Edge-Enabled Two-Stage Scheduling Based on Deep Reinforcement Learning for Internet of Everything

Xiaokang Zhou, *Member, IEEE*, Wei Liang, *Member, IEEE*, Ke Yan, *Member, IEEE*, Weimin Li, *Member, IEEE*, Kevin I-Kai Wang, *Member, IEEE*, Jianhua Ma, *Senior Member, IEEE*, and Qun Jin, *Senior Member, IEEE*

***Abstract*—Nowadays, the concept of Internet of Everything (IoE) is becoming a hotly discussed topic, which is playing an increasingly indispensable role in modern intelligent applications. These applications are known for their real-time requirements under limited network and computing resources, thus it becomes a highly demanding task to transform and compute tremendous amount of raw data in a cloud center. The edge–cloud computing infrastructure allows a large amount of data to be processed on nearby edge nodes and then only the extracted and encrypted key features are transmitted to the data center. This offers the potential to achieve an end–edge–cloud-based big data intelligence for IoE in a typical two-stage data processing scheme, while satisfying a data security constraint. In this study, a deep-reinforcement-learning-enhanced two-stage scheduling (DRL-TSS) model is proposed to address the NP-hard problem in terms of operation complexity in end–edge–cloud Internet of Things systems, which is able to allocate computing resources within an edge-enabled infrastructure to ensure computing task to be completed with minimum cost. A presorting scheme based on Johnson's rule is developed and applied to preprocess the two-stage tasks on multiple executors, and a DRL mechanism is developed to minimize the overall makespan based on a newly designed instant reward that takes into account the maximal utilization of each executor in edge-enabled two-stage scheduling. The performance of our method is evaluated and compared with three existing scheduling techniques, and experimental results demonstrate the ability of our proposed algorithm in achieving better learning efficiency and scheduling performance with a 1.1-approximation to the targeted optimal IoE applications.**

***Index Terms*—Deep reinforcement learning, edge computing, Internet of Everything (IoE), makespan, two-stage scheduling.**

Manuscript received 24 February 2022; revised 14 April 2022; accepted 13 May 2022. Date of publication 30 May 2022; date of current version 6 February 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB1705200; in part by the National Natural Science Foundation of China under Grant 62072171 and Grant 72091515; in part by the Key Research and Development Program of Hunan Province of China under Grant 2020SK2089; and in part by the Open Fund of Key Laboratory of Hunan Province under Grant 2017TP1026. *(Corresponding authors: Wei Liang; Weimin Li.)*

Xiaokang Zhou is with the Faculty of Data Science, Shiga University, Hikone 522-8522, Japan, and also with the RIKEN Center for Advanced Intelligence Project, RIKEN, Tokyo 103-0027, Japan (e-mail: zhou@biwako.shiga-u.ac.jp).

Wei Liang is with the Base of International Science and Technology Innovation and Cooperation on Big Data Technology and Management, Hunan University of Technology and Business, Changsha 410205, China (e-mail: weiliang@csu.edu.cn).

Ke Yan is with the College of Design and Engineering, National University of Singapore, Singapore 117566 (e-mail: yanke@nus.edu.sg).

Weimin Li is with the School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China (e-mail: wmli@shu.edu.cn).

Kevin I-Kai Wang is with the Department of Electrical, Computer, and Software Engineering, The University of Auckland, Auckland 1010, New Zealand (e-mail: kevin.wang@auckland.ac.nz).

Jianhua Ma is with the Faculty of Computer and Information Sciences, Hosei University, Tokyo 102-8160, Japan (e-mail: jianhua@hosei.ac.jp).

Qun Jin is with the Faculty of Human Sciences, Waseda University, Tokorozawa 359-1192, Japan (e-mail: jin@waseda.jp).

Digital Object Identifier 10.1109/JIOT.2022.3179231

I. Introduction

THE ADVANCEMENT of communication technologies has led to the widespread of edge computing and the Internet of Things (IoT)-enabled devices. These interconnected devices are capable of real-time data collection, processing, and communication with an edge server, and form the foundation of modern intelligent services [1], [2]. While these smart devices are providing unprecedented benefits to our daily lives, the amount of collected data and communication requirements are also growing dramatically. The growing quantity of devices, data, and security requirements is all relying on stable and efficient computation and communication to ensure the timely transfer of collected data in a secure manner [3], [4].

The growing penetration of IoT technologies is reflected in several different sectors, such as consumer electronics, healthcare, and industrial automation [5], [6], forming the so-called Internet of Everything (IoE). Extending from ordinary IoT applications, applications relying on cloud–edge infrastructure are facing more challenges in terms of efficient resource allocation to ensure reliable and optimal task completion time across the entire distributed system [7]–[9]. It is of critical importance for modern industrial systems to support multiple heterogeneous applications (e.g., multiple production lines), and make efficient use of the available edge servers to complete all tasks in time. How to arrange these tasks with sequential operations on multiple executors distributed on edge servers becomes the key problem investigated in edge computing environments with smart IoT devices.

Different from other existing methods, we focus on a novel methodology to pursue the optimal two-stage scheduling to ensure a more efficient use of the available computing and communication resources in a typical multiflowline production system implemented through the concept of IoE. The

diversity of multiple production lines and their corresponding application requirements make scheduling an NP-hard problem. In this article, a reinforcement-learning-based heuristic scheduling method, named deep-reinforcement-learning-enhanced two-stage scheduling (DRL-TSS), is proposed to support a more efficient data-intensive task allocation and execution in an end–edge–cloud infrastructure. An integrated deep learning framework, which consists of two basic modules as Johnson's rule-based presorter and DRL-based scheduler, is designed to cope with the makespan minimization problem with multiple executors. Two algorithms are then developed to realize Johnson's rule-based task presorting and reinforcement-learning-based two-stage scheduling, respectively. The major contributions are summarized as follows.

1) A unique two-stage scheduling problem is addressed and modeled to support the operation complexity in the end–edge–cloud IoT system, aiming to improve the efficiency of computing resources shared by devices from multiple heterogeneous applications.
2) Johnson's rule-based presorting scheme is designed and applied to preprocess the two-stage tasks on multiple executors, which can effectively avoid the worst situation in the targeted NP-hard scheduling problem and enhance the overall efficiency in the proposed reinforcement-learning-based scheduling.
3) A DRL mechanism is developed aiming to minimize the overall makespan in edge-enabled two-stage scheduling, in which the action value function is improved based on a newly designed instant reward that considers the maximal utilization of each executor for data-intensive tasks in IoE applications.

