Hindawi
Mobile Information Systems
Volume 2021, Article ID 7716654, 17 pages
https://doi.org/10.1155/2021/7716654

Research Article

Computation Offloading Optimization in Mobile Edge Computing Based on HIBSA

Yang Liu, Jin Qi Zhu, and Jinao Wang

College of Computer and Information Engineering, Tianjin Normal University, Tian Jin 300387, China

Correspondence should be addressed to Jin Qi Zhu; zhujinqi1016@163.com

Received 6 August 2021; Revised 3 October 2021; Accepted 13 October 2021; Published 5 November 2021

Academic Editor: Zhiyong Yu

Multiaccess edge computation (MEC) is a hotspot in 5G network. The problem of task offloading is one of the core problems in MEC. In this paper, a novel computation offloading model which partitions tasks into subtasksis proposed. This model takes communication and computing resources, energy consumption of intelligent mobile devices, and weight of tasks into account. We then transform the model into a multiobjective optimization problem based on Pareto that balances the task weight and time efficiency of the offloaded tasks. In addition, an algorithm based on hybrid immune and bat scheduling algorithm (HIBSA) is further designed to tackle the proposed multiobjective optimization problem. The experimental results show that HIBSA can meet the requirements of both the task execution deadline and the weight of the offloaded tasks.

1. Introduction

With the rapid development of the Internet of Things (IoT), intelligent mobile devices (IMDs) have become indispensable tools in people's daily life, and their functions have become more and more powerful, which can meet people's needs in social, shopping, travel, entertainment, and so on. Due to the physical size constraint, mobile devices are usually resource-constrained and have a limited power supply. However, most of the computation-intensive services, such as image processing or video-based applications, need high processing power and have high resources consumption [1, 2]. Compute-intensive tasks cannot be completed in time or may even be blocked if they are only processed locally [3]. Hence, how to solve the contradiction between the limited resources of mobile terminals and the high resources requirement of compute-intensive services has become one of the main problems to be solved [4].

Nowadays, multiaccess edge computation (MEC) [5] has been a promising paradigm to resolve the abovementioned problem [6, 7]. In MEC, an edge site/server is a microdata center, which is deployed attached to a small base station (SBS). By moving computing storage and service capabilities to the network edge, MEC can provide high reliability, high bandwidth, and low-latency computing services for mobile devices. Since mobile terminals can then offload tasks to the nearby edge computing servers with rich computing resources, the problem of resource limitation of IMDs can be resolved to some extent.

Obviously, when multiple IMDs upload tasks at the same time, they will inevitably compete with each other for both communicational and computing resources [8]. Unreasonable resource allocation can result in a low data transmission rate and high delay. Therefore, the designation of the task scheduling scheme has an important influence on the performance of the MEC system. So far, many researchers have focused on the computation offloading scheduling problem. However, most of these studies have performance limitations, which can be explained from the following aspects. Firstly, some researches allocate tasks to only one edge server. However, since the density of SBSs is high [9, 10] in the future and the signal coverages of the SBSs often overlap with each other in real-world scenarios, there are multiple options when unloading tasks. In addition, application partitioning and repartitioning have been studied in depth in mobile cloud computing and distributed systems [11, 12], which can be used in the MEC system. Following these two ideas, assigning tasks to multiple edge

servers is more reasonable. Secondly, some works [13] ignored the energy consumption of the IMDs. In fact, the energy consumption of mobile device must be considered because they usually cannot be recharged timely. Thirdly, most studies did not consider the weight of offloading tasks. However, those tasks that are important or have a long waiting time in the scheduling queue should be scheduled in priority for fairness. Finally, some studies have not jointly considered the allocation of both the communication and computing resources. Compared with the prior works, the contributions of our paper are as follows:

(i) The proposed task offloading model takes communication and computing resources, energy consumption of the IMD, and weight of tasks into account.

(ii) We consider the scenario that the mobile device can generate multiple tasks at the same time, which is more realistic compared with most related works. Moreover, we partition compute-incentive tasks into subtasks and then offload them to multiple edge servers for parallel computing. Compared to offloading a single task to an edge server, tasks can be executed in a more efficient way.

(iii) A novel multiobjective task scheduling algorithm is designed, which combines the advantages of both the bat algorithm and the immune algorithm to improve the reliability of task offloading while reducing the task completion time.

(iv) Extensive simulations have been conducted, and the results show that the proposed algorithm can effectively shorten the task execution time and has higher reliability compared with conventional algorithms.

The rest of this paper is organized as follows. Section 2 presents related works. Section 3 describes the system model and the problem formulation. In Section 4, we transform the offloading decision problem into a multiobjective optimization problem based on Pareto and then design a multiobjective task scheduling algorithm based on hybrid immune and bat scheduling algorithm (HIBSA). Section 5 gives the experimental comparisons of HIBSA with other algorithms, which validate the superior performance of HIBSA. Finally, we make the conclusion in Section 6.

2. Related Work

Task offloading refers to the process of allocating tasks to edge servers with sufficient resources according to some offloading policies. These policies determine both the efficiency and the achievable computation performance of the MEC. Task offloading is also called computing migration or computing offloading [14]. By delivering compute-incentive tasks such as face recognition and video optimization to MEC servers, high task quality of service (QoS) is achieved. In the last few years, task offloading problems have attracted great interest of researchers. For instance, Wu et al. [10] proposed an offloading algorithm based on support vector machine (SVM). The proposed algorithm firstly segments a task into several subtasks by using a weight allocation method. Then, each subtask is determined to be offloaded or executed locally. In [11], Mao et al. developed an online joint radio and computational resource management algorithm for a multiuser MEC system, with the objective of minimizing the long-term average weighted sum power consumption of the mobile devices and the MEC server, subject to a task buffer stability constraint. In literature [12], the authors proposed an efficient computation offloading algorithm by jointly optimizing user association and computation offloading where computation resource allocation and transmission power allocation are also considered. Also, the authors in work [13] proposed a novel offloading system to design robust offloading decisions for mobile services. This system considers the dependency relations among component services and aims to optimize task execution time and energy consumption of mobile devices. These abovementioned researches focus on the computation offloading problem in the single-server MEC system.

On the other hand, many researchers have devoted their efforts to task offloading problems in multiuser and multiserver MEC systems. For example, in [15], a cross-edge computation offloading framework for compute-incentive applications was proposed. The transmission cost, task execution cost, coordination cost, as well as penalty for task failure were considered together in the offloading model designation. An online algorithm based on Lyapunov optimization is proposed to jointly determine edge server site selection and energy harvesting. Work [16] investigated computation offloading in a dynamic MEC system with multiple edge servers, where computational tasks with various requirements were dynamically generated by IoT devices and then offloaded to MEC servers in a time-varying operating environment. The objective of this work is to maximize the task completion time and minimize the energy consumption of IoT devices. In [17], the authors used an improved glowworm swarm optimization algorithm to solve the task offloading problem for a multiuser-multi-MEC environment. Also, the authors in work [18] presented a reinforcement learning framework based on the theory of stochastic learning automata towards enabling the end-users to select an MEC server to offload their data. To realize the proposed framework, an iterative and low-complexity algorithm is introduced and designed. Literature [19] proposed a cooperative offloading technique based on the Lagrangian suboptimal convergent computation offloading algorithm (LSCCOA) for multiaccess MEC in a distributed Internet of Things (IoT) network. However, none of the abovementioned methods considered the weight of the offloaded tasks. In fact, different tasks are of different importance to users.

To indicate the importance of different tasks, the authors in [20] proposed a multiobjective task scheduling algorithm, which aimed to optimize the allocation of the weight of the offloaded tasks. However, this work has the following limitations. First, tasks in this work can be offloaded to an edge server only. Second, the energy consumption of the mobile terminal was ignored. Finally, a bat algorithm was

used in this work to get the optimization result. Since the bat algorithm has no mutation operation, sometimes, the solutions are lack of diversity. Specifically, Table 1 shows the differences between some related studies and our work proposed in this paper.

In this paper, the proposed model takes many aspects of task offloading into account. Moreover, the bat algorithm is combined with an immune algorithm to improve the performance of the bat algorithm and get better optimization results. Meanwhile, the scenario that mobile devices can generate multiple tasks simultaneously is considered. To the best of our knowledge, each device can only generate only one task in the related studies. Hence, our work is more realistic compared with relevant studies.

