Energy Efficient Computation Offloading for Energy Harvesting-Enabled Heterogeneous Cellular Networks (Workshop)

Mengqi Mao(✉), Rong Chai, and Qianbin Chen

School of Communication and Information Engineering,
Chongqing University of Posts and Telecommunications, Chongqing 400065, China
maomengqii@163.com, {chairong,chenqb}@cqupt.edu.cn

Abstract. Mobile edge computing (MEC) is regarded as an emerging paradigm of computation that aims at reducing computation latency and improving quality of experience. In this paper, we consider an MEC-enabled heterogeneous cellular network (HCN) consisting of one macro base station (MBS), one small base station (SBS) and a number of users. By defining workload execution cost as the weighted sum of the energy consumption of the MBS and the workload dropping cost, the joint computation offloading and resource allocation problem is formulated as a workload execution cost minimization problem under the constraints of computation offloading, resource allocation and delay tolerant, etc. As the formulated optimization problem is a Markov decision process (MDP)-based offloading problem, we propose a hotbooting Q-learning-based algorithm to obtain the optimal strategy. Numerical results demonstrate the effectiveness of the proposed scheme.

Keywords: Mobile edge computing · Heterogeneous cellular network · Computation offloading · Resource allocation · Hotbooting Q-learning

1 Introduction

Mobile edge computing (MEC) is regarded as an emerging paradigm of computation that aims at reducing computation latency and improving quality of experience through pushing mobile computing, network control and storage to the network edges of cellular networks [1]. On the other hand, to improve the transmission performance of the cellular users, heterogeneous cellular networks (HCNs) which consist of macro base stations (MBSs) and various heterogeneous small BSs (SBSs) have demonstrated promising advantages. By deploying high performance MEC servers at the MBS or the SBSs of the HCNs, MEC-enabled HCNs are expected to offer flexible network access and enhanced computation capability for the users.

In recent years, considerable efforts have been dedicated to studying the computation offloading schemes of the MEC-enabled HCNs [2,3]. In [2], computation offloading schemes were considered for MEC-enabled HCNs. Aiming to minimize the overall computation overhead of all the users, the authors formulated the computation offloading problem as a non-cooperative game model and demonstrated the existence of Nash

H. Gao et al. (Eds.): ChinaCom 2019, LNICST 313, pp. 391–401, 2020.
https://doi.org/10.1007/978-3-030-41117-6_32

equilibrium point. In [3], a two-tier computation offloading framework was proposed for HCNs. By formulating task execution problem as an energy consumption minimization problem, and solving the problem by means of separable semi-definite program and quadratically constrained quadratic program, an efficient joint user association and computation offloading algorithm strategy was obtained.

Energy harvesting (EH) technology can also be applied to reduce the on-grid energy consumption of the MEC servers. The authors in [4] considered the task offloading problem in an EH-powered MEC system. The task offloading problem was formulated as a system cost minimization problem and solved by using the Lyapunov optimization optimization technique. Specifically, the emerging EH-SBSs, which are equipped with EH devices (like solar panels or wind turbines) and exploit renewable energy as supplementary or alternative power sources, have received great attentions from both academia and industry. The possibility and reliability of self-powered cellular networks were investigated in [5]. In [6], a dynamic computation offloading framework was proposed for an EH-enabled MEC system. By examining the weighted sum of task execution delay and task dropping cost, the task execution problem was formulated as average weighted sum minimization problem and the optimal offloading strategy were obtained by solving the optimization problem.

The problem of task offloading and resource allocation can be jointly designed to further enhance task execution performance. In [7], the problem of task offloading and resource allocation was considered for MEC systems and an optimal joint design scheme was proposed to achieve the minimization of system-wide computation overhead. The authors in [8] addressed the problem of joint task offloading and resource allocation problem in an MEC-enabled HCN and proposed a joint optimal scheme to achieve the minimum energy consumption of the users.

In this paper, we consider the computation offloading and resource allocation problem in an MEC-enabled HCN. By defining workload execution cost as the weighted sum of the energy consumption of MBS and workload dropping cost, the joint computation offloading and resource allocation problem is formulated as a workload execution cost minimization problem. As the original optimization problem is a Markov decision process (MDP) problem, traditional Q-learning-based algorithm and hotbooting Q-learning-based algorithm are proposed to solve the problem.

