This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

Traffic and Computation Co-Offloading with Reinforcement Learning in Fog Computing for Industrial Applications

Yixuan Wang, Kun Wang, *Senior Member, IEEE* Huawei Huang, *Member, IEEE*
Toshiaki Miyazaki, *Senior Member, IEEE* and Song Guo, *Senior Member, IEEE*

***Abstract*—In the last decade, network data communication has experienced a rapid growth, which has led to explosive congestion in heterogeneous networks. Moreover, the emerging industrial applications such as automatic driving put forward higher requirements on both networks and devices. On the contrary, running computation-intensive industrial applications locally is constrained by the limited resources of devices. Correspondingly, fog computing has recently emerged to reduce the congestion of content-centric networks. It has proven to be a good way in industry and traffic for reducing network delay and processing time. In addition, device to device (D2D) offloading is viewed as a promising paradigm to transmit network data in mobile environment, especially for auto-driving vehicles. In this paper, jointly taking both the network traffic and computation workload of industrial traffic into consideration, we explore a fundamental tradeoff between energy consumption and service delay when provisioning mobile services in vehicular networks. In particular, when the available resource in mobile vehicles becomes a bottleneck, we propose a novel model to depict the users' willingness of contributing their resources to the public. We then formulate a cost-minimization problem by exploiting the framework of Markov decision progress (MDP) and propose the dynamic reinforcement learning scheduling algorithm and the deep dynamic scheduling algorithm to solve the offloading decision problem. By adopting different mobile trajectory traces, we conduct extensive simulations to evaluate the performance of the proposed algorithms. The results show that our proposed algorithms outperform other benchmark schemes in the mobile edge networks.**

***Index Terms*—Fog Computing, Traffic Offloading, Computation Offloading, Reinforcement Learning, Industrial Application.**

I. Introduction

THE volume of mobile network traffic increases drastically with the exponentially growing number of mobile devices in the recent years. Thanks to the rapid development of wireless access technologies, cellular networks are able to transmit network traffic at high rates. The traffic of many applications is migrated from conventional datacenter to wireless edge networks gradually for less time delay. However, the consequence is that access networks become even more congested. For example, Cisco released that the global mobile network traffic showed a growth by 74% in 2016 compared with that of the previous year [1]. It is also forecasted that the global mobile network traffic will exceed 30.6 exabytes each month by 2020. In addition to tablets and smartphones, Cisco also reported that the emerging D2D modules will further contribute to the explosion of network traffic. In the future, vehicular network also has potential demand on the network services, making the network more congested.

Moreover, lots of industrial applications, such as fleet tracking and emerging automatic driving, are delay-sensitive and computing-hungry. As a result, network devices often exceed their capacity [2]. Although cloud computing enables flexible configuration of hardware resources, and allows both the network traffic and the computation-intensive tasks to be uploaded to cloud. This is not an efficient approach for industrial applications, as it may not only impose a huge heavy burden on congested core networks [3], but also cause intolerable transmission delay that degrades the quality of service [4, 5].

As a solution to the dilemma, fog computing, which is also known as edge computing [6], has been widely adopted to direct partial workloads to be processed locally without transferring them to the centralized remote cloud. In the mobile network environment, network edge facilities, for instance, base stations and routers are equipped with computing and storage capabilities to meet devices' requirements. Therefore, the service latency can be substantially reduced because vehicles are close to data sources.

The advantages obtained from offloading network traffic and computation in fog computing include reducing both the task response time and the energy consumption for industrial applications. Edge offloading particularly benefits the industrial applications which require massive computation on a small volume of data. Nevertheless, as mentioned, there are also some industrial applications without heavy computation but with huge volume of data, e.g., product tracking. An option is to exploit the idle resources from other devices through D2D communications [7]. This manner is also called the D2D offloading. It has been shown in some studies that mobile traffic can be significantly reduced by D2D communication [8],

Y. Wang is with the National Engineering Research Center of Communications and Networking, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: yxwang.cs@gmail.com).

K. Wang is with Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China (e-mail: kwang@njupt.edu.cn).

H. Huang is with the Academic Center for Computing and Media Studies, Kyoto University, Japan (e-mail: hwhuang@media.kyoto-u.ac.jp).

T. Miyazaki is with the School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu City 965-8580, Japan (e-mail: miyazaki@u-aizu.ac.jp).

S. Guo is with The Hong Kong Polytechnic University Shenzhen Research Institute and Department of Computing, The Hong Kong Polytechnic University, China (e-mail: song.guo@polyu.edu.hk).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

especially when many network nodes in an event request the same content from an edge server. For instance, in vehicular networks for industrial traffic, lots of local and network data and intensive computing power are needed. In addition, we cannot ignore the applications that are with both huge volume and intensive computation. It is a valuable research issue to find the optimal policy to offload the network tasks and traffic.

In order to reduce the latency of industrial applications, devices can run the industrial applications locally because no transmission or queueing delay incurs. Nevertheless, running applications locally will consume a lot of energy, it may shorten the battery lifespan of mobile devices. On the other hand, it can reduce the energy consumption for devices to offload the computing workloads to an edge server. But transmission delay will be induced, including the communication delay between the device and the edge server [9] as well as the waiting time at the edge server. Therefore, it is significant to analyse the tradeoff between the service delay and the energy consumption, which has a great influence on the quality of network and industrial applications.

In general, for mobile devices, when the computation of task is not heavy, users incline to run it locally to avoid long delay [10]. By contrast, users would like to offload the task to edge server when it is compute-intensive. However, if mobile devices invariably get the content data with high volume from the edge server, it will lead to more congested network traffic and unavoidable cost [11]. Under the circumstances, it is promising to offload traffic via D2D communication, especially for intra-group users who have similar content preference. It would be helpful to improve user experience if devices or users can share their resources. However, the question is that users may not be willing to share when their resources are scarce. The sharing willingness of users is a critical factor in D2D communication. To the best of our knowledge, we have not found a model that depicts such willingness for D2D communication. Thus, a novel model should be proposed to achieve the resource sharing in fog networks. In reality, residual energy (remaining battery level), interface bandwidth and computing power (CPU clock frequency) are the main factors that will influence the offloading decisions [12].

In this paper, we study the co-offloading of both traffic and computation for vehicular networks in industrial traffic. First, we devise a feature table that records the characteristic of tasks based on the content-centric architecture. Jointly taking both the traffic size and computation workload into consideration, we then propose a method that can both offload the network tasks with large volume and intensive computation in fog computing.

We summarize our contributions here as follows.

- We propose a novel formulation that jointly considers vehicle mobility and resource constraint, aiming to meet the offloading requirement of traffic vehicles. By focusing on the response latency and execution consumption, we analyse the tradeoff between the service delay and energy consumption in the edge offloading.
- We devise two reinforcement learning (RL) based algorithms to address the proposed cost-minimization problem. The results show that our proposed algorithms achieve lower energy consumption and service delay compared with other benchmark offloading schemes.