3. System Model and Problem Formulation

3.1. System Model. Suppose the proposed system consists of n IMDs and m MEC servers. Let $D = \{D_1, D_2, \ldots, D_n\}$ denote the set of IMDs and $\mathrm{CS} = \{\mathrm{CS}_1, \mathrm{CS}_2, \ldots, \mathrm{CS}_m\}$ denote the set of MEC servers. We also discretize time into multiple time slots, and all time slots have equal length as σ.

Among set CS, one edge server in the central location is selected as the controller. The proposed HIBSA algorithm, which is detailed in Section 4, is executed on this controller. In each time slot σ, the IMDs generate computation task requests. Those requests along with the basic information of the IMDs (e.g., app type, local CPU-cycle frequency, and battery energy level) are then sent to the controller. By executing the HIBSA algorithm, the controller chooses the edge server for each IMD for task offloading. The architecture of this system model is shown in Figure 1.

For any IMD $D_i \in D$, there exists a task queue $T^i = (T^i_1, T^i_2, \ldots, T^i_k, \ldots)$, where T^i_k denotes the k-th task generated by the i-th IMD at a certain time slot. For task T^i_k, it can be denoted by four tuples $(\mathrm{dnum}_{ik}, \mathrm{cnum}_{ik}, w_{ik}, \varphi_{ik})$, where dnum_{ik} is the size of the input data for computation, cnum_{ik} is the CPU cycles to be processed for offloading, w_{ik} is the weight of the task, and φ_{ik} is the remaining battery energy value of this IMD when generating this task. All generated tasks can be divided into two types. One is the real-time task, and another is the delay-tolerant task. The real-time task owns a maximum latency and must be finished before the delay threshold whereas the delay-tolerant task can tolerate a much longer delay.

Furthermore, we use a binary matrix $s = \{s_{ij} | s_{ij} \in (0,1)\}_{n \times m}$ to represent one scheduling solution, where $s_{ij} = 1$ denotes that the latest task generated by the i-th IMD is allocated to the j-th server in this scheduling solution, while $s_{ij} = 0$, otherwise.

3.2. Performance Evaluation. For any scheduling solution, its performance is described by a vector (time cost and weight), where time cost is the sum of the time consumption for executing all tasks. Moreover, time cost consists of two parts: one is the time delay for successfully completed tasks, and the other is the punishment time for failed tasks. The time t_{ik} for successfully finishing task T^i_k is as follows:

$$t_{ik} = t_{comm} + t_{comp} + t_{coor} + t_{local}, \tag{1}$$

where t_{comm} is communication cost and is denoted as

$$t_{comm} = \max_{j \in CS, s_{ij}=1} \frac{\mathrm{dnum}_{ik}}{C_{ij} \cdot \sum_{j \in CS} s_{ij}}, \tag{2}$$

where C_{ij} is the approximate data rate between the i-th IMD and the j-th edge server, based on the 3GPS TS 38.306; the transmission rate is as follows:

$$C_{ij} = 10^{-6} \cdot \sum_{k=1}^{K} \left\{ v^{(k)}_{Layers} \cdot Q^{(k)}_m \cdot f^{(k)} \cdot R_{\max} \cdot \frac{N^{BW(k),\mu}_{PRB} \cdot 12}{T^{\mu}_S} \cdot \left(1 - \mathrm{OH}^{(k)}\right) \right\}, \tag{3}$$

where K is the number of aggregated component carriers (CC) in a band or band combination between the i-th IMD and the j-th edge server. $R_{\max} = 948/1024$. For the k-th CC, $v^{(k)}_{Layers}$ is the maximum number of layers; $Q^{(k)}_m$ is the maximum modulation order; $f^{(k)}$ is the scaling factor; μ is the numerology; T^{μ}_S is the average OFDM symbol duration in a subframe for numerology μ; $N^{BW(k),\mu}_{PRB}$ is the maximum RB allocation in bandwidth $BW^{(k)}$ with numerology μ, where $BW^{(k)}$ is the supported maximum bandwidth in the given band or band combination between the i-th IMD and the j-th edge server; and $\mathrm{OH}^{(k)}$ is the overhead. t_{comp}, which is the time for computing this offloaded task, is given as

$$t_{comp} = \max_{j \in CS, s_{ij}=1} \frac{\mathrm{cnum}_{ik}}{f_j \cdot \sum_{j \in CS} s_{ij}}, \tag{4}$$

where f_j is the CPU cycle frequency of the j-th edge server and t_{coor} is the coordination cost between multiple servers and is calculated as follows:

$$t_{coor} = u_{lc} \cdot \sum_{j \in CS} s_{ij}, \tag{5}$$

where u_{lc} is the unit latency cost and t_{local} is local execution time that includes data preprocessing time and data packing time. t_{local} is given as

$$t_{local} = \frac{c_{ik}}{f_i}, \tag{6}$$

where c_{ik} is the amount of CPU cycles to process the local execution and f_i is the CPU cycle frequency of the i-th IMD. Besides, t_{ik} should satisfy

Table 1: The differences between several references and our work.

Related works	Problem formulation	Optimization objectives	Algorithm proposed
[15]	Markov decision process (MDP)	Minimize the overall delay cost of all mobile devices subject to some constraints	The sampling and classification (SAC) based edge site selection (SES) algorithm
[18]	A two-layer optimization framework	Maximize its profit by processing the end-users' data for each MEC server while maximizing its perceived satisfaction for each end-user	Data offloading and MEC server selection (DO-MECS) algorithm
[19]	Single-objective optimization problem with multiple constraints	Lessen the weighted amount of power consumed by communicating devices subject to some constraints	The Lagrangian suboptimal convergent computation offloading algorithm (LSCCOA)
[20]	Multiobjective optimization problem with multiple constraints based on Pareto	Minimize the total execution time and maximize the total weight under the constraints of communication and computing resources	Bat algorithm
Our method	Multiobjective optimization problem with multiple constraints based on Pareto	Minimize the total execution time and maximize the total weight under the constraints of communication, computing and energy resources	Hybrid immune and bat scheduling algorithm (HIBSA)

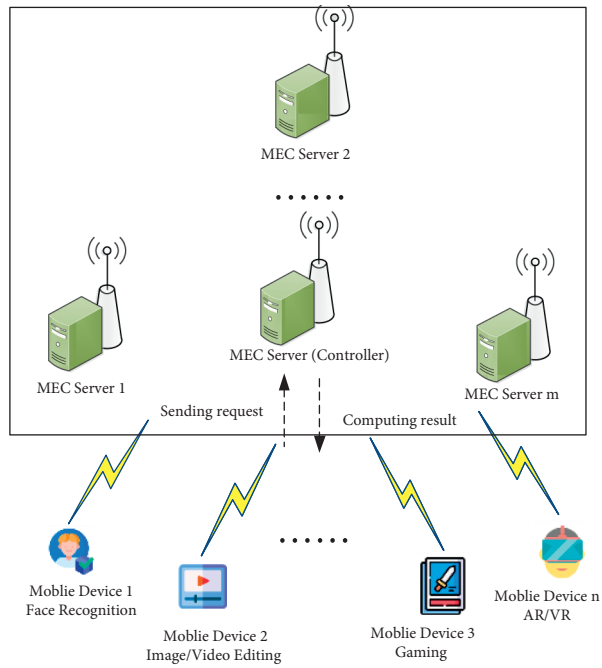

Figure 1: The architecture of system model.

$$t_{ik} \leq \mathrm{TC}_{ik}, \tag{7}$$

where TC_{ik} is the execution deadline of this task. Successfully completed tasks are put into the successful set G_{suc}. Accordingly, the whole time consumption for successfully completed tasks is $\sum_{i \in G_{suc}} t_{ik}$.

