IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON COLLABORATION FOR INTERNET OF THINGS

Received October 29, 2018, accepted November 13, 2018, date of publication November 21, 2018,
date of current version December 27, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2882452

A Density-Based Offloading Strategy for IoT Devices in Edge Computing Systems

CHENG ZHANG[1], HAILIANG ZHAO[1,2], AND SHUIGUANG DENG[1], (Senior Member, IEEE)
[1]College of Computer Science and Technology, Zhejiang University, Hangzhou 310058, China
[2]School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430063, China

Corresponding author: Shuiguang Deng (dengsg@zju.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1400601, in part by the Key Research and Development Project of Zhejiang Province under Grants 2015C01027 and 2017C01015, in part by the National Science Foundation of China under Grant 61772461, and in part by the Natural Science Foundation of Zhejiang Province under Grants LR18F020003 and LY17F020014.

ABSTRACT Collaboration spaces formed from edge servers can efficiently improve the quality of experience of service subscribers. In this paper, we first utilize a strategy based on the density of Internet of Things (IoT) devices and *k*-means algorithm to partition network of edge servers, then an algorithm for IoT devices' computation offloading decisions is proposed, i.e., whether we need to offload IoT devices' workload to edge servers, and which edge server to choose if migration is needed. The combination of locations of edge servers and the geographic distribution of various IoT devices can significantly improve the scheduling of network resources and satisfy requirements of service subscribers. We analyze and build mathematical models about whether/how to offload tasks from various IoT devices to edge servers. In order to better simulate operations of the mobile edge servers in more realistic scenarios, the input size of each IoT device is uncertain and regarded as a random variable following some probability distribution based on long-term observations. On the basis of that, an algorithm utilizing sample average approximation method is proposed to discuss whether the tasks to be executed locally or offloaded. Besides, the algorithm proposed can also help decide whether service relocation/migration is needed or not. Finally, simulation results show that our algorithm can achieve 20% of global cost less than the benchmark on a true base station dataset of Hangzhou.

INDEX TERMS Cooperative networks, edge computing, IoT devices, sample average approximation, service relocation.

I. INTRODUCTION

As the bridge between the physical world and the digital world, edge computing acts as the first entry point of user data. The real world deployment of edge computing is naturally distributed due to different urban layouts, which requires edge computing systems to support distributed computing and storage capacity, and to achieve the dynamic resource scheduling, state scheduling and unified management. Besides, the systems ought to support distributed intelligence and security capabilities. With network transmission latency reduced and hardware equipments' performance improved, dynamic and instant scheduling of edge servers in complex and heterogeneous networks becomes more and more imperious and significant.

Nowadays, plenty of data is created in various fields such as transport, industry, agriculture and our daily life. Besides, more and more IoT devices are created for a number of purposes. In the not-too-distant future, the global 5G communication will become reality. 5G technology has lower cost, lower power consumption, safer and more reliable characteristics than any previous technology [1]. Most importantly, it can provide users the premise of stable transmission rate quality and low latency under high-speed condition. Different from conventional cloud computing systems, edge computing systems face new challenges such as uncertainty about mobility of connected devices, uncertainty of personnal flow information, uncertainty of service demand generated by IoT devices, etc. Besides, the most important goal is to guarantee service subscribers' Quality of Experience (QoE) [2]. In real-life scenarios, users' movement are uncertain, which is affected by various factors. However, with long-term observations on devices' mobility [3] and user demand, it is possible to excavate the underlying rules on their probability distributions. In this paper, statistics on the density of IoT

devices with mobility information are utilized to minimize the execution and transmission latency.

II. RELATED WORKS

Both academia and industry have invested tremendous efforts in the field of edge computing and have made great progress in recent years [4]. Edge computing has experienced rapid development from theory to practice, and the core concepts of architecture and technology have been implemented in practice. Ge *et al.* [5] proposed an algorithm based on individual mobility model to evaluate the impact of user mobility performance on 5G small cell networks considering human tendency and clustering behaviors. The major challenge for edge computing systems is to provide reliable web services in highly dynamic wireless environment with limited resources. Besides, scalability of servers and the frequently changing of the connections between IoT devices and servers make the problem formulated even harder. In order to solve the problem legitimately, a service porvisioning architecture named *mobile service sharing community* was proposed [6]. The authors targeted at the service selection and composition of mobile composite services under the condition that both requesters and providers are moving. Hsu *et al.* [7] considered that realistic mobility models are fundamental to evaluate the performance of protocols in ad hoc networks, thus they proposed a time-variant community mobility model to analyze the mobility character1istics observed in daily life. Combining with the emerging of the *Service-Oriented Architecture* (SOA), they achieved dependable service composition by taking the mobility prediction of the service providers into consideration. Besides, they have studied uncertain mobility of the service providers in wireless ad hoc networks [8]. Deng *et al.* [9] focused on the problem of mobile service selection for composition in terms of energy consumption. They adopts the genetic algorithm to resolve problem and propose a replanning mechanism to deal with the changeable conditions and user behavior. Wang *et al.* [10] found that user mobility often makes prediction of Quality of Service (QoS) values deviate from actual values in traditional networks. In order to solve the problem, a service recommendation approach was proposed based on collaborative filtering. In addition to that, QoS prediction is carried out based on user mobility. Mao *et al.* [11] studied the green computing via *Energy Harvesting* (EH) technologies and adpoted the *Dynamic Voltage And Frequency Scaling* (DVFS) techniques to improve energy efficiency. Both the latency and punishment for dropping tasks are adopted as the algorithm performance metrics to design a low-complexity online algorithm, namely, the Lyapunov Optimization-based Dynamic Computation Offloading (LODCO) algorithm, which jointly decides the offloading decision, the CPU-cycle frequencies for local execution and the transmit power for offloading computation tasks. Ndikumana *et al.* [12] proposed the concept of '4C' (computing, caching, communication, and control) for edge computing platforms, which is necessary for big data-related applications. They divided the edge servers into several clusters to form *collaboration spaces*. Thus, each collaboration space can be taken to deploy big data service platform, which means the servers in the same cluster can share and allocate resources collectively. Chen *et al.* [13] studied peer offloading strategy in MEC-enabled networks. Then an online peer offloading framework allowing centralized and autonomous decision making was proposed. The battery charge of mobile devices is the main obstacle to process task with long execution time and high power consumption. Therefore, a scheme that can both reduce the execution time of task and energy consumption of the mobile devices is studied [14]. Wiesemann *et al.* [15] consider the service composition problem to be a multi-objective stochastic programming. The model in their paper is based on stochastic programming which accounts for quality uncertainty in a mathematically sound manner.

In this paper, we first propose a strategy to construct cooperative networks (also can be known as collaboration spaces) of edge servers. According to [16], we know that edge computing can provide computational capabilities for nearby service subscribers (also can be known as clients, IoT devices) through wireless access points (APs). Then an algorithm utilizing Sample Average Approximation (SAA) method [17] is proposed to discuss whether the tasks to be executed locally or offloaded. Besides, the algorithm proposed can also help decide server relocation (also can be known as service migration) is needed or not, and if relocation is needed, which edge server to choose. The main contributions of this paper can be summarized as follows.

1) We make great extensions on the framework proposed in [12] for 'Joint 4C' big data platforms, combining with the offloading strategy for IoT devices.
2) We proposed a more appropriate organization of edge servers' cooperative network, where overlapping of clusters is discarded.[1]
3) Long-term observations on the density of IoT devices, instead of specific mobility models, play a critical role for jointly deciding the offloading strategy and relocation policy.
4) Uncertain user requests with estimate of the probability distribution are adopted to simulate an environment which is more realistic. Then, in each collaboration space, with uncertain user requests, a joint dynamic computation offloading and resource allocation algorithm is proposed.
5) True data on base stations (BSs) in Hangzhou (a metropolis in east China) is utilized to verify the efficiency of the proposed algorithm.

