Hindawi
Scientific Programming
Volume 2018, Article ID 6860359, 13 pages
https://doi.org/10.1155/2018/6860359

*Research Article*

# Resource Scheduling Based on Improved Spectral Clustering Algorithm in Edge Computing

**Guangshun Li, Shuzhen Xu, Junhua Wu, and Heng Ding**

*School of Information Science and Engineering, Qufu Normal University, Rizhao 276800, China*

Correspondence should be addressed to Guangshun Li; 30752585@qq.com

Received 27 October 2017; Revised 28 March 2018; Accepted 17 April 2018; Published 8 July 2018

Academic Editor: Feiyi Wang

With the development of Internet of Things (IoT), the massive data generated by it forms big data, and the complexity of dealing with big data brings challenges to resource scheduling in edge computing. In order to solve the problem of resource scheduling and improve the satisfaction of users in edge computing environment, we propose a user-oriented improved spectral clustering scheduling algorithm (ISCM) in this paper. Based on the improved k-means algorithm, the ISCM algorithm solves the problem that the clustering result is sensitive to the initial value and realizes the reclustering, which makes the obtained clustering results more stable. Finally, the edge computing resource scheduling scheme is obtained based on the clustering results. The experimental results show that the resource scheduling scheme based on improved spectral clustering algorithm is superior to traditional spectral clustering algorithm in edge computing environment.

## 1. Introduction

Edge computing is an extension of cloud computing; it is a way to quickly process data at the edge of the network [1–3].

The goal of the development of Internet of Things (IoT) is to connect electronic devices, mobile terminals, household appliances, and so on [4–6]. These devices are not only large in number but also widespread in geographical distribution. In the real world, when users use these IoT devices, there are various requirements for cloud computing, such as low latency and location awareness [7–10]. In addition, with the advent of the big data era, traditional data analysis methods and storage technology encounter bottlenecks. Big data has the characteristics of large data volume, diversity of data structure, and the requirements of fast processing [11, 12]. However, the reception and transmission of large amounts of data may cause the I/O bottleneck between the cloud computing data center and the terminal; the transmission rate and the processing speed are greatly reduced and even cause a large resource scheduling delay.

Processing range of edge computing is closer to the data source. After the data collection, it is processed at the edge of the network rather than in the central server; it provides edge intelligent services nearly with low latency and location awareness and other features and support mobility. In the Internet of Things, the edge computing through the data analysis and processing not only realizes the sensing, interaction, and control between objects, but also provides real-time resource scheduling and allocation for users [13]. Obviously, edge computing can meet the needs of these IoT devices.

With the various demand for users [14], users can choose to store data to the local edge computing or send it to the cloud computing. In this case, resource allocation and scheduling are user oriented, so task-oriented scheduling algorithms are not applicable. At present, most of the existing scheduling algorithms are applied to grid computing and virtual machine scheduling; it can not satisfy the characteristics of edge computing, such as low latency, close to the end user, wide geographical distribution, and other characteristics, and the general scheduling algorithm can not meet the diversified demands of users. To solve these problems, this paper proposes a resource scheduling algorithm for end users, which aims at solving the resource scheduling problem in the edge computing.

## 2. Related Work

As a more mature service platform, cloud computing plays an important role in dealing with big data. In addition, there are many cloud computing resource scheduling and resource allocation algorithms [15–23].

In 2016, HeeSeok Choi et al. [24] proposed a task consolidation algorithm based on task classification and resource utilization for the cloud data center. Furthermore, they designed a VM consolidation algorithm to balance task execution time and energy consumption without violating a predefined service level agreement (SLA).

In 2016, Wu et al. [25] proposed a scheduling scheme for continuous clustering of mobile cloud resources based on the improved FCM (IGAFCM) algorithm to reduce the size of matching requirements in the search process. In addition, the experiment proved that the matching strategy can be dynamically adjusted according to the matching score and feedback training.

In 2016, Cheol-Ho Hong et al. [26] proposed a new mechanism, called ANCS (Advanced Network Credit Scheduler), to guarantee QoS through dynamic allocation of network resources in virtualization. To meet the various network demands of cloud users, ANCS aims to concurrently provide multiple performance policies; these include weight-based proportional sharing, minimum bandwidth reservation, and maximum bandwidth limitation.

Edge computing is a new platform that can serve local mobile devices. Edge computing makes resource sharing or resource caching among widely deployed edge computing clusters. Next, we introduce some scholars about the work of edge computing in resource management.

In 2015, Su et al. [27] studied the resource caching scheme in edge computing based on the Steiner tree. When the edge computing server caches resources, the Steiner tree is first generated to minimize the total path weight (cost); this Steiner tree can minimize the cost of resource cache. Then the article gave a running example, and compared with the shortest path scheme, the results show that based on the Steiner tree scheme can work more effectively.

In 2015, Aazam et al. [28] proposed a dynamic resource allocation through the Fog Micro Datacenter, which argued that edge computing is between the Internet of Things and the cloud. Its purpose is to manage resources, to achieve data filtering, data preprocessing and data encryption. For this purpose, the paper proposed a resource management architecture in an edge computing environment. The article took into account the factors that give up the customer, the type of service, the service price, and the different waiver probability fluctuations. The experimental results show that these factors can help the service provider to correctly estimate the number of resources.

In 2015, Datta et al. [29] proposed edge computing and consumer-centric networking, which discussed the edge computing paradigm and how it served the consumer-centric IoT. The edge computing architecture can be integrated into an one M2M standard system.

The resource scheduling algorithm proposed in this paper is based on clustering method. Cluster analysis is a multivariate statistical analysis and a part of unsupervised pattern recognition [30–32]. Compared with k-means clustering algorithm, spectral clustering has significant advantages, it can identify any shape of the cluster, such as nonconvex clusters, and it is difficult to fall into the local optimal solution, so that the clustering algorithm has a certain degree of improvement compared with the traditional clustering algorithm. In 2015, Zhou et al. [33] proposed a spectral clustering algorithm to solve the problem of resource scheduling and allocation in cloud computing. However, it is not difficult to see from the implementation steps of the spectral clustering algorithm, and a k-means algorithm is required in the last step. Because the k-means algorithm is sensitive to the initial value, the spectral clustering algorithm is difficult to ensure the stability of clustering results. In this respect, Duan et al. [34] put forward an improved k-means algorithm, and the improved k-means algorithm uses the "shooting target" principle to search the clustering center.

## 3. Resource Scheduling Architecture

In the edge computing environment supported by the Internet of Things, there are two forms of service available to the end user: one is based on the amount of task and the size of the load, and the end user submits the task to the edge computing micro data center [35, 36]. The other is that the user will submit the resources to the edge computing micro data center, and the edge computing micro data center allocates the resource according to the request submitted by the end user [37]. There are two forms of service, the former we call task oriented, and the latter we call user oriented. The difference between these two forms of service is as follows. For task-oriented resource scheduling, there is no explicit requirement on the computing capability of resources. According to the submitted the amount of task load and task calculation, the edge computing micro data center allocates resources with corresponding processing capabilities to execute tasks. For user-oriented resource scheduling, the user submits resource allocation requests to the edge computing micro data center. The requests of users include explicit requirements on the computing capability of the resource and a clear resource occupancy time interval. In this paper, we consider the problem of resource scheduling and allocation in the edge computing according to the user-oriented service form.

In the framework of the Internet of Things, edge computing is between cloud computing and Internet of Things equipment [38, 39], the three levels shown in Figure 1. The edge computing preprocesses the collected data and then uploads it to the cloud data center; this can greatly reduce the pressure on the cloud data center.

