2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovations

QoE Aware and Cell Capacity Enhanced Computation Offloading for Multi-Server Mobile Edge Computing Systems with Energy Harvesting Devices

Hailiang Zhao
College of Computer Science and Technology
Wuhan University of Technology
Wuhan, China
hliangzhao97@gmail.com

Wei Du†
College of Computer Science and Technology
Wuhan University of Technology
Wuhan, China
whutduwei@whut.edu.cn

Wei Liu
College of Computer Science and Technology
Wuhan University of Technology
Wuhan, China
wliu@whut.edu.cn

Tao Lei
College of Computer Science and Technology
Wuhan University of Technology
Wuhan, China
leitao1995@whut.edu.cn

Qiwang Lei
College of Computer Science and Technology
Wuhan University of Technology
Wuhan, China
leiqiwang@whut.edu.cn

***Abstract*—The increasing complexity of intelligent services requires new paradigm to overcome the problems caused by resource-limited mobile devices. Mobile edge computing system with energy harvesting devices is such a promising technology. By offloading the computation tasks to the MEC servers, users could experience services with low latency. In addition, energy harvesting technology releases the tension between high energy consumption of intelligent services and capacity-constrained mobile device batteries. However, in multi-user and multi-server scenarios where mobile devices can move arbitrarily, computation offloading strategies are faced with new challenges because of resource competition and server selection. In this paper, we develop an intelligent computation offloading strategy. An online algorithm, which is based on Lyapunov Optimization-based Dynamic Computation Offloading algorithm, is proposed. By choosing the execution mode among local execution, offloading execution and task dropping for each mobile device, our algorithm can asymptotically obtain the optimal results for the whole system. The algorithm not only inherits every advantage from the LODCO Algorithm but also adapts perfectly to the more complex environment. Simulation results illustrate that the algorithm can improve the ratio of offloading computation tasks by more than 10% while the QoE is guaranteed.**

Keywords—intelligent services, mobile edge computing, energy harvesting, computation offloading, genetic algorithm, greedy policy.

I. Introduction

The rapid development of wireless communication technology and the increasing popularity of smart mobile devices are paving the way towards more intelligent services, such as virtual reality, augmented reality and face recognition. However, those services are usually computational-intensive, latency-sensitive and energy-consuming, which could not be supported by mobile devices with limited computational resources and energy provided by batteries [1]. Therefore, new paradigms are needed to carter to the requirements of intelligent services. Mobile edge computing (MEC) systems with energy harvesting (EH) devices is such a promising technology that IT services and cloud computing capability are provided within the radio access network [2]. By offloading the computation tasks from the mobile devices to the MEC servers, users could experience services with low latency [3]. Furthermore, energy consumption becomes not such an urgent problem because the mobile devices could be powered by free recycling energy sources [4].

The design of efficient computation offloading strategies has attracted research interest in recent years. Most works focus on single-user and single-server MEC systems with energy harvesting devices [5]. Nevertheless, for a system with multiple users and multiple servers, which are more typical scenarios in the real world [8], those strategies will not be applicable directly because of new challenges. How to allocate limited resources among mobile devices should be investigated. Besides, how to choose a proper server according to the system optimization metrics or the user's preference has to be studied. Additionally, as is known, the infrastructure providers will earn more profit with larger cell capacity for each base station [9], which is quantified by the ratio of computation tasks offloaded in this paper. On the other hand, more offloading means fiercer resource contention and thus worse quality of user experience (QoE). How to achieve the tradeoff between cell capacity enhancement and QoE guarantee will be a realistic problem. In this paper, we design an intelligent computation offloading strategy for multi-user and multi-server MEC systems with EH devices. Our major contributions are summarized as follows:

- We consider a general MEC system with multiple EH mobile devices and multiple resource-constrained MEC servers where every mobile device can move arbitrarily.
- User mobility and its effect on resources competition and server selection are embodied in the system model,

† Corresponding author.

This paper is supported by the Ministry of Education Project of Humanities and Social Sciences (Project Name, Research on fuzzy QoE aware and trustworthy cloud services selection in mobile commerce, Project No. 16YJCZH014), the Natural Science Foundation of Hubei Province of China (Project Name, Research on the replica-aware data acquiring mechanism for mobile cloud computing, 2016CFB466), and the Fundamental Research Funds for the Central Universities (WUT: 2016III028).

which is formulated as a non-convex optimization problem finally.

- The quality of user experience cost (i.e. execution latency and penalty for dropped tasks) and the cell capacity in terms of the ratio of offloading computation tasks are optimized simultaneously.
- The proposed algorithm can deal with the correlation between any two mobile devices when choosing the computation modes, especially the offloading execution, which is out of the scope considered by the LODCO Algorithm.
- The proposed algorithm can obtain the optimal results after several iterations. By comparison with the LODCO Algorithm, our algorithm not only keeps perfectly the advantages of the LODCO Algorithm but also promotes notably the ratio of offloading computation tasks while guaranteeing the QoE.

The organization of this paper is as follows. We survey state of the art in Section II. In section III, the system model will be introduced. In section IV, the LODCO-Based Genetic Algorithm with Greedy Policy will be proposed based on the formulated problem. Simulation results and Conclusion of this paper will be demonstrated in Section V and Section VI, respectively.

II. Related Works

Computation offloading for multi-user and multi-server mobile systems is a very challenging problem because of complexity of the scenario and interdependence among users and servers. A few strategies have proposed in recent years. In [10], the power-delay tradeoff in the context of task offloading was studied. In [11], the power minimization for the mobile devices by data offloading was investigated. In [12], the problem of joint task offloading and resource allocation was formulated as a mix integer non-linear program. The task offloading decision, uplink transmission power of mobile users and computational resource allocation at the MEC servers were jointly optimized. However, the optimization objective of MEC systems with EH devices is shifted from minimizing the battery energy consumption as the harvested energy is ample and free. As a result, those computation offloading strategies dedicated to the energy conservation cannot be utilized without modification.

In [13], a device-edge-cloud MEC system was investigated. A network aware multi-user and multi-edge computation partitioning problem was formulated. Computation and radio resources were allocated such that the average throughput of the users was maximized. The system considered in [13] is similar to this paper. Nevertheless, a partial offloading model was utilized in [13] while a binary offloading is exploited in our work.