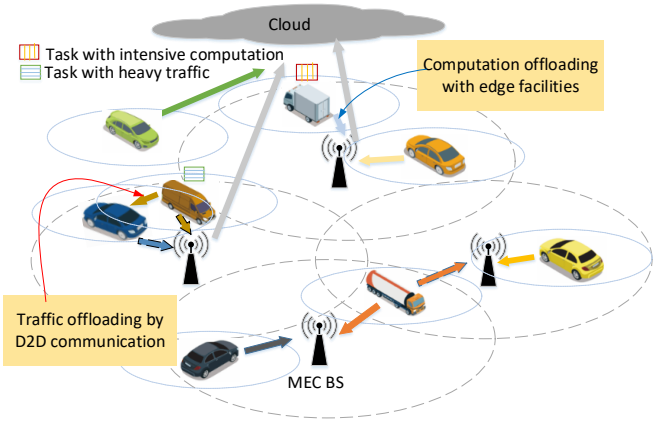


Fig. 1: System model

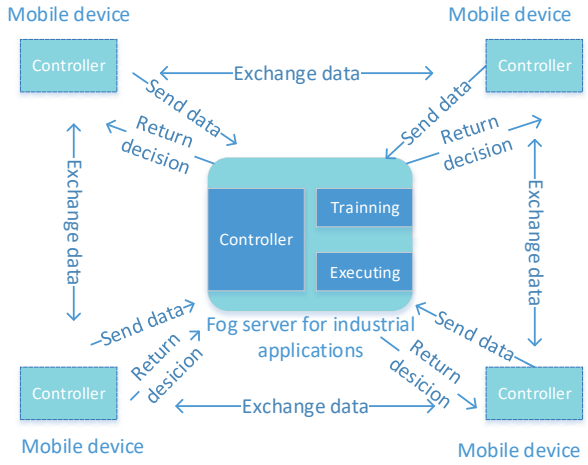


Fig. 2: Network architecture

The rest of this paper is organized as follows. Section II introduces the related work. Section III describes the system model of edge offloading in fog computing. RL-based algorithms are proposed in section IV. In Section V, the proposed approach is evaluated by simulations. Section VI concludes the paper and discusses some open issues.

II. Related Work

There are some studies about industrial applications in fog computing. Most of them focus on processing data and improving network services. Aazam *et al.* presented an architectural overview of Industry 4.0 and discuss how fog can provide local computing support in the industrial IoT [13]. Fu *et al.* designed a flexible and economical framework by integrating the fog computing and cloud computing to solve the problems about data in industrial applications [14]. Mishra *et al.* proposed a metaheuristic-based service allocation framework using three metaheuristic techniques, such as particle swarm optimization (PSO), binary PSO, and bat algorithm [15]. Kaur *et al.* presented an SDN-based edge-cloud interplay to handle streaming big data in industrial IoT environment, wherein SDN provides an efficient middleware support [16].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

TABLE I: NOTATIONS

Notation	Description
N	set of natural number
t	each decision time-slot
D	expiry time of system process
U	set of mobile devices that need to offload tasks
γ	time requirement of network task
u	attributes of network task
v	attributes of mobile device
s_i	traffic size of task i to transmit
c_i	computation workload of task i to dispose
$r(t)$	network transmission rate at time t
f_j	CPU clock frequency of mobile device j
T_{tran}	service delay to transmit data content
T_{comp}	service delay to handle computation workload
E_{tran}	energy consumption to transmit the data content
E_{comp}	energy consumption to handle the computation workload
Q	tradeoff index of service delay and energy consumption
$\mathbb{L}$	set of mobile device locations
$\mathbb{A}$	set of mobile device actions
$\mathcal{T}$	set of network tasks
C_t	network cost at time slot t
E	expected value
c_{max}	constant upper-bound of computation workload
s_{max}	constant upper-bound of traffic size

In order to address the challenges of dynamic network traffic, lots of methods have been proposed. For example, Ha *et al.* [17] proposed a pricing system architecture, which intends to handle the increasing demands in mobile networks. Zhuo *et al.* [18] considered utilizing delay tolerant networks and WiFi to offload traffic of cellular networks. Based on this, they proposed an incentive method to simulate mobile users to balance service delay in network traffic offloading.

These pioneering researches have concentrated on offloading mobile traffic to WiFi. On the other hand, as D2D communication emerges, some other studies have investigated that offloading network traffic via D2D communication. For example, Habak *et al.* [19] utilized an assemblage of co-location mobile devices to build a mobile edge system dynamically. Xie *et al.* [20] made use of energy efficient cooperative communication when guaranteeing transmission reliability.

We find the study that can yield the optimal policies of offloading network tasks is still missing. Many existing studies focused on only a particular type of network or type of task [21, 22]. Besides, it should not be ignored that the schemes to collect the distributed information of mobile devices and assign tasks to suitable device or edge server are the key challenges in mobile edge system [23]. Different from existing methods, the scheme we proposed not only considers reducing the energy consumption and service delay of mobile tasks, but also takes the resource condition of mobile devices into account. This work fills the gap that handles the co-offloading towards the traffic and computation in MEC.

III. System Model And Problem Formulation

A. System Model

In MEC, mobile devices can usually access to network services of industrial applications via edge networks or cellular networks. The service requirements are provisioned in either edge servers or remote cloud datacenter.

For minimizing the energy consumption and access delay, the service provider needs to make offloading decisions for each mobile device in real-life industrial traffic or services [15, 24]. We assume there is a centralized controller that makes decisions for the service provider.

As shown in Fig. 1, there are lots of mobile devices. We assume that a set U of mobile devices exists in the edge network. Each one requires a mobile service of an application during the process moving from a starting position to the next position. The mobile services are included in a set $\mathcal{T}$ of network tasks while the location is denoted as a set $\mathbb{L}$. When a mobile device moves around, it can receive data in two manners: (1) direct data transmission from edge network devices, and (2) D2D based data transmission when meeting other devices that have sufficient resources and hold the requested data. The chance of offloading mobile data to edge servers or mobile devices depends greatly on the network data characteristics (i.e. traffic size and computation workload). If a task is computation-intensive, then mobile device can offload this task to an edge server for execution. When the volume of task's data is large, controller can transmit the data though a D2D communication to avoid the network traffic congestion.

As shown in Fig. 2, the fog server has a controller which can receive the data from the mobile devices. The training process takes place on the fog server with strong computing power. The controller will make decisions according to the collected information and return the results to mobile devices to help them choose the optimal actions. On the other hand, each mobile devices also has a controller that can make decisions. For example, it will decide to execute the pre-trained model on the mobile device only in the case that edge network is unavailable. If needed, one mobile device will use D2D communication by exchanging data with others. As described above, the training process of deep neural network takes place on the fog server with strong computing power. The mobile devices do not need to train the model. Therefore, the overhead is not heavy and it is acceptable.

