Learning-based Green Workload Placement for Energy Internet in Smart Cities

Qihua Zhou, *Student Member*, *IEEE*, Yanfei Sun, *Member*, *IEEE*, Haodong Lu, *Student Member*, *IEEE*, and Kun Wang, *Senior Member*, *IEEE*

Abstract—**The Energy Internet is a fundamental infrastructure for deploying green city applications, where energy saving and job acceleration are two critical issues to address. In contrast to existing approaches that focus on static metrics with the assumption of complete prior knowledge of resource information, both application-level properties and energy-level requirements are realized in this paper by jointly considering energy saving and job acceleration during job runtime. Considering the online environment of smart city applications, the main objective is transferred as an optimization problem with a model partition and function assignment. To minimize the energy cost and job completion time together, a green workload placement approach is proposed by using the multi-action deep reinforcement learning method. Evaluations with real-world applications demonstrate the superiority of this method over state-of-the-art methods.**

Index Terms—**Energy saving, workload scheduling, Energy Internet, green city.**

I. Introduction

HIGH-EFFICIENCY energy management has become a crucial issue in the deployment of green cities [1], [2], where numerous large-scale distributed processing applications are handled by mobile devices, including wireless sensors, cell phones, and Internet of Things (IoT) equipment [3], [4]. In this resource-constrained environment, saving energy consumed by workload computations and reducing heat dissipation are two important factors that should be considered [5]. As a fundamental infrastructure to implement energy management systems, the Energy Internet (EI) [6]-[8] is a promising technology for generating and storing power in a "green" manner. Although the development of dedicated hardware (e.g., energy router [9]) makes energy scheduling more flexible, effectively handling computation intensive jobs (e.g., training neural networks) using mobile devices for green cities remains a challenging problem [10].

From the perspective of energy management, many studies have been conducted to minimize the energy cost of green city applications. Reference [11] proposed multi-level queues to tackle data rates and reduce the communication delay for IoT sensor devices. Reference [12] focused on the energy efficiency of photovoltaic storage systems by using machine learning methods and proposed an approximation offline policy to handle energy management, whereas [13] introduced the abstraction of virtual energy generation and designed an adaptive algorithm to improve the energy efficiency of dedicated microgrid networks. In addition, [14] improved the performance of hybrid energy storage systems by optimizing the energy cost and data size together, and [15] proposed a novel energy management approach using an intelligent agent and improved the efficiency of customized microgrid networks. Finally, [16] designed a decentralized energy management framework to optimize the offloading and consumption of the power grid for residential areas, and [17] proposed a priority-based method to reduce resource latency and improve the processing efficiency through the collaboration of edge computing.

However, the approaches mentioned above are based on an optimization of the energy scheduling, requiring the complete knowledge of resource information during job runtime, which is difficult to obtain in advance, particularly under the cutting-edge scenario of distributed model training applications. In addition, these approaches often focus on energy-level metrics, lacking awareness of the application-level properties, such as the job execution deadline and computation resource demand. Consequently, existing approaches cannot reduce the energy cost without degrading the distributed processing speed. Observing the limitations of previous studies, we intend to jointly consider energy saving and job acceleration. This objective can be transferred as a workload placement problem.

Considering the EI scenario for the green city shown in Fig. 1, numerous smart city applications are deployed on the cloud side. Each of these applications requires different numbers of devices, model complexity, data sizes, and resource demand. To efficiently handle these applications, the system

Manuscript received: April 29, 2020; revised: August 10, 2020; accepted: September 14, 2020. Date of CrossCheck: September 14, 2020. Date of online publication: December 10, 2020.

This work was supported by the National Natural Science Foundation of China (No. 61772286), the Jiangsu Key Research and Development Program (No. BE2019742), and the Natural Science Foundation of Jiangsu Province of China (No. BK20191381).

Q. Zhou and Y. Sun (corresponding author) are with the School of Automation and School of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing, China (e-mail: kimizqh@foxmail.com; sunyanfei@njupt.edu.cn).

H. Lu is with the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China (e-mail: ihaodonglu@gmail.com).

K. Wang is with the Department of Electrical and Computer Engineering, University of California, Los Angeles, USA (e-mail: wangk@ucla.edu).

DOI: 10.35833/MPCE.2020.000271

needs to properly schedule them among the device cluster, where a variety of mobile devices (e.g., intelligent vehicles, smart phones, personal hotspots, and wearable equipment) collaborate together to finish the jobs. Regarding the heterogeneity of the hardware configuration, network conditions, and application requirements among such devices, it is necessary to carefully design the job scheduling strategy. In addition, the training model of the application is quite large, often exceeding the computation capacity and memory storage of a single device. Therefore, we need to partition the model into a series of pieces, which is called model parallelism [18]. The corresponding neurons and functions are then assigned to different devices such that each device can operate the local calculation through its limited resources. At the end of each iteration, the intermediate results from different devices are aggregated and merged into the global model on the cloud side. All devices then use new input data and continue the next iteration. This iterative procedure continues until the model converges. During the runtime of a job, the device cluster is powered by the EI based on different types of energy sources, including chemical energy storage, the power grid, renewable energy, conventional power stations, and user-end batteries. Assigning the neurons and functions to different devices will generate various energy demands and computational cost. Therefore, the key to efficiently handling the job procedure contains two objectives: ① properly dividing the large training model into small subsets; and ② assigning the corresponding neurons and functions to the suitable devices with minimum energy cost and job completion time.

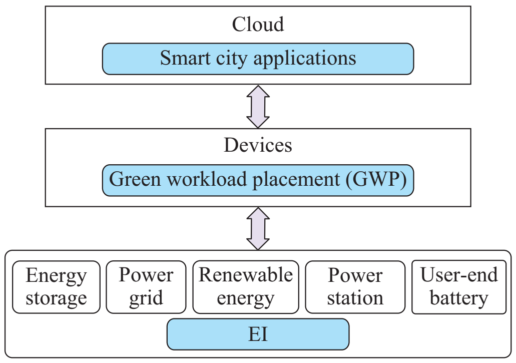

Fig. 1. EI in green cities.

