Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 5152598, 15 pages
https://doi.org/10.1155/2022/5152598

Research Article

An Efficient Computing Offloading Scheme Based on Privacy-Preserving in Mobile Edge Computing Networks

Shanchen Pang[1], **Huanhuan Sun**[1], **Min Wang**[2], **Shuyu Wang**[1], **Sibo Qiao**[1] and **Neal N. Xiong**[3]

[1]*Department of Computer Science and Technology, China University of Petroleum, Qingdao 266580, China*
[2]*Department of Control Science and Engineering, China University of Petroleum, Qingdao 266580, China*
[3]*Department of Mathematics and Computer Science, Northeastern State University, Tahlequah, OK, USA*

Correspondence should be addressed to Shanchen Pang; pangsc@upc.edu.cn

Received 29 December 2021; Accepted 18 May 2022; Published 14 June 2022

Academic Editor: Amrit Mukherjee

Computation offloading is an important technology to achieve lower delay communication and improve the experience of service (EoS) in mobile edge computing (MEC). Due to the openness of wireless links and the limitation of computing resources in mobile computing process, the privacy of users is easy to leak, and the completion time of tasks is difficult to guarantee. In this paper, we propose an efficient computing offloading algorithm based on privacy-preserving (ECOAP), which solves the privacy problem of offloading users through the encryption technology. To avoid the algorithm falling into local optimum and reduce the offloading user energy consumption and task completion delay in the case of encryption, we use the improved fast nondominated sorting genetic algorithm (INSGA-II) to obtain the optimal offloading strategy set. We obtain the optimal offloading strategy by using the methods of min-max normalization and simple additive weighting based on the optimal offloading strategy set. The ECOAP algorithm can preserve user privacy and reduce task completion time and user energy consumption effectively by comparing with other algorithms.

1. Introduction

The rapid development of the Internet of things leads to the increasing number of mobile devices and the explosive growth of various new mobile applications. These new types of applications (such as driverless cars, virtual reality, and face recognition) usually require intensive computing with high energy consumption [1–3]. However, the limited computing resources of mobile user equipment (UE) have brought challenges to the operation of new types of applications.

In order to solve the above challenges, mobile edge computing [4, 5] provides cloud computing capabilities for UEs on the network edge. Edge cloud is a cloud computing platform built on edge infrastructure. Edge cloud, central cloud, and IoT terminal form an end-to-end technical framework of cloud edge end three-body cooperation. Because edge cloud computing provides computing and network coverage nearby, the generation, processing, and use of data occur within a very close range from the data source, so receiving and responding to terminal requests have a low delay. For example, edge cloud computing applications in interactive live broadcasting. The media stream of the anchor is pushed to the nearest edge node, transcoded directly at the edge node, and then the transcoded media stream is distributed to the CDN edge node. When there is user access, the content is returned nearby. The services based on edge nodes, the upstream and downstream content push of live streams, and transcoding processing do not need to return to the cloud center, which greatly reduces the service delay and improves the interactive experience. At the same time, the edge processing architecture also saves the bandwidth cost. Resource-constrained UEs can offload tasks to edge servers, so MEC can achieve low latency and

high bandwidth to improve the quality of service and user experience [6].

However, computing tasks need to be offloaded to edge server (ES) through wireless link, which causes additional delay and energy consumption. In addition, ESs have limited resources different from traditional cloud computing centers [7]. Therefore, the offloading decision of computing tasks has become a key issue to achieve efficient offloading [8]. Chen et al. [9] studied the multiuser MEC system under in wireless interference environment, and a distributed efficient computing offloading algorithm has been proposed to achieve the Nash equilibrium. In [10], a new method of user collaborative computing offloading has been proposed to minimize energy consumption under the constraint on computing delay. Compared with [9, 10], the offloading problem was formalized as a multiobjective optimization problem in [11], and they try to find a compromise between delay and energy consumption. However, when UEs offload too many tasks to the same edge server, Chen et al. [9–11] neglected that the edge server may be overloaded, while other servers are in a light load state.

To solve the problem of load imbalance, Wei et al. [12] configured a data buffer for the MEC server to store data that cannot be executed immediately. Similar to [12], the problem of MEC server overload in [13] was solved by setting buffer queues on the mobile device side and the edge server side, respectively. In addition to queuing mobile user requests, we can also choose to reject and postpone user requests to decrease the load of MEC [14]. However, service interruption increases task waiting time and execution time, which reduce the quality of service for users. Therefore, it is critical to maximize system performance for task offloading in ultradense networks and balance the load of MEC servers.

On the other hand, due to the openness of wireless links, the task is prone to be exposed to external threats in the offloading process, which leads to the problem of privacy disclosure. For example, malicious eavesdroppers can eavesdrop on computing data offloaded by IoT devices. Therefore, the confidentiality of privacy is another key issue we need to consider [15]. There are currently two technologies: (1) one is physical layer security technology, which uses the status information about the wireless channel to effectively distinguish between legitimate users and eavesdroppers, to achieve information encryption; (2) the other is data encryption technology, which uses encryption algorithms, and the encryption key turns the plaintext into ciphertext. In this article, considering that the eavesdropper's eavesdropping ability and instantaneous channel state information are difficult to obtain, we use data encryption technology. In [16], a broadcast encryption based on anonymous attributes has been proposed to achieve an efficient and secure data sharing system. In [17], in order to ensure the security requirements of workflow intermediate data, the encryption algorithm and the hash function were sequentially applied to the output data of the task, which enables the implementation of the confidentiality service and integrity service. Xiong et al. and Chen et al. [16, 17] both researched on the security of cloud computing. In [18], the security of mobile edge computing has been considered, and the transmitted data were encrypted to prevent data from being threatened by the external world, but the impact of data size on encryption and decryption time was not considered. For the single server scenario, Wu et al. [19] proposed a joint optimization scheme of data confidentiality and computing offload to minimize the total delay of completing user computing requirements. Different from the above scenario, for the multiuser multicell MEC scenario in this paper, we need to use security services to protect the privacy of users.

To solve the above problems, we propose an efficient offloading method based on privacy-preserving. The main contributions of this paper can be summarized as follows.

(i) We introduce hybrid encryption technology to encrypt the offloaded data to protect user privacy and ensure the confidentiality of transmitted data. This encryption technology combines the encryption advantages of AES and RSA to improve the security and the speed of encryption

(ii) The load mean variance is proposed to evaluate the current load situation of edge servers and avoid overload or light load of some servers

(iii) We propose an improved NSGA-II algorithm (INSGA-II) to reduce the UE energy consumption and task completion delay and improve the system performance by introducing logistic chaotic sequence

The rest of the paper is organized as follows. Firstly, we review the relevant researches in the Section 2. In Section 3, we present the system model and the formation of the problem. In Section 4, we propose an efficient computing offloading method based on privacy-preserving. In Section 5, we present the simulation results. Finally, Section 6 summarizes the paper.

2. Related Work

In recent years, MEC as an emerging technology has attracted more and more attention [20–22], especially the problem of computing offloading of MEC. Most of the existing researches take the delay, energy consumption, weighted sum of energy consumption, and delay as the performance index of computing offloading. For delay-based computational offloading, to obtain the optimal task scheduling strategy, Liu et al. [23] proposed an efficient one-dimensional search algorithm to solve the problem of power constrained delay minimization. Considering the collaborative of MEC and cloud computing, Ning et al. [24] proposed an iterative heuristic resource allocation algorithm for dynamic offloading decisions. To minimize the total completing time of all mobile terminal tasks, Wu et al. [25] designed a computing offloading scheme based on nonorthogonal multiple access (NOMA) technology. Zhang et al. [26] integrated computing offloading, content caching, and resource allocation into one model and designed an asymmetric search tree to minimize the total delay consumption of computing tasks.

Under the constraint on computational delay, there are some researches on the problem of minimizing the total

mobile energy consumption. Chen et al. [27] designed a new communication and computing resource allocation method by clarifying the inherent characteristics of AR mobile applications. Al-Shuwaili and Simeone [28] proposed a joint optimization problem to optimize the total energy consumption of the entire system under delay constraint. Combined with the multiaccess characteristics of 5G, Yang et al. [29] considered the small-cell network architecture for task offloading and modeled the energy consumption of offloading from two aspects of task computing and communication. Wang et al. [30] designed an innovative framework to improve the performance of MEC, based on this framework, an optimal resource allocation scheme has been proposed to optimize total energy consumption of wireless access points.

