Hindawi
Journal of Robotics
Volume 2021, Article ID 3965689, 9 pages
https://doi.org/10.1155/2021/3965689

Research Article

Collaborative Task Offloading Strategy of UAV Cluster Using Improved Genetic Algorithm in Mobile Edge Computing

Hong Wang

Department of Electronic Information Technology, Sichuan Modern Vocational College, Chengdu, Sichuan 610207, China

Correspondence should be addressed to Hong Wang; whiam@163.com

Received 16 November 2021; Revised 7 December 2021; Accepted 13 December 2021; Published 29 December 2021

Academic Editor: Kaijian Xia

Aiming at the problem that traditional fixed base stations cannot provide good signal coverage due to geographical factors, which may reduce the efficiency of task offloading, a collaborate task offloading strategy using improved genetic algorithm in mobile edge computing (MEC) is proposed by introducing the unmanned aerial vehicle (UAV) cluster. First, for the scenario of the UAV cluster serving multiple ground terminals, a collaborative task offloading model is formulated to offload the tasks to UAVs or the base station selectively. Then, an objective function and related constraints are put forward to minimize the time delay and energy consumption by analysis of those in the communication and computing process in the system while considering many factors. Then, the improved genetic algorithm is introduced to solve the optimization problem, obtaining the optimal collaborative task offloading strategy. To verify the performance of the proposed method, simulations are conducted on MATLAB. Simulation results showed that the joint utilization of UAV and MEC improves the offloading efficiency of the proposed strategy. When the number of UAVs is 12, the total utility is up to 1.83 and the task completion time does not exceed 110 ms. In this case, the task can be reasonably offloaded to UAVs or accomplished locally.

1. Introduction

With the rapid development of big data, cloud computing, artificial intelligence, and Internet of things (IoT) technology and the maturity of 5G communication technology, the inevitable trend of interconnection of all things has greatly accelerated the evolution of IoT systems toward multiple source isomerism and intelligent online processing, such as smart city, smart home, and telemedicine [1, 2]. However, it is bound to bring great pressure to the network link and data center when transmitting all the data of massive IoT terminals to the cloud computing center for centralized processing and then returning the results. In addition, the cloud computing center may probably refuse to provide services due to the overload, which will greatly affect the user experience. In this context, mobile edge computing (MEC) is introduced [3].

The allocation of network resources in edge computing is similar to that in traditional cloud computing, as it mainly involves the reasonable allocation of computing resources and energy for user terminals and services. Additionally, in order to meet the requirements of energy consumption and packet loss rate, it needs to consider the offloading of users' tasks [4, 5]. Mobility management and computation offloading are two main technologies in mobile edge computing in recent research. Mobility management mainly concerns the resource discovery and switching, while computation offloading solves the problems of when to offload computation tasks, what to offload, and how much to offload [6]. Due to the flexibility of edge computing equipment, the service platform is no longer limited to fixed communication base stations or roadside units. Unmanned aerial vehicles (UAVs) with computing capability can also provide computing and relay services for mobile users [7]. Based on the strong mobility of UAVs, it can avoid the problems caused by many geographical factors and fill the blind area in the signal covering of communication base stations [8].

There has been some research investigating on task offloading and resource optimization allocation in MEC.

Saleem et al. proposed an online algorithm with low complexity, which realizes dynamic computation offloading based on Lyapunov optimization and minimizes the task execution time [9]. The performance needs to be further improved in the complex environment where multiple nodes and tasks processing are involved. A novel privacy-preserving and cost-efficient task offloading scheme based on the general Lyapunov optimization was given by He et al. [10]. This scheme can protect users' privacy while ensuring the best user experience. However, the processing efficiency of computing tasks is low, which is not suitable for large-scale multitask collaborative allocation. By applying the Lagrange multiplier method, Yang et al. formulated the offloading strategy and resource allocation optimization into a nonlinear equation with linear inequality constraints and then proposed the bisection search algorithm to solve them effectively, which improves the processing efficiency. Nevertheless, the communication delay has a great impact on the quality of service [11]. In order to reduce the influence of long distance between users and cloud servers on the quality of service, Ito and Koga proposed a flow splitting and aggregation scheme to lessen the offloading delay of cloud servers in edge computing, while the energy consumption still needs to be optimized [12]. Most of the aforementioned methods focus on the single-objective optimization, such as delay in computation task processing, energy consumption, and resource allocation. The overall multiobjective optimization still needs to be deeply studied.

With the continuous development of computer technology and advancement of communication technology, the advantages of heuristic algorithms in solving optimization problems have been gradually highlighted. In Reference [13], the vehicular mobile edge computing system was studied. Two independent heuristic algorithms were adopted to minimize the average delay of computation task processing while considering the limitations of vehicle mobility and energy consumption, but the energy consumption optimization in the computation process is not considered. Hossain et al. proposed a novel fuzzy-based collaborative task offloading scheme. By introducing the fuzzy logic method, the target MEC server is selected for task offloading, so as to accommodate more computation workload in the MEC system and reduce the dependence on remote cloud [14]. However, it only offloads the tasks to fixed edge servers, lacking the consideration of mobile computing servers, such as UAVs. Chen et al. formulated the task offloading as an integer nonlinear programming problem and proposed an efficient task offloading and channel resource allocation scheme based on differential evolution algorithm, which can significantly reduce the energy consumption and has a good convergence performance [15].

