Future Generation Computer Systems 115 (2021) 90–99

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Intelligent edge computing based on machine learning for smart city

Zhihan Lv [a,*], Dongliang Chen [a], Ranran Lou [a], Qingjun Wang [b,c]

[a] School of Data Science and Software Engineering, Qingdao University, Qingdao, China
[b] Shenyang Aerospace University, Shenyang 110136, China
[c] Nanjing University of Aeronautics and Astronautics, Nanjing, China

ARTICLE INFO

Article history:
Received 24 February 2020
Received in revised form 21 August 2020
Accepted 26 August 2020
Available online 5 September 2020

Keywords:
Machine learning
MEC
Artificial intelligence
Stackelberg principle-subordinate game theory
ADMM

ABSTRACT

To alleviate the huge computing pressure caused by the single mobile edge server computing mode as the amount of data increases, in this research, we propose a method to conduct calculations in a collaborative way. First, the method needs to consider how to encourage devices to cooperate when they are selfish. Second, the method answers the following question: how can collaborative computing be carried out when the device has the intention to cooperate? For example, how can calculations be conducted when there are extensibility and privacy problems in machine learning tasks? In view of the above challenges, a mobile edge server is taken as the focus, and the available resources around the mobile edge server are used for collaborative computing to further improve the computing performance of a mobile edge computing (MEC) system. The alternating direction multiplier method is used to solve the problem. First, the relevant techniques and theories of MEC, Stackelberg principle-subordinate game theory, and the alternating direction method of multipliers (ADMM) are introduced. Then, the problem description and model construction of distributed task scheduling in MEC and machine learning task-based device coordination computing are introduced, and machine learning is applied in the distributed task scheduling algorithm and distributed device coordination algorithm. Finally, the distributed task scheduling algorithm and distributed device coordination algorithm are tested by experiments.

1. Introduction

With the rapid development of Internet technology and network technology, the application of Internet of Things devices has been very extensive; and as the research continues to deepen, Internet of Things devices will become involved in increasingly more fields. The emergence of Internet of Things devices has greatly changed people's lifestyle, and the Internet of Things has greatly changed the ways in which information is collected, processed and shared [1,2].

Smart terminals are now ubiquitous, and new mobile applications (facial recognition, virtual enhancement, and natural language processing) are on the rise [3]. Traditional wireless cellular networks cannot meet the explosive growth of the network demand, especially in terms of high data rates and computing power. Obviously, running computation-intensive or delay-sensitive applications on IoT devices is incompatible with actual network capabilities. Because these new applications are often constrained by the computing resources and battery capacity of devices, they are spurring the development of IoT cloud platforms. Computing and analyzing complex tasks can be transferred to resource-rich cloud computing platforms, which contain a considerable amount of effective available resources [4]. In practice, IoT devices can improve battery, computing, and storage performance through ways such as purchasing these resources to break through hardware constraints. Based on the traditional centralized architecture of machine learning algorithms and tools, it is more important to use specialized servers with strong computing power and large storage capacities. Compared with MEC devices, MEC servers have greater advantages in terms of computing performance compared with cloud computing, which is more constrained in terms of its computing resources [5]. With the advent of the era of networked big data, the previous methods of storing data in memory for analysis are completely unable to meet the current needs. Today, with the massive increase in the amount of data, the development of distributed machine learning repositories is an effective way to solve the associated problems. Data analysis is sped up by scheduling multiple servers in parallel. The purpose of this approach is to achieve transparency between the client and the server; thus; data privacy issues are involved [6]. Based on the existing research, the MEC-based solution is adopted to realize the multidimensional consideration of the requirements, improve computing performance, and consider security and privacy.

* Corresponding author.
E-mail address: lvzhihan@gmail.com (Z. Lv).

https://doi.org/10.1016/j.future.2020.08.037
0167-739X/

To sum up, in order to improve the computing efficiency of an MEC system, a machine learning algorithm for a distributed device system is adopted. First, the relevant techniques and theories of MEC, Stackelberg principle-subordinate game theory, and ADMM are introduced. Then, the problem description and model construction of distributed task scheduling in MEC and machine learning task-based device coordination computing are introduced, and machine learning is applied in the distributed task scheduling algorithm and distributed device coordination algorithm. Finally, the distributed task scheduling algorithm and distributed device coordination algorithm are tested by experiments. It is expected that this research can provide a better idea for the development of cloud computing.

2. Literation review

MEC is the key technology among the next generation network technologies. MEC can solve the important calculations of task scheduling, content caching, collaborative processing and other problems in large-scale networks; thus, MEC has attracted the attention of relevant scholars in various countries since it was proposed [7]. Ren et al. (2018) proposed moving the provision of data computing and services from the cloud to the edge. Edge computing has emerged as a promising solution to address the limitations of cloud computing in supporting delay-sensitive and context-aware services in the IoT era [8]. Lv et al. (2019) proposed the MEC decision model and solution method based on the software defined network (SDN) and network functions virtualization (NFV) technologies in order to improve the stability of the mobile network system for the next generation of IoT, balance the network load, and ensure the quality of the user service experience. It is verified that the appropriate MEC center can be selected for multiattribute decision making using the SDN and NFV, which further reduce the server response time and improve the user service experience quality [9]. Chen et al. (2019) proposed a resource-efficient edge calculation method. Smart IoT device users can easily offload appropriate tasks on local devices, nearby ancillary devices, and nearby edge clouds to support their computationally intensive tasks. Unlike existing research on mobile computing unloading, a new perspective on resource efficiency is explored and an effective computing unloading mechanism is designed. The mechanism consists of a delay-aware task graph partitioning algorithm and an optimal virtual machine selection method to minimize the running time of smart IoT devices. The experimental performance evaluation confirms the effectiveness and superior performance of the proposed resource-efficient edge computing scheme [10]. Khelifi et al. (2018) explored the applicability of deep learning models (i.e., convolutional neural networks, recursive neural networks and enhanced learning) in combination with IoT devices and information-centric networks. The research seeks to assess the development trend of the network architecture in the future. Therefore, it is pointed out that the convolutional neural model can be used in the field of IoT, and data in complex environments can be reliably utilized [11]. In addition, enhanced learning and convolutional neural networks, which can be used to account for the multimodality of data in real-time applications, are integrated into IoT devices. Sangaiah et al. (2019) proposed a method of using machine learning technology to protect the location confidentiality of roaming PBS (position-based service) users in their study. The method identifies the user location by combining the decision tree and k-nearest neighbor values and estimates the user destination and location tracking sequence using the hidden Markov model. A mobile edge computing service strategy is followed to ensure the timely delivery of PBSs. The benefits of the mobile edge service strategy are location confidentiality and low latency, which are achieved through locating network and computing services near roaming users [12]. In their study, Kozik et al. (2018) pointed out that the increasing number of Internet of Things applications and network physical services poses a major challenge to network security. This article takes advantage of the flexibility of cloud-based architectures and recent advances in large-scale machine learning. By moving the operations that require more computing power and more storage to the cloud, edge computing can benefit from complex limit learning models built in advance through the cloud to effectively perform traffic classification [13]. In their research, Zhu et al. (2020) proposed a new set of design guidelines for wireless communication in edge learning, which are collectively referred as machine learning communication technology. Illustrative examples are provided to demonstrate the effectiveness of these design guidelines [14].