Computation offloading based on delay and energy consumption is another important research problem [31–33]. To meet the task processing delay and energy consumption constraints on mobile devices, Mashhadi et al. [31] proposed an auction in which edge servers were assigned to mobile devices executed by a pair of neural networks. In [32], to minimize the total overhead of MEC system, an improved genetic algorithm was used to solve the joint optimization problem of computing offload decision and channel resource allocation. Guo et al. [33] solved the problem of MEC offloading in ultradense networks and designed a two-layer game greedy offloading scheme to minimize the total computational overhead of processing time and energy consumption.

It is important to balance the system load to improve system performance. Fakhri et al. [34] proposed a discrete particle swarm optimization algorithm to solve the load balancing optimization problem. In order to balance the load of virtual machines, Tong et al. [35] proposed a new dynamic load balancing task scheduling algorithm based on reinforcement learning and service protocol. In [36], a load balancing algorithm based on autonomous agent has been designed, which presaves the information of candidate virtual machines to improve dynamic load balancing and reduce service time for the cloud environment.

Privacy-preserving is an important issue to be considered for wireless transmission. Aiming at the privacy problem when processing max/min queries in two-layer sensor networks, Yao et al. [37] proposed a privacy protection scheme for max/min queries. The scheme adopts the prefix member authentication method to ensure the privacy of sensitive data stored in nodes. To reduce the risk of user privacy material exposure, Wan et al. [38] proposed an optimized cloud computing security deployment structure and a security mechanism for material protection. In [39], an encrypted data processing and retrieval security solutions were designed to resolve the data security problem in cloud computing.

However, the aforementioned privacy protection and system load issues were designed for cloud computing or mobile cloud computing environments. This limitation has prompted our research to solve the problem of efficient offloading based on privacy protection. In this paper, we consider the privacy protection of data transmission and system load in edge computing environment, which can realize the confidentiality of user privacy, effectively reduce task completion time, and UE energy consumption.

Table 1: Key symbol definition.

Notation	Description
N	Set of N users
M	Set of M MEC servers
Q_m	Set of virtual machines in server m
T_n	Computation task of user n
β_n	Workload for computation tasks T_n
α_n	Data size for computation tasks T_n
$f_{s,m}$	Computing power of virtual machine in server m
$f_{l,n}$	Local computing capability of user n
p_n^{up}	Transmission power of user n
$p_{l,n}$	Local computing power of user n
$g_{n,m}$	Channel gain from user n to edge server m
B	Uplink bandwidth
$t_{l,n}$	Delay of task T_n execution locally
$t_{n,m}^{up}$	Delay for task T_n uploaded to edge server m
$t_{n,m}^{com}$	Delay of task T_n execution on edge server m
t_n^{en}	Encryption time of task T_n
$t_{n,m}^{de}$	Decryption time of task T_n at the server m
$o_{n,m}$	Task offloading strategy, $\forall n \in N, m \in M$
$r_{n,m}$	Rate of from task T_n uploaded to edge server m
rur_m	Resource utilization rate of serve m
VAR	Load balance mean variance rate
q_m	Number of virtual machines in server m

3. System Model and Problem Formation

In this section, we introduce the system model and expounds on the researched problems. Table 1 summarizes the key symbols used in this article.

3.1. System Model. As shown in Figure 1, we consider a MEC system with multicells and servers. $N = \{1, 2, 3, \cdots, N\}$ and $M = \{1, 2, 3, \cdots, M\}$ are used to represent UEs (such as iPad and smart phones) and ESs set in the system, respectively. Each UE has a computing task to complete, and each task is atomic and indivisible. And the capacity of each ES is equal to the number of virtual machines in the ES, and $Q_m = \{1, 2, 3, \cdots, q_m\}$ represents a collection of virtual machines in ES m. Each virtual machine performs only one task at a time, and the computing capability of virtual machines on the same server is the same. The system model is established from four aspects of local computation, MEC offloading computation, system load, and security transmission mode.

3.1.1. The Model of Local Computation. The two-tuple $\{\alpha_n, \beta_n\}$ is used to represent the task T_n of the user n, where

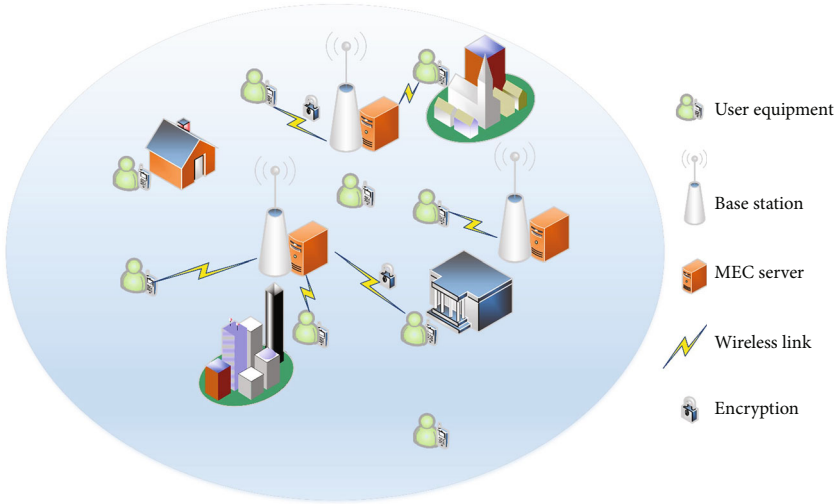

Figure 1: A cellular system with multiple MEC servers.

α_n (bits) represents the data size required to complete task T_n (including input parameters and program codes), and β_n represents the number of cycles required to complete task T_n. The values of α_n and β_n can be obtained by the program analyzer. The binary variable $o_{n,m} \in \{0, 1\}, \forall n \in N, m \in M$, is defined to represent the offloading decision of the task. $o_{n,m} = 1$ indicates that the task T_n is offloaded to the edge server m for execution; otherwise, the task T_n is executed locally. Each computing task can be offloaded to ES execution or executed locally. Therefore, a reasonable offloading strategy needs to meet the limitation:

$$\sum_{m=1}^{M} o_{n,m} \leq 1, \forall n \in N. \tag{1}$$

If $o_{n,m} = 0$, we perform task T_n locally. $f_{l,n}$ represents the local computing capability of user n. The total time to perform task T_n can be calculated as

$$t_{l,n} = \frac{\beta_n}{f_{l,n}}. \tag{2}$$

In order to calculate the energy consumption when the task T_n is executed locally, the energy consumption model $\varsigma(f_{l,n})^2$ in [40] is used, which represents the energy consumption of a calculation cycle, where ς is energy coefficient that depends on the chip structure. Therefore, the energy consumption when the task is executed locally can be calculated as

$$e_{l,n} = \beta_n \varsigma (f_{l,n})^2. \tag{3}$$

3.1.2. The Model of MEC Offloading Computation. When $o_{n,m} = 1$, the task $T_n, \forall n \in N$ is offloaded to the ES $m, \forall m \in M$ to execute. The task T_n offloading calculation includes three steps: (1) task T_n is uploaded to ES m, (2) ES m executes task T_n, and (3) the calculation results is returned to user n. We consider that the downlink transmission rate is much larger than the uplink transmission rate [41], and the amount of data for the calculation result is much smaller than that of the input task, so we ignore the delay of transmitting the calculation results from ES to UE n. Next, the two steps of task upload and task execution will be introduced in detail.