The remainder of this article is organized as follows. Section II compares the state-of-the-art techniques in reinforcement learning, followed by the scheduling approaches targeting IoE applications. Section III addresses the application scenario and problem formulation. Section IV discusses the proposed reinforcement-learning-based scheduling method with detailed mechanisms. We introduce the experimental design and evaluate the performance of the proposed method against existing scheduling control approaches in Section V. Section VI concludes this study and gives promising perspectives regarding future research.

II. Related Work

Several topics relating to this study, including reinforcement learning models in IoT systems and task scheduling algorithms for IoT applications, are studied and analyzed, respectively, in this section.

A. Reinforcement Learning in IoT Systems

Recently, the development of reinforcement learning has become one of the most attractive directions of machine learning and AI techniques in modern large-scale and complex network applications, such as heterogeneous networks, Internet of Vehicles (IoV), and IoT [10]–[12]. Jiang *et al.* [13] constructed a reinforcement-learning-based framework to optimize the number of served IoT devices for resource configuration in NarrowBand IoT networks. They designed an action aggregation method based on the deep Q-network to improve the convergence capability in multiparameter and multigroup scenarios using the multiagent learning strategy. Wang *et al.* [14] proposed a so-called mobile-IoT-based multimodal reinforcement learning service framework, in which the action-aware transition tensor was utilized for heterogeneous data fusion, and a Markov decision model was applied to enhance the multimodal reinforcement learning process with the optimal tensor policy. Considering the multihop ad hoc networks with IoT devices, Kwon *et al.* [15] built an autonomous network in which each IoT device was viewed as a decision-making agent based on the Markov decision process. They maximized the estimated cumulative future reward in a deep neural network, to improve the learning process with minimal transmission power consumption. Nassar and Yilmaz [16] considered the Markov decision process with the reinforcement learning model together to solve the adaptive resource allocation problem. They investigated and compared the performances of four reinforcement learning schemes in optimizing the fine-grained decision-making policies for IoT applications in fog radio access networks. Camelo *et al.* [17] focused on the parallel reinforcement learning and developed a partitioning algorithm to optimize communications for reinforcement-learning-based IoT applications in distributed environments. They employed a local affinity policy to improve the reinforcement learning algorithm with a dynamic partitioning scheme in a heuristic co-allocation process. Xiong *et al.* [18] formulated the resource allocation problem as a Markov decision process in IoT edge computing systems. They employed DRL to improve the resource allocation policy, in which the Q-network was redesigned based on the multiple replay memories to improve the training process. Ivoghlian *et al.* [19] introduced a deep Q-network-based multiagent framework for automatic network management targeting typical LoRaWAN-based IoT networks.

B. Task Scheduling for IoT Applications

In current years, with the prevalence of mobile and edge computing, the scheduling algorithm has become an important technique for task management and resource allocation in IoT-assisted applications. Leithon *et al.* [20] designed a framework to optimize the task scheduling within the off-grid IoT nodes. They proposed a mixed linear programming method-based online scheduling strategy with a sorting-based mechanism, which could result in a lower computational complexity. Lee and Lee [21] developed a hybrid algorithm to deal with the centralized resource and task scheduling issues, which aimed to minimize the average on-grid energy consumption, and thus satisfied the minimum average data rate requirement on each IoT device based on the distributed task scheduling. Qi *et al.* [22] focused on IoV, as one specific application of IoT in autonomous driving and applied DRL in parallel with multitask scheduling. They proposed a model-free scheduling method to improve the multitask learning problem by assigning parallel tasks with different computing resources in a

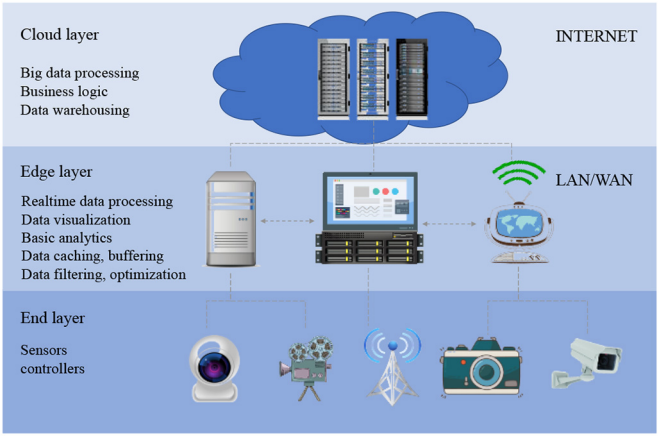


Fig. 1. End–edge–cloud infrastructure of an industrial workplace safety surveillance system.

reinforcement learning architecture. Several studies [23]–[27] developed algorithms to tackle energy efficiency problem for IoT applications. Specifically, He *et al.* [25] formulated the offload task scheduling problem as a constrained Markov decision process. They developed a deep Q-learning-enhanced algorithm to maximize the long-term average reward, in order to tackle the cost-constrained task scheduling problem. Shan *et al.* [26] presented a two-step scheduling method to reduce the energy consumption when offloading the task data in transparent computing-empowered IoT devices. Different from these existing approaches with two-stage scheduling and the Markov feature, our work focuses more on the concurrent computing makespan for IoT tasks distributed on multiple executors.

III. Two-Stage Scheduling in End–Edge–Cloud IoT Systems

In this section, to explain the proposed DRL-TSS model, we first describe the application scenario of a typical end–edge–cloud IoT system and then introduce an overview of the reinforcement learning scheme incorporated with Johnson's rule.

A. Application Scenario

The explosive growth of IoT brings great challenges for many industrial IoT deployments, ranging from latency, network bandwidth, to reliability and security. Edge computing, which is a distributed information technology architecture, is playing a more and more important role in addressing those challenges. The client data is stored at and processed by devices at the edge of networks rather than the central cloud data center for lower latency and better responsiveness. The processed results are then transferred to the cloud center as needed.

In a typical industrial IoT system, such as a workplace safety surveillance system, edge computing can combine and analyze data from on-site cameras, employee safety devices, and various other end devices to help businesses oversee workplace conditions or ensure that employees follow established safety protocols. As illustrated in Fig. 1, the workplace safety surveillance system is expressed as a three-layer end–edge–cloud system. The end layer is composed of various sensors and end devices (e.g., cameras). The video streams captured by cameras are generated continuously and then transmitted to the edge layer through LAN/WAN. The edge layer is made up of several edge nodes, which are responsible for real-time data processing, data caching, filtering, basic analytics, and M2M communication. Under our scenario, each edge node is regarded as an AI box, which provides a two-stage operation for this system. The two-stage operation includes the stage 1 data processing which splits the video stream into several segments and keeps only the key image frames containing the important information, and the stage 2 data transmission that caches and transmits the processed data (the key image frames) to the next layer. The cloud layer is a cloud data center that is in charge of big data processing such as safety inspection or other high-level applications. The deployed scheduling controller in this layer is responsible for conducting the task arrangement across the entire distributed system.