The computation tasks mentioned above are offloaded to intelligent terminals such as base stations, cars, and mobile phones, thus in some blind areas that are not covered by communication signals, those tasks cannot be processed. Based on the above analysis, aiming at the problem that most existing offloading strategies are not suitable for UAV clusters and taking the coverage of communication signals into consideration, a collaborate task offloading strategy of UAV cluster using improved genetic algorithm in MEC is proposed. In order to solve the signal coverage problem in the traditional cloud computing, the proposed strategy uses the high mobility of UAVs and designs a collaborative task offloading model, where tasks can be partially offloaded to the UAV cluster with computing and communication capabilities or transmitted to the remote base station, which improves the task processing efficiency.

2. System Model and Optimization Object

2.1. System Model. The system model of collaborate task offloading of UAV cluster is shown in Figure 1, which is composed of N number of mobile terminals (MTs), M number of UAVs, and one base station (BS). In order to decrease the computation delay and energy consumption of MTs, the tasks can be partially offloaded to the UAV cluster with computing and communication capabilities or transmitted to the remote base station indirectly. Assuming that mobile terminals are randomly distributed in the test area, UAVs can monitor tasks and obtain data by taking photos or videos to meet the needs of real application scenarios.

The goal of the system is to minimize the weighted delay and energy consumption of the whole system. In order to achieve this, the first thing is to obtain the delay and energy consumption of the whole system. Specifically, the whole process includes task offloading and the return of computation results, as shown by the arrows in Figure 1. Moreover, task offloading consists of two parts in the link, which are the communication and computation process, respectively. Generally, the wireless channel between UAV and MT or BS is mainly the line of sight (LoS) link, so the Doppler frequency shift can be well compensated when the UAV moves.

2.2. Communication Model. The UAV-assisted MEC system is mapped into a three-dimensional Euclidean coordinate system, where the coordinates of MT and BS are located at (0, 0, 0) and (L, 0, 0), respectively. In addition, the rotor UAV can fly or hover at a fixed height H (which can be the minimum height required to avoid ground buildings, etc.), and the UAV flies at a constant speed $v > 0$ (which satisfies the constraints of maximum speed of the UAV). In order to facilitate the analysis, it is assumed that UAVs can also execute computation tasks during the flight, but UAVs must hover at a fixed position to receive input data of tasks from MTs or transmit data to the BS, as shown in Figure 2.

In this paper, the initial hover position of the UAV when executing the first task group is expressed as $u_{k,i}^0 = (x_{k,i}^0, y_{k,i}^0, H)$; the hover position of the UAV when collecting the input data of k-th task group of MT is expressed as $u_{k,i}^C = (x_{k,i}^C, y_{k,i}^C, H)$; the hover position of the UAV when transmitting the data of k-th task group to BS is expressed as $u_{k,i}^B = (x_{k,i}^B, y_{k,i}^B, H)$. Note that the hover position of the UAV after the execution of k-th group is the initial hover position when the UAV executes the $(k+1)$-th task group. Thus, the channel gains from UAV to MT and BS can be formulated respectively:

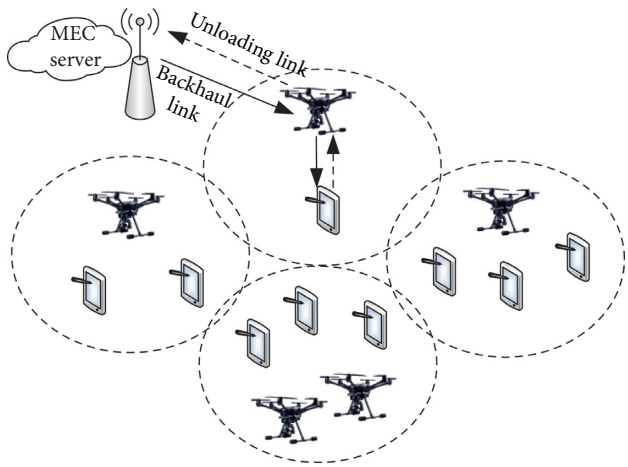

Figure 1: System model of collaborate task offloading of the UAV cluster.

$$g_k^C = \frac{p_0}{\left(x_k^C\right)^2 + \left(y_k^C\right)^2 + H^2},$$
$$g_k^B = \frac{p_0}{\left(x_k^B\right)^2 + \left(y_k^B\right)^2 + H^2}, \tag{1}$$

where p_0 is the received power when the reference distance is 1 m and the transmission power is 1 W.

p^C and p^B stand for the transmission power of the MT and the UAV, respectively, and it is assumed that each data link used for communication in the system is allocated with equal bandwidth W. Therefore, the maximum data transmission rates of the data upload link from MT to UAV and the relay link from UAV to BS are, respectively, as follows:

$$v_k^C = W\log_2\left(1 + \frac{p^C g_k^C}{\sigma^2}\right),$$
$$v_k^B = W\log_2\left(1 + \frac{p^B g_k^B}{\sigma^2}\right), \tag{2}$$

where σ^2 stands for the noise power from the UAV and the BS receiver.