B. Problem Formulation

We consider a scenario in a content-centric network, where each network task has a unique identifier. We use a global-sharing feature table to record the characteristics of tasks, such as traffic size and computation workload. The mobile devices and edge server can acquire the characteristics of a network task easily. We assume that task $i \in \mathcal{T}$ can be measured by the content size s_i and the amount of computation c_i. Note that, s_i and c_i can be discrete real numbers. For example, the task data can be divided into multiple equal chunks, and the computation is dependant on the number of instructions. When a mobile device $j \in U$ wants to transfer the i^{th} task to an edge server or to the other mobile devices, the time spending on transferring the data is calculated as:

$$T_{tran}(i, j, t) = \frac{s_i}{r(t)}, \forall i \in \mathcal{T}, \forall j \in U, t \in \mathbb{T}, \quad (1)$$

where $r(t)$ is the network transmission rate at time slot t, and its value depends on the environment where the device is located at time t. $\mathbb{T}$ is the set of all time slots. If a device processes a task locally, this transmission delay is not incurred. In this case, the content size s_i will be set to 0. On the other

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

hand, if a task is computation-intensive, it must be handled on mobile devices or offloaded to an edge server for execution. It may also happen that neither of these two modes is feasible, e.g., when energy is insufficient at the mobile device which should execute the task, and hence the computation task will be postponed till next available time slot. When the computation task is executed locally, it will utilize the computing resource of the mobile device, including CPU, memory, storage, battery capacity and so on. In this paper, we focus on the CPU and battery level of mobile devices, which are denoted by f_j and $b_j (j \in U)$, respectively. Therefore, we can write the execution time on a local mobile device, denoted by $T_{comp}(i, j, t)$, as follows:

$$T_{comp}(i, j, t) = \frac{c_i}{f_j}, \forall i \in \mathcal{T}, \forall j \in U, t \in \mathbb{T}. \quad (2)$$

Thus, the total response time of offloading a task for execution can be calculated as follows:

$$T_{total}(t) = \sum_{j \in U} \sum_{i \in \mathcal{T}} [T_{tran}(i, j, t) + T_{comp}(i, j, t)], t \in \mathbb{T}. \quad (3)$$

In addition, the energy consumption also stems from different procedures of the system, such as the computation and transmission. The specific facts are CPU utilization and I/O transmission. Thus, for the i^{th} network task and the j^{th} mobile device, we calculate the total power consumption as follows:

$$p_{total}(i, j, t) = p_{comp}(i, j) + p_{tran}(i, j, t), \forall i \in \mathcal{T}, \forall j \in U, t \in \mathbb{T}, \quad (4)$$

where $p_{comp}(i, j)$ is the power consumption when the task i is executed on the mobile device j, while $p_{tran}(i, j, t)$ is the power consumption when the task i is transferred from the mobile device j to the edge server or other mobile devices. Similar to [9], we assume that the power consumption for computation is represented as follows:

$$p_{comp}(j) \propto (f_j)^3, \forall j \in U. \quad (5)$$

Then, the energy consumption of execution is calculated as

$$E_{comp}(i, j, t) = \alpha_1 c_i (f_j)^2, \forall i \in \mathcal{T}, \forall j \in U, t \in \mathbb{T}, \quad (6)$$

where α_1 is the coefficient of energy consumption in computing. Moreover, according to [25], the energy consumption caused by transmission is proportional to the size of transferred data and the data transmission time. Hence, we can define the $E_{tran}(i, j, t)$ as follows:

$$E_{tran}(i, j, t) = \alpha_2 \frac{s_i}{r(t)}, \forall i \in \mathcal{T}, \forall j \in U, \forall t \in \mathbb{T}, \quad (7)$$

where α_2 is the transformation coefficient of energy consumption in transmission. The total energy consumption can be obtained as follows:

$$E_{total}(t) = \sum_{j \in U} \sum_{i \in \mathcal{T}} [E_{comp}(i, j, t) + E_{tran}(i, j, t)], \forall t \in \mathbb{T}. \quad (8)$$

As stated above, we can describe the tradeoff between the energy consumption and the service delay by tuning a knob $\delta \in [0, 1]$ denoting the weight of energy consumption. Finally, Q is calculated as follows:

$$Q_t = \delta E_{total}(t) + (1 - \delta) T_{total}(t), \delta \in [0, 1], t \in \mathbb{T}, \quad (9)$$

where Q_t denotes the system cost at time slot t. When we set δ to 1, the weighted sum will turn into energy cost. On the other hand, it will turn into time cost if δ is set to 0. Moreover, we can tune the parameter if we are more concerned about one aspect.

For all the tasks that can be offloaded, we assume they need to be accomplished before deadline $D, D \in N$. $\mathbb{L}$ is the set of coordinates that mobile devices could move around before D. The system state for devices and network tasks is defined as $S = (u, v, \mathbb{H})$, where u is the attributes of mobile devices and v is the attributes of network tasks. The set $\mathbb{H}$ includes the locations of other mobile devices that do not need offload. In detail, $u = (l, b, f)$, where l, b and f denote location of mobile device, battery level and CPU clock frequency, respectively. $v = (s, c, \gamma)$, where s, c and γ are the traffic size, computation workload and delay requirement of a task. To simplify, we can partition coverage areas into discrete ones by grid method and set the delay requirement to several levels. The state space is up to $O(|S|^{|\mathcal{T}|})$. The mobile controller should choose one way to handle the task at each time slot $t \in \mathbb{T}$.

In general, the cellular network can provide seamless coverage to all the mobile devices but its quality may not meet the demands in the future. Thus we only consider the edge networks and D2D communications, but cellular network is only be considered when a task is unfinished. We classify the locations by available edge networks or D2D communications at time t into the following four cases. (1) $\mathbb{L}_t^1 = l \in \mathbb{L}$: l can only access to cellular network, which is not be used in most cases; (2) $\mathbb{L}_t^2 = l \in \mathbb{L}$: l can access to edge network; (3) $\mathbb{L}_t^3 = l \in \mathbb{L}$: l can access to D2D network; (4) $\mathbb{L}_t^4 = l \in \mathbb{L}$: l can access to edge network and D2D network.

There are four actions corresponding to offloading decisions for mobile devices. Mobile controller will select one of the actions for dealing with a network task at each decision timeslot. Accordingly, we can show the actions using the set $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, where a_1 denotes that mobile device will wait for an opportunity to handle the task; a_2 denotes that a mobile device will dispose the task locally; a_3 denotes that task will be handled through an edge network, and a_4 denotes that task will be disposed through a D2D communication. Note that a_3 is available when mobile devices are in the coverage of an edge network and a_4 is available when a mobile device meets other devices.