To sum up, MEC has very good performance and low delay, which can provide users with good network service experiences. However, with the development of network technology and large-scale data access, there is still the challenge related to the computing rate. Existing studies have made corresponding improvements to the MEC using artificial intelligence technology and have achieved good results. With the development of artificial intelligence, machine learning was proposed. At present, some scholars predict that machine learning has a good effect on improving MEC, but there is no corresponding performance test. In this research, machine learning is used to improve MEC, and the corresponding experimental performance is tested.

3. Introduction of relevant techniques

3.1. MEC

MEC refers to a new information technology that uses IT technology to provide services to mobile users close to the mobile wireless access network through cloud computing [15]. NFV, SDN, and cloud computing are important technologies in MEC. These technologies improve the data center servers installed next to the wireless base stations, which can compress huge amounts of data. It is estimated that a data computing center server next to a wireless base station can reduce is usage of wired broadband by 36%, which can reduce the network transmission delay of users by 49% [16].

MEC is a distributed computing form based on the characteristics of low latency, fast data processing, and intelligence [17]. Instead of transferring all data to a remote cloud data center, MEC filters the data uploaded from mobile devices, which means that data are analyzed and filtered at the user level. This processing method not only effectively saves time, but it also reduces the pressure on the core network, and effectively reduces the amount of data transferred from the device to the cloud [18,19]. The intelligence of MEC is reflected in the fact that it transfers computing and storage from the core network to the edge network, which can effectively reduce network latency. Furthermore, the network load, user location information, and other real-time effective information can be provided to the developers. When developers add real-time network information to applications and provide service content to mobile users during the R&D process, they can not only increase the satisfaction of mobile users but also improve the quality of the service experience of mobile users. Fig. 1 is the network structure diagram of MEC.

MEC technology, with its high responsiveness and low latency, facilitates access for local users and promotes IoT. In addition, MEC technology is closer to the user and can process information from different IoT devices at the same time to accelerate the development of new businesses. Therefore, MEC technology has good performance in promoting smart cities and big data analysis, making future cities more intelligent [20–24].

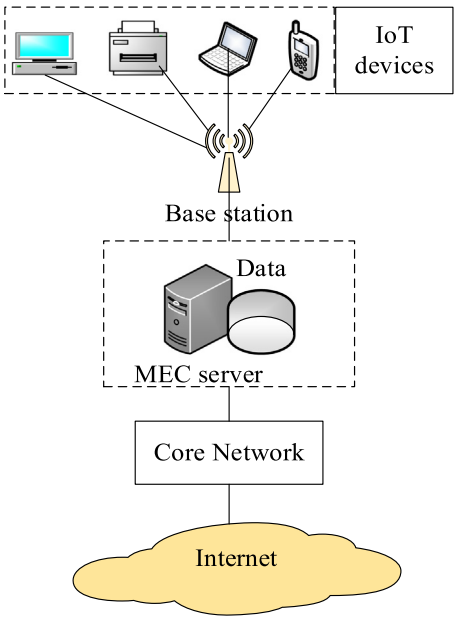


Fig. 1. MEC network structure diagram.

3.2. Stackelberg principle-subordinate game theory

Stackelberg game theory was put forward in the 1930s in order to solve the asymmetry of the subject status of both parties involved [22]. For a Stackelberg model, if there are two bodies, the roles of the two sides should be divided based on the difference in the status of the two sides. Assuming subject 1 is the leader and subject 2 is the follower, in the mathematical equation expression of a Stackelberg model, the independent variable of the leader is set as L_1, and then the policy choice of the followers themselves is set as L_2.

Then, the leader utility function can be expressed as Eq. (1).

$$\max_{L_1} \prod_1 (L_1, L_2) \tag{1}$$

The follower utility function can be expressed as Eq. (2).

$$\max_{L_2} \prod_2 (L_1, L_2) \tag{2}$$

In the Stackelberg model, the derivative of L2 is used in the follower's utility function, and then L2 is a function of L1 as $L_2 = V(L_1)$. Therefore, before the leader makes a decision, the leader knows that when he chooses strategy L1, the follower chooses strategy $L_2 = V(L_1)$. This allows the leader to obtain the water-containing form of L1: $\prod_1(L_1, V(L_1))$. Then, the derivative is taken and the optimal solution L_1^* is obtained. The optimal solution is input into $L_2 = V(L_1^*)$ to obtain L_2^*.

In a Stackelberg principle-subordinate game model, different subjects make rational decisions based on their own interests. There are often conflicts between the interests of leaders and followers, which objectively leads to the relative independence of participants.

In the long-term development of a network structure, leaders have a natural dominant position, which has a greater impact on the overall situation. The decisions of different roles will influence each other and eventually form a balanced decision acceptable to both leaders and followers [25–27].

3.3. ADMM

ADMM is a concise and efficient method suitable for solving a distributed large-scale convex optimization problem [28]. It is an advantageous solution for the distributed storage, large-scale volume of data, and high-dimensional data distribution problems in application scenarios. This algorithm can take advantage of machine learning, which allows the algorithm to be applied on a large scale. Therefore, ADMM can be applied in IoT device computing, artificial intelligence, logistics terminal management, finance, and other fields [29,30].

The mathematical model ADMM that we seek to solve is expressed as Eq. (3).

$$\begin{aligned} \min \quad & H(x) + K(y) \\ \text{s.t.} \quad & Ax + By = C \end{aligned} \tag{3}$$

In Eq. (3), $H(x)$ and $K(y)$ are convex functions, where $x \in X$; $y \in Y$; and the original variables X, Y are nonempty sets. *A, B, and C* represent the equality constraint matrixes.

The original variable is decomposed into two independent parts using the objective function. However, it needs to be in the constraint function and there needs to be variable coupling between the original variables. According to the characteristics of the augmented Lagrange multiplier algorithm, a penalty term is added to the objective function—the l_2 norm. Then, the objective function can be expressed as Eq. (4).

$$\begin{aligned} L_p(x, z, y) = {} & H(x) + K(y) + \lambda^T(Ax + By - C) \\ & + \frac{\rho}{2} \|Ax + By - C\|_2^2 \end{aligned} \tag{4}$$

In Eq. (4), $\lambda > 0$: Dual variable, and $\rho > 0$: Dual variable.

The introduction of the penalty term turns the primordial minimization problem into the problem of minimizing the primordial variables x, y, and λ in the alternating direction multiplier method. The solution process of this problem is shown as follows.

The original variables x, y and the dual variable λ are solved by $k + 1$.

$$x^{k+1} = \arg\min_x L_p(x, y^k, \lambda^k) \tag{5}$$

$$y^{k+1} = \arg\min_y L_p(x^{k+1}, y^k, \lambda^k) \tag{6}$$

$$\lambda^{k+1} = \lambda^k + \rho(Ax^{k+1} + By^{k+1} - C) \tag{7}$$

The specific iterative process of solving Eq. (3) using the augmented Lagrange multiplier method is as follows. The original variables x, y and the dual variable λ are solved by $k + 1$.

$$(x^{k+1}, y^{k+1}) = \operatorname*{argmin}_{x,y} L_\rho(x, y, \lambda_k) \tag{8}$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + By^{k+1} - C) \tag{9}$$

4. Distributed task scheduling in MEC and device coordination computing for machine learning tasks

4.1. Problem description and model construction of distributed task scheduling

Distributed task scheduling is an indispensable step in the execution of computing tasks. The MEC server wants as many edge computing devices as possible to coordinate and perform collaborative tasks, but the MEC server does not know if the edge computing device is willing to perform the services for it. When a selfish node edge computing device is encountered, the cost and overhead of the MEC server to perform the task increases, which also indicates that MEC servers and edge computing devices have different goals. Therefore, in the MEC task scheduling process, selfish node task scheduling will be encountered. To solve the

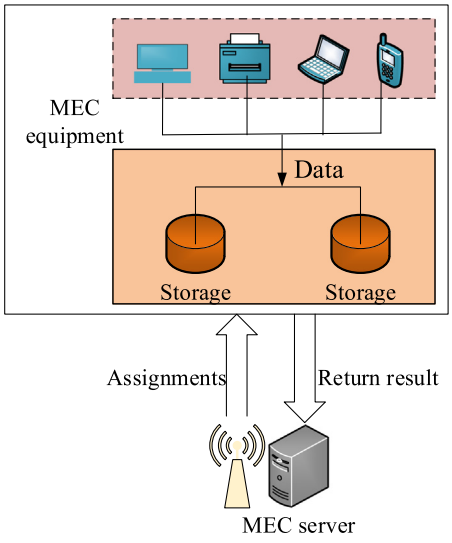


Fig. 2. Schematic diagram of task scheduling scenario.

