Hindawi
Mobile Information Systems
Volume 2021, Article ID 9911332, 16 pages
https://doi.org/10.1155/2021/9911332

Review Article

Resource Caching and Task Migration Strategy of Small Cellular Networks under Mobile Edge Computing

Runfu Liang,[1] Gaocai Wang,[1] and Jintian Hu[2]

[1]*School of Computer and Electronics Information, Guangxi University, Nanning 530004, China*
[2]*School of Computer, Central South University, Changsha 410083, China*

Correspondence should be addressed to Gaocai Wang; wanggcgx@163.com

Received 16 March 2021; Revised 27 April 2021; Accepted 3 June 2021; Published 21 June 2021

Academic Editor: Adrian Kliks

As computing-intensive mobile applications become increasingly diversified, mobile devices' computing power is hard to keep up with demand. Mobile devices migrate tasks to the Mobile Edge Computing (MEC) platform and improve the performance of task processing through reasonable allocation and caching of resources on the platform. Small cellular networks (SCN) have excellent short-distance communication capabilities, and the combination of MEC and SCN is a promising research direction. This paper focuses on minimizing energy consumption for task migration in small cellular networks and proposes a task migration energy optimization strategy with resource caching by combining optimal stopping theory with migration decision-making. Firstly, the process of device finding the MEC platform with the required task processing resources is formulated as the optimal stopping problem. Secondly, we prove an optimal stopping rule's existence, obtain the optimal processing energy consumption threshold, and compare it with the device energy consumption. Finally, the platform with the best energy consumption is selected to process the task. In the simulation experiment, the optimization strategy has lower average migration energy consumption and higher average data execution energy efficiency and average distance execution energy efficiency, which improves task migration performance by 10% ~ 60%.

1. Introduction

With the continuous update of mobile smart devices, computing-intensive mobile applications are increasingly diversified, such as augmented reality, virtual reality, mobile video call, and gesture recognition. Computation-intensive applications require strict latency and mighty processing power, implying tremendous pressure on mobile devices with limited resources and energy [1]. How to solve the insufficient computing power of mobile devices is an extremely critical problem. Therefore, MEC is born at the right time.

MEC evolved from centralized cloud computing, which allows mobile devices to offload computing tasks to the edge of the network, such as small base station (SBS), rather than local computing [2]. Then the MEC platform managed uniformly device computing tasks resource to reduce computing latency and energy consumption. However, the MEC server's computing capacity is usually limited. How to effectively allocate the limited computing resources of the MEC server to terminal devices is a crucial problem. When the local device's computing power is insufficient, the device migrates the computing task to a nearby MEC platform and uses its rich resources and supercomputing power to ensure the high-performance execution of the migrated terminal program. However, if MEC platform's resources are not used reasonably, it will cause a waste of resources and is not conducive to saving the device's energy consumption. For example, in a small cellular network, mobile user A performs computing by accessing the MEC platform, which downloads the device's required resources from the SBS. When the task of mobile user A is completed, the required resources will be released immediately. Simultaneously, when mobile user B visits, the MEC platform downloads the required resources for mobile user B and performs the next step. However, if user A and user B access the MEC platform

for the same resources, repeated download and release of resources will cause unreasonable use of resources and increase the MEC platform's operational burden. Therefore, the reasonable allocation and utilization of resources through the MEC platform can improve task migration performance.

In recent years, because MEC has powerful computing and storage performance and small cellular networks have excellent short-distance communication capabilities, the combination of MEC and SCN has become a potential research direction. In a small cellular network, because the types and quantities of resources requested by each user are different, the resources cached by the MEC platform are also different. The SBS always maintains the maximum coverage area which will cause unnecessary waste of resources. Therefore, using cell range extension (CRE) to adjust each SBS's coverage helps save devices' energy consumption.

This paper focuses on minimizing the energy consumption of task migration in small cellular networks. We propose a task migration energy consumption optimization strategy with a resource cache by combining the optimal stopping theory with the migration decision. Firstly, the process of device finding the MEC platform with the required task processing resources is formulated as the optimal stopping problem. Secondly, we prove an optimal stopping rule's existence, obtain the optimal processing energy consumption threshold, and compare it with the device energy consumption. Finally, the platform with the best energy consumption is selected to optimize migration performance and reduce related devices' overall energy consumption. The main contributions of this paper are as follows.

(i) We combine MEC with SCN to study the problem of resource caching and task migration and select the most frequently used resources to be stored as the precached resources of the platform within the next time interval t

(ii) We formulate the process of finding a platform with the required resources for task processing by the equipment as an optimal stopping problem and select the platform with the best energy consumption to process the task

(iii) Finally, the simulation experiment shows that the strategy proposed in this paper has lower average migration energy consumption and higher average data execution energy efficiency and average distance execution energy efficiency, and the task migration performance is greatly improved

This paper's organizational structure is as follows: Section 2 reviews the relevant research work; Section 3 describes the problem and establishes the model. Section 4 describes the task migration strategy with resource caching; Section 5 presents the simulation experiment and the experimental results analysis. Section 6 summarizes the whole paper.

2. Related Work

To reasonably use MEC platform resources and improve task migration performance, researchers have done a lot of research on resource management of the MEC platform task migration process. Based on the summary and analysis of the existing optimization strategies, the current research can be roughly divided into two categories.

The first category is the study of how MEC's computing resources are effectively allocated to requesting users. Tasks on mobile devices are processed by migrating to the MEC platform when they are difficult to perform. If there are multiple users migrating tasks to the MEC platform at the same time, it is important to reasonably allocate MEC resources to ensure that users' requests are completed with high quality. Zhang [3] modeled the problem of computing resource management as a profit maximization problem. And by analyzing the relationship between reserved resources, migration tasks, and repurchase costs, they proposed a fast-convergent real-time repurchase scheme for mobile edge servers to minimize the repurchase costs. Li [4] proposed a high computational density model for single-user multitasking based on the multitask unloading environment. They adopted a simulated annealing algorithm to effectively improve the unloading rate and save energy consumption. Similarly, Dai et al. [5] proposed a new two-layer computing offloading framework for heterogeneous networks. On this basis, by jointly optimizing user association and computing offloading, computing resources and transmission power are reasonably allocated to minimize the total energy consumption of each device. Besides, Jing et al. [6] proposed a heterogeneous network distributed joint computing offloading and resource allocation optimization (JCORAO) scheme for the allocation of uplink subchannels and transmission power and the scheduling of computing resources. At the same time, a cloud and wireless resource allocation algorithm is designed to realize the joint allocation of the above resources. The simulation results show that the proposed scheme can effectively reduce the system energy consumption, task completion time, and system complexity. Considering the characteristics of small cellular networks and mobile edge computing, Guo [7] used a distributed three-stage iterative method based on the energy harvesting SCN combined with the MEC environment to realize the network's green load management and computing resource allocation. Elgendy et al. [8] conducted joint processing of wireless resources and computing resources, adopted JPEG and MPEG4 compression algorithms to reduce transmission overhead, and introduced a security layer to protect transmitted data from network attacks. On this basis, an

efficient unloading algorithm is designed to save about 46% of the system overhead consumption. Most of the above research works are based on how the resources existing on the MEC platform can be effectively allocated to users who need to request task migration, but the scenarios of how the resources of the MEC platform can be cached in the process of task migration on mobile devices have not been considered. If all resources are stored on the MEC platform, this will require high storage costs, increase the burden on the MEC platform, and reduce the performance of the task migration.

The second category is the study of how MEC's computing resources are effectively cached. Due to the different number of users in the area covered by the cellular network and their different task requests, the MEC platform's resources need to be updated frequently. It is important to effectively cache and provide users with appropriate resources. Bi [9] used edge service caching and computational offloading for joint optimization. Therefore, based on a network environment with an edge server, they proposed mixed-integer nonlinear programming to jointly optimize the service cache layout, computational offloading, and system resource allocation. But they do not consider the size of the MEC platform and how to efficiently cache the resources required by the device. Different from the above scheme, Zhao et al. [10] cached the execution results of completed tasks on the MEC platform. They used the active caching algorithm to determine the cache state of the task and proposed a heuristic algorithm based on the greedy strategy to solve the problem of remaining resource allocation and task execution. Yang et al. [11] proposed a long-short-term memory (LSTM) network to predict the task popularity. And a single-agent Q-learning (SAQ-learning) algorithm is invoked to learn a long-term resource allocation strategy. Zhang et al. [12] proposed a blockchain-based caching and delivery market (CDM) as an incentive mechanism for distributed caching systems. On this basis, they constructed the cache sharing and transaction execution consistency models and further took cache placement and scenario selection as Markov decision process problems. Zhang [13] studied the collaborative task unloading and data caching model. They proposed an effective Lyapunov online algorithm that can perform joint task unloading and dynamic data caching strategies for computing tasks or data content. Elgendy et al. [14] cached the completed applications and related codes on the edge server. On this basis, two effective algorithms based on Q-learning and deep-q-network are used to solve the approximate optimal solution of the problem, effectively reducing the overhead of mobile devices. Peng [15] designed an intelligent federated caching and offload strategy under the assumption that the application can exist as a divisible service chain. Unlike the common method of reducing response delay for users only, their system took the rental cost into account and used the open Jackson queuing network to construct the joint optimization problem under the long-term cost constraints. A cost adaptive algorithm is designed to effectively reduce the average service delay of MEC system and kept the average rental cost low. Most of the above-mentioned research work is based on the resources required for precaching tasks and does not consider whether the cache resources are reused. The repeated download and release of resources will not only cause unreasonable use of resources but also increase the operational burden of the MEC platform Related work is list in Table 1.