The available action at location l is denoted as $a \in \mathbb{A}$. We define the possible actions on the basis of the attributes of system.

$$\mathbb{A} = \begin{cases} a_1, a_2, & l \in \mathbb{L}_t^1 \\ a_1, a_2, a_3, & l \in \mathbb{L}_t^2 \\ a_1, a_2, a_4, & l \in \mathbb{L}_t^3 \\ a_1, a_2, a_3, a_4, & l \in \mathbb{L}_t^4 \end{cases}, \forall t \in \mathbb{T}. \quad (10)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

We define the cost function for every action based on the action chosen at each time slot, denoted by C_t, as follows:

$$C_t = Q_t, \forall t \in \mathbb{T}. \quad (11)$$

The action cost occurs during the usage of the resource and transmission of data. In general, the consumption to offloading task using an edge network is lower than that using a D2D communication. The total cost of dealing with the task is the sum of cost units occurring at each time slot during the whole process.

However, there may be a few tasks that cannot be processed before the deadline. For the failed task (i.e. $s > 0$ or $c > 0$ when $t > D$), we define the penalty cost function as (12), which is calculated by the remaining traffic size s_i^r and computation workload c_i^r.

$$C_{D+1} = \sum_{i \in \mathcal{T}} \frac{1}{\gamma_i} \sqrt{\mu (s_i^r)^2 + (1-\mu)(c_i^r)^2}, \mu \in [0,1], \quad (12)$$

where $\mu \in [0,1]$ is a coefficient balancing traffic size and computation workload.

We assume that the movement mode of mobile devices is memoryless, which is defined in (13). The new location l' just depends on the past location l and is irrelevant with the characteristics of task. We think about a two dimension movement mode. For mobile devices, they can stay where they are with probability λ at each decision time-slot. The probability λ is called stable factor. Besides, they may move randomly to neighbour locations with probability $\eta_k, k \in \{east, south, west, north\}$ indicates one of four directions which mobile devices can move to. Specifically, the stable factor λ and the probability of moving to each direction η_k satisfy the following rules:

$$P(l'|l) = \begin{cases} \lambda, & l' = l \\ \eta_k, & l' \neq l \end{cases}, \quad (13)$$

where

$$\lambda + \sum \eta_k = 1, k \in \{east, south, west, north\} \quad (14)$$

Because mobile devices can randomly move before the deadline, we are concerned about the situation where they can encounter with each other at a certain location. The probability that mobile device n can meet with the other device m at location l at decision time-slot t is denoted as $P_t^n(l) \cdot P_t^m(l)$, in which $P_t^n(l)$ is defined in (15):

$$P_t^n(l) = \begin{cases} 1, & P_t^n(l) = P_0^n(l_n) \\ \sum_{l' \in \mathbb{L}} P_{t-1}^n(l') P(l|l'), & P_t^n(l) \neq P_0^n(l_n) \end{cases}. \quad (15)$$

It shows the probability during decision time-slot t that mobile device n stays in location l. l_n is the starting location of mobile device before offloading procedure. Furthermore, the mobile devices will decide whether to provide the device resources or not. When the resources are sufficient, users are ready to offer them for better benefits such as reducing service delay or improving energy-efficient. On the contrary, they will refuse to sacrifice their needs if the resources are scarce. Therefore, we devise the following indicator model which reflects the willingness of users to provide the resources for public benefits. It is reasonable that the share willingness will be higher if the network transmission rate is faster and CPU frequency is higher. Besides, the formula $\sqrt{b_t^n(2-b_t^n)}$ reflects the fact the share willingness will drop sharply when the battery power is at a low level while it will drop slowly when the battery power is at a high level.

$$W_t^n = \begin{cases} 1, & b_t^n > b_{thr1} \\ \beta f_n r(t) \sqrt{b_t^n(2-b_t^n)}, & \text{otherwise} \\ 0, & b_t^n < b_{thr2} \end{cases}, \quad (16)$$

where b_t^n is a battery level expressed as a percentage of device n at time t, and b_{thr1} and b_{thr2} are two different thresholds. b_{thr1} is the threshold that users are willing to provide the resources because the resources are abundant, while b_{thr2} is the threshold denoting users refuse to offer due to insufficient resources. β is a normalized coefficient to make the probability ranging from 0 to 1. Thus, the probability that one mobile device can connect with another satisfies:

$$P_t^n(l, W_t^n) = P_t^n(l) \cdot W_t^n, t \in \mathbb{T}, n \in U. \quad (17)$$

As a result, the probability that two mobile users can connect with each other satisfies:

$$P_t(l, n_1, n_2) = P_t^{n_1}(l, W_t^{n_1}) \cdot P_t^{n_2}(l, W_t^{n_2}), n_1, n_2 \in U. \quad (18)$$

The transition probability of system is the probability that the state of system varies from S to S' in the next time-slot when taking action $a \in \mathbb{A}$. Because the movement of mobile device is independent of the characteristics of task $\mathbb{T}$ and action $\mathbb{A}$, we get

$$P(u', v'|u, v, a) = P_t(l, n_1, n_2) P(l'|l) \prod_{h \in \mathbb{H}} P(h'|h) \cdot P(u', v'|u, v), \quad (19)$$

where

$$P(u', v'|u, v) = \begin{cases} 1, & u' = u - \Delta u(a) \text{ and } v' = v - \Delta v(a) \\ 0, & \text{otherwise} \end{cases}. \quad (20)$$

The probability that mobile device moves from location l to l' is $P(l'|l)$, and the probability that other mobile devices move from location h to h' is $P(h'|h)$. $P(u', v'|u, v, a)$ is the probability that attributes of system vary from the current state to another in the next decision time-slot.

IV. Reinforcement Learning Based Algorithms

In order to solve the optimal offloading problem, we propose the dynamic reinforcement learning scheduling (DRLS) algorithm (*Algorithm 1*) and the deep dynamic scheduling (DDS) algorithm (*Algorithm 2*), based on the framework of reinforcement learning algorithm [8, 26, 27]. The policy π denotes the decision according to each system state S and decision time-slot t, which is denoted as $a = \pi_t(S)$ in MDP. We define the policy space of π as Π. Then, our objective is to find the optimal policy π, which can minimize the system cost for processing the task before deadline D. Recall that the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

total cost during the offloading process can be calculated as follows:

$$C_{total} = \sum_{t=1}^{D} C_t(S, \pi_t(S)) + C_{D+1}(S), \forall t \in \mathbb{T}. \quad (21)$$

Then, a cost-minimization problem with the objective function (21) is defined as follows:

$$\min_{\pi \in \Pi} \quad E_S^{\pi}(C_{total}) \quad (22)$$

$$s.t.\ 0 \le t \le D, t \in \mathbb{T}, \quad (23)$$

$$0 \le b_j \le 1, j \in U, \quad (24)$$

$$0 \le c_i \le c_{max}, i \in \mathcal{T}, \quad (25)$$

$$0 \le s_i \le s_{max}, i \in \mathcal{T}, \quad (26)$$

where E means the expected value, c_{max} and s_{max} are the constant upper-bounds of c_i and s_i ($i \in \mathcal{T}$), respectively. Constraint (22) describes that each time slot t is before the deadline D. Constraint (23) indicates that the battery level expressed as a percentage ranges form 0 to 1. Constraints (24) and (25) show that traffic size and computation workload should be within the specified ranges, respectively.