2 System Model

In this paper, we consider an MEC-enabled HCN consisting of one MBS, one SBS and a number of users where the users are allowed to access both the MBS and the SBS through wireless links, and the SBS may also access the MBS via wireless link. To facilitate edge computation, we assume that both the MBS and the SBS are equipped with an MEC server, which are capable of executing user workloads. It is further assumed that the MBS is powered solely by on-grid power, while the SBS is equipped with an EH component and powered purely by the harvested renewable energy. The considered system model is plotted in Fig. 1.

We consider the computation offloading and resource allocation problem during a relatively long time duration. For convenience, the time duration is divided into equal-length slots with the length of each time slot being τ. We assume that at each time

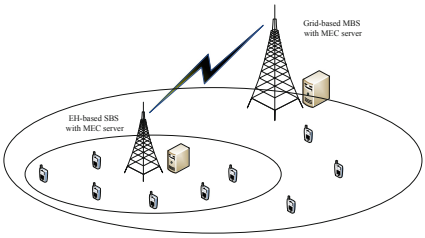

Fig. 1. System model

slot, the computation workloads may arrive at the SBS randomly, and the arrival of the workload follows a Poisson process with the average amount of the workload being λ. Let λ_k denote the amount of the workload arriving at the beginning of the kth time slot, we set $\mathrm{E}[\lambda_k] = \lambda$, $k = 1, 2, 3, \cdots$, where $\mathrm{E}[x]$ denotes the expectation value of x. At each time slot, upon receiving the workload from the users, the SBS may conduct local computing or offload the workload to the MBS. In addition, due to highly limited energy supply at the SBS, the SBS may also drop the workload. Let $x_k^{\mathrm{m}} \in \{0, 1\}$ and $x_k^{\mathrm{s}} \in \{0, 1\}$ denote respectively the computation offloading variable of the MBS and the SBS at the kth time slot, i.e., $x_k^{\mathrm{m}} = 1$, if the computation workload requested at the kth time slot is offloaded to the MBS, otherwise, $x_k^{\mathrm{m}} = 0$; $x_k^{\mathrm{s}} = 1$, if the computation workload requested at the kth slot is executed at the SBS, otherwise, $x_k^{\mathrm{s}} = 0$. We further define the binary variable of workload dropping. Let $x_k^{\mathrm{d}} \in \{0, 1\}$ denote the workload dropping variable at the kth time slot, i.e., $x_k^{\mathrm{d}} = 1$, if the workload requested at the kth time slot is dropped, otherwise, $x_k^{\mathrm{d}} = 0$.

In the case that the workload is executed locally at the SBS at the kth time slot, the SBS may allocate a portion of computation capability of the MEC server to the workload. Let f_k^{s} denote the computation capability of the SBS allocated for executing the workload at the kth time slot, we have $0 \leq f_k^{\mathrm{s}} \leq f_{\max}$, where $f_{\max}$ denotes the maximum computation capability of the MEC server deployed at the SBS.

3 Problem Formulation

In this section, we define the average long-term execution cost of the workloads and formulate the joint computation offloading and resource allocation problem as a workload execution cost minimization problem.

3.1 Objective Function

We define the average long-term execution cost of the workloads as

$$C = \lim_{T \to \infty} \frac{1}{T} \mathrm{E} \left[\sum_{k=0}^{T-1} C_k \right] \tag{1}$$

where C_k denotes the execution cost of the workload at the kth time slot. Jointly considering the energy consumption of the MBS and the workload dropping cost, we formulate C_k as

$$C_k = x_k^{\mathrm{m}} E_k^{\mathrm{m}} + \phi \psi_k \tag{2}$$

where E_k^{m} is the energy consumption of the MBS at the kth time slot, ψ_k denotes the workload dropping cost at the kth time slot and ϕ is the weight of the workload dropping cost. We formulate E_k^{m} as

$$E_k^{\mathrm{m}} = \delta^{\mathrm{m}} (f^{\mathrm{m}})^2 W_k \tag{3}$$

where δ^{m} is the energy consumption coefficient of the MBS, f^{m} is the computation capability of the MBS. ψ_k in (2) can be calculated as

$$\psi_k = x_k^{\mathrm{d}} \lambda_k. \tag{4}$$

3.2 Optimization Constraints

To design the optimal joint computation offloading and resource allocation strategy which minimizes the execution cost of the workloads, we should consider following optimization constraints.