There are two possibilities for the failed task. One is the task is overtime, namely, $t_{ik} > \mathrm{TC}_{ik}$. These tasks must be omitted or wait for the next time to be scheduled again, which depends on the type of the task. The other is that the task is not assigned to any server in this schedule at all, namely, $\sum_{j \in CS} s_{ij} = 0$. These tasks also need to wait for the next schedule or to be omitted. In these conditions, a time punishment F was given ($\mathrm{F} > \mathrm{TC}_{ik}$), and the failed tasks are put into unsuccessful set G_{fail}. Thus, at a certain time slot σ, the total time cost of all the scheduling is defined as follows:

$$t_{all} = \Sigma_{i \in G_{suc}} t_{ik} + \left| G_{fail} \right| \cdot F. \tag{8}$$

Another evaluation metric is weight. The sum of the weights of all successfully offloaded tasks is

$$w = \sum_{i \in S_{suc}} w_i. \tag{9}$$

3.3. Problem Formulation. For the task scheduling problem involved in this paper, there are $2^{n \times m}$ possible scheduling solutions. The set of scheduling solutions is denoted as S. The

optimization goal of this model is to find a scheduling $s(s \in S)$ under the constraints of communication, computing, and energy resources, which can minimize the total execution time and maximize the total weight. Here, any scheduling solution should satisfy two constraints: one is the computing resources constraint, which is shown as

$$\forall_{j \in CS} \sum_{i \in D} s_{ij} \leq M_j, \tag{10}$$

which means that the j-th server can be assigned up to at most M_j tasks due to its limited computational capability. The other one is the energy consumption constraint, which is shown as

$$\varepsilon_l + \sum_{j \in CS} \varepsilon_{i,j}^{tx} \cdot s_{ij} \leq \varphi_{ik}, \tag{11}$$

where ε_l is the energy consumption of local execution and ε_{ij}^{tx} is the energy consumption for transmitting between the i-th IMD and the j-th edge server and is given by

$$\varepsilon_{i,j}^{tx} = p_i^{tx} \cdot t_{comm}, \tag{12}$$

where p_i^{tx} represents the fixed transmit power of the IMD$_i$. Formula (11) means the total energy consumption of the i-th IMD must be less than the current battery energy value of this device.

Therefore, the proposed task scheduling problem can be formulated as the following combinatorial optimization problem P:

$$\begin{aligned} \mathrm{P} \begin{cases} \min f_1(s) = t_{all} = \Sigma_{i \in G_{suc}} t_{ik} + \left| G_{fail} \right| \cdot F, \\ \min f_2(s) = w = -\sum_{i \in S_{suc}} w_i, \end{cases} \\ s.t. \quad C1: \ \forall_{j \in CS} \sum_{i \in D} s_{ij} \leq M_j, \\ C2: \ \varepsilon_l + \sum_{j \in CS} \varepsilon_{i,j}^{tx} \cdot s_{ij} \leq \varphi_{ik}. \end{aligned} \tag{13}$$

3.4. Demo. In this section, a simple demo was given to illustrate the task scheduling problem in multiple server environments. As shown in Figure 2, suppose there are three devices (D_1, D_2, and D_3) and two MEC servers (CS_1 and CS_2) in the system, and there are three queues T^1, T^2, T^3, respectively. At the first time slot, tasks in each queue are denoted as $T_1^1(2, 2, 1, 1), T_1^2(5, 5, 3, 2), T_1^3(3, 3, 1, 1)$, respectively.

For simplify, we make the following assumptions: (1) each MEC server can run only one task at one time; (2) the delay constraint of all these three tasks is 7, namely $Tc = 7$; (3) the data transmission rate C_{ij}, the computation rate η_j, and the unit data energy consumption of transmission between i-th IMD and j-th server ε_{ij}^{tx} are all assumed to be 1; (4) the local execution time, energy consumption, and the coordination cost are ignored, namely $\varepsilon_l = 0$ and $t_{coor} = t_{local} = 0$. As shown in Table 2, three task scheduling solutions are given.

Since $s1$ cannot meet constraints (10) and (11), it is invalid scheduling. $s2$ satisfies constraints (10) and (11).

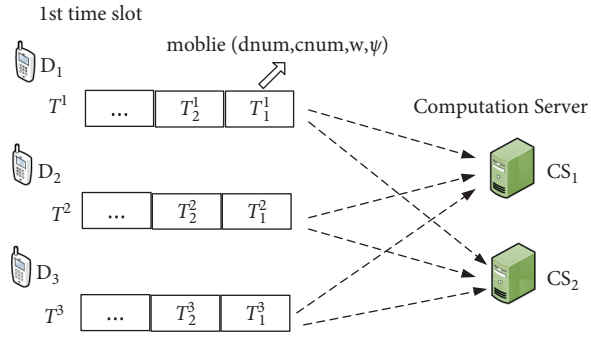

Figure 2: Initial condition of DEMO.

Table 2: Solutions.

Solution s	CS\D	D_1	D_2	D_3
$s1$	CS_1	1	1	1
	CS_2	1	1	1
$s2$	CS_1	1	0	0
	CS_2	0	0	1
$s3$	CS_1	0	1	0
	CS_2	0	1	0

According to formula (1), the task completion time of T_1^1 is 4, which is less than the delay constraint. Therefore, this is a successful offloading, and sequence number 1 is put into set G_{suc}. As a result, $G_{suc} = \{1\}$. T_1^2 is not assigned any computing server in this scheduling, so sequence number 2 is put into the failure set G_{fail}. According to formula (1), the task completion time of T_1^3 is 6, which is less than the delay constraint value. Since it is a successful offloading, sequence number 3 is put into the success set G_{suc}, and G_{suc} is updates to $\{1, 3\}$. According to formulas (8) and (9), $t_{all} = 4 + 6 + F = 10 + F$ and $w = 2$. Also, $s3$ satisfies the constraints (10) and (11). T_1^1 is not assigned any computing server in this scheduling. Thus, $s3$ is a failed offloading. According to formula (1), the completion time of T_1^2 is 5. Therefore, this is a successful offloading, and sequence number 2 is put into the success set G_{suc} and $G_{suc} = \{2\}$. T_1^3 is not assigned any computing server in this scheduling, so it is a failed offloading. Thus, we have $t_{all} = 5 + 2F$ and $w = 3$.

In all, none of these scheduling schemes has both the least time consumption time and the largest task completion weight. Generally, the best solution that meets all the objectives cannot be found. However, the noninferior solution can be found.

4. Proposed Algorithm

4.1. Multiobjective Optimization Problem

4.1.1. Problem Statement. The general description of the multiobjective optimization problem is as follows:

Given the vector $X = (x_1, x_2, \ldots, x_n) \in R^n$, and it satisfies the following constraints:

$$g_i(X) \le 0 \, (i = 1, 2, \ldots, k), \tag{14}$$

$$h_i(X) = 0 \, (i = 1, 2, \ldots, l). \tag{15}$$

Suppose that there are r optimization objectives, which are in conflict with each other. The optimization objective can be expressed as follows:

$$\min f(X) = \min\left(f_1(X), f_2(X), \ldots, f_r(X)\right). \tag{16}$$

We want to find $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ in order that $f(X^*)$ can be optimized while satisfying constraints (14) and (15). Obviously, the scheduling problem mentioned in this paper is a multiobjective optimization problem. Generally, it is necessary to consider the conflicting subobjectives comprehensively and make the trade-off among the subobjectives.

4.1.2. Pareto-Optimal Set. Multiobjective optimization is to simultaneously optimize multiple subobjectives, and these subobjectives often conflict with each other. The optimization of one objective may result in the deterioration of another objective. Normally, no single solution can optimize all the objectives simultaneously. The trade-off among multiobjectives can be properly attained by using Pareto optimality [21].

Definition 1 (Pareto dominance). A decision vector X_A is said to dominate another decision vector X_B (noted as $X_A \succ X_B$) if and only if

$$\forall_{i=1,2,\ldots,r,} \, f_i(X_A) \le f_i(X_B) \wedge \exists_{j=1,2,\ldots,r,} \, f_j(X_A) < f_j(X_B). \tag{17}$$

Definition 2 (Pareto optimal). A solution $X^* \in X_f$ is said to be Pareto optimal if and only if

$$\neg \exists X' \in X_f : \; f_j(X^*) \ge f_j(X'), \quad \forall j = 1, 2, \ldots, r, \tag{18}$$

where X_f represents the set of solutions.

Definition 3 (Pareto-optimal set). Set P^* includes all Pareto-optimal solutions, which can be defined as follows:

$$P^* = \{X^*\} = \left\{X \in X_f \mid \neg \exists X' \in X_f : \; f_j(X) \ge f_j(X'), \quad \forall j = 1, 2, \ldots, r\right\}. \tag{19}$$

4.1.3. General Framework of MOEA Based on Pareto. An evolutionary algorithm (EA) is a kind of random search algorithm that simulates the natural selection and evolution of organisms. It is widely used because it is suitable for solving highly complex nonlinear problems. At the same time, it has good versatility. The advantages of EA have been fully demonstrated in solving single objective complex system optimization problems. However, EA cannot resolve multiobjective optimization problems effectively. For the multiobjective optimization problem, it can be resolved by the multiobjective evolutionary algorithm (MOEA). Over the last decades, the design method of MOEA has attracted great interest of researchers [22–24].

