless and optical transmitting time. Considering that the distribution of diverse computation tasks requested by the MD follows a Poisson distribution, and the arrival of the offloaded tasks at the MEC server is a Poisson process, we can model the arrival and the execution of the tasks at the MEC server as an M/M/1 queue. In this model, the service time at the MEC server follows an exponential distribution with mean service time $1/\mu$. We denote $\bar{\lambda}_j$ as the average arrival rate of the tasks at the MEC server by adopting offloading decision j, and it can be calculated easily by adding up the computational capacity of these tasks that choose offloading decision j. Thus, the average processing time for accomplishing a type-i task at the MEC server by adopting offloading decision j, which contains both the execution time and the queuing time, can be given by its required CPU cycles divided by the difference between $\mu_j - \bar{\mu}_j$. Then the corresponding energy consumption of the MD's type-i task in this case can be obtained easily by the product of its processing time and the MD's transmit power.

Problem Definition and Solutions

Obviously, choosing an optimal offloading decision for each type of task at the MD depends not only on its maximum permissible processing delay, but also the corresponding energy consumption. In this article, in view of the MD's limited battery life, our target is to design an energy-efficient computation offloading scheme for all types of tasks at the MD, with the objective to minimize the overall energy consumption of the MD while satisfying the maximum permissible processing latency of each type of computation task. In particular, this problem can be defined as follows.

Definition 1: Energy Consumption Minimization with Processing Time Constraints (EMTC): Given the information of each type of computation task at the MD, such as its input data size, its required computing capacity, its maximum permissible processing delay, and its requesting probability, the wireless channel bandwidth, and the computing capabilities of the MD and the MEC servers connected to the MBS and the SC, the problem of EMTC in UDN is to find an optimal offloading decision profile for all the types of computation tasks at the MD, which can achieve the overall minimum energy consumption while satisfying the given processing time constraints for each type of computation task.

Unfortunately, the problem of EMTC is NP-hard since we can easily transform it into the traditional maximum cardinality bin packing problem [15]. Hereafter, we present two schemes to solve the EMTC problem.

An Optimal Enumeration Offloading Scheme: To solve the EMTC problem in UDN, a brute force method is to enumerate all optional offloading decision combinations for all types of computation tasks of the MD, and choose an optimal offloading decision profile with the minimum energy consumption while satisfying the given processing time constraints, that is, an optimal enumeration offloading scheme (OEOS). The details of OEOS are described as follows.

Step 1: We first enumerate all optional offloading decision combinations for all types of computation tasks, and then calculate the overall energy consumption and the processing time for each type of task.

Step 2: We discard offloading decision combinations that cannot meet given latency constraints of the computation tasks from the set of optional offloading decision combinations for all types of computation tasks.

Step 3: After that, we choose an optimal offloading decision profile from the remaining offloading decision combinations that has the overall minimum energy consumption; otherwise, return NIL.

For the OEOS algorithm, it certainly can find an optimal solution if there is an optimal solution to the EMTC problem, since it enumerates all possible offloading decisions for all types of tasks at the MD and traverses the whole solution space. However, OEOS has very high computational complexity (i.e., $O(3^n)$), where n is the number of types of computation tasks at the MD, as each type of task has at most three offloading choices. Due to its exponential computational complexity, OEOS can only be implemented for a small number of types of computation tasks, and thus it is just presented as a benchmark to evaluate our heuristic greedy offloading scheme.

A Heuristic Greedy Offloading Scheme: Considering that OEOS is impractical, we further propose an approximation method to solve the EMTC problem, that is, a heuristic greedy offloading scheme (HGOS). The details of HGOS are briefly given as follows.

Step 1: For all types of computation tasks at the MD, we compute their processing time by adopting local computing, offloading to the MEC server connected to the MBS, and offloading to the MEC server connected to the SC, respectively, and discard those computation tasks that cannot meet the given processing time constraints. In particular, if the processing time of a type-i task by adopting the three offloading strategies separately does not satisfy the given maximum permissible processing delay, we discard the MD's type-i task from the set of computation tasks.

