Hindawi
Mobile Information Systems
Volume 2018, Article ID 3601604, 16 pages
https://doi.org/10.1155/2018/3601604

Research Article

Multistage Dynamic Packet Access Mechanism of Internet of Things

Yi Meng[1], **Chen QingKui**[1,2] **and Zhang Gang**[1]

[1]*School of Management, University of Shanghai for Science and Technology, Shanghai, China*
[2]*School of Optical Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China*

Correspondence should be addressed to Chen QingKui; chenqingkui@usst.edu.cn

Received 1 October 2017; Revised 14 March 2018; Accepted 1 April 2018; Published 8 May 2018

Academic Editor: Paolo Bellavista

In the scenario of mass control commands requesting for network access, confined by the best-effort network service mode, it is easy to bring about resource competition and thus a phenomenon of access failure on major and urgent service request at the data access center for the Internet of Things. In this event, the dynamic diversification of control command is unable to access the necessary resources on a comparatively fair basis, causing low efficiency in heterogeneous resource utilization at the access center. This paper defines the problem of group request dynamic resource allocation and further converts it into the problem of 0-1 integer and linear programming and proposes a multistage dynamic packet access strategy. This strategy works first on dynamic group division on the users' mass control requests using the high ability of self-organizing feature maps and then searches for the optimized matching resources based on the frog-leaping algorithm which has a better capacity for global searching for the best resources. This paper analyzes the feasibility of this strategy and its astringency. The experimental results demonstrate that the strategy can effectively improve the success rate of access to the data center for the Internet of Things and reduce network blockage and response delay.

1. Introduction

The Data Access Center for the Internet of Things (DACIOT), serving as the core layer of Internet of Things (IoT), has already become a current research focus [1–3]. In the different applications of the smart city, such as smart healthcare [4, 5], smart home [6], intelligent transportation [7], smart logistics [8], and intelligent security [9], IoT firstly collects the mass sensing data in real time from hundreds of sensor devices in accordance with the corresponding communication protocols. Secondly, DACIOT carries out preprocessing of the sensing data by means of data cleaning, data integration, selection, and transformation. In order to support the upper application system for automatic identification, accurate positioning, intelligent tracking, and intelligent supervision of entities and processes, DACIOT integrates the perception information and coordinates communications between devices. Therefore, a universal and integrated network is generated by the fusion of human, machine, and things. In the process of such fusion of the upper IoT application system, a variety of value type and nonvalue type of data are continually transmitted by different types of sensors [10]. In this process, different kinds of control requests will be delivered to the masses of sensing devices by a great quantity of end users. From the perspective of the application architecture of IoT [11, 12], DACIOT is mainly located at the application level, which is the top level of IoT architecture [13], as shown in Figure 1.

As shown in the above figure, a large number of static or mobile sensing equipments constitute the sensing net in the form of self-organizing or multi-hop. After optimization, the data collected by sensing equipments are transmitted to information processing center via radio waves and then converged to sending net (network layer in Figure 1). The major role of DACIOT is to provide services of network access for end users (including mobiles and PCs) and coordinate that the service requests of a large number of end users can be transmitted onto the corrective sensing equipments [14] via Internet. Relatively speaking, the topology of sensing net is more vulnerable to the external environment than the topology of sending net. In terms of processing capacity of network, sending net has a stronger data processing and control ability,

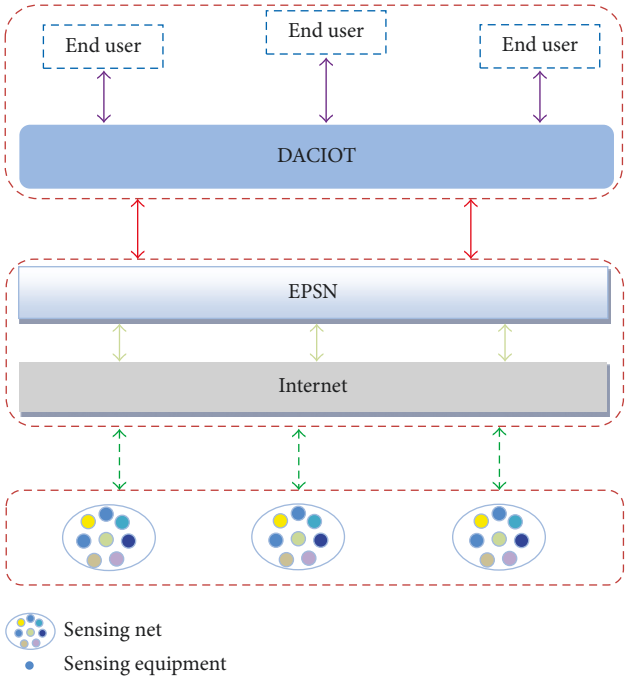

Figure 1: Structure of "man-machine-thing" integration of the IoT application system.

while the sensing net is only responsible for perception data collecting without the ability of intelligent data processing. Moreover, as the channel of information transmission of DACIOT, the sending net based on Internet exchanges the data from end users and sensors in real time accurately with the convergence of various wired and wireless networks. In respect of transmission of control demand, due to the features of Internet network of complicated and flexible structure and fast resource consumption when a large number of users request for access, EPSN (effective path statistic network) is constructed and deployed on the basis of the network coverage theory [15]. EPSN is a logic and virtual subnet of the Internet, and it is composed of a set of communication service agents, which contains a couple of simple functions, including packet sending, packet receiving, packet storage, and interaction. Using statistical theory, EPSN network explores the effective "gap" path which means the resources on the Internet [16]. Moreover, the primary function of EPSN lies in measuring and collecting the statistics of routing metrics on the logical link among communication service agents and constructing effective paths in real time based on the measurement and statistics result. In this significance, the network that end users request for access is transformed from an Internet network with uncontrollable parameters to a transparent EPSN network with controllable parameters. However, as the network features best-effort service [17], when mass requests from end users are accessed into the EPSN network via DACIOT, resource competition will take place and lead to the following problems: (1) the best-quality resources are taken and occupied by nonurgent requests so that urgent requests are inaccessible and cannot be transmitted onto the sensing device via the EPSN network transmission; for example, the switch control access command is blocked by ordinary access requests on the sensing device; (2) as users' requests are dynamic and diversified, the heterogeneous resources of DACIOT may not meet the demand for high service quality; for example, when users' request commands which are used to query sensed data come to access, they require a very strict response delay and packet loss rate; and (3) when allocating resources, it is hard for DACIOT to treat the various types of service requests fairly, causing service overleaning; for example, DACIOT may allocate more resources to users who request access to humidity and temperature sensing devices while allocating less resources to users who request access to carbon dioxide sensing devices. Therefore, it is critical and urgent for DACIOT to efficiently coordinate the mass access requests from users for quality resources.

2. Related Work

In relation to the current research on DACIOT, the work in [18] provides a summary of the system structure and characteristics of the M2M communication network in a 3GPP environment; analyses the potential problems existing during the access process in terms of the physical-layer transmission, random access procedure, and wireless resource allocation; and further proposes a solution with a QoS guarantee. A method of effective user authentication and access control at a perceptive layer is proposed in [19]. This method establishes a session key based on an elliptic curve encryption mechanism and further enhances the mutual authentication process between the user and sensor node. The work in [20] provides a cloud manufacturing service method based on the smart perception and resource visit control of IoT, which includes a five-level IoT resource framework, that is, levels of resources, perception, network, service, and application. The research in [21] proposes an IoT-based urgent medical service system on the basis of heterogeneous resource instant visits, which combines ambulance, nurse, doctor, and medical records into a huge resource repository to be the resource level of the system. The literature in [22] discusses a service-oriented QoS optimization scheduling algorithm based on the Markov decision process for the IoT application level. The work in [23] discusses an IoT congestion control mechanism based on improved random early detection, and the mechanism is analyzed using the Queueing Theory. The research in [24] provides a cost-efficient analytic model which can predict device performance according to a limited capacity queueing system with prioritized services and is specific to heterogeneous IoT devices with the feature of postponing sensitive information. The literature in [25, 26] discusses a kind of resource allocation algorithm which concentrates on searching for the Nash equilibrium. The literature [27] discusses the request access control mechanism for limited resources based on the Zero-sum Game, which makes a decision as to whether a request is to be accessed or not based on the request's utility function. The work in [28] analyses the behavioral characteristics of individual and group users of mobile, providing resource allocation models for wireless communication between devices. However, in the above proposed methods, the methods in [18, 21, 23] do not take the differences of access resources into consideration. The proposed method in [19] only solves the problem of

resource limitation of IoT at the perceptive layer. The work in [20, 24] is only applicable to small-size access requests. The research in [22, 25–28] does not have answers to address competition when core request services match high-quality resources. Generally, these models are not able to solve the problem of competition between important and urgent service requests in the process of IoT mass access.

In regard to DACIOT, a large amount of real-time data collected by thousands of sensors arrive continuously every day, and end user requests are always under the constraints of the "Things-Contact" features in IoT: (1) the scale of sensor data is large; (2) every end user's service request has a high constraint of reliability and quality of service [29, 30], such as it requires a shorter response time for intelligent terminal equipment, and the service request should be transmitted to the intelligent terminal successfully through the data access center; and (3) the group service requests require all requests in the same group to be able to access DACIOT under the condition of the group constraint set; moreover, the group requests as a whole have a strict limit of response delay for DACIOT. For instance, in a service group with 20 commands, each service group command needs to complete the task within a specific time and resource constraint, and the service group is regarded as having successful access if only these 20 commands have been given access within the limited response time and resources. Currently, the types of end user request commands tend to be relatively small; moreover, the network infrastructure and resource distribution are relatively concentrated [31]. Hence, under the condition of sharing a particular service resource, a large number of end users' service request commands in IoT require DACIOT to provide group access services at the same time with very high service constraints.