III. SYSTEM MODEL

We consider an edge computing system in one specific area with multiple IoT devices and multiple edge servers

[1]Notice that overlapping of cooperative networks is not the same with overlapping of the signal cover of base stations.

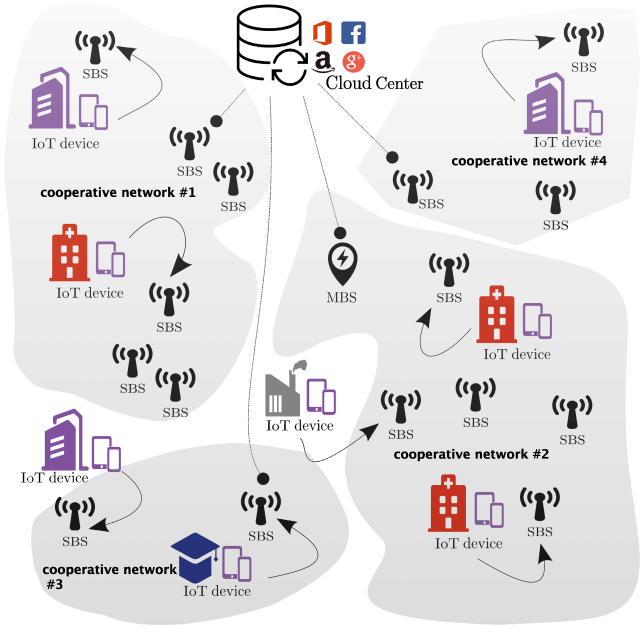

FIGURE 1. An edge computing system with four cooperative networks.

as shown in Fig. 1. Denote the index set of IoT devices by $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ and the index set of edge servers by $\mathcal{M} \triangleq \{1, 2, \ldots, M\}$. Summary of key notations can be found in Tab. 1. We first adpot a strategy to classify all edge servers in the specific region into clusters by both positions of edge servers and the devices' density in this area. It can imporve the global Quality of Service (QoS) [18] by decentralizing resources and computing power of servers. Computation task can be executed locally at IoT devices, or offloaded to its nearest edge server [19]. Integrity of computation tasks can be achieved by offloading to an edge server instead of a remote cloud center because the probability of transmission failure is greatly reduced [20]. After that, the system also needs to decide whether to relocate this task to another edge server in this cooperative network for better scheduling on computational resources or to execute it at the nearest edge server without relocation. Those strategic scheduling is made for obtaining the minimized latency, i.e., the best QoE for users.

Apparently, there always exists busy public transport roads and business districts in an arbitrary city. To illustrate this point, we pick out some daily walking paths from Hangzhou, a metropolis of east China, in Zhejiang province, and get the devices' density in some chosen area of this city, as shown in Fig. 2. We also utilize distribution information of China Mobile[2] base stations together with rail trajectory to help construct the cooperative networks of edge servers. The BSs information of Hangzhou is illustrated in Fig. 3. Service subscribers with IoT devices can offload their tasks to nearest edge servers while they are able to move arbitrarily in these areas [21].

[2]China Mobile is a Chinese state-owned telecoms company that provides mobile voice and multimedia services through its nationwide mobile telecoms network across mainland China.

TABLE 1. Summary Of key notations.

Notation	Description
ϱ_j	The specific cooperative network which the jth edge server belongs to
μ_k	The centroid of the kth collaboration space
$\mathcal{M}_k$	A set of edge servers who belong to the kth cooperative network
d_i	The input size of the task generated by the ith IoT device
I_i	Indicate that whether the computation task of the ith device is executed locally
O_{ij}	Indicate that whether the ith device offloads task to the jth edge server
$R_{ij'}^{kj}$	Indicate that whether the jth edge server relocate task of the ith device to the j'th edge server in the specific cooperative network which the jth edge server belongs to
$\mathcal{N}_j$	The set of IoT devices who offload computation task to the jth edge server
L_i^l	The number of CPU cycles for processing one bit of computation task at the ith IoT device
L_j^r	The number of CPU cycles for processing one bit of data at the jth edge server
f_i^w	The CPU cycle frequency of the wth CPU cycle for the ith IoT device
f_{ji}^w	The CPU cycle frequency of the wth CPU cycle of the jth edge server when executing task from the ith IoT device
p_{ij}	The transmission power from the ith device to the jth edge server
d_i	Input size of computation task generated by the ith IoT device

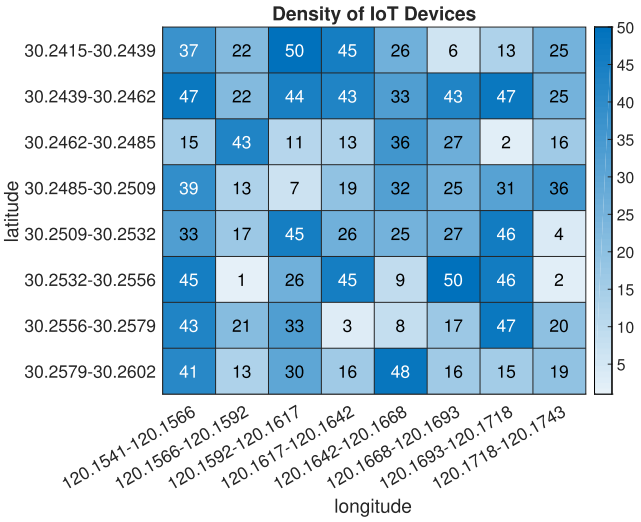

FIGURE 2. Density of IoT devices with grids embedded.

A. EDGE SERVERS IN COOPERATIVE NETWORKS

We classify all edge servers into K cooperative networks and use that $A_{m,k} = 1, m \in \mathcal{M}, k \in \mathcal{K} \triangleq \{1, \ldots, K\}$ to represent that the mth edge server is attached to the kth cooperative network, otherwise 0. Therefore, matrix $\mathbf{A}$ is naturally generated to show inclusion relationships of edge

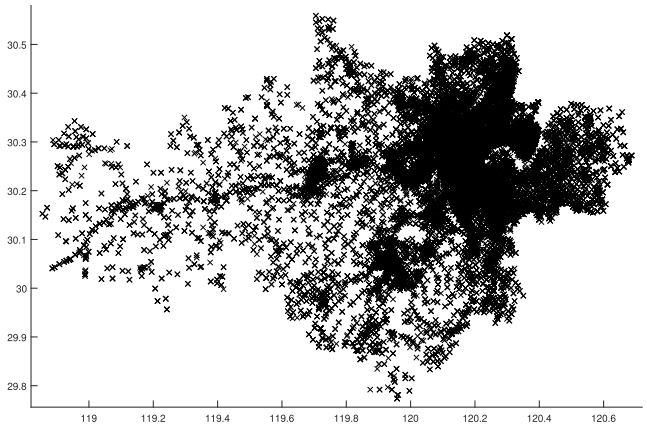

FIGURE 3. Distribution of China Mobile BSs in Hangzhou. The horizontal and vertical coordinates are longitude and latitude, respectively.

servers and cooperative networks.

$$\mathbf{A} \triangleq \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,K} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,K} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M,1} & A_{M,2} & \cdots & A_{M,K} \end{pmatrix}. \tag{1}$$

As we have mentioned before, overlapping of cooperative networks is disabled. Therefore, each edge server only belongs to one specific cooperative network, i.e., the following constraint should be satisfied:

$$\sum_{k \in \mathcal{K}} A_{j,k} = 1, \quad j \in \mathcal{M}. \tag{2}$$

we denote

$$\mathcal{M}_k \triangleq \{j \in \mathcal{M} | A_{j,k} = 1\}, \quad k \in \mathcal{K} \tag{3}$$

as the set of edge servers belonging to the kth cooperative network.