In addition, the combination of MEC and SCN is also a feasible solution. MEC has powerful computing and storage performance, and small cellular networks have excellent short-distance communication capabilities, so researchers combine MEC with SCN. Wang et al. [16] jointly optimized the video service cache decision and wireless computing resource allocation. They adopted robust optimization to obtain the optimal caching decision and allocated radio and computing resources for video transcoding accordingly. Cai [17] studied an offloading method of parallel communication and computation in a multiuser system to minimize task delay. Jia et al. [18] proposed an energy-saving computing offloading scheme with edge computing and device-to-device (D2D) communication in 5G cellular networks. They defined the computational offloading problem as a stochastic optimization problem and used the Lyapunov optimization technology framework to solve the problem.

The security issues of MEC in the process of computing migration should also draw the attention of researchers. Elgendy et al. [19] introduced AES encryption technology as a security layer to protect sensitive information from network attacks. On this basis, they proposed an integrated model that comprehensively considers security, computational offloading, and resource allocation and improved the performance of the entire system through optimal computational offloading decisions. Abd El-Latif [20] proposed a new authentication encryption protocol based on Quantum Excited Quantum Walk (QIQW) and used this protocol to build a blockchain framework to ensure secure data between IoT devices transmission. Besides, the researchers developed a novel VANET architecture by combining MEC, SDN, and D2D to solve the challenges of high traffic density and blind spots in the vehicle network [21]. They also used the powerful computing of the MEC platform to solve the optimization of video application migration performance [22] and resource constraints of industrial IoT equipment [23] and other issues.

3. Problem Description and Model Building

3.1. Problem Description. Assuming the small cellular network in the MEC network environment is shown in Figure 1, in which the cell of SCN has a small base station, the number of MEC platform and mobile users is N, and the devices will simultaneously perform a task migration in each time interval t. Mobile devices have independent channels when accessing the MEC platform through the SBS, so this paper is not considered channel interference. Usually, the SBS needs to maintain the maximum signal transmission power to cover the whole community so that all mobile devices' communication quality can be guaranteed. However, the user's location is continually changing, and the SBS always maintains the maximum power, which will cause a waste of base station energy. Therefore, the SBS must adjust the signal

Table 1: Summary of the above resource management literature

Type	Literature	Key research points
Resource allocation	3	(i) Fast-convergent real-time repurchase scheme
	4	(ii) Simulated annealing algorithm
	5	(iii) Two-layer computing offloading framework for heterogeneous networks
	6	(iv) Heterogeneous network distributed joint computing offloading and resource allocation optimization (JCORAO) scheme
	7	(v) Distributed three-stage iterative method
	8	(vi) Comprehensive model of resource allocation, compression, and security
Resource cache	9	(i) Mixed-integer nonlinear programming
	10	(ii) Active caching algorithm, greedy strategy
	11	(iii) Long-short-term memory (LSTM) network
	12	(iv) Markov decision process
	13	(v) Lyapunov online algorithm
	14	(vi) Q-learning and deep-q–network algorithm
	15	(vii) Open Jackson queuing network and a cost adaptive algorithm

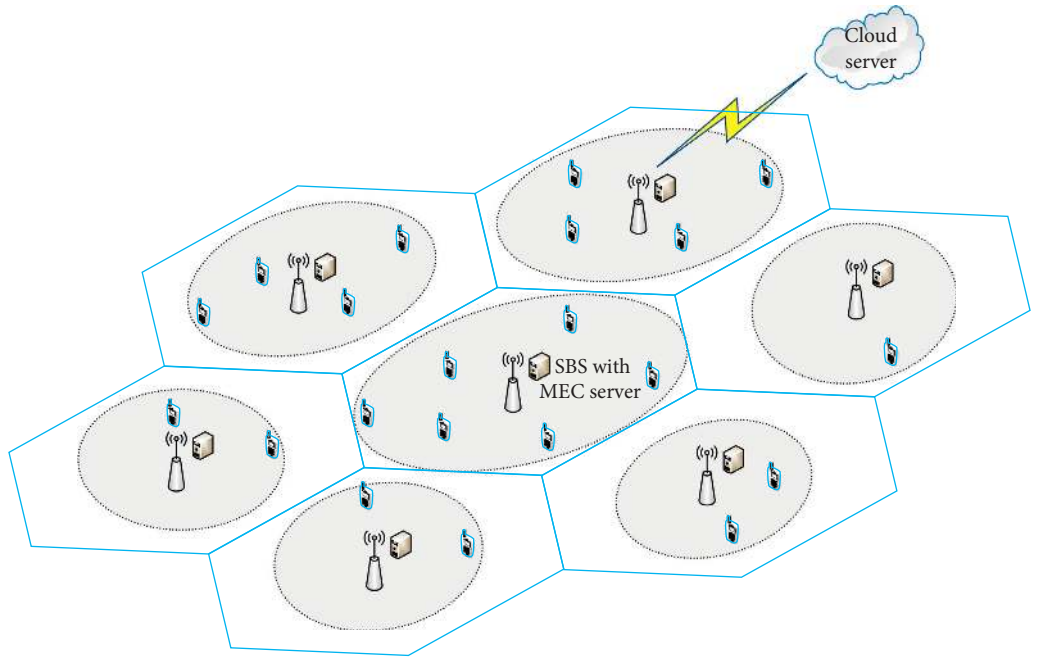


Figure 1: SCN model based on MEC.

coverage area appropriately according to the community's user position. Learning from the method of Guo [7] to solve this problem, this paper also uses cell range extension (CRE) to adjust the coverage range of each SBS. CRE is a technique to expand a picocell range virtually by adding a bias value to the pico received power, instead of increasing transmit power of pico base station, so that coverage, cell-edge throughput, and overall network throughput are improved. Simultaneously, the communication power between devices will also be adjusted adaptively based on device spacing.

In addition, the research on task migration in this paper is based on the background of resource cache. In previous studies, the resource requested by the MEC platform is cached in advance by the mobile device. When the device's task program migrated to the MEC platform, the platform will directly call resources for processing. However, there are many mobile devices in the cell, and the MEC platform caches the required resources for all devices in advance, which will inevitably cause the MEC platform itself to run too much. It does not efficiently utilize resources to release the previous resources directly after completing task program processing. This paper assumes that each cell's MEC platform in SCN will use the most frequently used resource in the last time interval t as the MEC platform's precached resource in the next time interval t. Besides, when the mobile device in cell A migrates the task program to the current MEC platform, it first detects whether the local platform's resources meet the processing requirements. If yes, the platform caches all the resources requested by the device and the platform directly processes the task. If not, the platform partially caches or does not cache the resources requested by the device; the platform will continue to detect the resources

of other cell platforms until it meets the required resources for cross-cell task processing. If the last cell is detected and no platform meets the processing requirements, the MEC platform of the cell where the user is located downloads the required resources from the small base station. The research goal of this paper is to minimize the overall energy consumption of all devices during task migration under the premise of resource cache.

There are N cells with a coverage area radius of l in the small cell network in the task migration scenario. The cell set is cells $= \{\text{cell}_1, \text{cell}_2, \text{cell}_j, \ldots, \text{cell}_N\}$ and the resource information set with the most frequent access to devices in the last time interval t cached on the MEC platform of each cell is $\text{Rs}_{\text{cache}} = \{\text{Rs}_1, \text{Rs}_2, \text{Rs}_j, \ldots, \text{Rs}_N\}$, $j \epsilon [1, N]$. As shown in Figure 2, there are M mobile devices in the cell and the information set of mobile devices is $\text{UEs} = \{\text{UE}_1 (d_1, C_1), \text{UE}_2 (d_2, C_2), \ldots, \text{UE}_i (d_i, C_i), \ldots, \text{UE}_M (d_M, C_M)\}$, $i \in [1, \text{M}]$. The UE_i location information is d_i and C_i is the information of the UE_i task to be processed. Because there are many cells and devices in SCN, this paper only selects all devices in one cell as the research object of task migration to reduce the research complexity.