There are works dedicated on the computation offloading for MEC systems with EH devices [5]. In [6], a "hotbooting" Q-learning based computation offloading scheme for IoT devices with energy harvesting to achieve the optimal offloading was presented. Also, a fast DQN-based IoT computation offloading scheme to compass the state space dimension was proposed. However, similar to [13], a partial offloading model was discussed in [6]. In [7], the problem of solving an optimal computation offloading policy was modeled as a Markov decision process, where the objective was to minimize the long-term cost. A deep Q-network-based strategic computation offloading algorithm was designed. Nevertheless, the system consisted of only one telecom cloud, which was connected to multiple base stations. Obviously, both the system and the problem considered in [7] are different from that considered in our work. In [5], a green MEC system with EH devices was analyzed, and an effective computation offloading strategy was developed. Moreover, the execution cost, which includes both the execution latency and task failure, was adopted as the performance metric. Finally, a low-complexity online algorithm was proposed.

Although [5] is the most similar work to ours, several key differences should be addressed. Firstly, while the system discussed in [5] focused on a single user and single MEC server system, we dedicate on a multi-user and multi-server scenario. Secondly, user mobility, which was ignored in [5], is considered in this paper. Thirdly, the MEC server in this paper has limited computation capability, which generalizes the work in [5], where the MEC server was assumed to be with unlimited computational resources. Fourthly, compared to the single-objective optimization problem formulated in [5], we are interested in optimizing two objectives simultaneously. Due to the differences mentioned above, our work is more complicated and difficult than that in [5].

III. System Model

A. System Description

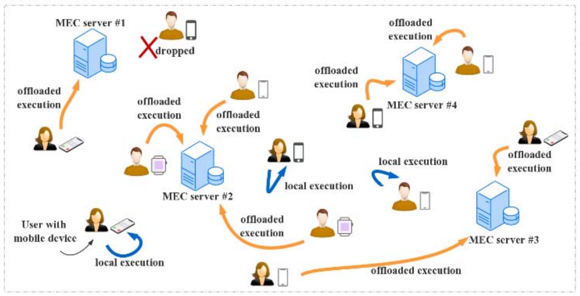


Fig. 1. The system with multiple mobile devices and multiple MEC servers.

We consider a MEC system consisting of N mobile devices equipped with EH components and M MEC servers. We assume that time is slotted. The time slot length and the time slot index set are denoted by τ and $\mathcal{T} \triangleq \{0,1,\ldots\}$, respectively. We use $\mathcal{N} \triangleq \{1,2,\ldots,N\}$ and $\mathcal{M} \triangleq \{1,2,\ldots,M\}$ to denote the set of mobile devices and MEC servers, respectively. As shown in Fig. 1, all elements are limited to a specific area, where servers are located at a particular position while each mobile device's location varies among different time slots. Denote the distance between the ith mobile device and the jth MEC server at the tth time slot as $d_{i,j}^t$, and $d_{i,j}^t \sim U(0,d)$, $t \in \mathcal{T}$, $i \in \mathcal{N}$, $j \epsilon \mathcal{M}$. We ensure that each mobile device does not out of the region.

B. Computation Task Model

We focus on delay-sensitive computation tasks with the execution deadline *no greater than* the length of each time slot [10]. Denote the computation task generated by the ith mobile device at the tth time slot as CT_i^t, which has a *fixed* size L (in bits). We assume the computation tasks are modeled as an i.i.d. Bernoulli process [5], i.e., computation task CT_i^t is requested with probability ρ where $0 < \rho < 1$. Let $\zeta_i^t = 1$ if the ith mobile device gets computation task request at the tth time slot.

In our system, there exists no *task buffers* either in mobile devices or MEC servers. Besides, if the energy of the ith mobile device is insufficient or $\zeta_i^t = 0$, the computation task at the tth time slot will be dropped. We denote the computation mode indicators for the ith mobile device at the tth time slot as $I_{i,c}^t \in \{0,1\}$ with $c = \{l, r, d\}$, where $I_{i,l}^t = 1$, $I_{i,r}^t = 1$ and $I_{i,d}^t = 1$ indicate that the computation task is executed locally, remotely and dropped, respectively. Apparently we have

$$I_{i,l}^t + I_{i,r}^t + I_{i,d}^t = 1, t \in \mathcal{T}, i \in \mathcal{N}. \quad (1)$$

C. Offloading Computation Model

Denote the indicator of the ith mobile device choosing the jth MEC server to offload as $c_{i,j}^t = 1$, where $c_{i,j}^t \in \{0,1\}$. We assume that each computation task is assigned to only one server when choosing the offloading computation mode, i.e.,

$$\sum_{j=1}^{M} c_{i,j}^t = 1, t \in \mathcal{T}, i \in \mathcal{N}, j \epsilon \mathcal{M}. \quad (2)$$

Denote the small-scale fading channel power gain $\gamma_{i,j}^t$. According to *communication theory*, the channel power gain is calculated by $h_{i,j}^t = \gamma_{i,j}^t g_0 (d_0 / d_{i,j}^t)^\theta$, where d_0 is the reference distance, θ is the path-loss exponent and g_0 is the path-loss constant. Thus, the achievable rate can be obtained according to *Shannon Theorem* by

$$\Gamma\left(h_{i,j}^t, p_i^t\right) = \omega \log_2\left(1 + h_{i,j}^t p_i^t / \sigma\right), t \in \mathcal{T}, i \in \mathcal{N}, j \epsilon \mathcal{M}, \quad (3)$$

where ω is the bandwidth allocated, σ is the noise power at each MEC server and p_i^t is the transmit power who cannot exceed the maximum value $p^{\max}$. Note that the bandwidth of each MEC server is equally divided into N sub-bands, and each mobile device is assigned to one sub-band with ω MHz. Denote the CPU cycles frequency required to process one bit task by each mobile device and each MEC server as X_{m} and X_{s}, respectively. We assume that the computational abilities of MEC servers are constrained, i.e.,

$$\sum_{i \in \mathcal{N}} \mathrm{I}\left(I_{i,r}^t\right) \cdot c_{i,j}^t L X_{\mathrm{s}} \le f_{\mathrm{s}}^{\max} \tau, t \in \mathcal{T}, j \epsilon \mathcal{M}, \quad (4)$$

where $f_{\mathrm{s}}^{\max}$ denotes the upper bound of each MEC server's CPU-cycle frequency and $\mathrm{I}(\cdot)$ is the indicator function. We take no consideration of execution latency consumed by the MEC server's execution process, i.e., for the ith mobile device, the total execution latency of this mode equals to *the transmission delay* for the input task. Thus,

$$D_{i,\mathrm{r}}^t = \frac{L}{\Gamma\left(h_{i,j}^t, p_i^t\right)}, t \in \mathcal{T}, i \in \mathcal{N}, j \epsilon \mathcal{M}. \quad (5)$$