The above objectives can be transferred as a workload placement problem, which is subject to the dynamic runtime properties and non-linear constraints of the available resources. Owing to the difficulty of directly solving this problem through conventional optimization methods, we employ multi-action deep reinforcement learning (DRL) [19]-[21] to overcome this challenge. In fact, introducing DRL-based methods to improve the performance of the EI is not new, and some previous studies have focused on this combination. For example, [22] reviewed existing approaches on controlling EI-based systems using DRL-based optimization. However, in contrast to existing studies, our objective is to propose a green workload approach that jointly reduces the accumulative energy cost and average job completion time to fundamentally improve the efficiency of the EI for green city applications. More precisely, we design a multi-action DRL agent by employing the proximal policy optimization (PPO) [23]-[25] method based on an actor-critic [26], [27] network, aiming at optimizing the two objectives mentioned above at the same time. We discuss how to design the DRL model in Section III-B, including the details of the state, action, reward, and step transition. In addition, we present the DRL training methodology in Section III-C, and point out the search direction to optimize the expected reward related to energy cost and job completion time.

We evaluate our GWP approach by conducting image classification applications with the support of real-world traces. With regard to the distributed job processing performance, our approach can achieve the most stable test accuracy and restrict the training loss into a lower bound under different configurations of CPU cycle frequency, percentile of remaining CPU computation capacity, and available network bandwidth. Moreover, our approach can effectively reduce the per-iteration time and waiting time delayed by slow devices, reducing the entire job completion time. Moreover, our approach can efficiently save energy by reducing the workload computation and heat dissipation. Consequently, the proposed approach requires less accumulative energy cost until job completion over state-of-the-art methods, including random [28], static [29], rule-based [30], and heuristic [31] approaches.

The contributions of our work are summarized as follows:

1) By jointly considering the energy saving and job acceleration requirements for green city applications, we formulate this objective as a workload placement problem and point out that the key to achieving this target is properly dividing the large training model into small subsets, and assigning the corresponding neurons and functions to suitable devices with the minimum energy cost and job completion time.

2) We propose a novel GWP approach by employing multi-action DRL. The DRL agent inside our approach collaborates with the PPO [23]-[25] method based on the actor-critic [26], [27] network, aiming at optimizing the energy saving and job acceleration concurrently.

3) We evaluate the performance of our approach in realistic application scenarios under different parameter configurations. Our approach outperforms other state-of-the-art methods [28]-[31] in three main metrics: distributed training procedure, job processing speed, and energy cost during job runtime.

The remainder of this paper is organized as follows. We present the problem description in Section II and discuss the corresponding DRL-based algorithm design in Section III. We evaluate the performance of our approach in Section IV and provide some concluding remarks in Section V.

II. Problem Description

To conduct the training process in the resource-constrained environment of a green city, a large-scale DL training job is often deployed in a distributed manner, where the entire neural network model is partitioned into pieces and each device handles a certain proportion. As shown in Fig. 2, we intend to divide the model according to the order of layers and allocate the neurons to different devices. In general, a neuron corresponds to a specific function in the computation graph marked in different colors. Assigning a neuron to different devices will yield different job completion time and energy costs. In the EI scenario, the final energy cost is strongly related to the time consumption of the workload computation and network communication. We therefore need to carefully determine how to divide the model among devices to optimize the following two objectives: ① minimizing the job completion time; and ② reducing energy cost. We call the optimization procedure for this target a GWP.

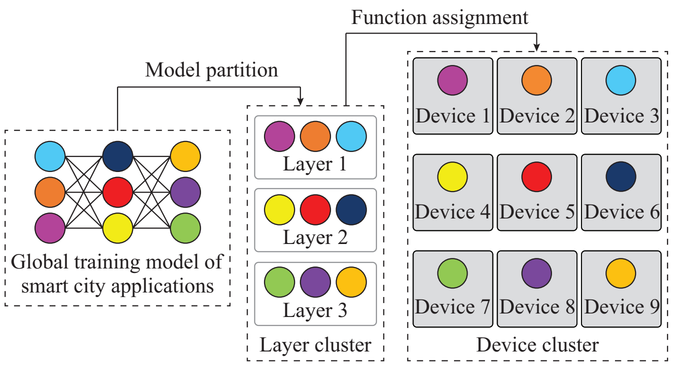

Fig. 2. Model partition and function assignment for large-scale distributed training jobs in smart cities.

A. System Model

1) Preliminary

Considering that a distributed training job contains a set of $\mathbb{M}=\{1,2,\ldots,i,\ldots,M\}$ mobile devices, each device i handles a part of the original neural network model, and these devices need to collaborate in a specific order based on the model partition result to complete the job. Therefore, different devices respond to different computation functions. We use D_i to represent the set of local data, corresponding to the computation function on device i. The entire global data $\mathbb{D}$ is the aggregation of all D_i of these devices. In a microview, we use $d_{i,j}$ to denote a sample belonging to D_i and the processing on $d_{i,j}$ is the basic computation granularity in the training task.

2) Job Completion Time

The time cost of processing each $d_{i,j}$ on device i is strongly related to the current CPU status and application-level characteristics. Given the CPU cycle frequency f_i of device i, the time cost on each cycle is $1/f_i$. Note that the processing of $d_{i,j}$ is constituted by a series of float-point operations [32], which dominate the time and energy cost during the entire training procedure [33]. We therefore mainly focus on float-point operations to provide deep insights into the job completion time. Assuming that each float-point operation takes δ_i CPU cycles and the current percentile of CPU remaining computation capacity is u_i, we can determine the completion time per floating-point operation $\delta_i/(f_iu_i)$. Considering that processing $d_{i,j}$ requires n_j float-point operations, we can calculate the completion time of $d_{i,j}$ as:

$$t_{i,j,comp}=n_{i,j}\frac{\delta_i}{f_iu_i}\quad i\in\mathbb{M} \tag{1}$$

where $n_{i,j}$ is the number of float-point operations to complete the processing of $d_{i,j}$.

