Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 18(6): 9163–9189.
DOI: 10.3934/mbe.2021452
Received: 19 July 2021
Accepted: 19 October 2021
Published: 25 October 2021

Research article

Dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in multi-access edge computing

Yanpei Liu*, Wei Huang, Liping Wang, Yunjing Zhu and Ningning Chen

School of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China

* **Correspondence:** Email: liuyanpei@zzuli.edu.cn; Tel: +8618319810322.

Abstract: The current computation offloading algorithm for the mobile cloud ignores the selection of offloading opportunities and does not consider the uninstall frequency, resource waste, and energy efficiency reduction of the user's offloading success probability. Therefore, in this study, a dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in a multi-access edge computing environment is proposed (DCO-PSOMO). According to the CPU utilization and the memory utilization rate of the mobile terminal, this method can dynamically obtain the overload time by using a strong, locally weighted regression method. After detecting the overload time, the probability of successful downloading is predicted by the mobile user's dwell time and edge computing communication range, and the offloading is either conducted immediately or delayed. A computation offloading model was established via the use of the response time and energy consumption of the mobile terminal. Additionally, the optimal computing offloading algorithm was designed via the use of a particle swarm with a mutation operator. Finally, the DCO-PSOMO algorithm was compared with the JOCAP, ECOMC and ESRLR algorithms, and the experimental results demonstrated that the DCO-PSOMO offloading method can effectively reduce the offloading cost and terminal energy consumption, and improves the success probability of offloading and the user's QoS.

Keywords: computation offloading; multi-access edge computing; offloading success rate; overload time

1. Introduction

According to the latest global data report of Hootsuite [1], there are currently 5.11 billion independent mobile users in the world. In 2019, the number of global Internet users increased by 9% as compared to 2018, and by 2023, the number of devices connected to IP networks will be more than three times that of the global population, and the global Wi-Fi hotspots will reach 628 million, an increase of four times over 2018. Additionally, with the development of technology, the number of mobile devices accessing the cloud environment is increasing; this leads to the increase of the network load in the traditional mobile cloud computing environment, and the phenomena of network congestion and response timeout occur from time to time. Therefore, in 2014, the European Telecommunications Standards Association took the lead in proposing a new technology called mobile edge computing, also known as multi-access edge computing [2], and its application scenario is shown in Figure 1.

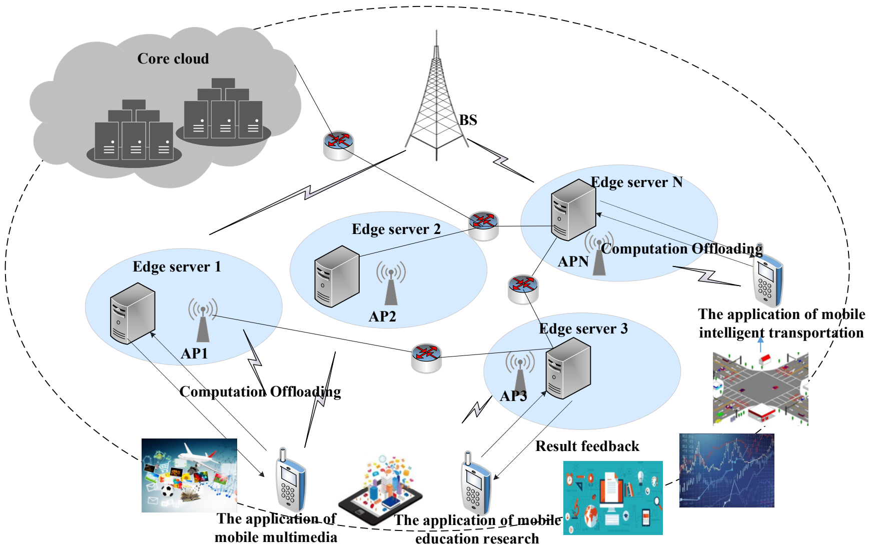

Figure1. Application scenarios of multi-access edge computing.

Compared with traditional mobile cloud computing, multi-access edge computing is characterized by distribution, real-time, and mobility [3]. Mobile cloud computing provides centralized server computing capability, while multi-access edge computing provides distributed computing capability. In multi-access edge computing, task processing and applications are concentrated in edge devices; mobile users do not need to connect to the data center to access services like mobile cloud computing, and its task processing is more dependent on the edge server and has better real-time performance. By combining its service environment and cloud computing technology on the network edge, multi-access edge computing can not only provide centralized and large amounts of storage and computing resources in the similar mobile cloud computing environment, but can also reduce the network load resulting from the large mobile terminal equipment, reduce the network delay, and ensure the real-time demand of business applications [4].

Although the computing task of multi-access edge computing is focused on the edge devices for processing, it can ensure a real-time advantage; however, because of its portability, the battery capacity and computing capacity of edge devices are low. If all tasks are placed in edge devices for processing, the energy consumption problem of mobile terminals will occur [5]. Therefore, some computing tasks need to be offloaded from the edge devices to the edge servers via a computation offloading algorithm to ensure that the energy consumption of the mobile devices can be optimized to satisfy the user response time constraints. However, some problems of multi-access edge computing remain to be solved, including determining how to select appropriate edge computing resources for computing tasks based on the information of mobile devices, task characteristics, and network conditions collected by the surrounding edge servers for network connection and computing offloading, and when and which computing tasks can be offloaded to extend the endurance time of mobile devices and improve the performance of applications.

To solve this problem, many researchers have carried out work from different perspectives. The author's team also proposed the computing offloading and resource optimization of adaptive data block size in edge environments in the early stage. The goal of the algorithm proposed in the early stage is different from that of this algorithm. The algorithm proposed in the early stage focuses on the adaptive data block size. However, most of the existing computation offloading algorithms ignore the selection of offloading opportunities in a mobile cloud environment, and fewer consider the probability that the user’s offload success rate leads to offloading too frequently, which causes problems including resource waste and energy efficiency reduction. Also, the computation offloading strategy in the mobile cloud environment cannot be directly applied to a multi-access edge computing environment. Therefore, this paper considers a mobile application as the task model of the linear topology and makes full use of the edge computing and computing resources of the mobile terminal in a dynamically changing wireless environment. Additionally, a dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in the multi-access edge computing environment is proposed. The main contributions and innovations of this paper are as follows:

1) The overload timing is dynamically obtained by a strong, locally weighted regression method based on the CPU utilization of the mobile terminal and the memory utilization rate;

2) The probability of successful offloading is predicted by the mobile user’s residence time in the multi-access edge computing communication range and determines whether to immediately offload or delay offloading. The computation offloading model is established via the use of the response time and energy consumption of mobile terminals, and the dynamic computing offloading algorithm based on particle swarm optimization with a mutation operator is proposed in a multi-access edge computing environment;

3) After establishing an experimental edge computing environment and comparing the new approach to similar algorithms, the proposed DCO-PSOMO algorithm is found to reduce the cost of offloading and the terminal energy consumption, and improves the success probability of offloading and the user’s QoS.

The remainder of this paper is organized as follows. Section 2 reviews some related work. Section 3 describes the dynamic computing offloading algorithm model based on particle swarm optimization with a mutation operator in an edge computing environment. Section 4 describes the implementation process of the proposed algorithm. Section 5 analyzes the performance of the proposed algorithm via the construction of an experimental environment with mobile edge computational properties. Section 6 presents the conclusions of this study.

2. Related work

Computation offloading strategies in mobile cloud computing have become a popular research topic in recent years [6–8]. This section introduces the development status of these strategies both domestically and abroad, and points out the problems in the existing studies.

2.1. Offloading decision for reducing delay

If a task is executed locally, the time consumed is the time required to execute the task. If the task is offloaded to the edge device, the time consumed will involve three components: the time to transmit the data to be offloaded to the edge device, the time to process the task on the edge device, and the time to receive the data returned from the edge device. In the work by Li et al. [9], the communication time of task transfer was reduced by using computational replication, which allows terminal devices to offload computing tasks to multiple edge computing nodes to repeatedly perform offloading tasks so that multiple edge computing nodes can return the collaborative computing results to terminal devices in the downlink. The proposed transmission cooperation mode can eliminate the channel interference among multiple users and significantly reduce the communication delay of multiple users and servers. Yi et al. [10] proposed a new MOTM mechanism by considering the trade-off between local computing and edge computing, wireless characteristics, and the non-cooperative game behavior of mobile users, and jointly determined the computation offloading scheme, transmission scheduling rules, and pricing rules. Khoda et al. [11] utilized the idea of load balancing between mobile devices and servers, and designed a heuristic task partition algorithm to minimize the completion time of the task. Deng et al. [12] proposed a computation offloading method for the robust design of mobile service offloading decisions that considers the dependency between component services, introduces the mobile mode and fault tolerance mechanism, and designs an offloading method based on a genetic algorithm. Mao et al. [13] proposed dynamic voltage frequency regulation and power control to optimize the calculation process and data transmission of computation offloading. Based on this model, a new algorithm of low-complexity Lyapunov optimization-based dynamic computation offloading was proposed.

2.2. Offloading decision for reducing energy consumption

The energy consumption of offloading tasks to the edge server mainly consists of two parts, namely the energy consumed by transmitting the offloading data to the edge server, and the energy consumed by receiving the data returned by the edge server. Liu et al. [14] proposed a fast heuristic algorithm that finds the solution that meets the constraint conditions, decomposes the overall reliability and delay constraints into multiple constraints of each subtask, and then finds the minimum solution in the offloading decision of each subtask to minimize the energy consumption of user equipment under the reliability and delay constraints. Li et al. [15] proposed an online batch scheduling heuristic algorithm for the dynamic offloading of independent tasks on a mobile node. The heuristic algorithm centers on the user and the system to optimize the performance of task offloading. Deng et al. [16] modeled mobile applications, such as DAG, with multiple subtasks, and aimed to minimize the terminal energy consumption under the constraint of the completion time of tasks. Their approach transforms the task offloading problem into a nonlinear 0–1 programming problem. Chen et al. [17] studied the RCS (remote cloud service) and CCS (connected ad-hoc cloudlet service) used in ad-hoc, small-cloud computing offloading and presented a new model for computation offloading. The main advantage of this model is that it does not restrict the movement of users; it also uses opportunity connections to realize the interaction between the computing nodes and multiple service nodes, and achieves energy-saving via efficient computation offloading. Kchaou et al. [18] analyzed the relationship between mobile cloud computing and big data, and proposed a mobile cloud computing offload framework for big data.