Taking the UE_i task migration process in cell_j as an example, when the task $C_i (A_i, B_i, D_i)$ (A_i is the task data size, B_i is the task requiring the resource type, and D_i is the task requiring the resource size) is migrated to the MEC platform, the resource cache information Rs_j is detected. When $\text{Rs}_j = B_i$, that is when the cached resources meet the needs of D_i task running and the MEC platform is idle. The MEC platform will start executing the program. Otherwise, it will continue to detect the resource cache information for the next cell (Rs_{j+1}). Assuming the communication power of mobile devices is P_{UEi}, the energy consumption per detection of the device is E_d. The task execution power of the MEC platform is P_{meci} and is randomly distributed. The SBS coverage power is P_{cell}. And the maximum coverage power of the SBS ($P_{\max}$) is equal to its download power (P_{sbs}). The transmission rate on the wireless channel between *UEs* and MEC platform is R, the task processing rate of the MEC platform is V_{ci}, and the resource downloading rate of the SBS is V_s. The time of transferring device task to the platform $T_{\text{ue}_{\text{trans}},i} = A_i/R$, the MEC platform executing time $T_{\text{mec}_{\text{exe}},i} = D_i/V_{\text{ci}}$, and the time for the MEC platform to return resources to the device $T_{\text{mec}_{\text{back}},i} = D_i/R$. The SBS resource downloading time $T_{\text{sbs}_{\text{off}},i} = D_i/V_s$, and the time for SBS to return resources to the device $T_{\text{sbs}_{\text{back}},i} = D_i/R$. Therefore, the total energy consumption of the task migration (E_{UEi}) is obtained as follows:

$$\begin{aligned} E_{\text{UEi}} = {} & \text{nE}_d + P_{\text{UEi}} T_{\text{ue}_{\text{trans}},i} \\ & + P_{\text{meci}} \left(T_{\text{mec}_{\text{exe}},i} + T_{\text{mec}_{\text{back}},i} \right) \\ & + P_{\text{sbs}} \left(T_{\text{sbs}_{\text{off}},i} + T_{\text{sbs}_{\text{back}},i} \right). \end{aligned} \tag{1}$$

The total energy consumption (E_{all}) of all devices in the SCN during the task migration of M users within the time interval t is given as follows:

$$E_{\text{all}} = \sum_{i=1}^{M} E_{\text{UEi}}. \tag{2}$$

3.2. Model Establishment

3.2.1. Distance Power Model of Device Communication. According to the migration energy consumption of a single UE shown in formula (1) and the total energy consumption of SCN-related devices in formula (2), it can be observed that the amount of energy consumption depends on the communication power and SBS coverage power. The received signal of the mobile terminal during the communication is obtained as follows:

$$\beta(t) = g\eta(t) + \varphi(t). \tag{3}$$

According to the literature [24], g is the attenuation factor of the wireless channel, $\eta(t)$ is the amplitude of the signal emitted by the mobile terminal, and $\varphi(t)$ is the Gaussian white noise of the channel and its noise power is σ.

According to Shannon's formula,

$$R = W \log_2 \left(1 + \frac{S}{N} \right). \tag{4}$$

As shown in Shannon's formula (4), R is the rate of task data transmission, W is the bandwidth of the mobile terminal, and S/N is the signal-to-noise ratio. Set R to constant in this paper. The signal-to-noise ratio is derived as follows:

$$\frac{S}{N} = 2^{R/W} - 1. \tag{5}$$

There is a mathematical relationship between the signal-to-noise ratio S/N and the received power P_{rec} of the device and the noise power σ:

$$\frac{S}{N} = \frac{P_{\text{rec}}}{\sigma}. \tag{6}$$

Therefore, combining formulas (5) and (6) can obtain the received power P_{rec} of the device: $P_{\text{rec}} = \sigma(2^{R/W} - 1)$.

According to the amplitude of the received signal $\sqrt{P_{\text{rec}}}/g$ and the relationship between the transmitted power and the received power of the mobile terminal, the transmit power is derived as follows:

$$P_{\text{sen}} = \frac{P_{\text{rec}}}{g^2} = \frac{\sigma\left(2^{R/W} - 1\right)}{g^2}. \tag{7}$$

Moreover, the relation between attenuation factor g, communication distance d, and coefficient λ is $g = \lambda/d$. The expression of g is taken into formula (7) to obtain the following:

$$P_{\text{sen}} = \frac{\sigma\left(2^{R/W} - 1\right) d^2}{\lambda^2}. \tag{8}$$

The transmit power of UE_i that can be derived from formula (8) is

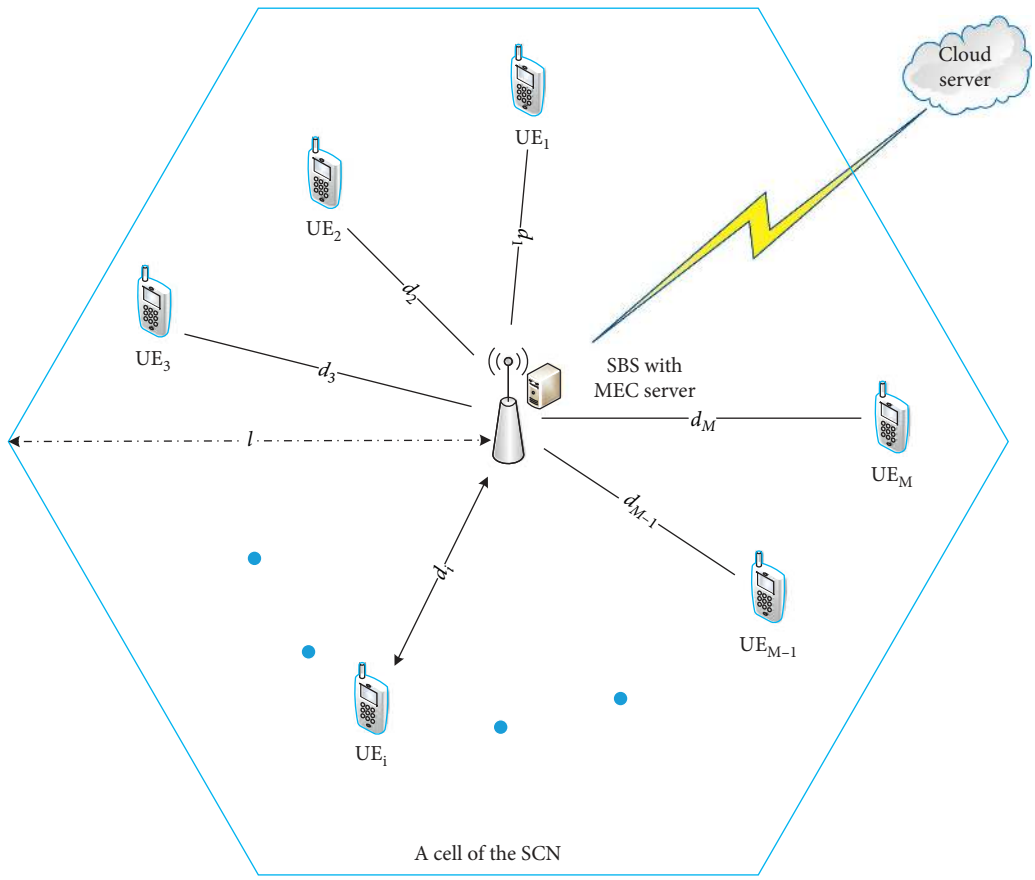


Figure 2: Communication scene model in SCN.

$$P_{\mathrm{UEi}} = \frac{\sigma\left(2^{R/W} - 1\right) d_i^{\,2}}{\lambda^2}. \tag{9}$$

Besides, this paper uses CRE to adjust the coverage of the SBS. Since the *UEs* in the cell are randomly distributed, the size of the SBS internal signal coverage area depends on the farthest device. Let $d_{j,\max} = \max\{d_1, d_1, \ldots, d_i, \ldots, d_M\}$ and, according to formula (9), the coverage radius l of the cell derives the coverage power of the SBS P_{cell}. The maximum power is $P_{\max}$, and $P_{\max} = P_{\mathrm{sbs}}$.

$$\begin{aligned} P_{\mathrm{cell}} &= \frac{\sigma\left(2^{R/W} - 1\right) d_{j,\max}^{\,2}}{\lambda^2}, \\ P_{\max} &= \frac{\sigma\left(2^{R/W} - 1\right) l^2}{\lambda^2}. \end{aligned} \tag{10}$$

3.2.2. Caching Model. As described in Section 3.1, in the task migration scenario, the cell set in the small cellular network is $\mathrm{cells} = \{\mathrm{cell}_1, \mathrm{cell}_2, \mathrm{cell}_j, \ldots, \mathrm{cell}_N\}$, and $\mathrm{Rs}_{\mathrm{cache}} = \{\mathrm{Rs}_1, \mathrm{Rs}_2, \mathrm{Rs}_j, \ldots, Rs_N\}$, $j \in [1, N]$ is the resource information set with the most frequent access to devices within the last time interval t cached on each cell's MEC platform.