Workload Processing Constraint. In this paper, we assume that the SBS can only choose one of the two computation offloading modes for the workloads or drop the workloads, thus, we can express the workload processing constraint as

$$\mathrm{C1}: \quad x_k^{\mathrm{m}} + x_k^{\mathrm{s}} + x_k^{\mathrm{d}} = 1. \tag{5}$$

Resource Allocation Constraints. The computing resource constraints of the SBS can be expressed as

$$\mathrm{C2}: \quad 0 \le f_k^{\mathrm{s}} \le f_{\max}, \tag{6}$$

$$\mathrm{C3}: \quad 0 \le P_k \le P_{\max} \tag{7}$$

where P_k denotes the transmit power of the SBS when transmitting the workload to the MBS at the kth time slot and $P_{\max}$ denotes the maximum transmit power of the SBS.

Delay Tolerant Constraint. In this paper, we consider delay-sensitive applications and assume that the execution latency of the workloads should be less the length of time slot, i.e.,

$$\mathrm{C4}: \quad D_k \le \tau \tag{8}$$

where D_k denotes the workload execution latency at the kth time slot, which can be expressed as

$$D_k = x_k^{\mathrm{m}} D_k^{\mathrm{m}} + x_k^{\mathrm{s}} D_k^{\mathrm{s}} \tag{9}$$

where D_k^{m} and D_k^{s} denote respectively the workload execution latency in MBS offloading mode and SBS computing mode at the kth time slot. D_k^{m} can be formulated as

$$D_k^{\mathrm{m}} = D_k^{\mathrm{m,t}} + D_k^{\mathrm{m,e}} \tag{10}$$

where $D_k^{\mathrm{m,t}}$ denotes the transmission latency required for SBS to offload its workload to the MBS and $D_k^{\mathrm{m,e}}$ denotes the workload computation latency at the MBS. $D_k^{\mathrm{m,t}}$ is given by

$$D_k^{\mathrm{m,t}} = \frac{\lambda_k}{R_k} \tag{11}$$

where R_k denotes the achievable data rate of SBS when transmitting to the MBS at the kth time slot and can be expressed as

$$R_k = B\log_2\left(1 + \frac{P_k g_k}{\sigma^2}\right) \tag{12}$$

where B denotes the bandwidth of the link between the SBS and the MBS, P_k denotes the transmit power of the SBS when offloading to the MBS at the kth time slot, g_k and σ^2 denote respectively the channel gain and the noise power of the link between the SBS and the MBS at the kth time slot. $D_k^{\mathrm{m,e}}$ can be computed as

$$D_k^{\mathrm{m,e}} = \frac{W_k}{f^{\mathrm{m}}} \tag{13}$$

where W_k denotes computation resource required for workload execution. W_k can be calculated as

$$W_k = \rho\lambda_k \tag{14}$$

where ρ denotes the number of CPU cycles required to process one bit. D_k^{s} in (9) can be formulated as

$$D_k^{\mathrm{s}} = \frac{W_k}{f_k^{\mathrm{s}}}. \tag{15}$$

It should be noticed that without loss of generality, we assume that the output of the computation execution at the MBS is of small size, hence, we may omit the transmission latency for sending the result back to the SBS.

Energy Causality Constraint. At each time slot, the energy consumed by the SBS will not exceed its battery level, i.e., the energy causality constraint must be satisfied, which can be expressed as

$$\mathrm{C5}: \quad E_k^{\mathrm{s}} \leq B_k \tag{16}$$

where E_k^{s} denotes the energy consumption of the SBS at the kth time slot, and B_k denotes the residual battery level at the beginning of the kth time slot. E_k^{s} can be calculated as the sum of basic energy consumption, the energy consumption resulted from executing the workload locally and that for transmitting the workload to the MBS, hence, can be expressed as

$$E_k^{\mathrm{s}} = E_k^{\mathrm{s},0} + x_k^{\mathrm{s}} E_k^{\mathrm{s,e}} + x_k^{\mathrm{m}} E_k^{\mathrm{m,t}} \tag{17}$$

where $E_k^{\mathrm{s},0}$ denotes the basic energy consumption of the SBS, $E_k^{\mathrm{s,e}}$ denote the energy consumption of the SBS for executing the workload, and $E_k^{\mathrm{m,t}}$ denotes the energy consumption required for the SBS to transmit the workload to the MBS. $E_k^{\mathrm{s,e}}$ can be formulated as

$$E_k^{\mathrm{s,e}} = \delta^{\mathrm{s}} (f_k^{\mathrm{s}})^2 W_k \tag{18}$$

where δ^{s} is the energy consumption coefficient of the SBS. $E_k^{\mathrm{m,t}}$ is given by