In this subsection, we develop the **C**ooperative **N**etworks **F**ormulation (CNF) algorithm based on *k-means* [22] and IoT devices' density to classify the edge servers in the chosen area into clustered collaboration spaces. We denote each cluster of edge servers as a cooperative network where all servers in it can jointly schedule their computing resources to reduce tasks execution latency [23]. For the purpose of minimizing the communication latency between edge servers, we take both distance and devices' density between any two edge servers into consideration.

Denote $\mathbf{z}_1, \mathbf{z}_2$ as two points in the area of Fig. 2. As we have mentioned before, the value of $\Delta_{\mathbf{z}_1,\mathbf{z}_2}$ is calculated by the formula

$$\Delta_{\mathbf{z}_1,\mathbf{z}_2} = \int_{l_{\mathbf{z}_1,\mathbf{z}_2}} f(\mathbf{z}_1, \mathbf{z}_2) dl, \tag{4}$$

where $l_{\mathbf{z_1},\mathbf{z_2}}$ denotes the segment between these two points, and $f(\mathbf{z}_1, \mathbf{z}_2)$ denotes the probability density function along $l_{\mathbf{z_1},\mathbf{z_2}}$. Apparently, the bigger devices' density, the greater weight on the line segment of those two points.

Based on *k-means* algorithm, we first randomly initialize centroids of K cooperative networks. The centroid of the kth cooperative network is denoted as μ_k, which is a bivector representing one position on the chosen two-dimensional region. We denote $\mathcal{K}_\mu \triangleq \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K\}$ as the set of all the centroids of cooperative networks.

We define ϱ_j as the index of the specific cooperative network which the jth edge server belongs to, i.e.,

$$\varrho_j \triangleq \arg\min_{k \in \mathcal{K}} \Delta_{\mathbf{z}_j,\boldsymbol{\mu}_k}, \tag{5}$$

where $\mathbf{z}_j$ is the position of the jth edge server. Thus, the centroid of the kth cooperative network can be updated by

$$\mu_k \triangleq \frac{\sum_{j=1}^{M} \mathbb{I}\{\varrho_j = k\} \cdot \mathbf{z}_j}{\sum_{j=1}^{M} \mathbb{I}\{\varrho_j = k\}}. \tag{6}$$

In the beginning of our algorithm, we randomly pick K points as initial centroids of $\mu_1, \mu_2, \ldots \mu_K$, then we classify each edge server to a specific centroid by (5). After one iteration, we could update the centroid $\boldsymbol{\mu}_k$ by (6). Those two steps then be repeated until convergence to an ideal state is obtained.

Our proposed strategy based on long-term observations of IoT devices can be dynamically adjusted at intervals of months, weeks or even days. i.e., If the long-term observations statistics of IoT devices in the region changes, then our strategy can timely adjust with it, too. Therefore, the proposed algorithm can flexibly and efficiently respond to the changing demand of IoT devices under various practical scenarios.

Actually, the accuracy of our offloading strategy depends on the statistical density information of long-term devices in the area. In this paper, for the convenience of calculation, we simply divide this area into grids, and the distribution density of the number of user devices in each grid is randomly generated. After that, by combining of IoT devices' density (in grid) and the actual geographic distance between them, more appropriate collaboration spaces generate. Apparently, the size of grid, as an important parameter, can significantly influence the effectiveness of the proposed algorithm. In our future work, grid-division will be replaced by dot density. Every cooperative network only serves the devices covered by it, which greatly improves the local efficiency, and also meets the characteristics of the edge property. In our cooperative network, IoT device does not need to directly connect to the cloud center with intolerable latency or an edge server with a relatively long physical distance. The excellent characteristic above is consistent with the edge server's features for the sake of service subscribers.

B. LOCAL EXECUTION MODEL

We use $d_i, i \in \mathcal{N}$ to denote the input size (in bits) of computation task generated by the ith IoT device. As we have mentioned before, every d_i is uncertain before the offloading/migrating decisions have been made. Actually, uncertain requests are more aligned with the reality because they help construct a more universal scenario [17]. Although the size of the computation task generated by each IoT

Algorithm 1 Cooperative Networks Formulation (CNF)

Input positions of edge servers: $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_M$ and IoT devices' density function: $\forall \mathbf{z}_m, \mathbf{z}_n, \Xi_{\mathbf{z}_m,\mathbf{z}_n} = f(\mathbf{z}_m, \mathbf{z}_n)$
Initialize K centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K$
$\mathbf{A} \leftarrow \mathbf{O}$
for $k = 1$ to K **do**
  repeat
    $\forall j \in \mathcal{M}$, calculate ϱ_j by (5)
    Update μ_k by (6)
  until $\forall j \in \mathcal{M}$, ϱ_j remains unchanged
end for
$\forall j \in \mathcal{M}$, set $A_{j,\varrho_j} \leftarrow 1$
return $\mathbf{A}$

device is unknown each time the decision has to be made, we assume that an estimate of IoT devices' density in the form of probability distribution is available. We denote $\mathbf{D} = \{D_1, D_2, D_3, \cdots D_N\}$[3] as the vector consisting of random variables for input size of tasks generated by IoT devices [24]. Besides, we assume that $\forall i \in \mathcal{N}, D_i$ is a random variable follows the independent and identically distributed (i.i.d.) rule. We denote L_i^l as the number of CPU cycles needed to process one bit of computation task locally at the ith IoT device, thus the local execution cost (i.e., latency) can be calculated by

$$c_i^{\text{local}} = \sum_{w=1}^{W_i} \frac{1}{f_i^w}, \quad W_i = L_i^l \cdot d_i, \ i \in \mathcal{N}, \tag{7}$$

where f_i^w represents the frequency of the wth CPU cycle for the ith IoT device. Naturally, it should satisfy the limit frequency constant:

$$0 < f_i^w \leq f_i^{\max}, \quad i \in \mathcal{N}, \tag{8}$$

where f_i^w represents the CPU frequency for the wth CPU cycle [25] of the ith IoT device, which can be realized by DVFS techniques.

We denote I_i as the indicator that represents the task of the ith IoT device is executed locally, i.e.,

$$I_i \in \{0, 1\}, \quad i \in \mathcal{N}. \tag{9}$$

$I_i = 1$ if local execution is chosen, otherwise $I_i = 0$.