Therefore, the corresponding consumed energy is

$$E_{i,\mathrm{r}}^t = p_i^t \frac{L}{\Gamma\left(h_{i,j}^t, p_i^t\right)}, t \in \mathcal{T}, i \in \mathcal{N}, j \epsilon \mathcal{M}. \quad (6)$$

D. Local Computation Model

In order to execute L bits computation task successfully, LX_{m} CPU cycles are required. By applying the dynamic voltage and frequency scaling technologies (DVFS) [14], the total execution latency of this mode can be obtained by

$$D_{i,\mathrm{l}}^t = \sum_{v=1}^{LX_{\mathrm{m}}} (f_{i,v}^t)^{-1}, t \in \mathcal{T}, i \in \mathcal{N}. \quad (7)$$

According to [5], the energy consumed can obtained by

$$E_{i,\mathrm{l}}^t = \sum_{v=1}^{LX_{\mathrm{m}}} \mathscr{s} (f_{i,v}^t)^2, t \in \mathcal{T}, i \in \mathcal{N}, \quad (8)$$

where $\mathscr{s}$ is the effective capacitance coefficient. Moreover, we denote the upper bound of each mobile device's CPU-cycle frequency as $f_{\mathrm{m}}^{\max}$, i.e., $\forall v \in \{1,2,\dots,LX_{\mathrm{m}}\}$, $f_{i,v}^t \le f_{\mathrm{m}}^{\max}, t \in \mathcal{T}, i \in \mathcal{N}$.

E. Energy Harvesting Model

In order to embody the stochastic and intermitted nature of the renewable energy process [15], we assume that the harvestable energy $E_{i,H}^t$ is uniformly distributed with the maximum value of $E_H^{\max}$. The system needs to decide the amount of energy which will be stored in the battery of the ith mobile device. Denote this part of energy as e_i^t, then we have

$$0 \le e_i^t \le E_{i,H}^t, t \in \mathcal{T}, i \in \mathcal{N}. \quad (9)$$

We assume that other kinds of energy consumption besides local-execution and remote-execution is *sufficient small*. We use $\mathcal{E}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t)$ to denote the energy consumed, where $\boldsymbol{I}_i^t \triangleq \left[I_{i,l}^t, I_{i,r}^t, I_{i,d}^t\right]$ and $\boldsymbol{f}_i^t \triangleq [f_{i,1}^t, \dots, f_{i,LX_{\mathrm{m}}}^t]$. Then we can obtain it by the following equation:

$$\mathcal{E}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) = I_{i,l}^t E_{i,\mathrm{l}}^t + I_{i,r}^t E_{i,\mathrm{r}}^t, t \in \mathcal{T}, i \in \mathcal{N}. \quad (10)$$

Denote the battery level of the ith mobile device at the tth time slot as b_i^t. Obviously, the energy consumption at each time slot cannot surpass the battery level, i.e.,

$$\mathcal{E}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) \le b_i^t, t \in \mathcal{T}, i \in \mathcal{N}. \quad (11)$$

Besides, b_i^t evolves according to the following equation:

$$b_i^{t+1} = b_i^t - \mathcal{E}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) + e_i^t, t \in \mathcal{T}, i \in \mathcal{N}. \quad (12)$$

F. QoE-Cost Function

User's QoE [16] consists of *execution delay* and *the penalty for dropping the task*. We can obtain the system QoE-cost by

$$\mathrm{cost}_{\mathrm{sum}}^t \triangleq \sum_{i \in \mathcal{N}} \mathrm{cost}_i^t$$
$$= \sum_{i \in \mathcal{N}} \left[\mathcal{D}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) + \emptyset \cdot \mathrm{I}\left(\zeta_i^t \cap I_{i,d}^t\right)\right], t \in \mathcal{T}, i \in \mathcal{N}, \quad (13)$$

where $\emptyset$ denotes the weight of task dropping cost, and $\mathcal{D}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t)$ denotes the execution delay, which is given by

$$\mathcal{D}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) = \mathrm{I}(\zeta_i^t) \cdot (I_{i,l}^t D_{i,\mathrm{l}}^t + I_{i,r}^t D_{i,\mathrm{r}}^t). \quad (14)$$

G. Optimization Model

In our system, the more computation tasks executed remotely, the better user's quality of experience. We can obtain *the number of offloading computation tasks* at tth time slot by

$$\sum_{i \in \mathcal{N}} \mathrm{I}\left(\zeta_i^t \cap I_{i,r}^t\right), t \in \mathcal{T}, \quad (15)$$

where $\mathrm{I}(\cdot)$ is the indicator function. Thus, we focus on two optimization goals, i.e., the average weighted sum QoE-cost minimization and the number of offloading computation tasks maximization. Denote the operation at the tth time slot as

$$\boldsymbol{SO}^t \triangleq [\boldsymbol{I}^t, \boldsymbol{f}^t, \boldsymbol{p}^t, C^t, \boldsymbol{e}^t], t \in \mathcal{T}, \tag{16}$$

where $\boldsymbol{I}^t \triangleq [I_{1,c}^t, \dots, I_{N,c}^t]$, and $\boldsymbol{f}^t$, $\boldsymbol{p}^t$, $\boldsymbol{e}^t$ are defined in the same way. Consequently, the optimization problem is

$$\mathcal{P}_1: \min_{\boldsymbol{SO}^t} \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathrm{cost}_{\mathrm{sum}}^t - \psi \cdot \sum_{i\in\mathcal{N}} \mathrm{I}\left(\zeta_i^t \cap I_{i,r}^t\right)$$

$$s.t. \text{ (1), (2), (4), (9), (10), (11)}$$

$$0 \le f_{i,v}^t \le f_{\mathrm{m}}^{\max}, t \in \mathcal{T}, i \in \mathcal{N}, v \in \{1, \dots, LX_{\mathrm{m}}\} \tag{17}$$

$$\mathcal{E}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) \le E_{\max}, t \in \mathcal{T}, i \in \mathcal{N} \tag{18}$$

$$0 \le p_i^t \le p^{\max}, t \in \mathcal{T}, i \in \mathcal{N} \tag{19}$$

$$I_{i,c}^t \in \{0,1\}, c \in \{l, r, d\}, t \in \mathcal{T}, \tag{20}$$

where ψ denotes the weight of the second optimization goal. (17) and (19) incarnate the constraints of mobile devices' CPU-cycle frequency and maximum transmit power, respectively. (18) incarnates the upper bound of battery discharging for security reasons, i.e., the amount of energy output energy cannot exceed $E_{\max}$ at each time slot. (20) represents the 0-1 indicator constraint which has been described in subsection B of Section III.