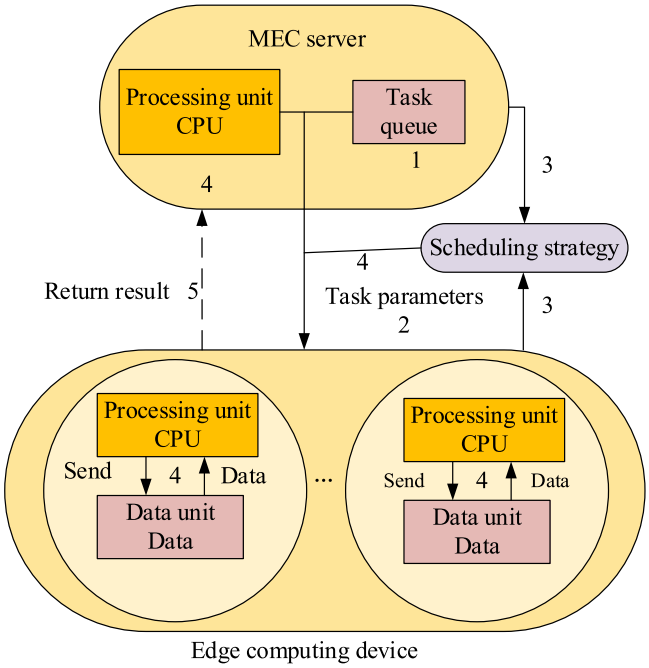


Fig. 3. Distributed task scheduling model.

problem of different goals between MEC servers and edge computing devices and to ensure the stability and good performance of the algorithm in the process of large-scale network use, distributed task scheduling is adopted to make scheduling policy decisions for MEC servers. Fig. 2 is a schematic diagram of the task scheduling scenario. An MEC server with computing and storage capacity is deployed at the base station, and multiple edge computing devices are connected to the network through the cellular base station of the wireless access point. The edge computing device has certain computing power and data resources. In the task scheduling process, the task scheduling policy can determine the number of tasks that the edge computing device can execute for the MEC server, and the MEC server and the edge computing device can jointly determine the final scheduling policy.

The construction of the task scheduling model accounts for an MEC server and N edge computing devices in the edge network. As the network further develops, increasing more edge computing devices will be generated. When the MEC server is connected to the wireless network by an optical fiber, the transmission delay generated can be ignored. However, the MEC server has limited processing capacity; thus, it cannot unload all devices efficiently. Therefore, it needs assistance from edge computing devices. When assisting a task, the edge computing device can only perform one task, and it will not perform a task for the MEC server unless it is a valid task scheduling policy. Given the selfishness of edge computing devices, the MEC server encourages edge computing devices to perform as many tasks as possible. Fig. 3 is the distributed task scheduling model built in this research. There is a task queue on the MEC server, which is the task queue that needs to be performed by the MEC server. A processing unit and a data unit are located at the edge of the edge computing device, and the data unit provides the processing unit with the data required for the computing task. The running process of the distributed task scheduling model is that MEC server selects a task from the task queue that needs to be executed and sends the relevant data of this task to the edge computing device. Then, the edge computing device and the MEC server together determine the scheduling policy.

4.2. Distributed task scheduling algorithm

The distributed task scheduling algorithm proposed in this research adopts Stackelberg game theory and ADMM. The leader can provide an incentive function to the followers, which can be expressed as follows.

$$\Phi_i(-\alpha_i x_i T_i + \beta_i E_i), \theta_i^* = L_i(x_i, \lambda) + \Phi_i^k(x_i) \tag{10}$$

In Eq. (10), i: A follower,
$L_i(x_i, \lambda)$: The function part of Lagrange, and
$\Phi_i^k(x_i)$: The independent part of followers after being inspired. The latter two can be respectively expressed as follows.

$$L_i(x_i, \lambda) = -\log(1 + x_i T) - \lambda \mu_i x_i T \tag{11}$$

$$\Phi_i^k(x_i) = -\alpha_i x_i T + \beta_i E_i - \theta_i^k x_i \tag{12}$$

In Eqs. (11) and (12), λ is the Lagrangian multiplier, and θ_i^k is the amount of work paid per unit.

In this research, according to the characteristics of Stackelberg game algorithm based on ADMM, a two-layer iterative optimization process of internal and external circulation is proposed. According to the internal and external circulation, the optimal scheduling strategy of the leader is obtained. The iterative process of ADMM is an internal circulation to solve the task scheduling policy provided by the follower side and find the best solution. The external circulation can ensure that the function of the leader side meets a standard and the optimal solution of the whole function can be obtained.

The internal update solution process of the leader can be expressed as follows.

$$\min L_i(x_i, \lambda^k(t)) + \Phi_i^k(x_i) + \frac{\rho}{2}\left\| \sum_{j=1}^{i-1} \mu_j x_j^k(t+1)T + \mu_i x_i T + \sum_{j=i+1}^{N} \mu_i x_j^k(t)T - C \right\|_2^2 \tag{13}$$

$$s.t. \quad f_i \le f_i^{\max}, \nabla 1 \le i \le N \tag{14}$$

In Eqs. (13) and (14), k: The number of external circulation iterations of the algorithm,
t: The number of internal circulation iterations of ADMM, and
ρ: The penalty factor.
In the algorithm, the large variable of the coordinate will use the latest iteration value of the small variable of the coordinate.

To ensure that the iteration results do not oscillate, all variables need to be updated sequentially.

The update of the Lagrange multiplier on the leader end is as follows.

$$L(\{x_i\}, \lambda) = \sum_{i=1}^{N} L_i(x_i, \lambda) + \lambda C + \frac{\rho}{2} \left\| \sum_{i=1}^{N} \mu_i x_i T - C \right\|_2^2 \tag{15}$$

The steps of Stackelberg game algorithm based on ADMM proposed in this research include two parts: external circulation and internal circulation. During the course of the external loop, the MEC server gets the parameter x^k, the excitation parameters are updated and θ^{k+1} is broadcast to each computing edge device until the stop criteria are met. In the internal loop, the MEC server will solve the dual function parameters through the task scheduling parameter x^k provided by the edge computing device and broadcast the data to each computing edge device.