Taking the resource caching process of cell_j within the time gap t as an example, when the task $C_i(A_i, B_i, D_i)$ is migrated to the MEC platform and it is found that the required resources are not cached, the MEC platform will download the required resources B_i for task processing from the SBS. Similarly, MEC will also download the required resources for others in the cell. Considering the problem of reasonable utilization of storage space in the MEC platform, it only downloads uncached resources. When all the MEC platform tasks are processed, the most frequently used resources are selected and stored as the platform's precached resources in the next time interval t, and other task processing resources are released. Let $\delta_{j,i}$ denote the call weight of resources in the cell_j on the MEC platform, so $\delta_j = \{\delta_{j,1}, \delta_{j,2}, \delta_{j,3}, \delta_{j,i}, \ldots, \delta_{j,N}\}$. And as the next time interval t, the cache Rs_j is the resource with the largest call weight $\delta_{j\max}$ on the MEC platform, that is, $\delta_{j\max} = \max\{\delta_{j,1}, \delta_{j,2}, \delta_{j,3}, \delta_{j,i}, \ldots, \delta_{j,N}\}$.

3.2.3. Task Execution Energy Consumption Model. Through the research model of computing offloading and resource allocation based on active buffering proposed by Zhao et al. [10], the cross-platform task processing model of the MEC is assumed for the research content in this paper. It is shown in Figure 3.

Assuming the SCN studied in this paper has a total of N cells, when the user migrates the task $C_i(A_i, B_i, D_i)$ to the MEC platform on the cell, the cached resource $\mathrm{Rs}_{(1,1)}$ of the current MEC platform is detected first ($\mathrm{Rs}_{(1,1)} = \mathrm{Rs}_{j=1}$). If

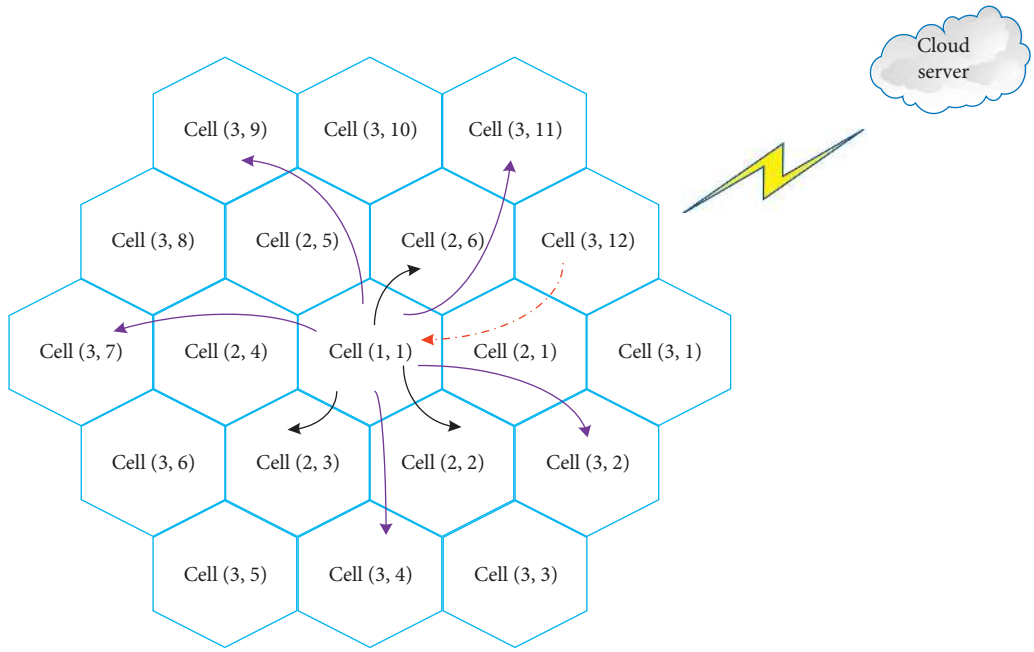


FIGURE 3: SCN task cross-platform execution model.

$\mathrm{Rs}_{(1,1)} = B_i$ and the platform is idle, the task is processed. If not, the device continues to detect the resource information on the MEC platform from $\mathrm{cell}_{(2,1)}$ as shown by the arrow in Figure 3, where

$$\begin{cases} \mathrm{nnN} = 3k(k-1)+1 \\ \mathrm{nnj} = 3k(k-1)(k-2)+h+1 \end{cases} \quad (k \in [1,3],\ h \in [1, 6(k-1)]). \tag{11}$$

If $\mathrm{Rs}_{(k,h)} = B_i$, the task $C_i(A_i, B_i, D_i)$ will perform immediately, and the result will be returned after the processing is finished. Otherwise, the MEC platform continues to detect the next cell. If $\mathrm{Rs}_{(k,h)} \neq B_i$, that is, the mobile device does not find the resource matching platform after detecting the NTH cell, the MEC platform of $\mathrm{cell}_{(1,1)}$ will download the required resource B_i from the SBS and return the resource to the device.

According to formula (1), the energy consumption of MEC platform execution is $P_{\mathrm{meci}}\mathrm{nT}_{\mathrm{mec\_exe},i}$. Task $C_i(A_i, B_i, D_i)$ must incur additional network overhead if it is not processed on the local MEC platform. According to the cell distribution of SCN, the extra cost of cross-platform processing of tasks is related to the location of $\mathrm{cell}_{(k,h)}$; the farther away from $\mathrm{cell}_{(1,1)}$, the greater the cost. The cross-platform execution cost α is proposed to replace the complex cross-platform processing process in this paper.

Therefore, the execution energy consumption of the task is derived as follows:

$$(1+\alpha)^{k-1} P_{\mathrm{meci}} T_{\mathrm{mec}_{\mathrm{exe}},i} = (1+\alpha)^{k-1} P_{\mathrm{meci}} \frac{D_i}{V_{ci}}. \tag{12}$$

Considering after UE_i detecting all $\mathrm{cell}_{(k,h)}$, if there is no matching resource, it still chooses the MEC platform on $\mathrm{cell}_{(1,1)}$ to complete the task processing. The resource download marker β is proposed to be adopted for description, where

$$\beta = \begin{cases} \mathrm{nn0}\ k \neq 1 \\ \mathrm{nn1}\ k = 1 \text{ and } \mathrm{Rs}_{(1,1)} \neq B_i \end{cases} \quad (k[1,3]). \tag{13}$$

Therefore, the downloading energy consumption of resource UE_i is derived as follows:

$$\beta \mathrm{nP}_{\mathrm{sbs}} T_{\mathrm{sbs}_{\mathrm{off}},i} = \frac{\beta n\sigma\left(2^{R/W}-1\right) l^2 D_i}{\lambda^2 V_s}. \tag{14}$$

The total energy consumption of E_{UEi} task migration is derived as follows:

$$\begin{aligned} E_{\mathrm{UEi}} &= \mathrm{nE}_d + \frac{\sigma\left(2^{R/W}-1\right) {d_i}^2 A_i}{\lambda^2 R} \\ &\quad + (1+\alpha)^{k-1} n P_{\mathrm{meci}} n\left(\frac{D_i}{V_{\mathrm{ci}}} + \frac{D_i}{R}\right) \\ &\quad + \frac{\beta n\sigma\left(2^{R/W}-1\right) l^2}{\lambda^2} n\left(\frac{D_i}{V_s} + \frac{D_i}{R}\right). \end{aligned} \tag{15}$$

Further simplification can be obtained as follows:

$$\begin{aligned} E_{\mathrm{UEi}} &= \mathrm{nE}_d + \frac{\sigma\left(2^{R/W}-1\right)}{\lambda^2} n\left(\frac{{d_i}^2 A_i}{R} + \beta l^2\left(\frac{D_i}{V_s} + \frac{D_i}{R}\right)\right) \\ &\quad + (1+\alpha)^{k-1} n P_{\mathrm{meci}} n\left(\frac{D_i}{V_E} + \frac{D_i}{R}\right). \end{aligned} \tag{16}$$

Let the energy consumption $E_{\text{pro}} = \sigma(2^{R/W} - 1)/\lambda^2 n((d_i{}^2 A_i/R) + \beta l^2((D_i/V_s) + (D_i/R))) + (1+\alpha)^{k-1} \text{nP}_{\text{meci}} n((D_i/V_{\text{ci}}) + (D_i/R))$, so $E_{\text{UEi}} = \text{nE}_d + E_{\text{pro}}$.

4. Task Processing Strategy with Energy Consumption Optimization

4.1. Construction of Optimal Stopping Rule Problem for Minimizing Energy Consumption. In the research model in this paper, the mobile terminal can obtain the resource information Rs_j cached on the MEC platform of the surrounding cell through random detection, and the maximum number of detections is N. Assume the detection number when the mobile terminal stops detection is $n, n \in N$, where $N = \{N: 1 \le n \le N\}$ is the collection of detection times.

