Online Task Scheduling and Resource Allocation for Intelligent NOMA-Based Industrial Internet of Things

Kunlun Wang, *Member, IEEE*, Yong Zhou, *Member, IEEE*, Zening Liu, *Student Member, IEEE*, Ziyu Shao, *Senior Member, IEEE*, Xiliang Luo, *Senior Member, IEEE*, and Yang Yang, *Fellow, IEEE*

***Abstract*— Fog computing (FC) has the potential to process computation-intensive tasks in Industrial Internet of Things (IIoT) systems. In parallel with the development of FC, non-orthogonal multiple access (NOMA) has been recognized as a promising technique to significantly improve the spectrum efficiency. In this paper, a NOMA-based FC framework for IIoT systems is considered, where multiple task nodes offload their tasks via NOMA to multiple nearby helper nodes for execution. We formulate a joint task scheduling and subcarrier allocation problem, with an objective to minimize the total cost in terms of the delay and energy consumption, while taking into account the practical communication and computation constraints. Note that the task scheduling includes task, computation resource, and power allocations. Since the task and subcarrier allocations involve binary variables, it is challenging to obtain an optimal solution for such a combinatorial problem. To this end, we solve the task scheduling and subcarrier allocation problem in an online learning fashion. During the online learning process, we propose an iterative algorithm to jointly optimize the subcarrier allocation and task scheduling in each time episode. Simulation results show that the proposed scheme can significantly reduce the sum cost compared to the baseline schemes.**

***Index Terms*— Fog computing, industrial Internet of Things, non-orthogonal multiple access, delay-energy tradeoff, online learning.**

I. Introduction

DUE to the rapid development of Internet of Things (IoT), more and more networked things and smart objects are being involved in industrial IoT (IIoT) [2]. IIoT is capable of providing endless networking to mobile devices and is regarded as an interconnection of many devices through a communication system to generate a top-notch system. Compounded with these, IIoT enables reliable and efficient information monitoring, collection, exchange, analysis, and delivery [3]–[5]. On the other hand, as the number of connected industrial devices increases rapidly, IIoT services urge for more and more computation and communication resources. However, as the industrial devices usually have limited computation and energy resources, it is a big challenge for the service providers to promote these novel applications. To overcome these limitations, fog computing (FC) has recently been integrated into IIoT, which has the potential to tackle the contradiction between the resource-constrained industrial devices and the computation-intensive applications. Specifically, FC is implemented at network systems regarded as highly virtualized platforms and it provides computation, storage, and networking services to meet the demands of neighboring industrial devices [6]–[8]. An industrial device that has task to execute, termed as a *task node*, may not have enough computation resource to execute its task and can offload its task to the nearby *helper nodes* for execution, which have under-utilized computation resources. As a result, FC can significantly reduce the computing burden of the industrial devices by efficiently utilizing the abundant computation resources around them.

A. Related Works

In recent years, FC has gained increasing popularity in a diverse range of IIoT scenarios [9]–[13]. Different from the traditional cloud-based IIoT, FC provides a more efficient platform that realizes low-latency and high energy-efficiency for IIoT systems. Byers in [9] discussed the architecture and techniques for fog-enabled IoT Networks. Li *et al.* in [10] proposed a service popularity-based smart resources partitioning scheme for FC-enabled IIoT. Chekired *et al.* in [11], proposed a hierarchical fog serversİ¯ deployment at the network service layer and introduced a workload assignment algorithm to offload peak loads over the fog hierarchy. Additionally, task scheduling has been envisioned as an important design aspect in the FC framework [14]–[16]. The multi-device task scheduling has been studied in multi-channel wireless interference networks [6], cognitive radio networks [15], and etc. With limited

Manuscript received July 16, 2019; revised January 14, 2020; accepted February 26, 2020. Date of publication March 16, 2020; date of current version May 7, 2020. This work was supported in part by the Natural Science Foundation of China (NSFC) under Grant 61932014, the Nature Science Foundation of Shanghai under Grant 19ZR1433900, and Huawei Technologies under Grant YBN2019125163. This article was presented in part at the IEEE GLOBECOM 2019, Waikoloa, HI, USA, December 2019. *(Corresponding author: Yang Yang.)*

Kunlun Wang, Yong Zhou, Ziyu Shao, and Xiliang Luo are with the School of Information Science and Technology (SIST), ShanghaiTech University, Shanghai 201210, China (e-mail: wangkl@shanghaitech.edu.cn; zhouyong@shanghaitech.edu.cn; shaozy@shanghaitech.edu.cn; luoxl@shanghaitech.edu.cn).

