Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8867157, 12 pages
https://doi.org/10.1155/2020/8867157

Research Article

User-Edge Collaborative Resource Allocation and Offloading Strategy in Edge Computing

Zhenquan Qin, Xueyan Qiu, Jin Ye, and Lei Wang

School of Software, Dalian University of Technology, 116620, China

Correspondence should be addressed to Zhenquan Qin; qzq@dlut.edu.cn

Received 16 March 2020; Revised 11 May 2020; Accepted 25 May 2020; Published 12 June 2020

Academic Editor: Wei Wang

The foundation of urban computing and smart technology is edge computing. Edge computing provides a new solution for large-scale computing and saves more energy while bringing a small amount of latency compared to local computing on mobile devices. To investigate the relationship between the cost of computing tasks and the consumption of time and energy, we propose a computation offloading scheme that achieves lower execution costs by cooperatively allocating computing resources by mobile devices and the edge server. For the mixed-integer nonlinear optimization problem of computing resource allocation and offloading strategy, we segment the problem and propose an iterative optimization algorithm to find the approximate optimal solution. The numerical results of the simulation experiment show that the algorithm can obtain a lower total cost than the baseline algorithm in most cases.

1. Introduction

The rapid development of the Internet of Things (IoT) and mobile devices has facilitated the development of emerging applications such as machine learning [1] and face recognition [2]. Relying on these rapidly developing new IoT technologies and the Internet of Everything scenarios, the process of urban informatization is greatly accelerated, and smart cities have become a vision for development [3–5]. Smart cities can perform real-time analysis based on urban big data, provide new models of efficient and sustainable urban governance, and establish effective communication between people and cities. In recent years, the topic of smart cities has attracted widespread attention. With the introduction of various learning technologies, people hope to make cities smarter by collecting, storing, and processing big data.

One problem is that most of the emerging IoT technologies, including various learning technologies, are computationally intensive. However, due to volume and heat considerations, sensor nodes or other mobile devices that collect big data for smart cities usually have limited processing power, battery power, and storage space, making it difficult to meet the needs of all computing tasks. Cloud computing provides a solution [6, 7]. Migrating computing tasks from the device to the cloud server eliminates the limitations of the device itself. However, the cloud server is remotely isolated from the device in terms of topology and geographical location, and multiple storage and forwarding, as well as long-distance communications, cause huge transmission and propagation delays [8]. Smart cities need to establish efficient communication with people, and high latency will reduce the user experience. And for applications with high real-time requirements, this high latency is intolerable, and even in extreme cases, high latency can endanger user safety, such as autonomous driving [9], automatic navigation boats [10], and health care [11].

As an emerging distributed cloud architecture, edge computing provides a new solution for smart cities [12, 13]. Deploy servers near the demanders of the service, avoiding the high latency of a centralized cloud. At the same time, offloading all tasks to the edge server saves more energy compared to placing all tasks locally on the device. Moreover, edge computing can effectively reduce the pressure on the core network. For example, smart transportation is

an important part of smart cities. The intelligent transportation system analyzes the videos and videos acquired by a large number of traffic cameras to generate highly efficient traffic strategies. Traditional video surveillance systems must upload all surveillance videos to the cloud before analyzed, which will increase the traffic load on the core network. If edge computing is used, you can analyze at the edge server and reduce network pressure and the energy consumption of the entire monitoring system [14], which gives new vitality to smart transportation.

However, edge servers often have limited computing resources [15], considering the economic benefits and scalability of deployment. One reason is that edge servers are deployed in large numbers and close to users, and the economic benefits of deployment need to be considered. So a single edge server does not need and cannot have as many resources as a cloud center. Then, the edge server cannot fully process all tasks. If all tasks are transmitted to the edge server without any difference, it will result in low processing efficiency and long processing time for the task. So how to offload tasks based on demand is a question worth studying.

The process of migrating computing tasks from a mobile device to an edge server or a cloud center is called offloading [16–18]. The development of an offloading strategy is a classic and important issue in the field of edge computing. Offloading tasks to the edge server will inevitably lead to higher latency than local execution while reducing power consumption. Therefore, mobile devices need to choose the appropriate offloading strategy to meet their needs, such as minimizing task execution time [19–21], minimizing power consumption [22, 23], balancing delays, and power consumption [24, 25].

In many of the past work, resource allocation problems have been discussed. Most of them focus on solving the allocation of computing resources or the allocation of communication resources for edge servers [15, 22–24]. Some discussed the computational resource allocation and transmission power control of mobile devices [19, 25]. In fact, the resource allocation of mobile devices and edge servers will affect each other and work together to generate an offloading policy. Therefore, it is necessary to study the relationship between computing resource allocation and offloading decision generation.

Based on the above, we summarize the motivation of the work as follows: (1) The offloading decision should be reasonable. The constraints of computing resources and energy need to be considered to achieve a better task assignment. (2) Delay and energy consumption are two important parameters of the task. The allocation of computing resources has an impact on latency and energy consumption. (3) A parameter should be developed based on the deadline to express the real-time requirements of the task. (4) In heterogeneous edge servers, the execution time of tasks is different from local execution. The task execution costs need to explicitly consider the heterogeneity of edge servers.

In this paper, we proposed a collaborative computing resource allocation scheme and an offloading strategy for mobile devices and the edge server. Our contributions are as follows:

(1) We propose an integrated framework for computing resource allocation and computational task offloading in a mobile edge computing network

(2) We propose a computation offloading scheme that achieves lower execution costs by cooperatively allocating computing resources by mobile devices and the edge server

(3) Compared to the weighting factors of execution delay and energy consumption based on user preferences, we add the deadline to the definition of the weighting factor to reflect the real-time requirements of the task

(4) We segment the complex problem and propose Genetic-Algorithm-Based-Iterative (GABI) algorithm to obtain an approximate optimal solution

(5) The paper is organized as follows. In Section 2, we review the related works. Section 3 introduces the system modeling and problem formulation. Section 4 gives the solution. The simulation results are given and discussed in Section 5. Section 6 concludes this paper

2. Related Works

For mobile edge computing, computing offloading is the key technology. A lot of work has been done on the computation offloading technology to achieve the purpose of reducing calculation delay [8, 19–21] or saving equipment energy [22, 23, 26]. In [8], Xiao and Krunz studied the trade-off between users' QoE and fog node power efficiency in fog computing networks, focusing on users' QoE measured by average service response time. The author proposed a novel cooperative strategy called offload forwarding, and further proposed a distributed optimization algorithm based on distributed alternating direction method of multipliers (ADMM) to maximize the user's QoE. Rodrigues et al. in [19] considered computing and communication at the same time. To minimize service latency, the author controlled processing delay through virtual machine migration and improved transmission delay through transmission power control. In [20], Kao et al. proposed a complete polynomial-time approximation scheme (FPTAS) Hermes to solve the formulated NP-hard problem, thereby, minimizing application delay while meeting the specified resource utilization constraints. Jia et al. in [21] present an online task offloading algorithm that minimizes the completion time of the application on the mobile device. In addition to the line topology task graphs, the authors further considered the general topology task graphs. For concurrent tasks, the author used a load-balancing heuristic to increase the parallelism between mobile devices and the cloud. Some literature focuses on the energy consumption of mobile devices. In [22], You and Huang minimized the weighted sum of mobile energy consumption by solving the convex optimization problem under the constraint of calculating the waiting time. The author discussed both the infinite and limited edge cloud computing capabilities. For the latter, the author proposed

a suboptimal resource allocation algorithm to reduce complexity. You et al. in [23] studied the resource allocation of multiuser mobile edge computing systems based on time division multiple access (TDMA) and orthogonal frequency division multiple access (OFDMA), and by solving the convex optimization problem and the mixed-integer problem, the weighted total mobile energy consumption under the constraint of computing the waiting time is minimized. In [26], Zhang et al. proposed an offloading scheme that optimizes energy with guaranteed delay. This scheme considers the link to the status of the fronthaul network and the backhaul network at the same time and uses an artificial fish swarm algorithm for global optimization.