C. SERVER EXECUTION MODEL

When the input of computation task is transmitted from the ith IoT device to the jth edge server, the transmission power p_{ij} should satisfy the maximum transmission power constraint:

$$0 \leq p_{ij} \leq p_i^{\max}, \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{10}$$

Denote the small-scale fading channel power gain from the ith to the jth edge server as ζ_{ij}, which is assumed to be exponentially distributed with a given unit mean. According to communication theory [26], the corresponding channel power gain can be obtained by $h_{ij} = \zeta_{ij} \cdot g_0 \cdot (\frac{d_0}{d_{ij}})^{\phi}$, where d_0 denotes the reference distance, d_{ij} denotes the geographic distance between the ith IoT device and the jth edge server, ϕ denotes the pass-loss exponent, g_0 denotes the pass-loss constant and δ_j denotes the noise power at the edge server j. According to Shannon-Hartley formula [26], we can obtain the achievable rate Λ_{ij} by

$$\Lambda_{ij} = \alpha_{ij} \cdot \omega_j \cdot \log_2(1 + \frac{h_{ij} \cdot p_{ij}}{\delta_j}), \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{11}$$

where ω_j is the bandwidth of the jth edge server and α_{ij} is the proportion of bandwidth allocated for the corresponding IoT device, i.e., each connected IoT device is assigned to one sub-band with $\alpha_{ij} \cdot \omega_j$ MHz. The bandwidth allocation ratio should satisfy

$$0 < \alpha_{ij} \leq 1, \quad i \in \mathcal{N}, \ j \in \mathcal{M} \tag{12}$$

and

$$\sum_{i \in \mathcal{N}} \alpha_{ij} \leq 1, \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{13}$$

We denote the location of the ith IoT device and the jth edge server as

$$\mathbf{u}_i = (x_{u_i}, y_{u_i})^\top, \quad \mathbf{s}_j = (x_{s_j}, y_{s_j})^\top, \ i \in \mathcal{N}, \ j \in \mathcal{M}, \tag{14}$$

respectively. Therefore, the distance between the ith IoT device and the jth edge server, denoted as d_{ij}, which has been mentioned before, is calculated by

$$d_{ij} = \sqrt{(x_{u_i} - x_{s_j})^2 + (y_{u_i} - y_{s_j})^2}, \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{15}$$

As we have mentioned before, if computation task of the ith IoT device is decided to be executed remotely, then it must be the nearest server for offloading.[4] Our system could identify this specific server by $j_i = \arg\min_{j' \in \mathcal{M}} \|\mathbf{u}_i - \mathbf{s}_{j'}\|_2$. Then the transmission latency can be calculated by

$$c_{ij}^{\text{tx}} = \frac{d_{ij}}{\Lambda_{ij}}, \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{16}$$

We denote O_{ij} as the indicator to represent whether the task from the ith IoT device is going to be offloaded to the jth edge server, i.e.,

$$O_{ij} \in \{0, 1\}, \quad i \in \mathcal{N}, \ j \in \mathcal{M}. \tag{17}$$

$O_{ij} = 1$ if task of the ith IoT device offloading to the jth edge server, otherwise 0. Apparently, $O_{ij} = 1$ if and only if $I_i = 1$ and $j = \arg\min_{j' \in \mathcal{M}} \|\mathbf{u}_i - \mathbf{s}_{j'}\|_2$. We denote the set of all IoT devices which offload tasks to the jth edge server as

$$\mathcal{N}_j \triangleq \{i \in \mathcal{N} | O_{ij} = 1\}, \quad j \in \mathcal{M}. \tag{18}$$

[3] Notice that d_i is a concrete realization of random variable D_i.

[4] According to [26], the shorter the geometric distance between the edge server and the IoT device, the stronger the signal of the established link between them. Thus, the IoT device always choose to offload task to its nearest edge server.

Each offloaded task can only be assigned to one edge server, thus

$$\sum_{j\in\mathcal{M}} O_{ij} = 1, \quad i \in \mathcal{N}. \tag{19}$$

The decision for each IoT device is binary, which means every computation task either be executed locally, or executed remotely at the edge server. Thus we have

$$I_i = 1 - \sum_{j\in\mathcal{M}} O_{ij}, \quad i \in \mathcal{N}. \tag{20}$$

In addition to the transmission latency, remote execution also consumes time. We can calculate the latency of remote execution at the jth edge server as:

$$c_{ij}^{\text{server}} = \sum_{w=1}^{W_j} \frac{1}{f_{j,i}^{w}}, \quad W_j = L_j^r \cdot d_i,\ i \in \mathcal{N},\ j \in \mathcal{M}, \tag{21}$$

where $f_{j,i}^{w}$ is the CPU cycle frequency of the wth CPU cycle for executing one bit of data at the jth edge server, which should satisfy

$$0 < f_{j,i}^{w} \le f_j^{\max}, \quad j \in \mathcal{M}. \tag{22}$$

Besides, L_j^r is the number of CPU cycles for processing one bit of data at the jth edge server.

Due to the limitations of the computer operating system and data structure, the total amount of input size of tasks that each server can execute is limited.

Last but not least, We assume that each edge server j has the capacity of executing C_j bits of tasks, where $j \in \mathcal{M}$.

D. TASK RELOCATION MODEL

Task in one edge server also might be relocated to another server to execute for minimization of the overall latency. We know that the index of cooperative network which server j belongs can be represented as $k_j = \arg\max_{k'\in\mathcal{K}} A_{jk'}$. As we have mentioned before, task relocation could only happen in servers from the same cooperative network, which means that if the task is relocated from the jth to the j'th edge server and $j' \neq j$, then we have $j' \in \mathcal{M}_{k_j}$.[5] We denote $R_{ij'}^{k_j}$ as the indicator of task relocation. If and only if $R_{ij'}^{k_j} = 1$ and $j \neq j'$, the task is relocated from server j to server j' in the k_jth collaboration space. Denote the relocation vector for the ith IoT device as $\mathbf{R}_i^{k_j}$, i.e.,

$$\begin{aligned} \mathbf{R}_i^{k_j} &= (R_{i,1}^{k_j}, R_{i,2}^{k_j}, \ldots, R_{i,|\mathcal{M}_{k_j}|}^{k_j}), \\ R_{ij}^{k_j} &\in \{0,1\}, i \in \mathcal{N}_j, j \in \mathcal{M}_{k_j}. \end{aligned} \tag{23}$$

Notice that if $R_{ij}^{k_j} = 1$, then no relocation happens. Therefore, we can naturally generate the following constraint[6]:

$$\sum_{j'\in\mathcal{M}_{k_j}} R_{ij'}^{k_j} = O_{ij}, \quad i \in \mathcal{N}_j. \tag{24}$$

[5]We have given the definition of $\mathcal{M}_k$. Notice that k_j is a specific k.

[6]Actually, constraint (24) is still satisfied for $i \in \mathcal{N}_j^{\text{latent}}$, which is a set defined in subsection IV. B.

Task relocation consumes time, too. We can calculate the latency by

$$c_{ijj'}^{\text{reloc}} = \frac{d_i}{\Gamma_{j,j'}}, \quad i \in \mathcal{N}_j,\ j, j' \in \mathcal{M}_{k_j}, \tag{25}$$

where $\Gamma_{j,j'}$ represents the X2 transmission rate from the jth server to the j'th server by using X2 link, which has a positive correlation with the transmission bandwidth ω_r [27].

As we have mentioned before, each edge server has maximum throughput limitation on executing the offloaded tasks, which can be described by

$$\sum_{j\in\mathcal{M}_k} \sum_{i\in\mathcal{N}_j} R_{ij'}^{k_j} \cdot d_i \le C_{j'}, \quad j, j' \in \mathcal{M}_k,\ k_j \in \mathcal{K}. \tag{26}$$

IV. PROBLEM FORMULATION

As discussed above, we consider some aspects which could be affected to the applications of IoT devices. Thus we construct local execution model, server execution model and task relocation model. Our main work is to make decisions for each IoT device in order to reach the minimum latency overall.