4.3. Problem description and model construction of distributed devices for machine learning

Machine learning algorithms are used to solve convex optimization problems. Among the different machine learning algorithms, the stochastic gradient algorithm is one of the commonly used algorithms. The stochastic gradient descent method is effective for solving logistic regression and linear prediction problems. However, the stochastic gradient descent method has randomness and the convergence speed is relatively slow; therefore, the convergence of stochastic gradient descent method is the difficulty of its research at present. As the research progresses, some scholars have used MEC technology to connect various IoT devices to obtain useful information. The coordinated centralized processing of various IoT devices has good performance, but its scalability is normal in the case of a surge of IoT devices. In addition, MEC server resources are limited, so it is difficult for MEC to realize large-scale data storage. Moreover, the data produced by IoT devices may involve private information and sensitive information. In view of the problems experienced by the above two machine learning algorithms, the Least Absolute Shrinkage and Selection Operation (LASSO) is used to predict the regression problems in this research. Therefore, this article will provide a solution for distributed device collaborative learning using data collected by multiple IoT devices to train the LASSO regression model.

Fig. 4 shows the coordinated computing network in the MEC framework. In an MEC server, N isomorphic IoT devices are connected to perform common awareness tasks. IoT devices constantly generate sensor data and convert raw data into feature vectors in a limited amount of time. Each eigenvector corresponds to a response variable, and multiple predictive variables can be generated. In the whole system model, the task of the MEC server is to analyze the data and construct the relationship between the feature vector and the response variable, that is, the data samples of IoT devices connected to MEC servers are learned and modeled collaboratively.

In this study, the LASSO regression model will be adopted to predict the model. For the LASSO model, when the dataset $\{(x_i, y_i), i = 1, \ldots, N\}$ is given, the mathematical expression of the optimization problem that can be solved is as follows.

$$\min \frac{1}{2} \sum_{i=1}^{N} (\boldsymbol{w}^T \boldsymbol{x}_i + b - y_i)^2 + \lambda \|\boldsymbol{w}\|_1 \tag{16}$$

In Eq. (16), $w \in R^n$: The eigenvector;
$y_i \in R$: The response variable;
$b \in R$: Intercept; and
λ: The regular parameter, where $\lambda > 0$.

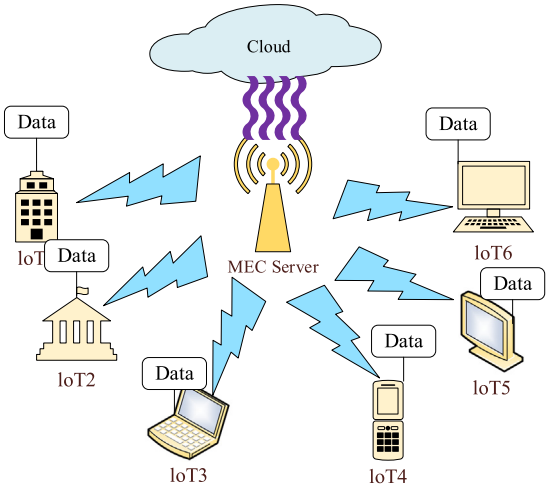


Fig. 4. Collaborative computing network in MEC framework.

Supposing that a set of eigenvectors $\boldsymbol{x} \in R^n$ are given, after training the LASSO model parameters (w, b), the equation of the response variable can be obtained as follows.

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b \tag{17}$$

Assuming that each device $i \in \{1, \ldots, N\}$ can produce a set of data samples $D_i = \{(x_{ij}, y_{ij}), j = 1, \ldots, M_i\}$ in an MEC system, among the variables, $x_{ij} \in \mathbb{R}^n$ is the eigenvector, $y_{ij} \in \mathbb{R}^n$ represents the response variable, and M_i represents the number of data samples contributed by distributed device i. Then, the expression of the minimum optimization problem can be expressed as follows.

$$\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M_i} (\mathbf{w}^T \boldsymbol{x}_{ij} + b - y_{ij})^2 + \lambda \|\mathbf{w}\|_1 \tag{18}$$

In Eq. (18), $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$.

4.4. Distributed device system algorithm

In this research, a distributed optimization technique ADMM is proposed to solve the LASSO regression. Although ADMM can realize distributed computing with decomposable variables, there is a coupling problem between the variables in Eq. (3). Therefore, ADMM cannot be directly applied in Eq. (3), and auxiliary variables need to be introduced. $\{(w_i, b_i), i = 1, \ldots, N\}$ is introduced to convert Eq. (18) into Eq. (19).

$$\begin{aligned} &\min \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M_i} (\mathbf{w}_i^T \boldsymbol{x}_{ij} + b_i - y_{ij})^2 + \lambda \|\mathbf{w}\|_1 \\ &s.t. \quad \mathbf{w}_i = \mathbf{w}, b_i = b, i = 1, \ldots, N. \end{aligned} \tag{19}$$

Eq. (19) is the ADMM algorithm after introducing the auxiliary variables. $\{(\mathbf{w}, b)\}$ is viewed as the global model parameters of the MEC server, and then $\{(\mathbf{w}_i, b_i), i = 1, \ldots, N\}$ are the local model parameters for each IoT device i controlled by the MEC server.

By introducing auxiliary variables, the model problem of distributed task scheduling can be changed into global model parameters $\{(\mathbf{w}, b)\}$ and local model parameters $\{(\mathbf{w}_i, b_i), i = 1, \ldots, N\}$. The distributed collaborative algorithm design for ADMM is as follows. First, Eq. (19) is expressed in an augmented Lagrangian

form.

$$L_p(\xi, \psi, \zeta) = \min \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\mathbf{w}_i^T \boldsymbol{x}_{ij} + b_i - y_{ij})^2 + \lambda \|\mathbf{w}\|_1 + \sum_{i=1}^{N}((\mathbf{w}_i - \mathbf{w})^T \xi_{i,w} + \zeta_{i,b}(b_i - b)) + \sum_{i=1}^{N}\frac{\rho}{2}((\mathbf{w}_i - \mathbf{w})^T(\mathbf{w}_i - \mathbf{w}) + (b_i - b)^2) \tag{20}$$

Then, variables ξ, ψ, ζ are defined respectively.

$$\xi = \{(\mathbf{w}, \mathrm{b})\} \tag{21}$$

$$\psi = \left\{(\mathbf{w}_j, b_j), \mathrm{i} = 1, \ldots, N\right\} \tag{22}$$

$$\zeta = \left\{\xi_{i,w}, \zeta_{i,b}\right\} \tag{23}$$

The newly defined variable is the dual variable associated with the equality constraint of Eq. (20). ξ, ψ, ζ are updated in the corresponding order. During the iteration, the update operation is divided into three steps.