$$E_k^{\mathrm{m,t}} = P_k D_k^{\mathrm{m,t}}. \tag{19}$$

The battery energy level B_k in (16) evolves according to the following equation:

$$B_{k+1} = \min \left\{ B_k + E_k^{\mathrm{h}} - E_k^{\mathrm{s}}, B_{\max} \right\} \tag{20}$$

where E_k^{h} denotes the harvested energy of the SBS at the kth time slot, and $B_{\max}$ denotes the battery capacity of the SBS. To capture the intermittent and unpredictable nature of the EH process, we model it as successive energy packet arrivals, i.e., at the beginning of the kth time slot, energy packets with the amount of energy being $E_k^{\mathrm{h}} \leq E_{\max}$ arrives at the SBS and then will be harvested and stored in the battery. Then, from the $(k+1)$th time slot, the stored energy will be available for computation and communication. We further assume that E_k^{h}, $k \geq 1, 2, \cdots$ is i.i.d. with the maximum value being $E_{\max}$.

3.3 Optimization Problem

To minimize the long-term average execution cost subject to the constraints, we formulate the optimization problem as follows:

$$\begin{aligned} \min_{x_k^{\mathrm{m}}, x_k^{\mathrm{s}}, x_k^{\mathrm{d}}, f_k^{\mathrm{s}}, P_k} \quad & \lim_{T \to \infty} \frac{1}{T} \mathrm{E} \left[\sum_{k=0}^{T-1} C_k \right] \\ \text{s.t.} \quad & \mathrm{C1} - \mathrm{C5} \end{aligned} \tag{21}$$

4 Solution of the Optimization Problem

It can be shown that the optimization problem formulated in (21) can be regarded as an infinite-horizon MDP problem and solved by using reinforcement learning method.

4.1 MDP-Based Offloading Problem Formulation

The problem of making an offloading decision for the SBS can be formulated as a finite MDP which can be characterized by four elements, i.e., state space, action space, reward function and policy.

State Space. The state of the MDP at any particular slot can be characterized by three metrics, i.e., the residual battery level of the SBS, the channel gain of the link between the SBS and the MBS, and the amount of the workloads arriving at each time slot. Let S denote the state space of the MDP and $s_k \in S$ denote the state at the kth time slot, we obtain $s_k = \{B_k, g_k, \lambda_k\}$.

Action Space. The action of the MDP taken at any time slot can be characterized by the computation offloading and resource allocation strategy made at the time slot. Let A denote the action space of the MDP and $a_k \in A$ denote the action taken at the kth time slot, we may express $a_k = \left\{x_k^{\mathrm{m}},\ x_k^{\mathrm{s}}, x_k^{\mathrm{d}}, f_k^{\mathrm{s}}, P_k\right\}$.

Reward Function. In the case that the MDP is at state s_k, various actions can be taken under the constraints C1–C5, immediate reward will be resulted accordingly. We define the immediate reward function of taking action a_k at state s_k as follows:

$$r\left(s_k, a_k\right) = \begin{cases} -E_k^{\mathrm{m}}, & \text{if } x_k^{\mathrm{m}} = 1 \\ -\phi\psi_k, & \text{if } x_k^{\mathrm{d}} = 1 \\ 0, & \text{if } x_k^{\mathrm{s}} = 1 \end{cases} \tag{22}$$

Policy. The policy of the formulated MDP at the kth time slot is defined as a mapping $\pi : s_k \rightarrow a_k$. Given initial state $s_0 \in S$, we focus on optimizing the policy to maximize the expected long-term system reward, the expected discounted long-term system reward is defined as

$$V^{\pi}\left(s_0\right) = \mathrm{E}\left(\sum_{k=0}^{\infty} \gamma^k r\left(s_k, a_k\right) | s_0\right) \tag{23}$$

where $\gamma \in (0, 1)$ is a constant discount factor. The discounted cumulative reward $V\left(s_k\right)$ of state s_k can be expressed as

$$V\left(s_k\right) = r\left(s_k, a_k\right) + \gamma \sum_{\tilde{s}_k \in \mathrm{S}} p\left(\tilde{s}_k | s_k, a_k\right) V\left(\tilde{s}_k\right) \tag{24}$$

where $p\left(\tilde{s}_k | s_k, a_k\right)$ is the transition probability from s_k to $\tilde{s}_k$ when choosing action a_k. The optimal policy π^* can be obtained when the total discounted expected reward is maximal according to the Bellman's theory, i.e.,

$$V^*\left(s_k\right) = \max_{a_k \in A} V\left(s_k\right) \tag{25}$$

where $V^*\left(s_k\right)$ is the expected optimal reward of s_k.