A. LATENCY MINIMIZATION PROBLEM

All the execution and transmission latency are taken as the metrics to evaluate the strategy. Therefore, the optimization problem can be formulated as

$$\begin{aligned} \mathcal{P}_1: \min_{\mathbf{\Theta}}\ & c^{\text{total}} = \sum_{i\in\mathcal{N}} (I_i \cdot c_i^{\text{local}} + \sum_{j\in\mathcal{M}} O_{ij} \cdot c_{ij}^{\text{tx}} \\ & + \sum_{j\in\mathcal{M}} O_{ij} \cdot \sum_{j'\in\mathcal{M}_{k_j}\setminus\{j\}} R_{ij'}^{k_j} \cdot c_{ijj'}^{\text{reloc}}) \\ & + \sum_{j\in\mathcal{M}} \sum_{i\in\mathcal{N}_j} \sum_{j'\in\mathcal{M}_{k_j}} R_{ij'}^{k_j} \cdot c_{ij'}^{\text{server}} \\ s.t.\ & (2), (8), (9), (10), (12), (13), (17), \\ & (19), (20), (22), (23), (24), (26). \end{aligned}$$

In problem $\mathcal{P}_1$, $\mathbf{\Theta}$ is the decision vector, which consists of integer variables $I_i, O_{ij}, R_{ij'}^{k_j}$ and real variables $p_{ij}, f_i^{w}, f_{j,i}^{w'}$, $i \in \mathcal{N}, j \in \mathcal{M}, w \in \{1, \ldots, W_i\}, w' \in \{1, \ldots, W_j\}$.

B. PROBLEM ANALYSIS

Lemma 1: If the ith IoT device's task is executed locally, then the minimum value of c_i^{local} (denoted as $c_i^{\text{local}\star}$) is $\frac{L_i^l \cdot d_i}{f_i^{\max}}$.

Proof: Accroding to basic inequality, we have $\frac{1}{\sum_{w=1}^{L_i^l\cdot d_i} \frac{1}{f_i^w}} \le \sqrt{\prod_{w=1}^{L_i^l\cdot d_i} f_i^w} \le \sqrt{\prod_{w=1}^{L_i^l\cdot d_i} f_i^{\max}}$, then the minimum value can be obtained, if and only if $f_i^1 = f_i^2 = \ldots = f_i^W = f_i^{\max}$. This can be realized by DVFS technologies. □

We adopt the static bandwidth allocation strategy in this paper. The bandwidth resource of the jth edge server is equally divided into several sub-bands, and every potential connected IoT device occupies one sub-band. Denote the set

of IoT devices whose tasks are possible to be offloaded to the jth edge server as $\mathcal{N}_j^{\text{latent}}$,[7] which can be obtained by

$$\mathcal{N}_j^{\text{latent}} = \{i \in \mathcal{N} | \arg\min_{j' \in \mathcal{M}} \|\mathbf{u}_i - \mathbf{s}_{j'}\|_2 = j\}. \quad (27)$$

Thus, we can obatain that

$$\alpha_{ij}^{\star} = 1/|\mathcal{N}_j^{\text{latent}}|, i \in \mathcal{N}_j^{\text{latent}}. \quad (28)$$

If $\mathcal{N}_j^{\text{latent}}$ is replaced by $\mathcal{N}_j$, then the problem formulated will be even harder to handle. Because in this way, the offloading decision is coupled with the bandwidth allocation, and each IoT device decison cannot be processed individually. Besides, if we adopt the dynamic bandwidth allocation strategy for connected IoT devices, then the problem formulated will become a mixed integer programming problem, which will be in-depth studied by our future work.

With $\alpha_{ij}^{\star}$ obtained, we can calculate $c_{ij}^{\text{tx}\star}$ easily. $c_{ij}^{\text{server}\star}$ and $c_{ijj'}^{\text{reloc}\star}$ can be obtained in the same way.

Therefore, P2 can be formulated as each cooperative network has capability to make optimal decision. Now we consider the kth cooperative network only. We denote the set of IoT devices which related to the kth cooperative network as $\mathcal{N}^k = \cup_{j \in \mathcal{M}_k} \mathcal{N}_j^{\text{latent}}$. Those IoT devcies could transmit task to edge servers in the kth cooperative network if they need to. Other cooperative networks is same as the kth. We can sum up latency costs of them to achieve a total latency cost of the system, thus we just need to focus on one cooperative network. A new problem $\mathcal{P}_2$ can be formulated from $\mathcal{P}_1$:

$$\begin{aligned}\mathcal{P}_2: \min_{\mathbf{\Theta'}} c^{\text{total}\star} \triangleq \sum_{i \in \mathcal{N}^k} \Bigg(& I_i \cdot c_i^{\text{local}\star} + \sum_{j \in \mathcal{M}_k} O_{ij} \cdot c_{ij}^{\text{tx}\star} \\ & + \sum_{j \in \mathcal{M}_k} \sum_{j' \in \mathcal{M}_k \setminus \{j\}} R_{ij'}^{kj} \cdot c_{ijj'}^{\text{reloc}\star} \\ & + \sum_{j \in \mathcal{M}_k} \sum_{j' \in \mathcal{M}_k} R_{ij'}^{kj} \cdot c_{ij}^{\text{server}\star} \Bigg)\end{aligned}$$

$$s.t.\ I_i \in \{0,1\}, \quad i \in \mathcal{N}^k, \quad (29)$$

$$O_{ij} \in \{0,1\}, \quad i \in \mathcal{N}^k, j \in \mathcal{M}_k, \quad (30)$$

$$\sum_{j \in \mathcal{M}_k} O_{ij} = 1, \quad i \in \mathcal{N}^k, \quad (31)$$

$$I_i = 1 - \sum_{j \in \mathcal{M}_k} O_{ij}, \quad i \in \mathcal{N}^k, \quad (32)$$

$$\sum_{j' \in \mathcal{M}_k} R_{ij'}^{kj} = O_{ij}, \quad i \in \mathcal{N}_j, j \in \mathcal{M}_k, \quad (33)$$

$$\sum_{j \in \mathcal{M}_k} \sum_{i \in \mathcal{N}^k} R_{ij'}^{kj} \cdot d_i \le C_{j'}, \quad j' \in \mathcal{M}_k. \quad (34)$$

In problem $\mathcal{P}_2$, $\mathbf{\Theta'}$ is the new decision vector consisting of $I_i, O_{ij}, R_{ij'}^{kj}, i \in \mathcal{N}^k, \quad j, j' \in \mathcal{M}_k$.

[7] Notice that $\mathcal{N}_j^{\text{latent}}$ is not the same with $\mathcal{N}_j$.

C. PROBLEM WITH UNCERTAIN DEMAND EMBEDDED

As we have mentioned before, input size of computation task of each IoT device is viewed as an uncertain random variable [28]. From historical data, it's reasonable to assume that the distribution of input size of task can be estimated, thus we can regard the optimization target of $\mathcal{P}_2$ as expected value of $c^{\text{total}\star}$ under probability distribution P, which can be denoted as $\mathbb{E}_P[C(\mathbf{\Theta'}, \mathbf{D})]$. The decision vector $\mathbf{\Theta'}$ can be chosen from a finite set $\mathcal{Q} \triangleq \{0,1\}^{|\mathcal{N}^k|+|\mathcal{N}^k|\times|\mathcal{M}_k|+|\mathcal{M}_k|\times|\mathcal{M}_k|}$. Therefore, the new target can be formulated as

$$\min_{\mathbf{\Theta'} \in \mathcal{Q}} \mathbb{E}_P[C(\mathbf{\Theta'}, \mathbf{D})]. \quad (35)$$

Thus, the new problem $\mathcal{P}_3$ has the format of

$$\begin{aligned}\mathcal{P}_3: \min_{\mathbf{\Theta'} \in \mathcal{Q}} \mathbb{E}_P[C(\mathbf{\Theta'}, \mathbf{D})] \triangleq \sum_{i \in \mathcal{N}^k} \Bigg(& I_i \cdot \mathbb{E}_P[c_i^{\text{local}\star}] \\ & + \sum_{j \in \mathcal{M}_k} O_{ij} \cdot \mathbb{E}_P[c_{ij}^{\text{tx}\star}] \\ & + \sum_{j \in \mathcal{M}_k} \sum_{j' \in \mathcal{M}_k \setminus \{j\}} R_{ij'}^{kj} \cdot \mathbb{E}_P[c_{ijj'}^{\text{reloc}\star}] \\ & + \sum_{j \in \mathcal{M}_k} \sum_{j' \in \mathcal{M}_k} R_{ij'}^{kj} \cdot \mathbb{E}_P[c_{ij}^{\text{server}\star}] \Bigg)\end{aligned}$$

$$s.t.\ (29), (30), (31), (32), (33), (34).$$

where $\mathbb{E}_P[c_{ij}^{\text{local}\star}] = \frac{L_i^j \mathbb{E}_P[d_i]}{f_i^{max}}$. $\mathbb{E}_P[c_{ij}^{\text{tx}\star}]$, $\mathbb{E}_P[c_{ijj'}^{\text{reloc}\star}]$ and $\mathbb{E}_P[c_{ij}^{\text{server}\star}]$ can be obtained in the same way [28], [29].