Time delay and energy are the main indicators to evaluate the performance of offloading calculation. It is meaningful to study the trade-off between them. The author in [15] designed the QoE maximization framework. QoE is a cost reduction achieved by offloading tasks to fog nodes or remote cloud servers, where the cost of performing tasks includes computing energy and computing latency. In [24], Zhang et al. proposed an online dynamic task allocation plan to study the trade-off between energy consumption and execution delay of the MEC system with energy harvesting function. An online dynamic Lyapunov-optimized offloading algorithm (ODLOO) is proposed, which could determine task allocation, reduce the execution delay by increasing the running frequency of the local CPU, and save energy by selecting a suitable data transmission channel. [25] proposed an energy-aware offloading scheme, which can jointly optimize the allocation of communication and computing resources with limited energy and sensitive delay. To calculate the mixed-integer nonlinear problem of shunting and resource allocation, Zhang et al. proposed an iterative search algorithm that combines internal penalty functions with DC programming to find the best solution. [27] studied the joint design of local execution and calculation offloading strategies in a multiuser MEC system. Mao et al. proposed an online algorithm based on Lyapunov optimization, which determines the CPU cycle frequency of local execution and the transmit power and bandwidth of the computation load distribution. The author used simulation experiments to verify that the proposed algorithm can balance the power consumption of mobile devices and the quality of computing experience. However, with the exception of [25], most of the work that studied the trade-off between energy consumption and delay has not clearly defined the weighting factors that weigh the two. Most of them described the weighting factors as a task attribute that changes according to user needs or verified the effectiveness of the proposed algorithm by adjusting the weighting factors. In [25], the author defined the weighting factor as the current residual energy ratio of mobile devices, which is an innovative idea and also gives some inspiration to this article.

The deadline is an important attribute for computing tasks [23, 24, 27–30]. The deadline of the task represents the delay tolerance of the task. In the work focusing on task scheduling, the deadline is used as an evaluation index. As in [31], the author formulated a task scheduling and offloading strategy so that more tasks can meet the deadline requirements, and the number of tasks that have exceeded the deadline, that is, the error rate, is used as the evaluation index. In most of the tasks that focus on calculating the offload, the deadline of the task is only a constraint, which does not give full play to the attribute of deadline and cannot express the real-time requirements of the task. In this paper, inspired by the literature [25], the deadline is introduced into the definition of the weighting factor.

Limited resources are a characteristic of edge computing systems, which prompted us to study the problem of resource allocation. Allocable resources include the communication power, channel bandwidth, and computing ability of user equipment and edge servers. Considering the allocation of multiple resources at the same time complicates the problem. The authors in [15] studied computing resource allocation, formulated a computational offloading game to model competition among IoT users, and effectively allocated the limited processing power of fog nodes. In [22], You and Huang formulated the optimal resource allocation problem as a convex optimization problem, considering the situation of unlimited and limited cloud computing capabilities. The authors in [23] further considered the OFDMA system, whose optimal resource allocation is formulated as a mixed-integer problem. Zhang et al. considered the allocation of communication and computing resources in [25], including channel selection, user equipment computing resource allocation, and communication power allocation. In [32], Yu et al. considered the allocation of radio and computing resources. They proposed a near-optimal algorithm for scheduling subcarriers and CPU time slots, respectively, and further proposed a joint scheduling algorithm to manage subcarriers and CPUs coordinately, thereby, achieving the goal of reducing energy consumption. In [33], Sun et al. studied the joint problem of network economics and resource allocation in MEC and considered the incentive framework that can maximize system efficiency in the Industrial Internet of Things (IIoT). They proposed two types of dual auction schemes with dynamic pricing, namely, a breakeven-based double auction (BDA) and a more efficient dynamic pricing based double auction (DPDA) to determine the matching pair between the IIoT MD and the edge server, and Pricing mechanism to achieve high system efficiency under local constraints.

3. System Model and Problem Formulation

3.1. System Model. We consider a network system with N mobile devices and an edge server as shown in Figure 1. The channel bandwidth between the mobile device $u_i (i \in \{1, 2, \cdots, N\})$ and the edge server is w. The propagation delay of edge server and mobile device communication is negligible because the distance between the two is often only a single hop or a limited number of hops.

In this network, the mobile device u_i with limited energy $E_i^{\max}$ will generate M tasks that need to be completed in each time slot. The jth $(j \in \{1, 2, \cdots, M\})$ task generated by device u_i in time slot t $(t \in \{1, 2, \cdots, T\})$ is represented as $\tau_{i,j,t} = (d_{i,j,t}, c_{i,j,t}^U, c_{i,j,t}^E, t_{i,j,t}^{\text{deadline}}, s_{i,j,t})$, where $d_{i,j,t}$ represents the

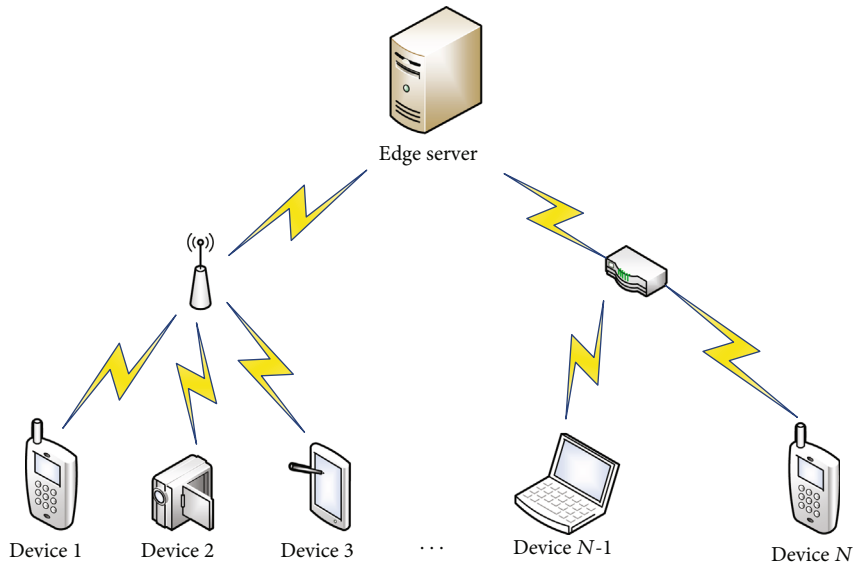

Figure 1: Network system model.

amount of data for the task, $c^U_{i,j,t}$ denotes the number of CPU cycles required for the task if it is executed locally, $c^E_{i,j,t}$ means the number of CPU cycles required for the task to execute on the heterogeneous edge server, and $t^{\text{deadline}}_{i,j,t}$ is the tolerable execution time. $s_{i,j,t}$ is the offloading strategies of task $\tau_{i,j,t}$. $s_{i,j,t} = 1$ when the task is performed locally, if not, $s_{i,j,t} = 0$.

In each time slot, after the mobile device generates the tasks and allocates the local computing resources, the information of the task and the local computing resource allocation scheme are transmitted to the edge server. The edge server calculates the offloading policy and returns it to the mobile device. The task is executed locally or offloaded according to the offloading policy.