Based on this background, to address the problem of competition among important and urgent service request in the process of IoT access, all the users' access requests should be seen as obtaining the best-matched resources under the condition of meeting more than one limit. When similar service requests are separated from those that are different, based on the type and response domain of access requests, the similar service requests tend to share a close resource distribution area, and these similar access requests can thus been seen as a bionic group. Each group's access request searches for the optimal-matching resources at that moment under the current network environment. As access requests change, so will DACIOT's resource distribution, requiring DACIOT to allocate the optimal resources at the moment for all the groups' service requests. This further converts IoT's mass access into the dynamic grouping resource allocation problem (GRAP) [32]. In GRAP, each of the access requests represents a compete process, which means that the access will fail if all the necessary resources cannot be obtained, and there will be an upper boundary constraint on the number of access service requests and a lower boundary constraint on resource measurement. With such a feature and for the purpose of finding a solution, GRAP can be converted into 0-1 integer and linear programming [33]. Moreover, this is NP Hard [34] that the general algorithm is difficult to solve due to the limitation of the special problem. Therefore, this paper proposes a multistage dynamic grouping access strategy (MSDGAS) based on self-organizing feature maps neural network [35] and frog-leaping algorithm [36].

The third part of the paper describes this issue and provides the mathematical model; the fourth part first designs a self-adaptive scheduling mechanism for resource dynamic grouping and further designs a group dynamic resource allocation algorithm based on the frog-leaping algorithm; the fifth part presents the experiment process and analysis; and the sixth part gives the conclusion and suggestions for future research.

3. Problem Description

3.1. Analysis of the Dynamic Grouping Resource Allocation Problem. To solve the problem of mass data access of IoT, it needs to meet users' demand of access request with a limited network load. With the confinement of the network service model, when the data access center meets an urgent service request, GRAP will face optimal resource competition, causing failure of the access request. This requires searching for a group of optimal solutions based on the confined network conditions in a dynamic network condition so that mass users' requests can access DACIOT. In the process of requesting access, as lots of the control commands have certain common attributes, such as being in the same geographic area, taking control of the intelligent equipments with the same type, and so on, these commands can be clustered into different command groups by similar attributes and these groups are sent at the same time in order to save network bandwidth and improve the efficiency of access. Suppose there are k-command groups in a single access process, and each command group G contains N control commands. Moreover, each command C in the command group G denotes that one end user can take control of L intelligent equipments simultaneously. For the sake of presentation, if the control command C does not intend to control an intelligent device, the corresponding value in the command is set to 0. Furthermore, these k command groups can be regarded as k two-dimensional matrices with N rows and L columns, and each control command C can be considered as a vector with L dimensions after regularization. Since the high-quality resources are limited, different command groups will compete for resources in the process of access, and then each command group G will have a corresponding competition weight. Meanwhile, each smart device has a variety of control types, which take on different urgency in different scenarios. The urgency can be expressed by the weight value: the greater the value is, the more urgent the command group is. Choose the weight of the most urgent control type in control command C as the weight of this control command C, and then use the weight of all the control commands in the command group G to measure the urgency of this command group G, namely, the comprehensive competition λ. Therefore, to solve the group dynamic resource allocation problem under the condition of limited network resources is to find a set of optimal resource access path, coordinating these different command groups to be transmitted onto the corresponding intelligent equipments using the minimum resource load. Here is the detailed process of mathematical modeling of this problem.

3.2. Modeling Process of the Dynamic Grouping Resource Allocation Problem. It is assumed that the command group set: $G_1, G_2, \ldots, G_k$, with a weight function $\lambda(\psi)(\psi \in \{1, 2, \ldots, k\})$, which reflects the competitive ability of a command group in the process of service request access in terms of geometrical significance. Moreover, $\lambda(\psi)$ has a range of values from 0 to 1, and the closer the value of $\lambda(\psi)$ is to 1, the stronger the competitive ability of the corresponding command group. Conversely, the closer value of $\lambda(\psi)$ to 0, the weaker the competitive ability. The request type set CAT is comprised of r command request types and $\mathrm{CAT} = \{\mathrm{cat}_1, \mathrm{cat}_2, \ldots, \mathrm{cat}_r\}$. In the command group of G_ψ, $\psi \in \{1, 2, \ldots, k\}$, $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_m$ are service request commands of DACIOT, $\mathbf{C}_i$ refers to the ith service request command, and $\mathbf{C}_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T$, cat_i^ψ refers to the request type of the most urgent subcommand in $\mathbf{C}_i$, and $\mathrm{cat}_i^\psi \in \mathrm{CAT}$. $\vartheta_{\mathrm{cat}_i^\psi}$ is the weight value of the request type cat_i^ψ, which is generally set according to the emergency degree of the commands in this request type upon initialization. Then, the weight value of G_ψ can be represented as

$$\lambda(\psi) = \frac{\sum_{i=1}^{m} \vartheta_{\mathrm{cat}_i^\psi}}{\sum_{\psi=1}^{k} \sum_{i=1}^{m} \vartheta_{\mathrm{cat}_i^\psi}}. \tag{1}$$

$R_1, R_2, \ldots, R_n$ represent the resource of DACIOT, R_j is the jth resource, LM_i is the load metrics function consumed by the command $\mathbf{C}_i$ in resource access, and E_i is the load value consumed by the command $\mathbf{C}_i$ to acquire all the resources it needs and

$$E_i = \sum_{d=1}^{n} \mathrm{LM}_i(c_{id}). \tag{2}$$

Φ_i is the resource confinement for $\mathbf{C}_i$, Q_ψ is the resource confinement for command group G_ψ, and F_{G_ψ} is the objective function of command group G_ψ at the current load confinement; Q means the system resource confinement and

$$Q \geq \sum_{\psi=1}^{k} Q_\psi. \tag{3}$$

Table 1 presents the parameter list for the GRAP modeling, and a mathematically formalized description is given as follows.

GRAP: when the load function meets the measurement confining condition, the GRAP objective function will reach the optimal value (the minimum means the optimal):

$$\min \mathrm{F} = \sum_{\psi=1}^{k} \lambda(\psi) F_{G_\psi}, \tag{4}$$

$$\begin{aligned} F_{G_\psi} &= \sum_{i=1}^{m} \sum_{d=1}^{n} \mathrm{LM}_i(c_{id}), \\ \text{subject to: } & E_i \leq \Phi_i, \\ & 0 \leq \lambda(\psi) \leq 1, \\ & \sum_{i=1}^{m} E_i \leq Q_\psi. \end{aligned} \tag{5}$$

In light of the GRAP characteristics, to obtain a convenient solution, GRAP can be converted into 0-1 integer and linear programming. Suppose δ_{ij} refers to the load value in the process of command $C_i(C_i = (c_{i1}, c_{i2}, \ldots, c_{in})^T)$ acquiring resource $R_j(R_j = (r_{j1}, r_{j2}, \ldots, r_{jn})^T)$, and

$$\delta_{ij} = \sum_{d=1}^{n} \mathrm{LM}_i\left(R_{jd}\right), \quad d \in \{1, 2, \ldots, n\}. \tag{6}$$

Table 1: Parameters of the GRAP modeling.

Parameters	Descriptions
k	The number of command groups
G_ψ	The ψth command group
CAT	The set of command request types
$\lambda(\psi)$	The weight function of command groups
m	The number of service request commands
C_i	The ith service request command
cat_i^ψ	The request type of the most urgent subcommand in $\mathbf{C}_i$
$\vartheta_{\mathrm{cat}_i^\psi}$	The weight value of cat_i^ψ
n	The number of effective resources
R_j	The jth resource
LM_i	The load metrics function consumed by the command $\mathbf{C}_i$ in resource access
E_i	The load value consumed by $\mathbf{C}_i$ to acquire all the resources it needs
Φ_i	The resource confinement for $\mathbf{C}_i$
Q_ψ	The resource confinement for G_ψ
F_{G_ψ}	Objective function of G_ψ at the current load confinement
Q	System total resource confinement
δ_{ij}	The load value of $\mathbf{C}_i$ acquiring resource $\mathbf{R}_j$
F	Objective function of GRAP

Suppose $\lambda_1, \lambda_2, \ldots, \lambda_k$ are, respectively, the weight factors of command group $G_1, G_2, \ldots, G_k$, then the mathematical model for the minimum load function of DACIOT is described as follows:

$$\min F = \lambda_1 F_{G_1} + \lambda_2 F_{G_2} + \ldots + \lambda_\psi F_{G_\psi} + \ldots + \lambda_k F_{G_k}, \tag{7}$$

$$\begin{aligned} F_{G_\psi} &= \sum_{i=1}^{m} \sum_{j=1}^{n} \delta_{ij} X_{ij}, \\ \text{s.t.} & \sum_{i=1}^{m} X_{ij} = 1, \quad j = 1, 2, 3, \ldots, n \\ & \sum_{j=1}^{n} X_{ij} = 1, \quad i = 1, 2, 3, \ldots, m \\ & X_{ij} \in \{0, 1\}, \quad \forall i, j \\ & \delta_{ij} \leq \phi_{ij}, \\ & 0 \leq \lambda_\psi \leq 1, \\ & \sum_{i=1}^{m} E_i \leq Q_\psi. \end{aligned} \tag{8}$$

In (7), F represents the objective function, which reflects the total load value consumed by all the group commands acquiring all the computing resources in the practical scene; $\min F$ means the minimum objective function; and λ_ψ refers the weight value of the command group G_ψ, which reflects the competitive ability of the command group G_ψ. In (8), F_{G_ψ}

means the load value consumed by the ψth command group acquiring the resource it needs in the real applications; δ_{ij} refers to the load value of command C_i acquiring resource R_j; s.t. refers to the limited conditions of the objective function F; $X_{ij} = 0$ means command $\mathbf{C}_i$ does not acquire the resource $\mathbf{R}_j$; $X_{ij} = 1$ means $\mathbf{C}_i$ acquires $\mathbf{R}_j$ successfully; and ϕ_{ij} means the confinement when the service request command $\mathbf{C}_i$ matches the optimal resource $\mathbf{R}_j$. In such a scenario, GRAP can be further defined as the problem to search for a set of optimal solutions so that the current objective function (7) is optimal (the minimum means optimal as a definition), when each of the request commands meets the condition of the load confinement.