Step 1: ξ is updated. ξ is updated by solving the following expression problem.

$$\min_{\zeta} \lambda \|\mathbf{w}\| + \frac{\rho N}{2}\mathbf{w}^T\mathbf{w} - \rho\mathbf{w}^T\sum_{i=1}^{N}\mathbf{w}_i^t - \mathbf{w}^T\sum_{i=1}^{N}\xi_{i,w}^t + \frac{\rho N}{2}b^2 - \rho b\sum_{i=1}^{N}b_i^t - b\sum_{i=1}^{N}\zeta_{i,b}^t \tag{24}$$

$\overline{\mathbf{w}}$ is used to represent the mean of the local model parameter $\mathbf{w}_i$ vector. Eq. (24) is converted into Eq. (25).

$$\min_{\zeta} \lambda \|\mathbf{w}\| + \frac{\rho}{2}N\mathbf{w}^T(\mathbf{w} - 2\overline{\mathbf{w}}^t - \frac{2\overline{\zeta}_w^t}{\rho}) + \frac{\rho N}{2}b(b - 2\overline{b}^t - \frac{2\overline{\zeta}_b^t}{\rho}) \tag{25}$$

The equation for solving Eq. (25) is shown below.

$$\mathbf{w}^{t+1} = \begin{cases} \overline{\mathbf{w}}^t + \overline{\xi}_w^t - \dfrac{\lambda}{\rho N}, & \overline{\mathbf{w}}^t + \dfrac{\overline{\xi}_w^t}{\rho} > \dfrac{\lambda}{\rho N} \\ 0, & \overline{\mathbf{w}}^t + \dfrac{\overline{\xi}_w^t}{\rho} \in \left[-\dfrac{\lambda}{\rho N}, \dfrac{\lambda}{\rho N}\right] \\ \overline{\mathbf{w}}^t + \overline{\xi}_w^t - \dfrac{\lambda}{\rho N}, & \overline{\mathbf{w}}^t + \dfrac{\overline{\xi}_w^t}{\rho} < -\dfrac{\lambda}{\rho N} \end{cases} \tag{26}$$

$$b^{t+1} = \overline{b}_t + \frac{\overline{\xi}_b^t}{\rho} \tag{27}$$

Step 2: ψ is updated. The problems to be solved are as described in Eq. (13).

$$\min_{\psi} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M_i}(\mathbf{w}_i^T\boldsymbol{x}_{ij} + b_i - y_{ij})^2 + \sum_{i=1}^{N}\frac{\rho}{2}\mathbf{w}_i^T(\mathbf{w}_i - 2\mathbf{w}^{t+1} + \frac{2\zeta_{i,w}^t}{\rho}) + \sum_{i=1}^{N}\frac{\rho}{2}\boldsymbol{b}_i(\boldsymbol{b}_i - 2\boldsymbol{b}^{t+1} + \frac{2\zeta_{i,b}^t}{\rho}) \tag{28}$$

According to the method proposed in this research, N subproblems in all IoT devices can be solved. For each device, only local subproblems of device i need to be solved, which can be expressed as follows.

$$\min_{\psi_i} \frac{1}{2}\sum_{j=1}^{M_i}\mathbf{w}_i^T\boldsymbol{x}_{ij} + b_i - y_{ij} + \sum_{i=1}^{N}\frac{\rho}{2}\mathbf{w}_i^T(\mathbf{w}_i - 2\mathbf{w}^{t+1} + \frac{2\zeta_{i,w}^t}{\rho}) + \sum_{i=1}^{N}\frac{\rho}{2}\boldsymbol{b}_i(\boldsymbol{b}_i - 2\boldsymbol{b}^{t+1} + \frac{2\zeta_{i,b}^t}{\rho}) \tag{29}$$

Step 3: ζ is updated. In this research, the obtained ξ^{t+1}, ζ^{t+1} will be used to update the dual variables. Eqs. (30) and (31) are obtained.

$$\xi_{i,w}^{t+1} = \xi_{i,w}^t + \rho(\mathbf{w}_i^{t+1} - \mathbf{w}_i^{t+1}) \tag{30}$$

$$\zeta^{t+1} = \zeta_{i,b}^t + \rho(b_i^{t+1} - b_i^{t+1}) \tag{31}$$

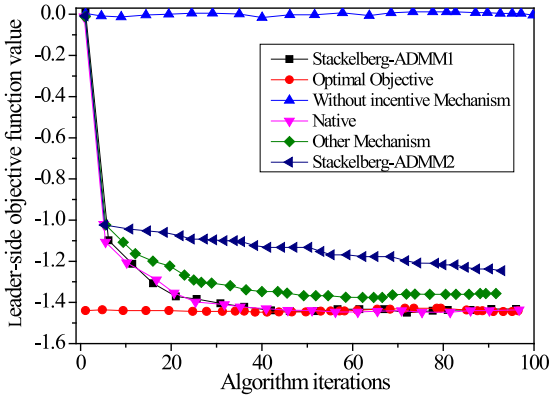


Fig. 5. The number of iterations of the leader objective function with or without a drastic mechanism.

5. Performance analysis of distributed scheduling algorithm and distributed device coordination algorithm

5.1. Performance test of distributed task scheduling algorithm

Fig. 5 shows the changes in the leader's objective function with or without a drastic mechanism as the number of iterations increases. The excitation function of the Stackelberg game method based on ADMM is $\theta_i^k = \nabla_{x_i}(-\alpha_i x_i^k T + \beta_i E_i)$, and the excitation function $\theta_i^{k+1} = \theta_i^k \pm \Delta$ is used as the naive method. Δ is a specific value. It can be observed that the Stackelberg game method based on ADMM can converge rapidly while the naive method cannot converge before 100 iterations. The reason for this phenomenon is that the excitation function of the Stackelberg game method based on ADMM changes in the direction of the tangent line of the follower end. The change rate is faster and more accurate. The variation of the naive incentive mechanism is a fixed value that changes in different directions according to the positive and negative derivatives of the follower. The performance of the Geely function in the Stackelberg game algorithm based on ADMM is better than that of other excitation functions.

Fig. 6 is the comparison diagram of different stop criteria. As ε_0 decreases, the value of the objective function of the leader is always decreasing. When $\varepsilon_0 = 10^{-5}$, the objective function value of the leader is the lowest. Therefore, in order to ensure the persuasion of the experiment, $\varepsilon_0 = 10^{-5}$ is adopted as the stop criterion in the other experiments.

Fig. 7 shows the influence of different numbers of followers on the performance of the algorithm. The number of different

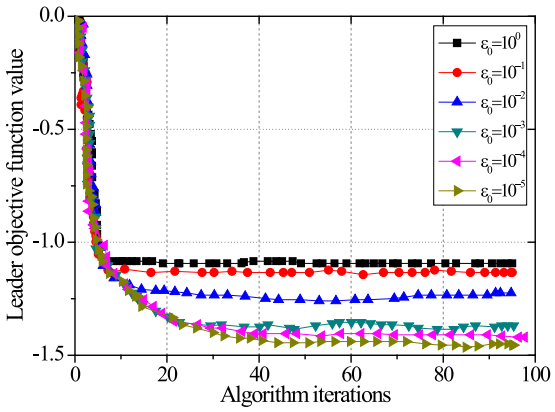


Fig. 6. Comparison of different stop criteria.

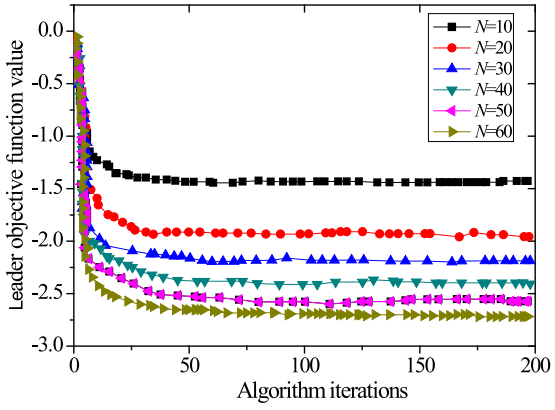


Fig. 7. A comparison of the number of different followers.