The above formulation aims at the average total cost of latency of all IoT devices. By Monte Carlo Sampling, with support from *the Law of Large Numbers* [30], when $\mathbf{\Theta'}$ is fixed, the value of target can converge with probability one to the expectation $\mathbb{E}_P[C(\mathbf{\Theta'}, \mathbf{D})]$.

To solve this stochastic discrete optimization problem, Sample Average Approximation (SAA) method can be utilized to form the target function

$$\min_{\mathbf{\Theta'} \in \mathcal{Q}} \hat{c}_{N'}^{\text{total}}(\mathbf{\Theta'}) \triangleq \frac{1}{N'} \sum_{q=1}^{N'} C(\mathbf{\Theta'}, \mathbf{D}^q), \quad (36)$$

where N' is the number of sampling times (also called scenarios).

V. JOINT COMPUTATION OFFLOADING AND RESOURCE MANAGEMENT OPTIMIZATION ALGORITHM

In this section, we propose a **J**oint **C**omputation **O**ffloading and **R**esource **M**anagement (JCORM) algorithm based on SAA method to solve the stochastic optimization problem $\mathcal{P}_3$.

We denote $\mathcal{D}_i = [d_i^1, d_i^2, d_i^3, \cdots, d_i^S]$ as the history of constructed scenarios (size S) for the input size of task generated on the ith IoT device under distribution P. We use the long-term task volume historical statistics of the device to construct the sample values in the SAA algorithm [31]. When we divide the edge servers of the entire area into a unified area, the edge servers of each area form a cooperative network. The advantage of this makes the calculation of the

Algorithm 2 Joint Computation Offloading and Resource Management (JCORM)

1: Generate L independent solutions
2: Set up a gap tolerance ϵ
3: **for** $l = 1$ to L in parallel **do**
4: Generate S independent scenarios
5: Obtain the minimum value of $\hat{C}_S^l(\hat{\boldsymbol{\theta}}_S^l)$ with the form of $\frac{1}{S}\sum_{q\in S} C(\hat{\boldsymbol{\theta}}_S^l, \mathbf{D}^q)$ by solving problem $\mathcal{P}_2$
6: Record the optimal solution variable $\hat{\boldsymbol{\theta}}_S^l$ and optimal value $\hat{v}_S^l$
7: **end for**
8: Evaluate L independent solutions with S' ($S' \gg S$) scenarios by $\hat{C}_{S'}(\hat{\boldsymbol{\theta}}) = \frac{1}{S'}\sum_{q\in S'} C(\hat{\boldsymbol{\theta}}, \mathbf{D}^q)$
9: Calculate the average value $\bar{v}_S^l = \frac{1}{L}\sum_{l=1}^{L} \hat{v}_S^l$
10: **if** the gap $\hat{C}_{S'}(\hat{\boldsymbol{\theta}}_S^l) - \bar{v}_S^l < \epsilon$ **then**
11: Choose the best solution $\hat{\boldsymbol{\theta}}_S^{l\star}$ among all independent candidates
12: **else**
13: Increase S (for drill) and S' (for evaluation), then go back to Step 2
14: **end if**
15: **return** the best solution $\hat{\boldsymbol{\theta}}_S^{l\star}$

optimization decision in one area greatly reduced, so that Algorithm 2 performs more efficiently in a relative small area.

In our algorithm we employ SAA-based method (or Monte Carlo simulation-based) approach to obtain a reliable estimation of the expected discrete optimization problem. We denote $\hat{C}_S^l$ as the approximation of $C(\boldsymbol{\Theta}', \mathbf{D})$ on the lth sample with S scenarios. In step 3 of algorithm 2, we generate S scenarios to calculate each sample value and a feasible solution $\hat{\boldsymbol{\theta}}_S^l$.[8] We also set up a precision for increasing S and S' if needed, by calculating the gap between $\hat{C}_{S'}^l(\hat{\boldsymbol{\theta}}_S^l, \mathbf{D})$ and $\bar{v}_S^l$, which allows our algorithm to converge with a reasonable rate [17], [32].

VI. SIMULATION RESULTS

In this section, we will demonstrate the results of the proposed algorithms and verify their effectiveness. Then, we will show the impacts of the system parameters by control variable method.

The simulation was run on a machine with an Intel Core 2.5 GHz i7-4710MQ CPU and 8G of memory. The algorithm was implemented in MATLAB R2015b and IBM ILOG CPLEX optimizer solvers (for academics).

In simulation, 1730 IoT devices are randomly located in the chosen area while location of China Mobile base stations is extracted from real data. By CNF algorithm, we divide 400 edge servers into 64 cooperative networks, as shown in Figure. 4. We assumed each edge server is integrated into

[8] θ is the notation for approximate solution of exact solution $\boldsymbol{\Theta}'$ under SAA method.

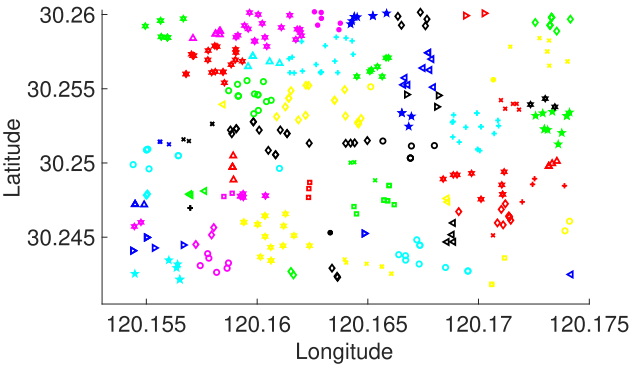

FIGURE 4. 64 cooperative networks with 400 edge servers.

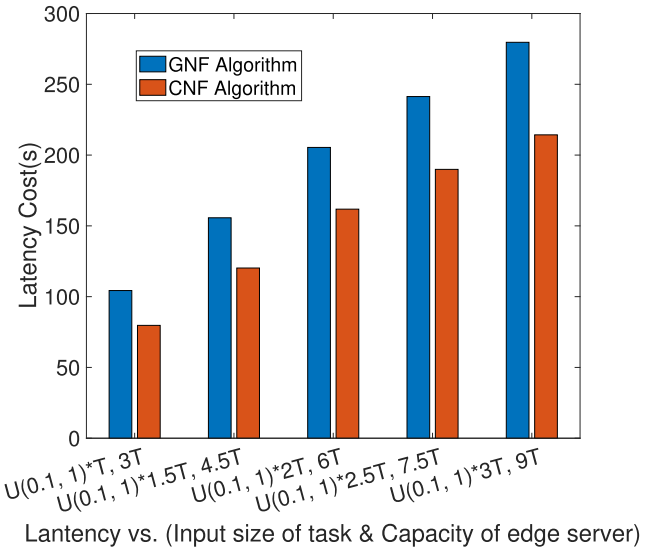

FIGURE 5. Global latency obtained by CNF and GNF, respectively.

one of the base stations. We set $M = 400$, $N = 1730$ and $K = 64$ in our simulation.