Zening Liu and Yang Yang are with the Shanghai Institute of Fog Computing Technology (SHIFT), SCA and SIST, ShanghaiTech University, Shanghai 201210, China (e-mail: liuzy1@shanghaitech.edu.cn; yangyang@shanghaitech.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2020.2980908

frequency resources, a key challenge is how to efficiently share the communication and computation resources among the cellular and IIoT systems to support the remote task computation for industrial devices. To address this challenge and improve the performance of task scheduling in a multi-device FC framework, various joint communication and computation resource allocation approaches have been proposed [17]–[19]. A number of research efforts have been dedicated to the binary offloading case, where the task of each device is not partitionable and can only be offloaded as a whole [6], [20], [21]. These multi-device offloading strategies are based on either frequency-division multiple access (FDMA) [6], [20] or code division multiple access (CDMA) [21]. On the other hand, to address the partial offloading issue, time-division multiple access (TDMA) [15] and orthogonal frequency-division multiple access (OFDMA) [14] based multi-device task offloading schemes were proposed for local computing and remote computing, respectively. The authors in [14], [15] formulated an optimization problem to maximize the time-average energy efficiency for task executions.

Although the above outstanding works have studied computation offloading, the potential benefits of non-orthogonal multiple access (NOMA) that can further enhance the performance of the FC framework have not been explored. Different from the conventional orthogonal multiple access (OMA), such as FDMA and TDMA, NOMA allows multiple devices to concurrently communicate with an access point (AP) over the same resource block, and hence enhancing the spectrum efficiency [22]–[24]. Note that the multiuser detection techniques such as successive interference cancellation (SIC) should be implemented at the receiver side [25], [26] to mitigate the co-channel interference. It has been shown in [27] that NOMA achieves much better performance in terms of the coverage probability and throughput compared to OMA. As expected, the integration of FC and NOMA can enhance the performance of task offloading in multi-user FC systems [28]–[30]. In particular, Liu *et al.* in [28] focused on the task offloading problem in fog and cloud integrated networks, where multiple users transmit their data to the same fog node using NOMA over the same spectrum resource. In [29], Zhang *et al.* investigated a solution of resource allocation to improve the network performance of NOMA-based Fog Radio Access Networks (F-RANs). First, they proposed the network architecture of NOMA-based F-RANs. Then they studied the resource management, which included the power and subchannel allocation. Moreover, they showed that the proposed resource management mechanisms enhance the net utility of NOMA-based F-RANs. In [30], Wen *et al.* formulated an energy efficiency maximization problem in downlink NOMA hierarchical fog networks. Fang *et al.* in [31] investigated the task completion time minimization problem in NOMA-enabled multiuser mobile edge computing (MEC) networks. Although the aforementioned studies have demonstrated the benefits of NOMA-based FC, they have not taken into account both the dynamic channel conditions and dynamic computational cost in the time domain, which are particularly important for time-variant IIoT systems.

B. Contributions and Organization

In this paper, we aim to jointly optimize the computation and communication resource allocation for NOMA and fog-enabled IIoT systems, where the spectrum efficiency gain of NOMA is exploited. In our proposed FC framework, the industrial devices running computation-intensive applications, termed as task nodes, can offload their computation tasks to nearby fog devices having under-utilized computation resources, termed as helper nodes. We assume that the task and helper nodes share the spectrum resources of cellular users, termed as busy nodes, in the underlay mode to improve the spectrum efficiency. By establishing a relationship between the task offloading cost and the energy and delay cost, we formulate a joint task scheduling and subcarrier allocation problem. The objective is to minimize the total cost in terms of the delay and energy consumption, while taking into account the practical communication and computation constraints. Since the task allocation and subcarrier allocations involve binary variables, it is challenging to obtain an optimal policy for such a combinatorial problem. In addition, considering dynamic channel conditions and dynamic computational cost further complicates the optimization problem. To this end, we solve the task scheduling and subcarrier allocation problem in an online learning fashion. During the online learning process, we propose an iterative algorithm to jointly optimize the subcarrier allocation and task scheduling in each time episode. In the iterative algorithm, we first solve the task scheduling subproblem by optimizing the power allocation, computational resource allocation. Then, we propose a subcarrier matching method to find the cost minimized subcarrier allocation. The main contributions of this paper are summarized as follows.

- We develop a novel NOMA-based task scheduling framework for IIoT systems, where multiple task nodes offload their tasks to the nearby helper nodes. We formulate an energy and delay cost minimization problem by jointly optimizing task, power, subcarrier, and computation resource allocation for each task node.
- Based on the problem formulation and theoretical analysis, we divide the original optimization problem into two subproblems in each time episode, i.e., task scheduling subproblem and subcarrier allocation subproblem. Due to the dynamic channel conditions and dynamic computational cost, we develop an online learning based algorithm to obtain the task scheduling and subcarrier allocation decisions. The proposed online learning algorithm is proved to always converge to the optimal policy.
- We formulate the subcarrier allocation subproblem as a one-to-many matching problem when the computation resource and power allocations for each task node are fixed. Furthermore, we prove the convergence of the proposed matching-based subcarrier allocation algorithm.
- During the online learning process, we jointly consider subcarrier allocation and task scheduling and proposed an iterative joint subcarrier allocation and task scheduling algorithm in each time episode, which achieves efficient task scheduling with low cost as long as the stable

matching between task nodes and subcarriers can be guaranteed.