2.3. Computation Model. From the perspective of a system, the task computation process mainly occurs on the devices like MT, UAV cluster, and BS. Due to the high-performance processing server and sufficient storage resources on the BS, the system does not take the delay and energy consumption into account when tasks are executed on the BS. Therefore, compared with tasks locally executed on MTs or offloaded to UAVs, the delay and energy consumption are much smaller when tasks are executed on the BS for the whole system [16]. The specific calculation process is as follows.

2.3.1. Local Execution Model. As the name implies, local execution means that the task is executed on MTs. In the process of local execution, the delay of the task can be represented as

$$T_n^{\text{local}} = \frac{G_n}{f_n}, \tag{3}$$

where G_n is the CPU cycle of the task on the mobile terminal n. f_n is the CPU frequency of the node n. By applying dynamic voltage and frequency scaling (DVFS) technology, the MT can control the energy consumption by reducing the frequency. Hence, the energy consumption of mobile terminal n can be written as

$$E_n^{\text{local}} = T_n^{\text{local}} \eta_n f_n^3, \tag{4}$$

where η_n is the effective capacity coefficient of the mobile terminal n. The volume of data executed locally, which is represented as D_n^{local}, can be calculated as

$$D_n^{\text{local}} = \frac{T_n^{\text{local}} f_n}{G_n}. \tag{5}$$

2.3.2. Execution on the UAV Model. Assuming that UAVs also apply DVFS technology to improve the energy efficiency, the delay and energy consumption when task is executed on UAV m can be calculated, respectively, as

$$T_m = \frac{G_m}{f_m},$$
$$E_m = T_m \eta_m f_m^3, \tag{6}$$

where G_m is the CPU cycle of the task on the UAV m. f_m is the CPU frequency of UAV m. η_n is the effective capacity coefficient of the UAV m. The bits of data executed on UAV m can be calculated as follows:

$$D_m = \frac{T_m f_m}{G_n}. \tag{7}$$

2.3.3. Energy Consumption Model when UAV Flies. E^{fly} represents the energy consumption when the UAV is in flight, which can be formulated as follows:

$$E^{\text{fly}} = \lambda \left\| v_u \right\|^2, \tag{8}$$

where $\lambda = 0.5$ and v_u is the flight speed of the UAV.

2.4. Model Constraints Description. In the UAV cluster collaborate task offloading system, considering the generally limited energy of UAVs, it is necessary to clarify the constraints of the energy consumption per unit time [17]. In this scenario, the UAV no longer needs to move in space, thus it only requires to calculate the energy consumed by the UAV when providing computing services for the ground terminal and hovering in the air [18].

$$T_m \eta_m f_m^3 + E_m^{\text{fly}} \le E_m^u, \quad \forall m \in M. \tag{9}$$

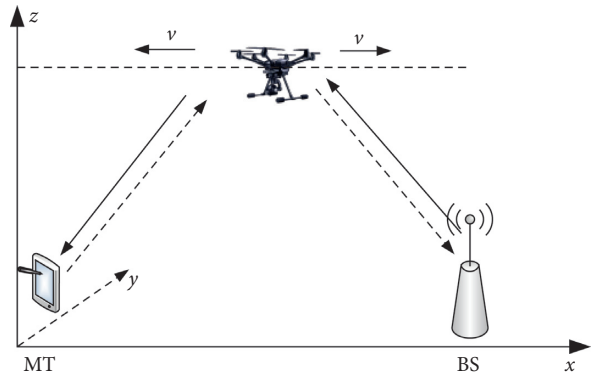

Figure 2: UAV-assisted MEC system model.

The inequality constraint represented in (9) shows the energy limitation of UAVs. The first term of the summation formula on the left of the inequality sign represents the total energy consumption of UAV $m \in M$ for the computation process, and the second term represents the energy consumption when the UAV is hovering.

Likewise, the energy of the ground terminal is also constrained. The energy consumption of the ground terminal mainly comes from local execution and transmission of data to the UAV, which is shown as

$$T_n^{\text{local}} \eta_n f_n^3 + \left(1 - \varpi_n^0\right) \lambda_n s_n e_n \le E_n^e, \quad \forall n \in N, \tag{10}$$

where ϖ^0 is the proportion of tasks offloaded to the UAV. λ_n is the number of tasks arrived on the MT in unit time, and s_n is the average size of tasks executed on the MT. e_n is the energy consumed by the MT when sending one bit data to the UAV through air interface. The inequality constraint in (10) represents the limited energy of the ground terminal. The two summation terms on the left of the inequality represent the energy consumption when the task is executed locally and task data are transmitted to the UAV, respectively. The right part of the inequality is the upper limit of available energy of the local ground terminal.

Additionally, not all the UAVs require to meet the communication conditions with each ground terminal in multi-UAVs scenario, but a communication link between them must be ensured when the ground terminal offloads the task to a UAV. Hence, we can write

$$\varpi_n^m \left[\left(x_m - a_n\right)^2 + \left(y_m - b_n\right)^2 \right] \le \varpi_n^m \left(H_m^u \tan \Theta^u\right)^2. \tag{11}$$

Equation (11) shows the constraint about the communication connectivity between the ground terminal and the UAV. If the offloading variable ϖ_n^m is ignored, the left part of the inequality represents the Euclidean distance between the projection point of the UAV's hovering position on the ground plane and the ground terminal, and the right part of the inequality represents the radius of the circular coverage area formed by the UAV's radar signal on the ground. When the inequality condition in (11) is satisfied, the UAV can communicate with the ground terminal. Noted that the offloading variables are multiplied on both sides of the inequality. If $\varpi_n^m \ne 0$, it means that the ground terminal $n \in N$ offloads ϖ_n^m proportional of the task to the UAV $m \in M$. At this time, the variables ϖ_n^m on both sides of the inequality can be reduced to meet the conditions described above. If $\varpi_n^m = 0$, it means the ground terminal does not offload any part of the task to the UAV. Thus, both sides of the inequality are 0 and the inequality holds. As there is no real data flow, whether the communication link between the terminal and the UAV exists does not affect the solution of the problem and the connectivity constraint no longer needs to be considered.