Specifically, in the edge layer, each AI box is regarded as an executor and a video stream from a camera in the end layer is regarded as a task. Under the LAN/WAN network environment, tasks (video streams) are assigned to multiple executors (AI boxes) under certain rules and are processed in a two-stage operation described previously. As captured by different kinds of cameras, video streams vary in quality, size, and length and contain different volumes of key image frames that need to be transmitted. Therefore, the processing time and transmission time for each task also vary correspondingly. To finish all tasks with a minimum time, the scheduling controller in the cloud layer needs to formulate a schedule that instructs each individual AI box to retrieve and execute the required tasks and achieves an optimal overall task completion time.

Suppose there are n two-stage tasks executed on m executors in the scheduling problem. Each task that arrives at the scheduling controller will be allocated to an executor through the scheduling controller. Each task in this scenario has different durations but contains the same two-stage operations, i.e., data processing and data transmission, while the data transmission operation must be performed after the completion of the data processing operation. According to the working mechanism, a set of assumptions is given below in this scheduling problem: 1) for each task, a data process operation needs to be completed first before the transmission operation can start; 2) each executor can execute a process operation and a transmission operation from different tasks simultaneously but can execute only one operation for a specific task at one time; 3) both operations of a task are executed and completed by the same executor; and 4) tasks cannot be preempted in this study.

B. Problem Formulation

In this article, we consider a two-stage scheduling problem of n tasks $J = \{J_1, J_2, \ldots, J_n\}$ scheduled to m executors $E = \{E_1, E_2, \ldots, E_m\}$. Each task J_i contains two operations $\{O_{i1}, O_{i2}\}$ with the duration $\{d_{i1}, d_{i2}\}$ for execution, where O_{i1} and O_{i2} represent the processing and transmission operation,

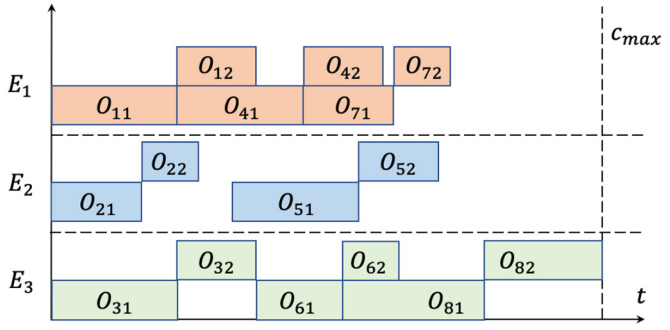


Fig. 2. Example of executed tasks in two-stage scheduling.

respectively, and d_{i1} and d_{i2} are the corresponding duration of each operation.

The goal of the scheduling algorithm is to find a feasible policy to allocate all the tasks to time intervals on the executors that minimize the total completion time, or makespan, denoted as $c_{\max}$. According to [28], we consider the two-stage scheduling problem as a typical multiprocessor flow-shop scheduling problem with the goal of minimizing the makespan.

Given a task J_i, we define T_{i1} and T_{i2} as the beginning times for O_{i1} and O_{i2}, c_{i1} and c_{i2} as the completion time, respectively. Accordingly, the completion time constraint can be concluded as follows:

$$\begin{aligned} c_{i1} &= T_{i1} + d_{i1} \\ c_{i2} &= T_{i2} + d_{i2}. \end{aligned} \tag{1}$$

A feasible two-stage schedule is given as the explanation, which is shown in Fig. 2.

As illustrated in Fig. 2, the makespan $c_{\max}$ equals to the completion time of the last executed task J_8, assigned to executor E_3.

In general, the makespan $c_{\max}$ means the max completion time within the m executors. Suppose task J_l is the last completed task, the makespan for the schedule can be expressed as follows:

$$c^*_{\max} = T_{l2} + d_{l2}. \tag{2}$$

Therefore, the optimization goal is to minimize $c^*_{\max}$, subject to $T_{i2} \geq c_{i1}, i = 1, 2, \ldots, n$, which means the starting time of the second operation should be later than the completing time of the first for each task.

IV. Deep-Reinforcement-Learning-Enhanced Two-Stage Scheduling

A. DRL-TSS Framework

The key objective of a two-stage scheduling algorithm is to assign tasks to executors in an optimal sequence, so as to ensure the minimal completion time. The basic framework of the proposed DRL-TSS is shown in Fig. 3.

Specifically, the proposed DRL-TSS model is constructed to handle n tasks by m executors, which includes two main modules: 1) Johnson's rule-based presorter and 2) DRL-based scheduler. Since the makespan minimization problem that schedules a set of two-stage tasks in multiple executors has been proved to be NP-hard, the DRL is utilized as a heuristic approach to obtain an approximate solution for the investigated two-stage scheduling problem. However, to avoid the worst situation occurred in the scheduling process, as shown in Fig. 3, all the tasks are preliminarily sorted into a specific Johnson's order and waiting for further scheduling. A scheduling controller is then introduced to observe the system state and make a scheduling action decision in the end–edge–cloud environment. Based on the newly designed DRL scheme which considers the maximal utilization of each executor, the controller outputs the scheduled action from the probabilistic transition according to the received cumulative rewards.

Algorithm 1 Johnson's Rule-Based Presorting

Input: Task list $J = \{J_1, J_2, \ldots, J_i, \ldots, J_n\}$, in which each task J_i contains two operations $\{O_{i1}, O_{i2}\}$ with execution durations $\{d_{i1}, d_{i2}\}$
Output: Sorted task list J' in Johnson's order

```
Initialize two task groups G1 = ∅, G2 = ∅, and J' = ∅
for each J_i in J do
    if d_i1 ≤ d_i2 then G1 = G1 ∪ J_i
    else G2 = G2 ∪ J_i
    end if
end for
Sort all tasks in G1 in ascending order based on the duration time d_i1 for each task J_i ∈ G1
Sort all tasks in G2 in descending order based on the duration time d_i2 for each task J_i ∈ G2
Merge the two task lists by appending G2 behind G1 as J' = G1 ∪ G2
return J'
```

B. Johnson's Rule-Based Presorting

Johnson's rule has proven to be able to obtain optimal solutions for scheduling problems in two-stage tasks with a single executor [29]. Therefore, it is used for task presorting in our model. It is known that the task list scheduled based on Johnson's rule is called Johnson's list, which has two theorems as follows.

Theorem 1: Johnson's list is an optimal solution for a two-stage, single executor scheduling problem.