2.3. Decision of offloading considering energy consumption and time delay

Energy consumption and time delay are important factors to consider when performing the offloading task. Chen et al. [19] put forward an effective heuristic algorithm and random mapping method that can effectively reduce the total cost of users in both delay and energy consumption. Munoz et al. [20] proposed a trade-off analysis between the energy consumption and execution delay of a partial offloading decision. Several parameters, including the total amount of data to be processed, the computing power of the mobile terminal and edge device, the channel state between the mobile terminal and base station, and the energy consumption of the mobile terminal, are considered in the process of offloading. On this basis, a dynamic scheduling mechanism that allows users to make offloading decisions according to the computing queue and wireless channel status of tasks was proposed. Pandey et al. [21] proposed a time- and energy-aware mobile application offloading algorithm based on the depth-first search that can dynamically analyze the network bandwidth and energy consumption, thereby effectively reducing the energy consumption and running time. Li et al. [22] investigated the problem of the computation offloading of mobile blockchain applications, and proposed an algorithm based on the depth-first search that can dynamically analyze the network bandwidth and energy consumption, thereby effectively reducing the energy consumption and running time. You et al. [23] reused edge cloud computing server resources through a time-division mode; a jointly optimized computing and offloading model was built according to the user’s offloaded data size and transmission time, and is solved through convex optimization. Khalili et al. [24] introduced physical layer parameters into the program partitioning mechanism and proposed a joint optimization algorithm for computation offloading. Mitsis [25] proposed a resource-based pricing and user risk-aware data offloading framework for UAV-assisted multi-access edge computing systems. Apostolopoulos [26] proposed a device-centric risk-based distributed approach to determine the users’ IoT devices’ computation offloading volume in a wireless MEC environment.

The aforementioned optimization strategies for computation offloading have been widely used; however, most of the algorithms in the mobile cloud environment ignore the selection of the offloading time, and seldom consider the problem that the probability of a user’s successful offloading leads to over-frequent offloading, ultimately resulting in resource waste and energy efficiency reduction. Therefore, this paper proposes a dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in a multi-access edge computing environment. The algorithm not only considers the offloading time, but also decides whether to offload immediately or delay according to the probability of the successful offload of mobile users, thereby reducing the cost and energy consumption of offloading and ensuring the probability of successful offloading and the QoS of mobile users.

3. Dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in multi-access edge computing

There are two decisions to be made in the dynamic algorithm based on particle swarm optimization with a mutation operator, namely the choice of offloading time and the offloading optimization strategy. According to the CPU utilization and memory utilization of the mobile terminal, the dynamic adaptive hotspot detection mechanism (the dynamic detection mechanism of CPU and memory utilization) based on strong local reinforcement regression is adopted to determine the overload opportunity. After the overload opportunity is detected, the probability of successful offloading is predicted by retention time of mobile users in the range of edge computing communication. The offloading model is then established by using the response time and mobile terminal energy consumption, and the optimal offloading algorithm is finally designed by using particle swarm optimization with a mutation operator. The dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator is shown in Figure 2.

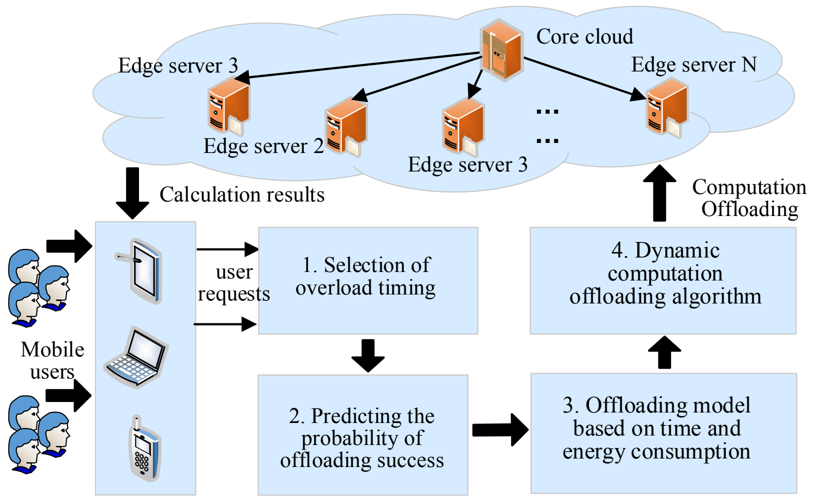
Core cloud
Edge server 3
Edge server N
Edge server 2
Edge server 3
Calculation results
Computation Offloading
user requests
1. Selection of overload timing
4. Dynamic computation offloading algorithm
Mobile users
2. Predicting the probability of offloading success
3. Offloading model based on time and energy consumption

Figure 2. Dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in a multi-access edge computing environment.

3.1. Selection of overload timing based on strong local reinforcement regression

To achieve better overload timing, a dynamic adaptive hotspot detection strategy based on strong, locally weighted regression is designed, and the utilization rate of CPU resources is taken as an example to determine the overload timing.

First, the CPU utilization of mobile terminals is monitored periodically, and the time series (x_i, y_i) of CPU utilization can be obtained, where x_i represents the time point for CPU utilization monitoring and y_i represents the CPU utilization of mobile terminals at the point x_i. The polynomial fitting of y_i is then carried out to obtain the smooth point $(x_i, \hat{y}_i)$ of the CPU utilization rate of the mobile terminal. The weighted least-squares method is used to estimate the value of fitting points, and the smooth regression curve $y_i = \hat{y}_i + \varepsilon_i$ of the CPU utilization rate of the mobile terminal is obtained, where ε_i is a random variable with a zero mean and fixed variance.

Finally, the time of CPU utilization overload can be obtained.

A set data (x_i, y_i) of the CPU utilization of the mobile terminal. For each x_i, try to select the window width as the centers. For each i, h_i is the distance of the rth nearest-neighbor point from x_j to x_i, i.e., for $j = 1, 2, ..., n$, h_i is the rth minimum value in $|x_j - x_i|$. For $k = 1, 2, ..., n$, its weight is [27].

$$w_k(x_i) = W(\frac{x_k - x_i}{h_i}) \tag{1}$$

Using a least-squares method for each (x_i, y_i) with a weight $w_k(x_i)$, the estimated value of the d-order polynomial regression coefficient $\hat{\beta}_l(x_i)$ for y_i on x_i is calculated such that:

$$\begin{aligned}&(\hat{\beta}_0(x_i), \hat{\beta}_1(x_i), \ldots, \hat{\beta}_l(x_i))\\&= \operatorname{argmin}\left[\sum_{k=1}^{n} w_k(x_i)(y_k - \beta_0 - \beta_1 x_k - \cdots - \beta_l x_k^d)\right]^2\end{aligned} \tag{2}$$

where $l = 0, 1\ldots, d$. Next, d-order polynomial fitting is performed to obtain the locally weighted regression smooth point $(x_i, \hat{y}_i)$, and the fitting value of $\hat{y}_i$ at the x_i point can be expressed as:

$$\hat{y}_i = \sum_{l=1}^{d} \hat{\beta}_l(x_i) x_i^l \qquad i = 1, 2, \ldots, n \tag{3}$$

All the observed values are fitted, and the smooth regression curve $y_i = \hat{y}_i' + \varepsilon_i$ is finally obtained. In the same manner, a smooth regression curve of the memory utilization rate of the mobile terminal is obtained, and the optimal offloading timing is dynamically obtained based on these two parameters.

3.2. Predicting the probability of offloading success based on dwell time

Because the mobile terminal has the potential to be moved at any time, the connection between the terminal and the edge server is interrupted, which may cause the failure of offloading when the mobile terminal moves out of the transmission range of the edge server. Therefore, the probability η_i that the computing task will be offloaded to the edge server i and successfully receive the calculation result is called the mobile user offloading success probability.

R represents the maximum transmission range of the edge server, and r_i denotes the distance between the mobile user and the edge server i, where the probability density function of r_i is $\pi r_i^2 / \pi R^2$. It is assumed that the user moves out of the edge computing communication range in the fastest manner, and v_i expresses the user's moving speed. The time that the mobile terminal stays within the communication range of the edge server i is then calculated as follows:

$$T_i = \int_0^R \left(\frac{\pi(r_i + dr_i)^2}{\pi R^2} - \frac{\pi r_i^2}{\pi R^2} \right) \frac{R - r_i}{v_i} dr_i = \frac{R}{3v_i} \tag{4}$$

Because the edge server can reject user service requests with certain constraints, the requested constraint is represented by χ. Therefore, the probability of success of a mobile user's job offloading to the edge server *i* can be expressed as:

$$\eta_i = \chi \cdot \frac{T_i - \tau}{T_i} = \chi \cdot (1 - \frac{3v_i\tau}{R}) \tag{5}$$

where τ represents the sum of the processing time of the task at the edge server *i* and the round-trip transmission time (RTT).

3.3. Offloading model based on time and energy consumption

Ⅰ) Communication model

For a terminal u in the set, b_u represents the decision to offload it. All terminal decision states in the set are represented by vectors $b_u = (b_1, b_2, ..., b_u)$. In the *t* phase, the upload data rate for the terminal *u* can be expressed as:

$$R_u(b) = C\log_2(+\frac{P_u H_{u,s}}{\omega_u + \sum_{m\in U\setminus\{u\}:a_m=1} P_m H_{m,s}}) \tag{6}$$

where *C* denotes the channel bandwidth, P_u denotes the transmit power allocated by the wireless access point to the user *u*, and $H_{u,s}$ is the channel gain between the terminal *u* and the base station. $\omega_u = \omega_u^0 + \omega_u^1$ denotes the background interference power, where ω_u^0 is the noise power (invalid power) and ω_u^1 is the interference power of other mobile terminals for wireless transmission. It can be seen from the communication model that if too many mobile terminal users choose to offload the computing through wireless access at the same time, it may cause serious interference, resulting in very low data transmission rate, which will have a negative impact on the performance of mobile cloud computing.