To make it easier to follow the above formulas, a real application of Smart Home described in reference [37] is taken, for example. As mentioned in the reference, Smart Home is a classic example of IoT, wherein smart appliances connected via home gateways constitute a local home network to assist people in activities of daily life, and the remote DACIOT controls the home gateways to build a wider network. Smart Home involves IoT-based automation (such as smart lighting, heating, window, power plug, and surveillance), remote monitoring, and control of smart appliances. In a scene, it starts to rain suddenly, and then twenty employees of the same company in a Central Business District will want to take control of smart appliances inside their own homes in the meantime. For example, the company employee Jack wants to query the screen of smart monitor device, turn off the smart window, and turn on heating and lighting appliance inside the home for his mother who is coming home. In general, the type of shutdown command has a higher priority than that of the query command, and the command of "turn off smart window" is the most urgent in this scene. As a supplement, each command of company employees has the properties of atomicity; in other words, Jack's control command is effective as a whole only if all the subcommands (query the screen, turn off smart window, turn on heating, and lighting appliance) run successfully. For this scenario, because of the same properties of the geographic region, the twenty employees' commands are classified as a same command group, and G_ψ is used to represent the group of the twenty employees' commands. Moreover, there are many companies in the Central Business District; that is, lots of command groups will be generated at the same time. The command type set CAT contains "turn off" which is coded as 1, "turn on" which is coded as 2, and "query" which is coded as 3; Jack's command can be represented by C_i, and the command has many subcommands; the control command of Jack can be represented as $C_i = (3, 1, 2, 2, 0, 0, 0, 0, 0, 0)$, where $C_{ij} = 0$ means that Jack does not take control of the jth appliance; the value of cat_i^ψ is "turn off" for Jack; $\lambda(\psi)$ refers to the weight value of the command group composed of twenty employees' commands; E_i means the network bandwidth consumed by Jack's control command when the command is transmitted to the smart devices inside home; Q_ψ means the network bandwidth consumed by the twenty employees' control command. Based on this, formula (1) is used to calculate the weight value of the company's control command group on smart appliances at DACIOT. Furthermore, the weight value reflects the priority of the command group since DACIOT needs to handle a large number of service requests from different companies at the same time. Formula (2) is used to calculate the value of network bandwidth consumed by the device control command; formula (3) is used to convey the system resource constraints of formula (2); formula (5) is to calculate the minimum value of network bandwidth consumed by all the group commands at the current time.

Regarding 0-1 integer and linear programming, this paper proposes a multistage dynamic grouping access strategy (MSDGAS). The following paragraphs provide a detailed description on MSDGAS.

4. Multistage Dynamic Grouping Access Strategy

The central idea of the MSDGAS is that when the amount of requests has become huge, the requests will be grouped based on certain properties at the users' request side, and requests with similar properties are included into the same user group, which refers to the service request collection divided by the request type and attributes. A user group can be distinguished by the type of request or scene area, and each of the groups has more or less the same request amount. At the resource side of IoT, similar resources are divided into groups and each group's resource volume varies. Their structure is shown in Figure 2.

MSDGAS is composed of three main parts: ReqEthnic, ResGrp, and Dev. ReqEthnic means the user group, and each group has many requests; Req refers to the user's request; for instance, the first request grouping includes the requests Req_1, Req_2, and so on; ResGrp refers to resource grouping, and each grouping has a number of relevant resources; Res refers to the resources of DACIOT; for instance, the first resource grouping includes resource Res_1, Res_2, Res_3, and Res_4; Dev refers to the responding device, including Dev_1, $\text{Dev}_2, \ldots, \text{Dev}_n$. In DACIOT, a large amount of service request at the ReqEthnic side will be grouped automatically, and then they will seek a group of optimal resources based on the constraint conditions of the request grouping from the ResGrp side. Finally, these service requests are transmitted to devices in the Dev side. MSDGAS first introduces optimized dynamic self-organizing feature maps (DSOM) to work on the dynamic grouping of the users' access request. It then applies improved frog-leaping algorithms to search for the optimal path for the grouped access requests.

4.1. Optimized Dynamic Self-Organizing Feature Maps

4.1.1. DSOM Introduction. Finnish scholar Kohonen first proposed an algorithm called self-organizing feature maps neural network (SOM) [38], which has been widely used in many fields of information processing based on the feature of distributed information storage, topographical structure maintenance, good self-organizing and self-learning ability,

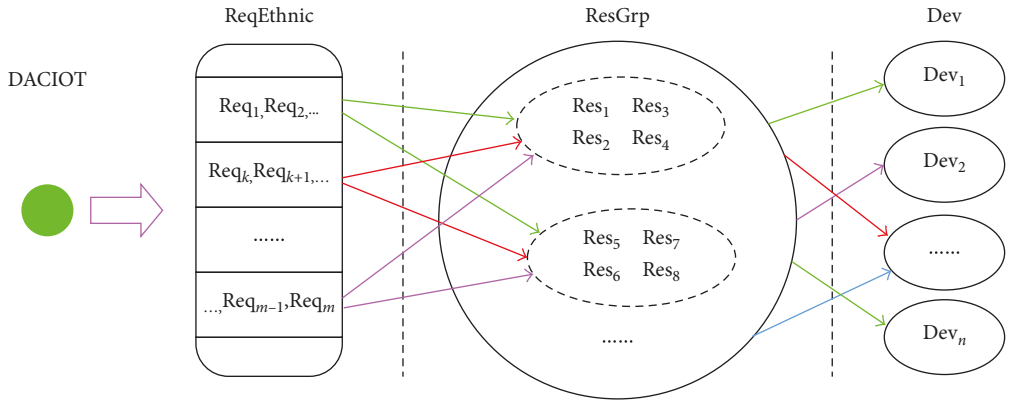

Figure 2: Structure of MSDGAS.

information visualization, nonsupervision clustering, and massive parallel processing. SOM not only includes an input layer but also contains a competitive layer, and the input layer is the status of full mesh with the competitive layer through each neuron which refers to one category [39]. According to the rules of learning, the objective of capturing the feature of each of the input models can be attained by repeated learning, and the self-organizing clustering will be carried out in these features. In the end, the cluster analysis results will be generated at the competitive layer. Figure 3 illustrates the topographical graph of SOM. The figure shows that the input layer is composed of N input neurons, and the competitive layer is composed of M output neurons. In the input layer, $X_1, X_2, \ldots, X_n$ denote the N input neurons and are mapped into the competitive output layer through clustering analysis. Dotted lines completely connect neurons between the input layer and the competitive layer. In order to acquire an accurate enough grouping result in the competitive layer, the repeated self-organizing clustering needs to be carried out. When making use of a SOM model, the number of neurons M is preset at the competitive layer, so the convergence speed will be astricted to a large extent. Therefore, under the environment of the Internet of things large-scale access requests, in order to meet the users' access requests to realize self-organizing clustering based on their measurement nature, the method of SOM needs to be improved. In this sense, an optimized method of DSOM is proposed, which combines a disturbance factor and growing threshold. Here, variable ξ is used to represent the disturbance factor, and variable φ means the growing threshold. φ will affect the network structure growth dynamically among the training process, while ξ can regulate the growth direction of network and restrain it falling into local optimization. Consequently, the goal of hierarchy clustering is achieved.

The calculation method of growing threshold φ is defined as follows:

$$\varphi = \begin{cases} (d \times f(\xi) \times n(t)) / (1 + n(t)), & \text{init} = 0 \\ d \times f(\xi), & \text{otherwise,} \end{cases} \tag{9}$$

where $n(t)$ refers to the current network node number at the tth time iteration; d represents input vector dimension;

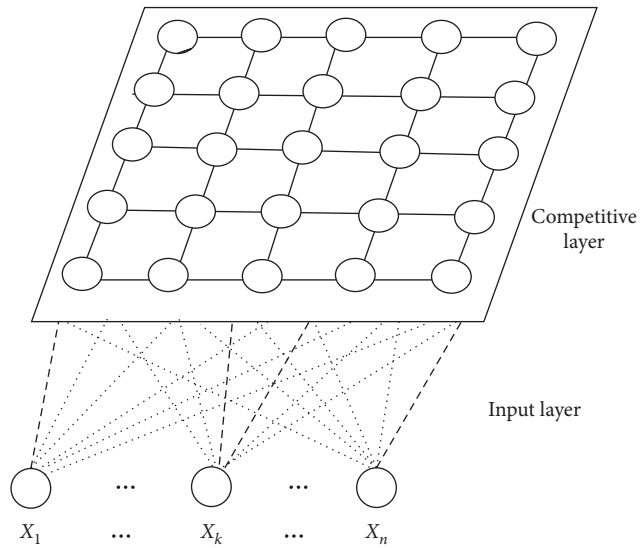

Figure 3: Topology of SOM neural network.

init = 0 stands for the initial status; and $f(\xi)$ is called the hierarchical classification function.

$$f(\xi) = \sqrt{1 - \xi}, \qquad \xi = \text{rand}(t); \tag{10}$$

where rand(t) is the random number, and the value of which ranges from 0 to 1 at the tth time of iteration.