IV. A Low-Complexity Algorithm for Joint QoE-Cost and Cell-Capacity Optimization

In this section, we develop the LODCO-Based Genetic Algorithm to solve $\mathcal{P}_1$ on account of the LODCO algorithm [5]. First, we will convert the original problem which is time-dependent to a per-time slot deterministic problem $\mathcal{P}_2$ by Lyapunov Optimization. Then the LODCO Algorithm will be *upgraded and reconstructed* for the multi-user and multi-server MEC system by virtue of Genetic Algorithm and Greedy Strategy. We will not demonstrate the details of the LODCO Algorithm, but we will explain why it can be utilized.

A. Drift Plus Penalty Formula

Lyapunov optimization demands that the allowable action sets are i.i.d., which cannot be satisfied by the time-dependent battery queues of mobile devices modeled in this paper. Thus, for each mobile device we use the perturbation parameter θ [5](which is lower bounded by $\tilde{E}_{\max} + V\emptyset$) to define virtual battery queue $\tilde{b}_i^t$ by

$$\tilde{b}_i^t \triangleq b_i^t - \theta, t \in \mathcal{T}, i \in \mathcal{N}, \tag{21}$$

where $\tilde{E}_{\max} \triangleq \min\{\max\{\kappa LX_{\mathrm{m}}(f_{\mathrm{m}}^{\max})^2, p^{\max}\tau\}, E_{\max}\}$.

Besides, if a task requested at the tth time slot is being executed locally, the optimal frequencies of the LX_{m} CPU cycles should be the same, i.e., $f_{i,v}^t = f_i^t, i \in \{1, \dots, LX_{\mathrm{m}}\}$, which can be obtained by *Inequality of arithmetic and geometric means.*

According to the analysis above, we define the Lyapunov function as

$$L(t) \triangleq \frac{1}{2}\sum_{i\in\mathcal{N}}\left(\tilde{b}_i^t\right)^2 = \frac{1}{2}\sum_{i\in\mathcal{N}}(b_i^t - \theta)^2, t \in \mathcal{T}. \tag{22}$$

Thus, the conditional Lyapunov drift can be written as

$$\Delta(t) \triangleq \mathbb{E}\left[L(t+1) - L(t)\middle|\tilde{b}^t\right], t \in \mathcal{T}, \tag{23}$$

where $\tilde{b}^t \triangleq [\tilde{b}_1^t, \dots, \tilde{b}_N^t]$. Then the Lyapunov drift-plus-penalty function can be written as

$$\Delta_V(t) \triangleq \Delta(t) + V \cdot \mathbb{E}\left[\mathrm{cost}_{\mathrm{sum}}^t\middle|\tilde{b}^t\right], t \in \mathcal{T}. \tag{24}$$

Because of (12), we can obtain that

$$\left(\tilde{b}_{i+1}^t\right)^2 \le \left(\tilde{b}_i^t\right)^2 + 2\tilde{b}_i^t\left(e_i^t - \mathcal{E}(\boldsymbol{I}_i^t, f_i^t, p_i^t)\right)$$
$$+(e_i^t)^2 + \mathcal{E}^2(\boldsymbol{I}_i^t, f_i^t, p_i^t), t \in \mathcal{T}, i \in \mathcal{N}. \tag{25}$$

Denotes the upper bound of the real energy consumed by the ith mobile device at the tth time slot as $\tilde{E}_{\max}$, we have $\mathcal{E}(\boldsymbol{I}_i^t, f_i^t, p_i^t) \le \tilde{E}_{\max}$. Because $e_i^t \le E_{i,H}^t$ and $E_{i,H}^t$ are i.i.d. among different time slots with the maximum value of $E_H^{\max}$ for each mobile device, we can get $e_i^t \le E_H^{\max}$. As a result, we have

$$\Delta(t) \le \sum_{i\in\mathcal{N}} \mathbb{E}\left[\tilde{b}_i^t\left(e_i^t - \mathcal{E}(\boldsymbol{I}_i^t, f_i^t, p_i^t)\right)\middle|\tilde{b}^t\right] + C, t \in \mathcal{T}, \tag{26}$$

where $C = \frac{N}{2}(\tilde{E}_{\max} + E_H^{\max})^2$. Combing with (24), we have

$$\Delta_V(t) \le \sum_{i\in\mathcal{N}} \tilde{b}_i^t\left(e_i^t - \mathcal{E}(\boldsymbol{I}_i^t, f_i^t, p_i^t)\right) +$$
$$V \cdot \mathbb{E}\left[\mathrm{cost}_{\mathrm{sum}}^t\middle|\tilde{b}^t\right] + C, t \in \mathcal{T}, \tag{27}$$

which means $\Delta_V(t)$ is upper bounded.

Obviously, the upper bound of $\Delta_V(t)$ has *the same structure* with the problem defined in [5]. Thus, the LODCO Algorithm can hopefully be applied to decide $\boldsymbol{SO}^t$ by solving the following deterministic problem

$$\mathcal{P}_2: \min_{\boldsymbol{SO}^t} \sum_{i\in\mathcal{N}} \tilde{b}_i^t\left(e_i^t - \mathcal{E}(\boldsymbol{I}_i^t, f_i^t, p_i^t)\right) + V \cdot$$
$$\mathbb{E}\left[\mathrm{cost}_{\mathrm{sum}}^t\middle|\tilde{b}^t\right] \in \mathcal{T}, t \in \mathcal{T}, i \in \mathcal{N},$$

which subjects to every constraint of $\mathcal{P}_1$. In the above deterministic problem $\mathcal{P}_2$, we temporarily ignore the second optimal goal because it is not relative to the stability and reliability of the system. Besides, it can be added to $\mathcal{P}_2$ directly. We will describe the process in detail in subsection C.

B. Application of the LODCO Algorithm

$\mathcal{P}_2$ is an optimized format by Lyapunov Optimization with maximizing the number of offloading computation tasks temporary omitted. According to the LODCO Algorithm, the problem can be decomposed to two sub problems.