In edge computing, the architecture of the resource scheduling process is shown in Figure 2, the user submits the request to the edge micro data center (EMDC), and the edge server allocates the resource according to the request of users. If the resource requested by the current user is temporarily short of the edge center (Edge1), the edge server can send broadcast messages to its adjacent edge center (Edge2). Edge1

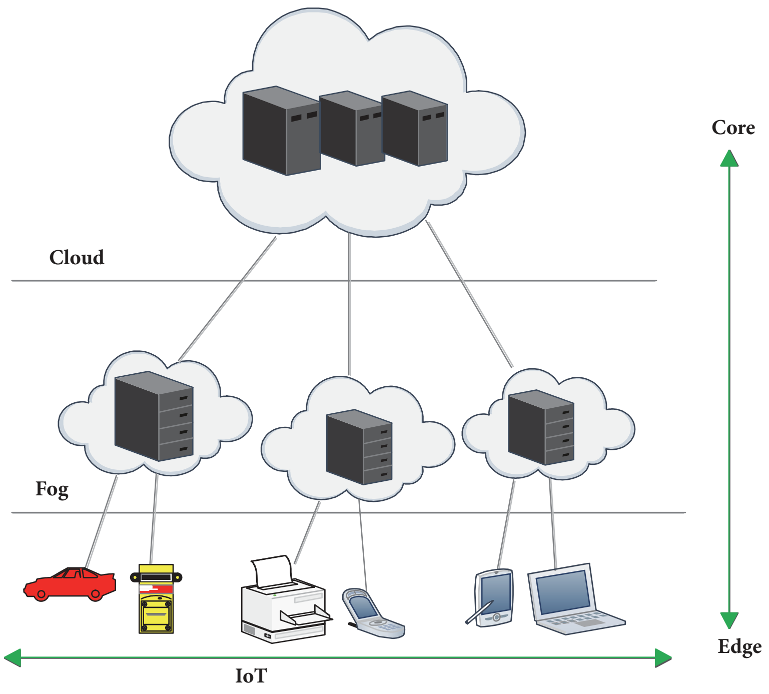

Figure 1: Edge computing between cloud and IoT.

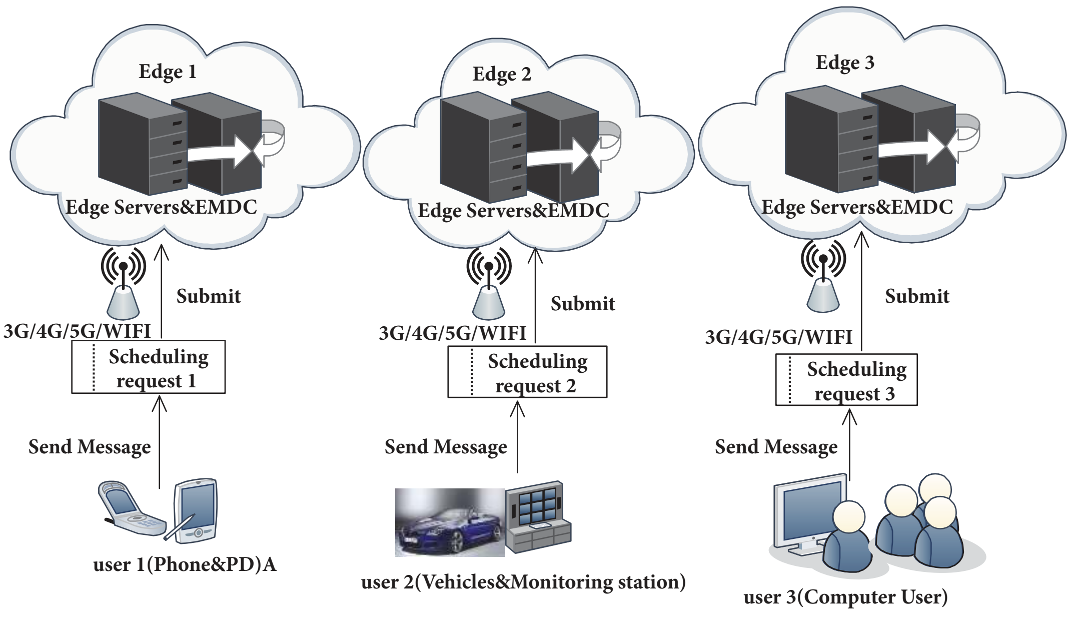

Figure 2: Edge computing resource scheduling architecture.

submits the request to Edge2 to process; if Edge2 does not have the resources that the user wants to request, then Edge2 can upload the user's request to its neighboring Edge3 to process. Edge servers in edge computing are mostly made up of weak performance computers, and the system resources are limited. When some users submit requests, if it exceeds the capability range of the edge computation processing data, then the edge computing will choose to transfer the data of users to the cloud server. In this paper, we study the resource scheduling problem in edge computing, so we do not consider the resource scheduling process after the data upload to the cloud.

## 4. Solving Scheduling Problems

*4.1. Mathematical Description of Scheduling Problem.* The request refers to the resource allocation request that the user submits to the edge computing micro data center. It mainly

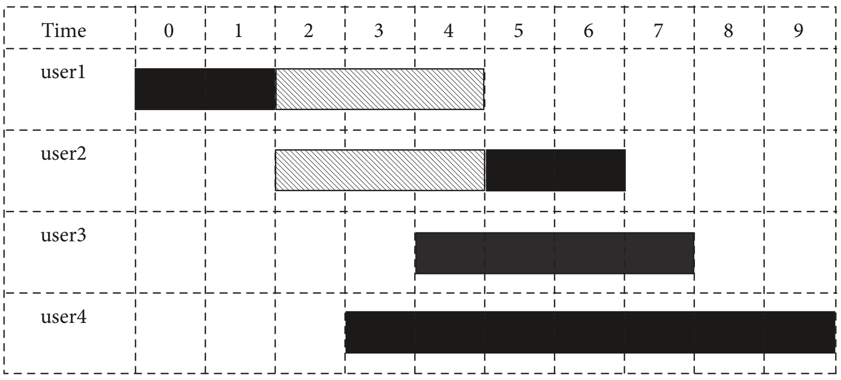

Figure 3: Example of overlapping degree.

includes the following aspects: computing resource capability (usually expressed by millions of instructions per second); time interval (including start time, completion time). In order to simplify the calculation process, the number of computing resources is used to represent the demand of users for computing resource capability.

The resource refers to the resources in the edge computing micro data center, such as computing resources, communication resources, and storage resources. When a user requests resource allocation to the edge computing, if the same computing unit is required by different users, there is an overlap in the use time interval of the resource, resulting in resource contention, reduced resource utilization, and service delay. In this paper, we call this overlap in time interval be overlap degree. The size of the overlap degree depends on the time interval of the user occupied the computing unit.

In this paper, the clustering algorithm solves the problem that reduce the overlap degree of user requests and improving resource utilization. The object of clustering is the requests of users. Clustering is based on the degree of time overlap between requests. The greater the overlap degree is, the higher the similarity is. By clustering, requests with a big degree overlap can be in the same category, and requests with a small degree of overlap can be assigned to different categories. During scheduling, resources can be allocated to different categories of requests at the same time to reduce the possibility of resource contention and improve service quality.

Each request of the user has a clear resource occupancy time interval (including the start time and completion time), and different user requests may overlap on the time interval. The overlap degree between all users constitutes a matrix, represented by $M$.

The matrix $M$ has $n*n$ elements, each of which can be defined as follows:

$$M_{ij} = \frac{P_{ij}}{T} \tag{1}$$

In formula (1), $P_{ij}$ is the overlap time between user i and user j. The time interval T is delimited by the edge computing, and there are multiple requests arriving at this time interval. As shown in Figure 3, the time interval T is from 0 to 9, for a total of 10 time intervals, T=10. The earliest start request is user1, and the latest finish request is user4. The user1 request time interval is $[0,4]$, and user4 request time interval is $[3,9]$. The degree of time overlap exists if and only if two users request the same resource. As shaded by the slash lines, when user1 and user2 request the same resource, there is an overlap between the time intervals of the two users. Therefore, according to formula (1), the overlap degree between the user 1 and the user 2 is $3/10 = 0.3$.

Assume at time 0 edge computing micro data center receives the $N$ service request, the amount of resources available is $m$, and the calculation ability of a single resource is constant, so we believe that there is no difference between the two independent computing units (nodes) on the ability. Different users have different requirements for computing capacity. According to the needs of users, multiple resources can be assigned to a user at the same time.