(1) Task Upload. In this paper, we use OFDMA as a multiaccess scheme for uplink. When a subband is occupied by multiple users, it will cause additional interference. Therefore, the signal-to-noise ratio (SNR) from UE n to ES m can be calculated as

$$\chi_{n,m} = \frac{p_n^{up} g_{n,m}}{\sum_{i=1,i\neq n}^{N} \sum_{j=1,j\neq m}^{M} o_{i,j} p_i^{up} g_{i,m} + \sigma^2}, \tag{4}$$

where $g_{n,m}$ represents the uplink channel gain between UE n and ES m, p_n^{up} represents the upload power of user n. The first term of the denominator represents the interference generated by other users on the same subband. The second term σ^2 of the denominator represents the background noise power. Therefore, the upload rate of user n to server m can be calculated as

$$r_{n,m} = B \log_2 (1 + \chi_{n,m}), \forall n \in N, m \in M, \tag{5}$$

where B represents the bandwidth of the uplink. According to (5), the time to upload to ES m can be calculated as follows

$$t_{n,m}^{up} = \frac{\alpha_n}{r_{n,m}}. \tag{6}$$

The energy consumption for user n to upload task T_n to ES m can be calculated as follows

$$e_{n,m}^{up} = p_n^{up} t_{n,m}^{up} = p_n^{up} \frac{\alpha_n}{r_{n,m}}. \tag{7}$$

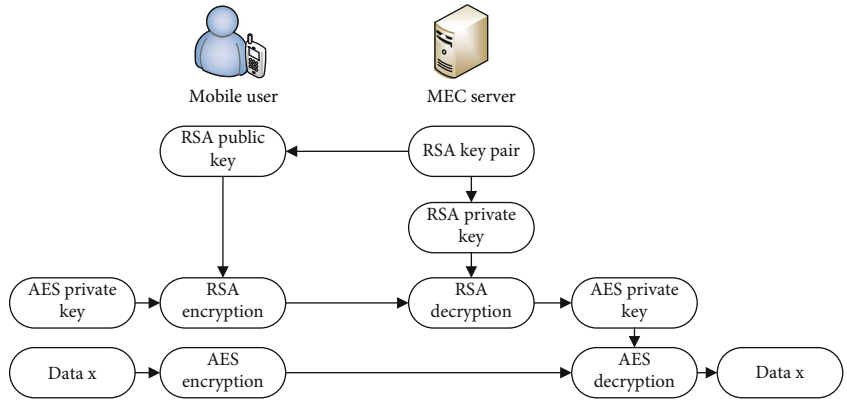

Figure 2: Flow chart of hybrid encrypted data transmission.

(2) Task Execution. The time to perform task is calculated as follows

$$t_{n,m}^{com} = \frac{\beta_n}{f_{s,m}}, \tag{8}$$

where $f_{s,m}$ represents the computing capability of virtual machine in ES m.

3.1.3. Load Model. In this paper, we propose the load mean variance to evaluate the current system load state. According to the occupancy of virtual machines in ES, the resource utilization rate of each ES is calculated as

$$rur_m = \frac{1}{q_m} \sum_{n=1}^{N} \sum_{k=1}^{q_m} o_{n,m} Q_{k,n}^m, \forall m \in M, \tag{9}$$

where $Q_{k,n}^m$ represents the occupancy of virtual machine in ES m and $Q_{k,n}^m = 1$ represents that the kth virtual machine of ES m is occupied by user n; otherwise, $Q_{k,n}^m = 0$. According to (9), the average load of all ESs is calculated as

$$AVG_{rur} = \frac{1}{M} \sum_{m=1}^{M} rur_m. \tag{10}$$

Thus, the load mean variance of the system is calculated as

$$VAR = \sqrt{\frac{1}{M} \sum_{j=1}^{M} \left(rur_j - AVG_{rur}\right)^2}. \tag{11}$$

3.1.4. The Security Transmission Mode. When UE offloads the task to be executed, it is easy to cause privacy leakage in the transmission process. In order to prevent the privacy leakage of the transmitted data, as shown in Figure 2, a hybrid encryption technology based on AES and RSA is used to protect the transmitted data. AES needs to transmit the key from the user to the server, if the key is not encrypted, it will lead to the problem of key leakage. Therefore, RSA is used to encrypt AES key to improve the security of encryption. Since the encryption speed of AES is faster than RSA, we use RSA to encrypt the key with a small amount of data, and AES encrypts the offloading data with a large amount of data, thereby improving the encryption speed.

Each offloading user determines whether to encrypt the transmitted data according to their own security requirements. We use $c_n, n \in N$ to represents UE n's security decision; if $c_n = 1$, it means that UE n encrypts the transmitted data; otherwise, $c_n = 0$. According to [42], the time for encrypting data is calculated as

$$t_n^{en} = c_n \left(\frac{k \times \alpha_n}{f_{l,n} \times v(aes)} \right), \tag{12}$$

where $v(aes)$ is the speed of AES encryption and $f_{l,n}$ is the computing capability of user n. The time to decrypt the offloaded data of MEC server is calculated as

$$t_{n,m}^{de} = c_n \left(\frac{f_{l,n} \times t_n^{en}}{f_{s,m}} \right), \forall n \in N, m \in M. \tag{13}$$

Then the total time of encryption and decryption can be expressed as

$$t_{n,m}^{enc} = t_n^{en} + t_{n,m}^{de}, \forall n \in N, \forall m \in M. \tag{14}$$

The energy consumed by UE to encrypt the offloaded data can be calculated as

$$e_n^{enc} = t_n^{en} \times p_{l,n} = c_n \left(\frac{k \times \alpha_n}{f_{l,n} \times v(aes)} \right) \times p_{l,n}. \tag{15}$$

3.2. Problem Formation. In this section, we will formulate the problem of computing offloading and task completion delay and maximize system performance.

According to (2), (6), (8), and (14), the total delay required to complete all tasks can be calculated as

$$f_T(o_1, o_2, \cdots, o_N) = \sum_{n=1}^{N} \left[\left(1 - \sum_{m=1}^{M} o_{n,m} \right) t_{l,n} + \sum_{m=1}^{M} o_{n,m} \left(t_{n,m}^{up} + t_{n,m}^{com} + t_{n,m}^{enc} \right) \right]. \tag{16}$$

According to (3), (7), and (15), the total energy consumption of UEs to complete all tasks can be calculated as

$$f_E(o_1, o_2, \cdots, o_N) = \sum_{n=1}^{N} \left[\left(1 - \sum_{m=1}^{M} o_{n,m} \right) e_{l,n} + \sum_{m=1}^{M} o_{n,m} \left(e_{n,m}^{up} + e_{n,m}^{enc} \right) \right]. \tag{17}$$

According to (11), the load mean variance of the system can be calculated as

$$f_V(o_1, o_2, \cdots, o_N) = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (rur_m - AVG_{rur})^2}. \tag{18}$$

Therefore, the efficient computing offload problem based on privacy protection is described as a multiobjective optimization problem.

$$\min_{O} [f_T(O), f_E(O), f_V(O)], \tag{19}$$

$$s.t. o_{n,m} \in \{0, 1\}, \forall n \in N, m \in M, \tag{20}$$

$$\sum_{m=1}^{M} o_{n,m} \leq 1, \forall n \in N, \tag{21}$$

$$\sum_{n=1}^{N} o_{n,m} \leq q_j, \forall m \in M. \tag{22}$$

The constraints on the above problem can be interpreted as follows: constraint (20) implies that each computing task can be offloaded to ES or local to execute; constraint (21) states that each task only can be offloaded to one ES; and constraint (22) that the number of tasks performed on each ES cannot exceed the total number of virtual machines on the ES.

4. Our Proposed Efficient Computing Offloading Scheme Based on Privacy-Preserving

In this section, the improved NSGA-II algorithm (INSGA-II) is used to solve the efficient computing offloading problem based on privacy protection, and the optimal offloading strategy set is obtained. Finally, the optimal offloading strategy is obtained by min-max normalization and weighted accumulation.

4.1. Optimize the Efficient Computing Offloading Model Based on Privacy Protection by INSGA-II. In this section, we mainly solve the multiobjective optimization problem (19). We can solve the problem by transforming the multiobjective optimization problem into a single objective problem and set weights for different objectives according to user needs. However, when the user's demands changes, we need to reset the weights and rerun the algorithm. Therefore, we can use the multiobjective optimization algorithm NSGA-II to solve the problem (19). Even if the user's demands changes, there is no need to rerun the algorithm. First, we encode the strategy of task offloading and give the fitness function. Then, we propose an improved NSGA-II algorithm to solve the problem (19). As shown in Figure 3, the basic idea of NSGA-II can be described as follows:

(1) First, initialize a population of size P. And the first-generation population is obtained by selection, cross-over, and mutation of the initial population

(2) Then, from the second generation, $2P$ individuals are obtained by combining the parent population with the offspring population. P individuals are selected from the combined population to form a new parent population by crowding degree calculation and fast nondominated sorting

(3) Finally, a new offspring population is generated through selection, cross-over, and mutation of genetic algorithm

(4) Iteration will stop until the maximum number of iterations are reached, so as to obtain the optimal population

Next, the preparation work and implementation steps of INSGA-II are introduced in detail.