A. Dynamic Reinforcement Learning Scheduling Algorithm

In order to solve the offloading decision problem, we propose the DRLS to obtain the optimal policy. In Algorithm 1, we define the following value function to compute the optimal policy.

$$V_t^*(S) = \arg\min_{a \in \mathbb{A}} Q_t(S, a), \forall t \in \mathbb{T}, \quad (27)$$

where

$$\begin{aligned} Q_t(S,a) &= \sum_{S' \in S} P(S'|S,a)[c_t(S,a,S') + V_{t+1}^*(S')] \\ &= \sum_{S'|S} Q_t + \sum_{(u',v',\mathbb{H}') \in \mathbb{S}} P(S'|S,a) V_{t+1}^*(S') \\ &= Q_t + \sum_{(u',v',\mathbb{H}') \in \mathbb{S}} P(u'|u) \prod_{h \in \mathbb{H}} P(h'|h) P(v'|u,v) \cdot \\ &\quad V_{t+1}^*(S') \\ &= Q_t + \sum_{(l',h') \in \mathbb{L}} P_t(l', W_t') \prod_{h \in \mathbb{H}} P(h'|h) V_{t+1}^*(S'). \end{aligned} \quad (28)$$

We should note that the first formula in (28) indicates that $Q_t(S,a)$ includes the current cost indicated by adopting the action a as well as the future cost when system state S changes into S'.

Our DRLS algorithm includes two phases: the offloading planning phase and the offloading executing phase. The inputs are initial states of system attributes and actions. The outputs are corresponding different $Q_t(S,a)$ values, among which the minimum value corresponds to the optimal action policy in that state. In each iteration, an optimal policy will be generated in the planning phase by referring to (29). In particular, we use $V_t^{n-1}(S)$ to calculate $V_t^n(S)$.

$$V_t^n(S) = \min_{a \in \mathbb{A}} \sum_{S' \in \mathbb{S}} P(S'|S,a)[C_t(S,a,S') + V_{t+1}^{n-1}(S')]. \quad (29)$$

Algorithm 1: Dynamic reinforcement learning scheduling algorithm

Input: $u(f,b,l), v(s,c,\gamma), \mathbb{H}, V_t^0(S)$
Output: $\pi_t^*(S)$

```
Planning phase
Initialize u, v and H
Initialize V_t^0(S) with (27) and (28)
repeat
    for t ∈ T do
        for l ∈ L do
            for s ∈ S, c ∈ C do
                compute V_t^n(S) using (29)
                compute the norm
                  rsd_t^n(S) = ||V_t^n(S) − V_t^{n−1}(S)||
until rsd_t^n(S) < ε
Return best policy π_t^*(S)
Executing phase
Set t = 1 and s = s_max, c = c_max
while t ≤ D and s > 0 and c > 0 do
    Choose action a = π_t^*(S)
    if s > r_2 × (D − t) × γ or c > f_2 × (D − t) then
        s = s − r(t)Δt
        c = c − f(t)Δt
    else
        u = u − Δu(a)
        v = v − Δv(a)
    t = t + 1
```

In executing phase, the controller will take the optimal action policy in the current state. Then, the task will be dealt with the chosen action until the completion of the task or the expiration of the time.

B. Deep Dynamic Scheduling Algorithm

In a MDP model, we assume that there is an agent in the environment. The agent is trained to run and make decision at each decision time-slot. The agent instructs the sequence with a certain state $S \in \mathbb{S}$ to perform a given action $a \in \mathbb{A}$. Then the system state moves to a new state S' and receives the reward $R(S,a)$ correspondingly. In this process, the chosen action and the current state jointly yield the probability that determines which state the process will move into.

The objective of a MDP model is to find the optimal strategies for different states according to the expectation of rewards without specific model. It does not even need to make a model. Therefore, the model can learn new states and actions while they become comprehensive. However, by training over time, the system has to learn the optimal policy. In detail, what we need is to optimize expectation reward at each decision time-slot and the discounted reward $\sum_{i=0}^{N} \varphi^i r_i$, where N is the step number. As a discount factor, $\varphi < 1$ reflects the importance of future rewards. The agent will strive for a long-term reward when $\varphi \to 1$, while agent is myopic if $\varphi = 0$ because it does not consider the future reward. If we define $Q(S,a)$ as the discounted expectation reward when the system agent chooses action a in state S, and $V(S) = \max_{a'} Q(S,a')$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

Algorithm 2: Deep dynamic scheduling algorithm

Input: Discount parameter ϕ, exploration rate ϵ, ξ and replay period K

Output: $(\Theta, \Omega, S_t, a_t)$

Initialize state table Θ, action table Ω to be empty table
Initialize prediction DQN Q with random weight θ, target DQN $\hat{Q}$ with $\theta^- = \theta$
Pre-train Q and $\hat{Q}$ with stored samples in $\mathcal{D}$
for *each decision time-slot t* **do**
  Capture the current state S_t
if $\Omega_t = \emptyset$ **then**
  Set action set Ω_t = State-Learning(S_t)
  Add Ω_t to Ω
Generate another random number ψ
if $\psi > \epsilon$ **then**
  Generate another random number σ
  if $\sigma > \xi$ **then**
    a_t = RandomActionSelection(S_t)
  else
    a_t = PrioritizedActionSelection(Ω_t)
else
  $a_t = \max_{\mathbb{A}} Q(S_t, \mathbb{A}; \theta)$
if $a_t = \emptyset$ **then**
  goto line 13
Execute action a_t in emulator, and observe reward R_t and new state S'
Set $Y_t^{DoubleDQN} = R_t + \phi \max_{\mathbb{A}} \hat{Q}(S', \mathbb{A}; \theta^-)$ and $\rho_t = |Y_t^{DoubleDQN} - Q(S_t, a_t)|$
Store transition $S_t, a_t, R_t, S', \rho_t$ in $\mathcal{D}$
Perform the stochastic gradient descent on $(Y_t^{DoubleDQN} - Q(S_t, a_t; \theta))^2$ regarding θ
if $t \equiv 0$ *mod* K **then**
  Sample minibatch of K transitions from $\mathcal{D}$ by prioritized experience replay
  Reset $\hat{Q} = Q$
UpdateQTable(Θ, Ω, S_t, a_t)

as the optimal value, our objective is to search an optimal policy with $V(S)$, which is calculated as:

$$V(S) = \max_a [R(S, a) + \varphi \sum_{S'} Pr(S'|S, a)V(S')]. \quad (30)$$

In fact, we can obtain the optimal solution by solving the equivalent Bellman's equation as follows: If we define $f(S_t)$ as the instantaneous revenue in state S_t, and a vector of utility function $V = [V(S_1), V(S_2) \cdots]$ that satisfies the Bellman's equation for the proposed problem such that:

$$\phi + V(S_t) = \max_{a^* \in \mathbb{A}} [f(S_t) + \sum_{S_t \in \mathbb{S}} Pr(S_t|S_t, a^*)V(S_t)]. \quad (31)$$

We then can obtain the optimal revenue scalar $\phi = \max_\Omega \overline{f}(\Omega)$.