Let $S_r = \{s_i \mid s_i \in S, s_i \longrightarrow r\}$ be the user set that would be executed on computing unit $r$; set $S$ refers to the set of all users. For two different units $r_1$ and $r_2$, a user $S_i$ may apply for the use of two computing units $r_1$ and $r_2$, so the intersection between the sets $S_{r1}$ and $S_{r2}$ is not empty. In fact, because a request can take up multiple computing units, which adds to the complexity of the problem. So, we need to divide the requests that take up multiple units. If the request of user needs to take up $a$ units, it should be divided into $a$ requests that have the same time interval with the previous request. So the request set $S = \{s_1, s_2, \ldots, s_n\}$ can be extended to set $\widehat{S} = \{s_1, s_2, \ldots, s_n, s_{n+1}, \ldots, s_{n+A}\}$ where $A = \sum_{i=1}^{n} a(s_i) - 1$ and $a(s_i)$ is the number of units required to represent the i$^{\text{th}}$ request.

We suppose that the service requests set $S$ is generated randomly, $S = \{S_1, S_2, \ldots, S_n\}$, $n$=30. Suppose the 30 requests are divided into 6 groups in turn; the number of units occupied by each group is represented by a set G=$\{G_1, G_2, G_3, G_4, G_5, G_6\}$ where $G_1$=3, $G_2$=2, $G_3$=5, $G_4$=1, $G_5$=2, $G_6$=1. According to the formula $A = \sum_{i=1}^{n} a(s_i) - 1$, so that the request set $S = \{s_1, s_2, \ldots, s_n\}$ can be extended to set $\widehat{S} = \{s_1, s_2, \ldots, s_n, s_{n+1}, \ldots, s_{n+A}\}$, we get $A = 3\times5+2\times5+5\times5+5+2\times5+5-1 = 69$, so the set $\widehat{S} = \{S_1, S_2, \ldots, S_{30}, S_{31}, \ldots, S_{99}\}$.

With regard to the process of resource allocation, the number of computing units in operation is $k$, and $k$ has the minimum value; that is, there is a boundary beyond the marginal calculation of micro data center services. Obviously, this minimum corresponds to the maximum value required by all users for computing power. Let $k_{\min}$ be the minimum value, and we know that $k_{\min} = \max\{a_s \mid s \in S\}$ from a previous analysis. For example, if the requirement for computing power is five units, the edge computing micro data center does not assign four or fewer units to the user. At the same time, the number of computational units in the operation also has the maximum value. Obviously, the maximum will not exceed the number of available computing units. Let $k_{\max}$ be maximum value and $k_{\max} \leq m$. Obviously, $k \in [k_{\min}, k_{\max}]$.

*4.2. Scheduling Objectives.* In edge computing, resource utilization and quality of service (QoS) should be taken into account in resource scheduling for end users. These are very important reference factors for resource scheduling [40–43]. Resource utilization affects economic benefit, and QoS affects customer satisfaction.

There are two users service request $s_i$ and $s_j$; $p_{ij}$ means overlap degree. We consider the simplification of standardization and computation, let $c_{ij} = 1 - \beta(1 - p_{ij})$, where $\beta$ is a constant and $0 < \beta < 1$. From this, we can see that the higher the overlap degree is, the greater $c_{ij}$ is.

Therefore, one of the objectives of scheduling can be expressed in this way, and a strategy satisfying the following formula is found:

$$\min \sum_{r=1}^{m} \sum_{i \in S_r, j \in S_r} c_{ij} \tag{2}$$

By formula (2), we can get the most satisfying policy for the user. However, it does not consider the optimization problem of resources, which is also one of the objectives of scheduling. For example, when the user requirements are not demanding on the computing ability, the edge computing micro data center will dispatch a small number of computing units to meet the request of users, and the resource utilization can be improved by reducing the number of computing units.

The length of the known time interval is $T$, and for each computing unit $r$, the resource utilization rate can be expressed as

$$U = \sum_{r=1}^{k} \frac{t_r}{kT} \tag{3}$$

From the perspective of the user, the user request is including the use time of resources and number of resources, which means that time of $T$ and the number of each user application resources are fixed. Therefore, the value $\sum_{r=1}^{k} t_r$ is constant, nothing to do with the value of $k$. According to formula (3), the size of resource utilization is determined by $k$. When the demand of users is responded, in order to improve resource utilization, we have to reduce the number of resources in the operation. However, reducing the amount of resources that can be put into operation will inevitably increase the overlap degree. This creates a problem that the system must make a choice between the number of resources and the overlap degree.

As mentioned earlier, overlap affects the quality of service (QoS) and reduces user satisfaction. As we all know, there are many factors that affect the quality of service (QoS). This paper focuses on the resource scheduling and allocation problem. In this paper, QoS is expressed as delay extent, actual finish time compared to the expected completion time of services. The QoS condition is defined as the following representation:

$$\delta = \frac{E}{T} \tag{4}$$

In summary, under the premise of satisfying the QoS condition, the goal of our resource scheduling can be expressed as

$$\begin{gathered} \min \sum_{r=1}^{m} \sum_{i \in s_r, j \in s_r} c_{ij} \\ \max \sum_{r=1}^{k} \frac{t_r}{kT} \end{gathered} \tag{5}$$

*4.3. Solutions.* As mentioned earlier, our goal is to find a formula to satisfy formula (5). If we regard the overlap degree as Euclidean distance, then large-scale requests should form a cluster structure, and the distance between different clusters is very large; in addition, the overlap degree between different clusters is very small. However, the smaller distance between nodes in the cluster is, the bigger overlap degree between nodes in the cluster is. This means that we can turn the scheduling problem into a clustering problem.

The detailed process of converting a scheduling problem to a clustering problem is as follows.

Firstly, a set $S$ is set up to store the requests of users, and the edge computing allocates resources required by the user.

Secondly, the overlap degree between each two requests in the set $S$ is calculated according to formula (1). It should be noted that a certain user request may require multiple resources and overlap with other requests for multiple times. In this case, the request needs to be divided into multiple subrequests. The set $S$ is expanded into a set $\widehat{S}$, and the Euclidean distance between the requests is calculated to construct a matrix of overlap degree.

Finally, the improved spectral clustering algorithm is used to solve the clustering problem. The overlap degree matrix is regarded as a similarity matrix, and the spectral clustering algorithm reduces the dimensions of the matrix. The improved k-means is used to complete the clustering.

According to the final clustering result, in order to facilitate the search for a better scheduling strategy, the requests located in different clusters are selected, and a set of scheduling strategies needs to be established to store the selected requests. Obviously, the overlap degree between requests within the set of scheduling policies is very small, and the overlap degree between requests from different scheduling sets is large.

*4.3.1. Improved k-Means Algorithm.* In order to make the clustering result that insensitive to the initial value, in this paper, we use the improved k-means algorithm proposed by Duan et al. and then combined it with the spectral clustering algorithm to achieve a relatively stable clustering results.

The improved k-means algorithm (referred to as myk-means) mainly adds the initial clustering center selection algorithm. The initial number of cluster centers is $k$, the number of searches is $C$, and the set $B$ is used to store $C*k$ cluster centers, $B = \{b_1, b_2, \ldots b_{C*K}\}$. With the algorithm loop search clustering center, each time the selection results are put into $H_i$, the set $H$ stores the final cluster center selection results, $H = \min\{distortion(H_i, B)\}$.

In the clustering algorithm, since the initial clustering center selection algorithm is added. When selecting the optimal initial center of the next cluster, the similarity degree matrix (the distance between all points in the dataset) of the entire data set needs to be calculated at each step, which makes the algorithm cost time is very large or needs to store the calculated similarity matrix back up, which in turn requires a lot of storage space. In short, when the algorithm is applied to a data set with a slightly larger amount of data, there is a problem that the calculation time is too long, or the storage space is insufficient, and it is not suitable for large-scale systems with many nodes.

In order to solve the above problems and reduce the computational complexity of the clustering algorithm, in this section we use the improved clustering algorithm proposed by Zhao [44].

When we select the cluster center of the next cluster, if the selected point is in close proximity to the found cluster center, then there will be very little difference in the center values of the two clusters in the final clustering result. Obviously, this is meaningless, so a point within a certain range around the found cluster center may not be the cluster center of the next cluster. When selecting the best initial center of the next cluster, these points need to be excluded first. The specific implementation process is as follows.