4.1.1. Encoding. Encoding is the first problem solved by NSGA-II algorithm. To solve problem (19), we transform the solution into chromosome embodied in the code. As shown in Figure 4, a solution is designed as a two-tuple, which includes execution Location = $\{1, 0, 1, 1, 0\}$ and Server = $\{4, 0, 7, 2, 0\}$, where the task is offloaded. For Location, if task is executed in ES, the value is 1; otherwise, the value is 0. For Server, its value represents the server number to which task is offloaded, and the value is 0 if task is executed locally.

4.1.2. Fitness Function. In the process of finding the best individual, the fitness function is used to evaluate the quality of the individual. We use Equations (16)–(18) as fitness functions to express task completion time, total energy consumption, and load mean variance, respectively. Our goal is to find an offloading strategy to make the values of the three fitness functions relatively good.

4.1.3. Initialize the Population. Under the constraint of decision space, the initial population with size P is randomly generated. Based on the ergodic characteristics of chaotic sequences [43], we introduce the chaotic sequences to initialize the population to improve the global optimization

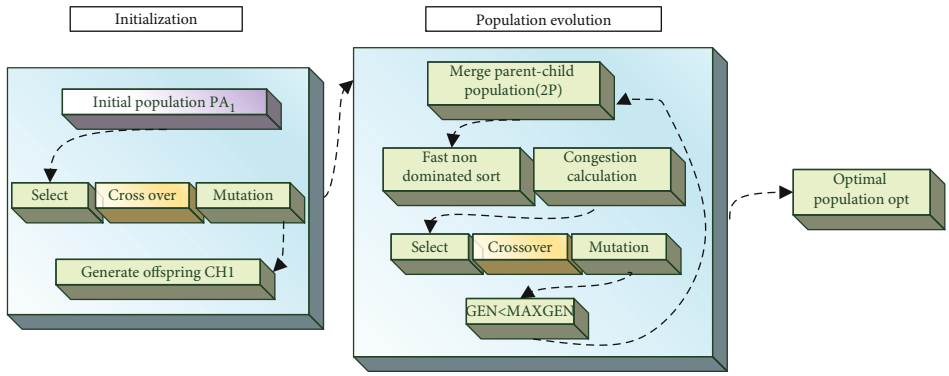

Figure 3: The INSGA-II algorithm flow chart (Algorithm 1 improves the initialization population operation. Algorithm 2 improves the cross-over operation).

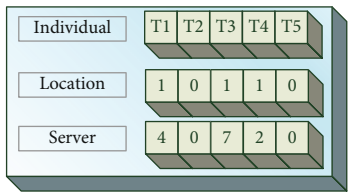

Figure 4: Encoding scheme.

ability and avoid the search process falling into local optimization. Algorithm 1 gives the specific steps of population initialization. Iterate for P individuals in the population (lines 1-2). Firstly, for each individual, the logistic chaotic map $x(n+1) = 4x(n)(1-x(n))$ is iterated N times to generate chaotic sequence Y_i (lines 3-6). Then, the initial value of the current individual i is obtained according to Y_i (lines 7-14). Incorporate the initialized individuals into the initial population set (lines 15-17). Finally, the above process is iterated P times to generate an initial population PA_0 with a size of P individuals.

4.1.4. Fast Nondominated Sorting and Crowding Calculation. In order to retain the best offloading strategy, we merge the offspring and the parent with population size of P and select the best P individuals as the new parent population by fast nondominated sorting and crowding calculation. The specific steps are described as follows: (1) Firstly, a new population PC_i with population size of $2P$ is obtained by combining parent PA_i with offspring CH_i, where $i \in (0, Gen)$, and Gen is the total number of evolutions. (2) According to the fitness functions (16)–(18), the individuals in population PC_i is arranged in fast nondominated sorting. (3) To ensure the diversity of individuals, we calculate the crowding degree of individuals in same dominant layer according to formula (23), where f_j is the j-th fitness function and i_d is crowding degree. (4) P individuals are selected to form a new parent population PA_{i+1} by crowding degree calculation and fast nondominated sorting.

$$i_d = \sum_{j=1}^{F} \left(f_j(i+1) - f_j(i-1) \right) / \left(f_j^{\max} - f_j^{\min} \right). \quad (23)$$

4.1.5. Selection. The tournament selection algorithm is used for selection operation. Firstly, k $(k < p)$ individuals are randomly selected from the P individuals in parent population. Then, the individuals with best fitness value are selected to enter next generation population. The above process is repeated until new P individuals are obtained.

4.1.6. Cross-Over and Mutation. Cross-over operation refers to the operation of replacing and reorganizing some structures of two parent individuals according to the cross-over probability to generate new individuals. Cross-over operation is the main operator to generate new individuals. As an auxiliary operator, mutation operation is to generate new patterns. Assuming that there is only cross-operation, the new solution generated in the iterative process can always only be the combination of existing patterns in the initial population. If the key modes of constructing the optimal solution are missing in the initial population, the optimal solution cannot be obtained only through cross-operation, and we also need to use the local random search ability of mutation operator to accelerate the convergence to the optimal solution. Therefore, both cross-over and mutation operations are indispensable. Next, we will describe these two operations, respectively.

(1) Cross-Over. Cross-over can retain the excellent genes left by each evolution. However, if the two crossed individuals are very similar, it will be difficult to produce new individuals, thus reducing the diversity of the population.

In order to solve this problem, the individual similarity judgment is introduced. In Algorithm 2, we give the specific process of cross-over operation based on similarity judgment. First, traverse P individuals in the population (lines 1). Generate a random number P_{cri} (lines 2) for individual i in the current iteration. If P_{cri} is less than the cross-over probability P_{cr}, add the current individual to the cross-over individual set (lines 3-7). Then, traverse the cross individual set (lines 8) and calculate the similarity between the two crossed individuals according to Equation (24) (lines 9). If the similarity is less than the similarity threshold P_θ, perform the cross-over operation (lines 10-13). The new population is obtained by the above cross-over operation. An

Input: Initial values of logistic chaotic map y_1; Number of iterations N; Population size P.
Output: first-generation population PA_0.
1: **for** $i = 1$ to P **do**
2: $Y_i \longleftarrow y_1 \cup Y_i$
3: **for** $j = 2$ to N **do**
4: $y_j = 4 \times y_{j-1} \times (1 - y_{j-1})$
5: $Y_i \longleftarrow y_j \cup Y_i$
6: **end for**
7: **for** $j = 1$ to N **do**
8: **if** $Y_i(j) \geq 0.5$ **then**
9: T_j is offloaded to MEC server, $o_j = 1$
10: **else**
11: T_j is executed locally, $o_j = 0$
12: **end if**
13: $O_i \longleftarrow o_j \cup O_i$
14: **end for**
15: $y_1 = Y_i(N)$
16: $PA_0 \longleftarrow O_i \cup PA_0$
17: **end for**

Algorithm 1: Population initialization.

Input: cross-over probability $p_{cr} = 0.8$; similarity threshold p_θ.
Output: Individuals after crossing *newCR*.
1: **for** $i = 1$ to P **do**
2: P_{cri}= Math. random
3: **if** $P_{cri} < P_{cr}$ **then**
4: $CR = CR \cup O_i$
5: $count++$
6: **end if**
7: **end for**
8: **for** $j = 1; j \leq count; j += 2$ **do**
9: Calculate $Sim(CR_j, CR_{j+1})$using (24)
10: **if** $Sim(CR_j, CR_{j+1}) < P_\theta$ **then**
11: $newCR$ =cross-over (CR_j, CR_{j+1})
12: **end if**
13: **end for**

Algorithm 2: Cross-overs based on similarity.

example of cross-over operation is given in Figure 5, which performs a single-point cross-over on two individuals.

$$Sim\,(Y_i, Z_i) = \frac{N\text{-}\sum_{i=1}^{N}(Y_i \oplus Z_i)}{N}. \tag{24}$$

4.1.7. Mutation. Mutation breaks through the limitations of the current search and is more conducive to the algorithm to find global optimal solution. Individuals whose mutation probability is less than $p_{mu} = 0.1$ will randomly select a gene for mutation operation. An example of a mutation operation is shown in Figure 6.

We get the offspring CH_g of evolution. Combine offspring CH_g with parent PA_g to form a new parent PA_{g+1}, and continue the evolution of next generation until the maximum evolutionary generation is reached. Solutions with good fitness will spread in the solution set, and solutions with poor performance will be slowly eliminated. Finally, an optimal set OPT of offloading strategies is obtained.