Step 2: For the MD's remaining computation tasks after step 1, we pick out those tasks that have to be offloaded to the MEC servers. Specifically, we compare all the remaining tasks' processing time by adopting local computing with their given processing time constraints. If their processing time by adopting local computing is greater than the given processing time constraints, these computation tasks must be offloaded to the MEC servers connected to the MBS/SC. After that, we perform the following step to obtain their offloading decision.

Step 3: For any type of computation task of the MD that has to be offloaded to the MEC servers, we determine whether it should be offloaded to the MEC server connected to the MBS or to the MEC server connected to the SC. For these computation tasks, we first compute their energy consumption by adopting offloading to the MEC server connected to the MBS and offloading to the MEC server connected to the SC, respectively. Then we assign those tasks that have the lower energy consumption by adopting offloading to the MEC server connected to the MBS with $\lambda_j^M = 1$, that is, the MD's type-j task should be offloaded to the MEC server connected to the MBS. After that, the remaining tasks are assigned with $\lambda_k^S = 1$, that is, the MD's type-k task should be offloaded to the MEC server connected to the SC.

Step 4: For the remaining tasks, which can be either accomplished locally at the MD's CPU or offloaded to the MEC servers connected to the MBS/SC, we first pick out the ones that can achieve the minimum energy consumption by adopting MECO, and then assign those tasks with the local computing decision. In particular, we assume that all remaining types of tasks should be offloaded to the MEC servers, and compute their corresponding energy consumption by adopting local computing, offloading to the MEC server connected to the MBS, and offloading to the MEC server connected to the SC, respectively. If the energy consumption of the MD's type-i task by adopting offloading to the MEC server connected to the MBS/SC is lower than that of adopting local computing, it is obvious that we should offload the type-i task to the MEC server connected to the MBS/SC. Otherwise, we compute the type-i task locally at the MD's CPU. Moreover, for those tasks that should be offloaded to the MEC server connected to the MBS/SC, we perform step 3 to obtain their final offloading decision.

After the above four steps, all types of computation tasks at the MD can find their offloading decision. Accordingly, the overall minimum energy consumption can be calculated easily. Compared to the OEOS algorithm, the HGOS algorithm can achieve a much higher computational efficiency (i.e., $O(n^2)$), where n is the number of types of computation tasks at the MD. Specifically, step 1 runs at most n times,

In view of the MD's limited battery life, our target is to design an energy-efficient computation offloading scheme for all types of tasks at the MD, with the objective to minimize the overall energy consumption of the MD while satisfying the maximum permissible processing latency of each type of computation task.

and the running time of step 3 is also $O(n)$. For step 2, it contains a two-tier loop, since step 3 needs to be performed to determine whether the task should be offloaded to the MEC server connected to the MBS or to the MEC server connected to the SC. In particular, the outer loop of step 2 runs at most n times, and the inner procedure (i.e., step 3) also runs at most n times, and thus, the running time of step 2 is $O(n^2)$. Similarly, we can calculate the running time of step 4, $O(n^2)$. Finally, the total running time of HGOS can be calculated by adding them together (i.e., $O(n + n^2 + n^2) = O(n^2)$). Note that our HGOS algorithm can only achieve a near-optimal solution to the EMTC problem, since it chooses task l from the remaining tasks randomly during step 4, and random selection of task l may result in different offloading decision profiles for all types of tasks and overall energy consumption.