If we know the final result $(m_1, m_2, \ldots m_{k-1})$ of the clustering problem of (k-1) clusters, we get all the clusters $(c_1, c_2, \ldots c_{k-1})$ with $(m_1, m_2, \ldots m_{k-1})$ as the cluster center. First we calculate the average distance $r_t$ between all data points in each cluster and the cluster center, where $t = 1, 2, \ldots, k-1$.

$$r_t = \frac{\sum_{x_j \in C_t} \left\| m_t - x_j \right\|^2}{n_t} \tag{6}$$

where $n_t$ is the number of all data points in $C_t$.

We refer to the parameter $\alpha_t = (\max_{x_j \in C_t}(\|m_t - x_j\|^2))/r_t$ of [44], obviously $\alpha_t > 1$. Let $\beta_t = \varepsilon(\alpha_t - 1)$, where $\varepsilon$ is a sufficiently small positive number. Then we introduce a parameter $\lambda_t = 1 + \beta_t * (k-1)$; obviously $\lambda_t \geq 1$, $(t = 1, 2, \ldots, k-1)$.

All points in the area with $r = r_t \bullet \lambda_t$, $(t = 1, 2, \ldots, k-1)$ as the radius centered on $m_t$ are removed, and the initial cluster center of the next cluster is selected from the remaining points, which will effectively reduce the number of data points that need to be calculated. This reduces the amount of calculation. In the incremental method, as the $k$ increases, the structure of the obtained cluster is also more and more stable, so the value of the parameter $\lambda_t$ can be increased accordingly to exclude more points.

*4.3.2. Spectral Clustering Algorithm.* In the spectral clustering algorithm, we first need to solve the similarity matrix, in which we define the overlap degree matrix P as the similarity matrix. For arbitrary $\mathrm{P_{i,j}} = \mathrm{P_{j,i}}$ and $\mathrm{P_{i,i}} = 0$; that is, diagonal element is 0. Then the overlapping degree matrix can be expressed as

$$P = \begin{bmatrix} 0 & p_{1,2} & \cdots & \cdots & p_{1,n} \\ p_{2,1} & 0 & \cdots & \cdots & p_{2,n} \\ \vdots & \cdots & 0 & \cdots & \vdots \\ \vdots & \cdots & \cdots & 0 & \vdots \\ p_{n,1} & \cdots & \cdots & p_{n,n-1} & 0 \end{bmatrix} \tag{7}$$

where $p_{i,j} = \exp(-\|s_{i-}s_j\|^2/2\sigma^2)$. Then we can calculate the normalized matrix $D(i,i) = \sum_{j=1}^{n} p_{i,j}$, and Laplacian matrix $L = D - P$.

We can assume that all the requests are divided into $k_1$, $k_2$, two categories, for the classification criteria; this paper uses the standard division (NORMALIZED CUT) method. Let $q$ be a vector. The elements $q_i$ are defined as follows:

$$q_i = \begin{cases} \sqrt{\dfrac{d_2}{d_1 d}} & , i \in k_1 \\ -\sqrt{\dfrac{d_1}{d_2 d}} & , i \in k_2 \end{cases} \tag{8}$$

Among them, $D_1$ is the sum of all overlaps in $k_1$, plus *Cut* $(k_1, k_2)$, $D_2$ is the sum of all overlaps in $k_2$, plus *Cut* $(k_1, k_2)$, and $d = d_1 + d_2$ - $\mathrm{Cut}(G_1, G_2)$.

Then $N_{cut}(k_1, k_2) = q^T L q$; the restrictions are

$$q^T D q = \sum_{i \in k_1} q_i^2 D_{ii} + \sum_{j \in k_2} q_j^2 D_{jj} = \left(\sqrt{\frac{d_2}{d_1 d}}\right)^2 d_1 + \left(\sqrt{\frac{d_1}{d_2 d}}\right)^2 d_2 = 1$$

$$L(q, \lambda) = Ncut(G_1, G_2) - \lambda\left(q^T D q - 1\right) = q^T L q - \lambda\left(q^T D q - 1\right)$$

$$\frac{\partial L(q, \lambda)}{\partial q} = 0 \Longrightarrow$$

$$Lq = \lambda D q$$

$$\text{Let } \mathrm{L}' = \mathrm{D}^{-1/2}\mathrm{L}\mathrm{D}^{-1/2},\ \mathrm{q}' = D^{-1/2}q$$
$$\therefore D \text{ is diagonal matrix}$$
$$\mathrm{D}^{1/2}\mathrm{D}^{1/2} = 1 \Longrightarrow$$
$$\mathrm{L}'q' = \lambda \mathrm{q} \tag{9}$$

At this time, the Laplacian matrix for the normalized (Normalized Laplacian, the diagonal elements are all 1) $\mathrm{L}'$= D-1/2 L D-1/2 eigenvalues and the corresponding eigenvectors. Because the eigenvalues of L and $\mathrm{L}'$ are the same, the relation between the eigenvectors is $\mathrm{q}' = \mathrm{D}^{1/2}\mathrm{q}$, so we can get the eigenvectors corresponding to the eigenvalues of $\mathrm{L}'$, and finally we can get the $q$ by multiplying $\mathrm{D}^{-1/2}$.

*4.3.3. Improved Spectral Clustering Algorithm (ISCM).* Based on the improved k-means algorithm, we propose an improved spectral clustering algorithm (ISCM). The algorithm not only solves the problem of initial value, but also realizes the secondary clustering. As described above, according to the QoS condition, we consider the parameters $\delta$. When the algorithm is executed, we can through judge the size of the parameters whether it needs secondary clustering. If the current QoS satisfies the demand of users for resource scheduling after the algorithm is executed, there is no need to recluster. Next, the implementation steps of spectral clustering optimization scheduling algorithm are introduced.

The detailed steps of the scheduling algorithm are as follows:

(1) Transforming a set $S = \{s_1, s_2, \ldots, s_n\}$ into a set $\widehat{S} = \{s_1, s_2, \ldots, s_n, s_{n+1}, \ldots, s_{n+A}\}$.

(2) According to the overlap degree between the elements in the set $\widehat{\mathrm{S}}$, the overlap degree matrix P is calculated, and the value of $c_{ij}$ is calculated according to the formula $c_{ij} = 1 - \beta(1 - p_{ij})$.

(3) Constructing Laplacian matrix L and computing eigenvalues of matrix L.

(4) Constructing matrix R by eigenvectors corresponding to eigenvalues.

(5) Computing $k_{\max}$ and $k_{\min}$.

(6) For $k \in [k_{\min,} k_{\max}]$.

Using improved k-means method to cluster row vectors of matrix R.

Computing delay E, QoS conditions $\delta'$, and system resource utilization U, according to formula (3): $U = \sum_{r=1}^{k} t_r / kT$.

(7) If $\delta' \leq \delta$, the loop is stopped; otherwise, go back to the previous step and restart.

(8) The scheduling strategy can be obtained according to k cluster.

The flowchart of the ISCM algorithm is shown in Figure 4.

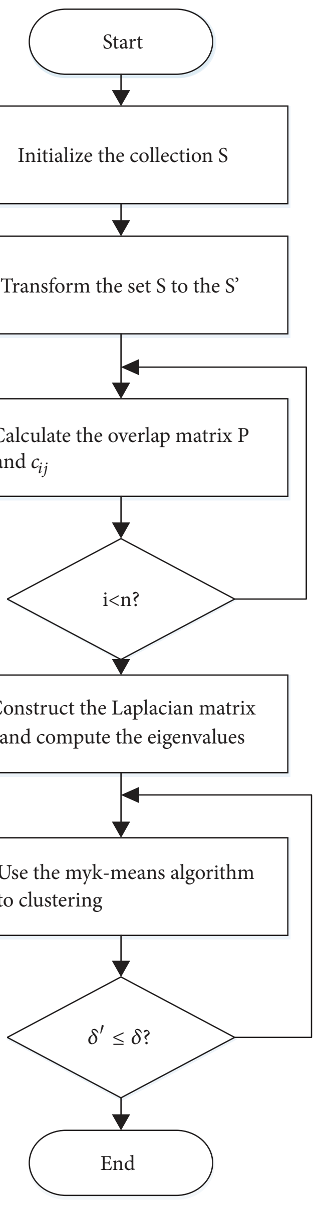
Start
Initialize the collection S
Transform the set S to the S'
Calculate the overlap matrix P and $c_{ij}$
i<n?
Construct the Laplacian matrix and compute the eigenvalues
Use the myk-means algorithm to clustering
$\delta' \leq \delta$?
End

FIGURE 4: Flowchart of ISCM algorithm.