3.2. Local Computing. We define $f^U_{i,j,t}$ as the CPU frequency of the mobile device u_i, which represents the local computation ability of the device. Note that this value could be changed to get a better offloading strategy. Then, the computation execution time $t^{\text{execute}^-U}_{i,j,t}$ for the locally performed task $\tau_{i,j,t}$ is

$$t^{\text{execute}^-U}_{i,j,t} = \frac{c^U_{i,j,t}}{f^U_{i,j,t}}. \tag{1}$$

We ignore the queue delay of local execution. Then, the time consumption $t^U_{i,j,t}$ of task $\tau_{i,j,t}$ local execution is

$$t^U_{i,j,t} = t^{\text{execute}^-U}_{i,j,t}. \tag{2}$$

The energy consumption can be expressed as

$$e^U_{i,j,t} = \kappa \left(f^U_{i,j,t}\right)^2 c^U_{i,j,t}, \tag{3}$$

where κ is the energy efficiency coefficient. It is related to the chip architecture, and we assume that this value is constant.

3.3. Edge Computing. When the mobile device u_i decides to offload the task $\tau_{i,j,t}$ to the edge server, it needs to obtain the data uplink transmission rate

$$r_{i,j,t} = w \log_2 \left(1 + \frac{p_i h_{i,j,t}}{\sigma + \sum_{k \in N, k \neq i} \left(1 - s_{k,j,t}\right) p_k h_{k,j,t}}\right), \tag{4}$$

where w is the channel bindwidth and p_i is the transmission power of the mobile device u_i. h_i and σ are channel gain and noise power. The summation term in the denominator represents interference between channels. Therefore, the transmission time of the uplink is

$$t^{\text{trans}}_{i,j,t} = \frac{d_{i,j,t}}{r_{i,j,t}}. \tag{5}$$

The time task $\tau_{i,j,t}$ consumes to execute on a heterogeneous edge server is

$$t^{\text{execute}^-E}_{i,j,t} = \frac{c^E_{i,j,t}}{f^E_{i,t}}, \tag{6}$$

where $f^E_{i,t}$ is the computing resource allocated by the edge server to the task from u_i in the time slot t. As with communication resources, computing resources are also allocated periodically.

So, we get the total time consumption of edge computing

$$t^E_{i,j,t} = t^{\text{trans}}_{i,j,t} + t^{\text{execute}^-E}_{i,j,t}. \tag{7}$$

We ignore the return time because many computationally intensive applications have a much smaller amount of data than the input data.

Energy consumption is

$$e^E_{i,j,t} = p_i t^{\text{trans}}_{i,j,t}. \tag{8}$$

In general, the time consumption and energy consumption constitute the execution cost of the task. The cost of locally executed task $G^U_{i,j,t}$ and offloaded task $G^E_{i,j,t}$ are expressed as the weighted sum of time and energy consumption

$$G^U_{i,j,t} = \alpha_{i,j,t} t^U_{i,j,t} + \left(1 - \alpha_{i,j,t}\right) \beta e^U_{i,j,t}, \tag{9}$$

$$G^E_{i,j,t} = \alpha_{i,j,t} t^E_{i,j,t} + \left(1 - \alpha_{i,j,t}\right) \beta e^E_{i,j,t}. \tag{10}$$

β is a normalization factor used to eliminate the unit difference between time and energy consumption, which can be equal to the ratio of average time to average energy consumption.

The weight factor $\alpha_{i,j,t}$ ($\alpha_{i,j,t} \in [0, 1]$) is the task's trade-off between time and energy consumption. By changing the weighting factor, the user can achieve faster execution or more energy-efficient execution. In this paper, we define the weighting factor by the time consumption of the task

$$\alpha_{i,j,t} = \frac{t^{\text{execute}^- U}_{i,j,t}}{t^{\text{deadline}}_{i,j,t}}. \tag{11}$$

The task is generated on the mobile device, and the deadline is established with the execution time on the mobile device, so it is reasonable to perform the weighting factor with the local execution time.

After adding the offloading decision $s_{i,j,t}$, the generalization of the cost of task $\tau_{i,j,t}$ is expressed as:

$$G_{i,j,t} = s_{i,j,t} G^U_{i,j,t} + \left(1 - s_{i,j,t}\right) G^E_{i,j,t}. \tag{12}$$

3.4. Problem Formulation. From what has been discussed above, we formulate problems as follows:

$$\min_{s, f^U, f^E} \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{j=1}^{M} \left[s_{i,j,t} G^U_{i,j,t} + \left(1 - s_{i,j,t}\right) G^E_{i,j,t} \right], \tag{13}$$

$$\text{s.t.} \quad f^{\min}_i \le f^U_{i,j,t} \le f^{\max}_i, \quad \forall i, j, t, \tag{14}$$

$$0 \le \sum_{i=1}^{N} f^E_{i,t} \le F^E, \quad \forall t, \tag{15}$$

$$s_{i,j,t} t^U_{i,j,t} + \left(1 - s_{i,j,t}\right) t^E_{i,j,t} \le t^{\text{deadline}}_{i,j,t}, \quad \forall i, j, t, \tag{16}$$

$$\sum_{t=1}^{T} \sum_{j=1}^{M} \left[s_{i,j,t} e^U_{i,j,t} + \left(1 - s_{i,j,t}\right) e^E_{i,j,t} \right] \le E^{\max}_i, \quad \forall i, j, t, \tag{17}$$

$$s_{i,j,t} \in \{0, 1\}, \quad \forall i, j, t. \tag{18}$$

Constraint (14) indicates that the computing resource allocated to a local task cannot exceed the maximum CPU frequency, and constraint (15) indicates that the computing resources allocated by an edge server in each time slot cannot exceed its maximum capacity. Constraint (17) ensures that the mobile device is able to complete computing tasks before all energy is consumed. Constraint (18) indicates that the offloading decision is a 0-1 variable.

The objective function is nonlinear and contains the multiplication of variables, so this is a complex mixed-integer nonlinear optimization problem with binary variables. This kind of problem is difficult to solve, so we will divide the problem in the following paper, and iteratively use genetic algorithm method to find the approximate solution and carry out simulation experiments to prove the validity of the solution.

4. Solution

4.1. Local Execution Cost. The cost $G^U_{i,j,t}$ of the local execution calculation task is split, and the specific expressions of the parameter weight factor, time consumption, and energy consumption are brought in.

$$\begin{aligned} G^U_{i,j,t} &= \alpha_{i,j,t} t^U_{i,j,t} + \left(1 - \alpha_{i,j,t}\right) \beta e^U_{i,j,t} \\ &= \frac{\left(c^U_{i,j,t}\right)^2}{\left(f^U_{i,j,t}\right)^2 t^{\text{deadline}}_{i,j,t}} + \left(1 - \frac{c^U_{i,j,t}}{f^U_{i,j,t} t^{\text{deadline}}_{i,j,t}}\right) \beta \kappa c^U_{i,j,t} \left(f^U_{i,j,t}\right)^2. \end{aligned} \tag{19}$$

After observation, we found that the value of $G^U_{i,j,t}$ depends only on $f^U_{i,j,t}$, so $G^U_{i,j,t}$ is a function of $f^U_{i,j,t}$.