Most MOEAs adopted the general process, which is shown in Figure 3. The whole process of MOEA is described as follows. Firstly, an initial population P is generated, and an algorithm is selected to operate on P to obtain a new evolutionary population R. Next, a strategy is adopted to construct the nondominated set (NDSet) of P∪R. Generally, the set size is set when designing the algorithm (such as N). If the size of the current set is greater than or less than N, the size of the NDSet needs to be adjusted according to a certain strategy. In the adjusting process, the NDSet must meet both the size requirements and the individual diversity. Then whether the termination condition is satisfied is judged. The process ends if and only if the termination condition is satisfied. Otherwise, we need to copy the individuals in NDSet to P and continue to the next round of evolution.

4.2. Individual Evaluation Method. Different from the single-objective optimization problem, the multiobjective optimization problem needs vector comparison. The multiobjective optimization strategy adopted in this paper is similar to the method of NGSA-II [25–27], but some changes have been made. All scheduling schemes are divided into three types. For any scheduling s of the first type, it can complete all tasks on the premise of satisfying constraints (10) and (11). Obviously, this type of scheduling is ideal, so it is set to the highest rank 0, namely $rank_s = 0$. The scheduling of the same level is ranked according to the task completion time, and the higher priority value has little task completion time. The second type is also to satisfy constraints (10) and (11), but it can only complete a part of offloaded tasks. We regard each scheduling s of this type as an individual in the evolutionary algorithm. According to the NSGA-II method [28], all individuals in the first nondominated front are found firstly. In order to find the individuals in the next nondominated front, the solutions of the first front are discounted temporarily, and the above procedure is repeated. The rank of the scheduling s in the first front is 1, namely $rank_s = 1$. Similarly, the rank of the scheduling in the second front is 2, so back and forth. Compared with the crowding distance of individuals in the same rank, individuals with higher aggregation density have a higher priority value. The third type is the scheduling that violates constraints (10) and (11). Assuming that the second type is divided into n ranks, for the scheduling s that belongs to the third type, its rank is $n+1$, namely $rank_s = n + 1$. The scheduling that violates constraint (10) to a lower degree has a higher priority. According to the abovementioned method, suppose there are scheduling p and scheduling q, the comparison strategy between them is described in Algorithm 1.

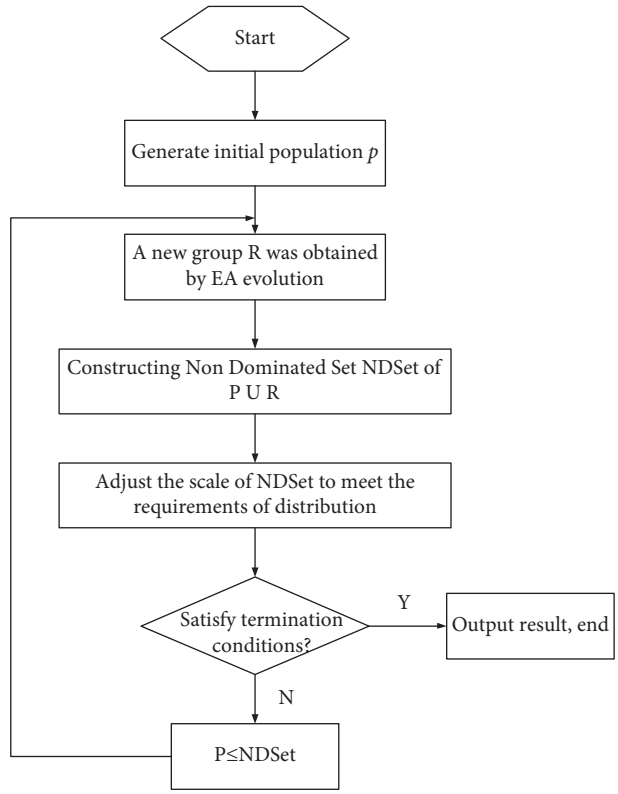

Figure 3: General flow of multiobjective evolutionary algorithm based on Pareto.

4.3. Our Algorithm

4.3.1. Bat Algorithm. As mentioned above, the basis of multiobjective evolutionary algorithm is an evolutionary algorithm. At present, the most commonly used evolutionary algorithms include simulated annealing algorithm, genetic algorithm, distribution estimation algorithm, and particle swarm optimization algorithm. However, in these algorithms, the individual's movement mode does not adopt the position update mode of the deterministic algorithm. That is, "only move to the solution that is better than the current position" method is used. Yang [29] proposed a bat algorithm (BA) that adopts the method of "only move to the solution which is better than the current position." BA simulates the behavior of microbats in nature, which uses echolocation to hunt prey and avoid obstacles. Compared with other evolutionary algorithms, BA has the characteristics of higher computational efficiency, stronger optimization ability, and robustness.

According to the bat's echolocation behavior and its correlation with objective optimization, the parameters and updating equations of n bats during flight are given below: Suppose that there are n virtual bats living in the domain. In the t-th generation, the information containing the i-th ($i = 1, 2, \ldots, n$) bat can be expressed as five tuples: $\langle \overrightarrow{x_i}(t)t, n\overrightarrow{v_i}q(t)h, {}_f xr_i 7(t)C, ; A_i(t), r_i(t)\rangle$, where $\overrightarrow{x_i}(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{ik}(t), \ldots, x_{iN}(t))$ denotes the position information of the t-th generation of the i-th bat and a solution of the search space, the speed $\overrightarrow{v_i}(t) = (v_{i1}(t), v_{i2}(t), \ldots, v_{ik}(t), \ldots, v_{iN}(t))$ represents the velocity direction of the i-th bat in the t-th generation, while frequency $fr_i(t)$, loudness $A_i(t)$, and pulse emission frequency $r_i(t)$ are three parameters needed by i-th bat in the algorithm. In the $(t+1)$-th generation, each bat firstly updates its speed according to the formula, which is described as follows:

$$v_i(t+1) = v_i(t) + \left(x_i(t) - \mathrm{p}(t) \cdot fr_i(t)m, \right. \tag{20}$$

where $\overrightarrow{p}(t) = (p_1(t), p_2(t), \ldots, p_k(t), \ldots, p_N(t))$ indicates the historical optimal position of the previous t generations, while $(x_i(t) - \mathrm{p}(t))\mathrm{fr}_i(t)$ represents the effect of the deviation between $\overrightarrow{x_i}(t)$ and $\overrightarrow{p}(t)$ on the speed of the next generation, and the frequency $fr_i(t)$ is randomly generated according to the following formula:

$$\mathrm{fr}_i(t) = fr_{\min} + \left(fr_{\max} - fr_{\min}\right) \cdot \mathrm{rand}_1, \tag{21}$$

where rand_1 is a uniformly distributed random number in (0, 1) and the two parameters fr_{max} and fr_{min} are the preset as the upper and lower frequency limits, respectively.

On this basis, when each bat performs a global or local search, the selection of its search mode is determined in a random way. It means random number $rand_2$, which is uniformly distributed between 0 and 1 needs to be determined. If $\mathrm{rank}_2 < r_1(t)$, the i-th bat will search for food in the following global search mode:

$$x_i'(t+1) = x_i(t) + v_i(t+1). \tag{22}$$

(1) If $(\text{rank}_p < \text{rank}_q)$
(2) $\quad p \succ q$
(3) If $(\text{rank}_p = \text{rank}_q = 0$ and $t_p < t_q)$
(4) $\quad p \succ q$
(5) **If** $(\mathbf{rank_p = rank_q = n+1}$ **and** $\sum_{\mathbf{j}\in\mathbf{cs}} \mathbf{max}(\sum_{\mathbf{i}\in\mathbf{D}} \mathbf{p_{ij}} - \mathbf{M_j}, \mathbf{0}) < \sum_{\mathbf{j}\in\mathbf{cs}} \mathbf{max}(\sum_{\mathbf{i}\in\mathbf{D}} \mathbf{q_{ij}} - \mathbf{M_j}, \mathbf{0}))$
(6) $\quad p \succ q$
(7) If $(1 \le \text{rank}_p \le n$ and $1 \le \text{rank}_q \le n$ and $d_p > d_q)$
(8) $\quad p \succ q$
(d_p and d_q represent the aggregation density of p and q, respectively)

Algorithm 1: Individual evaluation algorithm (IEA).