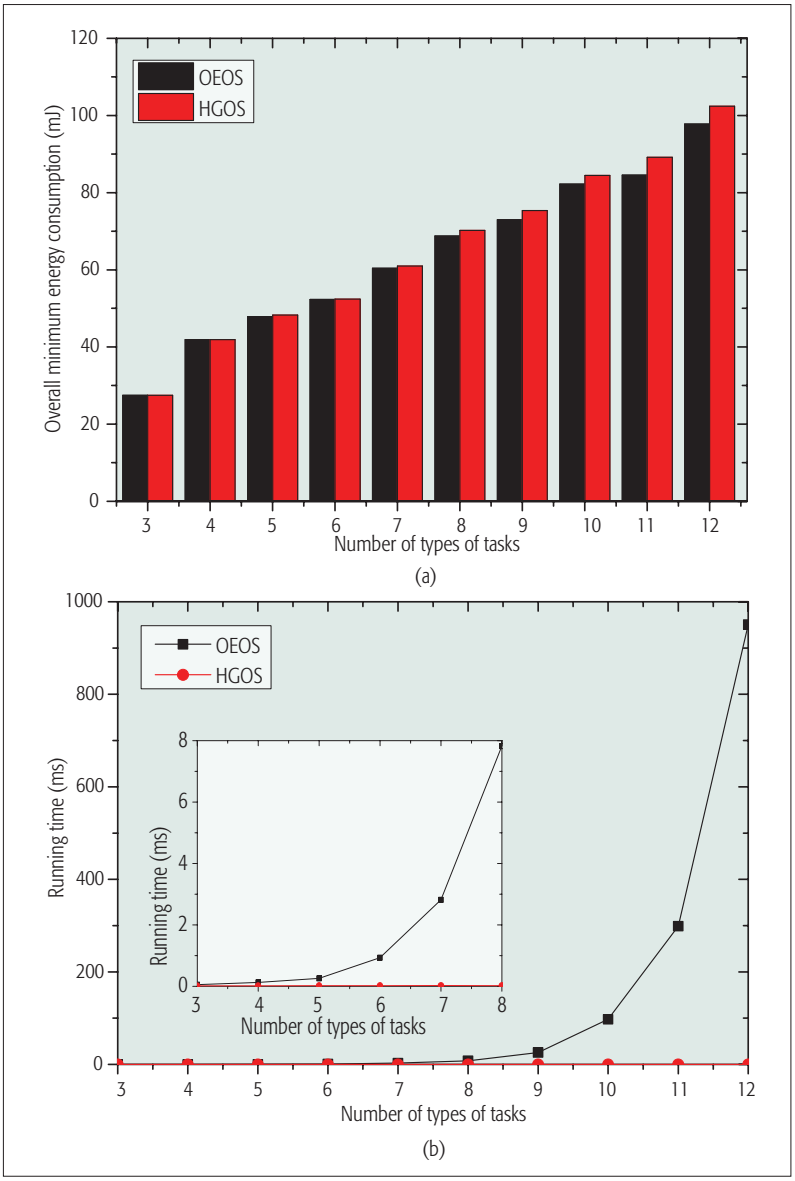

Figure 3. Illustration of the overall minimum energy consumption and the running time with different number of types of computation tasks: a) comparisons of the overall minimum energy consumption between OEOS and HGOS; b) comparisons of the running time between OEOS and HGOS.

Performance Evaluation

Numerical Results

In this section, we present numerical results to demonstrate the performance of our proposed heuristic greedy computation offloading scheme for MA-MEC in UDN. Without loss of generality, we consider a simple multi-user ultra-dense MEC server scenario in UDN, which consists of an MBS, an MD, and an SC. In particular, the MBS and the SC can separately cover a range of 100 m and 50 m, and the MD is randomly placed over their coverage area. The bandwidth of the MBS and the SC is set to 40 MHz and 20 MHz, respectively, and the uplink data rate of the fiber link is 1 Gb/s. We assume that the computing capabilities of the MD, the MEC servers connected to the MBS, and the SC are 0.8 GHz, 8 GHz, and 4 GHz, respectively, and the consumed power per CPU cycle of the MD is set to 200 mJ/gigacycle. Moreover, the transmit power of the MD is 100 mW, and the background noise power is set to −100 dBm. There are totally $N = 100$ types of computation tasks required by the MD, and the probability of a type-i task being required by the MD is randomly assigned from the interval (0, 1). Besides, the data size of type-i task varies from 300 kB to 800 kB, and the number of CPU cycles required by each type of task is set between 100 megacycles and 1000 megacycles. The maximum permissible processing delay for accomplishing each type of task is randomly distributed between 100 ms and 2 s.

Figure 3 illustrates the comparison results of the overall minimum energy consumption and running time by separately adopting OEOS and HGOS schemes, with different number of types of computation tasks. From Fig. 3a, we can see that the MD can achieve almost the same overall minimum energy consumption by adopting HGOS as that by adopting our benchmark (i.e., OEOS), which gives an optimal solution to the EMTC problem. Moreover, HGOS runs much faster than OEOS, as illustrated in Fig. 3b. This group of simulations demonstrates that although HGOS can only give a near-optimal solution to the EMTC problem, it has much higher computational efficiency and can be implemented with a large number of types of computation tasks in practice.

