A Time-Driven Cloudlet Placement Strategy for Workflow Applications in Wireless Metropolitan Area Networks

Jianshan Zhang
Fuzhou University
https://orcid.org/0000-0002-3006-1328

Ming Li
Fuzhou University

Xianghan Zheng (✉ xianghan.zheng@fzu.edu.cn)
Fuzhou University

Ching-Hsien Hsu
Asia University

Research Article

Keywords: mobile edge computing, wireless metropolitan area network, workflow application, cloudlet placement, computation offloading

DOI: https://doi.org/10.21203/rs.3.rs-693704/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
A Time-Driven Cloudlet Placement Strategy for Workflow Applications in Wireless Metropolitan Area Networks

Jianshan Zhang\textsuperscript{1,2}, Ming Li\textsuperscript{1,2}, Xianghan Zheng\textsuperscript{1,2*} and Ching-Hsien Hsu\textsuperscript{3}

Abstract
With the rapid development of mobile technology, mobile applications have increasing requirements for computational resources, and mobile devices can no longer meet these requirements. Mobile edge computing (MEC) has emerged in this context and has brought innovation into the working mode of traditional cloud computing. By provisioning edge server placement, the computing power of the cloud centre is distributed to the edge of the network. The abundant computational resources of edge servers compensate for the lack of mobile devices and shorten the communication delay between servers and users. Constituting a specific forms of edge servers, cloudlets have been widely studied within academia and industry in recent years. However, existing studies have mainly focused on computation offloading for general computing tasks under fixed cloudlet placement positions. They ignored the impact on computation offloading results from cloudlet placement positions and data dependencies among mobile application components. In this paper, we study the cloudlet placement problem based on Workflow Applications (WAs) in Wireless Metropolitan Area Networks (WMANs). We devise a cloudlet placement strategy based on a Particle swarm optimization algorithm using Genetic algorithm operators with the Encoding Library updating mode (PGEL), which enables the cloudlet to be placed in appropriate positions. The simulation results show that the proposed strategy can obtain a near-optimal cloudlets placement scheme, and compared with other classic algorithms, this algorithm can reduce the execution time of WAs by 15.04%-44.99%.

Keywords: mobile edge computing; wireless metropolitan area network; workflow application; cloudlet placement; computation offloading

1 Introduction
With the emergence of more advanced mobile hardware and a new generation of mobile communication technology that integrates wireless communication and modern Internet technologies, many delay-sensitive and computation-intensive mobile applications, such as voice recognition applications, online games, webcasts, and augmented reality applications, are present on mobile devices. These applications pose new challenges to mobile devices. Although mobile hardware-related technologies have made great progress, due to the restrictions of mobile devices in size, weight, battery life, and heat dissipation, the gap between the capability of limited computational resources and computation-intensive application requirements is gradually increasing [1]. Mobile cloud computing (MCC), a computing paradigm, integrates cloud computing and mobile computing to enhance the computation performance of mobile devices [2]. By offloading a portion or all of the components of mobile applications to a cloud server, the performance of mobile applications can be greatly improved [3]. However, modern mobile applications generally have high requirements for instantaneity. The traditional central cloud is usually remotely located and distant from the users. For most modern mobile applications, offloading to the central cloud may not be an optimal solution. To overcome these disadvantages, mobile edge computing (MEC) is an efficient solution that can better solve the problem of insufficient resources of mobile devices. As a new computing paradigm proposed after MCC, MEC has brought innovation to the working mode of traditional cloud computing. When edge servers are placed at the edge of the network, the computing power of the central cloud is distributed to the
network edge, and the rich computational resources of the edge servers compensate for the lack of mobile devices. In particular, constituting a specific form of edge server, cloudlets have been widely studied by academia and industry in recent years. The problem of cloudlet placement is the key to the efficient utilization of edge resources in the network [4-6]. A cloudlet is composed of computer clusters with many computational and storage resources; it resides at an access point (AP) in the network to provide edge computing services for mobile devices. Mobile applications can enter the network through the AP and ultimately be offloaded to the cloudlets in the network for processing. Compared with that for the traditional central cloud, the spatial distance between cloudlets and mobile devices is closer, which increases the network connection stability and speed between them while reducing the time cost of mobile devices expended on obtaining additional computational resources.

Existing research work has mainly focused on the problem of computation offloading of general tasks under conditions of fixed cloudlet placement positions [7-10]. In such studies, cloudlet placement positions are directly given as a part of the network environment, ignoring the impact on computation offloading results from cloudlet placement positions and the data dependence among mobile application components. In fact, the cloudlet placement position in the network is critical to the execution time of computing tasks offloaded from mobile devices and the resource utilization of cloudlets, especially in large-scale wireless metropolitan area networks (WMANs) with thousands of APs. A WMAN provides network services for mobile users in large-scale metropolitan areas. These networks are usually public infrastructure and operated by the local government [11], which brings the following benefits: 1) the metropolitan area covered by the WMAN has a high population density, and cloudlets can provide computing services for more mobile devices to improve the utilization of cloudlets; 2) in view of the large scale of WMANs, service providers can take advantage of economies of scale to reduce the use price of cloudlets, making cloudlet services more easily accepted by the public.

However, the correct placement of cloudlets in WMANs is a major challenge. In addition, the correct cloudlet placement position is highly important to improve the user experience of mobile applications. Failure to properly place cloudlets will result in computation offloading that cannot effectively reduce the execution time of mobile applications, which will negatively affect the user experience of mobile applications. This paper focuses on solving the \( K (\geq 1) \) cloudlets placement problem in large-scale WMANs and fully considers the impact of data transmission time and execution time of sub-tasks included in workflow applications (WAs). The object is to find a cloudlet placement scheme that minimizes the execution time of WAs. The main contributions of this paper are as follows:

- We propose an abstract model of a cloudlet placement system in the WMAN. In this system model, mobile applications are WAs with complex internal dependencies. With the object of minimizing the execution time of WAs, a detailed mathematical analysis and modelling of the \( K \) cloudlet placement problem in a WMAN is performed.
- The particle encoding and location update mode processes in the traditional particle swarm optimization (PSO) algorithm are improved, and the cloudlet placement strategy based on PSO algorithm using genetic algorithm (GA) operators with the encoding library updating mode (PGEL) is proposed. By introducing the update operator of the GA and the encoding library update mode in the particle update process, this strategy solves the problems of easily falling into local optima and redundant operations during the particle update process that are encountered in the traditional PSO algorithm.
- To verify the effectiveness and superiority of the proposed strategy, we conduct sufficient simulation experiments. The experimental results show that the proposed strategy not only obtains a nearly optimal cloudlet placement scheme in typical WMANs but also maintains excellent performance when the WMAN changes. This approach also has a major advantage when evaluated against several classic algorithms.