Optimal energy harvesting. the optimal amount of harvested energy e_i^{t*} for the ith mobile device can be obtained by solving the following problem:

$$\min_{0\le e_i^t \le E_{i,H}^t, i\in\mathcal{N}, t\in T} \sum_{i\in\mathcal{N}} \tilde{b}_i^t e_i^t.$$

Because each mobile device's decision on optimal energy harvesting is mutually independent, the optimal e_i^{t*} can be obtained separately for each mobile device. Thus, we obtain e_i^{t*} for each mobile device by (21) in [5].

Decide the computation modes. we need to obtain the mode with the minimum value of $J_{CO}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t)$ for each mobile device, where

$$J_{CO}(\boldsymbol{I}_i^t, \boldsymbol{f}_i^t, p_i^t) \triangleq \mathrm{I}\left(I_{i,l}^t\right) \cdot J_{\mathrm{m}}^t(f_i^t) + \mathrm{I}\left(I_{i,r}^t\right) \cdot J_{\mathrm{server}}^t(p_i^t)$$

$$+\mathrm{I}\left(I_{i,d}^t\right) \cdot V\emptyset, t \in \mathcal{T}, i \in \mathcal{N}, \tag{28}$$

where $J_{\mathrm{m}}^t(f_i^t)$ and $J_s^t(p_i^t)$ denote the optimal goals for $\mathcal{P}_{\mathrm{ME}}$ and $\mathcal{P}_{\mathrm{SE}}$[1], which are sub-problems for local-execution mode and remote-execution mode, respectively.

There exists correlation between any two mobile devices when choosing the offloading computation mode. As a result, the LODCO Algorithm cannot be utilized directly on our problem. In order to solve the problem, we propose the LODCO-Based Genetic Algorithm with Greedy Policy for the multi-user and multi-server system. At the tth time slot, the *genetic algorithm* will be used to obtain the minimum value of $\mathcal{P}_2$ when each mobile device's battery energy level is sufficient for local execution or offloaded execution. Otherwise, *the key-value pair method* will be used by virtue of ϵ-greedy policy [17].

C. The LODCO-Based Genetic Algorithm with Greedy Policy

In this section, we demonstrate the details of proposed algorithm.

Determine the decision variables. There will be much redundancy if we use the system operation $\boldsymbol{SO}^t$ as the decision variables directly, which can lead to slow convergence of genetic algorithm. To overcome the above problem, we define the decision vector $\boldsymbol{x}^t$ for the whole system as

$$\boldsymbol{x}^t \triangleq [\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \dots, \boldsymbol{x}_N^t], t \in \mathcal{T}, \tag{29}$$

where

$$\boldsymbol{x}_i^t \triangleq [I_{i,l}^t, I_{i,d}^t, c_{i,1}^t, c_{i,2}^t, \dots, c_{i,M}^t], t \in \mathcal{T}, i \in \mathcal{N} \tag{30}$$

is the decision vector for the ith mobile device. Each element in $\boldsymbol{x}^t$ is either 0 or 1, i.e.,

$$x_k^t \in \{0,1\}, t \in \mathcal{T}, k\epsilon\{1,2,\dots,|\boldsymbol{x}^t|\}. \tag{31}$$

Compared with system operation $\boldsymbol{SO}^t$, $\boldsymbol{I}^t$ is decomposed to $I_{i,l}^t$ and $I_{i,d}^t$ in $\boldsymbol{x}^t$, and then $I_{i,r}^t$ is replaced to $[c_{i,1}^t, c_{i,2}^t, \dots, c_{i,M}^t]$.. We ignore $\boldsymbol{f}^t$, $\boldsymbol{p}^t$ and $\boldsymbol{e}^t$ in $\boldsymbol{SO}^t$ because all of them can be obtained by the LODCO Algorithm separately.

Modify the constraints. The computational abilities of MEC servers are constrained in (4), which can be written as

$$S_{\mathrm{UB}} \cdot LX_{\mathrm{s}} \le f_{\mathrm{s}}^{\max}\tau, t \in \mathcal{T}, \tag{32}$$

where S_{UB} is the maximum number of mobile devices that could connect to a MEC server simultaneously. Having considered the fact that S_{UB} must be an integer, we can obtain it by

$$S_{\mathrm{UB}} \le \lfloor f_{\mathrm{s}}^{\max}\tau / LX_{\mathrm{s}} \rfloor. \tag{33}$$

Thus, we can rewrite (4) in the following way:

$$\sum_{i=1}^N x_{k+(i-1)\times(M+1)}^t \le S_{\mathrm{UB}}, t \in \mathcal{T}, k\epsilon\{3,4,\dots,M+2\}. \tag{34}$$

By combining (1) with (2), we have the following constraint:

$$\sum_{k=1}^{M+2} x_{k+(i-1)\times(M+1)}^t \le 1, t \in \mathcal{T}, i \in \mathcal{N}, \tag{35}$$

which means each mobile device can choose only one mode.

Modify the optimization goals. We formulate $\mathcal{P}_3$ by adding the second optimal goal to $\mathcal{P}_2$, i.e.,

$$\begin{aligned}\mathcal{P}_3:\ \min_{x^t} & \sum_{i=1}^N \tilde{b}_i^t\left(e_i^t - x_{1+(i-1)\times(M+1)}^t \cdot E_{i,\mathrm{local}}^{t*} - \sum_{k=3}^{M+2} x_{k+(i-1)\times(M+1)}^t \cdot E_{i,j}^{t*}\right) \\ & + \sum_{i=1}^N\left(x_{1+(i-1)\times(M+1)}^t \cdot D_{i,\mathrm{local}}^{t*} + \sum_{k=3}^{M+2} x_{k+(i-1)\times(M+1)}^t \cdot D_{i,j}^{t*} + \emptyset \cdot x_{2+(i-1)\times(M+1)}^t\right) \\ & + \psi \cdot \sum_{i=1}^N \sum_{k=3}^{M+2} x_{k+(i-1)\times(M+1)}^t, t \in \mathcal{T},\end{aligned}$$

where $E_{i,\mathrm{local}}^{t*}$ and $D_{i,\mathrm{local}}^{t*}$ are the optimal energy consumption and execution delay of local execution for the ith mobile device separately, $E_{i,j}^{t*}$ and $D_{i,j}^{t*}$ are the optimal energy consumption and execution delay when the ith mobile device offloads its tasks to the jth MEC server, respectively.