Previous research on task migration showed that after the mobile terminal migrates the task, the MEC platform of the current cell downloads and processes the resources required for the task. In this paper, the cross-platform task processing is carried out by looking for MEC with other SCN cells' required resources. Because of the extra cost of cross-platform processing tasks, this paper comprehensively considers each cell MEC's energy consumption to select a platform with lower overall energy consumption to achieve task processing requirements.

If the current detected platform resources are not required for task processing or the mobile terminal is not satisfied with the task processing performance, it can continue to detect. So the expected payoff for the cost of migration is given as follows:

$$Y_N = E_{\text{pro},n-1} - E_{\text{pro},n} - nE_d. \tag{17}$$

Let $X_N = E_{\text{pro},n-1} - E_{\text{pro},n}$, then the goal of optimal energy consumption migration is to maximize the expected return. It is given as follows:

$$\max E[Y_N] = \max E[X_N - \text{nE}_d]. \tag{18}$$

The purpose of the maximization problem (19) is to obtain the optimal comprehensive cost of dynamic migration by finding the optimal stop detection times N^* to minimize the energy consumption of service migration. Therefore, the problem of the device choosing the optimal migration platform is transformed into the problem of the optimal stopping rule, and the optimal stop detection number N^* is obtained as follows:

$$N^* = \text{argsup}_{N \in N^*} E[Y_N]. \tag{19}$$

4.2. Proof and Solution of Optimal Stopping Rules. To solve the optimal stopping rule problem of optimal energy consumption of migration task, we must first prove an optimal stopping rule for the problem and then solve the optimal stopping rule problem.

Proposition 1. *Formula (18) has an optimal stopping rule.*

Proof. According to the literature [25], if the optimal stopping rule of the proposition exists, it must meet the following two conditions:

$$\begin{aligned} &A1. E[\sup_n Y_n] < \infty, \\ &A2. \lim_{n \longrightarrow \infty} \sup Y_n \le Y_\infty a.s. \end{aligned} \tag{20}$$

According to the definition of formula (17), the expected reward Y_N that stops at the NTH detection can be expanded as

$$\begin{aligned} Y_N &= X_N - \text{nE}_d \\ &= (1+\alpha)^{k-1} n\left[P_{\text{meci}-1} n\left(\frac{D_i}{V_{\text{ci}-1}} + \frac{D_i}{R}\right) - P_{\text{meci}} n\left(\frac{D_i}{V_{\text{ci}}} + \frac{D_i}{R}\right)\right] - \text{nE}_d. \end{aligned} \tag{21}$$

Because α, D_i, and R are fixed values for the device and the sequence, k, P_{meci}, and V_{ci} are a sequence of random variables with a limited range, so $Y_N < \infty$. So $E[\sup_n Y_n] < \infty$ is established, and condition $A1$ is satisfied. When $n \longrightarrow \infty$, $\text{nE}_d \longrightarrow \infty$, so $Y_N = X_N - \text{nE}_d \longrightarrow -\infty$. Obviously, $Y_\infty \longrightarrow -\infty$, so $\text{limsup}_{n \longrightarrow \infty} Y_n \le -\infty = Y_\infty$, and condition $A2$ is satisfied. In summary, formula (18) satisfies conditions $A1$ and $A2$; therefore, the optimal stopping rule exists.

According to literature [25], when the mobile terminal obtains the optimal expected reward $V^* = \sup_{N \in N^*} E[Y_N]$, the stop detection number N is the solution of the optimal stopping rule question formula (18). Therefore, the optimal number of stop detection N^* is

$$N^* = \min\{N \ge 1: Y_N \ge V^*\}. \tag{22}$$

where the optimal expected reward V^* satisfies the optimal formula as follows:

$$V^* = E[\max(X_N, V^*)] - E_d. \tag{23}$$

Formula (23) compares the detected optimized value X_N with the expected reward V^* and takes the maximum value to obtain the new most desirable reward. In addition, the

optimized value X_N has the same function distribution over all stop detection times N. Then, formula (23) is changed to

$$E_d = E[\max(X_N, 0)]. \quad (24)$$

In SCN, each task can be cross-processed by selecting the cell with the optimal cross-platform processing energy consumption, which can minimize the task migration's energy consumption and improve migration performance. □

4.3. Algorithm Description. In SCN under the MEC environment, UE_i in $cell_n$ will migrate task $C_i(A_i, B_i, D_i)$ to MEC platform in the same cell for processing. When the MEC platform receives the task of migrating, it starts to detect the status information $Sc_i(Rs_i, P_{mec,i})$ of the MEC platform, where Rs_i is the resource information cached in the cell, and $P_{mec,i}$ is the executive power of the MEC platform in the cell.

If the platform cache resource matches the resource required for task processing, the MEC platform obtains the expected energy consumption of the task that was executed on the current platform. If the MEC platform finds that the task's energy consumption is less than or equal to the optimal energy consumption expectation and the state is idle, the MEC platform stops detection and selects this platform for task processing. If the platform cache resource does not match the resource required for task processing, the MEC platform will randomly detect the MEC platform's status information in the next cell. If the MEC platform detects the last cell and still does not find the cell whose expected energy consumption is less than or equal to the optimal energy consumption expectation, then the task will be cross-platform processed by those cells which have cached the resources needed for the task execution and have the lowest expected energy consumption. If the cell detected before does not cache the resources required for task execution, the local MEC platform downloads the resources needed for task execution from the SBS and then processes the task.

In the above Algorithm 1, the bandwidth is w, the transmission rate is R, the cross-platform cost is α, the noise power is σ, the distance and power relation factor is λ, the radius of the cell is l, the layer number of SCN is k, the detection energy consumption is E_d, the number of resource type is Rs_j, the number of device request resource type is B_i, the status of each platform is $status_j$, the number of successful platform matches is op, the flag of resource download is β, the minimum energy consumption is $E_{UEi,min}$, and the detection platform ordinal is $j \in [1, N]$. If the last cell is detected, that is, the detection of N times and the task is not processed, it is the algorithm's worst case. Therefore, the time complexity of this strategy is $O(N)$.

5. Simulation Results and Analysis

This paper proposes the optimal stopping migration strategy (OSTMS) with energy consumption optimization to minimize migration energy consumption. To test this strategy's performance in task migration, this experiment uses a laptop with CPU i5 – 8300H, 2.30GHZ, a memory of 8G, and the operating system of Windows10. We use the MatlabR2016a simulation tool to simulate the proposed migration strategy and compares it with other task migration strategies. This section will compare each strategy from three aspects: average migration energy consumption, average data execution energy efficiency, and average migration distance energy efficiency. The following is a brief description of the two migration strategies used for comparison.

(1) FMTMS (first match of task migration strategy): the mobile device selects the migratable MEC platform that first matches a resource required by the device and is idle for task migration.

(2) FDTMS (first detection of task migration strategy): the mobile device selects the local MEC platform that was first detected for the task migration.

In the simulation experiment carried out in this section, the three optimization strategies' migration performance is tested respectively by taking the task upload rate, detection energy consumption, cell radius, number of resource type, and SCN layer number as variables.

The parameters in the simulation experiment are shown in Table 2.

5.1. Average Migration Energy Consumption. The average migration energy consumption refers to the average value of the total energy consumption of multiple groups of devices during the task migration process. The average migration energy consumption can objectively reflect the devices' total energy consumption in the three strategies. The smaller the average migration energy consumption is, the more effective it is to reduce computing tasks' burden and further optimize migration tasks' performance.

Figure 4 shows the changes in the average migration energy consumption of different variables under the three strategies. We can see from Figure 4. The OSTMS strategy consistently has the optimal migration energy consumption. With the increase of the abscissa parameters, the three strategies also change accordingly. In Figure 4(a), as the transmission rate increases, the gap between OSTMS and FDTMS becomes larger, while the gap between OSTMS and FMTMS becomes smaller. FDTMS is due to its large download power resulting in a large energy consumption gap. FMTMS is because the increase of transmission rate will reduce the gap between data transmission energy consumption of different platforms. In Figures 4(b) and 4(e), the energy consumption of OSTMS is optimized by 10% ~ 60%, while the energy consumption of FDTMS is mostly at a high level. The reason for FDTMS is that it does not consider whether the local MEC platform holds the required resources of the device. If not, the resources need to be downloaded from the SBS, significantly increasing the device's energy consumption. FMTMS does not take into account the differences in execution power and speed between MEC platforms. There is no need to download resources from the SBS when the platform successfully matched that first match, but its high execution power will also increase energy consumption. OSTMS uses the threshold value obtained by the optimal stopping theory to compare with users' energy consumption. Then it selects the

```
Input: w, R, α, σ, λ, l, k, E_d, Rs_j, B_i
Output: E_UEi,min
Begin
  op = 0
    Calculate V* by formula (23)
    for j = 1, 2, 3 . . . , N
      If Rs_j = B_i and status_j = 0
        op = op + 1
        β = 0
        Calculate E_UEi by formula (15)
        If E_UEi ≤ V*
          E_UEi,min = E_UEi
          Break
        else
          E_UEi,min = min(E_UEi)
        else
          If j = N and op = 0
            β = 1
          Calculate E_UEi,min by formula (15)
    end for
End
```