Since completing the function of the subtask allocated to device i requires going through the entire local data D_i, we have the following function completion time:

$$t_{i,comp}=\sum_{j=1}^{|D_i|}n_{i,j}\frac{\delta_i}{f_iu_i}\quad i\in\mathbb{M},d_{i,j}\in D_i \tag{2}$$

After the computation stage is completed, we need to transmit the output data from device i to the devices belonging to the next layer. The communication time can be described as:

$$t_{i,comm}=\frac{o_i}{B_i}\quad i\in\mathbb{M} \tag{3}$$

where o_i and B_i are the size of the output data and the current available network bandwidth, respectively. Note that o_i is related to the application-level characteristics of the function conducted on the device, and B_i continues changing during the job execution period, which is difficult to predict in advance. Owing to the network-agnostic pattern, we focus on the optimization of the entire layer, instead of a single function deployed on a device. Combining the two stages of computation and communication, the per-iteration time on device i can be described as:

$$t_i=t_{i,comp}+t_{i,comm}\quad i\in\mathbb{M} \tag{4}$$

Recall that the neural network model is partitioned into a series of layers, and the neurons corresponding to different computational functions are allocated to the devices in the cluster. We use $\mathbb{V}$ to denote all neurons inside the neural network model and divide the model into several layers, each of which is a set of neurons. This relationship can be described as $\mathbb{V}=\{V_1,V_2,\ldots,V_l,\ldots,V_L\}$, where L is the number of partitioned layers. We can see that the job execution logic follows the layer-wise sequence, where only all functions belonging to the previous layer have been completed can the next layer continues the subsequent calculation. In addition, the execution speed of a layer is essentially determined by the most time-consuming function inside this layer. As a result, the layer completion time is bounded by the slowest device handling the corresponding function. Therefore, we can calculate the layer completion time as:

$$t_l=\max_{i\in V_l}(t_i)\quad V_l\in\mathbb{V} \tag{5}$$

By identifying the layer completion time t_l with the current iteration index k, the total job completion time to conduct the training workflow is expressed as:

$$T=\sum_{k=1}^{K}\sum_{l=1}^{L}t_{l,k} \tag{6}$$

where k is the current iteration index; and K is the cumulative number of iterations until the training converges.

3) Energy Cost

Based on an analysis of the job completion time, we can better establish the energy model because the energy cost is directly related to the time consumed on the job. In contrast to previous studies, we consider the energy cost from two aspects: the workload computation $E_{i,comp}$ and the heat dissipation $E_{i,heat}$. Note that $E_{i,comp}$ is the fundamental energy required to finish the job, whereas $E_{i,heat}$ is the wasted energy emitted from the cooling system. Based on the classical energy cost model [34] of portable devices, the energy cost required to finish the processing of D_i can be described as:

$$E_{i,comp}=\eta_i \sum_{j=1}^{|D_i|} n_{i,j}\delta_i (f_i u_i)^2 \tag{7}$$

where η_i is the effective co-efficiency of the processor capacitances on device i for conducting the computation workload. In addition, the energy cost from heat dissipation of the workload computation and network communication on device i can be calculated as:

$$E_{i,heat}=\int_{t_0}^{t_0+t_{i,comp}} g(f_i^2)\mathrm{d}t+\int_{t_0+t_{i,comp}}^{t_0+t_{i,comp}+t_{i,comm}} h(f_i^2)\mathrm{d}t \tag{8}$$

where $g(\cdot)$ and $h(\cdot)$ are the heat dissipation functions related to the CPU cycle frequency in terms of the computation stage and communication stage, respectively. Note that $E_{i,heat}$ is accumulated from the starting time point t_0 in each iteration. Combining these two perspectives of energy cost, we can describe the energy cost on device i in current iteration k as $E_{i,k}=E_{i,k,comp}+E_{i,k,heat}$, and the total energy cost to finish the job is calculated as:

$$E=\sum_{i=1}^{|M|}\sum_{k=1}^{K} E_{i,k} \tag{9}$$

B. Problem Formulation

With the description of the job completion time and energy cost, we can transfer our target into the optimization problem to minimize these two factors. Therefore, we can formulate this problem as:

$$\min(\alpha E+(1-\alpha)T) \tag{10}$$

$$\mathcal{L}_k(\omega)\le\epsilon \tag{11}$$

$$1\le k\le K \tag{12}$$

$$0\le u_i\le 1 \tag{13}$$

$$0\le f_i\le f_i^{\max} \tag{14}$$

$$0\le B_i\le B_i^{\max} \tag{15}$$

$$\begin{cases} D_i\in\mathbb{D} \\ V_l\in\mathbb{V}\end{cases} \tag{16}$$

where $\mathcal{L}_k(\omega)$ is the loss function of the current iteration k; and ϵ is the error threshold until model convergence. Note that we use the non-negative hyper-parameter α $(0\le\alpha\le 1)$ to adjust the priority weights between the energy cost and the job completion time. Users can set different values of α to express the application-level requirements. For example, the formulation will emphasize the importance of energy and neglect the job completion time when $\alpha=1$.

We can observe that solving the above optimization problem is quite difficult because of the dynamic properties of u_i, f_i, and B_i as well as the non-linear constraints of $\mathcal{L}_k(\omega)$. In addition, we can only obtain the status of these four factors belonging to the current iteration k and cannot obtain the future information in advance. Moreover, the partition strategy of $\mathbb{D}$ and $\mathbb{V}$ directly impacts the minimizing objective. Consequently, these challenges motivate us to propose a learning-based approach to solve the problem.