Theorem 2: The subset of Johnson's list is also Johnson's list.

To improve the overall efficiency of the reinforcement-learning-based scheduling method, all the two-stage tasks are presorted into Johnson's list using the following scheme. According to Theorem 2, the task list scheduled to each executor is the subset of Johnson's list, this is also Johnson's list. This presorting operation can ensure all the parallel scheduling in each individual executor would be an optimal solution, respectively, following which the next issue would be how to schedule the presorted tasks to the multiple executors and make a global optimal.

As demonstrated in Algorithm 1, all the tasks are separated into two groups $G1$ and $G2$ by comparing the two operations' execution time spans d_{i1}, d_{i2} for each task J_i. Particularly, $G1$ contains the tasks that satisfy $d_{i1} \leq d_{i2}$, while $G2$ contains the tasks that satisfy $d_{i1} > d_{i2}$. In addition, all the tasks need to be sorted into Johnson's list in both $G1$ and $G2$ before being merged together finally.

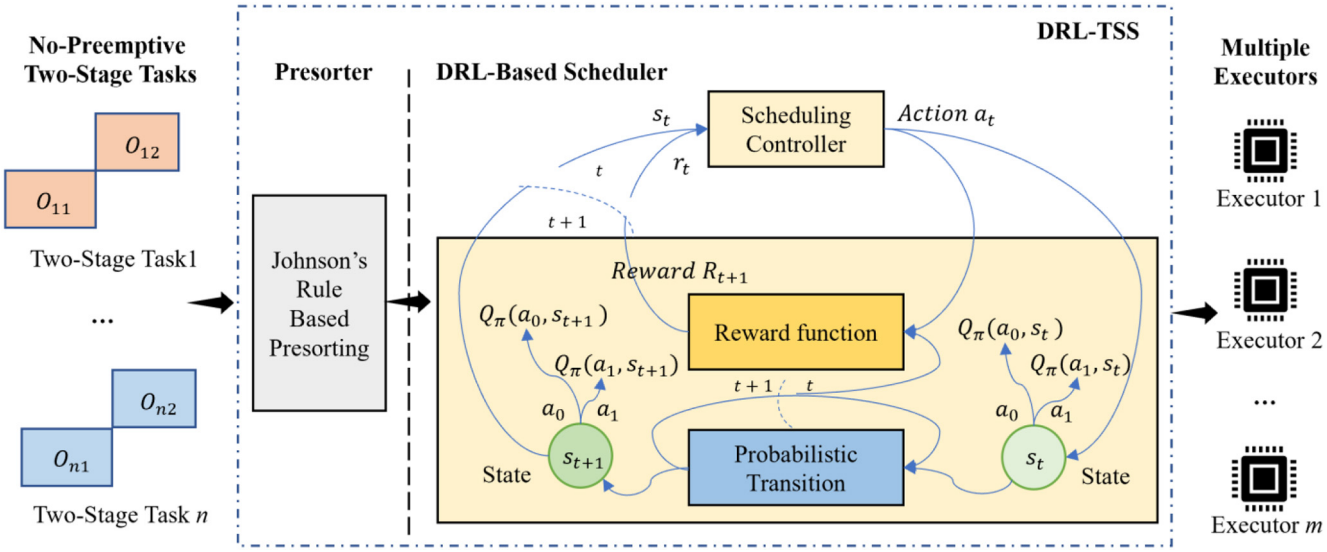


Fig. 3. Framework of DRL-TSS.

C. Deep Reinforcement Learning for Edge-Enabled Scheduling

Generally, reinforcement learning is based on the Markov decision process, which is characterized by the fact that the state transition of the system is only related to the current state. Typically, the Markov decision process-based reinforcement learning can be described using a four-tuple as follows:

$$\{S, A, P(s_t, a_t, s_{t+1}), R_t\} \tag{3}$$

where S is the state set at timestamp t, which describes the state space of the two-stage task scheduling. $s_t, s_{t+1} \in S$. A indicates the action space, which includes all the possible arrangement of any task J_i into any executor E_k. $a_t \in A$. $P(s_t, a_t, s_{t+1})$ refers to the state transition probability when it moves from state s_t to the next state s_{t+1} after executing action a_t. $R_t = R(s_t, a_t)$ is the corresponding reward obtained if executes action a_t at state s_t.

Specifically, in our scheduling problem, the Markov decision process of this scenario is formulated as follows.

State: The state s_t is a set of selected features for all the tasks and executors, which can be expressed as follows:

$$s_t = (D_{n1}(t), D_{n2}(t), C_{n1}(t), C_{n2}(t), U_m(t)) \tag{4}$$

where $D_{n1}(t) = \{d_{i1}\}$ and $D_{n2}(t) = \{d_{i2}\}$ denote the set of duration or processing time for the two stages of tasks that are executed at t, respectively. $C_{n1}(t) = \{c_{i1}\}$ and $C_{n2}(t) = \{c_{i2}\}$ indicate the set of completion time for the two stages of tasks that are executed at t, respectively, the value of each element is initialized to 0, and will be set to the corresponding completion time c_{i1} and c_{i2} once the task J_i has been finished. $U_m(t) = \{u_k(t)\}$ describes the payload of all m executors, in which each $u_k(t)$ indicates the utilization of the executor E_k.

In particular, the payload for each E_k at s_t can be defined as follows:

$$u_k(t) = \frac{c_k(t)}{c_{\max}(t)} \tag{5}$$

where $c_k(t)$ is the total completion time of all tasks scheduled to executor E_k according to Johnson's list.

It is noted that $u_k(t)$ is initialized to 0 at state s_0, which is then calculated and ranged from 0 to 1, according to the ratio of the completion time of current executor E_k to the makespan $c_{\max}(t)$ of the system at s_t.

Action: The action a_t at t is designed to determine which executor is scheduled to process the next task according to the action value function, until all tasks are executed.

Reward: The reward R_t stands for the instant reward at t when taking action a_t at state s_t. The goal of the DRL-based scheduler is to maximize the total rewards across all the states in S after t.

Actually, the total rewards will be evaluated by all the instant rewards in each step into the future after t. The weight for the reward in each step is defined and calculated as $\gamma \in [0, 1]$, which is interpreted as the probability to accumulate the reward score at every step and ensure the highest final return. Considering the optimal objective in this study is to minimize the ultimate makespan $c^*_{\max}$ and maximize the utilization of each executor, we first quantify the overall utilization of all the executors at t, which can be described as follows:

$$U_t = \frac{\sum_{k=1}^{m} u_k(t)}{m}. \tag{6}$$

It is noted that all the tasks scheduled to each executor are presorted into Johnson's list, according to Theorem 1, these scheduled two-stage tasks can be considered as the optimal solution in one executor. Thus, the overall equilibrium of the whole payload becomes the essential issue to realize the final optimal objective in our DRL-TSS model. Considering U_t reflects the overall payload performance of the whole system based on (6), it can be further used to measure the instant reward at t.