Otherwise, the i-th bat will perform a local search according to the following formula:

$$x_i'(t+1) = p(t) + \varepsilon_i \cdot \overline{A(t)}, \tag{23}$$

where ε_i is a uniformly distributed random number belongs to (−1, 1) and $\overline{A(t)} = \sum_{j=1}^{n} A_j(t)/n$ is the average loudness of the bat at the t-th time.

After the new location $\overrightarrow{x_i'}(t+1) = (x_{i1}'(t), x_{i2}'(t), \ldots, x_{ik}'(t), \ldots, x_{iN}'(t))$ is calculated, the bat will be judged whether to move instead of moving to the new location immediately according to the following update rules:

$$\overrightarrow{x_i}(t+1) = \begin{cases} \overrightarrow{x_i'}(t+1), & \text{rand}_3 < A_i(t) \text{and} f\left(\overrightarrow{x_i'}(t+1)\right) < f\left(\overrightarrow{x_i}(t)\right), \\ \overrightarrow{x_i}(t), & \text{else.} \end{cases} \tag{24}$$

When updating the position, a uniformly distributed random number $rand_3$ that belongs to (0, 1) was selected. When $\text{rand}_3 < A_i(t)$ and $f(\overrightarrow{x_i'}(t+1)) < f(\overrightarrow{x_i}(t))$ are satisfied simultaneously, the i-th bat updates the location to $\overrightarrow{x_i}(t+1)$. Otherwise, the location of i-th bat is still $\overrightarrow{x_i}(t)$ without updating the location to $\overrightarrow{x_i}(t+1)$.

The update formula of the pulse emission rate $r_i(t+1)$ is as follows:

$$r_i(t+1) = r_i(0) \cdot \left(1 - e^{-\gamma t}\right). \tag{25}$$

Loudness $A_i(t+1)$ is updated as follows:

$$A_i(t+1) = \alpha A_i(t), \tag{26}$$

where $\alpha > 0$ and $\gamma > 0$ are both preset parameters, $A_i(0)$ is the initial value of loudness, and $r_i(0)$ is the initial value of pulse emission rate. In the paper, $A_i(0)$ is random selected from [0, 1] and $r_i(0) = 0.1$.

In the basic bat algorithm, equations (20) and (22) represent the global search mechanism of the algorithm, while equation (23) represents the local search mechanism of the bat algorithm.

4.3.2. Hybrid Immune Bat Algorithm. It was shown that the optimization ability of BA mainly depends on the interaction and influence between bat individuals. Due to the lack of a mutation mechanism, it is difficult for individuals to get rid of the constraint of a local extreme value. Moreover, in the evolution process, the super bats in the population may attract other individuals to gather around them quickly, which results in a significant decline in population diversity. Meanwhile, the bat individuals are getting closer to the optimal individuals of the population in order that the population has lost the ability of further evolution [30]. In this paper, the clonal selection mechanism in the artificial immune system is introduced, which can enhance the diversity of bat population, enhance the ability of wide range variation, and increase the convergence rate.

Suppose that an individual population $B = \{b_1, b_2, \ldots, b_n\}$ is obtained through the process of the bat algorithm, which is a temporary clonal population. Each bat i ($i = 1, 2, \ldots, n$) in the temporary clonal population is regarded as an antibody. The specific methods are as follows:

Step 1: the k ($k < n$) antibody individuals selected in a random way were grouped into subpopulation $\text{Sub}_1 = \{b_1', b_2', \ldots, b_k'\}$. And clone and copy subpopulation Sub_1. The cloning operator is described as follows:

$$\text{Sub}_2 = T_c(\text{Sub}_1) = [T_C(b_1'), T_C(b_2'), \ldots, T_C(b_k')], \tag{27}$$

where $T_C(b_i') = q_i \times b_i'$, $(i = 1, 2, \ldots, k)$, where q_i is the number of clones of b_i', which is proportional to the fitness of b_i'. A new population Sub_2 was generated by cloning.

Step 2: implement high-frequency mutation for each individual in group Sub_2, and the mutation operator is adaptive, which is related to both evolution generations and individual fitness. For any $b_i' = (b_{i1}', b_{i2}', \ldots, b_{ij}', \ldots, b_{in}')$, the mutation formula is given as follows:

$$b_{ij}^{'new} = \begin{cases} \sim b_{ij}^{'old}, & \text{rand}_4 < \eta(t) \text{ and } \text{rand}_5 < \Delta, \\ b_{ij}^{'old}, & \text{else.} \end{cases} \quad (28)$$

$\eta(t)$ in formula (28) is given as follows:

$$\eta(t) = 1 - r_1^{[1-(t/T)]^b}, \quad (29)$$

where b is a positive constant, $r_1 \in (0, 1)$. In the early stages of evolution, $r_1^{[1-(t/T)]^b}$ was smaller, $\eta(t) \approx 1$, but in the later stage of evolution, when t approaches T, $\eta(t) \approx 0$, local search is carried out in a small space. Furthermore, Δ in formula (28) is formulated as follows:

$$\Delta = 1 - r_2^{R^\lambda}, \quad (30)$$

where $r_2 \in (0, 1)$, the parameter λ plays the role of adjusting the search area, and the value is generally 2–5 [31]. R in the above formula is formulated as follows:

$$R = 1 - \frac{\text{fit}(b_i')}{\text{fit}_{\max}}, \quad (31)$$

where $\text{fit}(b_i')$ represents the fitness of antibody b_i' and fit_{max} is the maximum fitness value, which is the aggregation density of the individual. Obviously, for those with higher fitness, the Δ value is smaller, and the mutation possibility is small, while for those with lower fitness, the mutation possibility is relatively large.

Step 3: immune clonal selection: the best individual from the clonal mutation individuals is selected for the next generation. The full algorithm of HIBSA is given in Algorithm 2.

4.3.3. The Full Algorithm of HIBSA. 4.4. Time Complexity Analysis. In this section, the time complexity of HIBSA is analyzed. Based on the flowchart in Figure 3, assuming that the population P size, the new group R size are both N, the number of objectives is M (in fact, M is 2 in this paper). The number of decision valuables is $m \times n$ (n is the number of IMDs and m is the number of MEC servers). The basic operators and their time complexity analysis are given as follows:

(1) Generate initial population P. Population initialization is to generate individuals randomly and calculate the values of objectives, so the time complexity is $O(m \times n \times N) + O(M \times N)$.

(2) A new group R was obtained by EA evolution. In this paper, the evolutionary algorithm combines the bat algorithm and the immune algorithm together. In both these two algorithms, identifying nondominated individuals was needed. When identifying nondominated individuals, individuals are compared with each other based on the objectives. Hence, the time complexity is $O(M \times N^2)$.

(3) Construct nondominated set (NDSet) of P∪R. As these operators are performed in the P∪R, so the time complexity of nondominated individuals identification is $O(M \times (2N)^2)$

(4) Adjust the scale of NDSet to meet the requirements of distribution. In this stage, adjust operator selects nondominated individuals with greater fitness values to preserve. Therefore, in the assignment of fitness values, the time complexity of the crowding-distance assignment is $O(M \times N \times \log(N))$.

Based on the above analysis, in a single generation, the worst time complexity can be written as follows:

$$O\left(M \times N^2\right). \quad (32)$$

5. Experimental Evaluation

In this section, we evaluate the performance of HIBSA through simulations and compare its performance against several algorithms.

5.1. Verification Policies. Suppose that at one time, there are n mobile devices and m computing servers, and each device currently has a task queue to be offloaded. Assuming that on the t-th slot, the k-th task T_i^k in the current queue on the i-th device is ready to be offloaded. If the offloading is successful, the device intends to offload the next scheduling task in the queue at the $(t+1)$-th slot, which means the $(k+1)$-th task T_i^{k+1} prepares to be offloaded. However, if task T_i^k fails to be offloaded at the t-th slot, the current task T_i^k can be omitted or wait to be scheduled at the $(t+1)$-th slot, which is decided by whether the task is a real-time task or not. In order to verify the performance of the proposed scheduling algorithm, it is compared with sequential scheduling algorithm (SSA), random scheduling algorithm (RSA), time priority greedy scheduling algorithm (TPGSA), and weight priority greedy scheduling algorithm (WPGSA). The comparison algorithms are described as follows:

(1) SSA: the scheduling is carried out according to the equipment number, and the one with a small equipment number is scheduled first.

(2) RSA: in this method, each task is scheduled randomly.

(3) TPGSA: at one time, all tasks to be scheduled are sorted by the expected completion time. The shorter the task completion time, the earlier the task is scheduled.

(4) WPGSA: at one time, all the tasks to be scheduled are sorted by the task weight.