Lemma 1. *The function $G^U_{i,j,t}(f^U_{i,j,t})$ is unimodal.*

Lemma 1 can be easily proved by the derivative and monotonicity of the function. Simplify the formula (19) and it looks like $y = ax^2 + bx + c(1/x^2)$, where $a > 0$, $b < 0$, and $c > 0$. The second derivative is $\mathrm{d}^2y/\mathrm{d}x^2 = 6c(1/x^4) + 2a$. In the positive interval, the second derivative is always positive, then, the first derivative is monotonically increasing. The first derivative $\mathrm{d}y/\mathrm{d}x = 2ax - 2c(1/x^3) + b$ is continuous when $x > 0$, and as x tends to zero and positive infinity, the derivative tends to be negative infinity and positive infinity. From the interval value theorem, there is a point that the first derivative is zero, and the point is a minimum point.

From Lemma 1, we get the CPU frequency $f^*_{i,j,t}$ that minimizes the local execution cost, which is the ideal best, but we still need to consider the constraints (14), (16), and (17).

In constraint (14), we have to consider the upper and lower bounds of $f^U_{i,j,t}$. And in constraint (16),

$$\begin{aligned} t^U_{i,j,t} = \frac{c^U_{i,j,t}}{f^U_{i,j,t}} \le t^{\text{deadline}}_{i,j,t}, \\ f^U_{i,j,t} \ge \frac{c^U_{i,j,t}}{t^{\text{deadline}}_{i,j,t}} \end{aligned}. \tag{20}$$

For a single task, we can override the constraint (17) to make it look simpler

$$\kappa\left(f_{i,j,t}^{U}\right)^{2} c_{i,j,t}^{U} \le E_{i,j,t}^{\max},$$
$$f_{i,j,t}^{U} \le \sqrt{\frac{E_{i,j,t}^{\max}}{\kappa c_{i,j,t}^{U}}}, \tag{21}$$

where $E_{i,j,t}^{\max}$ is a real-time value indicating the remaining battery power of the device while performing the task.

The CPU frequency of the mobile device $f_{i,j,t}^{U}$ fluctuates within $[f_i^{\min}, f_i^{\max}]$. Then, we get the new bound of $f_{i,j,t}^{U}$

$$f_{i,j,t}^{ub} = \min\left\{f_i^{\max}, \sqrt{\frac{E_{i,j,t}^{\max}}{\kappa c_{i,j,t}^{U}}}\right\},$$
$$f_{i,j,t}^{lb} = \max\left\{f_i^{\min}, \frac{c_{i,j,t}^{U}}{t_{i,j,t}^{\text{deadline}}}\right\}, \tag{22}$$

and the best value to get the lowest $G_{i,j,t}^{U}$

$$f_{i,j,t}^{U} = \begin{cases} f_{i,j,t}^{*}, & f_{i,j,t}^{lb} \le f_{i,j,t}^{*} \le f_{i,j,t,}^{ub} \\ f_{i,j,t}^{lb}, & f_{i,j,t}^{*} < f_{i,j,t}^{lb}, \\ f_{i,j,t}^{ub}, & f_{i,j,t}^{ub} < f_{i,j,t}^{*}. \end{cases} \tag{23}$$

4.2. Edge Execution Cost. We found that the problem of task execution cost on edge servers could not be considered separately like that on mobile devices.

$$G_{i,j,t}^{E} = \alpha_{i,j,t} t_{i,j,t}^{E} + \left(1-\alpha_{i,j,t}\right)\beta e_{i,j,t}^{E},$$
$$= \alpha_{i,j,t}\left(t_{i,j,t}^{trans} + \frac{c_{i,j,t}^{E}}{f_{i,t}^{E}}\right) + \left(1-\alpha_{i,j,t}\right)\beta e_{i,j,t}^{E}. \tag{24}$$

For locally executed computing tasks, different tasks are independent in terms of computing resource allocation. But it is different at the edge, where all tasks share the computing resources of one edge server. The task execution cost and the computing resources allocated by the edge server are inversely related. Therefore, it is meaningless to study the minimum cost of a computing task separately, and the resource allocation for all tasks needs to be considered.

We can consider the minimum cost of all tasks in a time slot. The edge server allocates computing resources for newly arrived tasks in each time slot, and tasks in different time slots are independent of each other.

$$\min_{f^E} \sum_{i=1}^{N}\sum_{j=1}^{M} \alpha_{i,j,t}\left(t_{i,j,t}^{\text{trans}} + \frac{c_{i,j,t}^{E}}{f_{i,t}^{E}}\right) + \left(1-\alpha_{i,j,t}\right)\beta e_{i,j,t,}^{E} \tag{25}$$

$$\text{s.t.} \quad 0 \le f_{i,t}^{E} \le F^{E}, \quad \forall i, \tag{26}$$

$$0 \le \sum_{i=1}^{N} f_{i,t}^{E} \le F^{E}. \tag{27}$$

Note that we have not defined time constraints and energy constraints here. We will mention them in the next section and use them as conditions for developing an offload strategy.

Problem (25) is a nonlinear optimization problem. Our goal is to obtain a global optimal solution. Here, we use the genetic algorithm to get an approximate optimal solution. Then, we can get the minimum edge computing cost $G_{i,j,t}^{E}$ according to $f_{i,t}^{E}$, the calculation result of question (25).

4.3. Offloading Decision and Resource Allocation. The initial allocation of computing resources is for all tasks in this time slot. That is, before the initial offload policy, the default offload policy is to offload all tasks, $\boldsymbol{s} = 0$. An obvious problem is that due to deadline constraints and energy constraints, offloading all tasks cannot be taken as the final offloading decision, and the allocation of computing resources with the default offloading condition is not reasonable. Therefore, in this section, we propose a Genetic-Algorithms-Based-Iterative (GABI) algorithm to obtain reasonable offloading decisions and resource allocation methods through iterative calculations.

The first step is to filter based on constraints. We denote the new offloading strategy $\boldsymbol{s}'$ and assign initial values to it $\boldsymbol{s}' = \boldsymbol{s}$. In the previous section, we mentioned that time constraint is one of the conditions for filtering. After obtaining the calculation resource allocation result $f_{i,t}^{E}$, we can get the total time of edge execution $t_{i,j,t}^{E}$. If $t_{i,j,t}^{E} > t_{i,j,t}^{\text{deadline}}$, then $s'_{i,j,t} = 1$.

For energy constraints, the energy consumption of the edge calculation is independent of the calculation resource allocation, so the energy constraint is not added in the problem (25). Moreover, we have obtained local computational resource allocations that satisfy energy constraints in the previous section, while the energy consumption of edge computations is generally less than local computation. So $s'_{i,j,t} = 1$ if $e_{i,j,t}^{E} > e_{i,j,t}^{U}$. We use the matrix $\boldsymbol{s}^T$ to record the current value of $\boldsymbol{s}'$, $\boldsymbol{s}^T = \boldsymbol{s}'$. Then, the process of filtering can be expressed as Algorithm 1.

$\boldsymbol{s}^T$ will be used later as one of the conditions for stopping the iteration. The other condition is G_1, which is the total task execution cost calculated by bringing $\boldsymbol{s}^T$ and $\boldsymbol{f}^E$ into (12).

The second step is to filter tasks based on the execution costs. After the computing resource allocation of the edge server is obtained, the execution cost of the task on the edge server can be calculated and further compared with the task's local execution cost. If a task is found to cost more to execute on the edge server, we think it is better to perform it locally on the user device. The second filtering is based on the first filtering, so we define a new offloading decision matrix $\boldsymbol{s}''$ and initialize it, $\boldsymbol{s}'' = \boldsymbol{s}'$. This step is shown in Algorithm 2.