The rest of the paper is organized as follows. In Section 2, we discuss works related to this paper. In Section 3, we describe the system models and define the problem to be studied. In Section 4, we introduce the cloudlet placement strategy based on PGEL. In Section 5, we present our experimental setup, evaluation, and analysis of the results. Finally, Section 6 concludes the paper.

### 2 Related Work

Using remote servers to expand the computational resources of mobile devices is a novel and important research direction. Mobile devices implement computational offloading by encapsulating computing tasks in a virtual machine and then offloading them to remote servers for execution [12]. Existing task offloading frameworks and algorithms mainly focus on offloading computing tasks to the cloud server [7, 13, 14]. However, owing to several advantages of cloudlets, they have replaced cloud servers and have become the new offloading destinations in many scenarios [8, 9]. Compared with a cloud server, which is far away from the
user, a cloudlet with certain computational resources placed on the AP in the network has a smaller spatial
distance from the user, which means that the transmission delay involved in the process of task offloading
to the cloudlet is greatly reduced, thereby effectively improving the user experience of delay-sensitive
applications. For example, the design of the Odessa [15] system can support interactive mobile applications. By
offloading some components of the application to the cloudlet instead of the central cloud, it reduces the
data transmission time and the execution time of computing tasks and enables the application to meet the
stringent requirements of response time. By considering the quality of service (QoS) requirements of mobile
users and aiming to maximize the revenue of service providers, Hoang et al. [16] proposed a linear program-
ming solution to the task offloading problem, which offloads computing tasks to the appropriate cloudlet.
Chen et al. [7] fully considered the task dependency and the need for failure recovery after offloading,
described the task offloading problem in a cloudlet system as the minimization of execution cost under the
constraint of a task completion deadline, and proposed a robust computation offloading strategy with
failure recovery characteristics. Chen et al. [17] proposed an innovative framework that uses a distributed
decision-making manner and effectively achieves cooperative load balancing among multiple edges based on
reinforcement learning in Industrial IoT environments. Similarly, Guan et al. [18] proposed a novel mobile
aware offloading model in an MEC environment, which considers the execution efficiency of offloading, task
processing time constraints and energy efficiency and solves the internal offloading scheduling problem and
load aware heterogeneous resource allocation problem in cloudlets.

Although research on using cloudlets as computational offloading destinations has become a popular
research topic, the correct cloudlet placement position in the network has been ignored. Some existing
studies assume that cloudlets are used in small network environments such as campuses, companies, and
factory parks [19–21]. In such a network environment, regardless of where cloudlets are placed, the
spatial distance between them and mobile devices is extremely limited, which makes the impact of cloudlet
placement position on the use of cloudlets negligible. However, in a WMAN, which is composed of tens of
thousands of APs, the placement of cloudlets becomes extremely important and complex. To minimize the
execution time of mobile applications, it is imperative to study the cloudlet placement position in the
WMAN. Several related studies exist on the placement of cloudlets in large-scale networks. Mike et al. [22]
proposed a multi-user WMAN cloudlet placement model based on a queuing theory model. This model considers
the placement of cloudlets in densely populated areas of a WMAN with mobile users and allocates mobile
users to balance the load of each cloudlets after the cloudlet placement is completed, thereby minimizing
the response time of the system. Ma et al. [23] studied the placement of cloudlets and resource allocation in
a large-scale WMAN. The authors presented a mathematical model of average access delay between mobile
users and cloudlets and place K cloudlets in important positions of WMANs, aiming to minimize the number
of cloudlets. Xu et al. [4] also studied the problem of cloudlet placement in large-scale networks, proposed
a cloudlet placement model that minimizes the average access time between mobile users and cloudlets,
and then reduced it to an integer linear programming (ILP) problem to obtain an exact solution. Considering
the poor scalability of ILP, the authors proposed a heuristic algorithm for cloudlet placement suitable for
the proposed network and designed an approximate algorithm that guarantees the approximate ratio for spe-
cial scenarios where the cloudlet capacities are equal. Meng et al. [24] considered placing cloudlets on wire-
less APs in the network and offloading the computing tasks of mobile users on the APs to cloudlets for ex-
cution to minimize communication delays. Since the computing tasks of mobile users arrive at the APs dy-
namically, the authors first constructed several snapshots based on historical data at different time inter-
vals. Then, an approximate algorithm was derived for each snapshot to facilitate the selection of some APs
as candidate positions of cloudlets. Chen et al. [25] proposed a runtime model-based approach to IoT ap-
plication development, which made the first attempt to manage various IoT devices based on runtime soft-
ware architecture. Zhu et al. [26] proposed a modified guided population archive whale optimization al-
gorithm to solve the joint problem of cloudlet placement and task offloading. The object is to minimize
user energy consumption, task response delay, and the number of cloudlets placed. Cao et al. [27] studied the
placement of edge servers in the Internet of Vehicles, considered the transmission delay, workload balance,
energy consumption, placement cost, network reliability and the number of edge servers, constructed a mul-
tiobjective edge server placement optimization problem, and proposed a multiobjective evolutionary al-
gorithm to solve the above problems. Although the methods proposed in the above related research work
can solve the problem of cloudlet placement to a certain extent, the proposed methods are unsuitable for
scenarios where the computing tasks have complex dependencies.
3 System Model and Problem Formulation

In this section, we first describe the WMAN and WA system models. We then define the key points in the execution process of WAs. Finally, we define the problem precisely. For ease of reference, we list the key notations of our system model in Table 1.

3.1 System Model

3.1.1 WMAN Model

As shown in Fig. 1, we consider a WMAN composed of several APs. The APs are connected through a wired network to form a connectivity graph structure. It is assumed that each AP in the WMAN will receive a WA in the considered time slot, and these WAs are jointly executed by several cloudlets placed on the APs. The cloudlet placement scheme and the offloading strategy of WAs will affect the execution time of WAs. A WMAN can be represented by a graph $G = (V, E)$, and each node in $V = \{v_1, v_2, ..., v_n\}$ represents an AP in the WMAN. $E$ is the set of edges between APs, and each edge $(v_i, v_j) \in E$ represents the unit data transmission delay between $v_i$ and $v_j$. In particular, when two APs are not directly connected, the unit data transmission delay is obtained by adding the unit data transmission delay of each side in the shortest path\[1] between them. We define a matrix $D \in \mathbb{R}^{n \times n}$, where $D_{i,j} \in D$ represents the unit data transmission delay between $v_i$ and $v_j$.