Ⅱ) Response time model

For a given offload strategy $A = \{a_i, a_2, ..., a_n\}$, $a_i \in \{0,1\}$, where $a_i = 0$ indicates that component *i* is executing at the edge server and $a_i = 1$ indicates that component *i* is executing on the mobile terminal. For a component *i*, $TK_i = (d_{i,j}, D_i)$ denotes its computational task, where $d_{i,j}$ represents the size (such as code size, parameters, etc.) of the data that component *j* needs to input to the component *i* when executing component *i*, and D_i denotes the amount of calculation required to complete this calculation (millions of instructions, CPU cycles). If both components *i* and *j* are executed locally or at the edge server, then components *i* and *j* have no data exchange, i.e., $d_{i,j} = 0$, and the transfer time between them is negligible. The time at which component *j* transmits data to component *i* can be expressed as follows:

$$t_{j,i} = (d_{i,j} \times |a_i - a_j|) / R_u(b) \tag{7}$$

Ⅲ) Mobile terminal energy consumption model

In the multi-access edge computing environment, the energy consumption of the computation offloading of the mobile terminal mainly includes the transmission energy consumption and the calculation energy consumption.

(1) Transmission energy model

P_{ta}, P_{ra}, and P_i represent the power (unit: watts) of the mobile terminal in the transmit mode, receive mode, and idle mode, respectively. In a single data execution process, the time the mobile terminal spends in these three modes is respectively divided into t_t, t_r, and t_i, and the transmission energy consumption during the execution of the data unit is:

$$E_t = P_{ta} \cdot t_t + P_{ra} \cdot t_r + P_i \cdot t_i \tag{8}$$

$$\begin{cases} t_t = \left(\sum_{(i,j)\in E, a_i - a_j = 1} d_{i,j} \right) \frac{1}{B} \\ t_r = \left(\sum_{(i,j)\in E, a_i - a_j = -1} d_{i,j} \right) \frac{1}{B} \\ t_i = T_{done} - t_t - t_r \end{cases} \tag{9}$$

where $d_{i,j}$ represents the size of the data input by component *j* to component *i* when component *i* is executed, B indicates the size of the network bandwidth between the mobile terminal and the edge server, and T_{done} represents the execution time of a data unit.

(2) Computational energy consumption model

Mobile terminals usually use microprocessors made of CMOS integrated circuits and CMOS processes. The terminals' energy consumption includes the static energy consumption and dynamic energy consumption, and the static energy consumption is almost zero and can be ignored. Therefore, only the dynamic energy consumption of the mobile terminal processor is considered in this paper, and can be expressed as:

$$P_C = f \times \alpha \times C \times V^2 \tag{10}$$

where f is the clock frequency, α is the capacitor, and V is the voltage. The power of the mobile terminal CPU in a running state can be calculated by Eq (10). It is assumed that, in the application program of the mobile terminal, the time for a data unit to execute an application component is T_{done}, for which the execution time is t_c, the idle time is t_{idle}, the calculation power of the CPU is P_c, the idle power of the CPU is P_{idle}, and the energy consumption generated by the CPU switching of the mobile terminal is ignored throughout the process. Therefore, a single data unit performs edge computing applications, and the calculated energy consumption of the mobile terminal E_C can be expressed as:

$$E_C = P_C \cdot t_c + P_{idle} \cdot t_{idle} \tag{11}$$

where $t_c = \sum_{i=1}^{n} D_i \cdot x_i / \eta p$ and $t_{idle} = T_{done} - t_c$; $x_i = 0$ indicates that the component *i* is offloaded to the edge server, and $x_i = 1$ indicates that the component *i* is on the mobile terminal. Therefore, Eq (11) can be rewritten as:

$$E_c = P_c \cdot \frac{\sum_{i=1}^{n} D_i \cdot x_i}{\eta p} + P_{idle} \cdot \left(T_{done} - \frac{\sum_{i=1}^{n} D_i \cdot x_i}{\eta p}\right) \tag{12}$$

Therefore, during computation offloading in an edge computing environment, the total energy consumption required to offload a data unit to a mobile terminal can be expressed as:

$$E = E_c + E_t \tag{13}$$

(3) Computation offload model based on time and energy consumption

The objective function of computation offloading based on the time and energy consumption is:

$$\begin{aligned} &\min_x [\textstyle\sum_{i=0}^{n} T_i, \sum_{i=0}^{n} E_i]^T \\ &s.t. \\ &a \in \{0,1\}, \quad i = 0,1,\ldots,n-1 \\ &cur_i > cur_{th} \\ &mur_i > mur_{th} \\ &\eta_i > \eta_{th} \end{aligned} \tag{14}$$

where T_i and E_i represent the time and energy consumption of component *i*, respectively, cur_{th} and mur_{th} represent the content utilization rate and memory utilization threshold of the CPU, respectively, and η_{th} represents the threshold of the probability of successful offloading.

3.4. Dynamic computation offloading algorithm based on a mutation operator particle swarm optimization

(1) Problem coding of computation offloading in a multi-access edge computing environment

The *n* components of the application are numbered from 1 to *n* in sequence, and the position vector of each particle is encoded using a one-dimensional binary encoding: $A = \{a_1, a_2, \ldots, a_n\}$, $a_i \in \{0,1\}$. When $a = 0$, the component is offloaded to the edge server, and $a = 1$ indicates that the component is executed on the mobile terminal. In other words, the position vector of the particles can represent a kind of computation offloading scheme. For example, $A = \{1,0,0,1,1,0,1,1\}$ represents 8 components in the current edge computing applications, components 1, 4, 5, 7 and 8 are executed on the mobile terminal, and components 2, 3 and 6 are offloaded to the edge server for execution. In this way, for applications with *n* components, 2^n solution spaces are needed, and the mutation operator particle swarm optimization algorithm searches for the optimal solution in these solution spaces.

In this paper, the particle velocity V in the computation offloading algorithm is also a one-dimensional vector encoding $V = \{v_1, v_2, \ldots, v_n\}$, where $v_i \in \{0,1,2\}$, $1 \le i \le n$. If $v_i = 2$, it means that the particle position is unchanged and the offloading scheme of component *i* does not change; if $v_i \neq 2$, the position vector of the particle is updated to v_i

(2) Computation offloading fitness function construction

The objective of the computation offloading algorithm proposed in this paper is to minimize the execution time and energy consumption. Two objective functions $f_1(x)$ and $f_2(x)$ are set, of which $f_1(x)$ represents the execution time of the edge computing application and $f_2(x)$ represents the terminal energy consumption. Using the power coefficient method, Eq (14) can be converted to:

$$\begin{aligned}
&\omega=\sqrt{\left(\frac{f_1(\max)-f_1(x)}{f_1(\max)-f_1(\min)}\right)\cdot\left(\frac{f_2(\max)-f_2(x)}{f_2(\max)-f_2(\min)}\right)}\\
&s.t.\\
&a\in\{0,1\},\quad i=0,1,\ldots,n-1\\
&cur_i>cur_{th}\\
&mur_i>mur_{th}\\
&\eta_i>\eta_{th}
\end{aligned}\tag{15}$$

$\omega=1$ indicates that the ideal design scheme is obtained, and $\omega=0$ indicates that this scheme is not feasible; therefore, it is necessary to solve the set of solutions with a value of ω that is closest to 1. $f_1(max)$ and $f_1(min)$ respectively represent the maximum and minimum values of $f_1(x)$, and $f_2(max)$ and $f_2(min)$ respectively represent the maximum and minimum values of $f_2(x)$. Therefore, the fitness function can be defined as:

$$f(X)=\sqrt{\left(\frac{f_1(\max)-f_1(x)}{f_1(\max)-f_1(\min)}\right)\cdot\left(\frac{f_2(\max)-f_2(x)}{f_2(\max)-f_2(\min)}\right)}\tag{16}$$

The goal of offloading in a multi-access edge computing environment is to reduce the application execution time while minimizing the terminal energy consumption. Therefore, the computation offloading can be transformed to solve the solution of $f(x)$ that is closest to 1.

(3) Design of offloading particle operator in multi-access edge computing

In multi-access edge computing environment, in order to use particle swarm optimization algorithm to solve the optimal computing offloading strategy, the particle operator needs to be redesigned.

1) Addition operator (+)

In particle swarm optimization algorithm, different positions of particles represent different offloading schemes. The driving force for particles to find the optimal offloading scheme is the update of particle velocity. The new offloading scheme A_i' is jointly determined by the original offloading scheme A_i and the particle velocity V, which can be expressed as:

$$A_i'=A_i+V\tag{17}$$

$$a_{ij}'=\begin{cases}a_{ij}, & if\ v_j=2\\ v_j, & otherwise\end{cases}\tag{18}$$

where a_{ij}' and a_{ij} represent the two offloading results of component j under the new scheme A_i' and the original scheme A_i. v_j represents the j dimensional component of speed V, which is the most direct cause of offloading strategy change.

2) Subtraction operator (−)

The flight speed of particles can be obtained by subtracting the particle position, which can be expressed as:

$$V = A_i' - A_i \tag{19}$$

The value v_j of each dimension in the velocity V can be calculated as:

$$v = \begin{cases} 2, & if\ a_{ij}' = a_{ij} \\ a_{ij}', & otherwise \end{cases} \tag{20}$$

3) Speed multiplication (×)

Let V_i be the velocity of particle i, and the value of particle velocity is updated by multiplying the velocity, which can be expressed as:

$$V_i' = c_1 \times V_i \times c_2 \tag{21}$$

c_1 and c_2 represent randomly generated natural number, $1 \le c_1 \le c_2 \le n$, n is the number of components divided by the application, the value v_{ij}' of velocity V_i' in each dimension can be expressed as:

$$v_{ij}' = \begin{cases} v_{ij}, & if\ j \in [c_1, c_2) \\ 2, & otherwise \end{cases} \tag{22}$$

4) Particle renewal equation

The new offloading scheme is determined by the original calculated offloading scheme, the historical optimal scheme and the optimal scheme of the population. The particle renewal equation can be expressed as:

$$\begin{cases} A = A + c_1 \times (P_b - A) \times c_2, & 1 \le c_1 \le c_2 \le n \\ A = A + c \times (P_g - A) \times c_4, & 1 \le c_3 \le c_4 \le n \end{cases} \tag{23}$$

where P_b represents the optimal offloading scheme of particles, P_g is the optimal offloading scheme of the population, n indicates the number of components divided by the application, c_1, c_2, c_3, c_4 represents a positive integer randomly generated from $1 \sim n$.