### 4.4. Resource Scheduling

*4.4.1. Scheduling Process.* In the resource scheduling process of edge computing, as shown in Figure 5, the end user submits the service request to the edge micro data center (EMDC) through the edge computing. The resource monitor is set up in EMDC, and the dynamic resource monitor will detect the status information of the available resources at any moment and save the status information of these resources in EMDC. The ISCM algorithm clusters the service request sets, sorts them into several different sets of requests, and then schedules the resources. Finally, the resource allocation of these requests is arranged.

*4.4.2. Pseudocode of ISCM Algorithm.* Through the above steps, the resource scheduling problem is transformed into a

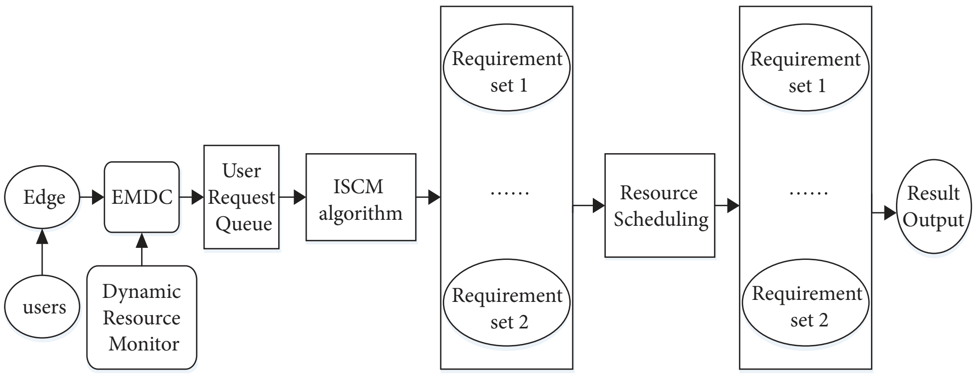

Figure 5: Resource clustering and scheduling process.

clustering problem, and the resource scheduling scheme can be obtained according to the clustering results. ISCM algorithm can effectively solve the resource scheduling problem in edge computing.

The pseudocode of ISCM algorithm is as in Algorithm 1.

*4.5. Algorithm Correctness and Performance Analysis.* The algorithm consists of three stages; the first stage calculates the overlap degree between the user service request sets; each service request contains the start time and end time; the overlapping degree matrix can be obtained according to the overlap in time interval. In the second stage, the overlap degree matrix is diagonalized, and the Laplacian matrix is established by using the normal cutting method, and the eigenvector is obtained. In the third stage, we use the myk-means algorithm to cluster, and on the basis of the original k-means algorithm, the initial clustering center selection algorithm is added, repeatedly traversing the search data samples to find $k$ best initial clustering centers. Then, according to the initial cluster centers, the $k$ cluster is obtained to form scheduling strategy. Throughout the above three stages, the degree of time overlap between users is regarded as the similarity between data samples, and the scheduling problem is transformed into clustering problem. We classify the large overlap degree requests into one class and divide them into different classes with small overlap degree. The clustering algorithm can make the similarity among samples the largest, and the similarity between samples is minimum, so our algorithm is correct.

Theorem 4-1 ISCM algorithm can complete the resource scheduling in edge computing; the time complexity is $O(nck)$. $n$ represents the number of data samples, $c$ represents the number of times that the initial cluster center searches, and $k$ represents the number of clusters.

In large-scale datasets, the number of searches $c$ is much less than the number of samples, so the time complexity of the proposed algorithm avoids sinking into worst-case $O(n^2)$.

According to the common performance metrics of clustering algorithm, as shown in Table 1, this paper compares with ISCM algorithm, traditional spectral clustering algorithm, and k-means algorithm.

```
(1) Input : K- > the total number of clustering
            U- > matrix of user data
(2) Output : Array (results)
(3) $\delta = 1$
//user request set S,
//$S' = \{S_1, S_2, S_3, \ldots S_n, S_{n+1}, \ldots S_{n+A}\}$
(4) $S'$ = transform(S)
(5) for i = 1, i <= $S'$.length, i + +
(6) $P_{i,j} = \exp(-\|s_{i-}s_j\|^2/2\sigma^2)$
(7) $C_{i,j} = 1 - \beta(1 - P_{i,j})$
(8) $D(i,i) = \sum_{j=1}^{n} C_{i,j}$
(9) L = D - C
(10) $L' = D^{-1/2}LD^{-1/2}$
(11) $q' = D^{1/2}q$
(12) $q = D^{-1/2}q'$ //feature vector
(13) Array Q.append(q)
(14) endfor
//Using SearchCenter algorithm
//compute classification center c
(15) SearchCenter(U, K) $\longrightarrow$ initcenter[c]
(16) Array results = myKMeans(c, Q, k)
(17) $\delta' = E/T$ //QoS condition
(18) *While* $\delta' > \delta$ do
(19) SearchCenter(U, K) $\longrightarrow$ initcenter[c]
(20) Array results = myKMeans(c, Q, k)
(21) endwhile
(22) return results
```

Algorithm 1: ISCM cluster algorithm.

## 5. Simulation Results

Our simulation experiments are based on the MATLAB platform, assuming the known service quality (QoS) conditions $\delta$ =0.1, $T$=60, $m$=5 (which means that each user can apply for *5* computing units at most). The busy time of the *5* computing units in $T$ is denoted as $T_1$, $T_2$, $T_3$, $T_4$, and $T_5$, and their values, respectively, are *35*, *20*, *30*, *39*, and *53*.

According to the starting time and completion time of each request, denoted as $t_s$ and $t_F$, and the number of units that need to be occupied, represented by the letter $a$, the request is represented as data point in three-dimensional

Table 1: Performance comparison of algorithms.

| Performance metrics | Algorithm | | |
|---|---|---|---|
| | ISCM algorithm | Traditional spectral clustering algorithm | k-means algorithm |
| Extendibility | High | High | High |
| Data types that can be processed | Numerical and non-numerical type | Numerical and non-numerical type | Numerical type |
| The clustering shape can be found | Arbitrary | Arbitrary | Convex and spherical type |
| Domain knowledge dependence on input parameter | Small | Large | Large |
| Sensitivity to noise | Sensitive | Sensitive | Sensitive |
| Sensitivity to input order | General | General | Sensitive |
| The ability to process high dimensional data | High | High | Lower |
| Time complexity | O(nck) | O(knt) | O(knt) |

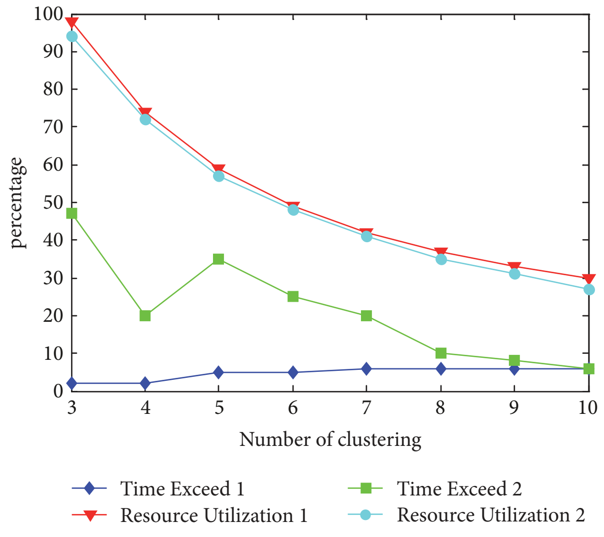

Figure 6: Resource utilization and time exceed curves.

coordinates. As shown in Figure 6, the number of clusters is the number of resources; the time exceed curve *1* and resource utilization rate *1* are obtained by ISCM algorithm, which are compared with the time exceed curve *2* and resource utilization rate *2* of traditional spectral clustering algorithm. By computing formula (3), we can see that when the number of clusters is $k=3$, the resource utilization rate of ISCM algorithm is $U=98\%$, and the resource utilization rate of traditional spectral clustering algorithm is $U=94\%$. The ISCM algorithm is *4%* higher than traditional spectral clustering in resource utilization.