III. Algorithm Design

A. Architecture Overview

We propose leveraging the learning-based approach to solve the problem formulated in Section II-B. Our objective is to dynamically partition the global training model and assign the computation functions to different devices to minimize the energy cost and job completion time. The key to achieving this target is to make the proper decision from the action of the DRL agent, which is based on the observation of the environment status and feedback of the reward. Considering the online environment of large-scale distributed processing in a smart city, we employ the PPO [23]-[25] method based on the actor-critic [26], [27] network to design our GWP algorithm. We demonstrate the architecture of the proposed algorithm in Fig. 3. The DRL agent is deployed on the parameter server at the cloud side, containing the critic and actor model. At the beginning of each iteration, the agent collects the execution status of all mobile devices and takes actions to adjust the strategy of the model partition and function assignment among the cluster. With the real-time feedback of the job completion time and energy cost, the agent optimizes the neural networks (actor and critic) inside the PPO. With the iterative procedure of the action, state, and reward, the PPO agent can finally learn how to make the near-optimal action to ensure a GWP. Since the action of the agent is based only on the current state, this learning workflow follows the Markov decision process (MDP) [35], [36]. Therefore, the proposed algorithm can make a decision based on the current action from the DRL agent.

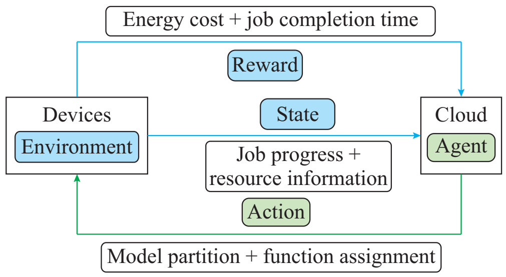
Energy cost + job completion time
Reward
Devices
Environment
State
Job progress + resource information
Cloud
Agent
Action
Model partition + function assignment

Fig. 3. Workflow of DRL-based GWP approach.

B. Model Runtime Design

Four crucial factors are used to design the model runtime environment of the PPO agent based on the actor-critic network.

1) State Space

The state space $\mathcal{S}$ in the GWP contains five factors: ① the current training progress represented by the current iteration index k; ② the current loss function denoted by $\mathcal{L}_k(\omega)$;

③ the percentile of the remaining CPU computation capacity u_i; ④ the current CPU cycle frequency f_i; and ⑤ the current available network bandwidth B_i. Combining the information from all devices, the current state s_k of the entire cluster can be formulated into the vector format as $s_k = \langle k; \mathcal{L}_k(\omega); u_1, u_2, \ldots, u_M; f_1, f_2, \ldots, f_M; B_1, B_2, \ldots, B_M \rangle$.

2) Action Space

The action space $\mathcal{A}$ reflects the strategy of the model partition and function (neuron) assignment to different devices. Therefore, the current action a_k can be formulated as $a_k = \langle V_1, V_2, \ldots, V_L; D_1, D_2, \ldots, D_M \rangle$. The action contains two objectives: ① dividing the neural network model into several subsets of neurons, i. e., forming the model partition collection of $\mathbb{V}$; and ② assigning the neurons inside V_l ($V_l \in \mathbb{V}$) and the corresponding local data D_i ($D_i \in \mathbb{D}$) of each neuron to different devices, i. e., forming the function assignment collection of $\mathbb{D}$. Thus, this proposal belongs to the multi-action DRL [37], [38] category.

3) Reward

The agent receives the feedback of the reward and measures the influence of the action. A good action will lead to a higher reward and vice versa. At the end of iteration k, the agent collects the information on the completion time and energy cost from all devices so as to calculate the current reward based on the problem formulation in (10). Therefore, the reward r_k in the current iteration k can be defined as:

$$r_k = -\alpha E - (1-\alpha) T \tag{17}$$

4) Step and State Transition

Since the DRL learning procedure contains a series of steps, we mark each iteration of the distributed training job as a step. At the end of the current iteration, we can re-conduct the workload placement among the devices based on the action of the agent and continue the training procedure for the next iteration. After that, the workflow of DRL learning moves onto the next step. Consequently, the DRL learning step follows the same index of the training job iteration k.

C. Agent Learning Methodology

With the four fundamental properties for building the model runtime environment, we can present the learning methodology of the PPO agent based on the surrogate objective [23]. As the rationale of our methodology, for the policy $\pi(a_k|s_k;\boldsymbol{\theta}_k)$ after updating, it will be more advisable to maximize the long-term reward over the previous policy. We can formulate the expected gain $E_{\pi(\boldsymbol{\theta})}[\mathcal{G}]$ of the policy $\pi(\boldsymbol{\theta})$ as:

$$E_{\pi(\boldsymbol{\theta})}[\mathcal{G}] = E_{\pi(\boldsymbol{\theta}_k)}[\mathcal{G}] + E_{\pi(\boldsymbol{\theta})}\left[\sum_{t=0}^{+\infty} \gamma_t a_{\pi(\boldsymbol{\theta})}(\mathcal{S}_t, \mathcal{A}_t)\right] \tag{18}$$

where γ_t is the discount factor of the step t; and $a_{\pi(\boldsymbol{\theta})}$ is the action according to the polity $\pi(\boldsymbol{\theta})$, under the restriction of the state space $\mathcal{S}_t$ and action space $\mathcal{A}_t$.

Our target is to maximize the expected return $E_{\pi(\boldsymbol{\theta})}[\mathcal{G}]$, which can be achieved by maximizing $E_{\pi(\boldsymbol{\theta})}\left[\sum_{t=0}^{+\infty} \gamma_t a_{\pi(\boldsymbol{\theta})}(\mathcal{S}_t, \mathcal{A}_t)\right]$. Here, we can replace the expectation of $\mathcal{A}_t \sim \pi(\boldsymbol{\theta})$ based on the expectation of $\mathcal{A}_t \sim \pi(\boldsymbol{\theta}_k)$. This transformation can be described as:

$$E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}), \mathcal{A}_t \sim \pi(\boldsymbol{\theta})}\left[a_{\pi_k}(\mathcal{S}_t, \mathcal{A}_t)\right] = E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}), \mathcal{A}_t \sim \pi(\boldsymbol{\theta}_k)}\left[\frac{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta})}{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta}_k)} a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t)\right] \tag{19}$$

where a_{π_k} represents the action based on the updated policy of step k.