Definition 1. For $\mathbf{v}$, the network's input dimension vector, the nearest node is viewed as the best matching node at the competitive layer, and the abbreviation of the best matching node is bmn. Therefore, the formula is defined as

$$\left|\left|\mathbf{v} - \boldsymbol{\omega}_{\text{bmn}}\right|\right| \leq \left|\left|\mathbf{v} - \boldsymbol{\omega}_{n_i}\right|\right|, \quad \forall n_i \in N, \tag{11}$$

where $\boldsymbol{\omega}$ is regarded as the weight dimension vector of the node, n_i is the ith network node, N stands for the total of n_i, and $||\bullet||$ is called the Euclidean distance. $||\mathbf{v} - \boldsymbol{\omega}_{\text{bmn}}||$ means the Euclidean distance between vector $\mathbf{v}$ and vector $\boldsymbol{\omega}_{\text{bmn}}$, which is the weight dimension vector of bmn.

Definition 2. The distance between the input dimension vector $\mathbf{v}$ and its best matching node bmn is regarded as the standard deviation value between them, which is represented by E. Therefore, the formula is defined as

$$E = \sum_{k=1}^{d} \left(v_k - \omega_{\text{bmn},k}\right)^2. \tag{12}$$

In the above formula, d is the dimension of $\mathbf{v}$.

Definition 3. The node of the competitive layer n_i and its direct subnode are referred to as the neighborhood of n_i, which is called $\sigma(n_i)$.

4.1.2. Description of Optimized Dynamic SOM. $N = \{n_0, n_1, n_2, \ldots, n_m\}$ is the node set of the competitive layer, and $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ stands for the input dimension vector set; $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ is the ith input dimension vector; d represents the dimension of the input vector; n_0 is called the initial node; $\sigma_k(n_i)$ denotes the neighborhoods of the node n_i at the kth time of iteration, and $\sigma_k(n_i) = \sigma_0 \exp(-(k/\tau))$ where σ_0 represents the initial value of σ_{n_i} which is a bigger value in general and τ is called the exponential decay constant. The process of DSOM algorithm is determined by the following steps.

Step 1. Initiate the node n_0, neighbourhoods $\sigma(n_0)$, weight dimension vector ω_{n_0}, and growth threshold φ; ω_{n_0} is a random value ranging from 0 to 1, at the initial period $t = 1$; φ is calculated according to formulas (9) and (10), and then each vector of the input vector set V is standardized between 0 and 1. When it is $\overline{\mathbf{v}_i} = \left(\sum_{j=1}^{d} v_{ij}^2\right)^{1/2}$, the standardized $\mathbf{v}_i$ is $\dot{\mathbf{v}}_i = \left(v_{i1}/\overline{v_i}, v_{i2}/\overline{v_i}, \ldots, v_{id}/\overline{v_i}\right)$.

Step 2. Select the input dimension vector sequentially from V, and search for the best matching node (bmn) of the vector $\mathbf{v}$ from the current network node collection N, calculated by formula (11).

Step 3. Calculate the deviation between bmn and v as E according to formula (12). If $E \le \varphi$, skip to Step 4 for weight value updating operation; if not, go to Step 5 for the node growth operation.

Step 4. Adjust bmn neighborhood's weight value via formula (13).

$$\omega_{n_j}(k+1) = \begin{cases} \omega_{n_j}(k), & j \notin \sigma_{k+1}(\text{bmn}) \\ \omega_{n_j}(k) + \text{LR}(k) \times \left(\dot{\mathbf{v}} - \omega_{n_j}(k)\right), & j \in \sigma_{k+1}(\text{bmn}), \end{cases} \tag{13}$$

In the above formula, $\text{LR}(k)$ is the learning rate, when $k \to \infty$, $\text{LR}(k) \to 0$, and $\omega_{n_j}(k)$ and $\omega_{n_j}(k+1)$ are the weight values of n_j prior and post adjustment, respectively. $\sigma_{k+1}(\text{bmn})$ is the neighborhood when bmn is at $k+1$ times of iteration.

Step 5. Generate a new node n_p of bmn, and make $\omega_{n_p} = \dot{\mathbf{v}}$.

Step 6. $\text{LR}(t+1) = \text{LR}(t) \times \alpha$, α is the regulating factor of LR, $0 < \alpha < 1$.

Step 7. Repeat Step 2 to Step 6 till all input dimension vectors in V have been trained completely.

Step 8. Make $t = t + 1$, repeat Step 2 to Step 7, and enter into the next iteration period till no more new nodes are generated in the network.

4.2. Dynamic Niche-Based Shuffled Frog-Leaping Algorithm. The shuffled frog-leaping algorithm (SFLA), a new-type bionic group intelligent optimized algorithm, was proposed first by Eusuff and Lansey in 2003 for the purpose of combinatorial optimization [40]. The algorithm combines the merits of the two intelligent algorithms, namely, the meme algorithm based on meme evolution, and particle swarm optimization based on group behavior. It integrates the overall information exchange and local depth search with an aim to realize a balance between the overall search for the best and local solutions. This algorithm has advantages [41] such as its simple concept, fewer parameters to adjust, fast calculation, powerful overall searching for the best solution, and robustness. However, SFLA is not fast in convergence when it is working on a local search within a group, and it is easy to converge to the local best solution, which will further impact the algorithm's overall convergence. In this regard, niche technology [42] is able to maintain the group diversity and constrain the overmature phenomenon effectively. Inspired by the niche technology, this paper proposes a dynamic niche shuffled frog-leaping algorithm (DNSFLA). The proposed DNSFLA absorbs the advantages of SFLA and niche technology, and it can find the global optimal solution with a faster and easier way. Furthermore, the essence of the proposed algorithm is to search the resource sequence for each group command from end users in the global resource space using parallel computing.

4.2.1. Idea behind DNSFLA. In the traditional shuffled frog-leaping algorithm, group $P = \{X_1, X_2, \ldots, X_F\}$ is comprised of many frogs of the same structure, and each of the frogs represents a viable solution for the problem of optimization. $f(X_i)$ means the adaptability value of the individual frog X_i; individuals in Group P are arranged in order based on their adaptability values, and then the individual frogs in order are allocated into m groups; that is, the first frog is allocated into Group 1, the second into Group 2, the m into Group m, and $m+1$ into the Group 1, and so forth, till F frogs are evenly allocated into Group P and in each of the groups, there are the same number (n) of frogs. In this sense, $F = m \times n$. Different groups can be regarded as frog collections with different thoughts. Each of the subgroups has its best adaptability X_b and the worst adaptability X_w. In the process of searching for the best solution among subgroups, it only needs to update the location of the worst solution X_w, and the formula for updating this is

$$\Delta = \text{rand}() \times (X_b - X_w), \tag{14}$$

$$X_w = X_w + \Delta, \quad -\Delta_{\max} \le \Delta \le \Delta_{\max}. \tag{15}$$

In the above formula, Δ is the length of movement, rand() is the random number between 0 and 1, and $\Delta_{\max}$ is the maximum length of movement.

After updating X_w, if a better solution is found, it will be applied to replace the worst solution X_w, otherwise the overall best solution X_g is used to replace X_b in formula (14), and the new solution is recalculated; if a better solution is not found, then a randomly generated new solution will be used to replace the worst solution X_w. This process is repeated till N, the predetermined evolutionary algebra, finishes the partial search amidst the subgroups. For the next step, all individuals in the subgroups are shuffled to form a new group P including F individual frogs, and their order is arranged from best to worst again, based on their adaptability for subgroup division and for next round of local search. Such an iteration is repeated till the termination condition is met.

In the process of local search according to the above algorithm, as updating the worst solution relies on the best solution, the algorithm is inclined to fall into searching for the local best, and the coefficient of the length of movement is a random function, which can make the algorithm premature and converge after several iterations. The DNSFLA algorithm, which is based on SFLA algorithm, applies the advantages of niche technology to maintain the solutions' diversity and has a better capacity in the overall search for an optimal solution and convergence and speeds up the elimination of the worst solution and increases the accuracy of the best solution. The idea behind the DNSFLA algorithm is that after dividing the subgroups by the SFLA algorithm, Chebyshev distance [43] is compared in pairs between X_{ib} and X_{jb} of the subgroups:

$$D\left(X_{ib}, X_{jb}\right) = \max_d \left(\left|X_{ib}^d - X_{jb}^d\right|\right). \tag{16}$$

If $D(X_{ib}, X_{jb})$ is smaller than niche distance L, the individual with the smaller value of adaptability in X_{ib} and X_{jb} is allocated with a punishment function $g(X_b)$ with the aim to change the adaptability value and place this individual into the subgroup which was initially to replace the original partially best individual and carry out the new local search till the iteration ends.

4.2.2. Realization of DNSFLA. In light of the aforementioned, the steps to realize the DNSFLA algorithm are detailed as follows:

(1) Initiate parameters including F, the total number of group solutions (frog); m, the number of the subgroups; n, the number of solutions in each of the subgroups; d, the number of problem dimension; G, times of shuffled iteration; N, times of partially updating each of the subgroups; $D_{\max}$, the maximum length of movement of solutions; L, the niche radius; $f(X)$, the function calculating adaptability; and $\Delta_{\min}$, the minimum change value of adaptability function. In the area of viable solutions, they will randomly generate F solutions, which form a group $P = \{X_1, X_2, \ldots, X_F\}$, and $f(X_i)$ the adaptability values of each of solutions are calculated, of which the ith solution is described as $X_i = \{X_i^1, X_i^2, X_i^3, \ldots, X_i^d\}$.