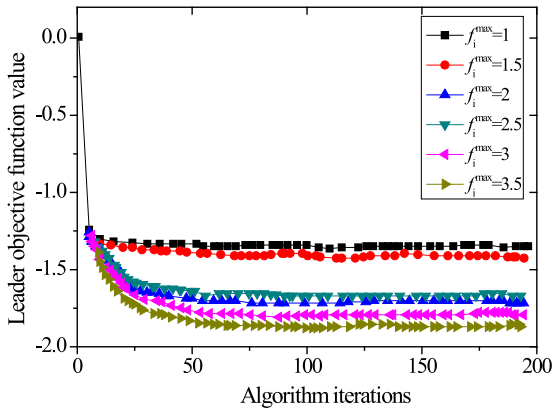


Fig. 8. Comparison of the convergence of algorithms with different maximum frequencies.

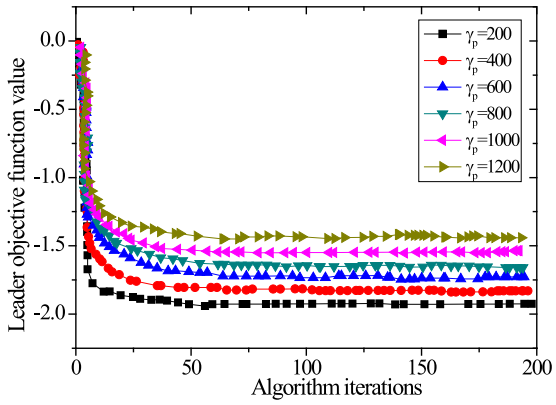


Fig. 9. The comparison the different task processing densities based on the number of iterations.

followers N is set to 10, 20, 30, 40, 50 and 60, respectively. As the number of followers increases, the value of the objective function on the leadership side further decreases. This is because more followers mean more tasks, and so more data need to be provided by the leader. Though leaders need to provide more data to their followers, more is not always better. As the scale of the network increases, the network costs will increase, and the number of iterations will change. The iteration times of the leader-side objective function are also different depending on the number of followers. The proposed ADMM-based Stackelberg game algorithm enables IoT devices to be used in large-scale network scenarios.

Fig. 8 compares the convergence of the algorithm with different maximum frequencies. At the same processing density, as the maximum frequency increases, the objective function of the leader side continuously decreases, but it does not decrease infinitely. When $f_i^{\max}$ changes from 1.5 to 2, there is a big drop. The drop after 2 continuously decreases. The results indicate that the performance of edge computing devices can be used to help leaders as much as possible, which also indicates that the performance of edge computing devices is limited.

Fig. 9 is the comparison diagram of number of iterations with different task processing densities. In this research, six groups of different iterative task processing densities are selected. At the same frequency, the higher the density of the task that is processed, the lower the number of tasks assisted by edge computing devices, and the larger the objective function of the leader, which shows that edge computing is not able to deal with very complex data.

5.2. Performance test of distributed device coordination algorithm

Fig. 10 shows the convergence effect of the objective function. The distributed algorithm based on ADMM proposed in this research can obtain the global optimal solution of the objective function after 40 iterations. By using the centralized algorithm, according to the existing convergence criterion, the optimal solution after convergence is obtained after 55 iterations. As concluded from the changes of the original and dual residuals in Fig. 10 with the number of iterations, as the number of iterations changes, the original and dual residuals are always changing and do not stay the same.

In this research, the convergence speed of the algorithm is tested. To make the data more scientific and rigorous, three groups of 10 independent repeated experiments are conducted. The cumulative distribution diagram of the algorithm under three experiments is obtained, as shown in Fig. 11. In the distributed algorithm based on ADMM proposed in this research, regarding the number of iterations of the three groups of experiments, in the best case, the algorithm converges after 20 iterations; and in the worst case, convergence is achieved after approximately 150 iterations. For the centralized algorithm, in the three experiments, the fastest one needs approximately 600 iterations for the algorithm to converge; and in the worst case, it takes 800 iterations to achieve convergence. Therefore, according to the convergence speed of the algorithm, the distributed algorithm based on ADMM proposed in this research is obviously superior to the centralized algorithm.

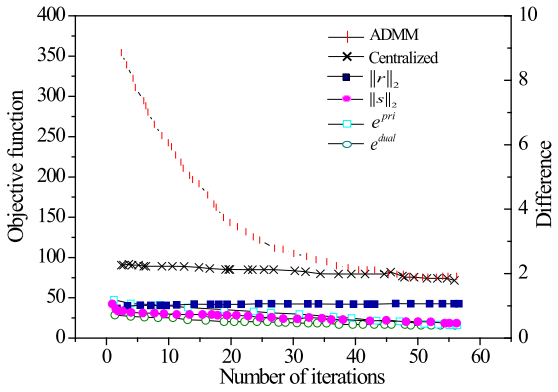


Fig. 10. The convergence effect of the objective function.

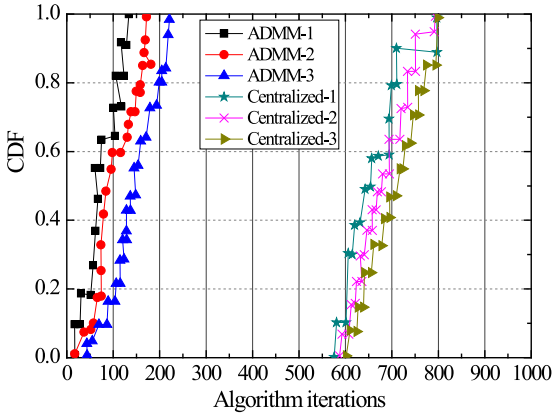


Fig. 11. The cumulative distribution of the algorithm.

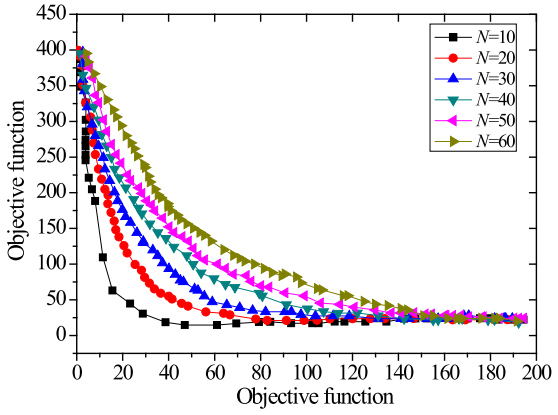


Fig. 12. The effect of N on algorithm convergence.

The algorithm proposed in this research is influenced by N, ρ, λ and other parameters. To study the effect of N, ρ, and λ on the convergence of the algorithm, the values of these three parameters are set differently. The performance of the algorithm is observed by looking at the values of parameters such as N, ρ, and λ. When assessing the value of a parameter, we ensure that the other parameters remain unchanged. The influences of N, ρ, and λ on the convergence performance of the algorithm are shown in Figs. 12∼14.

It can be concluded from Fig. 12 that, regardless of the number of IoT devices, the algorithm proposed in this research can obtain the same optimal target value under any conditions; and as N increases, the algorithm proposed in this research can converge

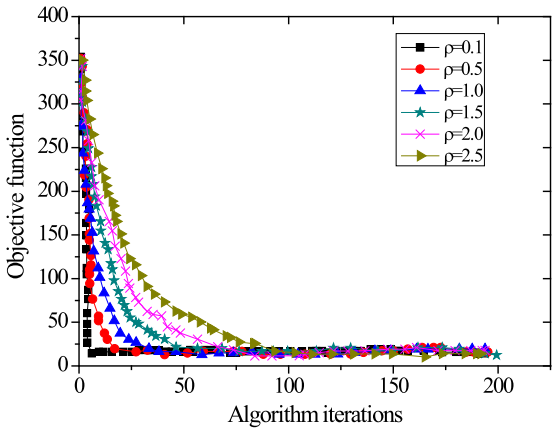


Fig. 13. The effect of ρ on algorithm convergence.