Now we present the proposed algorithms based on deep Q-learning, which is a good choice for resource scheduling and management. Note that, the inputs are states and actions. The outputs are corresponding Q-values, among which the minimum value corresponds to the optimal action policy in that state. In order to derive the correlation of different state-action pairs and corresponding Q-values, Deep Neural Network is adopted in this algorithm. In fact, enough samples should be accumulated in the construction phase to accomplish the training process. Moreover, it could apply arbitrary policy in this process. An unsorted sum tree $\mathcal{D}$ stores the samples for prioritized experience replay and double DQN is constructed based on that.

In Algorithm 2, at the beginning of each decision time-slot t, the controller is able to capture the current system state. There are two actions that it can choose. One is the action according to the second level exploration with probability $1-\epsilon$ and the other is the action with maximum $Q_t(\mathbb{S}, \mathbb{A})$. Then the agent will carry out the chosen action and receive the corresponding rewards. In addition, the approximate $Q_t(\mathbb{S}, \mathbb{A})$ could be obtained by performing inference. This may improve the performance of the Q-values initialization. The Double DQN can use prioritized experience replay to update the target network. It can work well in a finite time.

V. Performance Evaluation

A. Experiment Settings and Metrics

We conduct experiments using both tensorflow and Matlab. At each decision time-slot, mobile users or devices can choose a moving direction according to (13). Without loss of generality, we can assume that the battery level of mobile devices obeys normal distribution. Besides, we examine our model with different trajectory traces generated randomly. We show the average value with different network settings in terms of the locations of edge devices and attributes of network tasks in each group of simulations. The length between two adjacent decision time-slots is set to be 5 seconds. The network transmission rates of the edge network, D2D communication and cellular network are 10 Megabits per second (Mbps), 1 Mbps and 2 Mbps, respectively.

In order to evaluate the offloading performance, We focus on the following three metrics : (1) the energy cost, which is caused by mobile devices during the processes of data transmission and task computation. (2) the delay spending on data transmission and computation, and (3) the offloading ratio, which represents the percentage of network traffic that task transmits through the edge network or D2D communication. We compare our edge network and D2D communication joint offloading scheme (EDJO) with the following three schemes. (1) Non-offloading scheme (NO): task is handled by mobile device locally and the data is received from cellular network [28]. (2) Delayed D2D offloading scheme (DDO): task is disposed only by the D2D communication and cellular networks [28]. (3) Non-delayed offloading scheme (NDO): task is dealt with by edge networks and cellular networks [29].

We should notice that one mobile device can carry several types of network tasks and lots of mobile devices may require task content data simultaneously. Hence, only several mobile devices can receive the data from a certain mobile device at some point. This is very different from edge networks and cellular networks, where the availability is out of question. Besides, we assume that the mobile devices can access to the cellular networks all the time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

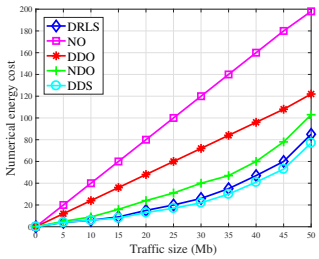


Fig. 3: Energy cost of schemes with traffic size

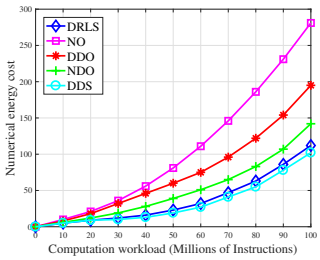


Fig. 4: Energy cost with computation workload

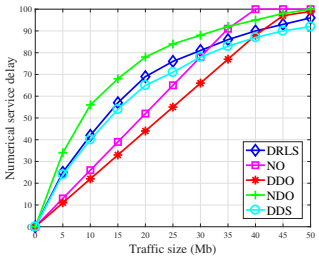


Fig. 5: Service delay of schemes with traffic size

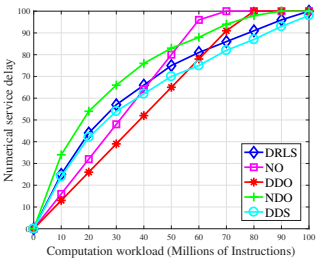


Fig. 6: Service delay with computation workload

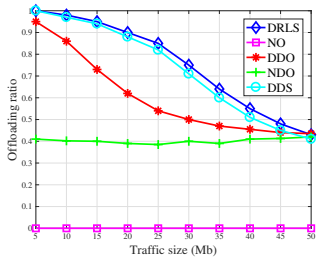


Fig. 7: Offloading ratio of schemes with traffic size

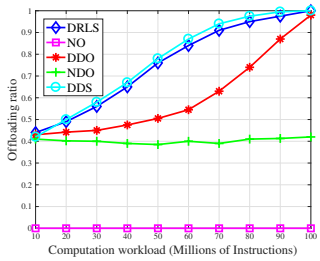


Fig. 8: Offloading ratio of schemes with computation workload

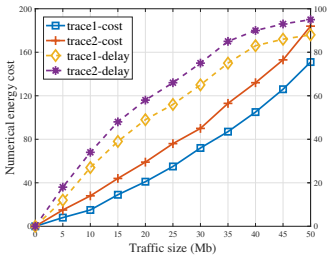


Fig. 9: Energy cost and service delay of two traces with traffic size

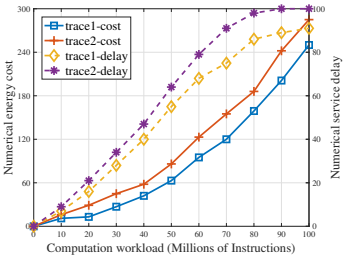


Fig. 10: Energy cost and service delay of two traces with computation workload

B. Simulation Results

In this section, we present the simulation performance of our offloading scheme in mobile edge network. As shown in Fig. 3 and Fig. 4, we can note that the variation of the energy consumption as traffic size and computation workload change. Moreover, in Fig. 3, the energy cost increases with the growth in traffic size. Besides, the DRLS and DDS we proposed have better performance than the three other offloading schemes due to the lower energy cost for any traffic size. In Fig. 4, similarly, the energy cost increases with the growth in computation workload. What's more, the DDS also outperforms other schemes with the computation workload changing. All of the observations demonstrate the good performance of our method.