After filtering, we get a new offloading strategy $\boldsymbol{s}''$. If $\boldsymbol{s}'' = \boldsymbol{s}$, then, we think that a stable offloading strategy has

Input: task set τ , offloading strategy $\boldsymbol{s}$
Output: new decision $\boldsymbol{s}'$
1: $\boldsymbol{s}' \longleftarrow \boldsymbol{s}$
2: **for** every task $\tau_{i,j,t}$ **do**
3: Get $t^E_{i,j,t}$ using $f^E_{i,t}$ from (25). $t^E_{i,j,t} = t^{trans}_{i,j,t} + t^{execute^-E}_{i,j,t} = (d_{i,j,t}/r_{i,j,t}) + (c^E_{i,j,t}/f^E_{i,t})$.
4: Get $e^E_{i,j,t}$. $e^E_{i,j,t} = p_i t^{trans}_{i,j,t} = p_i(d_{i,j,t}/r_{i,j,t})$.
5: **if** $t^E_{i,j,t} > t^{deadline}_{i,j,t}$ **or** $e^E_{i,j,t} > e^U_{i,j,t}$ **then**
6: $s'_{i,j,t} = 1$.
7: **end if**
8: **end for**

Algorithm 1: Filtering Based on Constraints

Input: task set τ , offloading strategy $\boldsymbol{s}'$
Output: new decision $\boldsymbol{s}''$
1: $\boldsymbol{s}'' \longleftarrow \boldsymbol{s}'$
2: **for** every task $\tau_{i,j,t}$ **do**
3: Get $G^E_{i,j,t}$. $G^E_{i,j,t} = \alpha_{i,j,t} t^E_{i,j,t} + (1-\alpha_{i,j,t})\beta e^E_{i,j,t} = (c^U_{i,j,t}/f^U_{i,j,t} t^{deadline}_{i,j,t}) t^E_{i,j,t} + (1 - c^U_{i,j,t}/f^U_{i,j,t} t^{deadline}_{i,j,t})\beta e^E_{i,j,t}$.
4: Get $G^U_{i,j,t}$. $G^U_{i,j,t} = ((c^U_{i,j,t})^2/(f^U_{i,j,t})^2 t^{deadline}_{i,j,t}) + (1 - c^U_{i,j,t}/f^U_{i,j,t} t^{deadline}_{i,j,t})\beta\kappa c^U_{i,j,t}(f^U_{i,j,t})^2$.
5: **if** $G^E_{i,j,t} \geqslant G^U_{i,j,t}$ **then**
6: $s''_{i,j,t} = 1$.
7: **end if**
8: **end for**

Algorithm 2: Filtering Based on Cost

been obtained, and the GABI algorithm can return directly. So we get the final offloading strategy $\boldsymbol{s}$ and the computing resource allocation method $\boldsymbol{f}^E$.

But if the offloading strategy $\boldsymbol{s}''$ is different from the default offloading strategy $\boldsymbol{s}$, the obtained computing resource allocation strategy cannot minimize the edge execution cost because some allocated resources are not used. Therefore, computing resources need to be reallocated.

Assign the value of $\boldsymbol{s}''$ to $\boldsymbol{s}$. Once again, the optimization problem of minimizing the edge execution cost is solved. This time, $\boldsymbol{s}$ is added to remove the locally executed task:

$$\min_{\boldsymbol{f}^E} \sum_{i=1}^{N}\sum_{j=1}^{M}\left(1 - s_{i,j,t}\right)G^E_{i,j,t}, \tag{28}$$

$$\text{s.t.} \quad 0 \leq f^E_{i,t} \leq F^E, \quad \forall i, \tag{29}$$

$$0 \leq \sum_{i=1}^{N} f^E_{i,t} \leq F^E. \tag{30}$$

After the new resource allocation scheme f^E is obtained, we get $\boldsymbol{s}'$ using Algorithm 1, and get G_2 using $\boldsymbol{s}'$ and $f^{E'}$ by (12). At this time, compare G_2 and G_1. If $G_2 > G_1$, then stop iteration, and the final offloading strategy is determined to be $\boldsymbol{s}^T$. Assign $\boldsymbol{s}^T$ to $\boldsymbol{s}$, then get the final computing resource allocation method $\boldsymbol{f}^E$ through (28), and the GABI algorithm returns.

If $G_2 \leqslant G_1$, the iteration continues: recording $\boldsymbol{s}'$ with $\boldsymbol{s}^T$, filtering based on cost, judging the iteration termination conditions, iterating again, and so on.

The computing resource allocation and the offloading decision is given in Algorithm 3. Algorithm 3 shows the whole process of the GABI algorithm.

5. Simulation Result

In this section, we design simulation experiments to verify the effectiveness of our proposed algorithm. We compare the performance of our proposed method with the baseline scheme, the probabilistic offloading scheme, and the energy-aware offloading scheme. We consider an edge computing network with one edge server and N mobile devices, where N ranges from 3 to 9. We set the maximum computing power of the edge server to 4 GHz, and the computing power of mobile devices ranges from 0.2 GHz to 1 GHz. The initial CPU frequency of the mobile device is randomly generated between 0.5 and 1 GHz, which is used to calculate the normalization factor β and the deadline of tasks. Mobile device transmission power is 20 dBm.

The simulation process consists of 10 time slots in which the mobile device generates 5 computational tasks. The amount of task data is randomly generated between [300,

Input: task set τ
Output: offloading decision $\boldsymbol{s}$, computing resource allocation scheme $\boldsymbol{f}^E$
1: **for all** $t \in T$ **do**
2: Initialization: $\boldsymbol{s} \longleftarrow \mathbf{0}$, $\boldsymbol{s}' \longleftarrow \boldsymbol{s}$, $\boldsymbol{s}^T \longleftarrow \boldsymbol{s}$, $\boldsymbol{s}'' \longleftarrow \boldsymbol{s}$.
3: Get G_1 according to (24)(12).
4: **repeat**
5: $\boldsymbol{s} \longleftarrow \boldsymbol{s}''$.
6: Calculate $\boldsymbol{f}^E$ using genetic algorithms through (28).
7: Get $\boldsymbol{s}'$ by calling Algorithm 1.
8: Get G_2 using $\boldsymbol{s}'$ through (24)(9)(10)(12).
9: **if** $G_2 > G_1$ **then**
10: $\boldsymbol{s} \longleftarrow \boldsymbol{s}^T$.
11: Calculate $\boldsymbol{f}^E$ using genetic algorithms through (28).
12: **return** $\boldsymbol{s}, \boldsymbol{f}^E$.
13: **end if**
14: $\boldsymbol{s}^T \longleftarrow \boldsymbol{s}$.
15: Calculate G_1 using $\boldsymbol{s}^T$ through (24)(9)(10)(12).
16: Get $\boldsymbol{s}''$ by calling Algorithm 2.
17: **until** $\boldsymbol{s}'' = \boldsymbol{s}$
18: **return** $\boldsymbol{s}, \boldsymbol{f}^E$.
19:**end for**

Algorithm 3: Genetic-Algorithms-Based-Iterative(GABI) algorithm

1200] KB. The local computing resources required to complete the task are randomly generated between [0.1, 1] GHz. Considering the heterogeneity of devices, the required computing resources performed on the edge server is [0.2, 1.2] times than that of local execution. The task deadline is randomly generated in [0.5, 5] s, and the value is guaranteed to be greater than the local task execution time calculated by the initial CPU frequency of the mobile device.

The total bandwidth of the network is 2 MHz, which is evenly distributed by all mobile devices. The path loss model between the mobile device and the edge server is considered as the lognormal distribution.