3.1.2 WA Model

In a WMAN, the WA received by AP consists of several sub-tasks, and there is data dependency between them. The set of WAs received by all APs is denoted as $G^W = \{G^W_1, G^W_2, ..., G^W_n\}$, and each WA can be represented by a directed acyclic graph $G^W_i = (L_i, E_i)$. As shown in Fig. 2, $L_i = \{l^i_1, l^i_2, ..., l^i_s\}$ represents $s$ sub-tasks included in $G^W_i$, and the computational requirement of $l^i_j$ is represented by $\theta^i_j$. $E_i = \{c^i_{jk}, \forall j, k \in \{1, 2, ..., s\}\}$ indicates the data dependence between sub-tasks included in $G^W_i$, where $c^i_{jk} > 0$, there is a data dependency relationship between $l^i_j$ and $l^i_k$, and when $c^i_{jk} = 0$, there is no data dependency relationship between $l^i_j$ and $l^i_k$.

3.1.3 Cloudlet Placement and Sub-task Offloading

We use a set $C = \{c_1, c_2, ..., c_K\}$ to represent $K$ cloudlets to be placed in a WMAN, where the computing power of cloudlet $c_i$ is represented by $\eta_i$. The K cloudlet placement scheme in the WMAN can be defined as $\omega = \{\omega_{i,j}, \forall i \in \{1, ..., K\}, j \in \{1, ..., n\}\}$. When cloudlet $c_i$ is placed at AP $v_j$, $\omega_{i,j} = 1$; otherwise, $\omega_{i,j} = 0$.

Because our work focuses on cloudlet placement in a WMAN, it does not consider the impact of the sub-task offloading strategy on the WA execution results. When the cloudlet placement positions are determined, we divide the whole WMAN into several areas on average. In each area, a greedy offloading algorithm similar to that in [28] is used to offload the sub-tasks; that is, the WAs received by each area are jointly executed by the cloudlets placed in the area.

The execution order of the sub-tasks included in the same WA is determined by the breadth-first traversal (BFT) of the graph [7]. When the above greedy offloading algorithm offloads the sub-tasks, the strategy of minimizing the total time cost of task execution, namely, the sum of execution time and transmission time, is adopted. According to the execution location of each sub-task’s precursor sub-tasks, it is offloaded to the appropriate cloudlet, and the global sub-task offloading strategy is ultimately obtained, which is denoted as $M = \bigcup_{i=1}^{K^W} \{(l^i_1, c_k)|l^i_1 \in L_i, c_k \in C\}$.

3.1.4 WA Execution Time

Given the cloudlet placement scheme $\omega$ and sub-task offloading strategy $M$, the following key points are defined.

The start execution time of the sub-task. The start time of sub-task $l^i_j$ on cloudlet $c_k$ is determined by the execution end time and the dependent data transmission time of all its predecessor sub-tasks. It is quantified as

\[
t_{\text{start}}(l^i_j, c_k) = \max_{\forall e^i_{jk} > 0, \forall e^i_{k\ell}} \{t_{\text{end}}(l^i_j, c_w) + t_{\text{trans}}(l^i_j, l^i_k, c_w, c_k)\}, \quad (1)
\]

where $t_{\text{end}}(l^i_j, c_w)$ and $t_{\text{trans}}(l^i_j, l^i_k, c_w, c_k)$ are the execution end time of sub-task $l^i_j$ and the dependent data transmission time between sub-tasks $l^i_j$ and $l^i_k$, respectively. Their calculation equations will be given in Eq. (4) and Eq. (3), respectively.

The execution time of the sub-task. The execution time of sub-task $l^i_j$ is determined by its computational requirements and the computing power of the cloudlet executing the sub-task. It is quantified as

\[
t_{\text{exe}}(l^i_j, c_k) = \frac{\rho^i_j}{\eta_k}, \forall (l^i_j, c_k) \in M \quad (2)
\]
The dependent data transmission time between the sub-tasks. The dependent data transmission time between sub-tasks \( t_{\text{tr}}^p \) and \( t_{\text{tr}}^q \) is determined by the size of the dependent data between the two sub-tasks and the unit data transmission delay between the APs with cloudlets placed. It is quantified as

\[
    t_{\text{tr}}(l_i^p, l_i^q, d_{w_k}) = c_{p,j} d_{w_k}, \quad \forall (l_i^p, l_i^q, d_{w_k}) \in M, \omega_{w,w'}, \omega_{k,k'} = 1
\]  

(3)

The execution end time of the sub-task. The execution end time of sub-task \( l_i^p \) on cloudlet \( c_k \) is determined by the start execution time of sub-task \( l_i^p \) and its execution time on cloudlet \( c_k \). It is quantified as

\[
    t_{\text{end}}(l_i^p, c_k) = t_{\text{start}}(l_i^p, c_k) + t_{\text{exe}}(l_i^p, c_k), \quad \forall (l_i^p, c_k) \in M
\]  

(4)

The execution end time of the WA. The execution end time of WA \( G_i^W \) is the maximum of the execution end time of its sub-tasks. It is quantified as

\[
    t_{\text{end}}(G_i^W) = \max \{ t_{\text{end}}(l_i^p, c_k) \}, \forall (l_i^p, c_k) \in M
\]  

(5)

The execution end time of all WAs. The end time of all WAs in the WMAN is the maximum of the execution end time of all the WAs. It is quantified as

\[
    t_{\text{end}}(G_i^W) = \max \{ t_{\text{end}}(G_i^W) \}
\]  

(6)

3.2 Problem Formulation

The \( K \) cloudlet placement based on the WA problem (KCPWP) in a WMAN is defined as follows. Given an integer \( K > 1 \) and system model parameters \((G, D, G_i^W)\), the problem is to find \( \omega \) such that the execution time of WAs in Eq. (6) is minimized, i.e.,

\[
    \min_{\omega} \ t_{\text{end}}(G_i^W)
\]  

(7)

s.t. \[
    \sum_{j=1}^{n} \omega_{i,j} = 1, \forall i \in \{1, ..., K\} \quad \text{(C1)}
\]

\[
    \sum_{i=1}^{K} \omega_{i,j} \leq 1, \forall j \in \{1, ..., n\} \quad \text{(C2)}
\]

Constraint C1 indicates that each cloudlet to be placed has only one placement position, and constraint C2 indicates that each AP has at most one cloudlet to be placed.

4 Cloudlet Placement Strategy Based on PGEL

In this section, we first introduce the traditional PSO algorithm and then introduce the cloudlet placement strategy based on PGEL in detail.

4.1 Traditional PSO

The PSO algorithm is a stochastic optimization technique based on population and was proposed by Eberhart and Kennedy in 1995 [29]. In nature, animals that belong to the same population cooperate with each other in a certain way. Each member of the population changes its behaviour by learning its own and others’ experiences. The PSO algorithm solves the optimization problem by imitating the clustering behaviour of animals. In the PSO algorithm, a particle represents a candidate solution of the optimization problem, and all particles can move in the whole solution space. In the process of each search, the particle move at a certain speed, which is affected by three factors: the situation of the particle itself, the best position of the particle itself, and the historical best position of the particle in the whole particle swarm. Although the traditional PSO algorithm has the advantages of good robustness and easy convergence, it also has defects of developing premature convergence and becoming trapped in a local optimum. In this paper, by introducing the crossover operator and mutation operator of the GA into the traditional PSO algorithm, the PGEL cloudlet placement algorithm is proposed to compensate for the defects of the traditional PSO algorithm, optimize its optimization ability, and better solve the \( K \) cloudlet placement problem in WMNs.