Algorithm 1: An energy consumption optimization strategy for task migration with resource cache

Table 2: Simulation experiment parameter value

Parameters	Describe	Values
W	Bandwidth (MHZ)	20
R	Transmission rate (m/s)	2
α	Cross-platform cost	0.1
σ	The noise power	2
λ	Distance and power relation factor	2
d_i	Distance between MEC and mobile device (m)	50 200
l	Cell radius (m)	200
Rs_i	Number of resource type	6
B_i	Number of device request resource type	4
k	The layer number of SCN	6
E_d	Detect energy consumption (J)	1
P_{mec}	MEC platform power (W)	100 1000

MEC platform with the optimal energy consumption for processing, which has a good performance optimization performance. In Figure 4(c), the energy consumption of FDTMS increases sharply with the increase of cell radius. This is because the distance between the user and the base station increases resulting in greater download power. In Figure 4(d), there are few types of resources in the early stage, so there is a high probability that the local platform can successfully match the resources, so the difference is not big. In the midterm, with the increase of types, the probability of successful matching for each platform is moderate, so OSTMS has higher performance optimization. There are too many types of resources in the later stage, so there is a high probability that unsuccessful platform resource matching will cause resources to be downloaded, so the gap between the three strategies is reduced and the energy consumption is high.

5.2. Average Data Execution Energy Efficiency. In the simulation experiment of multigroup task migration, because of the difference in resource types of each device request, the data volume of device requests is also caused. Therefore, the index of average data execution energy efficiency is introduced for comparison. Its value is the average value of the ratio of the data volume (bit) of multigroup device request resources to the total energy consumption (J) of task migration. Its unit is bit/J, which is the amount of data that the device can migrate per unit of energy consumption. The average data execution energy efficiency better reflects the energy utilization of the devices, and the higher the value, the better the optimization performance of the strategy.

Figure 5 shows the changes in average data execution efficiency under three strategies with different variables. We can see from Figure 5. The OSTMS strategy consistently has the highest average data execution energy efficiency. In Figure 5(a), as the transmission rate continues to increase, the average data execution energy efficiency of OSTMS increases sharply. This is because the larger the transmission rate, the lower the data transmission energy consumption and the greater the data execution energy efficiency under the same amount of data. FDTMS is due to the large download power and large total energy consumption, resulting in its average data execution energy efficiency basically unchanged. In Figures 5(b) and 5(c), the average data execution energy efficiency of OSTMS is stable by about 50% higher than that of FMTMS. The results showed that the data execution energy efficiency of FMTMS and FDTMS was low. Because the two strategies do not consider device energy consumption, they only focus on completing the device's task. Inestimable consequences may occur if the energy load of the device is excessive. OSTMS has a large amount of data migration per unit of energy consumption and a high energy

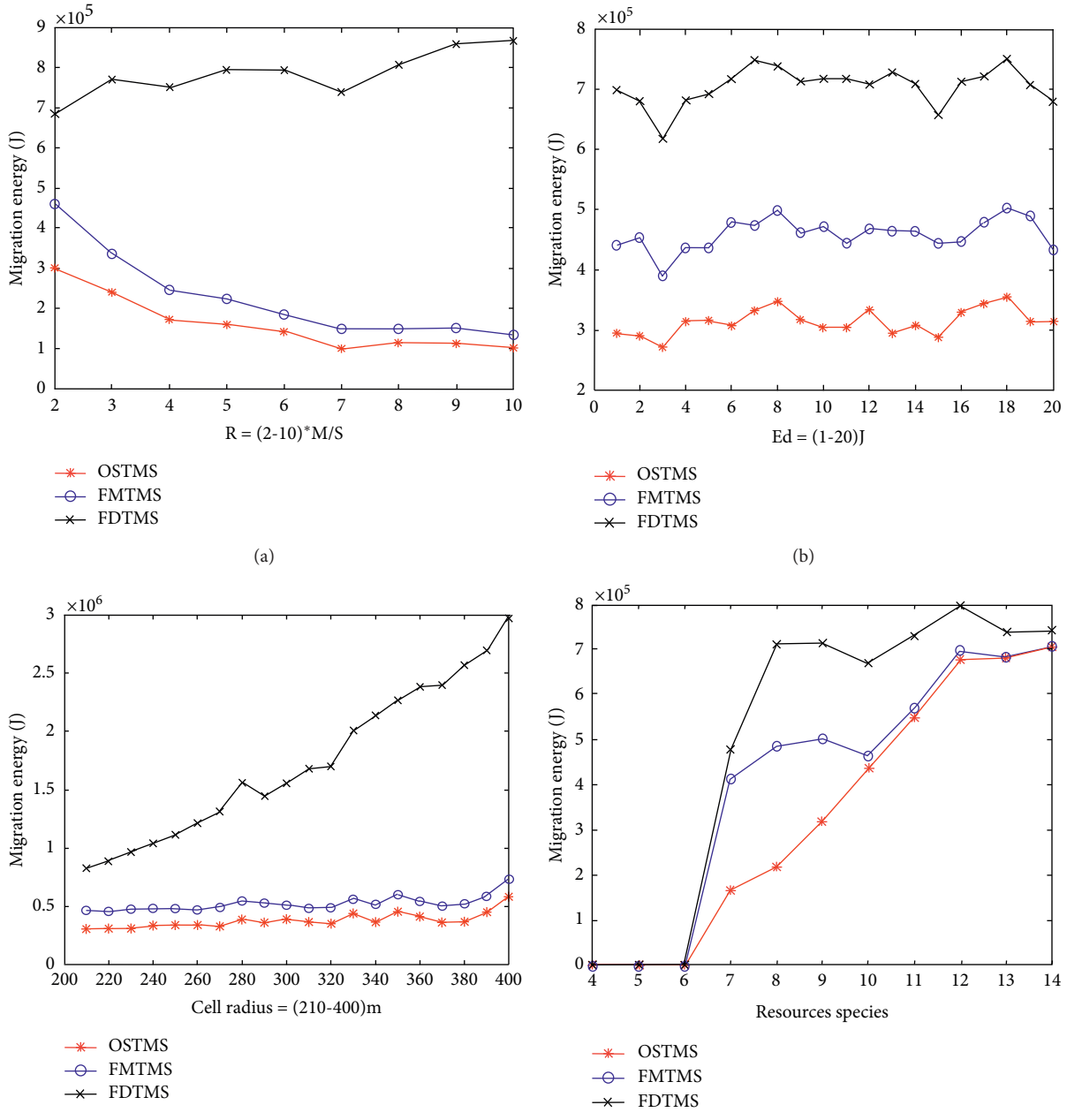
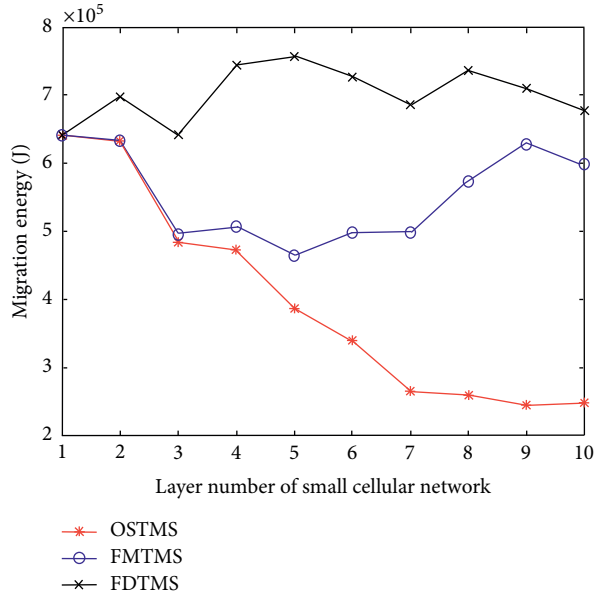
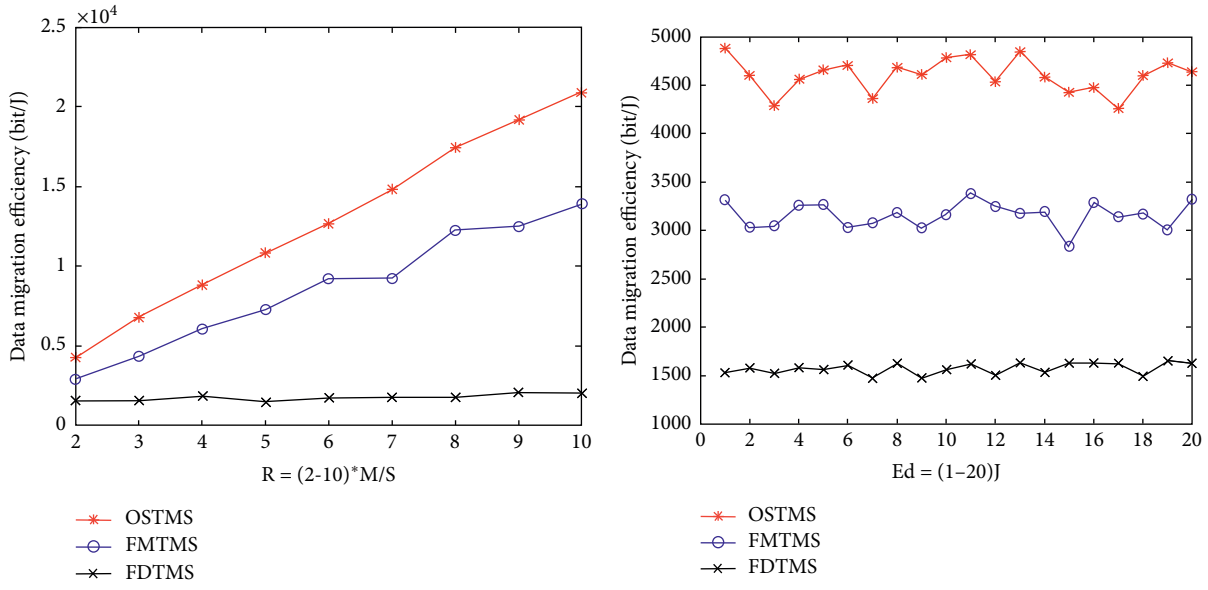