Owing to the difficulty of calculating the expectation of $\mathcal{S}_t \sim \pi(\boldsymbol{\theta})$ directly, we can use the approximate expectation of $\mathcal{S}_t \sim \pi(\boldsymbol{\theta}_k)$ and rewrite (19) as:

$$E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}), \mathcal{A}_t \sim \pi(\boldsymbol{\theta})}\left[a_{\pi_k}(\mathcal{S}_t, \mathcal{A}_t)\right] \approx E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}_k), \mathcal{A}_t \sim \pi(\boldsymbol{\theta}_k)}\left[\frac{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta})}{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta}_k)} a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t)\right] \tag{20}$$

Moreover, we can obtain the approximate expression $\hat{\mathcal{G}}(\boldsymbol{\theta})$ of $E_{\pi(\boldsymbol{\theta})}[\mathcal{G}]$ as:

$$\hat{\mathcal{G}}(\boldsymbol{\theta}) = E_{\pi(\boldsymbol{\theta}_k)}[\mathcal{G}] + E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}_k), \mathcal{A}_t \sim \pi(\boldsymbol{\theta}_k)}\left[\sum_{t=0}^{+\infty} \gamma_t \frac{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta})}{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta}_k)} a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t)\right] \tag{21}$$

where $\hat{\mathcal{G}}(\boldsymbol{\theta})$ and $E_{\pi(\boldsymbol{\theta})}[\mathcal{G}]$ have the same value of $E_{\pi(\boldsymbol{\theta}_k)}[\mathcal{G}]$ and gradient at the point of $\boldsymbol{\theta} = \boldsymbol{\theta}_k$. We can therefore increase the expected gain and update the policy parameters by following the gradient direction. The optimization rule can be formulated as:

$$E_{\mathcal{S}_t \sim \pi(\boldsymbol{\theta}_k), \mathcal{A}_t \sim \pi(\boldsymbol{\theta}_k)}\left[\sum_{t=0}^{+\infty} \gamma_t \frac{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta})}{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta}_k)} a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t)\right] \tag{22}$$

Combining the updating property of the PPO agent, we can finally rewrite the optimization objective as:

$$E_{\pi(\boldsymbol{\theta}_k)}\left[\min\left(\frac{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta})}{\pi(\mathcal{A}_t|\mathcal{S}_t;\boldsymbol{\theta}_k)} a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t) + \xi\right)\right] \tag{23}$$

where $\xi = \varepsilon \left| a_{\pi(\boldsymbol{\theta}_k)}(\mathcal{S}_t, \mathcal{A}_t) \right|$, and the hyper-parameter ε satisfies $\varepsilon \in (0, 1)$. Using this equation, we can optimize our objective in (17) from a long-term learning perspective, while avoiding large fluctuations between the two steps.

IV. Performance Evaluation

We highlight the performance of our GWP from three aspects.

1) How effective is the distributed training procedure? Our approach can achieve the most stable test accuracy and restrict the training loss into a lower bound under different configurations of the CPU cycle frequency, percentile of remaining CPU computation capacity, and available network bandwidth.

2) How efficient is the acceleration of the job progressing speed? Our approach can effectively reduce the per-iteration and waiting time delay of slow devices to reduce the entire job completion time over the four baseline approaches.

3) How efficient is the energy cost savings? Our approach can efficiently save energy over conventional methods by re-

ducing the consumption of the workload computation and heat dissipation, requiring less accumulative energy cost until the job completion.

A. Experiment Setting

1) Network Construction

To construct the environment of a smart city and EI, we use the communication traces of 5G wireless networks from state-of-the-art applications [39], [40].

2) Testbed

Deploying large-scale distributed processing tasks to handle green city applications requires a careful configuration of the underlying hardware. Therefore, we build a testbed using Alibaba cloud with eight 16-core CPUs, 128 GB of memory, four Nvidia Tesla P100 GPUs with 64 GB of graphic memory in total, and Infiniband connection support. We use these computation resources to establish a virtual cluster to simulate the execution traces of smart city applications.

3) Workload and Benchmark

Considering the property of mobile scenarios, we inspect the application-level properties by conducting image classification tasks on the MobileNet [41] model using the Fashion MNIST [42] and CIFAR-100 [43] datasets.

4) Comparison Baseline

We compare our proposed DRL-based GWP with the following four state-of-the-art baselines.

1) Random [28]: the training model is randomly partitioned and neuron functions are assigned to the devices in the corresponding order.

2) Static [29]: assume that the CPU cycle frequency f, the percentile of remaining CPU computation capacity u and the available network bandwidth B are static and known in advance, and traditional optimization programming approaches can be employed.

3) Rule-based [30]: the model partition and function assignment are based on predefined experience-driven rules.

4) Heuristic [31]: use the information on the computation status and network condition at the beginning of the epoch to formulate the processing procedure and decision on the workload placement.

B. Evaluation Results

Since our objective is to minimize the energy cost and job completion time for large-scale distributed processing applications, we focus on the metrics from three aspects: ① a distributed training procedure; ② job processing speed; and ③ energy cost during job runtime.

1) Inspection of Distributed Training Procedure

We adjust the parameters of f, u, and B in our proposed GWP, inspecting how these parameters affect the performance of the distributed training procedure.

As shown in Fig. 4(a), the training procedure achieves stable convergence and yields the highest test accuracy by up to 90.70%. In addition, as shown in Fig. 4(b), the training loss continues to decrease until convergence. Regardless of the intermediate fluctuation, the final loss is below 0.2 under different parameter configurations. Moreover, we can observe that the training can achieve a better performance with a higher CPU cycle frequency and remaining computation capacity, whereas the available bandwidth yields a slight impact. This phenomenon indicates the significance of optimizing the model partition and function assignment among the cluster, accelerating the convergence speed and reducing the workload overhead.