Accordingly, the value function $v_\pi(s)$ at state $s_t = s$ can be calculated as the expectation of the accumulative rewards

Algorithm 2 Training of DRL-TSS

Input: Sorted task list $J = \{J_1, J_2, \ldots, J_t, \ldots, J_n\}$ in Johnson's order
Output: Two-stage scheduling model M

1: Initialize action value function Q and target action value function Q' with weights $\theta' = \theta$ by Eq. (9)
2: Initialize learning step σ, greedy exploration probability ϵ, and discounting factor γ
3: Initialize experience replay buffer set D
4: **for** episode $eps = 1$ to $MaxBatchSize$ **do**
5: **for** $t = 1$ to n **do**
6: Select random action a_t that assigns J_t to a random executor with probability ϵ, otherwise $a_t = \text{argmax}_a Q(s_t, a; \theta)$
7: Obtain state s_{t+1} by executing action a_t, and calculate reward r_t by Eq. (6)
8: Store transition (s_t, a_t, r_t, s_{t+1}) in replay buffer D
9: Sample random minibatch of transition (s_j, a_j, r_j, s_{j+1}) from D
10: Calculate
$$y_j = \begin{cases} r_j & \text{if } eps \text{ terminates at } (j+1) \\ r_j + \gamma \max_{a'} Q'(s_{j+1}, a'; \theta') & otherwise \end{cases}$$
11: Calculate error $e_j = (y_j - Q(s_j, a_j; \theta))^2$ and conduct gradient descent step by e_j
12: Reset $Q' = Q$ in every σ steps
13: **end for**
14: **end for**

after t, which can be described as follows:

$$v_\pi(s) = \mathbb{E}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | s_t = s). \quad (7)$$

Finally, the action value function can be deduced as follows:

$$Q_\pi(s, a) = R(s_t, a_t) + \gamma \sum_{s' \in S} P(s_t, a_t, s'_{t+1}) v_\pi(s'_{t+1}). \quad (8)$$

The training process of the DRL-TSS model is shown in Algorithm 2, which is an approximation algorithm aiming to minimize the overall makespan for the edge-enabled two-stage scheduling.

This algorithm accepts a list of no-preemptive two-stage tasks J presorted in Johnson's order as the input. In each training episode, the possible scheduling, including the corresponding actions and state transitions, is then generated based on the ϵ-greedy exploration with $\epsilon < 10\%$ (in line 6). By executing the selected action a_t associated with the scheduled tasks in J at state s_t, the system reward r_t will be calculated based on (6). To speed up the state transition during the training process, the whole transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer D. We sample a minibatch of the transition from D to train the action value function $Q(s_j, a_j; \theta)$ to close to y_j during the gradient descent process (from lines 9 to 12).

V. Experiment and Analysis

In this section, simulation-based experiments are designed and conducted to demonstrate the performance of our proposed DRL-TSS in an end–edge–cloud IoT system, compared with three baseline methods.

A. Experiment Design

To simulate the scheduling scenario of arbitrary two-stage tasks executed on multiple executors in an end–edge–cloud IoT system, totally 1000 tasks are taken into account with randomly assigned execution and transmission time. To reflect the actual task scheduling in a typical IoT environment, tasks are composed of trivial tasks (processing time: 10–100 μs), median tasks (processing time: 100–1000 μs), and heavy tasks (processing time: 1000–10000 μs) with 30%, 45%, and 25% proportion, respectively, according to real end–edge–cloud IoT scenarios. The number of executors varies from 2 to 64. The makespan of all the tasks spent on the controller is investigated as our goal in the experiment. To mimic a real-world situation, a random workload is assigned to each executor in the initial state varying from 10 to 100 μs.

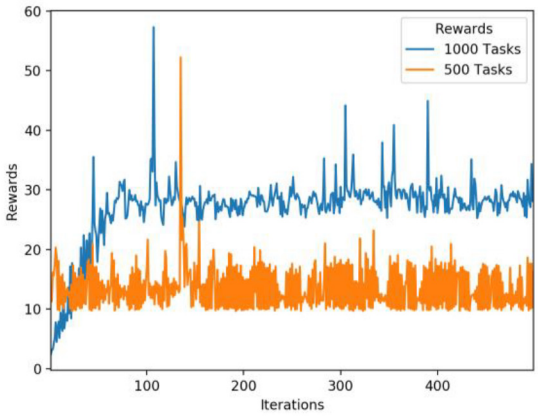


Fig. 4. Learning curves of DRL-TSS.

In addition, the DRL-TSS is compared with three two-stage scheduling algorithms, which are described as follows.

1) *Default First-in–First-Out scheduling (FIFO):* FIFO executes tasks on multiple executors by the same order as the tasks arrive in.
2) *Preprocessing List Scheduling (PLS) [30]:* PLS is a greedy algorithm that executes an ordered list of tasks by assigning them with some priorities in a preprocessing procedure.
3) *Johnson's Rule-based Genetic Algorithm (JRGA) [31]:* JRGA executes tasks on multiple executors by incorporating Johnson's rule in the decoding process of GA to optimize the makespan for each executor. Specifically, in this experiment, the settings for JRGA are configured as follows: the combination of TPX crossover operation and OM mutation operation, the population size of 20, the crossover probability of 60%, the mutation probability of 15%, and a maximum generation of 100.

The efficiency of all four methods is evaluated by the approximation ratio as follows:

$$\rho = \frac{T_c}{T_{\text{opt}}} \quad (9)$$

where T_c stands for the actual makespan spent for executing all the tasks by the algorithms. T_{opt} denotes the theoretically optimal makespan which indicates the performance standard to evaluate the proposed DRL-TSS and other methods.