(2) All viable solutions in P are arranged in an A–Z order, and X_g, the highest adaptability, is recorded as the best solution in the entire collection of the viable solutions, that is, the overall best solution. Then, divide group P into m groups and form subgroups and place the first solution into the first group, the second solution into the second group, No. m solution into the No. m group, and the No. $m+1$ into the first group. Each of the subgroup has n solutions. Separately record the solution of the maximum adaptability of each of the groups as X_b, and the solution of the minimum adaptability of each of the groups as X_w.

(3) Place the best solution of each of the subgroup into the niche group DN, that is, DN $= \{X_{1b}, X_{2b}, \ldots, X_{mb}\}$; No. i and No. j best solutions are X_{ib} and X_{jb}, respectively; set $i = 1$ and $j = 1$.

(4) If $i \ne m - 1$, then compare the Chebyshev distance between the best solution of No. i $(1 \le i \le m-1)$ and the best solution of No. j $(i+1 \le j \le m)$ in DN; otherwise, turn to step (6).

(5) If $D(X_{ib}, X_{jb}) < L$, in which L is the niche radius and $j \ne m$, set $j = j + 1$, otherwise set $i = i + 1$ and $j = j + 1$; return to step (4); If $D(X_{ib}, X_{jb}) > L$ and $f(X_{ib}) > f(X_{jb})$, use punishment coefficient $g(X_b)$ to update $f(X_{ib})$, randomly initiate X_{ib}, and place the best solution X_{ib} into the responding subgroup to recalculate the partial best solution; otherwise, update $f(X_{jb})$, randomly initiate X_{jb}, and return to step (4).

(6) If each of the subgroups has the best solution, then move to step (7); otherwise, return to step (3) and compare the best solutions of the subgroups again.

(7) For each of the subgroups, update the worst solution X_w via formulas (14) and (15), and the adaptability value of the worst solution after updating $f'(X_w)$ is compared to that of the original $f(X_w)$. If $f'(X_w) < f(X_w)$, replace the original solution with the updated worst solution; otherwise, calculate the length of movement of the worst solution:

$$\Delta_i = \text{rand}() \times \frac{i}{N} \times \left(X_g - X_w\right), \quad i = 1, 2, \ldots, N. \tag{17}$$

Update the location of the worst solution X_w as per formula (15). In the formula, i is the number of iterations for the current local search. Then, the adaptability value of the worst solution after updating $f''(X_w)$ is compared to that of the original $f(X_w)$. If $f''(X_w) < f(X_w)$, replace the original solution with the updated worst solution; otherwise, it will randomly generate a new solution to replace the original worst solution.

```
(1) Initialize m, D_max, L, f(X), g(X), P = {X_1, X_2, ..., X_F}, and i = 1, j = 1
(2) for i ≠ m − 1 do
(3)    Calculate D(X_ib, X_jb)
(4)    if D(X_ib, X_jb) < L then
(5)       if j ≠ m then
(6)          j = j + 1;
(7)       else
(8)          i = i + 1, j = j + 1
(9)       end if
(10)   end if
(11)   if D(X_ib, X_jb) > L then
(12)      if f(X_ib) > f(X_jb) then
(13)         f(X_ib) = g(X_b)
(14)         Randomly generate X_ib
(15)      else
(16)         update f(X_jb), and initialize X_jb
(17)      end if
(18)   end if
(19) end for
```

Algorithm 1: Local search of DNSFLA.

```
(1) Initialize f(X), g(X), P = {X_1, X_2, ..., X_F}
(2) Update the worst solution X_w via formulas (14) and (15), recorded as f'(X_w)
(3) if f'(X_w) < f(X_w) then
(4)    f(X_w) = f'(X_w)
(5) else
(6)    Δ_i = rand() × i/N × (X_g − X_w), X_w = X_w + Δ_i
(7)    Record the adaptability value after updating as f''(X_w)
(8)    if f''(X_w) < f(X_w) then
(9)       f(X_w) = f''(X_w)
(10)   else
(11)      randomly generate a new solution to replace the original worst solution
(12)   end if
(13) end if
```

Algorithm 2: Elimination mechanism of DNSFLA.

(8) Judge if the predecided local evolution algebra N has been reached. If yes, a round of local search in each of subgroups has finished; otherwise, return to step (7).

(9) After finishing the partial evolution, calculate the variation of the adaptability of the best solution of each of the subgroups. If the variation is smaller than a comparatively small value within the predecided continual algebra, then the relevant subgroup is eliminated and initiated randomly; otherwise, reshuffle the solutions of all subgroups and form them into a complete group including F solutions.

(10) Judge if it meets the predecided times of shuffled iteration or termination conditions. If yes, the algorithm is finished; otherwise, return to step (2), and continue to do the next round of local search.

The core of the DNSFLA algorithm is the local search and the elimination mechanism of the subgroups. In order to assist it to understand, DNSFLA is decomposed into two subalgorithms (Algorithm 1 and Algorithm 2), and both the pseudocodes are given below, respectively.

Algorithm 1 shows the pseudocode of the local search, and the elimination mechanism's pseudocode is shown in Algorithm 2.

Combine Algorithm 1 and Algorithm 2, then the pseudocode of the DNSFLA algorithm is as shown in Algorithm 3.

4.3. Steps for MSDGAS. Depending on the previously mentioned DSOM and DNSFLA, the steps in detail that realize the MSDGAS strategy are given as follows.

Step 1. Initiate GRAP and its relevant functions and parameters, give GRAP's relevant measurement parameters and confinement condition, and move to Step 2.

Step 2. Simplify GRAP into 0-1 programming, convert the relevant functions and confinement conditions, and move to Step 3.

(1) Initialize $F, d, G, N, f(X), \Delta_{\min}, P = \{X_1, X_2, \ldots, X_F\}$ and $gt = 0, lt = 0$
(2) **for** $i = 1 : F$ **do**
(3) Calculate $f(X_i)$, and $X_i = \{X_i^1, X_i^2, X_i^3, \ldots, X_i^d\}$
(4) **end for**
(5) **repeat**
(6) Descending $f(X_i)$ and record the best solution X_g
(7) **for** $j = 1 : m$ **do**
(8) Calculate X_g and X_w, put X_b into niche group DN
(9) Choose X_{ib} and X_{jb}
(10) Search X_w using Algorithm 1
(11) **repeat**
(12) Update X_w using Algorithm 2
(13) $lt = lt + 1$
(14) **until** $lt = N$
(15) Calculate the variation of $f(X_b)$, recorded as $\Delta f(X_b)$
(16) **if** $\Delta f(X_b) < \Delta_{\min}$ **then**
(17) Initiated X_j randomly
(18) **else**
(19) Reshuffle the solutions of all subgroups
(20) **end if**
(21) **end for**
(22) gt = gt + 1
(23) **until** gt = G

Algorithm 3: Dynamic niche-based shuffled frog-leaping algorithm (DNSFLA).

Step 3. Specific to 0-1 programming, dynamic grouping is carried out based on the DSOM algorithm, and move to Step 4.

Step 4. Specific to the grouped request models, calculate the model result based fon the DNSFLA algorithm, and move to Step 5.

Step 5. Save the calculation result and exit.

4.4. Analysis of MSDGAS. MSDGAS is made up of two parts, the DSOM algorithm and the DNSFLA algorithm. In DSOM, the φ value determines the network growth size. When the value of φ is larger, the DSOM algorithm applies the weight value update operation more frequently; less nodes are generated in the objective network, but it generates a network at a faster speed and can only realize coarse-granularity clustering. On the other hand, when the value of φ is smaller, the DSOM algorithm applies more node growing operations, and more nodes are generated in the objective network, but it generates a network at a slower speed and can realize fine-granularity clustering.

After introducing disturbance factor (ξ), ξ tends to set a larger value when it is initiated. From formula (10), it can be seen that φ is comparatively small at this moment and can make a rough classification of the transmission command, which is able to have a holistic mastering of the group transmission command. As the number of iterations increase, the value of ξ will increase, and an accurate clustering result can be acquired and so can layered clustering. Further, from formula (9), it can be seen that an increasing number of iterations will lead to a weaker strength of φ in adjustment and higher weight value updating. Select the properties playing a bigger role in the clustering result and ignore the properties playing a lesser role to decrease the number of dimensions of clustering as this will reduce the workload of the calculation and further enhance the algorithm's convergence speed and efficiency.

The work in [44] prudently demonstrates the convergence of SFLA algorithm by applying the Markov mathematical analysis model. The DNSFLA algorithm in this paper ensures that the SFLA algorithm has a higher capacity of overall searching for the best solution and convergence speed through introducing niche technology to maintain the solutions' diversity. It prevents the SFLA algorithm from becoming "premature" after the local search is done for a certain number of iterations. In the process of updating the worst solution, the calculation of the length of movement is improved by using formula (17). At the initial period of evolution, the length factor is made smaller to expand the search scope, forcing the algorithm to search for better space for viable solution at a faster pace. With the increase of the number of evolutions, the length factor will gradually grow and the overall search capacity of the solution will be enhanced, which is beneficial to a frog to leap out of the partial best, and hence speed up the algorithm's convergence speed. Both the proposed DSOM and DNSFLA have a wide range of real application scenarios. For reference, the rudiment example of formulas included in the DSOM algorithm can be found in [45, 46], and the rudiment example of formulas included in the DNSFLA algorithm can be found in [47, 48]. To make it easier to follow the formulas proposed in this section, we still take the application of Smart Home described in reference [37] as an example. Formula (9) is used to calculate the value of growing threshold, which reflects the precision of the control command clustering. In this

practical scenario, there is diversity upon control commands on different smart appliances from different families. Formula (16) is used to calculate the distance metric of two objective resources in the scenario, which suppresses the convergence due to collaboration of control command groups, enhances the global optimization ability of the algorithm, and improves the convergence speed under current conditions. Besides, the role of formula (17) is to lead the control commands of company employees to move towards the direction of optimal resource in DACIOT.