The experimental platform used in this paper is MATLAB 2016a [32], and the main simulation parameters are presented in Table 3.

(1) The scheduling population P^0 is randomly generated, the population size is n, and the initialization generation $t = 0$
(2) According to the individual evaluation algorithm, all the individuals of rank 0 and rank 1 in P are put into the set P_0^t and P_1^t, respectively
(3) If $(P_0^t \neq \Phi)$
select the optimal solution in P_0^t as the result and the algorithm ends
(4) else
(5) while $(t < T)$
(6) A solution is randomly selected from P_1^t as the historical optimal position of the current population, and BA is implemented for P^t to get P_B^t
(7) Clear set P_0^t, P_1^t. According to the individual evaluation algorithm, all the individuals of rank 0 and rank 1 in the set P_B^t are put into the set P_0^t and P_1^t, respectively
(8) If $(P_0^t \neq \Phi)$
(9) select the optimal solution in P_0^t as the result and the algorithm ends
(10) else
(11) For P_B^t, the immune clonal selection algorithm is implemented to get P_I^t
(12) Clear set P_0^t and P_1^t. According to the individual evaluation algorithm, all the individuals of rank 0 and rank 1 in the set P_B^t are put into the set P_0^t and P_1^t, respectively
(13) If $(P_0^t \neq \Phi)$
(14) select the optimal solution in P_0^t as the result and the algorithm ends
(15) else
(16) According to the individual evaluation algorithm, put the first n individuals in the set $P_B^t \cup P_I^t$ into P^{t+1}
(17) $t = t + 1$
(18) end if
(19) end if
(20) end while
(21) end if
(22) select a solution randomly from P_1^t as the result and the algorithm ends

Algorithm 2: Hybrid immune and bat scheduling algorithm (HIBSA).

Table 3: Simulation parameters for task offloading.

Parameter	Value	Note
m	3–5	Number of MEC servers
dnum	[1–7] Mb	The size of the input data
u_{cc} [20]	1,000 cycles/bit	Unit CPU cycles (per bit)
F	8 GHz	The CPU cycle frequency of CS
u_{lc}	0.05 ms	Unit latency cost
TC	5 ms	Maximum latency
M_j	5 tasks	MEC server computing capacity
p_i^{tx}	0.1 W	Fixed transmit power of IMD
φ	40 mJ	Safe discharge threshold
N	10	Number of IMDs
w_i	[1–100]	Task weight
fr_{max} [33–36]	2	The upper frequency of bat
fr_{min} [33–36]	0	The lower frequency of bat
α [37–39]	0.9	The update parameter of the loudness of bat
γ [37–39]	0.1	The update parameter of the pulse emission rate of bat
N	30	Number of initialization populations
Gen	30	Number of generations

5.2. Experimental Result. In this section, experimental results are given. Due to the limitation of space, we only show the experimental data of the first 20 time slots.

5.2.1. Execution Time Analysis. The results about total task execution time in the first 20 time slots obtained by the five algorithms are shown in Table 4.

We take the task completion time of TPGSA as the benchmark and normalize its value to 1. Thus, the comparison of the results about the task completion time of these five algorithms at 20 time slots is shown in Figure 4:

We can see from Figure 4 that SSA, RSA, and WPGSA are all not as good as TPGSA in task execution time. However, the performance of HIBSA is better than TPGSA. As shown in Figure 5, if the average execution time of the TPGSA algorithm is 1, the average task execution time of the SSA algorithm is 1.04; the average scheduling time of the RSA algorithm is 1.13; and the average scheduling time of the WPGSA algorithm is 1.05. However, the task execution

Table 4: The total task execution time in the first 20 time slots.

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SSA	66	120	167.33	228	293.33	346.67	419.33	485.33	558	618	690	749.33	815.33	880.67	940	994	1,059.33	1,112.67	1,178.67	1,244
RSA	66.67	130	196.67	262	324.67	380	454.67	535.33	604	668.67	738	810.67	878.67	949.33	1,015.33	1,088.67	1,167.33	1,234	1,316	1,396
TPGSA	54.00	107.33	160.67	220.67	280.67	340.00	406.00	466.00	539.33	605.33	670.67	736.00	796.00	862.00	926.67	992.67	1,058.67	1,118.00	1,171.33	1,236.67
WPGSA	73.33	120.67	167.33	240.67	313.33	360.00	407.33	480.00	534.00	607.33	680.00	732.67	804.00	876.00	949.33	1,020.67	1,073.33	1,146.00	1,186.67	1,258.67
HIBSA	54	95.33	134.67	188	261.33	334	370.67	428	501.33	552.67	622.67	689.33	762.67	835.33	887.33	940	999.33	1,072.67	1,108.67	1,174

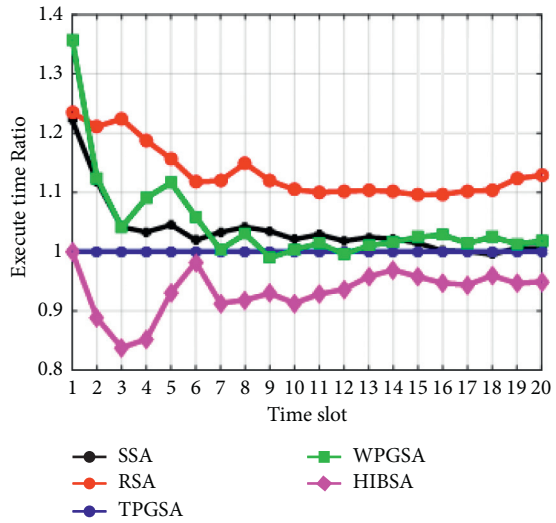

Figure 4: Comparison of task execution time ratio of various algorithms of 20 time slots.

time of HIBSA is only 0.93. That is, the scheduling time of the algorithm proposed in this paper is only 93% of the TPGSA algorithm.

5.2.2. Weight Analysis. The results about the sum of task weight in the first 20 time slots of the five algorithms are shown in Table 5.

We take the weight of WPGSA as the benchmark and normalize its value to 1. Thus, the comparison of the results about the weight of each algorithm at 20 time slots is shown in Figure 6:

Taking WPGSA as the benchmark, SSA, RSA, and TPGSA are not as good as WPGSA in weight, while HIBSA proposed in this paper is better than WPGSA sometimes and slightly worse than WPGSA. As shown in Figure 7, if the average scheduling weight of WPGSA is 1, the average scheduling weight of SSA is 0.92; the average scheduling weight of RSA is only 0.59; and the average scheduling weight of TPGSA is 0.87, while the scheduling weight of HIBSA is slightly worse than WPGSA. According to the scheduling strategy proposed in the paper, the scheduling weight of the proposed algorithm is 98% of WPGSA.

5.2.3. Analysis of the Total Number of Offloaded Tasks. The total number of tasks offloaded by each device in the first 20 time slots by using different algorithms is shown in Table 6

As shown in Table 6, 10 devices offload 132 tasks in 20 time slots by using SSA. RSA algorithm offloads 93 tasks only. TPGSA offloads 133 tasks. WPGSA offloads 130 tasks, while HIBSA has the largest number of offloaded tasks. This is because the algorithm proposed in this paper is a multiobjective optimization algorithm, and it also takes fairness into consideration in task scheduling. Figure 8 is a comparison box plot of the number of scheduling devices using these five algorithms. As shown in Figure 8, the quartile deviation between the SSA algorithm and the TPGSA algorithm is large, which results in the unfairness of the

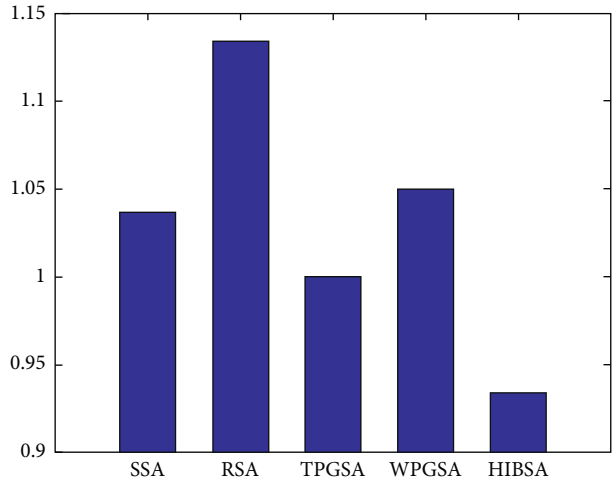

Figure 5: Comparison of average task execution time ratio of various algorithms.

algorithm for the task scheduling of each device. Some devices offload all generated tasks, while some devices have no tasks to offload. Obviously, the quartile deviation of HIBSA proposed in the paper is only 1.75, which is better than the other four algorithms.