According to the LODCO Algorithm, at the very beginning of the iteration, the computation task has to be dropped because of the insufficient battery energy, which means there exists no optimal energy consumption or execution delay, i.e., $\mathcal{P}_3$ will not work. Besides, if the sub-problems, i.e., $\mathcal{P}_{\mathrm{ME}}$ and $\mathcal{P}_{\mathrm{SE}}$, are infeasible, $\mathcal{P}_3$ will not work, too. In this case, we develop a key-value pair method by virtue of ϵ-greedy policy. Specifically, we use a *map* to store each mobile device and its chosen optimal MEC server. At each time slot, the key-value pair "i-j" with the minimum $J_s^t(p_i^t)$ will be chosen to compare with other modes for the ith mobile device. We always choose the offloading execution mode whatever $J_s^t(p_i^t)$ is the minimum among three modes with the probability of ϵ. With the probability of $1-\epsilon$, we choose the mode with the minimum value of optimal goal among $J_{\mathrm{m}}^t(f_i^t)$, $J_{\mathrm{s}}^t(p_i^t)$ and $V\emptyset$, where $J_{\mathrm{s}}^t(p_i^t)$ and $J_{\mathrm{m}}^t(f_i^t)$ are the optimization goals for $\mathcal{P}_{\mathrm{SE}}$ and $\mathcal{P}_{\mathrm{ME}}$, respectively.

Algorithm 1 LODCO-Based Genetic Algorithm with Greedy Policy

```
1: Initialize flag[$M$] with 0 and establish an empty map.
2: Initialize Boolean variable useKeyValuePair with false.
3: for each mobile device $i$ do
4:     Obtain $\zeta_i^t$, $\tilde{b}_i^t$, $E_{i,H}^t$, then generate the location of each mobile device.
5:     Obtain $e_i^{t*}$ by the LODCO Algorithm.
6:     Obtain $f_i^{t*}$, then record the optimal value $J_{\mathrm{m}}^t(f_i^t)$. If the battery energy is insufficient or $\mathcal{P}_{\mathrm{ME}}$ is infeasible, set useKeyValuePair as true.
7:     for each MEC server $j$ do
8:         Obtain $h_{i,j}^t$, $p_{i,j}^{t*}$ and then record the optimal value $J_{\mathrm{s}}^t(p_{i,j}^t)$. If the battery energy is insufficient or $\mathcal{P}_{\mathrm{SE}}$ is infeasible, set useKeyValuePair as true.
9:         Select the optimal $p_i^{t*}$ who has the minimum $J_{\mathrm{s}}^t(p_{i,j}^t)$, denote as $J_{\mathrm{s}}^t(p_i^t)$ and then record $j$.
```

[1] Both $\mathcal{P}_{\mathrm{ME}}$ and $\mathcal{P}_{\mathrm{SE}}$ are the sub-problems defined in [5], where the former one is a problem to obtain the optimal CPU-cycle frequency for a task being executed locally and the latter one is a problem to obtain the optima transmit power for computation offloading. Both of them are parts of the LODCO Algorithm.

```
10:    end for
11:    Compare $J_m^t(f_i^t)$, $J_s^t(p_i^t)$ and $V\emptyset$, choose the mode with the minimum value and set the corresponding indicator variable $I_{i,c}^t$ as 1.
12:    if $I_{i,r}^t = 1$ then
13:       Insert key $i$ and value $j$ into the map.
14:    end if
15: end for
16: if useKeyValuePair == false then
17:    Use Genetic Algorithm to solve $\mathcal{P}_3$ with constraints of (31), (34), (35).
18: else
19:    Call the Key-value Pair Method.
20: end if
21: Update $t$ to $t+1$.
```

Algorithm 1 gives the main parts of the proposed algorithm, then we will show the details of *Key-value Pair Method*.

Subroutine 1 Key-value Pair Method

```
1:  while the map is not null do
2:    Obtain "$i$-$j$" with the minimum $J_s^t(p_i^t)$ in the map.
3:    if rand() $< \epsilon$ then
4:      if flag[$j$] $\le S_{UB}$ then
5:        Remove the key-value pair "$i$-$j$" from the map and then flag[ $j$ ]++ no matter whether $J_s^t(p_i^t)$ is the minimum among $J_m^t(f_i^t)$, $J_s^t(p_i^t)$ and $V\emptyset$. Then set $J_s^t(p_{i,j}^t)$ as inf.
6:      else
7:        if min{$J_s^t(p_{i,:}^t)$} != inf then[2]
8:          Find the optimal $j$ by min{$J_s^t(p_{i,:}^t)$} and the insert them to the map. Then continue.
9:        else
10:         Select the optimal mode from local execution and task dropping. Then remove the key-value pair.
11:       end if
12:     end if
13:   else if rand()$\ge \epsilon$ then
14:     Compare $J_m^t(f_i^t)$, $J_s^t(p_i^t)$ and $V\emptyset$, choose the mode with the minimum value and set the corresponding indicator variable $I_{i,c}^t$ as 1[3].
15:     if $I_{i,r}^t = 1$ then
16:       if flag[$j$] $\le S_{UB}$ then
17:         Remove the key-value pair "$i$-$j$" from the map and flag[$j$]++. Then set $J_s^t(p_{i,j}^t)$ as inf.
18:       else
19:         if min{$J_s^t(p_{i,:}^t)$} != inf then
20:           Find the optimal $j$ by min{$J_s^t(p_{i,:}^t)$} and then insert them to the map. Then continue.
21:         else
22:           Select the optimal mode from local execution and task dropping. Then remove the corresponding key-value pair from map.
23:         end if
24:       end if
25:     else
26:       Keep the corresponding $I_{i,c}^t = 1$ then remove the corresponding key-value pair from map.
27:     end if
28:   end if
29: end while
```

V. Simulation Results

In this section, we first demonstrate the results of the proposed algorithm and verify its effectiveness. Then, we show the impact of the system parameters. As mentioned in section IV, we will not elaborate on verification of the LODCO Algorithm.

The simulation was run on a laptop with an Intel Core 2.5 GHz i7-4710MQ CPU. The algorithm was implemented in MATLAB R2015b and was given up to 8 GB of memory if needed.

In our system, the harvestable energy $E_{i,H}^t$ is uniformly distributed with the maximum value of $E_H^{\max}$, where $E_H^{\max}$ can be obtained by average EH power P_H, i.e.,

$$E_H^{\max} = P_H \cdot 2\tau, t \in \mathcal{T}. \quad (36)$$

We assume that $P_H = 12$ mW, $g_0 = -40$ dB, $\kappa = 10^{-28}$, $\tau = \emptyset = 2$ ms. In addition, $\omega = 1$ MHz, $\sigma = 10^{-13}$ W, $f_m^{\max} = f_s^{\max} = 1.5$ GHz, $E_{\max} = 2$ mJ, $L = 1000$ bits, $V = 10^{-5}$, and $X_s = X_m = 5900$ cycles per byte. Besides, we assume there are 10 mobile devices and 5 MEC servers, i.e., $N = 10$, $M = 5$, and $E_{\min} = 0.02$ mJ.