It is noted that T_{opt} cannot be calculated since the problem investigated in this study is NP hard. Therefore, the bound of

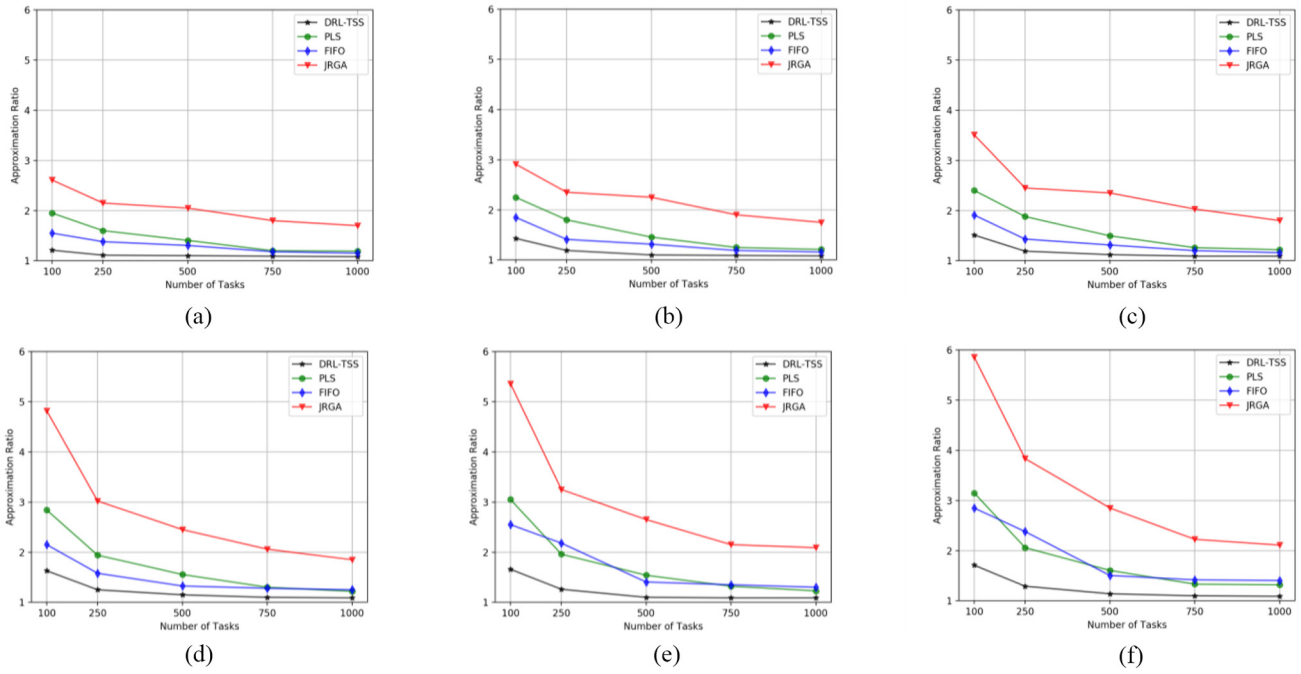


Fig. 5. Approximation ratio ρ achieved when executing specific number of tasks on 2–64 executors. Tasks executed by (a) 2 executors, (b) 4 executors, (c) 8 executors, (d) 16 executors, (e) 32 executors, and (f) 64 executors.

the optimal is applied in (10) instead. In the following experiment, the approximation ratio ρ is utilized to present how close the makespan gained by the algorithms to the bound optimal makespan, and we are interested in obtaining a minimum ρ in the experiment.

B. Evaluation on Learning Performance

The learning process of DRL-TSS is investigated via observing the reward gained along with the iterations. We set the discount factor of DRL to 0.9 and the learning rate to 0.5 empirically. The 1000 tasks and 500 tasks were assigned to a typical ten slave executors, respectively, to investigate how the rewards were gained in different task scheduling processes. The learning curves in these two situations are illustrated in Fig. 4.

As shown in Fig. 4, both the reward curves of 1000 tasks and 500 tasks increase fast and then stabilize after a few iterations (100 iterations for 1000 tasks and 20 iterations for 500 tasks) during the learning process. This convergence phenomenon depicts the fact that the proposed DRL-based scheduling method can quickly obtain a close to the optimal action strategy, which can be applied to the resource-constrained environment in end–edge–cloud IoT applications. In addition, it can be observed that the reward gained by the scenario of 1000 tasks (around 28) is greater than that of 500 tasks (around 13). This is likely due to the reason that more tasks may facilitate the learning process in such a situation.

C. Evaluation on Task Scheduling Efficiency

To demonstrate the efficiency of the proposed DRL-TSS algorithm, two evaluation scenarios are designed to investigate the performances of all four scheduling algorithms via the approximation ratio ρ.

First, we demonstrate how the approximation ratio ρ changes when different number of tasks are scheduled on varying number of executors, which means we try to investigate how the algorithms' approximation ratio varies with the growth of the number of tasks. The evaluation is performed by scheduling 100–1000 tasks on 2, 4, 8, 16, 32, and 64 executors, respectively. The evaluation results of the DRL-TSS and other three baseline methods are shown in Fig. 5.

As shown in Fig. 5(a)–(f), the proposed DRL-TSS is compared with PLS, FIFO, and JRGA by executing 100–1000 tasks on 2–64 executors. It can be seen that all the algorithms obtain a relatively low approximation ratio with a larger number of tasks and a higher approximation ratio in the cases with more executors. This is mainly due to a more balance scheduling situation occurring when executing more tasks on definite executors. It can be clearly observed that the proposed DRL-TSS outperforms all the baseline methods by achieving the lowest approximation ratio in all the tests and achieves a 1.1-approximation ratio when the number of tasks is 1000.

Second, we investigate how the approximation ratio ρ changes when an explicit number of tasks is assigned to different executors in the end–edge–cloud IoT environment. Different from the evaluation result shown in Fig. 5, Fig. 6 presents the influence of the varying number of executors on scheduling specific number of tasks.

As shown in Fig. 6, given a specific number of tasks, the uplifted approximation ratio curves demonstrate that all the algorithms perform worse when the number of executors is

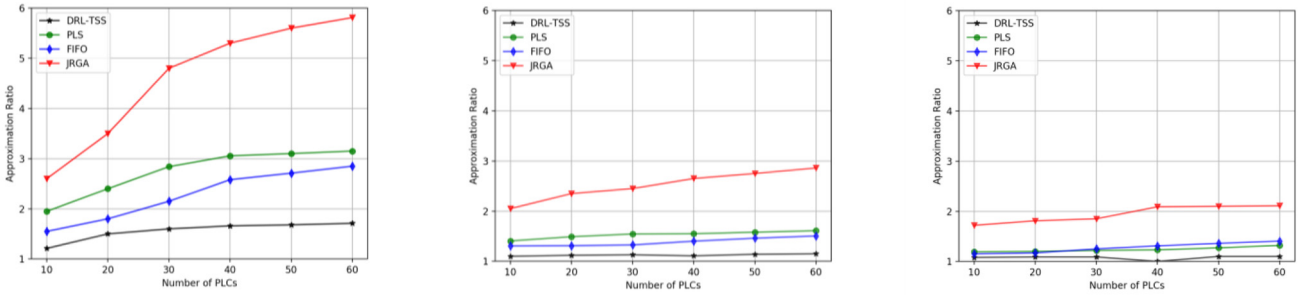


Fig. 6. Approximation ratio ρ of executing 100–1000 tasks on varying number of executors. (a) 100 tasks executed by multiple executors. (b) 500 tasks executed by multiple executors. (c) 1000 tasks executed by multiple executors.

increasing. This is due to the fact that the increasing number of executors brings unbalance and randomness to the scheduling process. In addition, all algorithms tend to be relatively stable along with the increasing executors except the JRGA, which indicates that the GA-based heuristic algorithm will be affected by the complex crossover and mutation operations. In particular, the performance of the proposed DRL-TSS remains stable when the executed tasks increase from 500 to 1000. Moreover, our algorithm achieves the lowest approximation ratio among all the methods, which approximates to a 1.0-approximation ratio in tests of 500 and 1000 tasks.