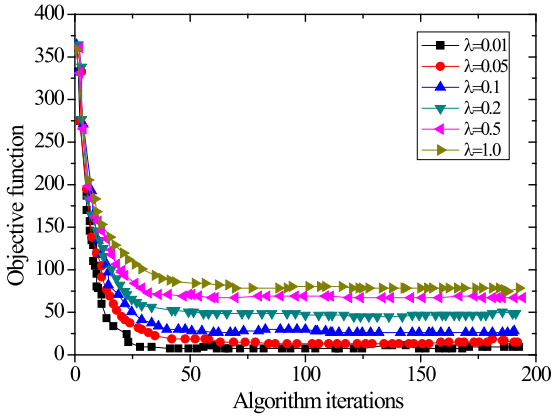


Fig. 14. The effect of λ on algorithm convergence.

through a certain number of iterations. Therefore, the proposed algorithm is scalable and the overhead of IoT devices is low. Similarly, the value of ρ in Fig. 13 does not affect the optimal result of the objective function. However, the smaller the value of ρ is, the faster the rate of convergence will be. λ in Fig. 14 optimizes the desired target, but it does not have much effect on the rate of convergence or the convergence.

In the end, the mean square error and the corrected R^2 are used to compare the performance of the model. Fig. 15 shows the influence of N on the MSE and the adjusted R^2. As the value of N increases, the MSE increases, and the adjusted R^2 decreases, which indicate that IoT devices with different amounts of data may have different data training results. The distributed algorithm's MSE is always 0.01 while the centralized algorithm's MSE is always 0.981. According to Fig. 15, the distributed algorithm can obtain the optimal state.

6. Conclusions

To improve the performance of MEC, a machine learning method is applied to MEC technology. According to the MEC device scenario, the task scheduling problem of mobile edge device is proposed. Based on the description and model construction of the distributed task scheduling problem and machine learning-oriented distributed devices, a distributed task scheduling algorithm based on an ADMM Stackelberg game algorithm and a distributed device system algorithm based on ADMM are proposed. Finally, the performance of the two algorithms proposed in this research is tested. The test results show that the

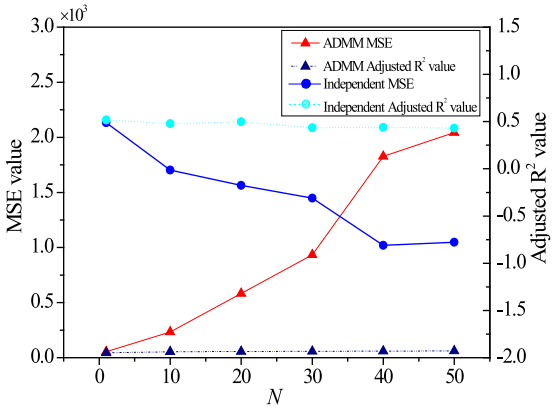


Fig. 15. The effect of N on the MSE and corrected R^2.

Stackelberg game algorithm based on ADMM can not only realize fast algorithm convergence, but it can also achieve good stability in a large scale network. The distributed coordination algorithm based on ADMM proposed in this research can achieve fast convergence and close to the optimal performance in intelligent edge computing, it has good scalability and it can avoid losing the diversity of the data.

The research provides a new idea for the development of intelligent MEC and promotes the development of IoT technology. However, there are still some limitations in this study. This study only considers changes in an MEC server. When there are multiple MEC servers, their scheduling policies may change. Therefore, subsequent studies can be carried out to consider multiple MEC servers and expand the depth of this study. In the future, this research can improve other previous studies in big data, IoT and AI [31–40].

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

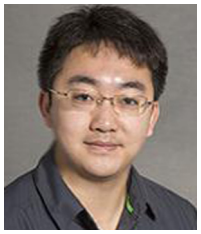
Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 61902203, Key Research and Development Plan - Major Scientific and Technological Innovation Projects of ShanDong Province (2019JZZY020101).