Input size of each IoT device in the network are uniformly distributed with mean of 16.5Mb. Besides, the small-scale fading channel power gains are exponentially distributed with mean $g_0\, d^{-4}$, whre $g_0 = -40$dB is the path-loss constant. $\forall i \in \mathcal{N}, j \in \mathcal{M}$ we set $\omega = 200$MHz, $\omega_r = 6$GHz, $\delta = 10^{-13}$W, $p_i^{max} = 1.5$W, device CPU frequency $f_i^{\max} = 1$GHz, $f_j^{\max} = 3$GHz, $L_i = 10$, $L_j = 0.02$, $C_j = 90$Mb.

The input size of computation tasks generated by IoT devices follows the uniform distribution of $U(0.1, 1) \cdot T$, where $T = 3e7$bits.

We set a benchmark called **G**rids **N**etwork **F**ormulation (GNF) algorithm for comparison with CNF algorithm. By GNF algorithm, the area evenly divided into rectangular grids, and each grid is a network consisting of edge servers located in it. We have experimentalized on dividing the edge servers into 64 networks by CNF and GNF, respectively. Figure. 5 shows that although input size of task of each device becomes larger with diffferent C_j ($C_j \in \{3T, 4.5T, 6T, 7.5T, 9T\}$), the cost of the whole system

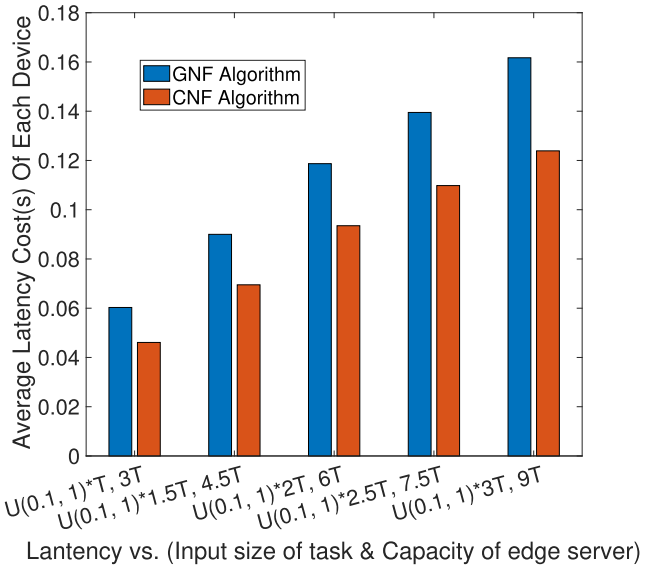

FIGURE 6. Average latency vs. (Input size of task and Capacity of edge server) under CNF and GNF, respectively.

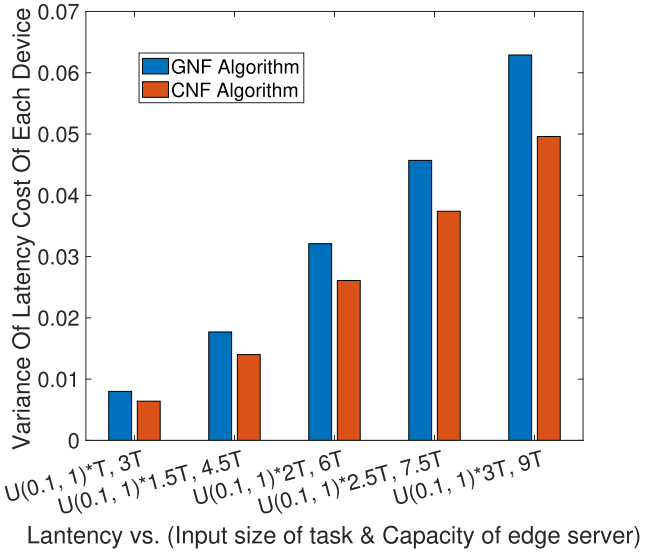

FIGURE 7. Variance of latency under CNF and GNF, respectively.

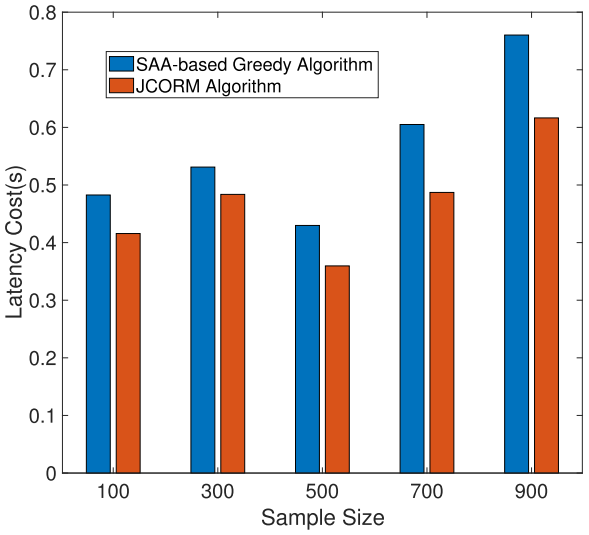

FIGURE 8. Latency under different sample size under SAAG and JCORM, respectively.

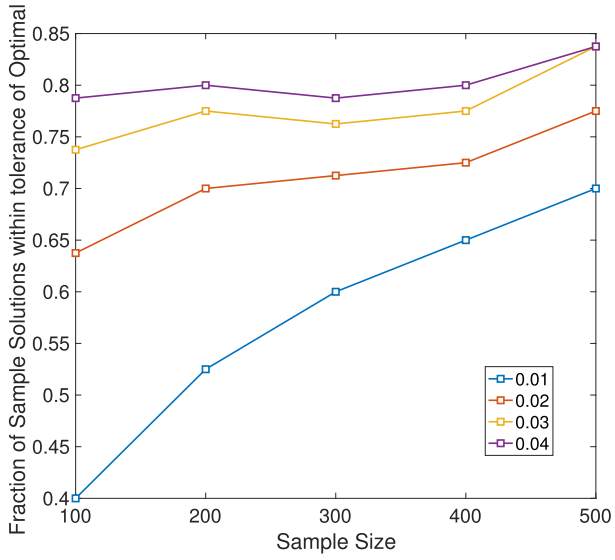

FIGURE 9. Number of `counter`-requesting solutions under different sample sizes.

obtained by CNF algorithm is still smaller than GNF's (nearly 20%).

Simulation results demonstrated in Figure.6 and Figure. 7 are under the same condition. Figure. 6 shows that the average latency of each IoT device obtained by CNF is smaller than GNF's. Figure. 7 shows that the variance of each IoT device's latency is smaller under CNF. It's noticeable that IoT devices in each cooperative network can get better service with a lower response time. If the edge servers are divided by GNF algorithm simply according to a geographical area, the important factor, i.e., the distribution of Iot devices is completely ignored.

As we have mentioned before, each cooperative network is an autonomous domain and has independent decision analysis capabilities. Simulation results demonstrated in Figure. 8 and Figure. 9 are based on the following setting: From 64 cooperative networks, we randomly choose one (denoted as k) with 2 edge servers (denote as j, j', respectively). Besides, $|\mathcal{N}_j^{\text{latent}} \cup \mathcal{N}_{j'}^{\text{latent}}| = 11$. In addation to this, we construct a benchmark called **SAA**-based **G**reedy (SAAG) algorithm for comparison with JCORM.