4. Realization of a dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in multi-access edge computing

4.1. Model DCO-PSOMO algorithm description

The dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in a multi-access edge computing environment is described as follows.

(1) *Determination of offloading time:* The dynamic adaptive hotspot detection strategy based on strong, locally weighted regression is used to solve the overload time with high CPU resource utilization or memory utilization to obtain the offloading time (lines 1–2 of algorithm 1);

(2) *Prediction of the offloading success probability:* The probability of successful mobile terminal computation offloading is calculated according to the service scope of edge computing and the residence time of users (line 4 of algorithm 1);

(3) *Optimization of computation offloading:* Particle swarm optimization with a mutation operator is used to determine the optimal offloading strategy of mobile users (lines 5–35 of algorithm 1). Lines 8-13 of algorithm 1 are the initial position and initial historical optimal position of each randomly generated particle, lines 16–22 are the particle fitness function for offloading, lines 23–27 are the updated particle position, and lines 29–31 are the execution mutation.

The pseudo-code of the algorithm is presented in algorithm 1.

Algorithm 1: Offloading algorithm DCO-PSOMO

```
Input：User's application request $A_i = \{s_1, s_2, \ldots, s_n\}$
Output：Optimal computation offloading strategy $V = \{s'_1, s'_2, \ldots, s'_k\}$
1:  $c_i \leftarrow CpuUtilizationRatio()$;
2:  $m_i \leftarrow MemoryUtilizationRatio()$;
3:  if ( $c_i > c_{threshold} \,||\, m_i > m_{threshold}$ )
4:    $\eta_i \leftarrow getOffloadSuccessProbability()$;
5：     if ( $\eta_i > \eta_{threshold}$ )
6:        Initializing $PopNum$、$MaxIter$、$Particles$
7:          $i \leftarrow 1$
8:       while( $i <= PopNum$ )
9:         Randomly generated particle i position s;
10:          Add i to the particle swarm;
11:          $pbest[i] \leftarrow s$
12:          $i \leftarrow i+1$;
13:      end while
14:          $j \leftarrow 1$;
15:      while( $j <= MaxIter$ )
16:           for(each particle k in particles)
17:              Calculate the fitness of k $fitness[k]$;
18:                   if( $fitness[k] < pbest[k]$ )
19:                      $pbest[k] \leftarrow k$;
20:                      if( $fitness[k] < gbest$ )
21:                          $gbest \leftarrow k$;
22:             end for
23:             for(each particle k in particles)
24:               Update k's position;
25:               if( $C_k < \beta$ )
26:                   Mutate k;
27:             end for
28:        if( $D < \gamma$ )
29:                for(each particle k in particles)
30:                    Mutate k;
31:                end for
32:         $j \leftarrow j+1$;
33:     end while
34:  $Stragegy \leftarrow gbest$;
35:  return $V = \{s'_1, s'_2, \ldots, s'_k\}$
```

4.2. Algorithmic complexity analysis

Time complexity analysis of the DCO-PSOMO algorithm: In the preceding description of the algorithm, lines 8–13 describe the random initial position of each particle, and, based on this position and the optimal value of the population history, the complexity is $O(M*N)$, where M represents the size of the particle swarm and N is the number of components applied to the cloud. Lines 16–22 describe the calculation of the particle fitness function, and the time complexity is $O(M*N)$. Lines 23–27 describe an update of the particle position, and the complexity is $O(M*N)$. Lines 23–27 perform a mutation operation that allows particles to enter a new area for searching, and the time complexity is $O(M*N)$. Therefore, the time complexity of the entire algorithm is $O(M*N)$.

4.3. Algorithm correctness analysis

Theorem 1: Assuming any mobile terminal P and edge cloud user application request B, a reasonable calculation offloading strategy $V=\{s_1',s_2',\ldots,s_k'\}$ and $E_{p_{off}} < E_{p_{Noff}}$ are obtained through algorithm 1.

Proof: The correctness proof of the algorithm can be divided into the following three steps:

1) Acquisition of offload opportunity: Using the strong local weighted regression method, the overload opportunity of mobile terminal (Eq (3)) can be correctly obtained, that is, the offloading opportunity can be calculated

2) Calculation of offloading success probability: The calculation formula of the residence time of the mobile terminal in a cloud is equivalent to:

$$\Leftrightarrow T_i=\frac{1}{v_iR^2}\int_0^R[(r_i+dr_i)^2-r_i^2](R-r_i)dr_i=\frac{R}{3v_i}$$

$$\Leftrightarrow T_i=\int_0^R[(r_i+dr_i)^2-r_i^2](R-r_i)dr_i=\frac{R^3}{3}$$

If $f(x)=x^2$, then $f(r_i+dr_i)-f(r_i)=(r_i+dr_i)^2-(r_i)^2$

$$\lim_{dr_i\to 0}\frac{f(r_i+dr_i)-f(r_i)}{dr_i}=f'(r_i)$$

$$\Leftrightarrow T_i=\int_0^R f'(r_i)(R-r_i)dr_i=\int_0^R(R-r_i)df(r_i)$$

$$=(R-r_i)f(r_i)|_0^R-\int_0^R f(r_i)d(R-r_i)=0+\int_0^R r_i^2dr_i=\frac{r_i^3}{3}|_0^R=\frac{R^3}{3}$$

The specific meaning of the symbol is in Section 3.2.2, and the offloading success probability is calculated according to Eq (5).

3) Optimal offloading policy: According to the calculation offloading model of time and energy consumption, the optimal calculation offloading strategy V is obtained, and condition $E_{p_{off}} < E_{p_{Noff}}$ is satisfied.

5. Experimental verification and comparison

5.1. Experimental environment and configuration

1) Experimental environment

As shown in Figure 3, a multi-access edge computing experimental environment was set up, in which the feasibility and effectiveness of the proposed dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator were verified. The experimental environment included three components, namely the mobile terminal, edge computation, and remote cloud. In the experiment, mobile terminals used Samsung phones and Lenovo laptops to interact with the edge server through Wi-Fi hotspots. To simulate a real scenario, three edge servers were set up in the experiment, each of which was built with 1 Inspur server and 6 different Lenovo desktop institutions such that the three edge servers could interact with each other in their coverage areas. The remote cloud used 11 AliCloud instances to build the environment. The server used a CentOS operating system, and OpenStack was installed to virtualize its resources. Based on these conditions, the Apache Hadoop distributed platform was built to implement the computation offloading in a multi-access edge computing environment.

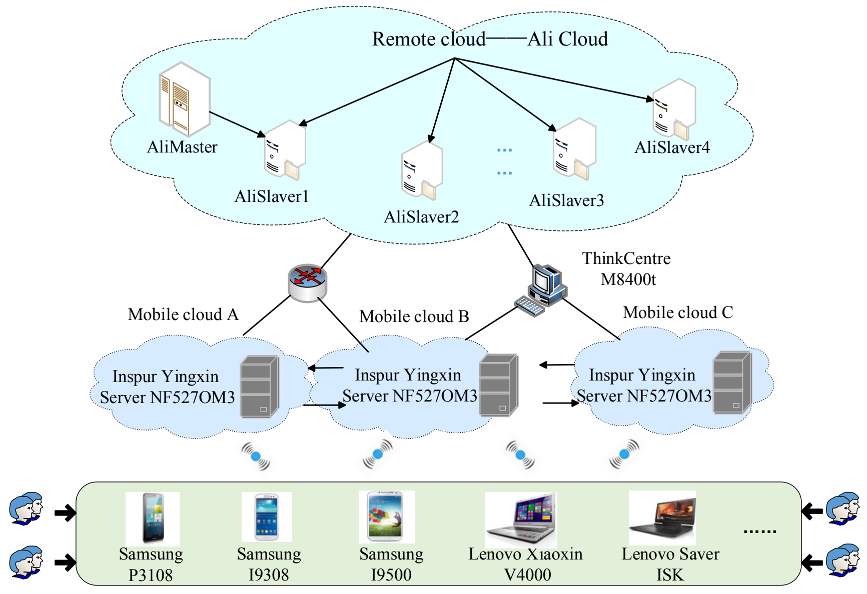

Figure 3. Experimental test environment.

2) Experimental data source

In this experiment, the power consumption of the mobile terminal in different states was determined via the powerprofile.xml provided by the Android operating system application framework, and the unit of power consumption was milliwatts. The power consumption parameters of the Samsung I9500, Samsung I9308, Samsung p3108, Lenovo Xiaoxin V4000, and Lenovo deliverer 15-ISK i5 devices are listed in Tables 1 and 2.

Table 1. Hardware power consumption of the mobile terminal in different CPU states.

Mobile terminal equipment	CPU state	
	Idle	Function
Samsung p3108	25.9	1106.3
Samsung I9308	14.8	921.3
Samsung I9500	16.7	914.9
Lenovo Xiaoxin V4000	197.4	7892.3
Lenovo deliverer I5-ISK i5	228.6	8012.5

Table 2. Hardware power consumption of the mobile terminal in different states under Wi-Fi.

Mobile terminal equipment	WIFI			
	Closed	Idle	Sending	Receive
Samsung p3108	0	3.7	555.0	549.7
Samsung I9308	0	1.1	355.2	340.7
Samsung I9500	0	0.9	360.1	358.2
Lenovo Xiaoxin V4000	0	12.3	1984.2	1970.4
Lenovo deliverer I5-ISK i5	0	12.5	2012.8	2020.6

3) Experimental parameter settings

In the algorithm verification experiment, the number of mobile users was 10, and included two Samsung i9500 mobile phones, two Samsung i9308 mobile phones, two Samsung p3108 mobile phones, two small, new Lenovo V4000 laptops, and two Lenovo Savior 15-isk i5 laptops. All users identified the same QR code, the uplink bandwidth and downlink bandwidth were the same, and the bandwidth size was [120 KB/s, 480 kb/S]. The experimental configuration parameters are listed in Table 3.

Table 3. Experimental configuration parameters.