5.2.4. Scalability Analysis. As analyzed in Section 4.4, the performance of the algorithm proposed in this paper is only related to the number N of the initialization population. However, some studies have shown that the setting of the number of initialization population should be related to the length of the problem. In this paper, the length of the problem is the product $(m \times n)$ of the number of available MEC servers and the number of IMDs. That is, with the increase of the number of servers and the number of IMDs, the number of initialization population should be increased to meet the diversity of the population.

Table 5: The sum of task weight in the first 20 time slots.

T	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SSA	312	626	851	1,254	1,485	1,718	2,128	2,440	2,850	3,172	3,492	3,733	4,063	4,303	4,544	4,849	5,089	5,322	5,634	5,874
RSA	203	316	537	650	962	1,176	1,396	1,597	1,809	1,931	2,242	2,444	2,665	2,696	2,737	3,237	3,637	3,750	3,960	4,071
TPGSA	314	547	771	1,084	1,406	1,638	1,950	2,263	2,763	3,084	3,315	3,555	3,877	4,198	4,348	4,660	4,981	5,213	5,446	5,686
WPGSA	500	725	869	1,369	1,779	1,923	2,148	2,558	2,872	3,372	3,782	3,925	4,155	4,475	4,975	5,205	5,348	5,758	5,894	6,214
HIBSA	314	747	982	1,206	1,706	2,116	2,261	2,484	2,984	3,018	3,419	3,830	4,330	4,740	4,811	4,954	5,177	5,677	5,903	6,134

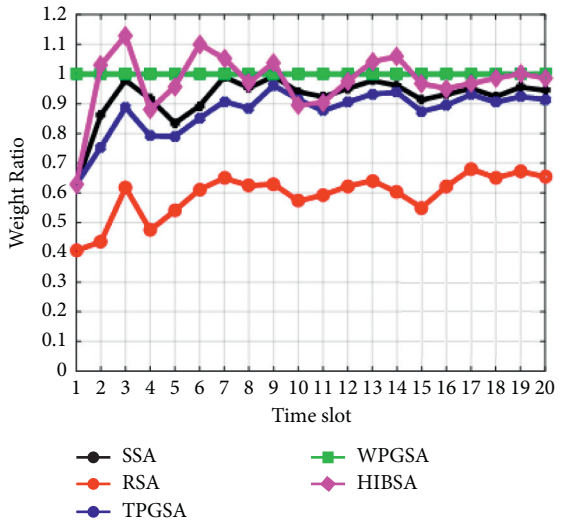

Figure 6: Comparison of task weight ratio of various algorithms of 20 time slots.

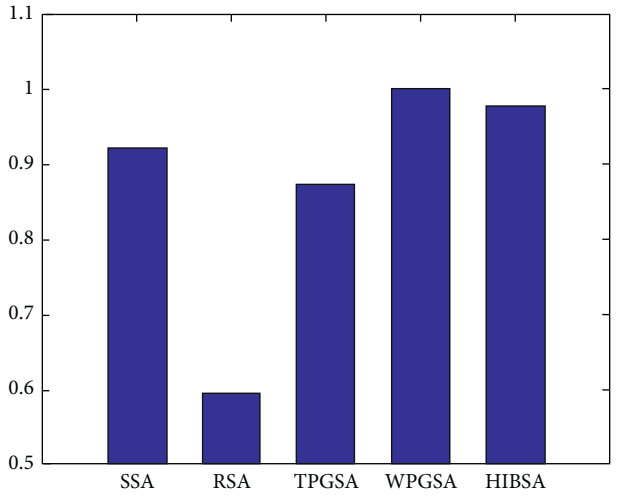

Figure 7: Comparison of average task weight ratio of various algorithms.

Table 6: The total number of scheduled tasks for 10 mobile devices in the first 20 time slots.

DEV ID	1	2	3	4	5	6	7	8	9	10	Sum
SSA	20	20	20	20	20	17	9	5	1	0	132
RSA	12	11	8	8	12	10	8	9	6	9	93
TPGSA	20	20	20	19	18	18	7	7	4	0	133
WPGSA	18	17	19	17	13	10	18	8	5	5	130
HIBSA	16	18	14	14	13	13	13	15	13	11	140

In the algorithm proposed in this paper, we adopt the method of adding diversity judgment in the iterative execution of the algorithm. If the population diversity is lower than the threshold preset, we can improve the population diversity through population diversity regulation. As shown in Figure 9, when the number of available servers is increased from 3 to 5, the number of solution space will grow from 2^{30} to 2^{50} rapidly. However, the number of initialization populations we set is all 30. The experimental results indicate that the solutions obtained by HIBSA are not

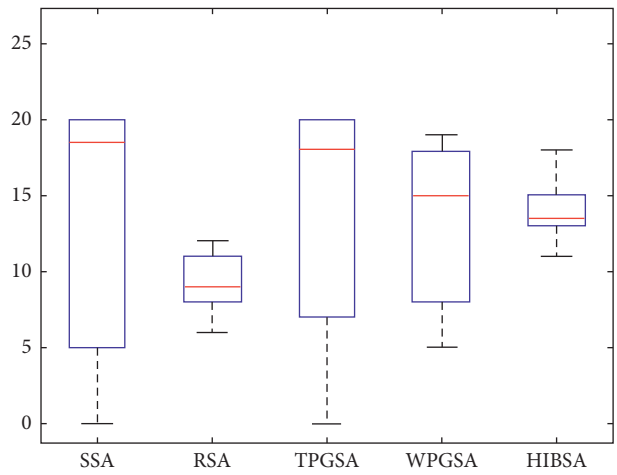

Figure 8: Comparison of the number of tasks offloaded by different algorithms.