Based on the above setting, 5 controlled experiments on sample size (i.e., the number of independent solutions) of 11 IoT devices are done. Each experiment generating L independent solutions, each of which has scenarios of the same size (S and S' both). Figure. 8 shows that JCORM can achieve smaller latency than SAAG can do.

Now we evaluate the convergence of the optimal value obtained by the JCORM algorithm. We denote

$Q[g(\hat{\theta}) - v^\star] \leq \epsilon \cdot v^\star$ as a `counter` to calculate the number of feasible solutions, where the formula $g(\hat{\theta})$ can be obtained by the assumption of distribution of input size of tasks. We set $S = 80$, $l \in \{100, 200, 300, \cdots, 900\}$. The proportion of obtaining `counter`-requesting solutions $\hat{\theta}_S^l$ with relative tolerance of the optimal value $v^\star$ is shown in Figure. 9.

It shows that when $\epsilon = 0.04$ and the sample size is 900, nearly 85% of the sample values satisfy with the guaranteed range of the gap. According to the simulation we have done, we find that by increasing the number of the samples, the number of the scenarios, or decreasing the value of ϵ, more `counter`-requesting solutions can be obtained.

VII. CONCLUSIONS

In this paper, we study a computation offloading strategy based on density of IoT devices. We first apply CNF algorithm to divide edge servers into cooperative networks. We evaluate that the proposed algorithm on network division can obtain smaller global cost and variance of latency. Then we formulate a stochastic integer programming problem and propose SAA-based JCORM algorithm to solve it, which is significantly outperform the greedy algorithm method. Notice that theproposed algorithm is suitable for heterogeneous networks.

There are several aspects we need to focus on in future. Firstly, energy consumption for executing and transmission should be jointly considered with the overall cost of latency. Secondly, it's significant to provide an optimal offloading strategy under the premise of maintaining the stability of the time domain system. We need to dig further on how to construct a big data platform on formed cooperative networks in practice. Furthermore, we plan to investigate the mobility of each IoT devices to make our algorithm more reliable in a real scene. Due to the complexity of integer optimization problem on a large scale, we need to apply a more efficient strategy by reducing the number of samples in our algorithm and optimizing our mathematical model.

REFERENCES

[1] S. Deng *et al.*, "Toward mobile service computing: Opportunities and challenges," *IEEE Cloud Comput.*, vol. 3, no. 4, pp. 32–41, Jul. 2016.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," ETSI, Sophia Antipolis, France, ETSI White Paper 11, 2015, pp. 1–16.

[3] S. Deng, L. Huang, D. Hu, J. L. Zhao, and Z. Wu, "Mobility-enabled service selection for composite services," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 394–407, May 2016.

[4] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.

[5] X. Ge, J. Ye, Y. Yang, and Q. Li, "User mobility evaluation for 5G small cell networks based on individual mobility model," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 528–541, Mar. 2016.

[6] S. Deng, L. Huang, J. Taheri, J. Yin, M. Zhou, and A. Y. Zomaya, "Mobility-aware service composition in mobile communities," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 3, pp. 555–568, Mar. 2017.

[7] W. J. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling spatial and temporal dependencies of user mobility in wireless mobile networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 5, pp. 1564–1577, Oct. 2009.

[8] J. Wang, "Exploiting mobility prediction for dependable service composition in wireless mobile ad hoc networks," *IEEE Trans. Services Comput.*, vol. 4, no. 1, pp. 44–55, Jan. 2011.

[9] S. Deng, H. Wu, W. Tan, Z. Xiang, and Z. Wu, "Mobile service selection for composition: An energy consumption perspective," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 3, pp. 1478–1490, Jul. 2017.

[10] S. G. Wang, Y. L. Zhao, L. Huang, J. L. Xu, and C. H. Hsu, "QoS prediction for service recommendations in mobile edge computing," *J. Parallel Distrib. Comput.*, pp. 1–11, Oct. 2017, doi: 10.1016/j.jpdc.2017.09.014.

[11] Y. Mao, J. Zhang, Z. Chen, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[12] A. Ndikumana *et al.* (2018). "Joint communication, computation, caching, and control in big data multi-access edge computing." [Online]. Available: https://arxiv.org/abs/1803.11512

[13] L. Chen, S. Zhou, and J. Xu. (2017). "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks." [Online]. Available: https://arxiv.org/abs/1703.06058

[14] J. L. D. Neto, S.-Y. Yu, D. F. Macedo, J. M. S. Nogueira, R. Langar, and S. Secci, "ULOOF: A user level online offloading framework for mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 17, no. 11, pp. 2660–2674, Nov. 2018.

[15] W. Wiesemann, R. Hochreiter, and D. Kuhn, "A stochastic programming approach for qos-aware service composition," in *Proc. 8th IEEE Int. Symp. Cluster Comput. Grid (CCGRID)*, May 2008, pp. 226–233.

[16] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[17] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello, "The sample average approximation method for stochastic discrete optimization," *SIAM J. Optim.*, vol. 12, no. 2, pp. 479–502, 2002.

[18] S. Deng, L. Huang, H. Wu, and Z. Wu, "Constraints-driven service composition in mobile cloud computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2016, pp. 228–235.

[19] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[20] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[21] S. Deng *et al.*, "Toward risk reduction for mobile service composition," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1807–1816, Aug. 2016.

[22] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[23] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

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

[25] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, nos. 2–3, pp. 203–221, Aug./Sep. 1996.

[26] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[27] CB Networks. *Backhauling X2*. Accessed: Feb. 2, 2018. [Online]. Available: http://cbnl.com/resources/backhauling-x2/

[28] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York, NY, USA: Springer, 2011.

[29] W.-K. Mak, D. P. Morton, and R. K. Wood, "Monte Carlo bounding techniques for determining solution quality in stochastic programs," *Oper. Res. Lett.*, vol. 24, nos. 1–2, pp. 47–56, 1999.

[30] G. Casella and R. L. Berger, *Statistical Inference*, vol. 2. Pacific Grove, CA, USA: Duxbury, 2002.

[31] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "A sample average approximation-based parallel algorithm for application placement in edge computing systems," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2018, pp. 198–203.

[32] S. Ahmed, A. Shapiro, and E. Shapiro, "The sample average approximation method for stochastic programs with integer recourse," Georgia Inst. Technol., Atlanta, GA, USA, ISyE Tech. Rep., 2002, pp. 1–24.

CHENG ZHANG received the M.S. degree in electrical engineering from Zhejiang University, China, in 2013, where he is currently pursuing the Ph.D. degree in computer science and technology. His research interests include edge computing and machine learning.

HAILIANG ZHAO was born in 1997. He is currently pursuing the bachelor's degree. His research interests include edge computing and machine learning.

SHUIGUANG DENG received the B.S. and Ph.D. degrees in computer science from the College of Computer Science and Technology, Zhejiang University, China, in 2002 and 2007, respectively. In 2014, he joined the Massachusetts Institute of Technology as a Visiting Scholar. In 2015, he joined Stanford University as a Visiting Scholar. He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang University. During the past 10 years, he has published more than 100 papers in journals and refereed conferences. His research interests include edge computing, service computing, mobile computing, and business process management. In 2018, he received the Rising Star Award by the IEEE TCSVC. He serves as an Associate Editor for the IEEE ACCESS and the *IET Cyber-Physical Systems: Theory & Applications*.