4.2. Get the Optimal Offloading Strategy. In this section, we select an optimal individual from the solution set OPT. Min-max normalization is used to normalize the fitness values to ensure the reliability of results. The total time delay $T = t_l + t_s + t_{enc}$ of individual opt_i to complete all tasks is normalized as

$$T'_{opti} = \frac{T_{opti} - T_{\min}}{T_{\max} - T_{\min}}, \tag{25}$$

where $T_{\max}$, $T_{\min}$, and T_{opti} represent the maximum task completion time, the minimum task completion time, and

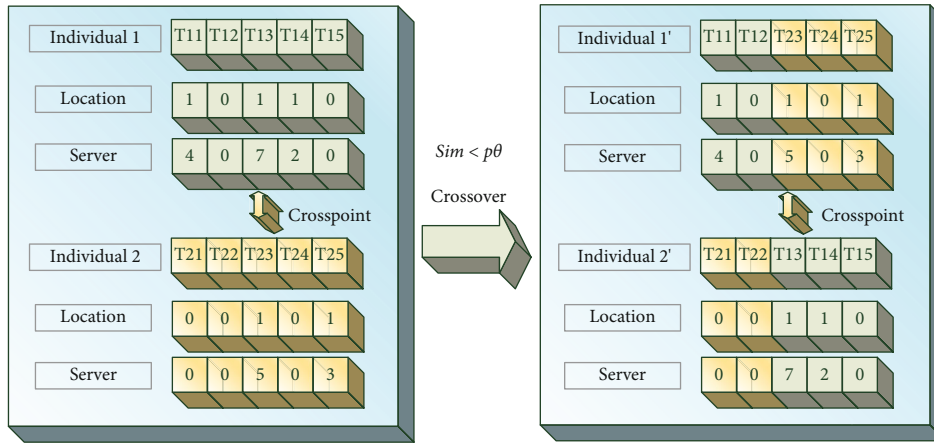

Figure 5: Cross-over operation.

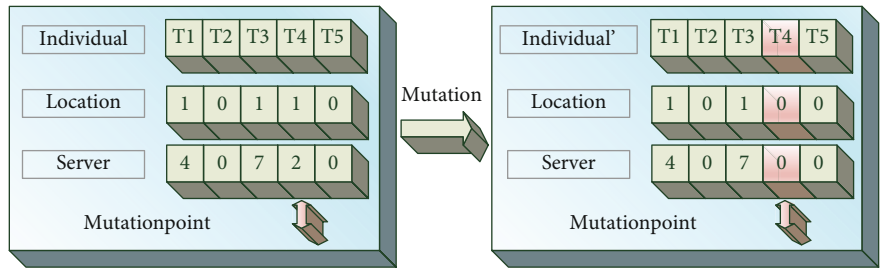

Figure 6: Mutation operation.

the delay for i-th individual to complete task, respectively. The total energy consumption $E = e_l + e_s + e_{enc}$ of UEs is normalized as

$$E'_{opti} = \frac{E_{opti} - E_{\min}}{E_{\max} - E_{\min}}, \tag{26}$$

where $E_{\max}$, $E_{\min}$, and E_{opti} represent the maximum energy consumption of UEs, the minimum energy consumption of UEs, and the UE energy consumption of i-th individual, respectively. Finally, the load mean variance of system is normalized as

$$VAR'_{opti} = \frac{VAR_{opti} - VAR_{\min}}{VAR_{\max} - VAR_{\min}}, \tag{27}$$

where $VAR_{\max}$, $VAR_{\min}$, and VAR_{opti} represent the maximum load mean variance, the minimum load mean variance, and the load mean variance of i-th individual, respectively.

Next, we use simple additive weighting for normalizing fitness values to measure the quality of individuals in population OPT.

$$F(opt_i) = \phi_1 T'_{opti} + \phi_2 E'_{opti} + \phi_3 VAR'_{opti}. \tag{28}$$

where ϕ_1, ϕ_2, and ϕ_3 represent the weight of delay, energy consumption, and load mean variance, respectively. And ϕ_1, ϕ_2, and ϕ_3 satisfies $\phi_1 + \phi_2 + \phi_3 = 1$.

In Algorithm 3, the main process of obtaining offloading strategy based on INSGA-II is introduced. Firstly, the initial population PA1 is generated by Algorithm 1 (line 1). Then, the optimal offloading policy set OPT is obtained by the INSGA-II algorithm (lines 2-10). Finally, the optimal offloading strategy is obtained by the min-max normalization and weighted accumulation (lines 11-18).

5. Performance Analysis

In this section, we evaluate the performance of our proposed ECOAP algorithm through simulation results. The simulation is performed on MATLAB based on simulator 2018. We consider a multiuser multicell scenario, and each cell has a base station. We assume that a single antenna is used for communication between the user and the base station. The parameters used in the simulation are given in Table 2.

To evaluate the performance of our proposed algorithm, we compare it with the following five basic offloading methods.

(i) Offloading based on NSGA-II (NSGA-II): based on the current environment, the NSGA-II algorithm is used to obtain the offloading strategy

(ii) Unsecured offloading (UO): regardless of the privacy-preserving of offloading data, the offloading decision is made in the current setting environment

Input: Evolutionary algebra gen; population size P; cross-over and mutation probability.$p_{cr} = 0.8$, $p_{mu} = 0.1$.
Output: Optimal offloading strategy opt.
1: initial population PA_1 by Algorithm 1
2: g =1
3: **while** $g < gen$ **do**
4: $PC_g = CH_g + PA_g$
5: PA_{g+1} = fast nondominated sort (PC_g) and crowding calculation(PC_g)
6: select(PA_{g+1}) by tournament selection strategy
7: CH_{g+1}= cross-over (PA_{g+1}) by Algorithm 2 and mutation (PA_{g+1}).
8: $g = g + 1$
9: **end while**
10: get O_{opt} by INSGA-II
11: calculate T_{opt}, E_{opt}, VAR_{opt} by $g_T(opt)$, $g_E(opt)$, $f_V(opt)$
12: get $T_{\max}$, $E_{\max}$, $VAR_{\max}$ from T_{opt}, E_{opt}, VAR_{opt}
13: get $T_{\min}$, $E_{\min}$, $VAR_{\min}$ from T_{opt}, E_{opt}, VAR_{opt}
14: **for** $j = 1 : P$ **do**
15: calculate T_{opt}, E_{opt}, VAR_{opt} by min-max standardization
16: calculate $F(opt_j) = \phi_1 T'_{optj} + \phi_2 E'_{optj} + \phi_3 VAR'_{optj}$
17: **end for**
18: $opt = F_{\min}(OPT)$

Algorithm 3: Obtain offloading strategy based on INSGA-II.

Table 2: Summary of key notations.

Parameters description	Value
Number of mobile users N	[5,60]
Number of servers M	7
System total bandwidth	$10MHZ$
Size of computation task T_n	$[300, 600]kb$
Uplink channel gains g	$127 + 30 \log_{10} d_{[km]}$
CPU frequency of servers $f_{s,m}$	$[10, 15]GHZ$
CPU frequency of users $f_{l,n}$	$[0.4, 1]\ GHZ$
Transmission power of users p_n^{up}	$[0.4, 1]W$

(iii) Offloading without considering system load (OWSL): the load problem of ES is not considered, and offloading decision is made based on the current environment

(iv) Local execution (LE): all UE's tasks are executed locally without offloading

(v) All offloading (AO): all UE's tasks are offloaded to ES for execution

(vi) Offloading based on genetic algorithm (GA): based on the current environment, improved GA algorithm is used to solve the optimization problem of offloading decision [35]

5.1. Comparison between the INSGA-II Algorithm and NSGA-II Algorithm. Figures 7–9 show the comparison of average delay, average energy consumption, and average load mean variance between INSGA-II and NSGA-II algorithms. Figure 7 shows that the average delay of the INSGA-II algorithm decreases with the increase in iterations and tends to stabilize when the iteration reaches the 15th generation. However, the NSGA-II algorithm tends to stable until 45 generations, and the average delay of NSGA-II algorithm is higher than that of the INSGA-II algorithm. Similarly, the average energy consumption of the INSGA-II algorithm is lower than that of the NSGA-II algorithm in Figure 8. Figure 9 shows the comparison of the average load mean variance between the two algorithms. When the INSGA-II algorithm and NSGA-II algorithm are iterated to 45 generations, the load mean variance of the INSGA-II

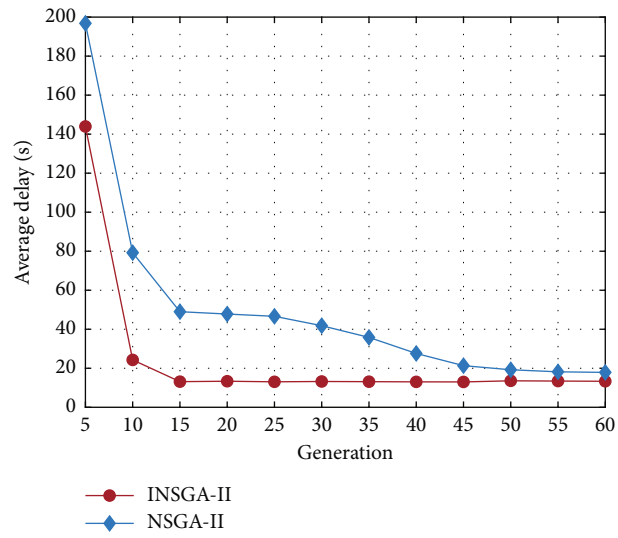

Figure 7: Influence of different evolutionary generations on average delay.