In the following part, the default value of ρ, ψ and ϵ in Algorithm 1 are 0.7, 0.002 and 0.1 unless stated, respectively. $\gamma_{i,j}^t$ is exponentially distributed with mean 1. Besides, we assume the maximum distance between random mobile and random MEC server is 80 m unless stated, which is the upper bound for the uniform distribution.

Because no work has been found to solve the same problem as we have done, we introduce one benchmark algorithm, namely, the LODCO-Based Greedy Algorithm, it works as follows: At each time slot, this algorithm always chooses the mode with the minimum target value of $\mathcal{P}_2$ between local execution and offloaded execution for each mobile device, which means it will not take the second optimization goal into consideration; otherwise, if neither one is infeasible, then the computation tasks will be dropped. Besides, the greedy policy in this algorithm has the same structure with *Key-value Pair method*.

A. Validation of Effectiveness

In this subsection, we will compare the effectiveness of the LODCO-Based Genetic Algorithm with Greedy Policy compared with the LODCO Algorithm and the LODCO-Based Greedy Algorithm. As shown in Fig. 2, the battery energy level

[2] $J_{\text{server}}^t(p_{i,:}^t)$ is defined as $[J_{\text{server}}^t(p_{i,1}^t), J_{\text{server}}^t(p_{i,2}^t), \dots, J_{\text{server}}^t(p_{i,M}^t)]$.

[3] we have to do the search again because it is possible that $J_s^t(p_i^t)$ has been modified.

of each mobile device demonstrates the feasibility of our improvement on the LODCO Algorithm. The energy level of each mobile device keeps accumulating at earlier stage, and finally stabilizes around the perturbed energy level, which exactly keeps the advantages of the LODCO Algorithm. Under current parameters, the battery energy level of each mobile device stabilizes at the beginning of the 200th time slot.

Fig. 3 demonstrates the average QoE-cost of all mobile devices at each time slot obtained by the LODCO-Based Genetic Algorithm with Greedy Policy. As we can see, the average QoE-cost declines steeply at the early stage and then stabilizes at a low level, which is the embodiment of inheriting advantages from the LODCO Algorithm. In Fig. 4, the Y-axis describes the average value of each mobile device's energy level during 500 time slots. Apparently each energy level is confined within $[0.00002, 0.005]$ J[4], which conforms to the theoretical results derived in the LODCO Algorithm.

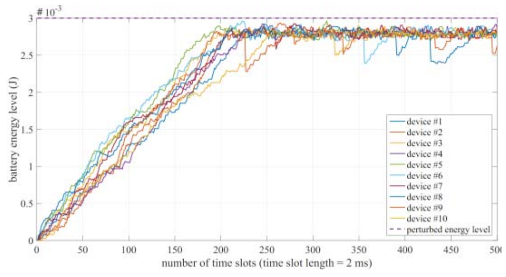


Fig. 2. Battery energy level of each mobile device vs. time.

Fig. 5 demonstrates the ratio of each chosen modes in our multi-user and multi-server system by the LODCO-Based Genetic Algorithm with Greedy Policy. At the very earlier stage, plenty of computation tasks are dropped due to the insufficient energy level. During the first 150 time slots, the average ratio of dropped tasks is 1.8942%, which significant decreases to 0 along with the ascending battery energy level of each mobile device. Meanwhile, the ratio of offloading tasks (which is 94.6291%) clearly greater than the ratio of locally-executed tasks (which is 3.4767%) and the average ratio of offloading tasks obtained by the LODCO Algorithm (which is 84.4401%), which means the LODCO-Based Genetic Algorithm with Greedy Policy can obtain better performance than the LODCO Algorithm can do in terms of maximizing the number of offloading computation tasks.

Note that this is a stochastic problem with many parameters changing at each time slot, the ratio of offloading by our algorithm may exist slightly difference after the convergence has achieved. For instance, the distances between some mobile devices and MEC servers and other parameters are not friendly enough to offload the task to MEC servers, thus, the tasks will be executed locally, which can bring the descend of offloading rate.

In Fig. 6, the performance of the LODCO-Based Genetic Algorithm with Greedy Policy is compared with that of the benchmark algorithm (the LODCO-Based Greedy Algorithm) on the second optimization goal (the number of offloading computation tasks maximum). As we can see, the average ratio of offloading tasks obtained by the LODCO-Based Genetic Algorithm with Greedy Policy (which is 95.0698%) is greater than the one obtained by the LODCO-Based Greedy Algorithm (which is 92.8549%[5]) , while both ratios are larger than the average ratio of offloading tasks obtained by the LODCO Algorithm (which is 84.4401%) . The results prove that our algorithm can obtain better cell capacity than the benchmark algorithm can do.

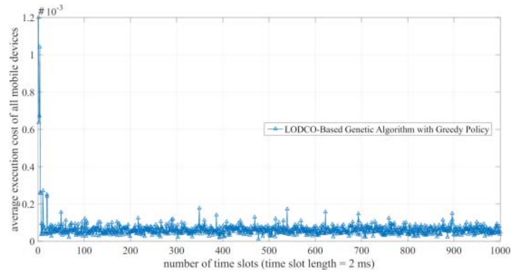


Fig. 3. Average QoE-Cost of all mobile device vs. time.

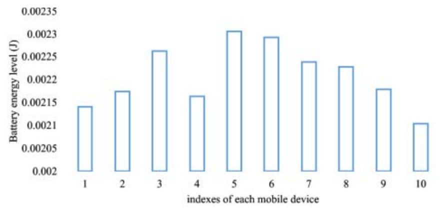


Fig. 4. Average energy level of each mobile device.

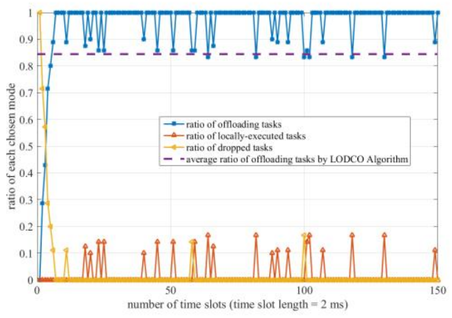
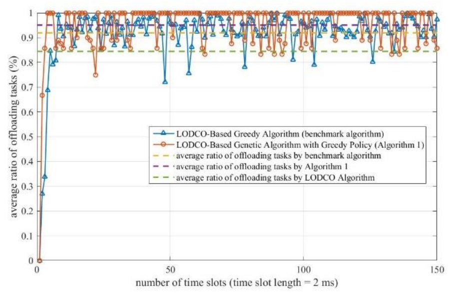


Fig. 5. The ratio of each chosen modes vs. time.