4.2 PGEL

4.2.1 Problem Encoding

To solve the KCPWP in WMNs, we use the cloudlet position sequence encoding strategy to encode the particles. Each particle in the particle swarm composed of \( \Omega \) particles represents a placement scheme of \( K \) cloudlets in a WMAN. The state of the \( i \)-th particle in the \( t \)-th iteration is as follows:

\[
    P_i^t = (p_{i,1}^t, p_{i,2}^t, ..., p_{i,K}^t)
\]  

(8)

where \( p_{i,j}^t \in \{1, 2, ..., n\} \) represents the placement position of the \( j \)-th cloudlet mapped by particle \( i \) in the \( t \)-th iteration.

4.2.2 Fitness Function

To judge the cloudlet placement scheme corresponding to each particle, a fitness function is introduced. Our purpose is to obtain a cloudlet placement scheme that can minimize the execution time of WAs in a
WMAN. Therefore, the particle with a smaller execution time corresponding to the mapped cloudlet placement scheme can be simply regarded as a better particle. The fitness function of particle $P_i^t$ can be defined as

$$fitness(P_i^t) = Time(P_i^t)$$ (9)

where $Time(P_i^t)$ represents the execution time of WAs calculated by Eq. (6) when the placement scheme corresponding to $P_i^t$ is adopted. The particle with a smaller fitness obviously corresponds to the cloudlet placement scheme with a shorter execution time for WAs.

### 4.2.3 Update Strategy

In our previous work [30], the update strategy of the traditional PSO algorithm was introduced in detail. The PSO includes three core parts: inertia, personal cognition, and social cognition. In the iterative process of the algorithm, the update of each particle is affected by its personal optimal position and the current global optimal position. To avoid the tendency of the traditional PSO algorithm to fall into the local optimum prematurely and to enhance the search ability of the algorithm, we introduce the crossover operator in the personal cognitive and social cognitive domains and the mutation operator in the inertia part to compensate for the defects of the traditional PSO algorithm. The update strategy of particle $i$ in the $(t + 1)$-th iteration is as follows:

$$P_i^{t+1} =$$
$$SC(PC(IT(P_i^t, w^{t+1}, \mu), pBest_i^{t+1}, c_1^{t+1}), gBest^t, c_2^{t+1})$$ (10)

where $PC(x, y, z)$ and $SC(x, y, z)$ are the personal cognition update operation and the social cognition update operation, respectively, and $IT(x, y)$ is the inertia update operation.

The crossover operator of the GA is introduced into the personal cognitive update operation and social cognitive update operation. The results of the personal cognition update operation and social cognition update operation are quantified as

$$PC(A_i^{t+1}, pBest_i^t, c_1^{t+1}) =$$
$$\begin{cases} 
CO(A_i^{t+1}, pBest_i^t), & \text{if } r_1 \leq c_1^{t+1} \\
A_i^{t+1}, & \text{if } r_1 > c_1^{t+1} 
\end{cases}$$ (11)

$$SC(B_i^{t+1}, gBest^t, c_2^{t+1}) =$$
$$\begin{cases} 
CO(B_i^{t+1}, gBest^t), & \text{if } r_2 \leq c_2^{t+1} \\
B_i^{t+1}, & \text{if } r_2 > c_2^{t+1} 
\end{cases}$$ (12)

respectively, where $r_1$ and $r_2$ are random numbers from 0 to 1, and $CO(x, y)$ represents the crossover operator of GA. The crossover operator randomly selects an encoded segment of particle $x$ to be updated and replaces it with the corresponding encoded segment of particle $y$. The crossover operator in the personal (or social) cognitive update operation is shown in Fig. 3.

The mutation operator of the GA is introduced into the inertia update operation. The result of the inertia update operation is as follows

$$IT(P_i^t, w^{t+1}, \mu) = \begin{cases} 
MU(P_i^t, \mu), & \text{if } r_3 \leq w^{t+1} \\
P_i^t, & \text{if } r_3 > w^{t+1} 
\end{cases}$$ (13)

where $r_3$ is a random number from 0 to 1 and $MU(x)$ represents the mutation operator of the GA. The mutation operator randomly selects $\mu$ encodings of particle $x$ to be updated and then randomly changes the values of these encodings. The mutation operator in the inertial update operation is shown in Fig. 4. When $\mu = 2$, the encoded $ind_1$ and $ind_2$ are selected; that is, the mutation operator selects the corresponding two cloudlets have changed.

### A. Traditional updating mode

In our previous work [30], we used the traditional updating mode to update the encodings of the particle. The traditional updating mode is as follows.

1) Initialization: Randomly encode all encoding bits of the particle and ensure that the encodings are not equal.

2) Crossover: If the particle’s encoding obtained by the crossover operator conflicts with the original encoding; that is, it does not meet the constraint that an AP places only a cloudlet but needs to adjust the encoding according to certain rules (such as the most equivalent replacement method) to ensure that every encoding is not equal. As shown in Fig. 5, the crossed 2 and 4 conflict with the original encoding and need to be adjusted to avoid conflict.

3) Mutation: Similarly, if the particle’s encoding obtained by the mutation operator conflicts with the original encoding, the mutated encoding needs to be adjusted to ensure that the two encodings are not equal. As shown in Fig. 6, 5 in the original encoding changes to 1, and the new encoding conflicts with the original encoding. One of the adjustment schemes is to change 1 to 5 in the mutated encoding.

After the mutation operator and crossover operator are executed in the traditional updating mode, the encodings in the same particle may conflict and need an additional adjustment process, which will result in additional execution time during the execution process of the algorithm.

### B. Encoding library updating mode
In PGEL, we use the encoding library updating mode to avoid possible encoding conflicts to avoid the extra time caused by the adjustment process. We give the encoding library for each particle and update it in the following way.

1) Initialization: When the particles are initialized, the encodings are removed from the encoding library one by one so that they do not conflict.

2) Crossover: In the process of executing the crossover operator, first add the replaced encodings of the particle back to their own encoding library, and then check the encodings obtained by the cross one by one. If encoding exists in the encoding library, delete it from the encoding library. If encoding does not exist in the encoding library, the closest encoding is selected from the encoding library to replace it. The label of the AP is determined by its specific spatial location, so the location of the access point in the network represented by the closest value is the closest, so the above operation can retain the original particle placement scheme to the greatest extent.