Parameter name	Value
Inspur NF5270M3 rack server	3
Number of mobile users	14
Sam sung I9500	2
Sam sung I9308	2
Sam sung P3108	2
Lenovo Xiaoxin V4000	2
Lenovo deliverer 15-ISK i5	2

4) Performance index

The following three indicators were used to evaluate the performance of the dynamic computation algorithm based on particle swarm optimization with a mutation operator:

a) *Energy consumption:* the energy consumption required for the mobile terminal to perform the offloading task;

b) *Response time:* the time required for task offloading to edge devices and the calculation of the feedback results to mobile terminals;

c) *Offloading success rate*: the ratio of the number of tasks successfully uninstalled to the total number of tasks uninstalled.

5.2. Experimental results and analysis

To verify the feasibility and effectiveness of the DCO-PSOMO algorithm proposed in this paper, it was compared with the ECOMC [11], ESRLR [14], and JOCAP [19] algorithms.

1) Influence of wireless network bandwidth on the performance of the algorithm

This experiment mainly investigated the effects of different wireless network bandwidths on the performances of the proposed DCO-PSOMO offloading algorithm and other similar algorithms, namely the JOCAP, ECOMC, and ESRLR algorithms. In the experiment, the mobile terminal adopted a Samsung I9500, the wireless network used a Wi-Fi connection, and the terminal moved at a constant speed of 10 km/h. The experiment involved adjusting the network bandwidth from 120 kbps to 480 kbps with a step size of 60, and the experimental results are presented in Figures 4–6.

Figure 4 presents the response time of each algorithm under different network bandwidths, from which it can be seen that as the bandwidth of the wireless network increased from 120 kb/s to 480 kb/s, the response time of each computation offloading algorithm decreased. In addition, the response times for both the JOCAP and DCO-PSOMO algorithms were similar, because both algorithms consider the terminal energy consumption and response time. The response time of the DCO-PSOMO algorithm was slightly higher than that of the JOCAP algorithm, but still within the acceptable range, when the bandwidth was 240 kb/s and 420 kb/s; this was because the DCO-PSOMO algorithm predicts the success probability of offloading before offloading.

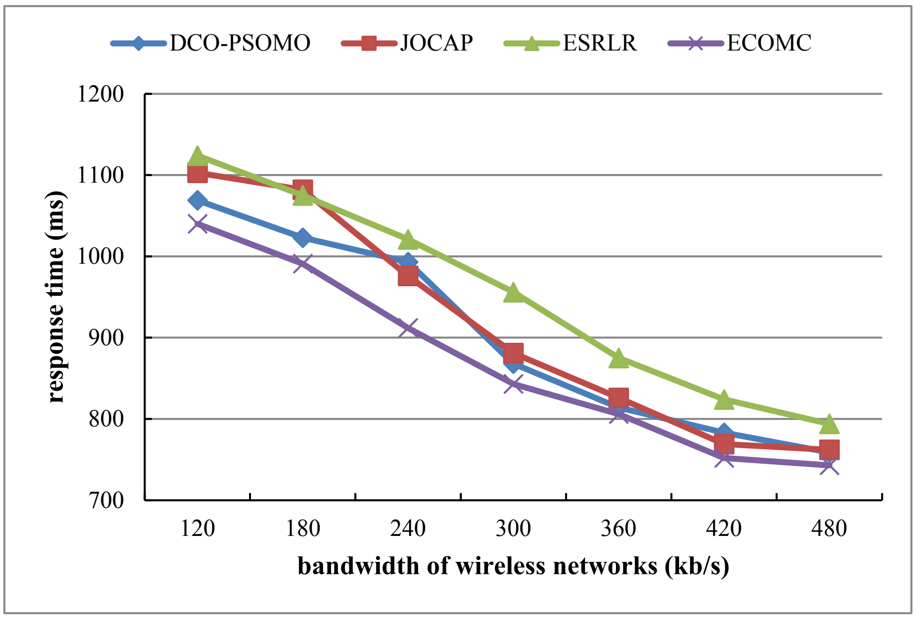

Figure 4. Comparison of response times for different network bandwidths.

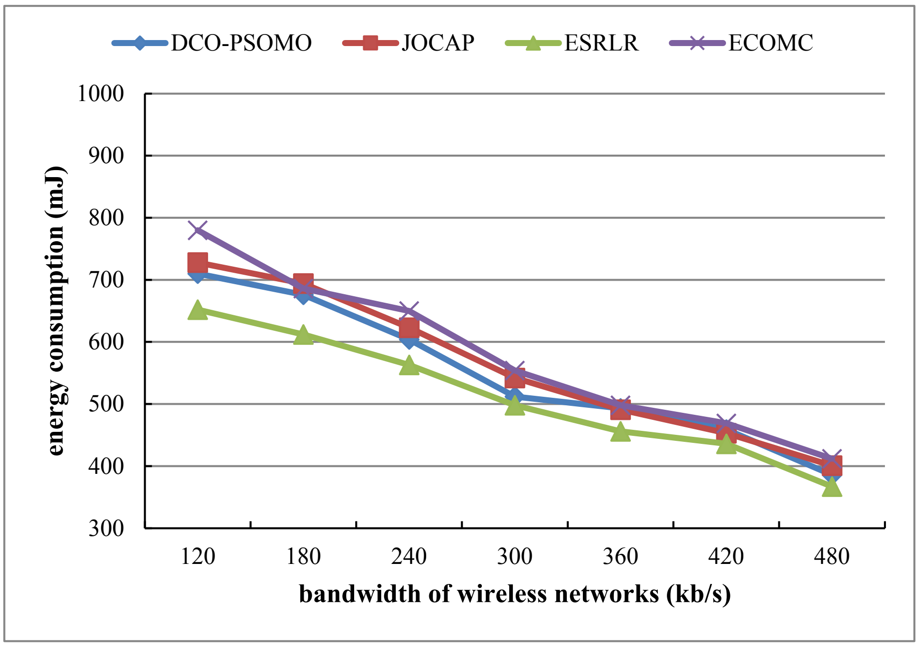

Figure 5. Comparison of terminal energy consumption for different network bandwidths.

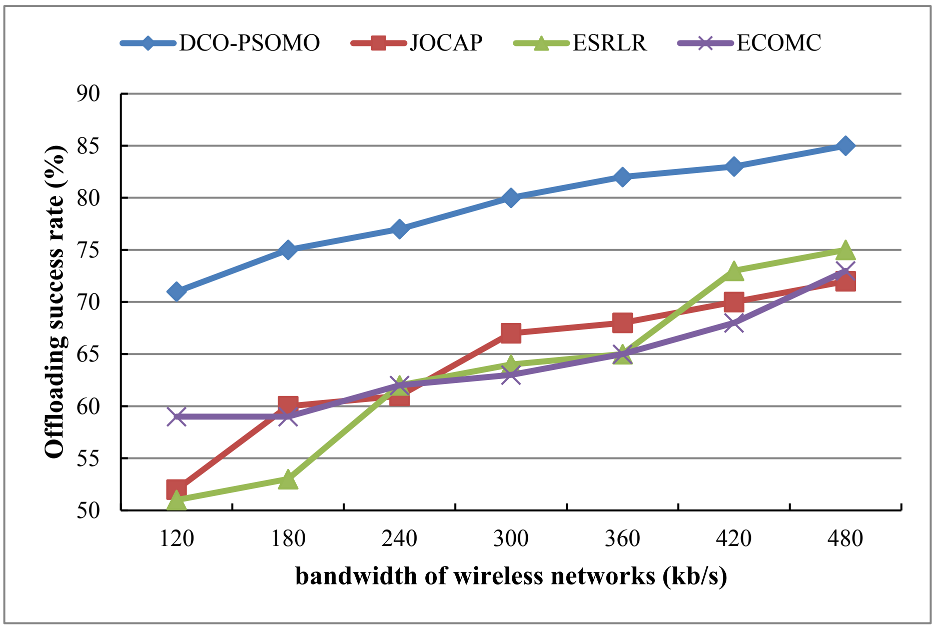

Figure 6. Comparison of offloading success rates for different network bandwidths.

Figure 5 presents the mobile terminal energy consumption of each algorithm under different network bandwidths. The ESRLR offloading strategy had the lowest energy consumption of the mobile terminals under different network bandwidths because it is a high-efficiency offloading strategy that is aimed at minimizing the energy consumption of the mobile terminal without considering the cost of offloading. The terminal energy consumption of the ECOMC offloading strategy was found to be the highest in most cases, as the ECOMC algorithm minimizes the response time and does not optimize the energy consumption of the mobile terminal. The energy consumptions of the JOCAP and DCO-PSOMO algorithms were in the middle because these two strategies take into account the energy consumption and response time of mobile terminals. When the bandwidth was 420 kb/s, the energy consumption of the DCO-PSOMO algorithm was higher than that of the JOCAP algorithm because the DCO-PSOMO algorithm predicted the timing and success rate of offloading, thereby leading to a slightly higher energy consumption.

Figure 6 presents the offloading success probability of each algorithm under different network bandwidths. The DCO-PSOMO offloading success rate was higher than those of the other three offloading strategies because DCO-PSOMO predicts the probability of offloading success according to the mobile terminal's stay time in the cloud cover to decide whether to offload immediately or delay the offloading.

2) Influence of mobile terminal type on algorithm performance

Considering the differences of the mobile terminal itself, the impacts of different mobile terminal types on the proposed DCO-PSOMO offloading algorithm and the other similar algorithms were determined through a set of experiments. The mobile terminals in the experiments included a Samsung P3108, Samsung I9308, Samsung I9500, Lenovo Xiaoxin V4000, and Lenovo saver ISK. The wireless network type adopted Wi-Fi, the network bandwidth was 240 kb/s, and the user moved at a speed of 10 km/h. The experimental results are presented in Figures 7–9.

Figure 7 presents the response times of mobile terminals running different offloading algorithms under different terminal types. The response time of the ESRLR algorithm was the highest for most cases, and that of the ECOMC algorithm was the lowest. The response time of the DCO-PSOMO algorithm was close to that of the JOCAP algorithm, as the JOCAP and DCO-PSOMO algorithms both take into account the energy consumption and response time of the mobile terminal. Additionally, when the DCO-PSOMO algorithm was run on the mobile terminal for the Samsung I9308 and the Samsung I9500 devices, its response time was higher than that of the JOCAP algorithm. This occurred because the DCO-PSOMO algorithm predicts the probability of successful offloading before offloading, but the response time was still within the acceptable range.