Finally, the ground terminal may transmit the task to multiple UAVs at the same time, so the conservation of data flow can be written as

$$\sum_{m=0}^{M} \varpi_n^m = 0 = 1, \quad \forall n \in N. \tag{12}$$

The equality constraint in (12) shows that the sum of the proportion of transmitted data and the proportion of local data should be equal to 1 for each ground terminal.

2.5. Optimization Object. Combined with the detailed description of the objectives and constraints of the problem mentioned above, the system model can be formulated as follows:

$$\begin{aligned} P:\ & \min\left[\omega_1\left(E_n^{\text{local}} + E_m + E^{\text{fly}}\right) + \omega_2\left(T_n^{\text{local}} + T_m\right)\right], \\ \text{s.t. } & 0 \le \varpi_n^m \le 1, \quad n = 1, 2, \ldots, N, m = 1, 2, \ldots, M, \\ & x_m^{\min} \le x_m \le x_m^{\max}, \quad m = 1, 2, \ldots, M, \\ & y_m^{\min} \le y_m \le y_m^{\max}, \quad m = 1, 2, \ldots, M, \\ & H_{m,\min}^u \le H_m^u \le H_{m,\max}^u, \quad m = 1, 2, \ldots, M, \end{aligned} \tag{13}$$

where ω_1 and ω_2 are the weights of energy consumption and delay, respectively.

In addition to the constraints from (9) to (12), those new constraints in (13) are the initial feasible region of the optimization variables so as to find the initial search window of the algorithm.

3. Improved Genetic Algorithm-Based Computation Allocation Strategy

Considering the task dependency between multiple sites of the UAV cluster, the complexity of solving the optimal computation offloading strategy is greatly improved, and it is hard to derive the optimal solution in polynomial time. Therefore, an improved genetic algorithm is introduced to find the optimal solution of task offloading strategy [19, 20].

3.1. Encoding and Population Initialization. The first step to realize a genetic algorithm is to encode the solution space of the problem. In the proposed UAV cluster collaborative task offloading model, the application is divided into K subtasks to be offloaded. Therefore, in the genetic algorithm, each chromosome should be made up from K genes. The tasks deployed at each site can be executed locally, transmitted to

edge nodes, or accomplished on the cloud. Hence, each gene has three possible values (cloud: −1, edge: 0, local: 1) [21]. Each chromosome represents a possible solution for the computation offloading. Figure 3 illustrates a chromosome composed of K genes.

3.2. Fitness Function. The quality of an individual (solution) in the genetic algorithm is evaluated by the fitness value, which is defined as the reciprocal of the total cost of computation offloading for the application to judge the merits of each computation offloading scheme. A higher fitness value represents a lower total cost and a better computation offloading scheme [22]. The fitness function can be formulated as follows:

$$\text{fitness} = \frac{1}{\omega_1\left(E_n^{\text{local}} + E_m + E^{\text{fly}}\right) + \omega_2\left(T_n^{\text{local}} + T_m\right)}. \tag{14}$$

In each process of the iterative evolution, the strategy of elite selection is adopted to retain the elite solution for the next generation of populations. In the i-th iteration, the fitness value of each individual in the i-th population is calculated first, and these individuals will be arranged in a descending order according to their fitness value. Then, the latter half of individuals with lower fitness values are discarded, and the remaining half of individuals with higher fitness values in the population are selected into the next crossover operation [23].

3.3. Genetic Operation. After completing the encoding process of chromosomes and the selection of fitness function, the following part focuses on three operations related to chromosomes, including selection, crossover, and mutation [24, 25].

3.3.1. Selection. In the process of population evolution, an individual with a higher fitness value should be selected for the next generation of populations, while an individual with lower fitness value should be eliminated by the environment. The selection operation simulates the biological evolution that only the fittest can survive in nature, which is used to find individuals with larger fitness values in the population and inherit their excellent genes to the next generation.

The roulette wheel method is often adopted for the selection process, namely the selection is made by randomly rotating the wheel disc. Each individual is likely to be selected repeatedly. Usually, the number of selected individual chromosomes is determined by the proportion size of the fitness value of each chromosome. Let $P(z_j)$ be the probability that a chromosome z_j is selected for the next generation, as shown in the following equation:

$$P\left(z_j\right) = \frac{f\left(z_j\right)}{\sum_{j=1}^{2M} f\left(z_j\right)}, \tag{15}$$

where z_j is the chromosome in the population and $\sum_{j=1}^{2M} f(z_j)$ is the sum of fitness values of all chromosomes in the population.

1	0	0	-1	…	-1

Figure 3: A chromosome composed of K genes.