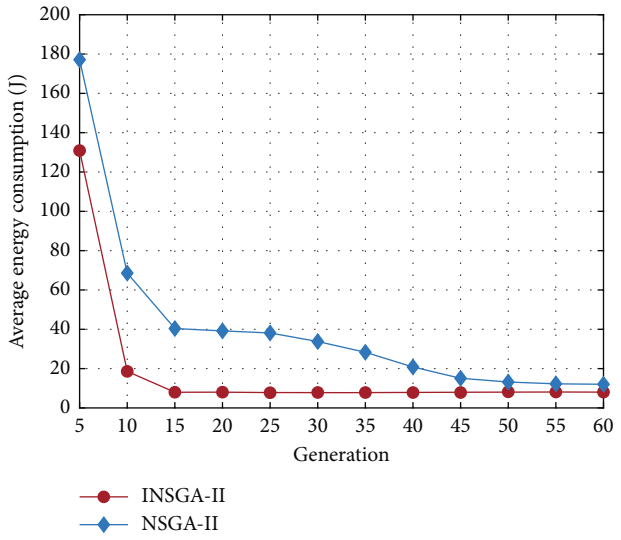

Figure 8: Influence of different evolutionary generations on average energy consumption.

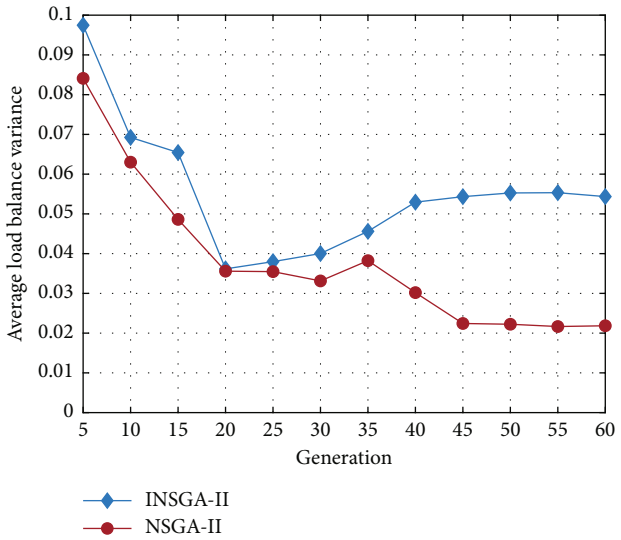

Figure 9: Influence of different evolutionary generations on average load balance variance.

algorithm is better than that of the NSGA-II algorithm. Compared with the NSGA-II algorithm, the INSGA-II algorithm has better global optimization ability, reduces the number of iterations, and requires less time delay and energy consumption.

5.2. Comparison of UEs' Energy Consumption. As shown in Figure 10, the influence of different number of UEs on average energy consumption is described. We compared energy consumption in five different scenarios. It can be seen that with the increase in the number of UEs, the average energy consumption of all schemes are growing. Because ECOAP optimizes the energy consumptions of users, the average energy consumption of ECOAP is relatively small. When the number of users is less than 20, the energy consumptions

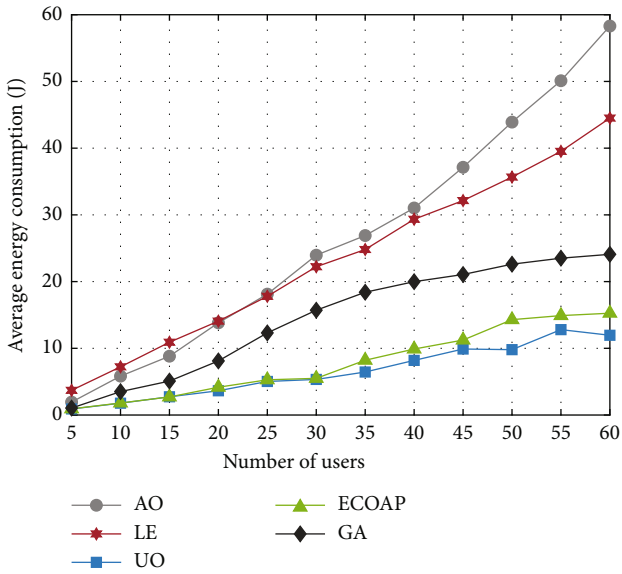

Figure 10: Influence of different numbers of UEs on average energy consumption.

of UO, ECOAP, GA, and AO are less than that of LE, and the energy consumption of UO and ECOAP is basically the same. However, the growth of users leads to additional energy consumption of data transmission, and the energy consumption of AO exceeds the energy consumption of local execution (LE). At the same time, the increase in offloading users leads to the increase of energy consumption for encryption and decryption, and the energy consumption of ECOAP exceeds that of UO.

5.3. Comparison of Tasks' Completion Delay. As shown in Figure 11, the comparison of average delay against different number of UEs is described. We can see that with the growth of UEs, the delays of all schemes are increasing. Because ECOAP optimizes the delay of completing tasks, the average delay of ECOAP is relatively small. When the number of users is less than 20, the delays of UO, ECOAP, GA, and AO is less than that of LE, and the delays of UO and ECOAP are basically the same. When the number of users exceeds 20, the offloading users will compete for limited wireless resources, which results in the delay of AO exceeding the latency of LE. In addition, with the growth of users, the possibility of encrypting offloading data becomes greater, so that the delay of ECOAP exceeds that of UO, but ECOAP increases the confidentiality of offloaded data.

5.4. Comparison of Load Mean Variance. As shown in Figure 12, the comparison of load mean variance against different number of UEs is described. We can see that ECOAP always performs best and load mean variance is the lowest. By optimizing the mean variance of system load, ECOAP can improve the system load to a certain extent and has a good performance in load balancing. However, OWSL does not consider the system load, so some ES may be overloaded or lightly loaded with the growth of users, which will reduce the performance of system.

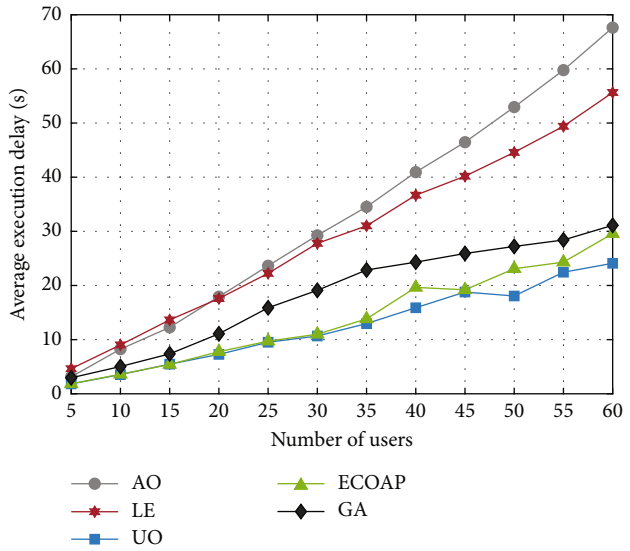

Figure 11: Influence of different number of UEs on average delay.