3) Mutation: When the mutation operator needs to be executed, an encoding is randomly selected from the encoding library to replace the original encoding, and then, the original encoding is added to the encoding library.

The encoding library update mode can ensure that the encodings in the particles do not conflict and effectively shorten the execution time of the crossover operator and the mutation operator.

### 4.2.4 Map from a Particle-to-Cloudlet Placement Scheme

The mapping algorithm of the particle-to-cloudlet placement scheme is shown in **Algorithm 1**.

**Algorithm 1**: Mapping particle to cloudlet placement scheme.

```plaintext
Input: Particle \( P^t \).
Output: Cloudlet placement scheme \( \omega \).

1. begin
2. \( \text{foreach} \: \omega_{i,j} \in \omega \: \text{do} \)
3. \( \omega_{i,j} = 0; \)
4. \( \text{end} \)
5. \( \text{foreach} \: P^t_{i,j} \in P^t \: \text{do} \)
6. \( \omega_{j,p^t_{i,j}} = 1; \)
7. \( \text{end} \)
8. return \( \omega \)
9. end
```

The input of **Algorithm 1** is particle \( P^t \), and the output is cloudlet placement scheme \( \omega \). First, cloudlet placement scheme \( \omega \) is initialized (Lines 2-4); second, the placement position of each cloudlet is determined according to each encoding of the particle (Lines 5-7); and last, cloudlet placement scheme \( \omega \) is output (Line 8).

#### 4.2.5 Parameter Settings

The inertia weight factor \( w \) determines the convergence and search ability of PGEL, so its setting is very important. According to Eq. (13), when \( w \) is small, the probability of particle mutation is small, and the current state of the particle has a great influence on the update of the next state, so PGEL has strong local search ability. In contrast, when \( w \) is large, the probability of particle mutation is large, and the current state of the particle has little influence on the next state update, so PGEL has strong global search ability. In the early stage of algorithm implementation, our emphasis is on giving more attention to the global search ability. As the search deepens, our focus is on concentrating more attention on the local search ability. In conclusion, \( w \) should evolve with the evolution of population particles. In PGEL, an inertia weight factor adjustment strategy is used, which is adjusted adaptively according to the advantages and disadvantages of the current population of particles. In the \( t \)-th iteration, the value of the inertia weight factor is as follows:

\[
w^t = w^t_{\text{max}} - (w^t_{\text{max}} - w^t_{\text{min}}) \times \exp \left( \frac{d(P^t-1)}{d(P^t-1) - 1.01n} \right)
\]

where \( w^t_{\text{max}} \) and \( w^t_{\text{min}} \) are the maximum and minimum values of the inertia weight factor, respectively, and \( d(P^t-1) \) is the number of bits encoded differently between particle \( P^t-1 \) and the global best particle.

In addition, the personal cognitive factor \( c^1_{\text{max}} \) and the social cognitive factor \( c^2_{\text{max}} \) are set by the linearly increasing strategy and the linearly decreasing strategy in the iterative process [31], respectively. They are quantified as

\[
c^1_1 = c^1_{\text{min}} + t \times \frac{c^1_{\text{max}} - c^1_{\text{min}}}{t_{\text{max}}}
\]

\[
c^2_2 = c^2_{\text{max}} - t \times \frac{c^2_{\text{max}} - c^2_{\text{min}}}{t_{\text{max}}}
\]

Here, where \( c^1_{\text{max}} \) and \( c^2_{\text{max}} \) are the maximum values of the personal cognitive factor and social cognitive factor, respectively; \( c^1_{\text{min}} \) and \( c^2_{\text{min}} \) are the minimum values of the personal cognitive factor and social cognitive factor, respectively; and \( t_{\text{max}} \) is the maximum number of iterations of the algorithm.
4.2.6 Algorithm Flow

The main steps of PGEL are as follows.

**Step 1:** The parameters of PGEL are initialized, and then the initial population is generated.

**Step 2:** According to the mapping between cloudlets and APs, the fitness of each particle is calculated according to Eq. (9). The personal best state of each particle is initialized by its initial state, and the particle with the least fitness in the initial particle swarm is set as the global best particle.

**Step 3:** The particles are updated one by one according to Eq. (10), and the fitness of the particles is calculated after updating.

**Step 4:** If the fitness of the updated particle is less than that of its personal best, it is set as its personal best. Otherwise, go to **Step 6**.

**Step 5:** If the fitness of the updated particle is less than that of the global best particle, it is set as the global best particle.

**Step 6:** Verify that the stop condition is met. If the stop condition is not satisfied, return to **Step 3**; otherwise, the algorithm will terminate.

5 Performance Evaluation

In this section, to verify the effectiveness of the proposed placement strategy based on PGEL in solving KCPWP in WMANs, the experimental evaluation is carried out in the simulation environment. In particular, the following research questions (RQs) are verified by simulation experiments.

- **RQ1:** Can PGEL obtain a near-optimal cloudlet placement scheme in typical WMANs? (5.3)
- **RQ2:** What is the impact of changes in the WMAN on the performance of PGEL? (5.4)
- **RQ3:** In typical WMANs, compared with several classic algorithms, does PGEL have performance advantages when solving KCPWP? (5.5)

For **RQ1**, the experimental results show that PGEL can obtain a near-optimal cloudlet placement scheme in typical WMANs. For **RQ2**, it can be seen from the experimental results that regardless of how WMANs change, the performance of PGEL is not affected, and a near-optimal cloudlet placement scheme can be obtained. For **RQ3**, the experimental results show that PGEL is superior to other algorithms in typical WMANs.

5.1 Experimental Settings

All simulation experiments in this section are carried out on a PC equipped with an i5-8500 CPU and 32 GB of RAM. The operating system version is Windows 10-2004. PGEL and all classic algorithms are implemented in Python 3.7. The relevant parameters of PGEL refer to [30] and are set as $t_{\text{max}} = 1000$, $\Omega = 100$, $w_{\text{max}} = 0.9$, $w_{\text{min}} = 0.4$, $c_{1\max} = 0.9$, $c_{1\min} = 0.2$, $c_{2\max} = 0.9$, $c_{2\min} = 0.4$, and $\mu = 0.1K$.

We select the regions from the dataset of the Shanghai Telecom base station [32] to simulate the WMANs according to a certain longitude and latitude span. Different WMANs include different numbers and connection topologies for APs. The unit data transmission delay of each edge in the WMAN is generated randomly from 5 ms to 50 ms [33]. During the process of sub-task offloading, the WMAN is divided into $2 \times 2$ areas.

5.2 Classic Algorithms

To verify the advantages of PGEL, we introduce the following classic algorithms in the simulation experiments.