3.3.2. Crossover. Crossover operation combines two individual chromosomes in different ways to generate a new individual and improves the fitness of the new population by retaining the genes with better parents for the next generation of populations. The crossover operator recombines the genes of the parent populations to generate new chromosomes, introduce the diversity, and expand the search range of potential solutions [25].

The two-point crossover method is often used for the crossover operation in the genetic algorithms. The chromosomes are randomly selected with a certain crossover probability P_0 ($0 \le P_0 \le 1$), and then two crossover points are randomly set at the corresponding position of the selected parent chromosomes. The column gene fragments from the first crossover point to the second crossover point are exchanged with each other to generate a new pair of offspring chromosomes. Such crossover operation can inherit the characteristics of the parent generation well while maintaining the diversity of the evolution.

3.3.3. Mutation. Mutation operation simulates gene mutation in the process of evolution. Mutations in biological evolution can enrich the diversity of new individuals and make the same species have different phenotypes. Although the crossover operation expands the search range, it is limited to various combinations of existing genes. The mutation operation makes individuals generate new genes, introduce new possibilities, and maintain the diversity of the populations [26].

Individuals in the population usually mutate with a certain mutation probability P_1 ($0 \le P_1 \le 1$). Generally, individuals with lower fitness values have a larger mutation probability, while individuals with higher fitness values should be set a lower mutation probability to maintain the superiority of genes. In this paper, chromosomes are applied with 0-1 encoding while satisfying all constraints. Hence, for each chromosome in the population, the mutation operation specifically refers to the mutation of some columns of genes in the chromosome with a certain mutation probability (that is, 0 changes to 1 and 1 changes to 0). It should be noted that the position of the mutated gene is random, but the new generated individual should meet all constraints in order to ensure that it is still the feasible solution of the optimization problem, otherwise the mutation is failed and should be eliminated directly.

4. Experiment and Analysis

In the experiment, a network composed of 8 UAVs and 26 MTs is set up and the measured area is 10×10 km. MTs are randomly distributed in the test area, and their positions remain stable for a certain time. UAVs collaborate with mobile terminals to execute and transmit tasks, and the

coordinates of the base station are (−5, −5). The main simulation parameters are presented in Table 1.

In addition, the parameters of the improved genetic algorithm are set as follows: the maximum number of iterations is considered 200, the number of populations is 150, the crossover probability is 0.6, and the initial mutation probability is 0.01.

4.1. Effect of Mutation Probability on Algorithm Performance. The set of parameters of the improved genetic algorithm have a great influence on the performance of the proposed strategy, such as initial population size, mutation probability, and the number of genetic iterations. In order to obtain the best performance of the proposed strategy, the effect of mutation probability and the number of genetic iterations on the total cost is studied using the control variable method. The effect of mutation probability on the total cost is illustrated in Figure 4. At this time, the parameters of the improved genetic algorithm are set as follows: the initial population size is 150, the number of iterations is 200, the mutation probability varies from [0, 0.2], and the time delay and energy consumption weight in the fitness function are both 0.5.

As can be seen from Figure 4, when the mutation probability is within the interval [0, 0.08], the total cost initially shows a downward trend and then rises with the increase of mutation probability, and when the mutation probability exceeds 0.15, the total cost begins to fluctuate. With the increase of mutation probability, a better solution in the population may mutate into a worse solution, resulting in the degradation of performance. Therefore, the mutation probability is set at about 0.08 to achieve the optimal performance of the proposed strategy.

4.2. Effect of the Number of Genetic Iterations on the Algorithm Performance. Likewise, the effect of the number of genetic iterations on the total cost of the algorithm is illustrated in Figure 5. The specific parameters are set as follows: the initial population size 150, the mutation probability 0.08, the number of iterations increases from 1 to 200, and the time delay and energy consumption weight in the fitness function are set to 0.5.

As Figure 5 shows, although the total cost fluctuates with the increasing number of iterations, it still shows a decreasing trend. When the number of iterations reaches 80, the total cost curve is basically stable, reaching 0.66. Based on the above experiments, the genetic algorithm parameters are set as follows: the number of populations is 150, mutation probability is 0.08, and the number of genetic iterations is 200.

4.3. Relationship between the Number of Computation Tasks and the Average Time Delay. Figure 6 illustrates the performance of time delay with the increasing number of tasks arriving in the system when there are different numbers of UAVs.

It can be clearly shown in Figure 6 that no matter how many UAVs are running in the system, the overall average delay of the system is positively correlated with the arrival amount of ground terminal tasks. For any curve in Figure 6, it can be found that the growth rate of the curve rises with the increasing number of ground terminal tasks. As the M/M/1 queuing model is used to calculate the queuing time delay, a larger amount of tasks arriving in unit time causes the rapid rise of the overall delay of the system due to the limited computing capacity of the ground terminals and UAVs.

In addition, when the number of UAVs is increasing in the system, the growth of system delay becomes gentler, and the system delay is lower with more UAVs under the same number of arriving tasks. This means that the increasing number of UAVs reduces the load pressure of a single UAV as the tasks from the ground terminal can be offloaded partially to multiple UAVs. Each UAV does not need to deal with a large amount of tasks. Under the same computing capacity, it is obvious that the fewer the tasks offloaded to each UAV can lead to the lower the average delay and the lower the overall delay of the system.