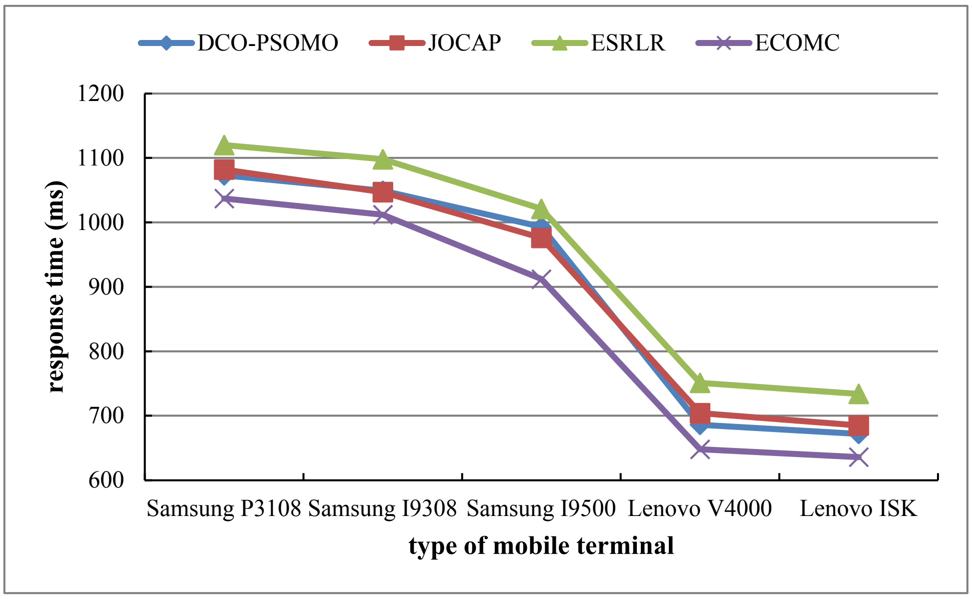

Figure 7. Comparison of response times for different mobile terminals.

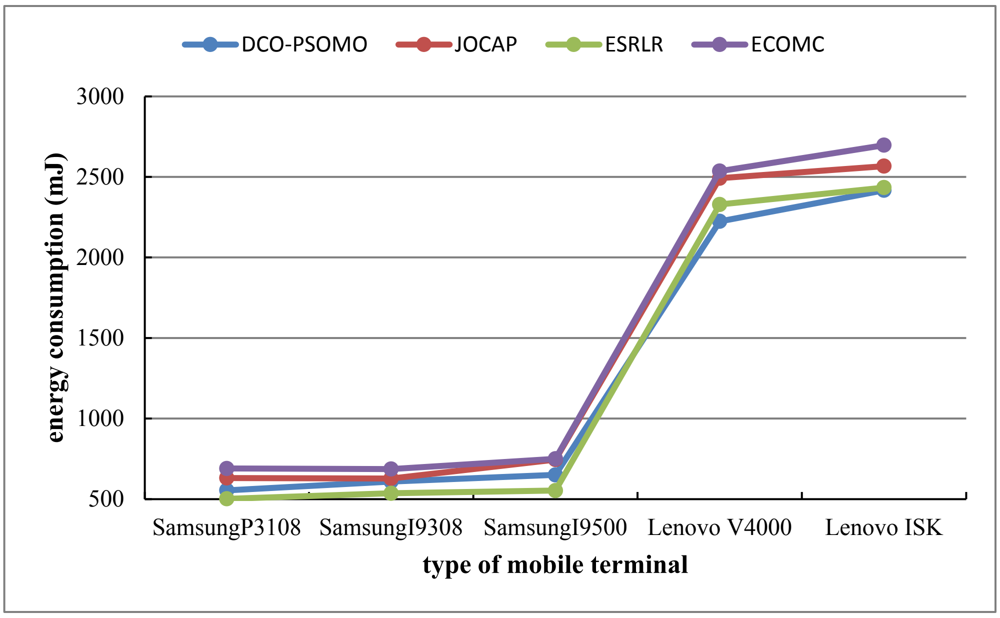

Figure 8. Comparison of terminal energy consumption for different mobile terminals.

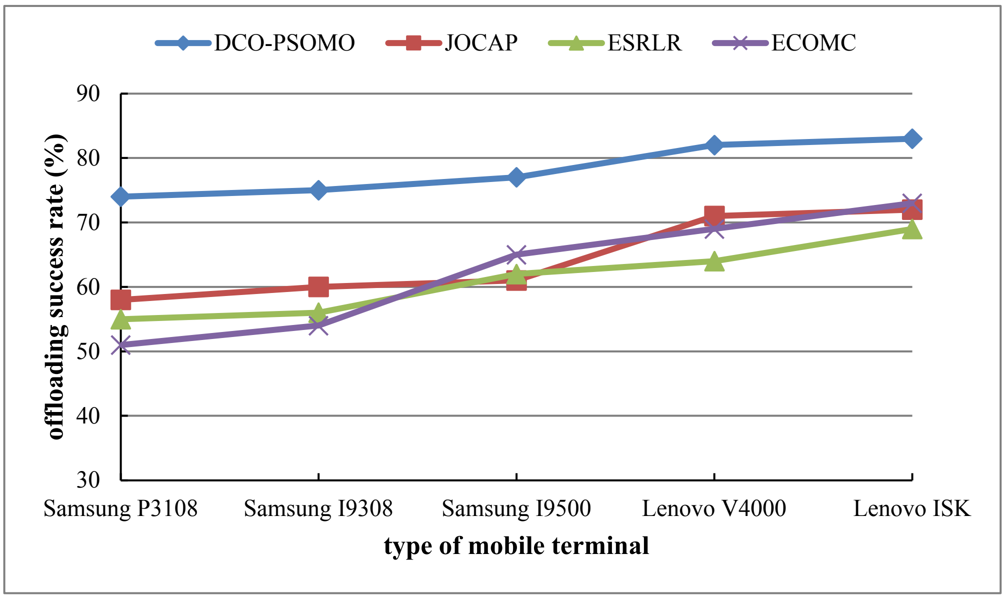

Figure 9. Comparison of offloading success rates for different mobile terminals.

Figure 8 presents the energy consumption of different types of mobile terminals running different offloading algorithms, from which it is evident that the ESRLR algorithm had the lowest mobile terminal energy consumption. This occurred because the ESRLR algorithm is an energy-efficient offloading strategy, the goal of which is to minimize the energy consumption of mobile terminals without considering the cost of offloading. The ECOMC offloading strategy was found to have the highest terminal energy consumption in different types of terminals, as ECOMC minimizes the response time and does not over-optimize the energy consumption of mobile terminals. The mobile terminal energy consumptions of the JOCAP and DCO-PSOMO algorithms were in the middle because these two strategies take into account the energy consumption and response time of mobile terminals. When the DCO-PSOMO and JOCAP were operated on a Lenovo V4000, the energy consumption of the DCO-PSOMO terminals was higher than that of the JOCAP terminals because the DCO-PSOMO algorithm predicts the offload timing and offload success rate; however, the energy response time was still within the acceptable range.

Figure 9 presents the probability of successful offloading of each algorithm running on different types of terminals, from which it is evident that the DCO-PSOMO offloading success probability was much higher than those of the other three offloading strategies. This outcome arose because the DCO-PSOMO algorithm is calculated the residence time before offloading. The residence time predicts the probability of successful offloading and determines whether to immediately offload or delay offload, which greatly increases the probability of offloading success.

3) Influence of network type on algorithm performance

A performance comparison was made between the proposed DCO-PSOMO algorithm and the JOCAP, ECOMC, and ESRLR algorithms in different network environments. In the experiment, a Samsung i9500 was used as the mobile terminal, and 2G, 3G, 4G and Wi-Fi were used as the types of wireless networks. The user moved at a speed of 10 km/h. The effects of different network bandwidths on the response time, terminal energy consumption, and probability of successful offloading of the algorithms were tested, and the results are respectively presented in Figures 10–12.

Figure 10 presents the response time of each algorithm in different wireless networks environments, from which it is evident that the response time of the mobile terminal in the 2G wireless network environment was the longest, and that of mobile terminal in the Wi-Fi wireless network environment was the shortest. This is because the wireless network bandwidth of the 2G network was the smallest, while that of Wi-Fi was the largest. With the increases of the wireless network bandwidth and the data transmission per unit time, the offloading time and the return time of the calculation results were reduced, so the response time during computation offloading was also reduced. It can also be seen that the response time of the high-energy and efficient computation offloading ESRLR algorithm was the highest in most cases, and that of the ECOMC algorithm was the lowest. The response time of the JOCAP algorithm based on energy consumption and time was close to that of the DCO-PSOMO algorithm.

Figure 11 presents the energy consumption of each algorithm in different wireless network environments, from which it is clear that the mobile terminal with the 2G wireless network type calculated the maximum terminal energy consumption for offloading, and the mobile terminal with the Wi-Fi network type calculated the minimum terminal energy consumption for offloading. This is because the 2G network had the smallest network bandwidth, while the Wi-Fi network had the largest bandwidth. The energy consumption of data transmission during computation offloading gradually increased with the decrease of the wireless network bandwidth. In addition, the mobile terminal energy consumption of the ESRLR offloading strategy was the lowest for different network bandwidths, that of the ECOMC offloading strategy was the highest in most cases, and those of the JOCAP and DCO-PSOMO algorithms were in the middle.