VI. Conclusion

The growing quantity of IoT devices demands a more efficient usage of shared computing and communication resources in an end–edge–cloud environment. This is particularly important for IoE applications, which are usually time critical and very costly to expand their computation and communication facilities. This article introduced a DRL-TSS method to find more efficient schedules that allow optimal use of the available resources, especially in a distributed IoE system that manages heterogeneous big data with multiple executors.

Following a deep learning framework for two-stage scheduling in end–edge–cloud IoT systems, two algorithms, namely, Johnson's rule-based task presorting scheme in multiple executors, and an improved DRL scheme that considers the maximal utilization of each executor in a newly designed instant reward, were developed to pursue the minimal overall makespan in dealing with the NP-hard scheduling problem. The proposed algorithm was evaluated in a simulated IoE setting compared with three existing scheduling approaches, in terms of learning efficiency and scheduling performance. The experimental results demonstrated that our proposed DRL-TSS algorithm could achieve a 1.1-approximation ratio to the optimal bound makespan when handling intensive tasks in end–edge–cloud-enabled IoE applications.

In future studies, we will investigate a more efficient deep learning scheme for scheduling optimization in IoE environments. More evaluations in more complex situations will be conducted to improve and examine the scheduling algorithm with better efficiency.

References

[1] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised deep reinforcement learning in support of IoT and smart city services," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, Apr. 2018.

[2] W. Liang, Y. Hu, X. Zhou, Y. Pan, and K. Wang, "Variational few-shot learning for microservice-oriented intrusion detection in distributed industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5087–5095, Aug. 2022, doi: 10.1109/TII.2021.3116085.

[3] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2300–2317, Feb. 2021.

[4] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. Wang, "Hierarchical adversarial attacks against graph neural network based IoT network intrusion detection system," *IEEE Internet Things J.*, early access, Nov. 24, 2021, doi: 10.1109/JIOT.2021.3130434.

[5] M. Panoff, R. G. Dutta, Y. Hu, K. Yang, and Y. Jin, "On sensor security in the era of IoT and CPS," *SN Comput. Sci.*, vol. 2, no. 1, pp. 1–9, 2021.

[6] X. Zhou, W. Liang, S. Shimizu, J. Ma, and Q. Jin, "Siamese neural network based few-shot learning for anomaly detection in industrial cyber-physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5790–5798, Aug. 2021.

[7] K. Chen *et al.*, "Internet-of-Things security and vulnerabilities: taxonomy, challenges, and practice," *J. Hardw. Syst. Security*, vol. 2, pp. 97–110, May 2018.

[8] X. Zhou, X. Yang, J. Ma, and K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet Things J.*, early access, May 6, 2021, doi: 10.1109/JIOT.2021.3077937.

[9] 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 Trans. Emerg. Topics Comput.*, vol. 8, no. 3, pp. 781–796, Jul.–Sep. 2020.

[10] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[11] H. Yang, W.-D. Zhong, C. Chen, A. Alphones, and X. Xie, "Deep-reinforcement-learning-based energy-efficient resource management for social and cognitive Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5677–5689, Jun. 2020.

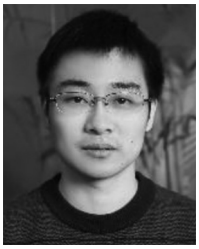
[12] X. Zhou, W. Liang, K. I.-K. Wang, and L. T. Yang, "Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations," *IEEE Trans. Comput. Soc. Syst.*, vol. 8, no. 1, pp. 171–178, Feb. 2021, doi: 10.1109/TCSS.2020.2987846.

[13] N. Jiang, Y. Deng, A. Nallanathan, and J. A. Chambers, "Reinforcement learning for real-time optimization in NB-IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1424–1440, Jun. 2019.

[14] P. Wang, L. T. Yang, J. Li, X. Li, and X. Zhou, "MMDP: A mobile-IoT based multi-modal reinforcement learning service framework," *IEEE Trans. Services Comput.*, vol. 13, no. 4, pp. 675–684, Jul./Aug. 2020.

[15] M. Kwon, J. Lee, and H. Park, "Intelligent IoT connectivity: Deep reinforcement learning approach," *IEEE Sensors J.*, vol. 20, no. 5, pp. 2782–2791, Mar. 2020.

[16] A. Nassar and Y. Yilmaz, "Reinforcement learning for adaptive resource allocation in fog RAN for IoT with heterogeneous latency requirements," *IEEE Access*, vol. 7, pp. 128014–128025, 2019.
[17] M. Camelo, M. Claeys, and S. Latré, "Parallel reinforcement learning with minimal communication overhead for IoT environments," *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1387–1400, Feb. 2020.
[18] X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020.
[19] A. Ivoghlian, Z. Salcic, and K. I.-K. Wang, "Adaptive wireless network management with multi-agent reinforcement learning," *Sensors*, vol. 22, no. 3, p. 1019, 2022.
[20] J. Leithon, L. A. Suárez, M. M. Anis, and D. N. K. Jayakody, "Task scheduling strategies for utility maximization in a renewable-powered IoT node," *IEEE Trans. Green Commun. Netw.*, vol. 4, no. 2, pp. 542–555, Jun. 2020.
[21] H. Lee and J. Lee, "Resource and task scheduling for SWIPT IoT systems with renewable energy sources," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2729–2748, Apr. 2019.
[22] Q. Qi *et al.*, "Scalable parallel task scheduling for autonomous driving using multi-task deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 13861–13874, Nov. 2020.
[23] Y. Wang, J. Liu, and J. Hu, "Communication-aware task scheduling for energy-harvesting nonvolatile processors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 8, pp. 1796–1806, Aug. 2020.
[24] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. Zhou, "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4076–4087, Oct. 2018.
[25] H. He, H. Shan, A. Huang, Q. Ye, and W. Zhuang, "Edge-Aided Computing and Transmission Scheduling for LTE-U-Enabled IoT," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7881–7896, Dec. 2020.
[26] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4411–4422, Jun. 2019.
[27] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative task scheduling for IoT-assisted edge computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020.
[28] J. A. Hoogeveen, J. K. Lenstra, and B. Veltman, "Preemptive scheduling in a two-stage multiprocessor Flow shop is NP-hard," *Eur. J. Oper. Res.*, vol. 89, no. 1, pp. 172–175, 1996.
[29] S. M. Johnson, "Optimal two and three stage production schedules with setup times included," *Naval Res. Logist.*, vol. 1, no. 1, pp. 61–68, 1954.
[30] W. Liang, C. Hu, M. Wu, and Q. Jin, "A data intensive heuristic approach to the two-stage streaming scheduling problem," *J. Comput. Syst. Sci.*, vol. 89, pp. 64–79, Nov. 2017.
[31] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 597–610, Jul.–Sep. 2019.