4.4. Effect of the Number of UAVs on Total Utility under Different Strategies. Figure 7 depicts the comparison of the total utility of strategies proposed in this paper and in References [11, 12, 14] with the increasing number of UAVs when there are 20 mobile terminals in the system.

As can be seen in Figure 7, compared with other strategies, the total system utility of the proposed strategy is largest, and when the number of UAVs is 12, the time efficiency is highest, reaching about 1.83. When the number of UAVs is less than 12, the load on UAVS is larger, and consequently more tasks are transmitted to the base station, resulting in a lower utility. On the contrary, when the number of UAVs is greater than 12, the flight energy consumption of UAVs accounts for a big proportion leading to a lower utility. The proposed strategy uses the improved genetic algorithm to find the optimal solution for the system object and offload the computation task efficiently, jointly considering the energy consumption and time delay. The study in [14] selects the target MEC server for task offloading by using the fuzzy logic method. Although it reduces the dependence on the remote cloud, it only considers the fixed MEC servers and the applicability in the UAV cluster needs to be improved. The study in [12] proposed a flow splitting and aggregation scheme to decrease the offloading delay of the cloud server in edge computing and realize the task offloading. However, this scheme does not introduce any selection mechanism for the optimization algorithms, thus the overall utility is low, which is less than 0.95. The study in [11] formulated the task offloading problem into a nonlinear equation by using the Lagrange multiplier method and proposed the bisection search algorithm to achieve the optimal solution, which can effectively reduce the energy consumption. Nevertheless, the communication delay affects the overall utility.

Table 1: Main simulation parameters.

Parameter description	Value
Hovering height, m	10
Computation task data size, bits	$[10, 20, \ldots, 200] \times 10^3$
CPU cycle number, cycles·bit^{-1}	1000
Channel gain	1/2
Network bandwidth, MHz	5
Transmission power of MT, W	1.5
Noise power, W	2×10^{-13}
Transmission power of UAV, W	4
CPU frequency of MT, GHz	1
CPU frequency of UAV, GHz	10
Effective capacity coefficient	10^{-26}
Flight speed of UAV, m·s^{-1}	5

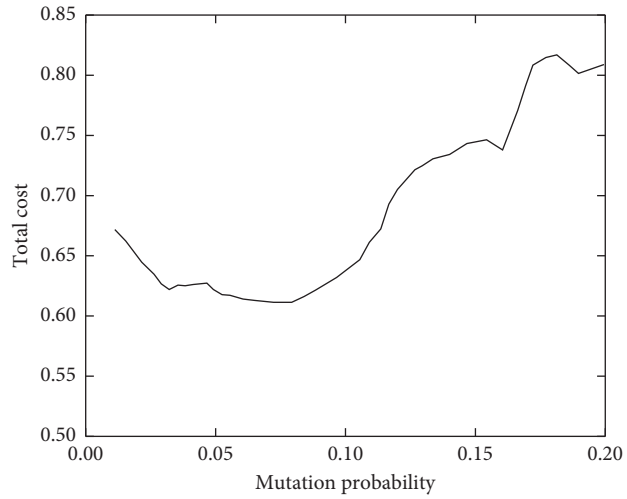

Figure 4: The effect of mutation probability on algorithm performance.

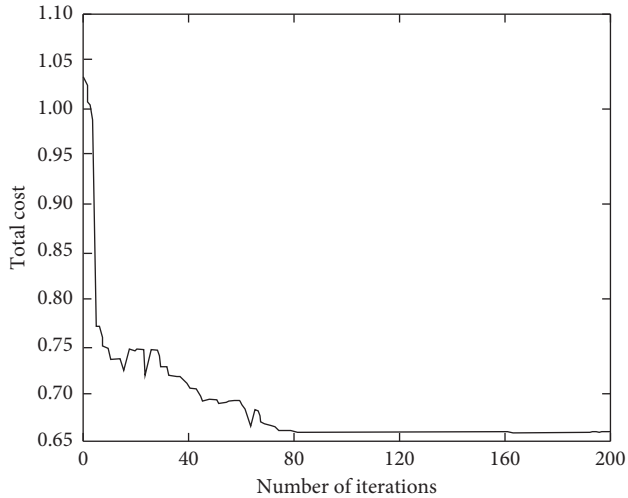

Figure 5: Effect of the number of genetic iterations on the algorithm performance.

4.5. Relationship between Application Size and Task Completion Time. In Figure 8, the MT task completion time of four offloading strategies under different task sizes is shown, in which the application in MTs is divided into four task groups and the data volume of each task group is randomly allocated.

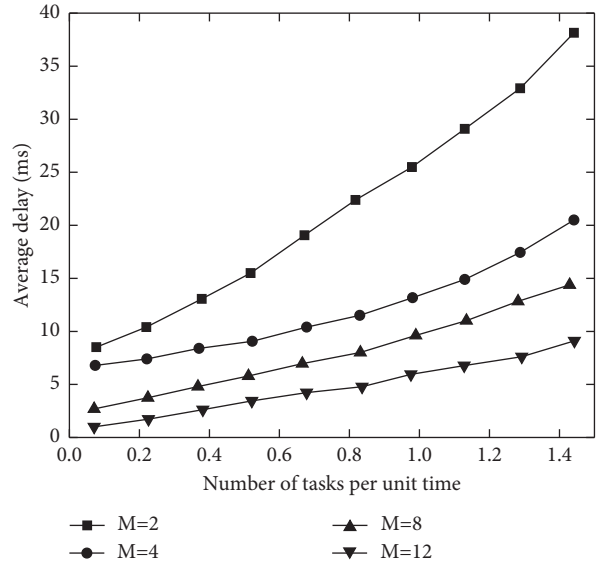

Figure 6: The relationship between the number of tasks and the average time delay under different numbers of UAVs.