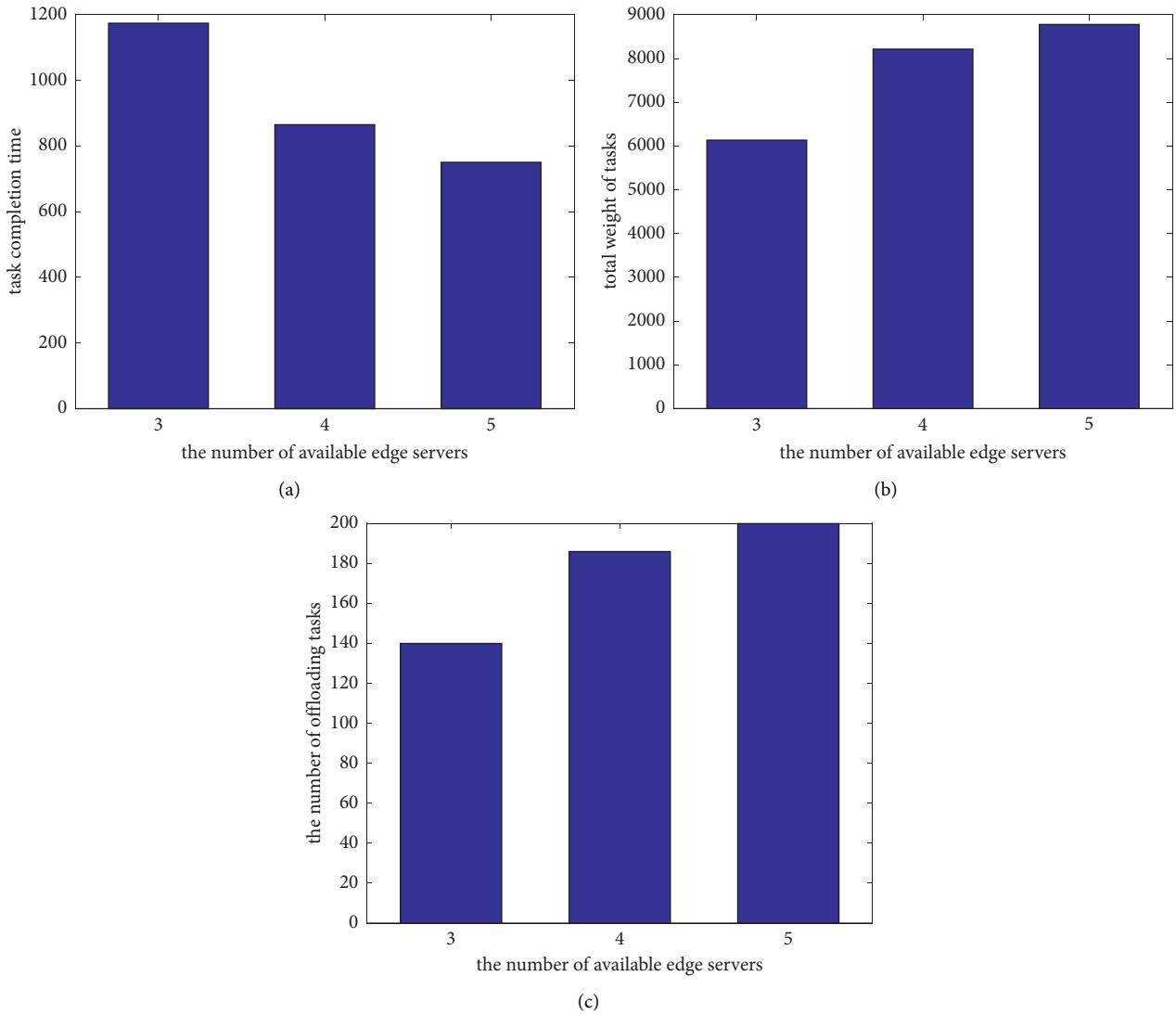

Figure 9: the scalability of the algorithm: (a) tasks execution time, (b) the total weight of tasks, and (c) the number of tasks offloading.

affected. They are significantly improved in terms of tasks execution time, the total weight of tasks, and the number of offloaded tasks, which shows the scalability of the algorithm.

6. Conclusions

Task offloading in mobile edge computing relieves the data computing pressure of local devices and central cloud by offloading data to the edge cloud, which also reduces the task execution delay caused by the lack of computing resources. In this paper, an algorithm based on hybrid immune and bat scheduling algorithm (HIBSA) is proposed to tackle the multiobjective optimization problem. Three main contributions are presented in this paper. Firstly, the proposed system model considers communication and computing resources, energy consumption of intelligent mobile devices, and weight of tasks. Secondly, the scenario that the mobile device can generate multiple tasks at the same time is considered, which is more realistic compared with most of the related works. Thirdly, the evolutionary algorithm presented combines the advantages of the bat algorithm and the immune algorithm that ensures the convergence and diversity of solutions. Finally, the practicability of the proposed algorithm is well verified by simulation. We can see from experimental results that the algorithm can meet the requirements of both the task execution time of the offloaded tasks and the weight of the completed tasks. Moreover, the algorithm has good scalability. However, the performance of the proposed algorithm can further be improved in the future. It needs to be further verified by using real scene data also.

Data Availability

The main purpose of this paper is to study the scheduling algorithm, so all the data are obtained through MATLAB simulation, not from the real scene. All data included in this study are available upon request from the corresponding author.

Conflicts of Interest

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

Acknowledgments

This work was supported in part by the Natural Science Foundation of China under Grant nos. 61902282 and 62002263, the Doctoral Fund of Tianjin Normal University (No. 52XB1909), and the Science and Technology Development Fund of Tianjin Education Commission for Higher Education (no. JW1702).

References

[1] T. Liu, Y. Zhang, Y. Zhu, W. Tong, and Y. Yang, "Online computation offloading and resource scheduling in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6649–6664, 2021.

[2] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 3, no. 19, pp. 1628–1656, 2017.

[3] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in Mobile Edge Computing: a deep learning-based hybrid approach," *Journal of Network and Computer Applications*, vol. 178, 2021.

[4] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2015.

[5] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.

[6] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-Enabled mobile-edge computation offloading for multiuser IoT network," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12490–12504, 2021.

[7] H. Tout, A. Mourad, N. Kara, and C. Talhi, "Multi-persona mobility: joint cost-effective and resource-aware mobile-edge computation offloading," *IEEE/ACM Transactions on Networking*, vol. 29, no. 3, pp. 1408–1421, 2021.

[8] Z. Kuang, Z. Ma, and Z. Li, "Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing," *Journal of Systems Architecture*, vol. 118, no. 99, pp. 102–167, 2021.

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

[10] S. Wu, W. Xia, W. Cui et al., "An efficient offloading algorithm based on Support vector machine for mobile edge computing in vehicular networks," in *Proceedings of the 2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pp. 1–6, Hangzhou, China, October 2018.

[11] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[12] 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.

[13] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[14] J. M. Zhang, F. Y. Yang, and Z. Y. Wu, *Multi-access Edge Computing (MEC) and Key Technologies*, pp. 264-265, Post & Telecom Press, Beijing, China, 2019.

[15] H. Zhao, S. Deng, C. Zhang, W. Du, Q. He, and J. Yin, "A mobility-aware cross-edge computation offloading framework for partitionable applications," in *Proceedings of the 2019 IEEE International Conference on Web Services (ICWS)*, pp. 193–200, Milan, Italy, July 2019.

[16] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, 2021.

[17] K. Fu and J. Ye, "Computation offloading based on improved glowworm swarm optimization algorithm in mobile edge computing," *Journal of Physics: Conference Series*, vol. 1757, no. 1, pp. 012–195, 2021.

[18] G. Mitsis, P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Intelligent dynamic data offloading in a competitive mobile edge computing market," *Future Internet*, vol. 11, no. 5, p. 118, 2019.

[19] J. H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, and M. Mittal, "Optimal cooperative offloading scheme for energy efficient multi-access edge computation," *IEEE Access*, vol. 8, pp. 53931–53941, 2020.

[20] J. Sun, Q. Gu, and T. Zheng, "Joint communication and computing resource allocation in vehicular edge computing," *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, pp. 1–13, 2019.

[21] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003.

[22] Z. Liang, R. Song, Q. Lin et al., "A double-module immune algorithm for multi-objective optimization problems," *Applied Soft Computing*, vol. 35, pp. 161–174, 2015.

[23] Q. Lin and J. Chen, "A novel micro-population immune multiobjective optimization algorithm," *Computers & Operations Research*, vol. 40, no. 6, pp. 1590–1601, 2013.

[24] J. Gao and J. Wang, "A hybrid quantum-inspired immune algorithm for multiobjective optimization," *Applied Mathematics and Computation*, vol. 217, no. 9, pp. 4754–4770, 2011.

[25] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[26] G. Rudolph, "Some theoretical properties of evolutionary algorithms under partially ordered fitness values," in *Proceedings of the Evolutionary Algorithms Workshop*, pp. 9–22, San Francisco, CA, USA, July 2001.

[27] G. Rudolph, "Evolutionary search under partially ordered fitness sets," in *Proceedings of the International Symposium on Informationence Innovations in Engineering of Natural and Artificial Intelligent Systems*, pp. 818–822, Hersonissos, Greece, June 2001.

[28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[29] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, vol. 284, pp. 65–74, 2010.

[30] I. Fister, I. Fister, X.-S. Yang, S. Fong, and Y. Zhuang, "Bat algorithm: recent advances," in *Proceedings of the 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 163–167, Timisoara, Romania, May 2014.

[31] R. H. Shang, L. C. Jiao, and W. P. Ma, "Immune clonal multiobjective optimization algorithm for constrained optimization," *Journal of Software*, vol. 19, no. 11, pp. 2943–2956, 2008.

[32] Research with MATLAB and Simulink. https://ww2.mathworks.cn/academia/research.html.

[33] O. N. Roeva and S. S. Fidanova, "Hybrid bat algorithm for parameter identification of an E. Coli Cultivation process model," *Biotechnology & Biotechnological Equipment*, vol. 27, no. 6, pp. 4323–4326, 2013.

[34] A. Alihodzic and M. Tuba, "Improved bat algorithm applied to multilevel image thresholding," *Science World Journal*, vol. 2014, Article ID 176718, 16 pages, 2014.

[35] L. Li and Y. Zhou, "A novel complex-valued bat algorithm," *Neural Computing & Applications*, vol. 25, no. 6, pp. 1369–1381, 2014.

[36] A. H. Gandomi and X.-S. Yang, "Chaotic bat algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp. 224–232, 2014.

[37] E. S. Ali, "Optimization of power system stabilizers using BAT search algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 61, pp. 683–690, 2014.

[38] S. Kashi, A. Minuchehr, N. Poursalehi, and A. Zolfaghari, "Bat algorithm for the fuel arrangement optimization of reactor core," *Annals of Nuclear Energy*, vol. 64, pp. 144–151, 2014.

[39] X. S. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, no. 5, pp. 464–483, 2012.