Xiaokang Zhou (Member, IEEE) received the Ph.D. degree in human sciences from Waseda University, Tokyo, Japan, in 2014.

He is currently an Associate Professor with the Faculty of Data Science, Shiga University, Hikone, Japan. From 2012 to 2015, he was a Research Associate with the Faculty of Human Sciences, Waseda University. He has also been working as a Visiting Researcher with the RIKEN Center for Advanced Intelligence Project, RIKEN, Tokyo, since 2017. He has been engaged in interdisciplinary research works in the fields of computer science and engineering, information systems, and social and human informatics. His recent research interests include ubiquitous computing, big data, machine learning, behavior and cognitive informatics, cyber–physical–social systems, and cyber intelligence and security.

Dr. Zhou is a member of the IEEE CS; ACM, USA; IPSJ; JSAI, Japan; and CCF, China.

Wei Liang (Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Central South University, Changsha, China, in 2005 and 2016, respectively.

From 2014 to 2015, he was a Researcher with the Department of Human Informatics and Cognitive Sciences, Waseda University, Tokyo, Japan. He is currently working with the Base of International Science and Technology Innovation and Cooperation on Big Data Technology and Management, Hunan University of Technology and Business, Changsha. His research interests include information retrieval, data mining, and artificial intelligence.

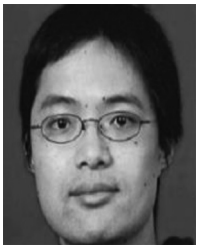
Dr. Liang is a member of the IEEE CS, and CCF, China.

Ke Yan (Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from the School of Computing, National University of Singapore (NUS), Singapore, in 2006 and 2012, respectively.

He is currently an Assistant Professor with NUS. He has published more than 70 full-length papers with highly ranked conferences and journals, including Association for the Advancement of Artificial Intelligence, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON SUSTAINABLE ENERGY, IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS, and *Applied Energy*. He is actively engaged in cross-discipline research fields, including machine learning, artificial intelligence, cyber intelligence, applied mathematics, sustainability, and applied energy.

Weimin Li (Member, IEEE) received the bachelor's and master's degrees from the Shandong University of Science and Technology, Qingdao, China, and the Ph.D. degree from Donghua University, Shanghai, China.

He is a Professor with the School of Computer Engineering and Science, Shanghai University, Shanghai, China. He was a JSPS Research Fellow with the Department of Human Informatics and Cognitive Sciences, Waseda University, Tokyo, Japan, from 2012 to 2013. He was a Visiting Scholar with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA, from 2015 to 2016. He has been involved in the extensively research works in the fields of computer science, service computing, group behavior, and database technology. His current research interests include social computing, data mining and analytics, group behavior modeling and simulating, and service recommendations.



Kevin I-Kai Wang (Member, IEEE) received the Bachelor of Engineering degree (Hons.) in computer systems engineering and the Ph.D. degree in electrical and electronics engineering from the Department of Electrical and Computer Engineering, The University of Auckland, Auckland, New Zealand, in 2004 and 2009, respectively.

He is currently a Senior Lecturer with the Department of Electrical, Computer, and Software Engineering, The University of Auckland. He was also a Research Engineer designing commercial home automation systems and traffic sensing systems from 2009 to 2011. His current research interests include wireless sensor network-based ambient intelligence, pervasive healthcare systems, human activity recognition, behavior data analytics, and biocybernetic systems.

Jianhua Ma (Senior Member, IEEE) received the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1990.

He is a Professor with the Department of Digital Media, Faculty of Computer and Information Sciences, Hosei University, Tokyo, Japan. He is one of pioneers in research on Hyper World and Cyber World (CW) since 1996. He first proposed Ubiquitous Intelligence toward Smart World, which he envisioned in 2004, and was featured in the European ID People Magazine in 2005. He has conducted several unique CW-related projects, including the Cyber Individual (Cyber-I), which was featured by and highlighted on the front page of IEEE Computing Now in 2011. He has published more than 300 papers, coauthored five books, and edited over 30 journal special issues. He has founded three IEEE Congresses on "Smart World," "Cybermatics," and "Cyber Science and Technology," respectively, as well as IEEE Conferences on Ubiquitous Intelligence and Computing, Pervasive Intelligence and Computing, Advanced and Trusted Computing, Dependable, Autonomic and Secure Computing, Cyber Physical and Social Computing, Internet of Things, and Internet of People. His research interests include multimedia, networking, pervasive computing, social computing, wearable technology, IoT, smart things, and cyber intelligence.

Dr. Ma is the Chair of the IEEE SMC Technical Committee on Cybermatics, the Founding Chair of the IEEE CIS Technical Committee on Smart World, and in the advisory board of the IEEE CS Technical Committee on Scalable Computing. He is a member of ACM.

Qun Jin (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from Nihon University, Tokyo, Japan, in 1992.

He is a Professor with the Networked Information Systems Laboratory, Department of Human Informatics and Cognitive Sciences, Faculty of Human Sciences, Waseda University, Tokorozawa, Japan. He has been extensively engaged in research works in the fields of computer science, information systems, and social and human informatics. He seeks to exploit the rich interdependence between theory and practice in his work with interdisciplinary and integrated approaches. His recent research interests cover human-centric ubiquitous computing, behavior and cognitive informatics, big data, data quality assurance and sustainable use, personal analytics and individual modeling, intelligence computing, blockchain, cyber security, cyber-enabled applications in healthcare, and computing for well-being.

Dr. Jin is a Senior Member of the Association for Computing Machinery and Information Processing Society of Japan.