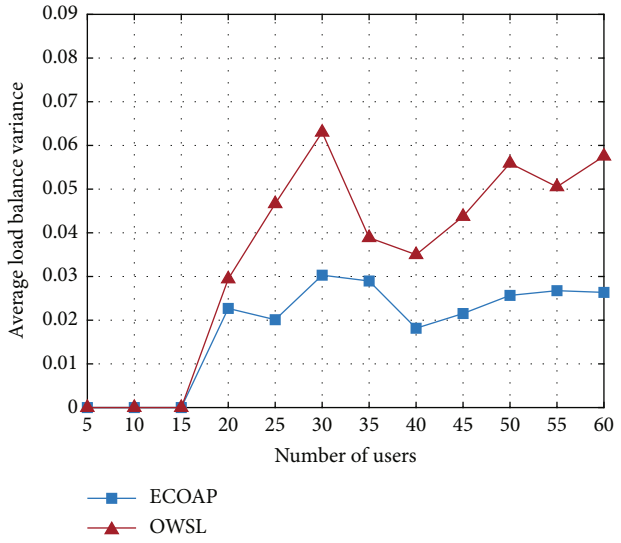

Figure 12: Influence of different numbers of UEs on average load mean variance.

5.5. Influence of UEs' Preferences. Figures 13–15 show the changes of delay, energy consumption, and load mean variance when user preferences vary from 0.1 to 0.9, where $w1$, $w2$, and $w3$ represent user preferences for delay, energy consumption, and system load, respectively. It can be seen from Figure 13 that under the premise that other parameters remain constant, the average delay of task completion decreases with the increase of $w1$. Similarly, we can observe from Figure 14 that the average energy consumption of UEs is decreasing as the weight $w2$ increases. Figure 15 shows that when the user's preference $w3$ for the load average variance increases, the load mean variance is reduced. In addition, the average delay to complete tasks and the average energy consumption of users will increase with the growth of users. This is because the growth of users leads to competition for limited resources, which leads to the increase in time and energy consumption.

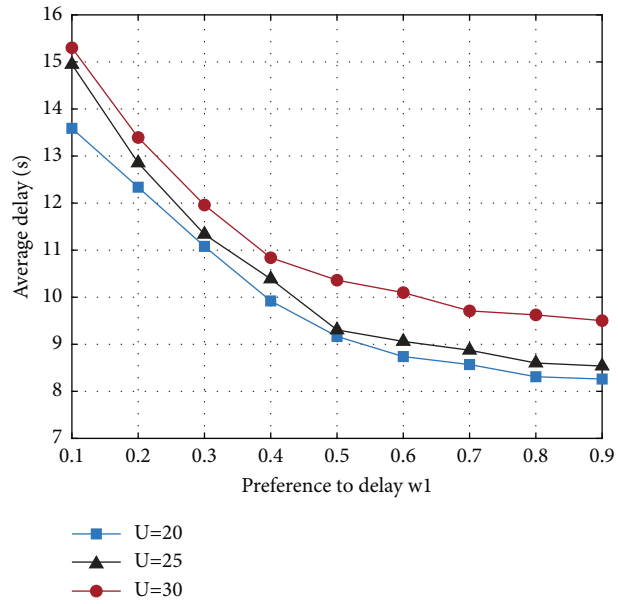

Figure 13: The influence of time preference on delay.

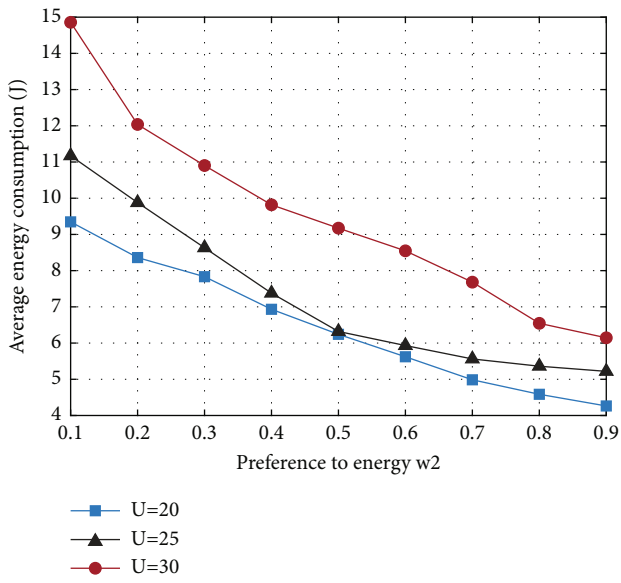

Figure 14: The influence of energy preference on energy consumption.

5.6. Engineering Applications. With the continuous evolution of modern industry towards intelligent direction, the number of industrial field equipment is increasing rapidly, and the demand for computing resources is increasing. As shown in Figure 16, edge computing is widely used in industrial Internet of things (IIoT) environment to provide localized computing resources with low latency and high reliability for factory equipment. Edge computing

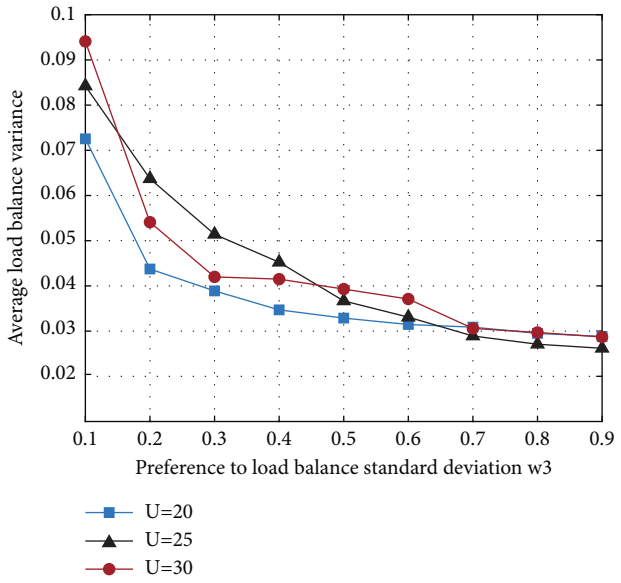

Figure 15: The influence of load preference on load mean variance.

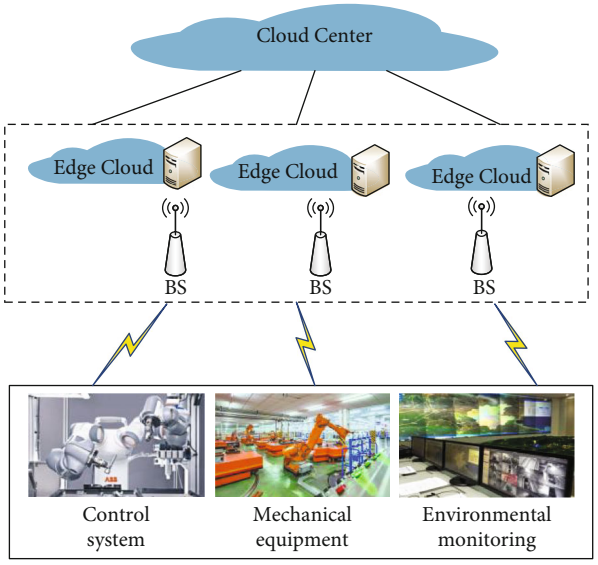

Figure 16: Industrial Internet of things (IIoT) based on edge computing.

distributes computing nodes in the factory production environment, bringing the computing resources closer to factory equipment.

However, in some industrial processes, the requirements for computing delay, energy consumption, and data privacy are particularly high. For example, IIoT may face problems of privacy leakage and risk warning needs real-time response. Therefore, as shown in Figure 17, we use the hybrid encryption method to encrypt the data offloaded from factory equipment to ensure the privacy of the offloaded data. Then, the INSGA-II algorithm is proposed to efficiently offload and obtain the optimal policy set. Finally, the Pareto optimal offloading strategy is selected, so as to

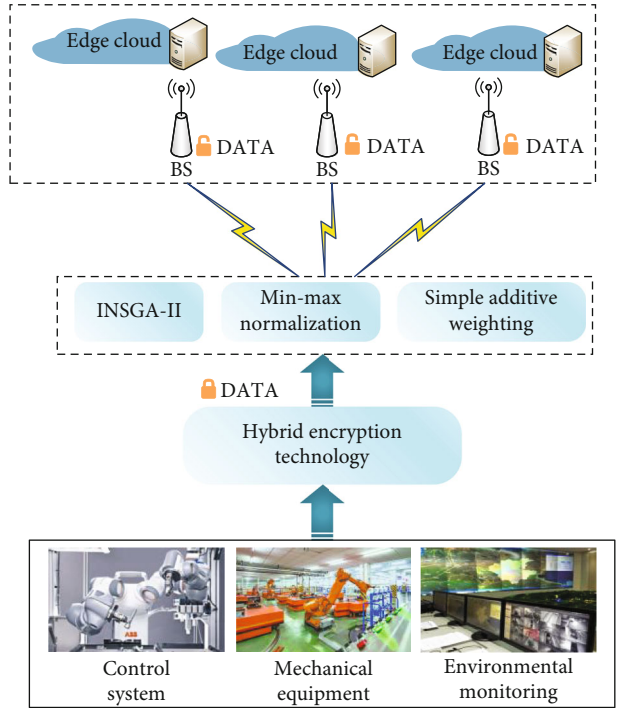

Figure 17: Application diagram of ECOAP algorithm in IIoT.

improve the localized computing services with low delay, low energy consumption, and high reliability for the field equipment of factory.