- **Optimal placement algorithm (OPT):** This algorithm traverses all possible cloudlet placement schemes and selects the placement scheme with the minimum execution time of WAs as the optimal cloudlet placement scheme.
- **PSO:** This traditional PSO algorithm has the same encoding and parameter settings as PGEL.
- **GA:** According to the update strategy of GA, it uses the elite retention strategy, binary tournament selection operator, two-point crossover operator, and exchange mutation operator to update the chromosome and takes the final elite solution as the optimal solution.
- **Random cloudlet placement algorithm (RAN):** This algorithm randomly selects $K$ from all APs of the WMAN and randomly places $K$ to-be-placed cloudlets on these APs. The result is an average of 100 repetitions.

5.3 RQ1. PGEL can obtain a near-optimal cloudlet placement scheme in typical WMANs

According to the administrative divisions of Shanghai, we selected regions with spans of 8 and 4 points in latitude and longitude, respectively, from Huangpu, Xuhui, Minhang, and Pudong as typical WMANs. The distribution and topology of APs in the WMAN are simulated by the data of the Telecom base station in [32]. It is assumed that the number of to-be-placed cloudlets $K$ in the WMAN is determined by the number of APs $n_i$ that is, $K = \kappa \cdot n_i, \kappa \in (0, 1)$. According to the population density of the above administrative divisions, the $\kappa$ values of the selected regions from Huangpu, Xuhui, Minhang, and Pudong are set to be 0.35, 0.3, 0.25, and 0.2, respectively. In the above regions, the WAs received by APs are randomly generated from various network architectures, namely, AlexNet, Visual Geometry Group network (VGGNet), GoogLeNet, and residual network (ResNet) [34]. In addition, the clock frequency of the CPU equipped with
the to-be-placed cloudlets in the same WMAN satisfies a uniform distribution of 2 GHz to 3 GHz.

By running OPT and PGEL in different typical WMANs, we can obtain the local cloudlet placement results, which are shown in Fig. 7. Although the cloudlet placement scheme obtained by PGEL in each typical WMAN is not equivalent to the optimal cloudlet placement scheme, there are only 1-2 cloudlet placement positions that differ locally; other cloudlet placement positions are the same, and the overall placement scheme is close.

By calculating the execution time of the WAs corresponding to the cloudlet placement scheme shown in Fig. 7, the histogram shown in Fig. 8 can be obtained. The results show that in four different typical WMANs (Huangpu, Xuhui, Minhang, and Pudong), and the execution times of the WAs corresponding to the cloudlet placement scheme obtained by PGEL are 7.51\%, 11.91\%, 2.06\%, and 0.83% larger than those corresponding to the optimal placement scheme, respectively. The combined insights of Fig. 7 and Fig. 8 indicate that although the cloudlet placement schemes obtained by PGEL in the WMANs somewhat differ from the optimal placement schemes, the difference is very small. PGEL can obtain the approximate optimal solution of KCPWP in typical WMANs. Although there is still a certain gap between the optimal placement scheme and the placement scheme obtained by PGEL, it is unfeasible to obtain the optimal placement scheme by brute force when the network scale increases, so PGEL has a certain practicality.

5.4 RQ2. The changes in the WMAN have almost no impact on the performance of PGEL

From Subsection 5.3, we know that PGEL can very closely approach the optimal cloudlet placement scheme in typical WMANs. However, the WMAN cannot be fixed; it is both temporal and spatial. Next, we will discuss the impact of WMAN changes on PGEL performance.

5.4.1 The impact of changes in AP topology on the performance of PGEL

We use the regions selected from different administrative divisions of Shanghai as described in Subsection 5.3 to simulate changes in AP topology. In this part, it is assumed that the number of to-be-placed cloudlets in each region and the clock frequency of the CPU equipped with the to-be-placed cloudlets are the same, and other settings are the same as those in Subsection 5.3. In addition, the WAs received by APs are simulated by GoogLeNet, the structure of which is shown in [35].

As shown in Fig. 9(a), in four WMANs with different AP topologies, although the number of to-be-placed cloudlets and their CPU clock frequency are the same, the execution times of the identical number and type of WAs are different. This is because the AP topology will affect the placement positions of cloudlets, resulting in different sub-task offloading schemes and data transmission times. From the experimental results, it can be concluded that in four WMANs with different AP topologies (Huangpu, Xuhui, Minhang, and Pudong), the execution times of the WAs corresponding to the cloudlet placement scheme obtained by PGEL are 5.80\%, 7.51\%, 3.04\% and 3.13\% larger than those corresponding to the optimal placement scheme, respectively. Although there is a gap between them, it can almost be ignored. On the basis of these results, we can conclude that the changes in the AP topology have little impact on the performance of PGEL, and PGEL can still obtain a near-optimal cloudlet placement scheme.

5.4.2 The impact of the changes in the to-be-placed cloudlet CPU clock frequency on the performance of PGEL

In Subsection 5.3, we assume that the CPU clock frequency of to-be-placed cloudlets meets the uniform distribution within a certain range. In this part, we simulate the changes in the CPU clock frequency of the to-be-placed cloudlets by changing the maximum and minimum values of the uniform distribution. In addition, the first typical WMAN (i.e., Huangpu) of Subsection 5.3 is adopted. The WAs received by APs are simulated by implementing GoogLeNet.

The results shown in Fig. 9(b) indicate that the execution time of the WAs is negatively correlated with the CPU clock frequency of to-be-placed cloudlets. This is because the CPU clock frequency determines the execution time of sub-tasks. The greater the CPU clock frequency is, the smaller the execution time of sub-tasks. The execution time of sub-tasks largely determines the execution time of WAs. In addition, the experimental results indicate that the execution time of the WAs corresponding to the cloud placement scheme obtained by PGEL is 3.94\%, 3.76\%, 2.03\%, and 6.41\% larger than the optimal placement scheme under the four to-be-placed cloudlet CPU clock frequencies, respectively. Although there is a gap between the values, it can almost be ignored. From these results, we can conclude that the CPU clock frequency changes of the to-be-placed cloudlets have little impact on the performance of PGEL, and PGEL can still be used to obtain the near-optimal cloudlet placement scheme.

5.4.3 The impact of the changes in WAs on the performance of PGEL

In Subsection 5.3, we assume that the WAs randomly select from four typical tasks. In this part, we simulate
the changes in WAs by changing the type of WAs. In addition, the relevant settings of the first typical WMAN of Subsection 5.3 are adopted, and all WAs are equivalent.

As shown in Fig. 9(c), when the types of WAs are different, the execution times of the WAs are different. This is because different types of WAs have different computational requirements and data transmission rates, so the execution time is different. From the experimental results, when the WA types are AlexNet, VGGNet, GoogLeNet, and ResNet, the execution times of the WAs corresponding to the cloudlet placement scheme obtained by PGEL are 4.27%, 2.06%, 0.15%, and 0.95% larger, respectively, than those corresponding to the optimal placement scheme. Although there is a gap between the values, it can almost be ignored. From these results, we can conclude that the changes in the WAs have little impact on the performance of PGEL, and PGEL can still obtain a near-optimal cloudlet placement scheme.