In terms of MSDGAS execution complexity, the number of the SOM algorithm iterations is t, the number of input nodes is N, the maximum number at each iteration is $N \log N$, the maximum space occupied is N, the overall time complexity after t iterations is $O(t \times N \log N)$, and the space complexity is $O(N)$. For the DNSFLA algorithm, the mixed iteration number is G, the solution number is F, run time complexity is $O(G \times F^2)$, and space complexity is $O(F)$. Hence, MSDGAS' execution time complexity is $O(\mathrm{Max}\{t \times N \log N, G \times F^2\})$. However, in the process of actual operation, $G \times F^2$ is greater than $t \times N \log N$ and N is greater than F; therefore, the time complexity of MSDGAS is $O(G \times F^2)$, and space complexity is $O(N)$.

5. Experiment and Analysis

The settings for the experiment environment are as follows: 12 sets of Inspur servers, each configured with a CPU of Intel (R) Xeon(R) E5-2603 v2 @ 1.80 GHz, 40 G memory, 1 T hard disk, CentOS-6.4; 4 sets of them are used for the access of cluster servers; and the remaining 8 sets are used for servers for mass requests at users end. The experiment involves placing 8 sets of Linux servers at different areas as the user's end to send requests. Each of the host machines imitates 500 users, respectively; the remaining 4 sets of Linux servers are placed in a computer lab in order to build up the DACIOT's server cluster. A long-distance control system is adopted to control the 8 host machines when they send service requests of different sizes and types, in order to imitate a mass access environment. In the following, the experiment parameters settings for MSDGAS and the results of the performance tests are discussed.

5.1. Discussion of Experiment Parameters. In MSDGAS, some experiment parameters are optimized based on the work in [49, 50]. The user requests in the experiment are simulated data generated by a developed computer program. Generally, the dimension for a request command in an actual situation ranges from 3 to 6, and the larger the request command dimension, the more difficult it is for the algorithm to implement and the easier to test the bottleneck of the algorithm. Here, we take the maximum dimension value of 6. Further, calculate the growth threshold φ and disturbance factor ξ according to formulas (9) and (10). As to the initial neighborhood radius, it reflects the clustering range of the request commands, and also the command classification is often very vague at the beginning, so the value σ_0 is bigger, but the value has little influence on the clustering result because a different initial neighborhood can obtain the same result after a different number of iterations. Here, take the value σ_0 as 12. The regulator α of learning rate LR reflects the ease of request command clustering because as the number of iterations increases, the size of α increases and the easier it is for the request command to find its category. In this regard, according to the characteristics of the simulated service request of users, α varies between 0.6 and 0.9, and further experiments are needed to test the α sensitivity influence on the MSDGAS strategy. Based on the work in [40, 50], the DNSFLA algorithm can obtain the global optimal solution before 500 hybrid iterations. Here, set the maximum number of hybrid iterations as $G = 500$ and the maximum number of local iterations as $N = 20$. The niche distance L and the largest movement step length $D_{\max}$ reflect the scope of the service request command to avoid invalid service resources, indicating the rate of obtaining the optimal-matching resource to a certain extent. Further experiments are needed to test their sensitivity influence on the MSDGAS strategy. In the experiments of choosing parameters, we use 8 clients impersonating different scales of users to generate the test data sets. Limited by the number of data center servers in the test experiment, data center network architecture is fixed, while the request commands are generated randomly. In order to avoid the influence of the initial solutions on the test results, we randomly generate initial solutions for MSGDAS in each test, and to reduce the randomness of the parameters settings, the MSDGAS strategy is independently executed 10 times in each experiment, recording the maximum, minimum, and average.

Table 2 presents the test results of parameter variation among regulator α, niche distance L, and largest movement step length $D_{\max}$. In Table 2, T is the trial number, F_{best} represents the optimal solution tested out in the experiments, F_{mean} represents the average solution, and F_{worst} represents the worst solution. It can be seen, respectively, from the experiments $T1$–$T4$, $T5$–$T9$, and $T12$–$T15$ that the three parameters, α, L, and $D_{\max}$, do not have a linear influence on the performance of the proposed algorithm, which also indicates that the experimental factors (the regulator, the niche distance, and the largest movement step length) have a certain comparative independence. The experiments $T1$–$T4$ are used to test the sensitivity of MSDGAS when the value of α varies. The result indicates that the average solution does not become better with the increasing value of niche distance, while the best value of which is obtained when the value of niche distance is 5. Similarly, the experiments $T5$–$T9$ are used to evaluate the impact of regulator α on MSDGAS sensitivity, and the result shows that the average solution reaches the best when the value of α is set to 0.75. Moreover, the result of experiments $T7$, $T10$, and $T11$ further demonstrates it and obtains a better performance in the condition that niche distance is set to 5 and regulator is set to 0.75. The effect of the parameter $D_{\max}$ (the largest moving step) on MSDGAS performance is evaluated by experiments $T12$–$T15$, and the result shows that it reaches a better solution when the $D_{\max}$ is set to 10 under the condition that the value of α is set to 0.75 and the value of L is set to 5. As a supplementary, when the

Table 2: The sensitivity analysis for MSDGAS.

T	α	L	D_{max}	F_{best}	F_{mean}	F_{worst}
1	0.6	4	8	3.8312	3.9126	3.9513
2	0.6	5	8	3.4243	3.6831	3.8257
3	0.6	6	8	3.6581	3.8105	3.9018
4	0.6	7	8	3.7917	3.8372	3.8924
5	0.65	5	8	3.2536	3.5918	3.8049
6	0.7	5	8	2.8313	3.9395	3.0782
7	0.75	5	8	2.8109	2.8192	2.8201
8	0.8	5	8	2.8981	2.9203	2.9316
9	0.85	5	8	3.2546	3.3189	2.3247
10	0.75	4	8	2.9385	3.0138	3.1412
11	0.75	6	8	3.2821	3.3231	3.3397
12	0.75	5	9	2.5597	2.5735	2.5783
13	0.75	5	10	2.2386	2.2387	2.2390
14	0.75	5	11	2.6105	2.6732	2.7218
15	0.75	5	12	2.9133	2.9779	3.1023
16	0.75	6	9	2.8849	2.8905	2.9021
17	0.75	6	10	2.6312	2.6329	2.6376
18	0.75	6	11	2.8914	2.9143	2.9198
19	0.7	5	10	2.6219	2.7346	2.7912
20	0.8	5	10	2.6831	2.7128	2.7254

value of α is set to 0.75 and L is set to 6 in the experiments $T11$ and $T16$–$T18$, it also shows that the proposed algorithm gets the optimal solution with the value of D_{max} being set to 10. However, the average optimal solution is inferior to that when L is assigned a value of 5. For experiments $T13$, $T19$, and $T20$, in the given conditions that parameter L and parameter D_{max} are all set to the best value, it tries to test the change of MSDGAS' optimal solution as the value of parameter α increases. The result indicates that the proposed algorithm reaches a better optimal solution when parameter α is set to 0.75, which is consistent with the results of experiments $T7$, $T10$, and $T11$. Based on the above analysis, the MSDGAS is able to obtain better performance when experiment parameters take $\alpha = 0.75$, $L = 5$, and $D_{max} = 10$.

5.2. Performance Tests of MSDGAS. Depending on the above experiment parameter settings, we start the performance tests. The next experiments are carried out using the following methods. In addition, without loss of generality, both algorithms are executed ten times independently to obtain the average statistical results in each experiment.

(1) Under the situation of increasing IoT access, compare the access success rate (ASR) of the users' service requests between MSDGAS and the other algorithm (WQoS [51]) and define ASR = number of access success/total number of access requests.

(2) Under the situation of increasing IoT access, compare the response delay rate (RDR) between MSDGAS and the current other algorithm and define RDR = time of response delay/time of normal response.

(3) Under the situation of increasing IoT access, compare the access blocking rate (ABR) between MSDGAS and the other algorithm and define ABR = blocked number of service requests of the same type/total number of service requests of the same type.

(4) Under the situation of increasing IoT access, compare the resource utilization rate (RUR) between MSDGAS and the other algorithm. The number of online access machines is set as κ, the time consumed by Number j access server when requesting access is AT_j, the real occupation time of Number j access server is OT_j, and then RUR is defined as $RUR = (1/\kappa)\sum_{j=1}^{\kappa}(AT_j/OT_j)$.

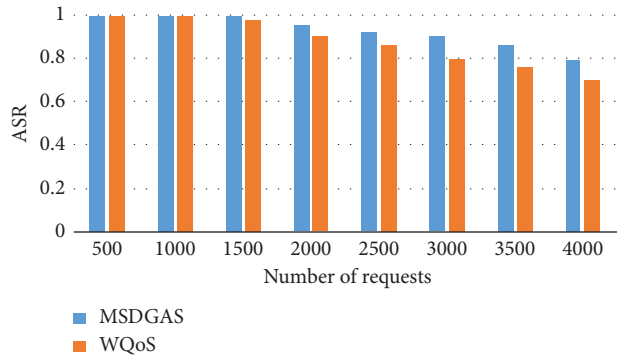

Figure 4: ASR with different access scales.

Experiment 1. Test ASR performance with the maximum number of requests being 4,000 and the minimum 500.