Figure 4: Continued.



(e)

Figure 4: Average migration energy consumption (a) $R = (2-10)^*$ M/S, (b) Ed = (1 – 20) J, (c) cell radius = (210 – 400) m, (d) resources categories, (e) average migration energy consumption.



(a) (b)

Figure 5: Continued.

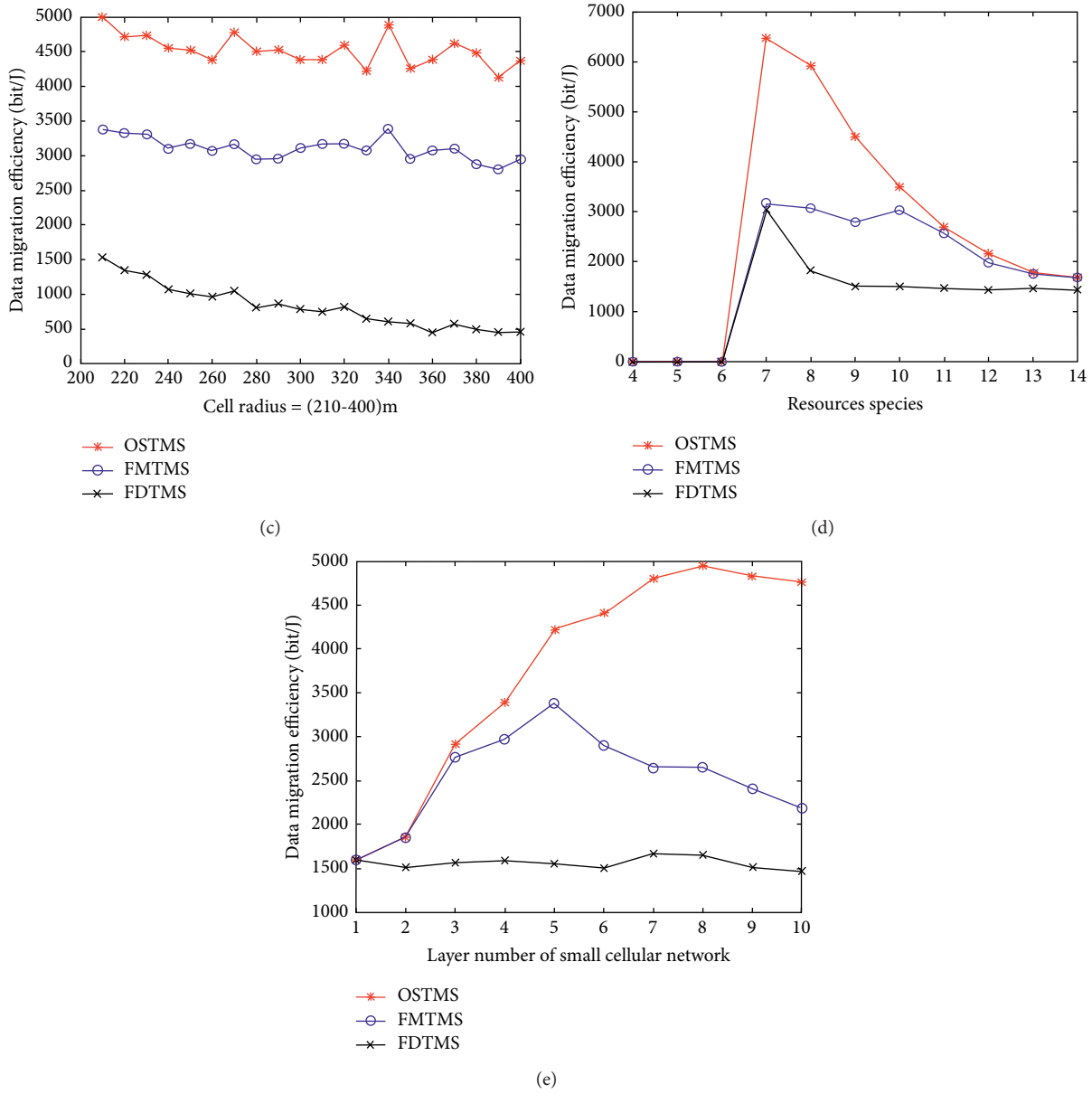


Figure 5: Average data execution energy efficiency. (a) $R = (2-10)^*$ M/S, (b) Ed = (1 – 20) J, (c) cell radius = (210 – 400) m, (d) resources categories, (e) average migration energy consumption.

utilization rate. In Figure 5(d), each platform has a moderate probability of matching success in the medium term, and OSTMS performs less energy consuming so its average data execution efficiency is higher than other strategies. In the later stage, the matching rate of platform resources is low, so the three strategies are similar. In Figure 5(e), as the number of layers in the small cellular network increases, the number of platforms also increases, and the energy consumption of the OSTMS optional platform will be lower, so the gap with other strategies will be greater.

5.3. Average Distance Execution Energy Efficiency. In SCN, the distance between the mobile device and the MEC platform performing the task is different, and the cross-region processing also has to pay a specific cost. Therefore, the use of average distance performance efficiency indicates the degree of performance optimization. Its value is the average of the ratio of the distance between multiple groups of devices and the MEC platform performing the task to the total energy consumption of the task migration. Its unit is m/J. It means the distance that the device can migrate per unit of energy consumption. The higher the value is, the better the optimization performance of the strategy is.

Figure 6 shows the variation of average distance execution energy efficiency under three strategies with different variables. It can be observed from Figure 6 that the OSTMS strategy consistently has the highest average distance execution energy efficiency. It can be seen from Figures 6(a) and