As Figure 8 depicts, with the increase of data size, the task completion time also becomes higher, but the performance of the proposed strategy is always better than the other three strategies, and the maximum task completion time is about 110 ms. It should be noted that the performance gap between the proposed strategy and the strategy in Reference [14] becomes smaller and smaller as the application data size increases. In the proposed strategy, when the application data size is large, the UAV will choose to relay the task to the BS for execution and the UAV trends to fly to the places closer to the MT (or BS) to collect data (or transmit offloaded data), thus the communication channel conditions can be improved, thus reducing the task completion time. The proposed offloading scheme is similar to that in [14], which uses the fuzzy logic method, so the performance gap with task completion time is gradually narrowed. Similarly, the study in [11] proposed the bisection search algorithm to solve the offloading and resource allocation optimization problem. Although an ideal offloading scheme was selected, the algorithm is very complex and the computation and communication time are very long, taking more than 200 ms. The study in [12] uses the flow splitting and aggregation scheme to improve the offloading delay of the cloud server in edge computing, but it lacks an efficient optimization algorithm. Therefore, with the increase of application data volume, the task completion time soars, reaching more than 330 ms. Due to the complex task dependency between multiple sites of the UAV cluster, it is much more difficult to find the optimal task offloading scheme. Therefore, the proposed strategy adopts the improved genetic algorithm to solve this problem, taking energy consumption and delay into consideration

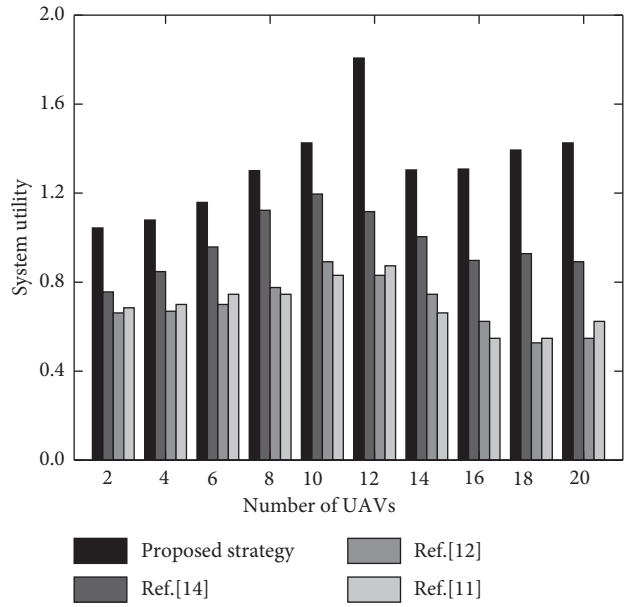

Figure 7: Effect of the number of UAVs on total utility under different strategies.

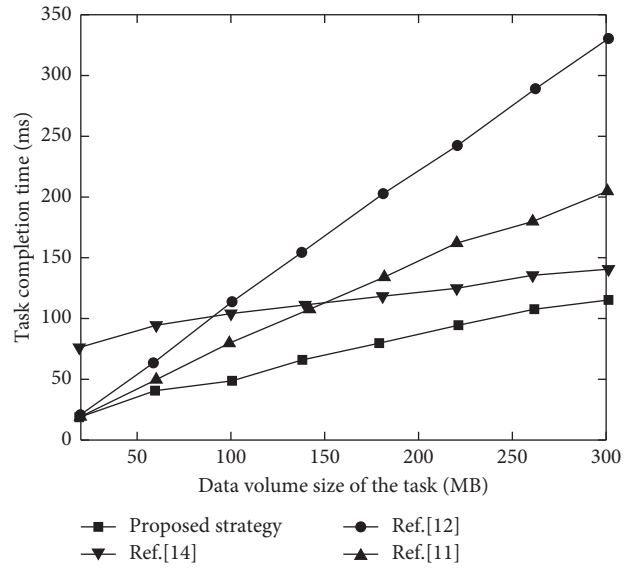

Figure 8: Relationship between application size and task completion time.

simultaneously, and finds the optimal offloading scheme with minimum target, which can reduce the system delay and energy consumption.

5. Conclusion

The emergence of edge computing has solved the problem of high network delay in the traditional cloud computing, but it needs to invest a lot of infrastructure construction. Therefore, a UAV cluster collaborate task offloading strategy using improved genetic algorithm in MEC is proposed, taking advantages of low cost and high mobility of UAVs. The improved genetic algorithm is used to solve the optimization function of minimizing the delay and energy consumption in the UAV cluster collaborate task offloading model, so as to obtain the optimal task offloading scheme. The simulation results conducted on MATLAB show that the improved genetic algorithm can improve the performance of offloading strategy. When the mutation probability is 0.08 and the number of genetic iterations is 200, the total cost of the system tends to be 0.66, and its energy consumption and delay reach the optimal value. The total utility is highest, reaching about 1.83 when the number of UAVs is 12, and the maximum task completion time is about 110 ms, which are both better than other comparison methods.