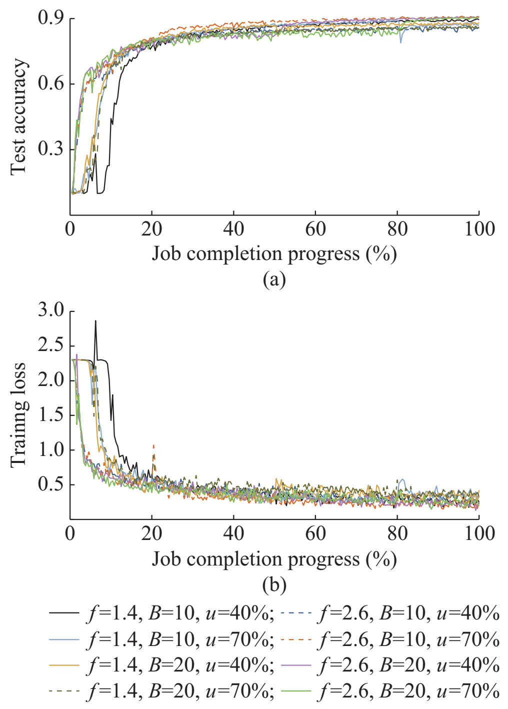

Fig. 4. Performance of distributed training process with different parameter configurations using Fashion MNIST dataset. (a) Test accuracy. (b) Training loss.

2) Inspection of Job Processing Speed

We compare the average job processing performance of the proposed GWP approach with four other baseline approaches in terms of the per-iteration time, waiting time due to slow devices, and the job completion time.

As shown in Fig. 5(a), our approach yields a shorter per-iteration time and follows an approximately linear trend with an increase in the number of functions, whereas the other four baseline approaches generate a much longer per-iteration time and the curves increase sharply. In addition, we accumulate the time wasted by slow workers in each layer and use the waiting time to show the system efficiency of parallel computation. As shown in Fig. 5(b), the DRL agent can properly divide the training model and assign each function to the most suitable device for execution. Therefore, our approach can effectively reduce the accumulative waiting time delay of slow devices, whereas the other four baseline approaches still face a much longer waiting time. Moreover, we compare the average job completion time of our approach with these baseline approaches by conducting image classification tasks on the MobileNet-V2 model using the Fashion MNIST and CIFAR-100 datasets. As shown in Fig. 5(c), our approach generates much fewer training epochs until the training converges, consuming less job completion time, and reducing the variance in computational cost. For this metric, our approach is up to 1.48-, 2.07-, 3.45-, and 4.67-times faster on average than the heuristic, rule-based, static, and random placement approaches, respectively.

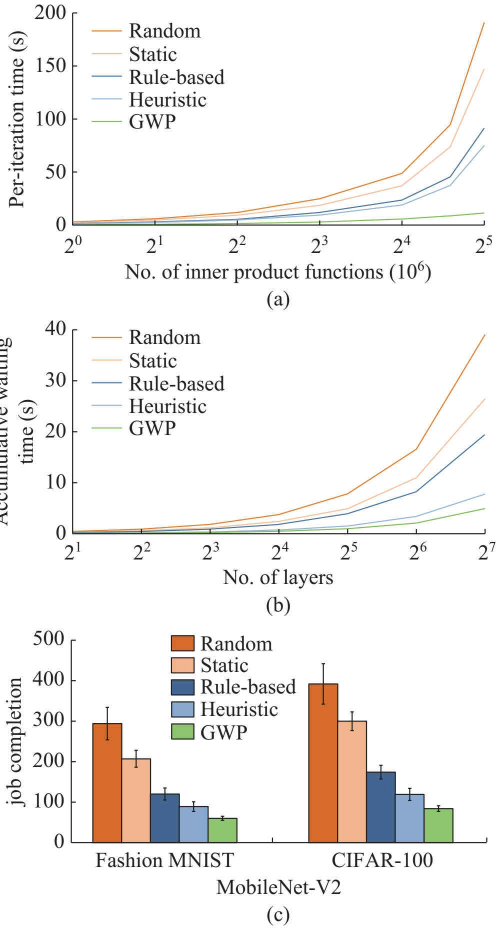

Fig. 5. Average performance of job processing. (a) Ratio of per-iteration time to number of fuctions and neurons. (b) Ratio of waiting time to layer numbers delayed by slow devices. (c) Epoch numbers until job completion on MobileNet-V2 model with Fashion MNIST and CIFAR-100 datasets.

3) *Inspection of Energy Cost*

To inspect the energy saving efficiency, we also compare the energy cost performance of our approach with the four baselines in terms of the per-iteration energy cost of the workload computation, the per-iteration energy cost of heat dissipation, and the cumulative energy cost until job completion.

The energy cost of the workload computation during each iteration depends strongly on the computational cost of the float-point operations. We inspect how the workload size impacts this metric. As shown in Fig. 6(a), all five approaches consume more energy with an increase in the number of workload data. However, the curve of our approach grows much more slowly and consumes less energy under different workload configurations. In addition, as the energy of heat dissipation is wasted during the computation and communication stages, we also inspect the energy cost in this aspect in Fig. 6(b). We can observe that our approach can effectively reduce the heat dissipation by accurately assigning functions to the most energy-saving devices, whereas the other four baseline approaches generate a much higher energy cost of heat dissipation. Moreover, we record the accumulative energy cost until job completion and compare the performance of all five approaches in Fig. 6(c). Our approach can achieve better energy saving efficiency and reduce the energy consumption variance over the other four baselines, in the training of both the Fashion MNIST and CIFAR-100 datasets when applying the MobileNet-V2 model. More precisely, the proposed approach can save energy costs by up to 40.63%, 61.97%, 65.46%, and 68.92% on average over the heuristic, rule-based, static, and random placement approaches, respectively.