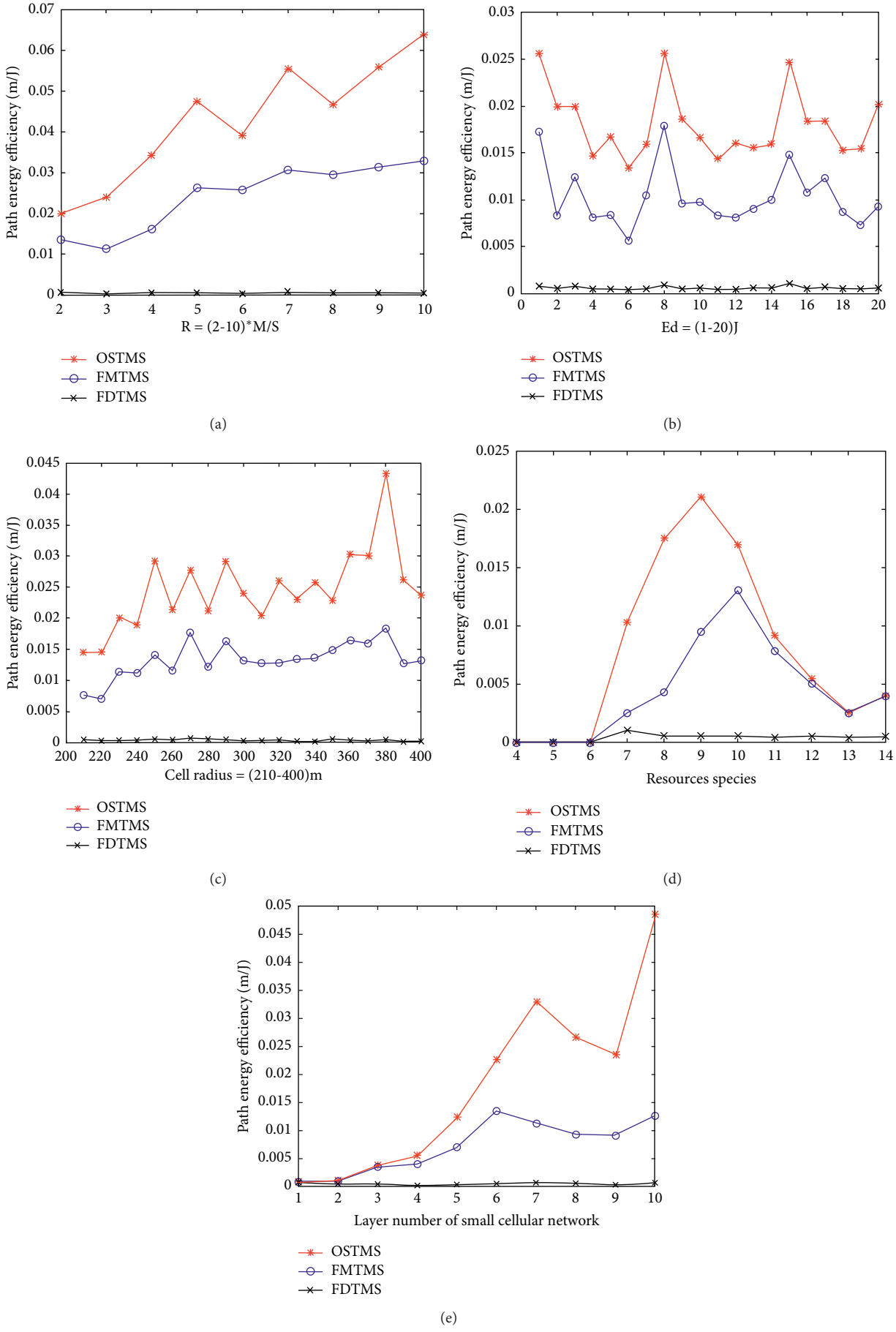


FIGURE 6: Average distance execution energy efficiency (a) $R = (2 - 10)^*$ M/S, (b) Ed = (1 – 20) J, (c) cell radius = (210 – 400) m, (d) resources categories, (e) average migration energy consumption.

6(e) that with the increase of the device task transmission rate and the number of SCN layers, the average distance execution energy efficiency of OSTMS and FMTMS also increases correspondingly, and the optimization degree of OSTMS is higher than FMTMS. However, the FDTMS strategy is always at a low level with little fluctuation. Although the FDTMS is executed on a local platform and the distance between it and the execution platform is minimal, it is always low due to high energy consumption. FMTMS does not take into account the advantages and disadvantages of the platform. In contrast, OSTMS has higher average distance execution energy efficiency. In Figure 6(b), with the increase of detection energy consumption, FDTMS has little fluctuation because it only needs to detect once. While OSTMS has a performance optimization of about 40% compared with FMTMS because it can select the platform with the lowest processing energy consumption through continuous detection. In Figure 6(c), as the cell radius increases, the distance between the user and the execution platform will also increase. The average distance execution energy efficiency of OSTMS and FMTMS also has an upward trend, while FDTMS is relatively stable due to its excessive energy consumption. In Figure 6(d), there are few resource types in the early stage, so the local platform can match resources successfully with high probability, and there is little difference in performance of the three strategies. The performance of OSTMS is much better than that of FMTMS when the resource type is 9 in the middle stage. In the later stage, there are more types of resources, and the success rate of platform resource matching is small, and the performance gap of the three strategies is reduced.

In conclusion, the optimal stopping task migration strategy proposed in this paper can effectively reduce the energy consumption of task migration and optimize task migration performance while ensuring the high performance and high-quality execution of the task.

6. Conclusions and Future Works

This paper studies the energy consumption of resource caching and task migration in small cellular networks. A task migration energy consumption optimization strategy with resource caching is proposed by combining the optimal stopping theory with the migration decision. Firstly, the process of device finding the MEC platform with the required task processing resources is formulated as the optimal stopping problem. Secondly, we prove an optimal stopping rule's existence, obtain the optimal processing energy consumption threshold, and compare it with the device energy consumption. Finally, the platform with the best energy consumption is selected to process the task. The simulation experiment proves that OSTMS has lower average migration energy consumption, higher average data execution energy efficiency, and average distance execution energy efficiency, which realizes the optimization of task migration performance.

In future work, we will consider the following research issues: (1) optimization of time delay during task migration; (2) channel interference during task migration; (3) how to optimize the task migration performance in the scenario of multiple cells and multiple devices; (4) security issues in the process of task migration.

Data Availability

All data generated or analyzed during this study are included in this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundation of China under grant no. 62062007.

References

[1] P. Cong, J. Zhou, L. Li, K. Cao, T. Wei, and K. Li, "A survey of hierarchical energy optimization for mobile edge computing," *ACM Computing Surveys*, vol. 53, no. 2, pp. 1–44, 2020.

[2] B. Shang and L. Liu, "Mobile-edge computing in the sky: energy optimization for air-ground integrated networks," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7443–7456, 2020.

[3] Y. Zhang, "Efficient computation resource management in mobile edge-cloud computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3455–3466, 2018.

[4] H. Li, "Multi-task offloading and resource allocation for energy-efficiency in mobile edge computing," *International Journal of Computer Techniques*, vol. 5, no. 1, pp. 5–13, 2018.

[5] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12313–12325, 2018.

[6] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, 2018.

[7] F. Guo, "Joint load management and resource allocation in the energy harvesting powered small cell networks with mobile edge computing," in *Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Honolulu, HI, USA, April 2018.

[8] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, "Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2410–2422, 2020.

[9] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4947–4963, 2020.

[10] H. Zhao, Y. Wang, and R. Sun, "Task proactive caching based computation offloading and resource allocation in mobile-edge computing systems," in *Proceedings of the 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 232–237, Limassol, Cyprus, June 2018.

[11] Z. Yang, Y. Liu, Y. Chen, and N. Al-Dhahir, "Cache-aided NOMA mobile edge computing: a reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6899–6915, 2020.

[12] R. Zhang, F. R. Yu, J. Liu, T. Huang, and Y. Liu, "Deep reinforcement learning (DRL)-based device-to-device (D2D) caching with blockchain and mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 10, pp. 6469–6485, 2020.

[13] N. Zhang, "Joint task offloading and data caching in mobile edge computing networks," *Computer Networks*, vol. 182, Article ID 107446, 2020.

[14] I. A. Elgendy, W.-Z. Zhang, H. He, B. B. Gupta, and A. A. Abd El-Latif, "Joint computation offloading and task caching for multi-user and multi-task MEC systems: reinforcement learning-based algorithms," *Wireless Networks*, vol. 27, no. 3, pp. 2023–2038, 2021.

[15] K. Peng, "Joint optimization of service chain caching and task offloading in mobile edge computing," *Applied Soft Computing*, vol. 103, Article ID 107142, 2021.

[16] C. Wang, D. Feng, S. Zhang, and Q. Chen, "Video caching and transcoding in wireless cellular networks with mobile edge computing: a robust approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9234–9238, 2020.

[17] Y. Cai: Latency Optimization for D2D-Enabled Parallel Mobile Edge Computing in Cellular Networks," 2021.

[18] Q. Jia, R. Xie, Q. Tang et al., "Energy-efficient computation offloading in 5G cellular networks with edge computing and D2D communications," *IET Communications*, vol. 13, no. 8, pp. 1122–1130, 2019.

[19] I. A. Elgendy, W. Zhang, Y.-C. Tian, and K. Li, "Resource allocation and computation offloading with data security for mobile edge computing," *Future Generation Computer Systems*, vol. 100, pp. 531–541, 2019.

[20] A. Abd El-Latif Ahmed, "Quantum-inspired blockchain-based cybersecurity: securing smart edge utilities in IoT-based smart cities," *Information Processing & Management*, vol. 58, no. 4, Article ID 102549, 2021.

[21] S. Muthanna, "A mobile edge computing/software-defined networking-enabled architecture for vehicular networks," *Internet Technology Letters*, vol. 3, no. 6, Article ID e109, 2020.

[22] S. Manariyo, D. Poluektov, K. Abdukodir, A. Muthanna et al., "Mobile edge computing for video application migration," in *Proceedings of the 19th International Conference, NEW2AN 2019, and 12th Conference, ruSMART 2019*, Petersburg, Russia, August 2019.

[23] I. A. Elgendy, "Advanced deep learning for resource allocation and security aware data offloading in industrial mobile edge computing," *Big Data*, 2021.

[24] H. Wu, "A cache placement strategy for energy savings in CCN," in *Proceedings of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing*, pp. 788–795, Guangzhou, China, October 2018.

[25] T. S. Ferguson, "Optimal stopping and applications," 2006, http://www.math.ucla.edu/~tom/Stopping/Contents.html.