In Figure 2, we consider the impact of offloading strategy on total cost, time, and energy consumption under different mobile device quantities. "All local" and "All MEC" in Figure 2 represent all tasks executed locally and all offloading schemes. As shown, compared with two baselines, our solution maintains lower cost, medium execution time, and power consumption in most cases. When there are fewer mobile devices, the edge server allocates more resources to each user, so the task execution time can be even lower than the local execution, and the total task cost is lower. As the number of users increases, the edge execution time becomes longer, so the total cost of edge execution approaches or even exceeds local execution, even though the energy consumed by edge execution is still much lower than local execution. The energy consumption of local execution tasks is several times or even ten times that of edge execution, so even if the time consumption is slightly lower, there is still a higher total cost.

In Figure3, we compare the performance of our scheme and the probabilistic offloading scheme. The idea of the probabilistic offloading scheme is that the computing server has a certain probability of being offloaded by the edge server. Whether to offload is determined by probability, without comparing the cost of tasks performed locally and edge. We compared the cases with unloading probabilities of 0.2, 0.5, and 0.8. The experimental results show that our proposed scheme can always achieve the lowest task execution cost. When the number of users is small, the computing resources of the edge server are less competitive, and the total cost of the solution with a high probability of offloading is lower. This is consistent with the "ALL MEC" baseline solution in Figure2 when the number of users is low. As the number of users increases, the low-probability offloading solution gradually achieves lower costs. In terms of time and energy consumption, our solution is also at a relatively low level.

In Figure4, we compare the impact of the definition and treatment of weighting factors in this article and in the literature [25]. The document [25] defines the weighting factor $\alpha = \alpha * r^E$, where r^E is the ratio of the remaining energy of the current user equipment to the maximum battery capacity. In order to reflect the effect of the method in [25], we have selected a special piece of data. In this simulation, the remaining power of the user equipment is small. As a result, at the beginning of the third time slot, the weighting factor of the task calculated by the [25] method decreases rapidly, as shown by the polyline on the way with the legend "Rest Energy Defines α". In order to save energy, most tasks choose to offload, even if the cost of edge execution is much higher than local execution. The method in this paper does not focus on energy perception, so in most cases, the strategy of keeping the lowest cost is selected. When the local execution cost of sending some tasks is low, choose not to offload.

In Figure 5, we consider the impact of local computing resource allocation on total cost, time, and energy

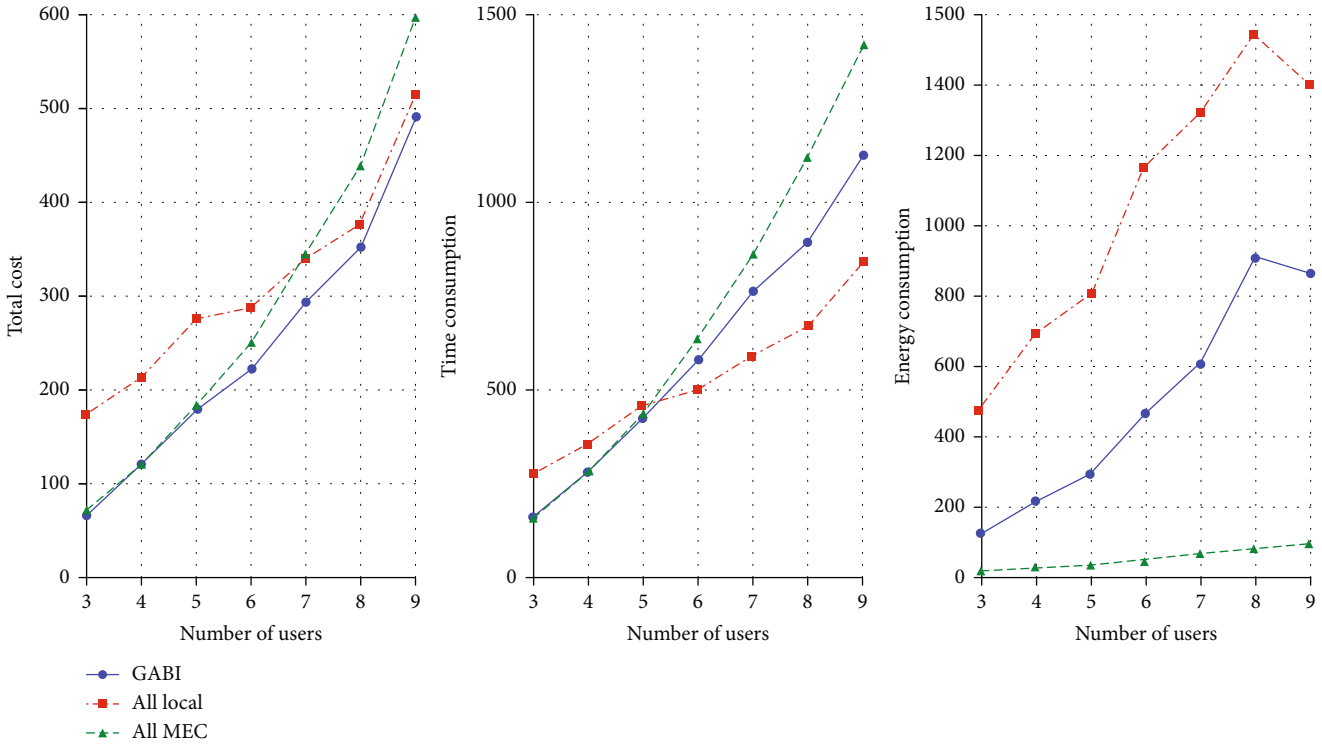

Figure 2: The impact of offloading strategy on total cost, time, and energy consumption under different mobile device quantities.