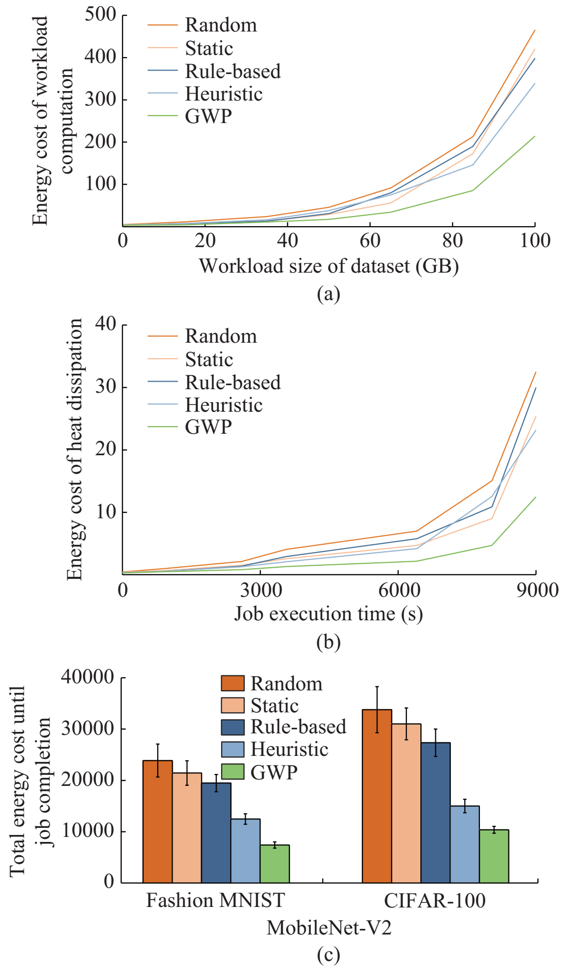

Fig. 6. Average performance of energy cost. (a) Energy cost of workload computation in each iteration. (b) Energy cost of heat dissipation in each iteration. (c) Accumulative energy cost until job completion.

C. *Summary*

Overall, our proposed approach outperforms conventional approaches in terms of the training processing speed, convergence efficiency, job completion time, and accumulative energy cost. As the key to this, the DRL-based GWP approach can properly divide the large model into pieces and assign each function or neuron to the most suitable device, which aims to finish the corresponding workload computation by jointly minimizing the job completion time and energy cost through experience-based learning.

V. Conclusion

This paper focuses on the workload placement problem in the EI for green cities. Observing the limitations of existing approaches that aim to minimize the energy cost while neglecting application-level properties, this study jointly considers the optimization objectives of both energy saving and job acceleration. Owing to the online environment of smart city applications, the main target is formulated as an optimization problem with model partition and function assignment. To minimize the time and energy-level factors, a GWP approach is designed using the multi-action DRL method. The evaluation based on real-world scenarios demonstrates the advantages of the proposed approach over state-of-the-art methods in terms of the distributed training procedure, job processing speed, and energy cost during job runtime.

References

[1] C. Zhu, H. Zhou, V. C. M. Leung *et al*., "Toward big data in green city," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 14-18, Nov. 2017.

[2] M. Gao, K. Wang, and L. He, "Probabilistic model checking and scheduling implementation of an energy router system in energy Internet for green cities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 4, pp. 1501-1510, Apr. 2018.

[3] P. Jia, X. Wang, and K. Zheng, "Distributed clock synchronization based on intelligent clustering in local area industrial IoT systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3697-3707, Jun. 2020.

[4] B. V. Philip, T. Alpcan, J. Jin *et al*., "Distributed real-time IoT for autonomous vehicles," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1131-1140, Feb. 2019.

[5] H. Zhang, Y. Li, D. W. Gao *et al*., "Distributed optimal energy management for Energy Internet," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3081-3097, Dec. 2017.

[6] Z. Lv, W. Kong, X. Zhang *et al*., "Intelligent security planning for regional distributed energy Internet," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3540-3547, May 2020.

[7] K. Wang, J. Yu, Y. Yu *et al*., "A survey on energy Internet: architecture, approach, and emerging technologies," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2403-2416, Sept. 2018.

[8] C. Tu, X. He, X. Liu *et al*., "Resilient and fast state estimation for energy Internet: a data-based approach," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2969-2979, May 2019.

[9] Q. Sun, Y. Zhang, H. He *et al*., "A novel energy function-based stability evaluation and nonlinear control approach for Energy Internet," *IEEE Transactions on Smart Grid*, vol. 8, no. 3, pp. 1195-1210, May 2017.

[10] J. Qi, L. Liu, Z. Shen *et al*., "Low-carbon community adaptive energy management optimization toward smart services," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3587-3596, May 2020.

[11] N. Ashraf, A. Hasan, H. K. Qureshi *et al*., "Combined data rate and energy management in harvesting enabled tactile IoT sensing devices," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3006-3015, May 2019.

[12] C. Keerthisinghe, A. C. Chapman, and G. Verbic, "Energy management of PV-storage systems: policy approximations using machine learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 257-265, Jan. 2019.

[13] R. R. Deshmukh, M. S. Ballal, H. M. Suryawanshi *et al*., "An adaptive approach for effective power management in DC microgrid based on virtual generation in distributed energy sources," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 362-372, Jan. 2020.

[14] X. Lu and H. Wang, "Optimal sizing and energy management for cost-effective PEV hybrid energy storage systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 5, pp. 3407-3416, May 2020.

[15] M. Manbachi and M. Ordonez, "Intelligent agent-based energy management system for islanded AC-DC microgrids," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4603-4614, Jul. 2020.

[16] M. S. H. Nizami, J. Hossain, and E. Fernandez, "Multiagent-based transactive energy management systems for residential buildings with distributed energy resources," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1836-1847, Mar. 2020.

[17] L. Ruan, Y. Yan, S. Guo *et al*., "Priority-based residential energy management with collaborative edge and cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1848-1857, Mar. 2020.

[18] Y. Huang, Y. Cheng, A. Bapna *et al*., "GPipe: efficient training of giant neural networks using pipeline parallelism," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019, pp. 103-112.

[19] R. S. Sutton and A. G. Barto, "Adaptive computation and machine learning," in *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge: MIT Press, 2017.

[20] Y. Duan, X. Chen, R. Houthooft *et al*., "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33nd International Conference on Machine Learning*, New York, USA, Jun. 2016, pp. 1329-1338.