Besides, we also find that NDO is better than NO and DDO on energy cost. From the experimental results, NO and DDO have inferior performance. NO is a scheme that mobile device does not choose to offload traffic or computation. The energy cost of mobile device will be heavy because it executes the tasks locally. On the other hand, DDO will make the mobile devices delayed to choose D2D communication. This will cause more energy cost than that of edge offloading. Both of these reasons can cause inferior performance. In addition, we observe that the computation workload has a larger effect on the energy cost than traffic size. This can be explained that energy consumption caused by transmitting the data is lower than that by executing the task when the amount of instructions is large. However, several studies may show that the communication consumes more energy than that of computation. The reason of the difference is that the large amount of instructions involved in computation of our experiments. The consumption of the computation is larger than that of communication due to heavy computational cost.

In Fig. 5 and Fig. 6, we can observe the variation of the service delay as the traffic size and computation workload change. As expected, Fig. 5 shows that the service delay increases with the growth in traffic size. We find that most of time NO and DDO achieve lower service delay compared with DRLS and DDS. The reason is that we assign greater weight on the energy consumption. Consequently, it will sacrifice the service delay to reduce energy consumption. If we put a greater weight on the service delay of network, our DRLS and DDS will be better than others in this regard. Moreover, they will outperform other schemes when traffic size is larger than 40 Mb or computation workload is greater than 65 M/I (millions of instructions). Besides, compared with the NDO, our scheme has a better performance on saving energy consumption due to it makes best of D2D communications. In Fig. 6, we also observe that service delay is longer with the increase in computation workload. We observe that our scheme also has a better performance than NDO because it optimizes the utilization of resources. Besides, the scheme DDO outperforms NO since it looks for opportunities to use a D2D communication, and it reduces the penalty value of using cellular network.

Fig. 7 and Fig. 8 show the variation of offloading ratio as traffic size and computation workload change. We can observe that offloading ratios of DDO, DRLS and DDS decrease with the increase of traffic size while the offloading ratio of NDO does not seem to change much. The reason is that NDO offloads network tasks based on the location, ignoring current system state. Besides, with the increase of traffic size, the offloading ratio of DRLS, DDS and DDO decease because the cost is high to offload the tasks. The difference of the experimental results is that DRLS and DDS have higher values than DDO. What makes this phenomenon is that DRLS and DDS uses alternative D2D communications to offload tasks except for edge networks. On the other hand, the offloading

ratios of DRLS, DDS and DDO raise with the increase of computation workload. This is because offloading the task with heavy computation to edge server can greatly reduce the energy consumption and execution time. We draw the conclusion that our scheme is able to achieve the lowest network cost while meeting the task completion deadline. It could also achieve the minimal energy cost or service delay compared with other schemes. Moreover, in most cases, it can outperform than others on all the three metrics. Besides, DDS has a slightly advantage over DRLS shown in results, but we could not ignore the overhead of DDS is generally larger than that of DRLS. This is an important problem for industrial applications in fog computing.

We also analyse the effect of offloading with different trajectory traces. We generate these two traces according to formula (13) and (14). In this paper, we can get trajectory trace1 with a distinct regularity by setting the parameter λ to 0.1, parameter η_{south} and η_{north} to 0.05, parameter η_{east} and η_{west} to 0.4, respectively. On the other hand, we can get trace2 with a large randomness by setting all the parameters to 0.2. Actually, these are used to simply simulate movement in two directions and random movement. As shown in Fig. 9 and Fig. 10, for any traffic size and computation workload, trace1 has a better performance than trace2 does on both energy cost and service delay. It means that we should consider the difference of movement modes in order to make better use of edge offloading for industrial applications.

Besides, the reliability of tasks is an important factor due to various network elements. It is generally true that the service provider can guarantee the reliability of the edge servers. However, the user-side reliability is difficult to solve. Of course, we do not have to be pessimistic. In future work, we can take advantage of 5G technology and design an incentive mechanism to alleviate this problem.

VI. Conclusion and Open Issues

In this paper, we mainly focus on the problem of edge offloading in fog computing. For D2D offloading, we propose an evaluation model to depict the willingness of mobile users in industrial traffic to contribute their resources to the public. Then, we devise a dynamic reinforcement learning scheduling algorithm and a deep dynamic scheduling algorithm to find a fine-balanced solution to offloading the tasks for mobile devices. The simulation results show that our scheme outperforms the other benchmark schemes in terms of both energy consumption and service delay. According to the results, we find that the energy cost and service delay increase with increasing of traffic size and computation workload. On the other hand, the computation workload has a larger effect on the energy cost than traffic size. This can be explained that energy consumption caused by transmitting the data is lower than that by executing the task when the amount of instructions is large. The offloading ratio decreases as the traffic size increases while it increases as the computation workload increases. Besides, considering the different effects in the case of two traces, we should analyse the difference of movement modes in order to make better use of offloading for industrial applications.

However, to deploy real industrial applications in fog computing, we need to address several challenges. Some of them are listed here for helping carry out research in this direction. The first issue is interoperability of edge devices. Because many edge facilities are involved in industrial applications, it is important to seek an effective and efficient method to handle the interoperability of edge devices. The second issue is security and privacy of data. Industrial edge facilities are vulnerable to attacks, which can affect the availability, confidentiality, and integrity of transmitted or stored data. The third issue is user friendliness in the application deployment and usage. The user interaction is very frequent in industrial applications. Therefore, it is challenging to design devices and user interfaces for applications because we should make them easy to accept for people of different backgrounds. Only by solving those challenges can we realize the true potential of industrial fog computing.

Acknowledgment

This work is supported by NSFC (61872195, 61572262, 61872310); China Postdoctoral Science Foundation (2017M610252) and China Postdoctoral Science Special Foundation (2017T100297); Shenzhen Basic Research Funding Scheme under research (JCYJ20170818103849343); Strategic Information and Communications R&D Promotion Programme (SCOPE No.162302008), MIC, Japan; the Open Research Fund of the Jiangsu Engineering Research Center of Communication and Network Technology, NJUPT; and the National Engineering Research Center of Communications and Networking (Nanjing University of Posts and Telecommunications) (TXKY17014).

References

[1] C. V. N. Index, "Global mobile data traffic forecast update, 2015–2020 white paper," *link: http://goo. gl/ylTuVx*, 2016.

[2] Y. Cui, J. Song, K. Ren, M. Li, Z. Li, Q. Ren, and Y. Zhang, "Software defined cooperative offloading for mobile cloudlets," *IEEE/ACM Transactions on Networking*, 2017.

[3] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial internet of things architecture: An energy-efficient perspective," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 48–54, 2016.