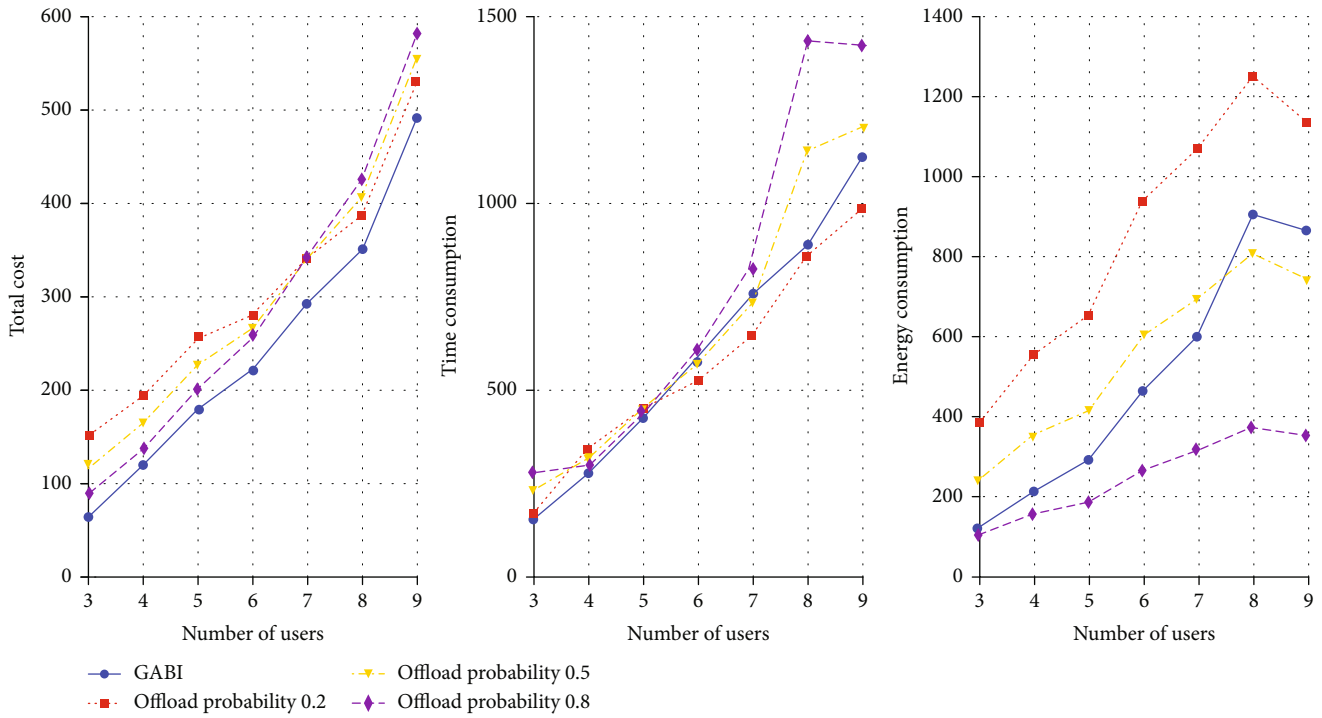

Figure 3: The impact of offloading strategy on total cost, time, and energy consumption under different mobile device quantities.

consumption under different user quantities. "fUmax", "fUmin", and "fUmid", respectively, represent local computing resource allocation with $f_i^{\max}$, $f_i^{\min}$, and $(f_i^{\max} + f_i^{\min})/2$. "fUoriginal" stands for the CPU frequency preset by the user equipment. Our approach to finding the best local resource allocation scheme is feasible and effective. Compared to the for baselines, our proposed method always finds the lowest cost. Higher CPU frequencies result in faster

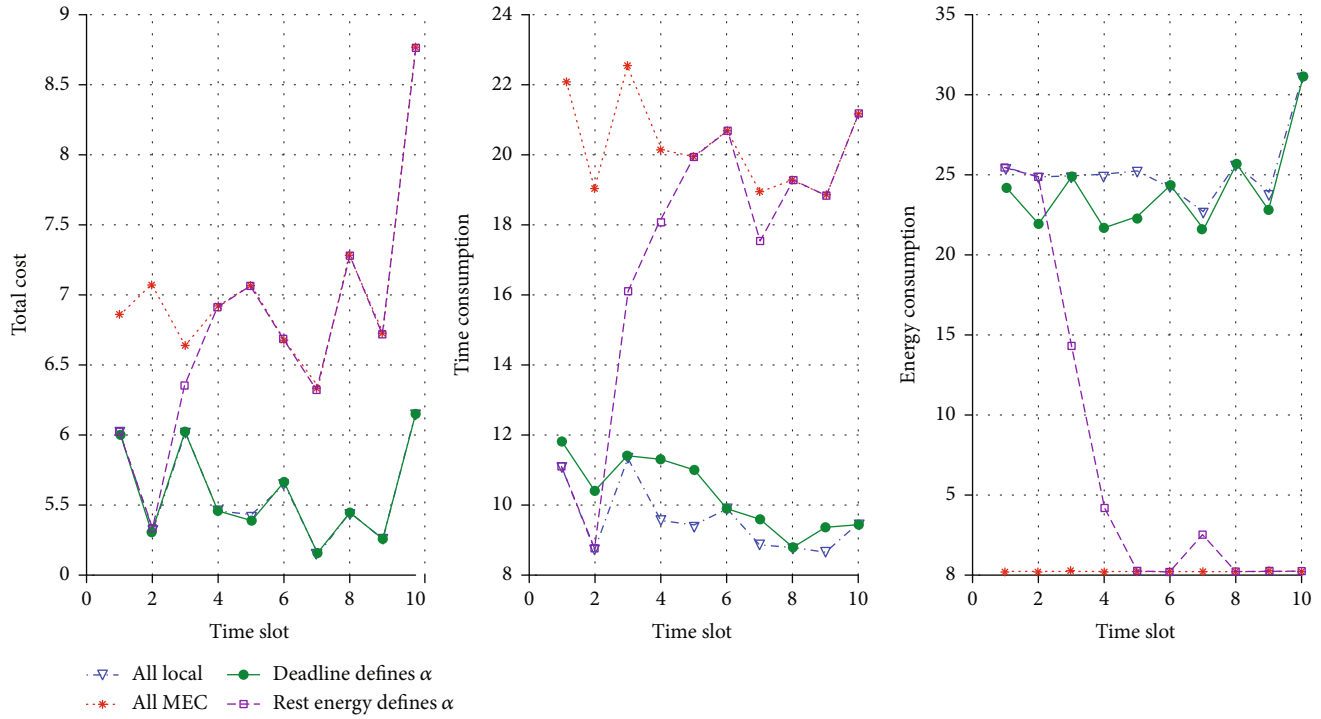

Figure 4: In multiple consecutive time slots, compare the impact of the proposed scheme and the energy-aware offloading scheme on the total cost, delay, and energy consumption.

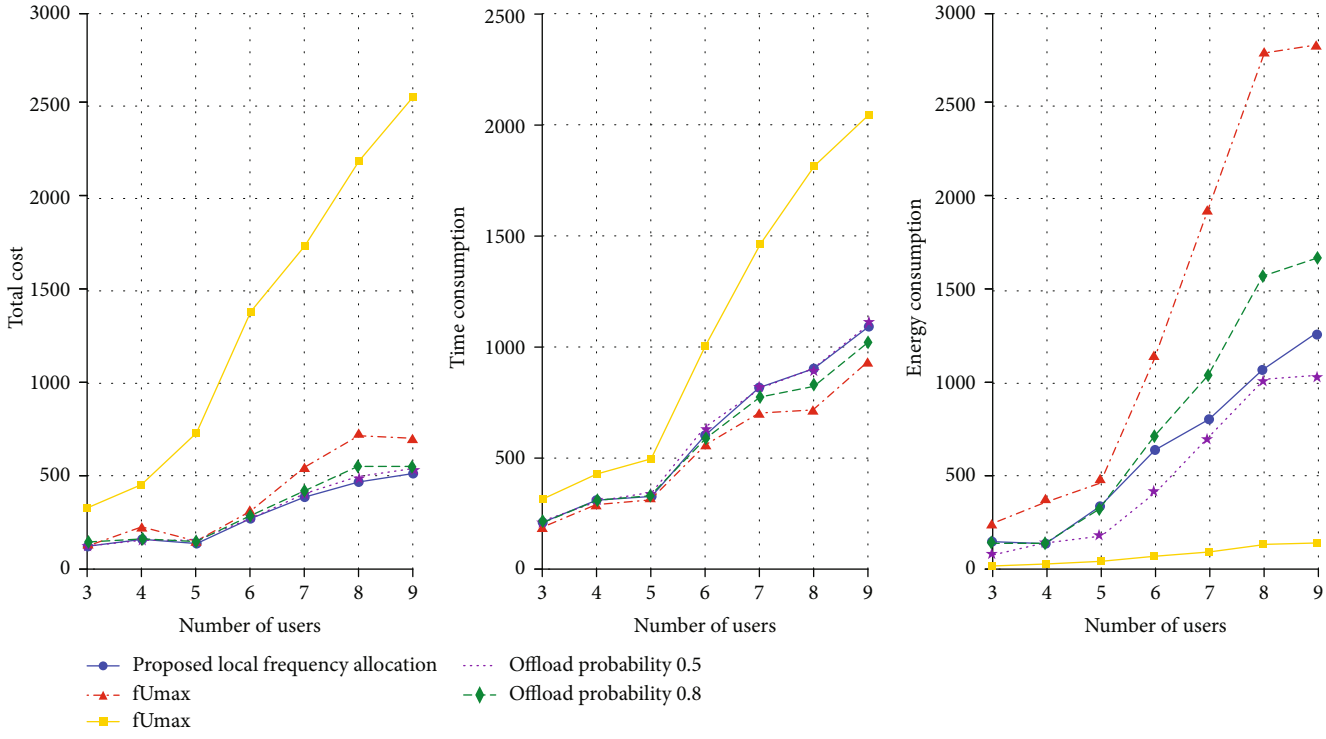

Figure 5: The impact of local computing resource allocation on total cost, time, and energy consumption under different mobile device quantities.