[4] 0.005 and 0.00002 are the specific value of $\theta + E_H^{\max}$ and $E_{\min}$, respectively.

[5] Considering that this is a stochastic optimization problem, the simulation results may exist slightly difference.

Fig. 6. Average ratio of offloading tasks by different algorithms.

B. *Effects of system parameters*

In this subsection, we will demonstrate the impacts of system parameters on the performance of the proposed algorithm.

Fig. 7 depicts the impact of the maximum distance between random mobile device and random MEC server on cell capacity. As can be seen, along with the increase of the maximum distance, the average ratio of offloading tasks gradually decreases. When the distance is arbitrarily far (under the maximum distance constraints), the number of offloading computation tasks will be very low. The reason is that the channel power gain grows with the distance between each mobile device and each MEC server, which will lead to larger energy consumption and longer execution delay. As a result, more mobile devices choose to execute the computation tasks locally.

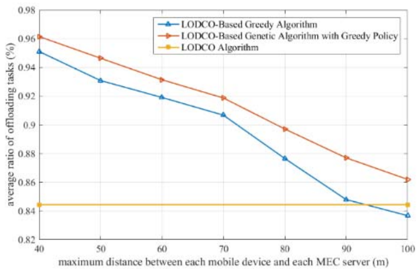


Fig. 7. Average ratio of offloading tasks vs. maximum distance.

As depicted in Fig. 8, along with the increase of ϵ, which belong to $[0,1]$, the average ratio of offloading tasks gradually increases with a slowdown rate and finally converges to the specific value (which is 97.5731%). The reason is that the LODCO-Based Genetic Algorithm with Greedy Policy is based on the ϵ-Greedy Strategy, i.e., a greater ϵ will bring a lager probability to choose the offloading mode.

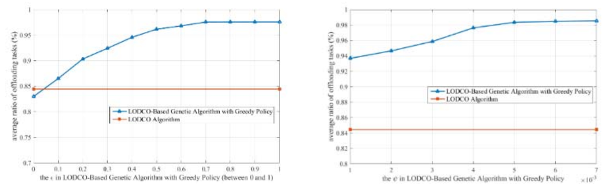


Fig. 8. Average ratio of offloading tasks vs. ϵ and ψ, respectively.

In the right part of Fig. 8, along with the increase of ψ, the average ratio of offloading tasks gradually increases with a slowdown rate and finally converges to the specific value (which is 98.5315%). The reason is that ψ determines the weight of the second optimization goal, which is an important component of $\mathcal{P}_3$.

VI. Conclusions

In this paper, we investigate a multi-user and multi-server mobile-edge computing system with energy harvesting devices. Then we propose one algorithm to obtain the lowest execution cost and largest number of offloading computation tasks based on the LODCO Algorithm, i.e., the LODCO-Based Genetic Algorithm with Greedy Policy, which is an online algorithm with low-complexity. Most importantly, it has no need of too much priori knowledge. After simulation and performance analysis, we can see that the proposed algorithm inherits every advantage from the LODCO Algorithm and adapts to the more complex environment perfectly and offers more than 10% ratio of offloading computation tasks. The proposed algorithm can choose the offloading mode as far as possible, which can bring resource-limited MEC servers' superiority into full play. In conclusion, our study provides a viable strategy to design a complex system which much more approaches to reality.

References

[1] Y.C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing – A key technology towards 5G", ETSI White Paper, 2015.

[2] T.X. Tran, H. Abolfazl, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges", IEEE Communication Magazine, vol. 55, 2017, pp. 54-61.

[3] S. Barbarossa, S. Sardellitti, and P.D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks", IEEE Single Processing Magazine, vol. 31, 2014, pp. 45-55.

[4] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications", IEEE Communications Surveys & Tutorials, vol. 13, 2011, pp. 443-461.

[5] Y.Y. Mao, J. Zhang, K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices", IEEE Journal of Selected Areas Communications, vol. 34, 2016, pp. 3590-3605.

[6] M.H. Min, D.J. Xu, L. Xiao, Y.L. Tang, and D. Wu, "Learning-based computation offloading for IoT devices with energy harvesting", arXiv: 1712.08768, 2017, unpublished.

[7] X.F. Chen, H.G. Zhang, C. Wu, S.W. Mao, Y.S. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning", arXiv: 1804.00514, 2018, unpublished.

[8] X.H. Ge, S. Tu, G.Q. Mao, C.X. Wang, and T. Han, "5G ultra-dense celluar networks", IEEE Wireless Communications, vol. 23, 2016, pp. 71-79.

[9] A. Alfayly and L.F. Sun, "QoE-driven LTE downlink scheduling for multimedia services", Ph.D. thesis, Plymouth University, 2016.

[10] C.F. Liu, M. Bennis, and H.V. Poor, "Latency and reliability-aware task offloading and resource allocation for mobile edge computing", IEEE Global Communications Conference Workshops (GLOBECOM Workshops), Singapore, 2017.

[11] M. Masoudi, B. khamidehi, and C. Cavdar, "Green cloud computing for multi cell networks", IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, USA, 2017.

[12] T.X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks", arXiv: 1705.00704, 2017, unpublished.

[13] L. Yang, J.N. Cao, Z.Y. Wang, and W.G. Wu, "Network aware multi-user computation partitioning in mobile edge clouds", International Conference on Parallel Processing (ICPP), Bristol, United Kingdom, 2017.

[14] Y.Y. Mao, C.S. You, J. Zhang, K.B. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective", IEEE Communication Surveys & Tutorials, vol. 19, 2017, pp. 2322-2358.

[15] L.B. Huang and M.J. Neely, "Utility optimal scheduling in energy-harvesting networks", IEEE/ACM Transactions on Networking, vol. 21, 2013, pp. 1117-1130.

[16] W.W. Zhang, Y.G. Wen, K. Guan, D. Kilper, H.Y. Luo, and D.O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel", IEEE Transactions on Wireless Communications, vol. 12, 2013, pp. 4569-4581.

[17] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems", Foundations and Trends® in Machine Learning, vol. 5, 2012, pp. 1-22.