4.2 Traditional Q-Learning-Based Method

Examining (23)–(25), we can see that given the transition probability $p\left(\tilde{s}_k | s_k, a_k\right)$, the optimal policy can be derived by solving the Bellman equation. However, in many practical scenarios, it can be very different or computationally prohibitive to obtain these probability distributions. To tackle this problem, model-free reinforcement learning method such as Q-learning method can be employed to derive the optimal policy.

Applying Q-learning method to solve the joint computation offloading and resource allocation problem formulated in (21), we define Q value denoted by $Q\left(s_k, a_k\right)$ for

state-action pair (s_k, a_k). Given an initial value of $Q(s_k, a_k)$, for $\forall(s_k, a_k)$, $Q(s_k, a_k)$ can be updated according to the following formula:

$$Q(s_k, a_k) = (1-\alpha) Q(s_k, a_k) + \alpha \left(r(s_k, a_k) + \gamma \max_{a_k' \in A} Q\left(s_k', a_k'\right) \right) \tag{26}$$

where $0 \leq \alpha \leq 1$ is the learning rate. Once $Q(s_k, a_k)$ achieves convergence for $\forall(s_k, a_k)$ pair, the optimal action a_k^* taken at state s_k can be determined, and the optimal joint computation offloading and resource allocation strategy can be obtained.

It should be noticed that to achieve the tradeoff between exploring the new actions and exploitating the existing actions, ε-greedy policy can be applied. More specifically, although with a high probability $(1-\varepsilon)$, the action which maximizes the reward function will be selected; with a relatively small probability ε, one action is randomly selected to avoiding the algorithm converging to the local maximum.

4.3 Hotbooting Q-Learning-Based Method

For simplicity, applying traditional Q-learning algorithm to solve engineering problem, we may set the initial Q value as zero for $\forall(s_k, a_k)$ pair, however, this may require relative long time for the algorithm to achieve convergence, which is highly undesired. To solve this problem, hotbooting technique can be applied. In particular, instead of initializing Q value as zero, we may obtain a set of training data in advance from large-scale experiments conducted in similar scenarios, then set the initial Q value based on the training data. In this manner, as the training data and initial Q value can be relatively close to the optimal value, the time required for the algorithm to achieve convergence can be reduce significantly. The proposed hotbooting Q-learning-based method for solving the joint computation offloading and resource allocation problem is summarized in Algorithm 1.

Algorithm 1. Hotbooting Q-learning based offloading decision

1: Initialization α , γ and B_0
2: Set **Q** = **Q*** according to the hotbooting technique
3: **for** $k = 1, 2, 3, ...$ **do**
4: Observe the channel condition g_k, workload size λ_k and the current battery level B_k
5: Select $a_k = \left\{x_k^{\mathrm{m}},\ x_k^{\mathrm{s}}, x_k^{\mathrm{d}}, f_k^{\mathrm{s}}, P_k\right\}$ via ε-greedy policy
6: Evaluate the energy consumption, workload drop rate and the execution cost
7: Update $Q(s_k, a_k)$ via (26)
8: **end for**

5 Simulation Results

In this section, simulation results are provided to evaluate the proposed Hotbooting Q-learning-based algorithm. For comparison, we also examine the performance of two benchmark algorithms, i.e., Q-learning-based algorithm and greedy algorithm. To conduct Greedy algorithm, At each time slot, the joint computation offloading and resource allocation strategy was designed with the aim of minimizing the instantaneous workload execution cost. The parameters used in the simulation are given in Table 1.

Table 1. Simulation parameters

Parameters	Value
Channel bandwidth B	10 MHz
Noise power σ^2	-95 dBm
Battery capacity $B_{\max}$	100 J
Maximum transmit power $P_{\max}$	2 W
Circuit power consumption $E_k^{s,0}$	20 J
Time slot length τ	1s
Bit/CPU conversion coefficient ρ	1000