References

[1] M. Chen, W. Li, Y. Hao, et al., Edge cognitive computing based smart healthcare system, Future Gener. Comput. Syst. 86 (2018) 403–411.
[2] H. Li, K. Ota, M. Dong, Learning IoT in edge: Deep learning for the Internet of Things with edge computing, IEEE Netw. 32 (1) (2018) 96–101.
[3] A. Munir, P. Kansakar, S.U. Khan, IFCIoT: Integrated Fog Cloud IoT: A novel architectural paradigm for the future Internet of Things, IEEE Consum. Electron. Mag. 6 (3) (2017) 74–82.
[4] Y. Liu, C. Yang, L. Jiang, et al., Intelligent edge computing for IoT-based energy management in smart cities, IEEE Netw. 33 (2) (2019) 111–117.
[5] Z. Zhou, H. Liao, B. Gu, et al., Robust mobile crowd sensing: When deep learning meets edge computing, IEEE Netw. 32 (4) (2018) 54–60.
[6] A. Abeshu, N. Chilamkurti, Deep learning: the frontier for distributed attack detection in fog-to-things computing, IEEE Commun. Mag. 56 (2) (2018) 169–175.
[7] L. Hu, Y. Miao, G. Wu, et al., IRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing, Future Gener. Comput. Syst. (2019) 569–577.
[8] J. Ren, Y. Pan, A. Goscinski, et al., Edge computing for the internet of things, IEEE Netw. 32 (1) (2018) 6–7.
[9] Z. Lv, W. Xiu, Interaction of Edge-Cloud Computing based on SDN and NFV for Next Generation IoT, IEEE Internet Things J. (2019).
[10] X. Chen, Q. Shi, L. Yang, et al., Thriftyedge: Resource-efficient edge computing for intelligent IoT applications, IEEE Netw. 32 (1) (2019) 61–65.
[11] H. Khelifi, S. Luo, B. Nour, et al., Bringing deep learning at the edge of information-centric Internet of Things, IEEE Commun. Lett. 23 (1) (2018) 52–55.
[12] A.K. Sangaiah, D.V. Medhane, T. Han, et al., Enforcing position-based confidentiality with machine learning paradigm through mobile edge computing in real-time industrial informatics, IEEE Trans. Ind. Inf. 15 (7) (2019) 4189–4196.
[13] R. Kozik, M. Choraś, M. Ficco, et al., A scalable distributed machine learning approach for attack detection in edge computing environments, J. Parallel Distrib. Comput. 119 (2018) 18–26.
[14] G. Zhu, D. Liu, Y. Du, et al., Toward an Intelligent Edge: Wireless Communication Meets Machine Learning, IEEE Commun. Mag. 58 (1) (2020) 19–25.
[15] Y. Dai, D. Xu, S. Maharjan, et al., Artificial intelligence empowered edge computing and caching for internet of vehicles, IEEE Wirel. Commun. 26 (3) (2019) 12–18.
[16] Y. Liu, C. Yang, L. Jiang, et al., Intelligent edge computing for IoT-based energy management in smart cities, IEEE Netw. 33 (2) (2019) 111–117.
[17] Q.D. La, M.V. Ngo, T.Q. Dinh, et al., Enabling intelligence in fog computing to achieve energy and latency reduction, Digit. Commun. Netw. 5 (1) (2019) 3–9.
[18] B. Tang, Z. Chen, G. Hefferman, et al., Incorporating intelligence in fog computing for big data analysis in smart cities, IEEE Trans. Ind. Inform. 13 (5) (2017) 2140–2150.
[19] M.G. Kibria, K. Nguyen, G.P. Villardi, et al., Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks, IEEE Access 6 (2018) 32328–32338.
[20] Y. Dai, D. Xu, S. Maharjan, et al., Blockchain and deep reinforcement learning empowered intelligent 5g beyond, IEEE Netw. 33 (3) (2019) 10–17.
[21] Y. He, F.R. Yu, N. Zhao, et al., Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach, IEEE Commun. Mag. 55 (12) (2017) 31–37.
[22] V. Vijayakumar, D. Malathi, V. Subramaniyaswamy, et al., Fog computing-based intelligent healthcare system for the detection and prevention of mosquito-borne diseases, Comput. Hum. Behav. 100 (2019) 275–285.
[23] I. Stojmenovic, S. Wen, X. Huang, et al., An overview of fog computing and its security issues, Concurr. Comput.: Pract. Exper. 28 (10) (2016) 2991–3005.
[24] B. Cao, L. Zhang, Y. Li, et al., Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework, IEEE Commun. Mag. 57 (3) (2019) 56–62.
[25] B. Mao, Z.M. Fadlullah, F. Tang, et al., Routing or computing? The paradigm shift towards intelligent computer network packet transmission based on deep learning, IEEE Trans. Comput. 66 (11) (2017) 1946–1960.
[26] H. Ahuett-Garza, T. Kurfess, A brief discussion on the trends of habilitating technologies for Industry 4.0 and Smart manufacturing, Manuf. Lett. 15 (2018) 60–63.
[27] N. Javaid, A. Sher, H. Nasir, et al., Intelligence in IoT-based 5G networks: Opportunities and challenges, IEEE Commun. Mag. 56 (10) (2018) 94–100.
[28] G. Liu, Y. Xu, Z. He, et al., Deep learning-based channel prediction for edge computing networks toward intelligent connected vehicles, IEEE Access 7 (2019) 114487–114495.
[29] J. Ren, Y. Guo, D. Zhang, et al., Distributed and efficient object detection in edge computing: Challenges and solutions, IEEE Netw. 32 (6) (2018) 137–143.
[30] B. Chen, J. Wan, L. Shu, et al., Smart factory of industry 4.0: Key technologies, application case, and challenges, IEEE Access 6 (2017) 6505–6519.
[31] Z. Lv, X. Li, H. Lv, W. Xiu, BIM Big Data Storage in WebVRGIS, IEEE Trans. Ind. Inform. (2019).
[32] Z. Lv, H. Song, Mobile Internet of Things under data Physical Fusion Technology, IEEE Internet Things J. (2019).
[33] R. Shi, Y. Gan, Y. Wang, Evaluating scalability bottlenecks by workload extrapolation, in: 2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE, 2018, pp. 333–347.
[34] Z. Lv, W. Kong, X. Zhang, D. Jiang, H. Lv, X. Lu, Intelligent security planning for regional distributed energy internet, IEEE Trans. Ind. Inf. (2019).
[35] Z. Lv, S. Zhang, W. Xiu, Solving the Security Problem of Intelligent Transportation System with Deep Learning, IEEE Trans. Intell. Transp. Syst. (2020).
[36] J. Chen, Z. Lv, H. Song, Design of personnel big data management system based on blockchain, Future Gener. Comput. Syst. 101 (2019) 1122–1129.
[37] Z. Lv, B. Hu, H. Lv, Infrastructure monitoring and operation for smart cities based on IoT system, IEEE Trans. Ind. Inf. (2019).

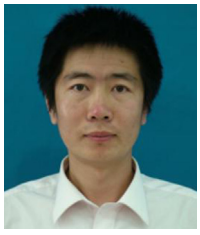
[38] Z. Lv, H. Song, Mobile Internet of Things under data Physical fusion Technology, IEEE Internet Things J. (2019).
[39] Z. Lv, L. Qiao, Analysis of healthcare big data, Future Gener. Comput. Syst. (2020).
[40] Z. Lv, L. Qiao, Deep belief network and linear perceptron based cognitive computing for collaborative robots, Appl. Soft Comput. (2020) 106300.



Zhihan Lv is currently an associate professor of Qingdao University, he was a Postdoctor Researcher of University of Barcelona, Spain. He was Research Associate at University College London (UCL). He received his Ph.D. from Paris7 University and Ocean University of China in 2012. He worked in CNRS (France) as Research Engineer, Umea University/ KTH Royal Institute of Technology (Sweden) as Postdoc Research Fellow, Fundacion FIVAN (Spain) as Experienced Researcher. He was a Marie Curie Fellow in European Union's Seventh Framework Program LANPERCEPT. His research mainly focuses on Multimedia, Augmented Reality, Virtual Reality, Computer Vision, 3D Visualization & Graphics, Serious Game, HCI, Big data, and GIS. He has contributed 200+ papers in the related fields on journals such as Plos one, ACM TOMM, and conference such as ACM MM, ACM CHI, ACM Siggraph, ICCV, IEEE Virtual Reality.

He is the guest editors for IEEE Transactions on Industrial Informatics, Future Generation Computer Systems, Neurocomputing, Neural Computing and Applications, Computers & Electrical Engineering, IET Image Processing, Multimedia Tools and Applications, Journal of Intelligent and Fuzzy Systems. He is Program Committee member of ACM IUI 2015 & 2016 & 2019, IEEE BIGDATA4HEALTH Workshop 2016, IEEE/CIC WIN Workshop 2016, IIKI2016, WASA2016, IEEE PDGC2016, and ACM SAC2017-WCN Track. He has also served as a reviewer for journals such as IEEE Transaction on Multimedia, IEEE Multimedia, Neurocomputing, Elsevier Computer Networks, Springer Telecommunication Systems, Multimedia Tools and Applications, KSII Transactions on Internet and Information Systems, Journal of Medical Internet Research, Intelligent Service Robotics, PRESENCE: Tele-operators and Virtual Environments, and conference e.g. ACM MUM 2013, ACM CHI 2014 & 2015, ACM DIS 2014 & 2016, IEEE Info-Vis 2014, ACM UIST 2014& 2016, ACM Mobile-HCI 2014 & 2015& 2016, ACM CHIPLAY 2014 & 2015& 2016, ACM CSCW 2015& 2016, ACM SUI 2014& 2015, ACM ITS 2014 & 2015, IEEE VAST 2014, IEEE VR 2015 & 2016, ACM IUI 2015 & 2016, IEEE 3DUI ACM 2015 Creativity & 2016, & Cognition ACM TVX 2015, IEEE Euro-Vis 2015 & 2016, 2015, ACM EICS 2015, ACM IDC 2015 & 2016, IEEE ICSIPA 2015, GI 2016, IEEE ITSC 2016, IEEE Sensors 2016, ACM ACI 2016, ACM VRST 2016, and ACM ISS 2016.

Lou Ranran is currently a graduate student of software engineering in the School of Data Science and Software Engineering of Qingdao University, and his research direction is Deep Learning and Big Data. He received a bachelor's degree in engineering from Qingdao University in 2019. He has extensive experience in marine data processing.



Qingjun Wang received his M.S. degree from the Northeast University in 2009. He is currently a Graduate Student studying for Ph.D. degree in the College of Automation Engineering, Nanjing University of Aeronautics and Astronautics. His research interests include Computing Communications, Pattern Recognition, EEG Recognition and Fatigue Detection.