[4] L. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[5] H. Huang and S. Guo, "Service provisioning update scheme for mobile application users in a cloudlet network," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–6.

[6] C. Xu, K. Wang, P. Li, S. Guo, J. Luo, B. Ye, and M. Guo, "Making big data open in edges: A resource-efficient blockchain-based approach," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[7] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 4, pp. 627–640, 2015.

[8] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5g wireless networks," *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[9] J. Ho, J. Zhang, and M. Jo, "Selective offloading to wifi devices for 5g mobile users," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*. IEEE, 2017, pp. 1047–1054.

[10] Y. Lin, E. Chu, Y. Lai, and T. Huang, "Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds," *IEEE Systems Journal*, vol. 9, no. 2, pp. 393–405, 2015.

[11] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, 2017.

[12] Y. Pan, C. Pan, H. Zhu, Q. Ahmed, M. Chen, and J. Wang, "On consideration of content preference and sharing willingness in d2d assisted offloading," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 4, pp. 978–993, 2017.

[13] M. Aazam, S. Zeadally, and K. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, 2018.

[14] J. Fu, Y. Liu, H. Chao, B. Bhargava, and Z. Zhang, "Secure data storage and searching for industrial iot by integrating fog computing and cloud computing," *IEEE Transactions on Industrial Informatics*, 2018.

[15] S. Mishra, D. Putha, J. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable service allocation using metaheuristic technique in fog server for industrial applications," *IEEE Transactions on Industrial Informatics*, 2018.

[16] K. Kaur, S. Garg, G. Aujla, N. Kumar, J. Rodrigues, and M. Guizani, "Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 44–51, 2018.

[17] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "Tube: Time-dependent pricing for mobile data," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 247–258, 2012.

[18] X. Zhuo, W. Gao, G. Cao, and S. Hua, "An incentive framework for cellular traffic offloading," *IEEE Transactions on Mobile Computing*, vol. 13, no. 3, pp. 541–555, 2014.

[19] K. Habak, M. Ammar, K. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 9–16.

[20] K. Xie, J. Cao, X. Wang, and J. Wen, "Optimal resource allocation for reliable and energy efficient cooperative communications," *IEEE Transactions on wireless communications*, vol. 12, no. 10, pp. 4994–5007, 2013.

[21] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning." in *HotNets*, 2016, pp. 50–56.

[22] W. Zhang, Y. Wen, and D. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 190–194.

[23] H. Deng and I. Hou, "Online scheduling for delayed mobile offloading," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 1867–1875.

[24] B. Baron, P. Spathis, H. Rivano, M. Amorim, Y. Viniotis, and M. Ammar, "Centrally-controlled mass data offloading using vehicular traffic," *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 401–415, 2017.

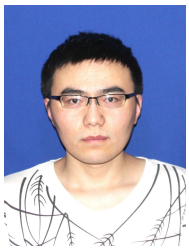
[25] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, 2017.

[26] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric iot," *IEEE Transactions on Emerging Topics in Computing*, 2018.

[27] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 372–382.

[28] Y. He, M. Chen, B. Ge, and M. Guizani, "On wifi offloading in heterogeneous networks: Various incentives and trade-off strategies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2345–2385, 2016.

[29] M. Cheung and J. Huang, "Dawn: Delay-aware wi-fi offloading and network selection," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 6, pp. 1214–1223, 2015.



Yixuan Wang received the B.S. degree in network engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2017. He is currently working toward the M.S. degree in computer network at The University of Aizu, Japan. His research interests include machine learning system, edge network, distributed and parallel computing.

Kun Wang (M'13-SM'17) received the B. Eng. and Ph.D. degree in the School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004 and 2009. From 2013 to 2015, he was a Postdoc Fellow in Electrical Engineering Department, University of California, Los Angeles (UCLA), CA, USA. In 2016, he was a Research Fellow in the School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu City, Fukushima, Japan. He is currently an Associate Professor in the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. He has published over 50 papers in referred international conferences and journals. Dr. Wang received Best Paper Award at IEEE GLOBECOM'2016. He serves as Associate Editor of IEEE Access, Journal of Network and Computer Applications, EAI Transactions on Industrial Networks and Intelligent Systems and Editor of Journal of Internet Technology. He was the symposium chair/co-chair of IEEE IECON16, IEEE EEEIC16, IEEE WCSP16, IEEE CNCC17, etc. His current research interests are mainly in the area of big data, wireless communications and networking, smart grid, energy Internet, and information security technologies. He is a member of IEEE and ACM.

Huawei Huang (M'16) received his Ph.D in Computer Science and Engineering from the University of Aizu, Japan. His research interests mainly include SDN/NFV, mobile computing, and blockchain. He received the best paper award in TrustCom-2016. He was a visiting scholar at the Hong Kong Polytechnic University from 2017 to 2018. He was a post-doctoral research fellow of JSPS between Oct. 2016 and Mar. 2018. He is currently an assistant professor in Academic Center for Computing and Media Studies, Kyoto University, Japan. He is a member of IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2018.2883991, IEEE Transactions on Industrial Informatics

Toshiaki Miyazaki (M'86-SM'12) received the BE and ME degrees in applied electronic engineering from the University of Electro-Communications, Tokyo, Japan in 1981 and 1983, respectively, and the PhD degree in electronic engineering from the Tokyo Institute of Technology in 1994. He is a professor of the University of Aizu, Fukushima, Japan, and the dean of the Undergraduate School of Computer Science and Engineering. His research interests are in reconfigurable hardware systems, adaptive networking technologies, and autonomous systems. Before joining the University of Aizu, he has worked for NTT for 22 years, and engaged in research on VLSI CAD systems, telecommunications-oriented FPGAs and their applications, active networks, peer-to-peer communications, and ubiquitous network environments. He was a visiting professor of the graduate school, Niigata University in 2004, and a part-time lecturer of the Tokyo University of Agriculture and Technology in 2003-2007. He is a senior member of the IEEE, IEICE and IPSJ.

Song Guo (M'02-SM'11) is a Full Professor at Department of Computing, The Hong Kong Polytechnic University. His research interests are mainly in the areas of big data, cloud computing, mobile computing, and distributed systems with over 400 papers published in major conferences and journals. His work was recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. He is the recipient of the 2018 IEEE TCGCC Best Magazine Paper Award, 2017 IEEE Systems Journal Annual Best Paper Award, and other six Best Paper Awards from IEEE/ACM conferences. Prof. Guo was an Associate Editor of IEEE Transactions on Parallel and Distributed Systems and an IEEE ComSoc Distinguished Lecturer. He is now an Associate Editor of IEEE Transactions on Cloud Computing, IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Sustainable Computing, IEEE Transactions on Green Communications and Networking, and IEEE Network. Prof. Guo also served as General and Program Chair for numerous IEEE conferences. He currently serves as a Director and Member of the Board of Governors of IEEE ComSoc.