When the expected completion time is *60s* and the number of clusters is at least *3*, the running time of the ISCM algorithm is *1.9111s*, and the QoS condition is guaranteed. It can be seen from Figure 6 that the time exceed curve of traditional spectral clustering algorithm fluctuates up and down; this is because the original k-means algorithm is sensitive to the initial value, resulting in a drastic change in the delay degree of the traditional spectral clustering algorithm. The time exceed curve of ISCM algorithm is close to stability and shows low delay characteristic.

According to the diversity of big data in actual situation, we choose two kinds of data sample set. For the first sample set, we use k-means algorithm and myk-means algorithm, respectively. For the second sample set, we use the traditional spectral clustering algorithm and ISCM algorithm, respectively; at the same time, it is compared with k-means algorithm.

When using the traditional k-means clustering algorithm, the experimental results are shown in Figures 7 and 8; the $X$ axis represents the start time of the service request, the $Y$ axis represents the finish time of the service request, and the $Z$ axis represents the number of computing units required for the service request. Here we set up *5* computing units at most that users can apply. It is not difficult to find from Figure 8 that the same color is the same class, the traditional k-means algorithm clustering results are wrong, and this is because the clustering results depend on the initial clustering center.

Figure 9 shows the results of the algorithm using myk-means. As shown in Figure 9, the same color is as the same class, the time overlap between the same classes is large, and users have to wait for resources until the resource is idle, so it reduces customer satisfaction easily. Different types of requests can respond simultaneously, because there is no time overlap, and edge computing allocates resources to them simultaneously, which can avoid resource contention, thus it forms scheduling policy. Different categories of requests have different colors; moreover, different categories of requests are combined into a scheduling set in the scheduling process, which can reduce the overlap degree and service delay and improve QoS and resource utilization.

From the comparison of two algorithms under the experimental results, we can conclude that myk-means clustering algorithm used in this paper is superior to the traditional k-means algorithm, myk-means clustering algorithm selects the appropriate initial cluster center, and it can converge to a better local optimal solution and obtain more accurate resource scheduling.

In Figure 10, the blue line represents the running time of traditional spectral clustering algorithm, and the red line represents the running time of ISCM algorithm. As can be seen from Figure 10, by improving the spectral clustering algorithm and reducing the amount of data calculation, the

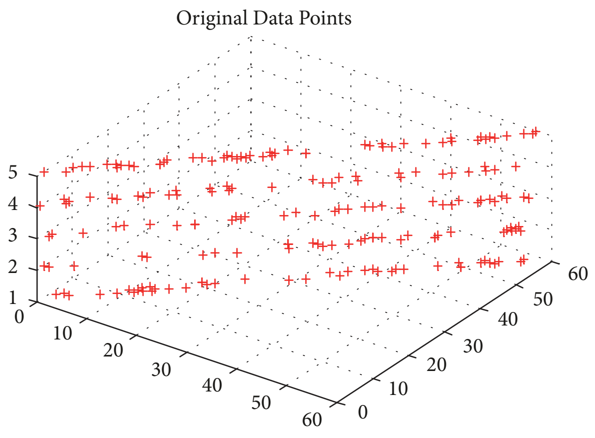

FIGURE 7: Initial data samples.

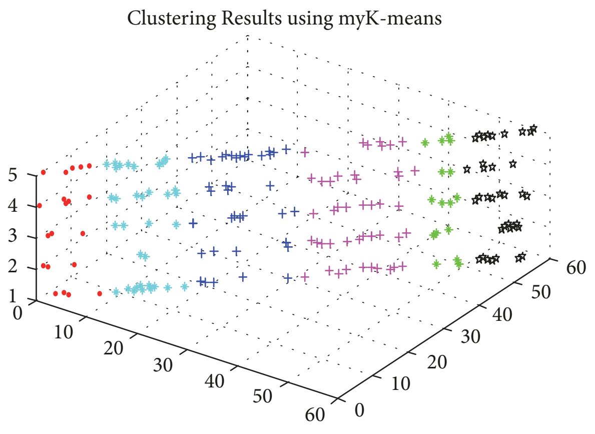

FIGURE 9: myk-means algorithm clustering results.

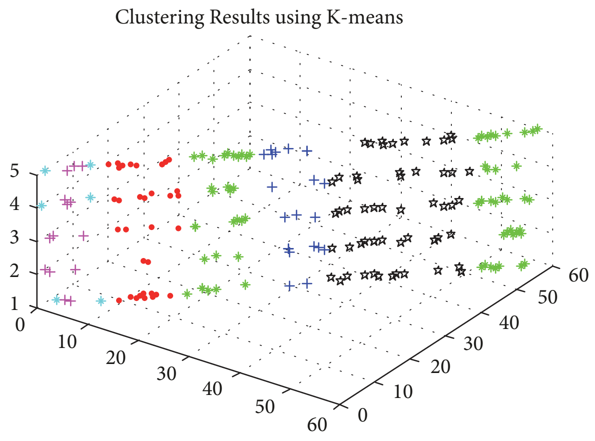

FIGURE 8: k-means algorithm clustering results.

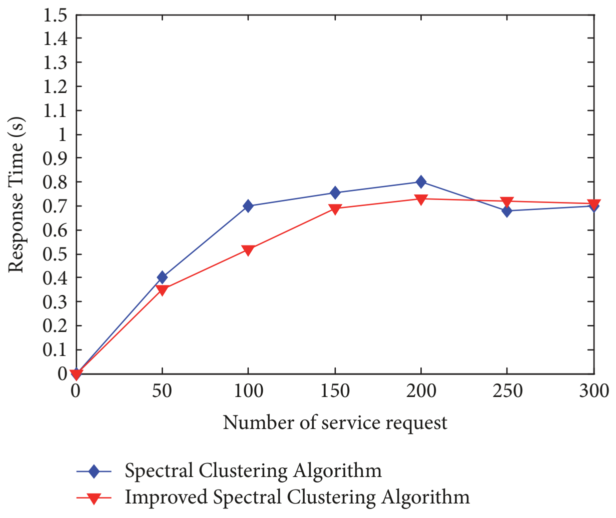

FIGURE 10: Response time comparison of algorithms.

execution efficiency of the ISCM algorithm can be effectively improved. In addition, as the scale of requests continues to increase, the time-growth trend of the ISCM algorithm is slower than the traditional spectral clustering algorithm.

In order to show that the ISCM algorithm is better than k-means algorithm and traditional spectral clustering algorithm in clustering accuracy and applicability scope, the experimental results obtained in this paper are shown in Figures 11 and 12.

In Figure 11, we can see that the clustering results of traditional spectral clustering algorithm are not stable, due to the clustering results depending on the selection of the initial clustering center; when poor initial clustering center is selected, the clustering of traditional spectral clustering algorithm to produce the result is not stable.

In Figure 12, the traditional k-means algorithm can not solve the clustering problem, and our algorithm can get stable solution to the problem of clustering. In addition, when the QoS conditions do not meet the needs of users, the proposed algorithm will be reclustering until meeting the user needs. In summary, the algorithm of this paper is superior to the traditional k-means algorithm and traditional spectral clustering algorithm in solving the problem of edge computing resource scheduling. Comparing with Figure 11, it is not difficult to find that the clustering results obtained by ISCM algorithm are ideal.