In summary, we can conclude that changes in the WMANs have almost no impact on the performance of PGEL, and PGEL can still obtain a near-optimal cloudlet placement scheme.

5.5 RQ3. PGEL has greater performance advantages than several classic algorithms in typical WMANs do.

Finally, we use experiments to test the performance advantages of the proposed PGEL over several classic algorithms when solving KCPWP in typical WMANs. As shown in Fig. 10, in the four typical WMANs discussed in Subsection 5.3, PGEL reduced the execution time of WAs by 15.04%, 31.32%, and 44.99% on average compared with those of the PSO, GA, and RAN, respectively. PGEL adaptively adjusts the search capability according to the current situation and iteratively updates from a global perspective. In the four different WMANs, PGEL performs better than several classic algorithms do. Although the PSO algorithm has a certain search ability, it easily falls into a local optimum, so its performance is limited. The GA performs only a partial search during each iteration, so its performance is poor, and RAN can reflect only the average level of all placement schemes. In summary, it can be concluded that PGEL has a greater performance advantage than do several classic algorithms in solving KCPWP in WMANs.

6 Conclusion

For mobile users seeking solutions to the problem of insufficient resources through remote servers, cloudlet technology is essential. However, the problem of cloudlet placement has been largely ignored. To solve the problem of cloudlet placement, this paper proposed a time-driven cloudlet placement strategy based on PGEL, which aims to reduce the execution time of WAs in WMANs. We set up a reasonable simulation environment and conducted a complete simulation experiment based on this approach. The results show that compared with the existing solutions, our proposed strategy has better performance and is extraordinarily close to the optimal placement scheme.

In the future, we will consider the impact of environmental fluctuations (i.e., network delays, bandwidth fluctuations, and cloudlet failures) on cloudlet placement and improve existing strategies to adapt to the dynamic environment. In addition, we will fully consider the characteristics of the sub-tasks included in the WA and the placement cost of the cloudlet in different positions and design a more scientific and complete system cost model.

Acknowledgements

We thank AIE (https://secure.aie.com) for its linguistic assistance during the preparation of this manuscript.

Funding

This work was partly supported by the Key-Area Research and Development Program of Guangdong Province under Grant No.2020B0101090005.

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

Jianhan Zhang and Ming Li drafted the original manuscript and designed the experiments. Xianghui Cheng provided ideas and suggestions. Ching-Hsien Hsu provided critical review and helped to draft the manuscript. The authors read and approved the final manuscript.

Authors' information

Jianhan Zhang received his M.S. degree in Materials Engineering from Fujian Normal University, China, in 2018. He is currently working toward the M.Degree at the College of Mathematics and Computer Science, Fuzhou University. He has also been a part of the Fujian Key Laboratory of Network Computing and Intelligent Information Processing at Fuzhou University since September 2019. His current research interests include edge computing, computational intelligence and cloud computing. Ming Li received the B.S. degree in Computer Science and Technology from Fuzhou University, Fujian, China, in 2019, where he is currently working toward the M.S. degree in computer science in the College of Mathematics and Computer Science, Fuzhou University. He has also been a part of the Fujian Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, since September 2019. His current research interests include systems software, edge computing. Xianghui Cheng is currently a Professor with the College of Mathematics and Computer Sciences, Fuzhou University, China. He received the M.S. degree in distributed systems and the Ph.D. degree in information communication technology from the University of Agder, Norway, in 2007 and 2011, respectively. His current research interests include new generation networks with a special focus on cloud computing services and applications, and big data processing and security. Ching-Hsien Hsu is currently a Chair Professor and the Dean of the College of Information and Electrical Engineering, Asia University, Taiwan, and a Professor with the Department of Computer Science and Information Engineering, National Chung Cheng University. His research interests include high performance computing, cloud computing, parallel and distributed systems, big data analytics, and ubiquitous/pervasive computing and intelligence.
Author details
1 College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. 2 Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou, China. 3 Department of Computer Science and Information Engineering, Asia University, Taichung, Taiwan.