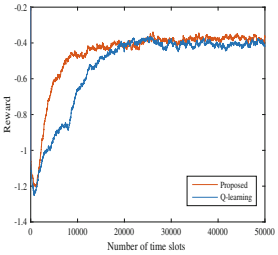

Fig. 2. Reward vs number of time slots

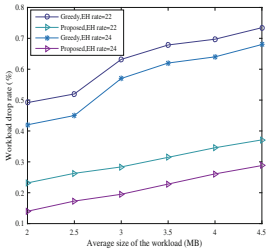

Fig. 3. Workload drop rate vs average size of the workload

Figure 2 shows the system reward versus number of time slots. From the figure, we can see that the Q-learning algorithm can almost achieve the same reward as the proposed Hotbooting Q-learning algorithm, the time required for the Q-learning algorithm to reach convergence is much longer, demonstrating the effectiveness of the proposed hotbooting Q-learning algorithm. This is because the hotbooting technique can formulate the emulated experiences in dynamic radio environments to effectively reduce the exploration trials and thus significantly improve the convergence speed of Q-learning.

In Fig. 3 shows workload drop rate versus the average size of the computation workload. It is shown in the figure that the workload drop rate increases with the increase of the average size of the computation workload for both our proposed scheme and the

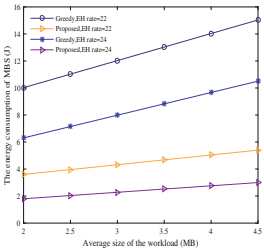

Fig. 4. The energy consumption of MBS vs average size of the workload

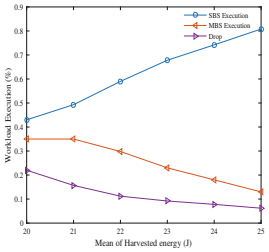

Fig. 5. Workload execution vs mean of harvested energy

greedy scheme. This is because larger workload size requires larger energy consumption for completing workload computation, thus resulting in larger workload drop rate due to insufficient residual energy of the SBS's battery.

In Fig. 4, we can see that the energy consumption of MBS increases with the increase of the average size of the computation workload for the proposed scheme and the greedy scheme. This is because larger workload size results in larger energy consumption for completing workload computation, thus the algorithm prefers to execute workload at the MBS for energy savings. Comparing the results obtained from our proposed scheme and the greedy scheme, we can see that our proposed scheme outperforms the greedy scheme.

In Fig. 5, it is shown in the figure that the ratio of the workload being executed at the SBS increases with the increase of the mean of harvested energy, while the ratio of the workload being executed at the MBS and that being dropped decrease with the increase of the mean of harvested energy. It is because that the SBS tends to execute the workloads for high rewards in the case that the energy of the SBS is relatively sufficient.

6 Conclusion

In this paper, we consider an MEC-enabled HCN system and formulate the joint computation offloading and resource allocation as an optimization problem which minimizes the long-term execution cost of the workloads. As the formulated optimization problem is MDP-based offloading problem, we propose a Q-learning-based algorithm and a hot-booting Q-learning-based algorithm. Numerical results demonstrate the effectiveness of the proposed algorithms.

References

1. Mao, Y., You, C., Zhang, J., Huang, K., Letaief, K.B.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. **19**(4), 2322–2358 (2017)
2. Guo, H., Liu, J., Zhang, J.: Efficient computation offloading for multi-access edge computing in 5G HetNets. In: Proceedings of the IEEE International Conference on Communication (ICC), Kansas City, MO, pp. 1–6 (2018)

3. Dai, Y., Xu, D., Maharjan, S., Zhang, Y.: Joint computation offloading and user association in multi-task mobile edge computing. IEEE Trans. Veh. Technol. **67**(12), 12313–12325 (2018)
4. Wu, H., Chen, L., Shen, C., Wen, W., Xu, J.: Online geographical load balancing for energy-harvesting mobile edge computing. In: Proceedings of the IEEE International Conference on Communication (ICC), pp. 1–6, May 2018
5. Dhillon, H.S., Li, Y., Nuggehalli, P., Pi, Z., Andrews, J.G.: Fundamentals of heterogeneous cellular networks with energy harvesting. IEEE Trans. Wirel. Commun. **13**(5), 2782–2797 (2014)
6. Mao, Y., Zhang, J., Letaief, K.B.: Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J. Sel. Areas Commun. **34**(12), 3590–3605 (2016)
7. Pham, Q., Le, L.B., Chung, S., Hwang, W.: Mobile edge computing with wireless backhaul: joint task offloading and resource allocation. IEEE Access **7**, 16444–16459 (2019)
8. Song, Z., Liu, Y., Sun, X.: Joint radio and computational resource allocation for NOMA-based mobile edge computing in heterogeneous networks. IEEE Commun. Lett. **22**(12), 2559–2562 (2018)