## 6. Conclusions

In this paper, we analyze the user-oriented resource scheduling problem in the edge computing of the Internet of Things and propose the ISCM algorithm to solve the scheduling problem in edge computing. In edge computing, users can simultaneously apply a variety of resources, which results in the overlap in time interval and increases the difficulty of resource scheduling problem. Firstly, we transform the problem of overlap degree in resource scheduling into clustering problem. Secondly, we carry on the overlap degree matrix transformation and the eigenvector clustering. Finally carry out resource scheduling for the categories with different colors. Experiments show that the scheduling algorithm can stabilize the clustering results under the guaranteed QoS condition, and it can obtain the higher resource utilization rate. In this problem, we assume that each computing unit

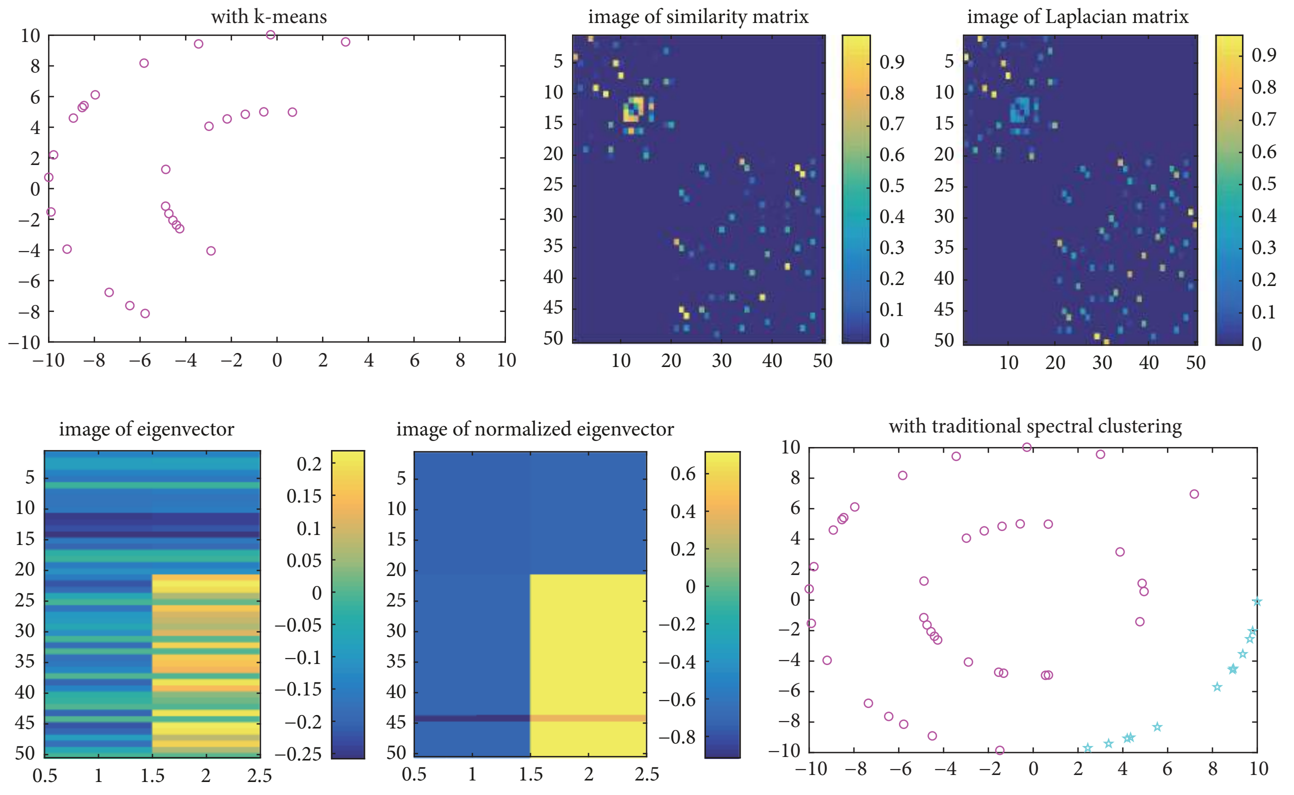

Figure 11: Clustering results comparison of traditional spectral clustering algorithm.

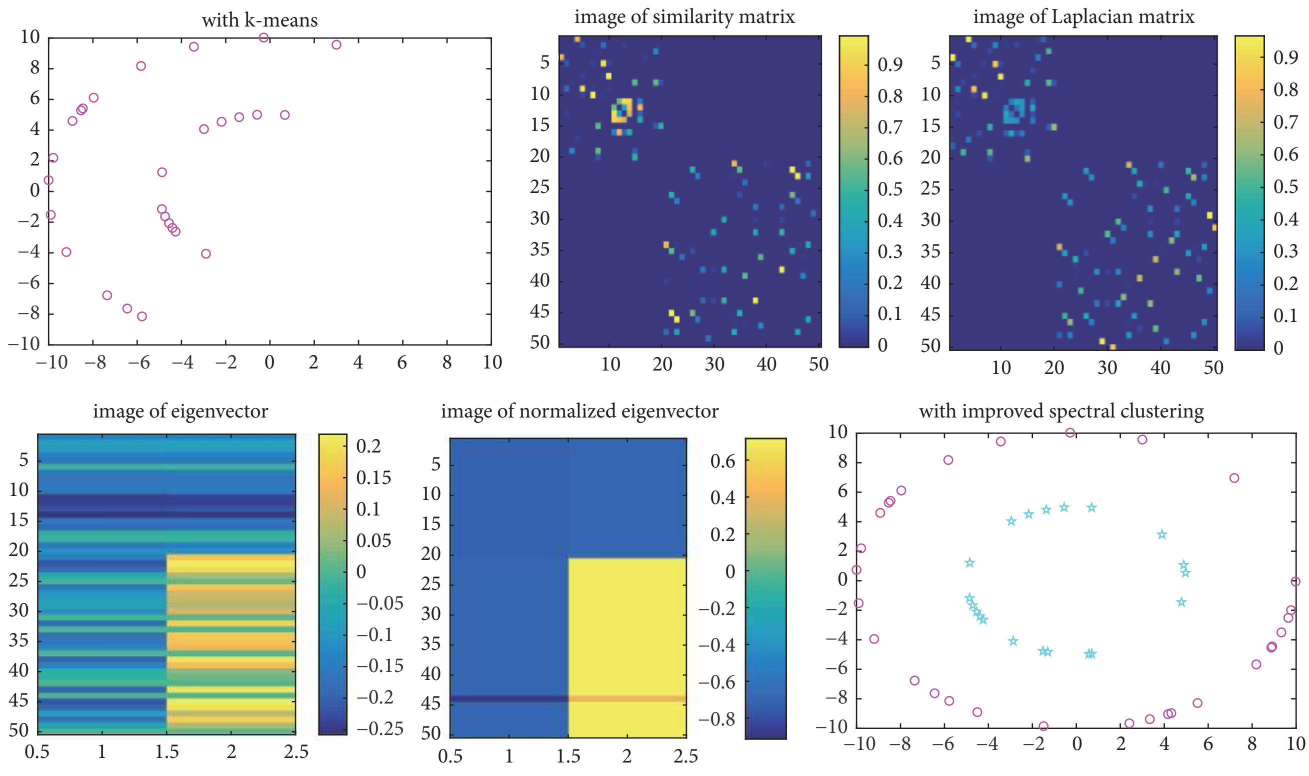

Figure 12: Clustering results comparison of ISCM algorithm.

has the same calculation ability, and the future research work should focus on the difference between the calculation ability of different computing units.

## Conflicts of Interest

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

## Acknowledgments

This work is supported by the National Natural Science Foundation of China (61672321 and 61771289), the Shandong Provincial Graduate Education Innovation Program (SDYY14052 and SDYY15049), Qufu Normal University Science and Technology Project (xkj201525), Shandong Province University Science and Technology Planning Project (J16LN15), the Shandong Provincial Specialized Degree Postgraduate Teaching Case Library Construction Program, and the Shandong Provincial Postgraduate Education Quality Curriculum Construction Program.

## References

[1] X. Ma, S. Zhang, W. Li, P. Zhang, C. Lin, and X. Shen, "Cost-efficient workload scheduling in Cloud Assisted Mobile Edge Computing," in *Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Vilanova i la Geltrú, Spain, June 2017.

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

[3] B. Blanco, I. Taboada, J. O. Fajardo, and F. Liberal, "A robust optimization based energy-aware virtual network function placement proposal for small cell 5G networks with mobile edge computing capabilities," *Mobile Information Systems*, vol. 2017, 2017.

[4] D. Choi, J. Jung, H. Kang, and S. Koh, "Cluster-based CoAP for message queueing in Intemet-of-Things networks," in *Proceedings of the 2017 19th International Conference on Advanced Communication Technology (ICACT)*, pp. 584–588, Pyeongchang, Kwangwoon Do, South Korea, Feburary 2017.