References
1. Liu, F., Shu, P., Jin, H., Ding, L., Yu, J., Niu, D., Li, B.: Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wireless communications 20(3), 14–22 (2013)
2. Huang, D., Xie, T., Wu, H.: Mobile cloud computing service models: a taxonomic approach. IEEE network 27(5), 6–11 (2013)
3. Jarareh, Y., Doubt, A., AlQudah, O., Ahmed, E., Al-Ayyoub, M., Benkhelifa, E.: The future of mobile cloud computing: Integrating cloudlets and mobile edge computing. In: International Conference on Telecommunications (ICT) (2016)
4. Xu, Z., Liang, W., Xu, W., Jia, M., Guo, S.: Efficient algorithms for capacitated cloudlet placements. IEEE Transactions on Parallel and Distributed Systems 27(10), 2866–2880 (2015)
5. Xu, Z., Liang, W., Xu, W., Jia, M., Guo, S.: Capacitated cloudlet placements in wireless metropolitan area networks. In: IEEE 49th Conference on Local Computer Networks (LCN) (2015)
6. Jia, M., Cao, J., Liang, W.: Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Transactions on Cloud Computing 3(4), 725–737 (2015)
7. Chen, M., Guo, S., Liu, K., Luo, X., Xiao, B.: Robust computation offloading and resource scheduling in cloudlet-based mobile cloud computing. IEEE Transactions on Mobile Computing 20(5), 2025–2040 (2021)
8. Chun, B.-G., Ihm, S., Manalis, P., Nalk, M., Patti, A.: Cloud: elastic execution between mobile device and cloud. In: The Sixth Conference on Computer Systems (CCS) (2011)
9. Cuervo, E., Babušekmanian, A., Cho, D.-K., Wolman, A., Saroiu, S., Chandra, R., Bahl, P.: MAUI: making smartphones last longer with code offload. In: The 8th International Conference on Mobile Systems, Applications, and Services (ICMAS) (2010)
10. Kosta, S., Ascinas, A., Hui, P., Mortier, R., Zhang, X.: Thinks: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In: IEEE International Conference on Computer Communications (INFOCOM) (2012)
11. Committee, L.S., et al.: IEEE standard for local and metropolitan area networks: Overview and architecture (en line). New York-NY-USA, The Institute of Electrical and Electronics Engineers Inc (2002)
12. Hu, K., Filiz, P., Richter, W., Abe, Y., Satyanarayanan, M.: Just-in-time provisioning for cloud computing. In: The 11th Annual International Conference on Mobile Systems, Applications, and Services (ICMAS) (2013)
13. Zhang, Y., Liu, H., Xiao, X., Fu, X.: To offload or not to offload: An efficient code partition algorithm for mobile cloud computing. In: IEEE 1st International Conference on Cloud Networking (CLUDNET) (2012)
14. Zhang, Y., Huang, G., Liu, X., Zhang, W., Mei, H., Yang, S.: Refactoring android java code for on-demand computation offloading. In: Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA) (2012)
15. Ra, M.R., Seth, A., Mummert, L., Filiz, P., Wetterhall, D., Govindan, R.: Odem: enabling interactive perception applications on mobile devices. In: The 9th International Conference on Mobile Systems, Applications, and Services (ICMAS) (2011)
16. Hoang, D.T., Niyato, D., Wang, P.: Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In: IEEE Wireless Communications and Networking Conference (WCNC) (2012)
17. Chen, X., Hu, J., Chen, Z., Liu, B., Xiong, N., Min, G.: A reinforcement learning empowered feedback control system for industrial internet of things. IEEE Transactions on Industrial Informatics (2021). doi:10.1109/TII.2021.3076393
18. Guan, S., Boukerche, A.: A novel mobility-aware offloading management scheme in sustainable multi-access edge computing. IEEE Transactions on Sustainable Computing (2021). doi:10.1109/TUSC.2021.3053310
19. Zhang, Y., Niyato, D., Wang, P.: Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Transactions on Mobile Computing 14(12), 2516–2529 (2015)
20. Gai, K., Qi, M., Zhao, H., Tao, L., Zong, Z.: Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. Journal of Network and Computer Applications 59, 46–54 (2016)
21. Mukhejee, A., De, D., Roy, D.G.: A power and latency aware cloudlet selection strategy for multi-cloud environment. IEEE Transactions on Cloud Computing 7(1), 141–154 (2016)
22. Jia, M., Cao, J., Liang, W.: Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Transactions on Cloud Computing 5(4), 725–737 (2015)
23. Ma, L., Wu, J., Chen, L.: Datas: Delay bounded optimal cloudlet deployment and user association in WMANs. In: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (2017)
24. Meng, J., Shi, W., Tan, H., Li, X.: Cloudlet placement and minimum-delay routing in cloud computing. In: International Conference on Big Data Computing and Communications (BIGCOMP) (2017)
25. Chen, X., Li, A., Guo, W., Huang, G., et al.: Runtime model based approach to iot application development. Frontiers of Computer Science 9(4), 540–553 (2015)
26. Zhu, X., Zhou, M.C.: Multi-objective optimized cloudlet deployment and task offloading for mobile edge computing. IEEE Internet of Things Journal (2021). doi:10.1109/JIOT.2021.3073113
27. Cao, B., Fan, S., Zhao, J., Tian, S., Zheng, Z., Yan, Y., Yang, P.: Large-scale many-objective deployment optimization of edge servers. IEEE Transactions on Intelligent Transportation Systems 22(6), 3841–3849 (2021)
28. Chen, Z., Hu, J., Chen, X., Hu, J., Zheng, X., Min, G.: Computation offloading and task scheduling for DNN-based applications in cloud-edge computing. IEEE Access 8, 115537–115547 (2020)
29. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: The International Conference on Neural Networks (ICNN) (1995)
30. Guo, W., Lin, B., Chen, G., Chen, Y., Liang, F.: Cost-driven scheduling for deadline-based workflow across multiple clouds. IEEE Transactions on Network and Service Management 15(4), 1571–1585 (2018)
31. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation Proceedings (ICECP) (1998)
32. Wang, S., Zhao, Y., Huang, L., Xu, J., Hsu, C.-H.: Qos prediction for service recommendations in mobile edge computing. Journal of Parallel and Distributed Computing 127, 134–144 (2018)
33. Xu, Z., Liang, W., Xu, W., Jia, M., Guo, S.: Efficient algorithms for capacitated cloudlet placements. IEEE Transactions on Parallel and Distributed Systems 27(10), 2866–2880 (2015)
34. Chavan, T.R., Nandedkar, A.V.: A hybrid deep neural network for online learning. In: International Conference on Advances in Pattern Recognition (ICPR) (2017)
35. Zhang, J.: GitHub. https://github.com/JamesZJS/dataset

Tables

Figures
**Table 1 Summary of key notations**

| Notation | Description |
|----------|-------------|
| $G = (V, E)$ | a WMAN |
| $v_i \in V$ | an AP in WMAN |
| $D_{i,j} \in D$ | unit data transmission delay between $v_i$ and $v_j$ |
| $G^W = \{G^W_1, ..., G^W_n\}$ | set of WAs received by all APs |
| $G^W_i = (L_i, E_i)$ | a WA |
| $L_i = \{l^i_1, ..., l^i_s\}$ | sub-tasks included in $G^W_i$ |
| $E_i = \{e^{i,k}_j\}$ | data dependence between sub-tasks included in $G^W_i$ |
| $\theta^i_j$ | computational requirement of $l^i_j$ |
| $C = \{c_1, ..., c_K\}$ | computing power of cloudlet $c_i$ |
| $\omega = \{\omega_{i,j}\}$ | cloudlet placement scheme |
| $M = \bigcup_{i=1}^{G^W} \{(l^i_1, c_k)\}$ | global sub-task offloading strategy |
| $t_{start}(l^i_1, c_k)$ | start time of sub-task $l^i_1$ on cloudlet $c_k$ |
| $t_{exe}(l^i_1, c_k)$ | execution time of sub-task $l^i_1$ |
| $t_{tras}(l^i_1, l^i_2, c_w, c_k)$ | dependent data transmission time between sub-tasks $l^i_1$ and $l^i_2$ |
| $t_{end}(l^i_1, l^i_2, c_w, c_k)$ | execution end time of sub-task $l^i_1$ on cloudlet $c_k$ |
| $t_{end}(G^W_i)$ | execution end time of WA $G^W_i$ |
| $t_{end}(G^W)$ | The end time of all WAs in the WMAN |

**Figure 1** Cloudlet placement and WA offloading in a WMAN.

**Figure 2** Diagram of the WA.

**Figure 3** Crossover operator in the personal cognitive update operation and the social cognitive update operation.

**Figure 4** Mutation operator in the inertial update operation.
Cloudlets 1 2 3 4 5
Old particle 1 0 1 5 4 2
Old particle 2 5 2 4 3 6
Crossover particle 0 2 4 4 2
New particle 0 2 4 3 1

Figure 5 Traditional crossover operator.

Cloudlets 1 2 3 4 5
Old particle 0 1 5 4 2
Mutant particle 0 1 1 4 2
New particle 0 3 1 4 2

Figure 6 Traditional mutation operator.
Figure 7 Comparison of local cloudlet placement results in typical WMANs.

Figure 8 Comparison of execution times of WAs in typical WMANs.
Figure 9 The impact of WMAN changes on the performance of PGEL.

Figure 10 Comparison of cloudlet placement.