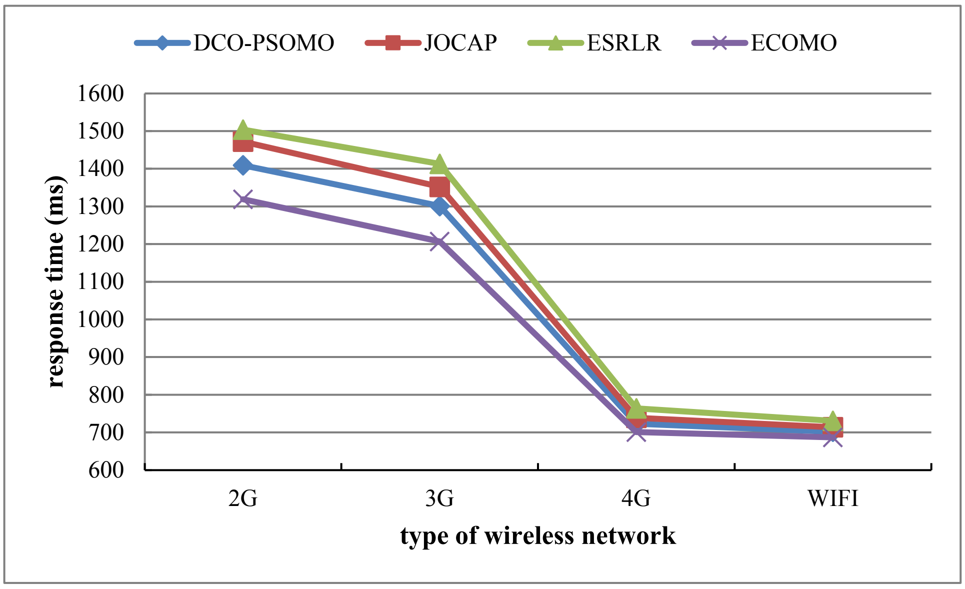

Figure 10. Comparison of response times in different wireless network environments.

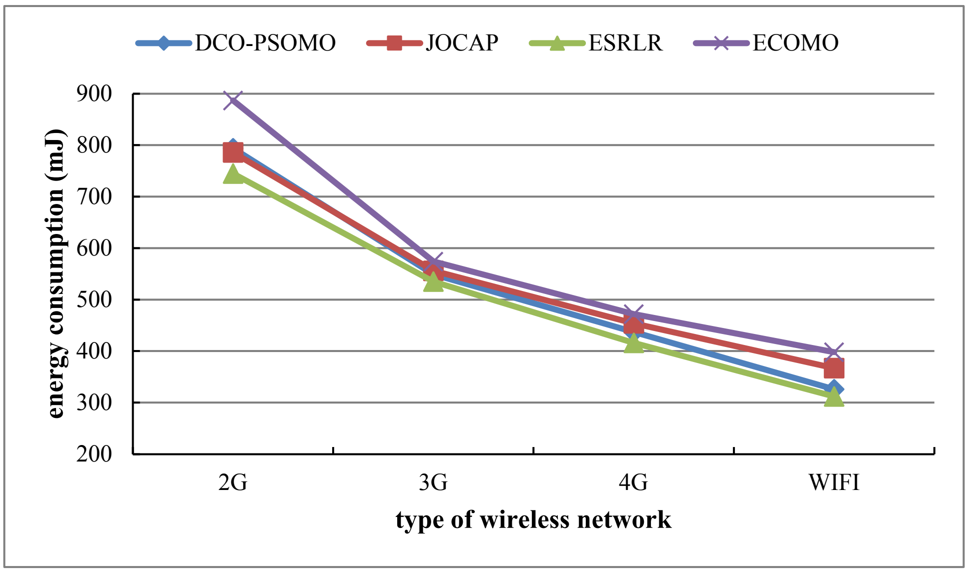

Figure 11. Comparison of energy consumption of mobile terminals in different wireless network environments.

Figure 12 presents the success probability of each algorithm in different types of wireless network environments. The mobile terminal with the 2G wireless network type had the lowest probability of successful computation offloading, and the mobile terminal with the Wi-Fi network type had the highest probability of successful computation offloading. This is because the 2G network had the smallest network bandwidth while Wi-Fi had the largest network bandwidth. The increase of the wireless network bandwidth will reduce the response time, and the terminal will move at a constant speed, thereby increasing the possibility that the mobile terminal will receive the feedback calculation result from the edge device, ultimately leading to the increase of the offloading success rate. It is also clear that the DCO-PSOMO algorithm had a much higher probability of successful uninstallation than did the uninstallation strategies of the other algorithms.

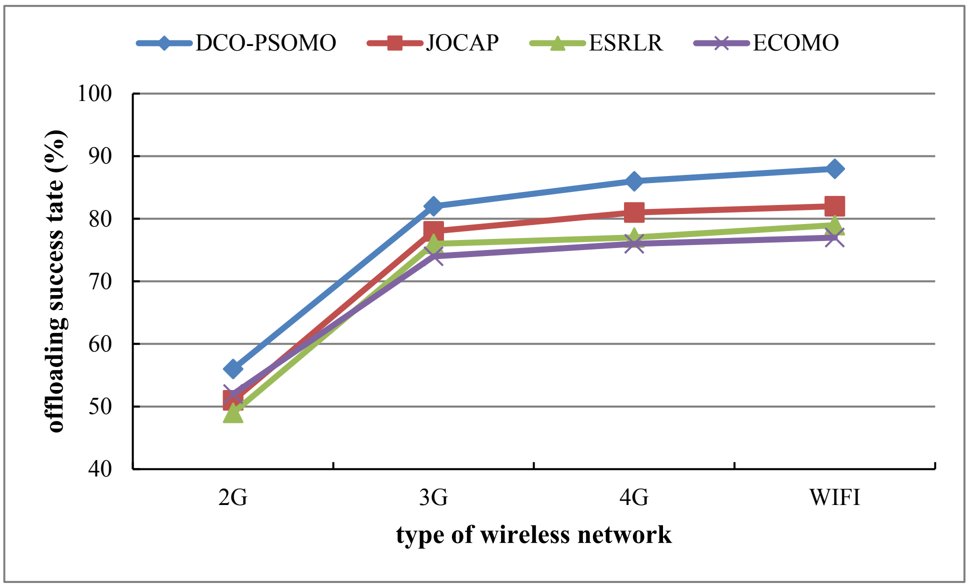

Figure 12. Comparison of the probability of successful offloading in different wireless network environments.

4) Influence of terminal moving speed on the performance of the algorithm

A performance comparison between the proposed DCO-PSOMO algorithm and the JOCAP, ECOMC, and ESRLR algorithms at different terminal moving speeds was subsequently made. In the experiment, the mobile terminal adopted a Samsung I9500, the wireless network used a Wi-Fi connection, and the network bandwidth was 240 kb/s. The moving speed of the terminal was adjusted from 5 km/h to 40 km/h in increments of 5 km/h. The experimental results are presented in Figures 13–15.

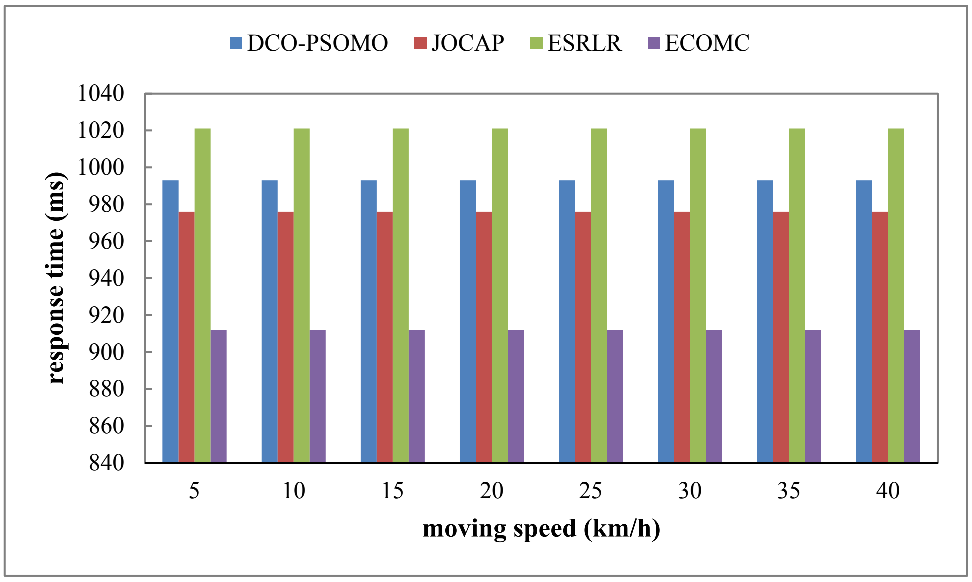

Figure 13. Comparison of response times at different moving speeds.

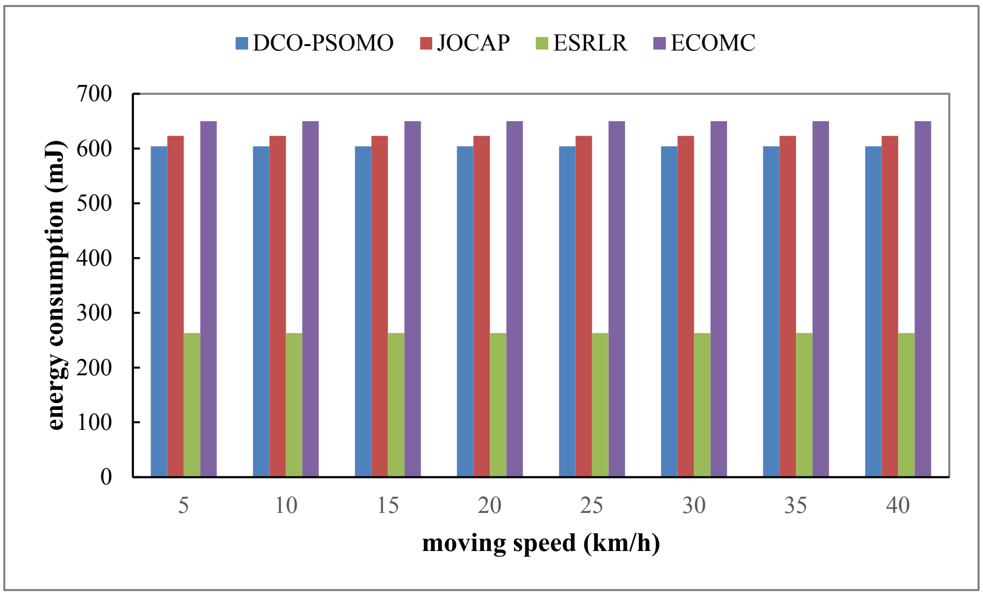

Figure 14. Comparison of terminal energy consumption at different moving speeds.