As shown in Figure 4, the horizontal coordinate represents the number of requests at DACIOT ranging from 500 to 4000 while the vertical coordinate represents the access success rate. It can be seen from the figure that when the access scale is within 1,500, both MSDGAS and WQoS can maintain smooth access. When the scale of command requests is between 500 and 1000, due to the smaller size of the requests, the access cluster computing ability and resource service ability are not under any pressure, and both algorithms show good service performance, without the occurrence of requests loss as a result. When requests number reaches 1500, the ASR of MSDGAS is 100% and the ASR of WQoS algorithm is 98.6%. However, when the access scale is over 1,500, WQoS starts to show partial access failure while MSDGAS can still maintain full access. When the request scale is 2000, the access success rate of MSDGAS is 95.8%, while WQoS only has a access success rate of 90.4%, it shows the phenomenon of competition gradually, particularly, over high-quality system resources and ASR becomes lower and lower for both algorithms. With a rising service request, the network bears a larger load and network links start being blocked and requests loss begins at the data access center for Internet of things. Generally speaking, however, MSDGAS is more advantageous. Moreover, the ASR of the MSDGAS changes quite gently, which shows that MSDGAS has a good robustness.

Experiment 2. Test RDR performance with the maximum number of requests being 4,000 and the minimum 500.

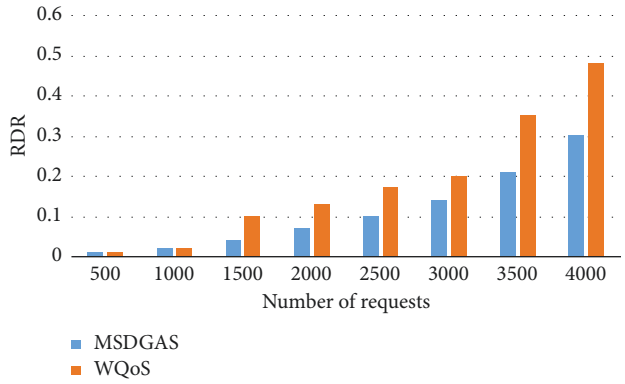

Figure 5: RDR with different access scales.

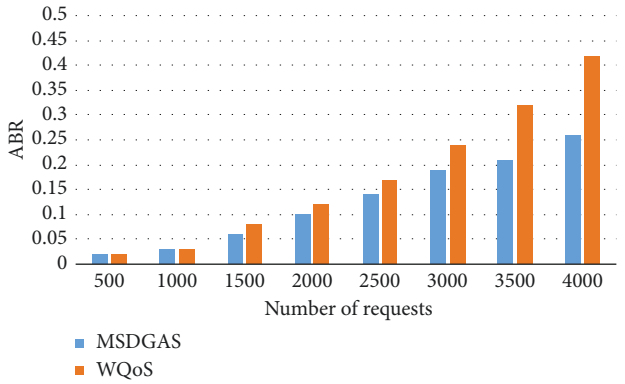

Figure 6: ABR with different access scales.

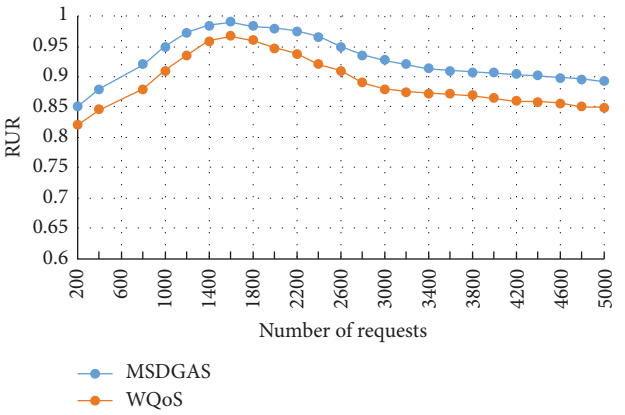

Figure 7: RUR with different access scales.

As shown in Figure 5, the vertical coordinate represents the response delay rate of requests at DACIOT while the horizontal coordinate represents the amount of access requests. It can be seen from the figure that when the access scale is within 1,000, both MSDGAS and WQoS only have a very low delay rate ranging from 1% to 2%. In the process of the request number gradually increasing from 500 to 1000, the figure shows that the pressure on each access server remains low and network link congestion does not appear as the access cluster enjoys ample network resources when the access service request number is low. Network response delay mainly refers to the transmission time at this stage. In this sense, the response delay of the two algorithms is all low and at similar levels. As the request scale grows and more resources are consumed, urgent requests have a more rigid demand for the response speed of DACIOT. In such scenarios, both algorithms show an increasing response delay rate. When the access service request number surpasses 1500, the two algorithms clearly start to differ from each other. With the ever-growing pressure from the access servers, the time to process the service request and network transmission gradually starts to increase, and the two algorithms start to differ from each other clearly in the response delay rate. Furthermore, when the number of access requests grows to 2,000, the RDR of MSDGAS is 7% while that of WQoS is 13%, nearly double that of MSDGAS, indicating that the MSDGAS algorithm is more stable in the condition of mass access requests.

Experiment 3. Test ABR performance with the maximum number of requests being 4,000 and the minimum 500.

As shown in Figure 6, the vertical coordinate represents the access blocking rate of requests at DACIOT while the horizontal coordinate represents the amount of access requests. When the service request number is below 1500, the access cluster server at the DACIOT can deal efficiently with the service requests and the DACIOT does not have pressure from network communication and resource consumption. It can be seen from Figure 6 that when the access scale is within 1,500, both MSDGAS and WQoS have a very low blocking rate, at around 3%, and overall, DACIOT does not appear to be under pressure. When access requests grow to 1,500, both algorithms perform differently—the former has a blocking rate of 6% and the latter 8%. With ever-growing service requests concurrency, the MSDGAS needs more network bandwidth and computing resource to cluster service requests into groups. When access requests surpass 1,500, the two algorithms show an obvious difference—the former has a lower blocking rate than the latter strictly. As the MSDGAS algorithm considers grouping the users' requests, it shows an advantage in the aspect of average network load. Furthermore, this advantage is increasingly apparent as the request scale grows.

Experiment 4. Test RUR performance with an increasing number of access requests at DACIOT.

As shown in Figure 7, the vertical coordinate represents the DACIOT's resource utilization rate for MSDGAS while the horizontal coordinate represents the number of access requests. At the beginning, the effective resources at DACIOT are not the bottleneck of the current service requests, network bandwidth resource is sufficient for both MSDGAS and WQoS algorithms. In other words, each server at the DACIOT has the ability to process service requests quickly. As the experimental results shows, the resource utilization rate of system is growing along with the increasing number of requests. When the access scale is around 1,600, both algorithms reach the highest RUR, and the MSDGAS algorithm is able to ensure that the DACIOT

maintains full utilization status. With the growth in the request scales and resource consumption, urgent requests have more rigid demands on the DACIOT's response speed and the real occupation time of each server becomes longer, causing partial access failure and a decline in RUR. However, from the decreasing curve trend, it can be seen that the MSDGAS algorithm can ensure that the DACIOT maintains a higher RUR than WQoS and the system can maintain reasonably good stability, and with the growing request scale, the system load can be effectively controlled. When the scale of access requests exceeds 4,000, RUR is inclined to be stable, indicating that the MSDGAS algorithm has better convergence.

However, when facing a more realistic workload where the number of requests varies over time, the MSDGAS will show little difference compared to the simulation experiments. This paper uses the increasing request scale to identify the performance limit of the algorithms under the condition of resource constraints. In a real scenario, the number of requests is disorderly and cannot be controlled accurately, making it difficult to measure the bottleneck of algorithms in this way [52]. Namely, it shows a bigger value at a point in time but a smaller value at the next time. Thus, the results shown in Figure 7 will be alternately high and low, but on some level, this situation is included in the above simulated experiments when the number of requests varies over time with a specific trend. In conclusion, it can be understood from the above analysis that, compared with the other algorithms, MSDGAS is able to maintain better service quality under the scenario of mass access requests. It works to dynamically group the users' requests and comprehensively weigh the performance of DACIOT in order to search for the best matching resources and maintain the maximum request load that the current network can bear. Finally, it shows an advantage over other algorithms in terms of access success rate, response rate, blocking rate, and resource utilization rate.

6. Conclusions and Future Work

In this paper, we formally defined a dynamic grouping resource allocation problem based on the background of mass control commands requesting for network access. In order to solve this problem, a novel resource access model was designed and a multistage dynamic packet access strategy was proposed along with it. The proposed strategy first adopted an optimized dynamic self-organizing feature map to cluster users' service requests into different command groups and then applied the dynamic niche-based shuffled frog-leaping algorithm to quickly and accurately acquire the best matching resources for these command groups. The experiments show that the proposed strategy can enhance the access success rate and resource utilization of service requests and reduce the network blocking and response delay. In conclusion, it can effectively solve the mass requests access problem of Internet of Things.

As further work, there are a number of directions that can enhance access capabilities of the proposed strategy in the context of IoT:

(i) *Support for delay access*. The strategy proposed in this paper does not consider the delay access mechanism when users' requests fail. Hence, considering network congestion, the delay access of DACIOT after failure will be the focus of further research.

(ii) *Pretreatment mechanism with Fog and Edge computing*. As the increasing computing and storing capacity of mobile devices or sensors, Fog and Edge computing are emerging as attractive solutions to the problem of data processing in IoT. In the future, we would like to preprocess the massive service requests using Edge computing, at the stage of request commands clustering in MSDGAS.

(iii) *Dynamic resource scheduling and virtualization techniques*. In DACIOT, heterogeneous network and sensing resources have to be often shared with multiple applications or services with different and dynamic quality of service requirements. Therefore, in order to allow more flexible scheduling, future research studies can also consider and compare the performance of resource virtualization (as instances of network virtualization) and operating system level virtualization such as containers.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors gratefully acknowledge the support of the National Natural Science Foundation of China (61572325 and 60970012), Ministry of Education Doctoral Fund of Ph.D. Supervisor of China (Grant no. 20113120110008), Shanghai Key Science and Technology Project in Information Technology Field (14511107902 and 16DZ1203603), Shanghai Leading Academic Discipline Project (no. XTKX2012), and Shanghai Engineering Research Center Project (GCZX14014 and C14001).