[21] M. Hessel, J. Modayil, H. van Hasselt *et al*., "Rainbow: combining improvements in deep reinforcement learning," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, New Orleans, USA, Feb. 2018, pp. 3215-3222.

[22] M. Glavic, "Deep reinforcement learning for electric power system control and related problems: a short review and perspectives," *Annual Reviews in Control*, vol. 48, pp. 22-35, Dec. 2019.

[23] J. Schulman, F. Wolski, P. Dhariwal *et al*. (2017, Aug.). Proximal policy optimization algorithms. [Online]. Available: https://arxiv. org/abs/1707.06347

[24] Y. Wang, H. He, X. Tan *et al*., "Trust region-guided proximal policy optimization," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019, pp. 624-634.

[25] B. Liu, Q. Cai, Z. Yang *et al*., "Neural trust region/proximal policy optimization attains globally optimal policy," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019, pp. 10564-10575.

[26] L. A. Prashanth and M. Ghavamzadeh, "Actor-critic algorithms for risk-sensitive MDPs," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Lake Tahoe, USA, Dec. 2013, pp. 252-260.

[27] P. Thomas, "Bias in natural actor-critic algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, Beijing, China, Jun. 2014, pp. 441-448.

[28] B. S. K. Patnam and N. M. Pindoriya, "Centralized stochastic energy management framework of an aggregator in active distribution network," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1350-1360, Mar. 2019.

[29] N. H. Tran, W. Bao, A. Y. Zomaya *et al*., "Federated learning over wireless networks: optimization model design and analysis," in *Proceedings of IEEE International Conference on Computer Communications*, Paris, France, Apr.-May 2019, pp. 1387-1395.

[30] P. Yi, T. Zhu, B. Jiang *et al*., "Deploying energy routers in an energy Internet based on electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 4714-4725, Jun. 2016.

[31] S. Wang, T. Tuor, T. Salonidis *et al*., "When edge meets learning: adaptive control for resource-constrained distributed machine learning," in *Proceedings of IEEE International Conference on Computer Communications*, Honolulu, USA, Apr. 2018, pp. 63-71.

[32] Y. Yu, C. Wu, T. Zhao *et al*., "OPU: an FPGA-based overlay processor for convolutional neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 35-47, Jan. 2020.

[33] C. Wu, M. Wang, X. Chu *et al*., "Low precision floating point arithmetic for high performance FPGA-based CNN acceleration," in *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Seaside, USA, Feb. 2020, pp. 1-8.

[34] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, vol. 13, no. 2-3, pp. 203-221, Nov. 1996.

[35] S. H. Lim and A. Autef, "Kernel-based reinforcement learning in robust Markov decision processes," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, Jun. 2019, pp. 3973-3981.

[36] E. Lecarpentier and E. Rachelson, "Non-stationary Markov decision processes, a worst-case approach using model-based reinforcement learning," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019, pp. 7214-7223.

[37] S. Iqbal and F. Sha, "Actor-attention-critic for multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, Jun. 2019, pp. 2961-2970.

[38] Y. Du, L. Han, M. Fang *et al*., "LIIR: learning individual intrinsic reward in multi-agent reinforcement learning," in *Proceedings of Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, Dec. 2019, pp. 1-9.

[39] C. Wang, M. D. Renzo, S. Stanczak *et al*., "Artificial intelligence enabled wireless networking for 5G and beyond: recent advances and future challenges," *IEEE Wireless Communication*, vol. 27, no. 1, pp. 16-23, Feb. 2020.

[40] D. M. G. Estévez, M. Gramaglia, A. D. Domenico *et al*., "Artificial intelligence for elastic management and orchestration of 5G networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 134-141, Oct. 2019.

[41] A. G. Howard, M. Zhu, B. Chen *et al*. (2017, Apr.). Mobilenets: efficient convolutional neural networks for mobile vision applications. [Online]. Available: http://www.arxiv.org/abs/1704.04861

[42] H. Xiao, K. Rasul, and R. Vollgraf. (2017, Aug.). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. [Online]. Available: http://www.arxiv.org/abs/1708.07747

[43] A. Krizhevsky. (2009, Jun.). Learning multiple layers of features from tiny images. [Online]. Available: http://citeseerx. ist. psu. edu/viewdoc/summary?doi=10.1.1.222.9220

Qihua Zhou received the B.S. degree in information security from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2015. His research interests include operating systems, distributed processing, parallel computing, deep learning and graph computing.

Yanfei Sun is a Full Professor with the School of Automation and School of Artificial Intelligence, Nanjing University of Posts and Telecommunications, Nanjing, China. He is also a Director of the Jiangsu Engineering Research Center of High-performance Computing and Intelligent Processing, Nanjing, China. His current research interests include future networks, industrial Internet, Energy Internet, big data management and analysis, and intelligent optimization and control.

Haodong Lu is a M.Sc. student in the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, China. His current research interests include distributed processing, parallel computing and deep learning.

Kun Wang received two Ph.D. degrees in computer science from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2009, and from the University of Aizu, Aizuwakamatsu, Japan, in 2018. He was a Post-doctoral Fellow in University of California, Los Angeles, USA, from 2013 to 2015, where he is a Senior Research Professor. He was a Research Fellow at Hong Kong Polytechnic University, Hong Kong, China, from 2017 to 2018, and a Professor at Nanjing University of Posts and Telecommunications. He is the recipient of the ACM SenSys 2019 Best Paper Award, IEEE ISJ Best Paper Award 2019, IEEE GLOBECOM 2016 Best Paper Award, IEEE TCGCC Best Magazine Paper Award 2018, IEEE TCBD Best Conference Paper Award 2019, and CBD 2019 Best Student Paper Award. He serves/served as an Associate Editor of IEEE Access, an Editor of the Journal of Network and Computer Applications, and a Guest Editor of IEEE Network, Future Generation Computer Systems, Peer-to-Peer Networking and Applications, IEICE Transactions on Communications, Journal of Internet Technology, and Future Internet. His current research interests include big data, wireless communications and networking, energy Internet, and information security technologies.