execution speeds, but at the same time result in higher energy consumption, as evidenced by the figure. Our algorithm achieves less energy consumption than "fUmax" and gets lower execution time than "fUmin", fully weighing the impact of time and energy on mission cost. It is noted that the average allocation "fUmid" can approach the optimal

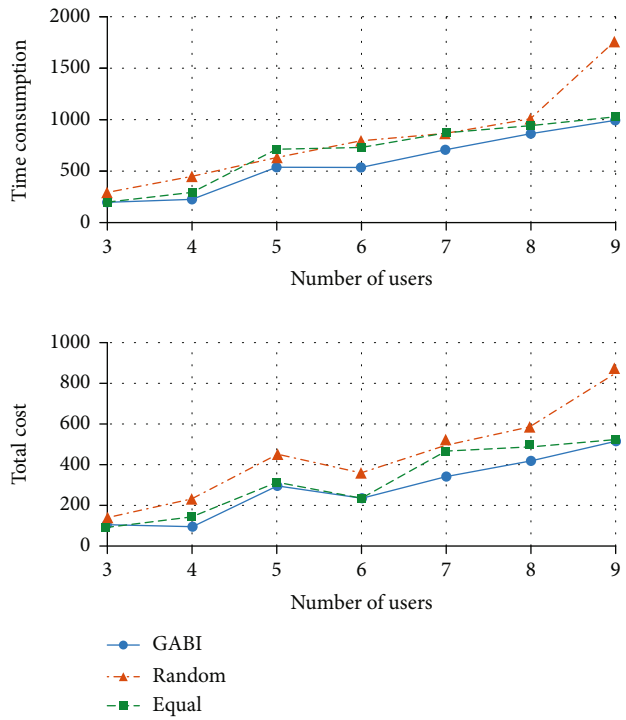

Figure 6: The impact of edge computing resource allocation on total cost and time consumption under different mobile device quantities.

solution in many cases. The reason is that in most cases, the value range of the user CPU is not extreme. The calculated local optimal resource allocation plan is a value that can be obtained in most cases and is very close to $(f_i^{\max} + f_i^{\min})/2$. Although the preset CPU frequency of the user equipment can also achieve better results, it is unstable.

In Figure 6, we compare the impact of edge server computing resource allocation schemes on time consumption and total cost under different user numbers. "Equal" and "Random", respectively, indicate that the edge server evenly allocates and randomly allocates computing resources to users. The edge server resource allocation scheme does not affect the energy consumption of edge execution, but it indirectly affects the offload decision and total cost. As shown in the figure, it is obvious that the random allocation scheme is costly and unstable, and the method of the average allocation is close to the optimal solution in many cases, but the proposed method can achieve the lowest cost and time consumption.

6. Conclusions

In this paper, we investigate the resource allocation and offloading decisions of mobile devices and edge servers in edge computing. To minimize the total cost of task execution, we define weighting factors based on deadlines to consider the impact of time and energy consumption on costs. We considered an edge computing network consisting of edge servers and users and built mathematical models. For difficult MINLP problems, we split the problem and iteratively optimize it. The result is that an approximate optimal solution is not a globally optimal solution. In future work, we will consider designing heuristic algorithms to reduce complexity and improve performance.

Data Availability

The simulation data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The work was supported by "National Natural Science Foundation of China" with No. 61842601 and 61902052, "National Key Research and Development Plan" with No. 2017YFC0821003-2, "Dalian Science and Technology Innovation Fund" with No. 2019J11CY004, and "the Fundamental Research Funds for the Central Universities" with No. DUT19RC(3)003.

References

[1] S. Wang, T. Tuor, T. Salonidis et al., "When edge meets learning: adaptive control for resource-constrained distributed machine learning," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pp. 63–71, Honolulu, HI, USA, April 2018.

[2] P. Hu, H. Ning, T. Qiu, Y. Zhang, and X. Luo, "Fog computing based face identification and resolution scheme in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1910–1920, 2017.

[3] R. Kitchin, "The real-time city? Big data and smart urbanism," *GeoJournal*, vol. 79, no. 1, pp. 1–14, 2014.

[4] K. Su, J. Li, and H. Fu, "Smart city and the applications," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, pp. 1028–1031, Ningbo, China, September 2011.

[5] R. E. Hall, B. Bowerman, J. Braverman, J. Taylor, H. Todosow, and U. Von Wimmersperg, "The vision of a smart city," Tech. Rep., Brookhaven National Lab., Upton, NY, USA, 2000.

[6] K. Nowicka, "Smart city logistics on cloud computing model," *Procedia-Social and Behavioral Sciences*, vol. 151, pp. 266–281, 2014.

[7] T. Clohessy, T. Acton, and L. Morgan, "Smart City as a Service (SCaaS): a future roadmap for E-Government smart city cloud computing initiatives," in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 836–841, London, UK, December 2014.

[8] Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, GA, USA, May 2017.

[9] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[10] T. Yang, H. Feng, C. Yang, Y. Wang, J. Dong, and M. Xia, "Multivessel computation offloading in maritime mobile edge computing network," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4063–4073, 2019.

[11] A. M. Rahmani, T. N. Gia, B. Negash et al., "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[12] A. Giordano, G. Spezzano, and A. Vinci, "Smart agents and fog computing for smart city applications," in *Smart Cities. Smart-CT 2016. Lecture Notes in Computer Science, vol 9704*, E. Alba, F. Chicano, and G. Luque, Eds., pp. 137–146, Springer, Cham, 2016.

[13] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing: a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[14] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann, "Dynamic urban surveillance video stream processing using fog computing," in *2016 IEEE Second International Conference on Multimedia Big Data (BigMM)*, pp. 105–112, Taipei, Taiwan, April 2016.

[15] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for iot systems: a computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[16] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

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

[18] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: a fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[19] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[20] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: latency optimal task assignment for resource-constrained mobile computing," *IEEE Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[21] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 352–357, Toronto, ON, Canada, April-May 2014.

[22] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, December 2016.

[23] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[24] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4642–4655, 2018.

[25] J. Zhang, X. Hu, Z. Ning et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2018.

[26] H. Zhang, J. Guo, L. Yang, X. Li, and H. Ji, "Computation offloading considering fronthaul and backhaul in small-cell networks integrated with mec," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 115–120, Atlanta, GA, USA, May 2017.

[27] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, December 2016.

[28] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4854–4866, 2019.

[29] Y. Xing and H. Seferoglu, "Predictive edge computing with hard deadlines," in *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 13–18, Washington, DC, USA, June 2018.

[30] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2019.

[31] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 2287–2295, Paris, France, April-May 2019.

[32] Y. Yu, J. Zhang, and K. B. Letaief, "Joint subcarrier and cpu time allocation for mobile edge computing," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Washington, DC, USA, December 2016.

[33] W. Sun, J. Liu, Y. Yue, and H. Zhang, "Double auction-based resource allocation for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4692–4701, 2018.