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT)–when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[3] R. Bonetto, N. Bui, V. Lakkundi, A. Olivereau, A. Serbanati, and M. Rossi, "Secure communication for smart IoT objects: protocol stacks, use cases and practical examples," in *Proceedings of the IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1–7, San Francisco, CA, USA, June 2012.

[4] C. Luca, D. D. Danilo, and M. Luca, "An IoT-aware architecture for smart healthcare systems," *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 515–526, 2015.

[5] F. Firouzi, A. M. Rahmani, and K. Mankodiya, "Internet-of-Things and big data for smarter healthcare: from device to architecture, applications and analytics," *Future Generation Computer Systems*, vol. 2018, pp. 583–586, 2018.

[6] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of internet of things for smart home: challenges and solutions," *Journal of Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[7] A. Al-Dweik, R. Muresan, M. Mayhew, and M. Lieberman, "IoT-based multifunctional scalable real-time enhanced road side unit for intelligent transportation systems," in *Proceedings of the IEEE 30th Canadian Conference on Electrical and Computer Engineering*, pp. 1–6, Windsor, ON, Canada, April-May 2017.

[8] T. Qu, S. P. Lei, Z. Z. Wang, D. X. Nie, X. Chen, and G. Q. Huang, "IoT-based real-time production logistics synchronization system under smart cloud manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 84, no. 1–4, pp. 147–164, 2016.

[9] K. M. Cheng, C. E. Tseng, and C. H. Tseng, "Maker and Internet of Things application in the intelligent security by example of power extension cord," in *Proceedings of the IEEE International Conference on Applied System Innovation (ICASI)*, pp. 239–241, Sapporo, Japan, May 2017.

[10] D.-Y. Kim and M. Jung, "Data transmission and network architecture in long range low power sensor networks for IoT," *Wireless Personal Communications*, vol. 93, no. 1, pp. 119–129, 2017.

[11] R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future internet: the internet of things architecture, possible applications and key challenges," in *Proceedings of the 10th International Conference on Frontiers of Information Technology (FIT)*, p. 257C260, Islamabad, Pakistan, December 2012.

[12] S. Andreev and Y. Koucheryavy, Internet of Things, Smart Spaces, and Next Generation Networking, in Lecture Notes in Computer Science, vol. 7469, Springer, Berlin, Germany, 2012.

[13] D. Singh, G. Tripathi, and A. J. Jara, "A survey of internet-of-things: future vision, architecture, challenges and services," in *Proceedings of the IEEE World Forum on Internet of Things (WF-IoT)*, pp. 287–292, Seoul, Korea, March 2014.

[14] Y. Meng and C. QingKui, "DCSACA: distributed constraint service-aware collaborative access algorithm based on large-scale access to the Internet of Things," *Journal of Supercomputing*, vol. 2018, pp. 1–20, 2018.

[15] G. Zhang and Q. K. Chen, "Group command transmission model based on effective path statistics network," *Chinese Journal of Electronics*, vol. 43, no. 9, pp. 1826–1832, 2015.

[16] G. Zhang and Q. K. Chen, "Construction and application of effective path statistics network," *International Journal of Hybrid Information Technology*, vol. 8, no. 9, pp. 369–380, 2015.

[17] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: automation networks in the era of the internet of things and industry," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 17–27, 2017.

[18] S. Y. Lien, K. C. Chen, and Y. Lin, "Toward ubiquitous massive accesses in 3GPP machine-to-machine communications," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 66–74, 2011.

[19] N. Ye, Y. Zhu, R. C. Wang, R. Malekian, and L. Min, "An efficient authentication and access control scheme for perception layer of internet of things," *Applied Mathematics and Information Sciences*, vol. 8, no. 4, pp. 1617–1624, 2014.

[20] F. Tao, Y. Zuo, X. L. Da, and L. Zhang, "IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1547–1557, 2014.

[21] B. Xu, X. L. Da, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in IoT-based information system for emergency medical services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.

[22] L. Li, S. Li, and S. Zhao, "QoS-aware scheduling of services-oriented internet of things," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1497–1505, 2014.

[23] J. Huang, D. Du, Q. Duan et al., "Modeling and analysis on congestion control in the internet of things," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 434–439, Sydney, NSW, Australia, June 2014.

[24] I. Awan and M. Younas, "Towards QoS in internet of things for delay sensitive information," in *Proceedings of the International Conference on Mobile Web and Information Systems*, pp. 86–94, Paphos, Cyprus, August 2013.

[25] G. Zhang, L. Cong, E. Ding, K. Yang, and X. Yang, "Fair and efficient resource sharing for selfish cooperative communication networks using cooperative game theory," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1–5, Kyoto, Japan, June 2011.

[26] G. Wei, A. V. Vasilakos, and Y. Zheng, "A game-theoretic method of fair resource allocation for cloud computing services," *Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, 2010.

[27] D. E. Charilas and A. D. Panagopoulos, "A survey on game theory applications in wireless networks," *Computer Networks*, vol. 54, no. 18, pp. 3421–3430, 2010.

[28] L. Song, D. Niyato, Z. Han, and E. Hossain, "Game-theoretic resource allocation methods for device-to-device communication," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 136–144, 2014.

[29] J. Jin, J. Gubbi, and T. Luo, "Network architecture and QoS issues in the internet of things for a smart city," in *Proceedings of the IEEE International Symposium on Communications and Information Technologies*, vol. 2012, pp. 956–961, Gold Coast, QLD, Australia, October 2012.

[30] F. Al-Turjman, "QoS—aware data delivery framework for safety-inspired multimedia in integrated vehicular-IoT," *Computer Communications 2018*, vol. 121, 2018.

[31] D. Guinard, V. Trifa, and F. Mattern, "From the internet of things to the web of things: resource-oriented architecture and best practices," in *Architecting the Internet of Things*, , pp. 97–129, 2011.

[32] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[33] Y. Zhang, X. Sun, and B. Wang, "Efficient algorithm for k-barrier coverage based on integer linear programming," *China Communications*, vol. 13, no. 7, pp. 16–23, 2016.

[34] G. J. Woeginger, *Exact Algorithms for NP-Hard Problems: A Survey*, Optimization Eureka, You Shrink!, Springer, Berlin, Heidelberg, Germany, 2003.

[35] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[36] X. Li, J. Luo, M. R. Chen, and N. Wang, "An improved shuffled frog-leaping algorithm with extremal optimisation for continuous optimisation," *Information Sciences*, vol. 192, pp. 143–151, 2012.

[37] M. Bansal, I. Chana, and S. Clarke, "Enablement of IoT based context-aware smart home with fog computing," *Journal of Cases on Information Technology*, vol. 19, no. 4, pp. 1–12, 2017.

[38] K. Tasdemir and E. Merenyi, "Exploiting data topology in visualization and clustering of self-organizing maps," *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 549–562, 2009.

[39] D. Sacha, M. Kraus, and J. Bernard, "SOMFlow: guided exploratory cluster analysis with self-organizing maps and analytic provenance," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 120–130, 2018.

[40] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[41] L. Wang and C. Fang, "An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem," *Inform Sciences*, vol. 181, no. 20, pp. 4804–4822, 2011.

[42] X. Liu, H. Liu, and H. Duan, "Particle swarm optimization based on dynamic niche technology with applications to conceptual design," *Advances in Engineering Software*, vol. 38, no. 10, pp. 668–676, 2007.

[43] H. Esmaeili-Najafabadi, M. Ataei, and M. F. Sabahi, "Designing sequence with minimum PSL using Chebyshev distance and its application for chaotic MIMO radar waveform design," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 690–704, 2017.

[44] J. P. Luo, X. Li, and M. R. Chen, "The markov model of shuffled frog leaping algorithm and its convergence analysis," *Dianzi Xuebao (Acta Electronica Sinica)*, vol. 38, no. 12, pp. 2875–2880, 2010.

[45] J. L. Wu, P. C. Chang, and C. C. Tsao, "A patent quality analysis and classification system using self-organizing maps with support vector machine," *Applied Soft Computing*, vol. 41, pp. 305–316, 2016.

[46] V. Caquilpan, D. Sáez, and R. Hernández, "Load estimation based on self-organizing maps and Bayesian networks for microgrids design in rural zones," in *Proceedings of the IEEE PES Innovative Smart Grid Technologies Conference-Latin America (ISGT Latin America)*, pp. 1–6, Quito, Ecuador, September 2017.

[47] S. Kayalvili and M. Selvam, "Hybrid SFLA-GA algorithm for an optimal resource allocation in cloud," in *Proceedings of the Cluster Computing*, pp. 1–9, Belfast UK, September 2018.

[48] Y. Miao, F. Rao, and L. Yu, "Research on the resource scheduling of the improved SFLA in cloud computing," *International Journal of Grid Distribution Computing*, vol. 8, no. 1, pp. 101–108, 2018.

[49] J. Faigl, M. Kulich, and V. Vonasek, "An application of the self-organizing map in the non-Euclidean Traveling Salesman Problem," *Neurocomputing*, vol. 74, no. 5, pp. 671–679, 2011.

[50] B. Amiri, M. Fathian, and A. Maroosi, "Application of shuffled frog-leaping algorithm on clustering," *International Journal of Advanced Manufacturing Technology*, vol. 45, no. 1-2, pp. 199–209, 2009.

[51] C. H. Wu, M. L. Chiang, and T. L. Lu, "Kernel mechanisms for supporting differentiated services and content-aware request distribution in web clusters providing multiple services," in *Proceedings of the IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 473–478, Biopolis, Singapore, March 2010.

[52] S. Verma, Y. Kawamoto, and Z. M. Fadlullah, "A survey on network methodologies for real-time analytics of massive IoT data and open research issues," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1457–1477, 2017.