After verifying the effectiveness and efficiency of our proposed HGOS, we further run experiments on HGOS with more types of computation tasks (i.e., $N = 10, 20, ..., 100$) and compare the achieved overall minimum energy consumption with those separately by adopting local computing at the MD and computation offloading without SCs [6]. Figure 4 shows the comparison results among them. From this figure, one can easily observe that our HGOS scheme with SCs can achieve on average 76 and 37 percent energy consumption reduction over those by adopting local computing at the MD and the computation offloading scheme without SCs being considered, respectively. This group of experimental results not only demonstrates the necessity of computation offloading for MA-MEC in UDN, but also reveals the efficiency of deploying more MEC servers and performing computation offloading among multiple MEC servers in UDN.

Discussions

In this article, we investigate the MECO problem in UDN in the case of multi-user ultra-dense MEC server. Different from previous MECO research, which only focused on multi-user single-MEC scenarios, we consider that each type of task is required by the MD with a specified probability, and discard the commonly held assumption that the number of computation tasks at the MDs is pre-determined during a computation offloading period. Moreover, during modeling the processing time of a type-*i* task, not only its execution time on the MEC servers but also the queuing time are calculated, which may be more in line with the actual situation. Nevertheless, as our first step in studying the MECO problem in UDN, only the scenario with one MD was considered in this article. For future works, it should be meaningful to study scenarios with more MDs, where wireless channel interference among them should be taken into account. Besides, moving MDs in the case of a multi-user ultra-dense MEC server should be another focus in the future.

Conclusion

In this article, we study the problem of computation offloading for MA-MEC in UDN, that is, the MECO problem in a multi-user ultra-dense MEC scenario. In particular, we first analyze the processing time and the energy consumption by separately adopting local computing at the MD and mobile edge computation offloading, and then present our problem definition. After that, we propose a heuristic greedy offloading scheme as our solution. Numerical results confirm that our proposed scheme, which performs computation offloading among multiple MEC servers, can reduce the overall energy consumption of the MD significantly in UDN, and this trend scales well as the number of types of tasks increases.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61771374, 61771373, and 61601357), in part by the China 111 Project (B16037), and in part by the Fundamental Research Fund for the Central Universities (JB171501, JB181506, JB181507, and JB181508).

References

[1] D. Zhang *et al.*, "Capacity Analysis of NOMA with mmWave Massive MIMO Systems," *IEEE JSAC*, vol. 35, no. 7, July 2017, pp. 1606–18.
[2] L. P. Qian *et al.*, "Dynamic Cell Association for Non-Orthogonal Multiple-Access V2S Networks," *IEEE JSAC*, vol. 35, no. 10, Oct. 2017, pp. 2342–56.
[3] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-Dense Networks: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 4, 2016, pp. 2522–45.
[4] S. Wang *et al.*, "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," *IEEE Access*, vol. 5, Mar. 2017, pp. 6757–79.
[5] K. Zhang *et al.*, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, vol. 4, Aug. 2016, pp. 5896–907.
[6] J. Zheng *et al.*, "Stochastic Computation Offloading Game for Mobile Cloud Computing," *ICCC*, 2016, pp. 1–6.
[7] N. Bhushan *et al.*, "Network Densification: The Dominant Theme for Wireless Evolution into 5G," *IEEE Commun. Mag.*, vol. 52, no. 2, Feb. 2014, pp. 82–89.
[8] J. Liu *et al.*, "New Perspectives on Future Smart FiWi Networks: Scalability, Reliability, and Energy Efficiency," *IEEE Commun. Surveys & Tutorials*, vol. 18, no. 2, 2016, pp. 1045–72.