6. Conclusions and Future Work

In this paper, we propose an efficient computing offloading algorithm based on privacy protection for investigating the privacy protection and task offloading in the multicell MEC network. A hybrid encryption technology is introduced to protect the privacy of offloaded users. The encryption technology combines the advantages of AES and RSA encryption technology to improve the security and speed of encryption. To reduce the UEs' energy consumption and task completion delay in the case of encryption, we propose an improved NSGA-II algorithm (INSGA-II). Simulation results show that the ECOAP algorithm can realize the confidentiality of user privacy and effectively reduce the task completion time and the UEs' energy consumption. In future work, to further reduce energy consumption and improve spectrum efficiency, we will consider using NOMA as multiaccess scheme for uplink. In addition, we will take the mobility of users into consideration to be more in line with actual scenario.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (under Grant 61672033, Grant 61873280, Grant 61873281, Grant 61972416, Grant 61672248, and Grant 61902430), in part by the National Key Research and Development (under Project 2018YFC1406204), in part by the Key Research and Development Program of Shandong Province (under Grant 2019GGX101067), in part by the Natural Science Foundation of Shandong Province (under Grant ZR2019MF012), in part by the Taishan Scholars Fund (under Grant ZX20190157), in part by the Independent Innovation Research (under Project 18CX02152A), and in part by the Fundamental Research Funds for the Central Universities (under Grant 19CX02028A). We appreciate Dr. Neal Xiong for his initial contributions to improve this paper. He helps us reorganize, rewrite, and extend this paper.

References

[1] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: new paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[2] Y. Zhu, Q. He, J. Liu, B. Li, and Y. Hu, "When crowd meets big video data: cloud-edge collaborative transcoding for personal livecast," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 42–53, 2020.

[3] H. Li, J. Liu, K. Wu, Z. Yang, R. W. Liu, and N. Xiong, "Spatio-temporal vessel trajectory clustering based on data mapping and density," *IEEE Access*, vol. 6, pp. 58939–58954, 2018.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] B. Lin, F. Zhu, J. Zhang et al., "A time driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 4254–4265, 2019.

[6] J. Yin, W. Lo, S. Deng, Y. Li, Z. Wu, and N. Xiong, "Colbar: a collaborative location-based regularization framework for QoS prediction," *Information Sciences*, vol. 265, pp. 68–84, 2014.

[7] Y. Qu and N. Xiong, "RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage," in *2012 41st International Conference on Parallel Processing*, pp. 520–529, Pittsburgh, PA, USA, 2012.

[8] Z. Kuang, Z. Ma, Z. Li, and X. Deng, "Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing," *Journal of Systems Architecture*, vol. 118, pp. 1–9, 2021.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[10] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[11] L. Cui, C. Xu, S. Yang, J. Z. Huang, and N. Lu, "Joint optimization of energy consumption and latency in mobile edge computing for internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4791–4803, 2019.

[12] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for internet of things with energy harvesting: a learning method," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4436–4447, 2019.

[13] D. Han, W. Chen, and Y. Fang, "Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints," *IEEE Transactions on Wireless Communications*, vol. 19, no. 6, pp. 3938–3951, 2020.

[14] W. Fan, Y. Liu, B. Tang, F. Wu, and Z. Wang, "Computation offloading based on cooperations of mobile edge computing-enabled base stations," *IEEE Access*, vol. 6, pp. 22622–22633, 2018.

[15] Y. Lu, S. Wu, Z. Fang, N. Xiong, S. Yoon, and D. S. Park, "Exploring finger vein based personal authentication for secure IoT," *Future Generation Computer Systems*, vol. 77, pp. 149–160, 2017.

[16] H. Xiong, H. Zhang, and J. Sun, "Attribute-based privacy-preserving data sharing for dynamic groups in cloud computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2739–2750, 2019.

[17] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 9, pp. 2674–2688, 2017.

[18] 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.

[19] Y. Wu, J. J. Shi, K. J. Ni et al., "Secrecy-based delay-aware computation offloading via mobile edge computing for internet of things," *IEEE Internet Things Journal*, vol. 6, no. 3, pp. 4201–4213, 2019.

[20] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: issues, methods, and perspectives," *ACM Computing Surveys*, vol. 52, no. 1, 2020.

[21] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[22] M. Wu, L. Tan, and N. Xiong, "A structure fidelity approach for big data collection in wireless sensor networks," *Sensors*, vol. 15, no. 1, pp. 248–273, 2015.

[23] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, 2016.

[24] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things," *IEEE Internet Things Journal*, vol. 6, no. 3, pp. 4804–4814, 2019.

[25] Y. Wu, L. P. Qian, K. Ni, C. Zhang, and X. Shen, "Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 3, pp. 392–407, 2019.

[26] J. Zhang, J. Zhang, X. Hu et al., "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things Journal*, vol. 6, no. 3, pp. 4283–4294, 2019.

[27] X. Chen, Y. Cai, L. Li, M. Zhao, B. Champagne, and L. Hanzo, "Energy-efficient resource allocation for latency-sensitive mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2246–2262, 2020.

[28] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[29] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji, "Mobile edge computing empowered energy efficient task offloading in 5G," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6398–6409, 2018.

[30] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1784–1797, 2017.

[31] F. Mashhadi, S. A. S. Monroy, A. Bozorgchenani, and D. Tarchi, "Optimal auction for delay and energy constrained task offloading in mobile edge computing," *Computer Networks*, vol. 183, article 107527, 2020.

[32] C. Du, Y. Chen, Z. Li, and G. Rudolph, "Joint optimization of offloading and communication resources in mobile edge computing," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2729–2734, Xiamen, China, 2019.

[33] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultradense IoT networks," *IEEE Internet Things Journal*, vol. 5, no. 6, pp. 4977–4988, 2018.

[34] Z. H. Fakhri, M. Khan, F. Sabir, and H. S. Al-Raweshidy, "A resource allocation mechanism for cloud radio access network based on cell differentiation and integration concept," *IEEE transactions on network science and engineering*, vol. 5, no. 4, pp. 261–275, 2018.

[35] Z. Tong, X. M. Deng, H. J. Chen, and J. Mei, "DDMTS: a novel dynamic load balancing scheduling scheme under SLA constraints in cloud computing," *Journal of Parallel and Distributed Computing*, vol. 149, pp. 138–148, 2021.

[36] F. Zhao, C. Li, and C. F. Liu, "A cloud computing security solution based on fully homomorphic encryption," in *16th International Conference on Advanced Communication Technology*, pp. 485–488, Phoenix Park, PyeongChang Korea, 2014.

[37] Y. Yao, N. Xiong, J. H. Park, L. Ma, and J. Liu, "Privacy-preserving max/min query in two-tiered wireless sensor networks," *Computers & Mathematics with Applications*, vol. 65, no. 9, pp. 1318–1325, 2013.

[38] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.

[39] G. Soni and M. Kalra, "A novel approach for load balancing in cloud data center," in *2014 IEEE International Advance Computing Conference*, pp. 807–812, ITM University Gurgaon, India, 2014.

[40] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[41] X. Lyu, H. Tian, C. Sengul, and P. Zhang, "Multiuser joint task offloading and resource optimization in proximate clouds," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 4, pp. 3435–3447, 2017.

[42] B. Huang, Z. Li, P. Tang et al., "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 97, pp. 755–774, 2019.

[43] R. M. Yin, J. Wang, J. Yuan, and S. X. Wang, "Weak key analysis for chaotic cipher based on randomness properties," *Science China Information Sciences*, vol. 55, no. 5, pp. 1162–1171, 2012.