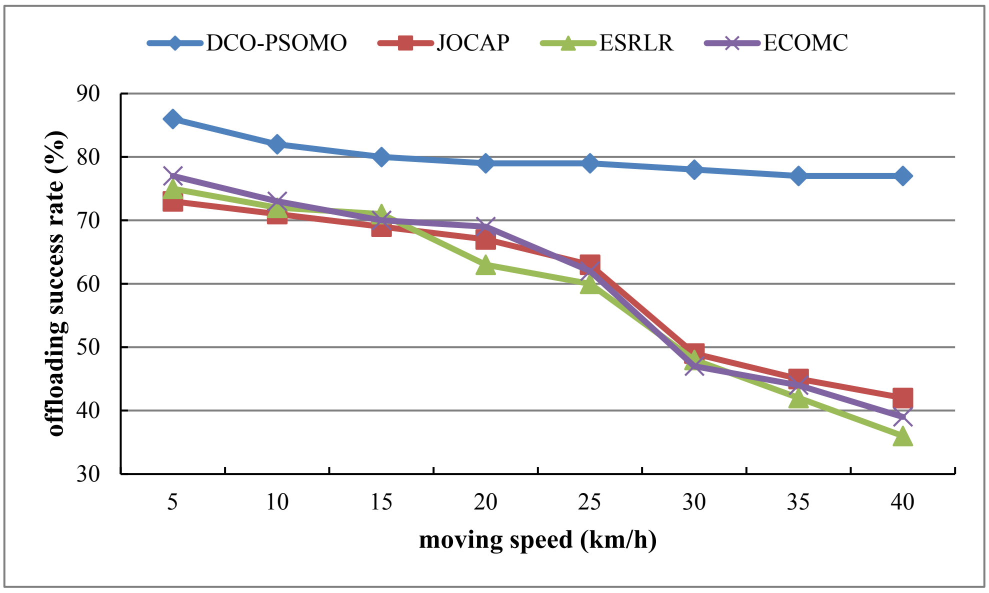

Figure 15. Comparison of offloading success rates at different moving speeds.

It is evident from Figures 13 and 14 that the moving speed of the terminal had no effect on the response time and terminal energy consumption, as the network bandwidth was not affected by the terminal movement speed, and the data transmission time and component calculation did not change. Figure 15 reveals that the offloading success rates of the JOCAP, ESRLR, and ECOMC offloading strategies decreased very quickly as the terminal moving speed increased, and the probability of the successful offloading of DCO-PSOMO was much higher than those of the other three strategies. This result occurred because the DCO-PSOMO algorithm predicts the offloading success probability of a mobile terminal based on its dwell time in the edge of the computing coverage area to determine whether to immediately offload or delay offloading, which reduces the influence of the movement speed of the terminal on the probability of offloading success and increases the probability of successful offloading.

5.3. Experiment summary

The following conclusions can be drawn from the analysis of the four groups of experiments:

1) The network bandwidth and network type have impacts on the offloading algorithm; the larger the wireless network bandwidth, the shorter the response time, and the higher the probability of successful offloading;

2) The mobile speed of the terminal has a certain influence on the success probability of offloading. The faster the mobile speed of the terminal, the lower the success probability of offloading, and vice versa;

3) The DCO-PSOMO offloading algorithm proposed in this paper considers both the time of offloading and the probability of successful offloading, which makes the algorithm superior to other similar offloading algorithms in terms of the response time and probability of successful offloading.

6. Conclusions and future work

The decision of whether to offload immediately or delay is determined according to the success probability of the offloading of mobile users. On this basis, a dynamic algorithm based on particle swarm optimization with a mutation operator in a multi-access edge computing environment was proposed in this work. The algorithm can dynamically determine the overload time via a strong, locally weighted regression method, the computation offloading model is established by using the response time and the energy consumption of the mobile terminal, and the optimal algorithm is designed based on particle swarm optimization with a mutation operator. The DCO-PSOMO algorithm was compared with the JOCAP, ECOMC, and ESRLR algorithms, and the experimental results demonstrated that the DCO-PSOMO offloading method can effectively reduce the offloading cost and terminal energy consumption, as well as improve the offloading success probability and the user's QoS.

In future research, more attention will be paid to the dynamic offloading scheme of multi-access edge computing in a 5G environment, and a more advanced offloading decision algorithm will be proposed. For example, by using various prediction techniques of user mobility and channel quality during the process of offloading, the cost of offloading in different conditions can be better estimated.

Acknowledgement

The authors thank the editor and the anonymous reviewers for their helpful comments and suggestions. The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61802353), Natural Science Foundation of Henan Province (No. 202300410505), The project of Science and Technology of Henan Province (NO. 192102210270) and Henan Provincial Department of Science and Technology (NO. 192102210270), and Dr Fund of Zhengzhou University of Light Industry.

Conflict of interest

The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

References

1. VZKOO, *Hootsuite—global digital report 2021*, 2021. Available from: https://www.vzkoo.com/doc/31188.html.
2. D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, F. Giust, Mobile–edge computing architecture: the role of MEC in the internet of things, *IEEE Consum. Electron. Mag.*, **5** (2016), 84–91.
3. J. Yan, S. Bi, Y. J. Zhang, M. Tao, Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency, *IEEE Trans. Wireless Commun.*, **19** (2019), 235–250.
4. T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, *IEEE Commun. Surv. Tutorials*, **19** (2017), 1657–1681.
5. Y. H. Kao, B. Krishnamachari, M. R. Ra, F. Bai, Hermes: Latency optimal task assignment for resource–constrained mobile computing, *IEEE Trans. Mobile Comput.*, **16** (2017), 3056–3069.
6. N. Cheng, F. Lyu, W. Quan, C. Zhou, H. He, W. Shi, et al., Space/aerial–assisted computing offloading for IoT applications: A learning–based approach, *IEEE J. Sel. Areas Commun.*, **37** (2019), 1117–1129.
7. M. G. R. Alam, M. M. Hassan, M. Z. I. Uddin, A. Almogren, G. Fortino, Autonomic computation offloading in mobile edge for IoT applications, *Future Gener. Comput. Syst.*, **90** (2009), 149–157.
8. S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, K. K. Leung, Dynamic service placement for mobile micro–clouds with predicted future costs, *IEEE Trans. Parallel Distrib. Syst.*, **28** (2016), 1002–1016.
9. K. Li, M. Tao, Z. Chen, A computation–communication tradeoff study for mobile edge computing networks, in *2019 IEEE International Symposium on Information Theory*, (2019), 2639–2643.
10. C. Yi, J. Cai, Z. Su, A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications, *IEEE Trans. Mobile Comput.*, **19** (2019), 29–43.
11. M. E. Khoda, M. A. Razzaque, A. Almogren, M. M. Hassan, A. Alamri, A. Alelaiwi, Efficient computation offloading decision in mobile cloud computing over 5G network, *Mobile Netw. Appl.*, **21** (2016),777–792.
12. S. Deng, L. Huang, J. Taheri, A. Y. Zomaya, Computation offloading for service workflow in mobile cloud computing, *IEEE Trans. Parallel Distrib. Syst.*, **26** (2014), 3317–3329.
13. Y. Mao, J. Zhang, K. B. Letaief, Dynamic computation offloading for mobile-edge computing with energy harvesting devices, *IEEE J. Sel. Areas Commun.*, **34** (2016), 3590–3605.

14. H. Liu, L. Cao, T. Pei, Q. Deng, J. Zhu, A fast algorithm for energy-saving offloading with reliability and latency requirements in multi-access edge computing, *IEEE Access*, **8** (2019), 151–161.
15. B. Li, Y. Pei, H. Wu, B. Shen, Heuristics to allocate high–performance cloudlets for computation offloading in mobile ad hoc clouds, *J. Supercomput.*, **71** (2015), 3009–3036.
16. M. Deng, H. Tian, B. Fan, Fine-granularity based application offloading policy in cloud–enhanced small cell networks, in *2016 IEEE International Conference on Communications Workshops (ICC)*, (2016), 638–643.
17. M. Chen, Y. Hao, Y. Li, C. F. Lai, D. Wu, On the computation offloading at ad hoc cloudlet: architecture and service modes, *IEEE Commun. Mag.*, **53** (2015), 18–24.
18. H. Kchaou, Z. Kechaou, A. M. Alimi, Towards an offloading framework based on big data analytics in mobile cloud computing environments, *Proc. Comput. Sci.*, **53** (2015), 292–297.
19. M. H. Chen, M. Dong, B. Liang, Joint offloading decision and resource allocation for mobile cloud with computing access point, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (2016), 3516–3520.
20. O. Munoz, A. Pascual-Iserte, J. Vidal, Optimization of radio and computational resources for energy efficiency in latency–constrained application offloading, *IEEE Trans. Veh. Technol.*, **64** (2014), 4738–4755.
21. V. Pandey, S. Singh, S. Tapaswi, Energy and time efficient algorithm for cloud offloading using dynamic profiling, *Wireless Pers. Commun.*, **80** (2015), 1687–1701.
22. Y. Li, J. Wu, L. Chen, POEM+: Pricing longer for mobile blockchain computation offloading with edge computing, in *IEEE International Conference on High Performance Computing and Communications*, (2019), 162–167.
23. C. You, K. Huang, H. Chae, B. H. Kim, Energy-efficient resource allocation for mobile–edge computation offloading, *IEEE Trans. Wireless Commun.*, **16** (2017), 1397–1411.
24. S. Khalili, O. Simeone, Inter-layer per-mobile optimization of cloud mobile computing: a message-passing approach, *IEEE Trans. Emerging Telecommun. Technol.*, **27** (2016):814–827.
25. G. Mitsis, E. E. Tsiropoulou, S. Papavassiliou, Data offloading in UAV-assisted multi-access edge computing systems: A resource-based pricing and user risk-awareness approach, *Sensors*, **20** (2020), 1–21.
26. P. A. Apostolopoulos, E. E. Tsiropoulou, S. Papavassiliou, Cognitive data offloading in mobile edge computing for internet of things, *IEEE Access*, **8** (2020), 55736–55749.
27. A. Nurunnabi, G. West, D. Belton, Robust locally weighted regression techniques for ground surface points filtering in mobile laser scanning three–dimensional point cloud data, *IEEE Trans. Geosci. Remote Sens.*, **54** (2015), 2181–2193.