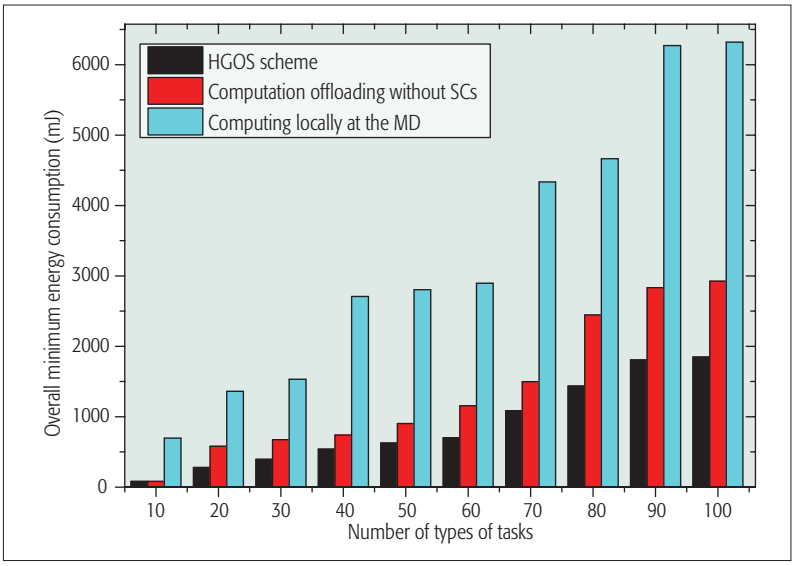

Figure 4. Comparisons of the overall minimum energy consumption by adopting different computation offloading schemes, i.e., computing locally at the MD, computation offloading without SCs, and our HGOS scheme.

[9] Z. M. Fadlullah *et al.*, "Cooperative QoS Control Scheme Based on Scheduling Information in FiWi Access Network," *IEEE Trans. Emerging Topics in Computing*, vol. 1, no. 2, Dec. 2013, pp. 375–83.
[10] K. Saito *et al.*, "A MPCP-Based Centralized Rate Control Method for Mobile Stations in FiWi Access Networks," *IEEE Wireless Commun. Letters*, vol. 4, no. 2, Apr. 2015, pp. 205–08.
[11] J. Liu *et al.*, "Device-to-Device Communication in LTE-Advanced Networks: A Survey," *IEEE Commun. Surveys & Tutorials*, vol. 17, no. 4, 4th Quarter 2015, pp. 1923–40.
[12] J. Liu *et al.*, "Device-to-Device Communications for Enhancing Quality of Experience in Software Defined Multi-tier LTE-A Networks," *IEEE Network*, vol. 29, no. 4, July 2015, pp. 46–52.
[13] E. Cuervo *et al.*, "MAUI: Making Smartphones Last Longer with Code Offload," *MobiSys*, 2010, pp. 49–62.
[14] B.-G. Chun *et al.*, "CloneCloud: Elastic Execution Between Mobile Device and Cloud," *EuroSys*, 2011, pp. 301–14.
[15] K.-H. Loh, B. Golden, and E. Wasil, "Solving the Maximum Cardinality Bin Packing Problem with a Weight Annealing-Based Algorithm," *Operations Research and Cyber-Infrastructure*, vol. 47, 2009, pp. 147–64.

Biographies

Hongzhi Guo [S'08, M'16] (hzguo@xidian.edu.cn) received his B.S. degree in computer science and technology from Harbin Institute of Technology in 2004, and his M.S. and Ph.D. degrees in computer application technology from Harbin Institute of Technology, Shenzhen, China, in 2006 and 2011, respectively. He is currently a lecturer with the School of Cyber Engineering, Xidian University. His research interests cover a wide range of areas including mobile edge computing, fiber-wireless networks, IoT, 5G, and smart grid.

Jiajia Liu [S'11, M'12, SM'15] (liujiajia@xidian.edu.cn) is currently a full professor at the School of Cyber Engineering, Xidian University. His research interests cover wireless mobile communications, FiWi, IoT, and so on. He has published more than 90 peer-reviewed papers in many prestigious IEEE journals and conferences, and currently serves as an Associate Editor for *IEEE Transactions on Communications* and *IEEE Transactions on Vehicular Technology*, an Editor for *IEEE Network*, and a Guest Editor for *IEEE TETC* and the *IEEE IoT Journal*. He is a Distinguished Lecturer of IEEE ComSoc.

Jie Zhang [S'18] received his B.S. degree in electronics science and technology from Xidian University in 2017. He is currently a master student at the School of Cyber Engineering, Xidian University. His research interests cover mobile edge computing and IoT security.