[5] T. Song, R. Li, B. Mei, J. Yu, X. Xing, and X. Cheng, "A Privacy Preserving Communication Protocol for IoT Applications in Smart Homes," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1844–1852, 2017.

[6] J. Yu, L. Feng, L. Jia, X. Gu, and D. Yu, "A local energy consumption Prediction-Based clustering protocol for wireless sensor networks," *Sensors*, vol. 14, no. 12, pp. 23017–23040, 2014.

[7] Z. Jianming, L. Fan, and L. Qiuyuan, "Resource management technique based on lightweight and compressed sensing for mobile internet of things," *Journal of Sensors*, vol. 2014, 2014.

[8] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, "SmartCityWare: A service-oriented middleware for cloud and fog enabled smart city services," *IEEE Access*, vol. 5, pp. 17576–17588, 2017.

[9] A. Abdelgawad and K. Yelamarthi, "Internet of things (IoT) platform for structure health monitoring," *Wireless Communications and Mobile Computing*, vol. 2017, Article ID 6560797, 2017.

[10] K. C. Okafor, I. E. Achumba, G. A. Chukwudebe, and G. C. Ononiwu, "Leveraging Fog Computing for Scalable IoT Datacenter Using Spine-Leaf Network Topology," *Journal of Electrical and Computer Engineering*, vol. 2017, 2017.

[11] W. Xiao, L. Guoqi, and L. Bin, "Research on big data integration based on Karma modeling," in *Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 245–248, Beijing, November 2017.

[12] Y. Leng, Z. Chen, and Y. Hu, "STLIS: A Scalable Two-Level Index Scheme for Big Data in IoT," *Mobile Information Systems*, vol. 2016, 2016.

[13] V. Sharma, J. D. Lim, J. N. Kim, and I. You, "SACA: Self-Aware Communication Architecture for IoT Using Mobile Fog Servers," *Mobile Information Systems*, vol. 2017, 2017.

[14] T. Zhao, S. Zhou, X. Guo, and Z. Niu, "Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing," in *Proceedings of the 2017 IEEE International Conference on Communications, ICC 2017*, fra, May 2017.

[15] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, AINA2010*, pp. 400–407, Australia, April 2010.

[16] J. Hu, J. Gu, G. Sun, and T. Zhao, "A scheduling strategy on load balancing of virtual machine resources in cloud computing environment," in *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming (PAAP '10)*, pp. 89–96, Dalian, China, December 2010.

[17] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol. 47, no. 4, article 63, 2015.

[18] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[19] L. Zhu, Q. Li, and L. He, "Study on Cloud Computing Resource Scheduling Strategy Based on the Ant Colony Optimization Algorithm," *International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 1–5, 2012.

[20] D. Ergu, G. Kou, Y. Peng, Y. Shi, and Y. Shi, "The analytic hierarchy process: Task scheduling and resource allocation in cloud computing environment," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 835–848, 2013.

[21] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *Journal of Cloud Computing*, vol. 7, no. 1, 2018.

[22] S. Loganathan and S. Mukherjee, "Job Scheduling with Efficient Resource Monitoring in Cloud Datacenter," *The Scientific World Journal*, vol. 2015, 2015.

[23] C. Zhijia, Z. Yuanchang, D. Yanqiang, and F. Shaochong, "A Dynamic Resource Scheduling Method Based on Fuzzy Control Theory," *Journal of Control Science and Engineerin*, vol. 2015, Article ID 383209, 10 pages, 2015.

[24] H. Choi, J. Lim, H. Yu, and E. Lee, "Task Classification Based Energy-Aware Consolidation in Clouds," *Scientific Programming*, vol. 2016, pp. 1–13, 2016.

[25] W. Hong-Qiang, L. Xiao-Yong, F. Bin-Xing, and W. Yi-Ping, "Resource Scheduling Based on Improved FCM Algorithm for Mobile Cloud Computing," in *Proceedings of the 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 128–132, Wuhan, China, December 2016.

[26] C.-H. Hong, K. Lee, H. Park, and C. Yoo, "ANCS: Achieving QoS through dynamic allocation of network resources in virtualized clouds," *Scientific Programming*, vol. 2016, 2016.

[27] J. Su, F. Lin, X. Zhou, and X. Lu, "Steiner tree based optimal resource caching scheme in fog computing," *China Communications*, vol. 12, no. 8, Article ID 7224698, pp. 161–168, 2015.

[28] M. Aazam and . Eui-Nam Huh, "Dynamic resource provisioning through Fog micro datacenter," in *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 105–110, St. Louis, MO, March 2015.

[29] S. K. Datta, C. Bonnet, and J. Haerri, "Fog Computing architecture to enable consumer centric Internet of Things services," in *Proceedings of the IEEE International Symposium on Consumer Electronics, ISCE 2015*, Spain, June 2015.

[30] K. Chaudhuri, F. Chung, and A. Tsiatas, "Spectral clustering of graphs with general degrees in the extended planted partition model," *Journal of Machine Learning Research*, vol. 23, p. 35.23, 2012.

[31] S. B. Belhaouari, S. Ahmed, and S. Mansour, "Optimized K-means algorithm," *Mathematical Problems in Engineering*, vol. 2014, Article ID 506480, 14 pages, 2014.

[32] H. Rong, H. Wang, J. Liu, J. Hao, and M. Xian, "Privacy-Preserving k-Means Clustering under Multiowner Setting in Distributed Cloud Environments," *Security and Communication Networks*, vol. 2017, 19 pages, 2017.

[33] H. Zhou, S. Deng, H. Huang, and Y. Wu, "Resource allocation in cloud computing based on clustering method," in *Proceedings of the 2015 9th Annual IEEE International Systems Conference (SysCon)*, pp. 489–494, Vancouver, BC, April 2015.

[34] F. Duan and X. Shen, http://d.wanfangdata.com.cn/Thesis/D279482.

[35] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-Efficient Admission of Delay-Sensitive Tasks for Mobile Edge Computing," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2603–2616, 2018.

[36] C. Liu, X. Guo, Z. Li, Y. Wang, and G. Wei, "Multisensors Cooperative Detection Task Scheduling Algorithm Based on Hybrid Task Decomposition and MBPSO," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[37] Y. Li, P. A. Frangoudis, Y. Hadjadj-Aoul, and P. Bertin, "A Mobile Edge Computing-assisted video delivery architecture for wireless heterogeneous networks," in *Proceedings of the 2017 IEEE Symposium on Computers and Communications, ISCC 2017*, pp. 534–539, grc, July 2017.

[38] C. Long, Y. Cao, T. Jiang, and Q. Zhang, "Edge Computing Framework for Cooperative Video Processing in Multimedia IoT Systems," *IEEE Transactions on Multimedia*, vol. 20, no. 5, pp. 1126–1139, 2018.

[39] A. A. Alsaffar, H. P. Pham, C.-S. Hong, E.-N. Huh, and M. Aazam, "An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing," *Mobile Information Systems*, vol. 2016, Article ID 6123234, pp. 1–15, 2016.

[40] M. Yang, X. Gao, C. Yuan et al., "Resource scheduling of workflow multi-instance migration based on the shuffled leapfrog algorithm," *Journal of Industrial Engineering & Management*, vol. 8, no. 1, pp. 217–232, 2015.

[41] N. Xing, "Resources Scheduling Algorithm in Power Wireless Private Network Based on SDON," *International Journal of Optics*, vol. 2017, Article ID 2064638, 8 pages, 2017.

[42] A. Asheralieva and Y. Miyanaga, "Dynamic resource allocation with integrated reinforcement learning for a D2D-enabled LTE-A network with access to unlicensed bandt," *Mobile Information Systems*, vol. 2016, 2016.

[43] J. Yu, Y. Qi, G. Wang, and X. Gu, "A cluster-based routing protocol for wireless sensor networks with nonuniform node distribution," *AEÜ - International Journal of Electronics and Communications*, vol. 66, no. 1, pp. 54–61, 2012.

[44] L. Zhao, *Research and improvement of global K-means clustering algorithm*, Xidian University, 2013.