The proposed strategy does not consider that the tasks of different terminals or tasks in different periods of the same terminal may not be the same type and their requirements for computing resources or storage capacity of edge nodes are generally different. Therefore, we intend to classify the tasks into different types to meet the rapid growth of various application scenarios in the future work.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

References

[1] Z. Song, Y. Liu, and X. Sun, "Joint task offloading and resource allocation for NOMA-enabled multi-access mobile edge computing," *IEEE Transactions on Communications*, vol. 69, no. 3, pp. 1548–1564, 2021.

[2] F. Wang, J. Xu, and S. Cui, "Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2443–2459, 2020.

[3] Q.-V. Pham, L. B. Le, S.-H. Chung, and W.-J. Hwang, "Mobile edge computing with wireless backhaul: joint task offloading and resource allocation," *IEEE Access*, vol. 7, no. 99, pp. 16444–16459, 2019.

[4] T. Zhang, Y. Xu, J. Loo, D. Yang, and L. Xiao, "Joint computation and communication design for UAV-assisted mobile edge computing in IoT," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5505–5516, 2020.

[5] Y. He, D. Zhai, F. Huang, D. Wang, X. Tang, and R. Zhang, "Joint task offloading, resource allocation, and security assurance for mobile edge computing-enabled UAV-assisted VANETs," *Remote Sensing*, vol. 13, no. 8, pp. 1547–1557, 2021.

[6] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4832–4841, 2020.

[7] G. S. S. Chalapathi, V. Chamola, C.-K. Tham, G. Gurunarayanan, and N. Ansari, "An optimal delay aware task assignment scheme for wireless SDN networked edge cloudlets," *Future Generation Computer Systems*, vol. 102, no. 3, pp. 862–875, 2020.

[8] M. Mukherjee, S. Kumar, C. X. Mavromoustakis et al., "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6050–6058, 2020.

[9] U. Saleem, Y. Liu, S. Jangsher, X. Tao, and Y. Li, "Latency minimization for D2D-enabled partial computation offloading in mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 99, pp. 4472–4486, 2020.

[10] X. He, R. Jin, and H. Dai, "Peace: privacy-preserving and cost-efficient task offloading for mobile-edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1814–1824, 2020.

[11] L. Yang, G. Xu, J. Ge, P. Liu, and X. Fu, "Energy-efficient resource allocation for application including dependent tasks in mobile edge computing," *KSII Transactions on Internet and Information Systems*, vol. 14, no. 6, pp. 2422–2443, 2020.

[12] Y. Ito and H. Koga, "Improving offload delay using flow splitting and aggregation in edge computing," *IEICE Communications Express*, vol. 8, no. 12, pp. 468–473, 2019.

[13] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: a matching-theoretic framework," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[14] M. D. Hossain, T. Sultana, V. Nguyen et al., "Fuzzy based collaborative task offloading scheme in the densely deployed small-cell networks with multi-access edge computing," *Applied Sciences*, vol. 10, no. 9, pp. 3115–3126, 2020.

[15] X. Chen, Z. Liu, Y. Chen, and Z. Li, "Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks," *IEEE Access*, vol. 7, no. 3, pp. 184172–184182, 2019.

[16] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, no. 8, pp. 26652–26664, 2019.

[17] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: an energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2163–2176, 2021.

[18] C. Li, W. Chen, J. Tang, and Y. Luo, "Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment," *Computer Communications*, vol. 145, no. 9, pp. 193–202, 2019.

[19] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. Franklin de Castro, "An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing," *Computer Networks*, vol. 194, no. 4, pp. 108146–108156, 2021.

[20] X. Xu, Q. Liu, Y. Luo et al., "A computation offloading method over big data for IoT-enabled cloud-edge computing," *Future Generation Computer Systems*, vol. 95, no. 6, pp. 522–533, 2019.

[21] L. Tang, B. Tang, L. Kang, and L. Zhang, "A novel task caching and migration strategy in multi-access edge computing based on the genetic algorithm," *Future Internet*, vol. 11, no. 8, pp. 181–193, 2019.

[22] R. Ezhilarasie, A. Umamakeswari, M. S. Reddy, and P. Balakrishnan, "Grefenstette bias based genetic algorithm for multi-site offloading using docker container in edge computing," *Journal of Intelligent and Fuzzy Systems*, vol. 36, no. 3, pp. 2419–2429, 2019.

[23] H. A. Almashhadani, X. Deng, S. N. A. Latif, M. M. Ibrahim, and A. H. Alshammari, "An edge-computing based task-unloading technique with privacy protection for Internet of connected vehicles," *Wireless Personal Communications*, vol. 6, no. 1, pp. 1–22, 2021.

[24] L. Bing, 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.

[25] B. Huang, Z. Li, P. Tang et al., "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Future Generation Computer Systems*, vol. 97, no. 8, pp. 755–774, 2019.

[26] L. Ruiz, R. J. D. Barroso, I. De Miguel et al., "Genetic algorithm for holistic VNF-mapping and virtual topology design," *IEEE Access*, vol. 8, no. 7, pp. 55893–55904, 2020.