- Simulation results demonstrate that the proposed scheduling algorithm achieves significant performance improvement compared to the traditional scheduling strategies under various system settings.

We extend the work in [1] from several aspects. In [1], the objective is to optimize task scheduling results. In this paper, we have extended the system model to take into account the subcarrier allocation and also proposed a novel matching theory based algorithm to solve the cost minimization problem. The proposed matching theory based algorithm is motivated by its ability to tackle the combinatorial optimization problems and result in a distributed solution. In addition, we also proposed an iterative subcarrier allocation and task scheduling algorithm to jointly optimize the subcarrier allocation and task scheduling results.

The rest of the paper is organized as follows. Section II describes the system model, including the NOMA transmission, the channel model, and the computation model. In Section III, we formulate a task scheduling and subcarrier allocation problem. In Section IV, we optimize the power allocation, computational resource allocation, subcarrier allocation, and task offloading by proposing a reinforcement learning based algorithm in NOMA-based FC networks. In Section V, we show the simulation results. Finally, the conclusions are made in Section VI.

C. Notations

Notations: $|\mathbf{x}|$ and $\|\mathbf{x}\|$ denote the cardinality and the Euclidean norm of vector $\mathbf{x}$, respectively. $\mathsf{E}[\cdot]$ denotes the expectation. $\mathrm{diag}(\cdot)$ is a diagonal matrix. The distribution of a circularly symmetric complex Gaussian random variable x with mean ϖ and covariance Δ is denoted by $x \sim \mathcal{CN}(\varpi, \Delta)$, where $\sim$ means "distributed as".

II. System Model

Consider an IIoT system consisting of two types of nodes, i.e., task node and helper node. In particular, multiple single-antenna task nodes can offload their tasks to M nearby single-antenna helper nodes, denoted as $\mathcal{M} = \{1, 2, \cdots, M\}$, as shown in Fig. 1. Let $\mathcal{I} = \{1, 2, \cdots, |\mathcal{I}|\}$ denote the set of task nodes. Specifically, each task node is equipped with a sensor, which is a low-power wireless device. We denote the set of task nodes associated with the mth helper node as $\mathcal{I}_m$, and we have $\mathcal{I} = \bigcup_{m=1}^{M} \mathcal{I}_m$. Each helper node with an empty queue has under-utilized computation resources, which can be utilized to perform the tasks offloaded from the task nodes. We assume that there are M subcarriers available for task offloading. Since the number of helper nodes is the same as that of subcarriers, we refer to the set of subcarriers interchangeably with the set of helper nodes. Without loss of generality, we assume that the mth subcarrier corresponds to the mth helper node. We assume that the task and helper nodes share the subcarriers of the cellular users (i.e., busy nodes) in the underlay mode.

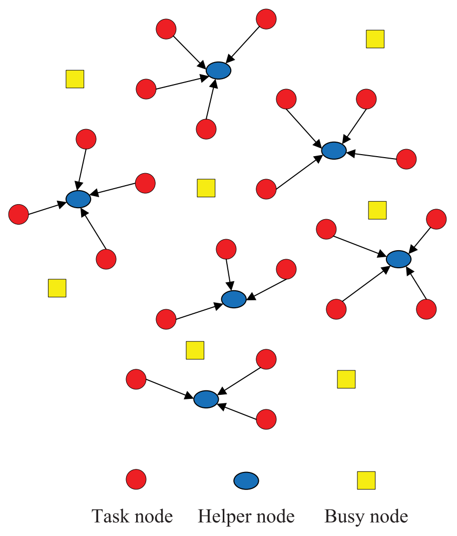


Fig. 1. System model.

Different from the conventional OMA transmission where each subcarrier can be allocated to at most one task node, we consider NOMA transmission, which enables multiple task nodes to simultaneously offload their tasks to the same helper node. With NOMA, the signals from different task nodes are superimposed at the helper node. Note that the two-user NOMA scheme, termed as multi-user superposition transmission (MUST), has been specified in Long Term Evolution Advanced (LTE-A) [32].

A. Transmission Model

Recall that the mth helper node occupies the mth subcarrier. Let $\eta_{i,m} \in \{0, 1\}$ be a binary variable, which is defined as

$$\eta_{i,m} = \begin{cases} 1, & \text{if node } i \text{ offloads its task at the } m\text{th subcarrier,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We denote $\boldsymbol{\eta}$ as the subcarrier allocation matrix, which can be shown as

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_{1,1} & \eta_{1,2} & \cdots & \eta_{1,M} \\ \eta_{2,1} & \eta_{2,2} & \cdots & \eta_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ \eta_{|\mathcal{I}|,1} & \eta_{|\mathcal{I}|,2} & \cdots & \eta_{|\mathcal{I}|,M} \end{bmatrix}. \tag{2}$$

To fully exploit the under-utilized licensed spectrum, we assume that each subcarrier occupied by a busy node can be reused by all the task nodes allocated to this subcarrier. We denote $\alpha_i \in \{0, 1\}$ and $\mathbf{a} = [\alpha_1, \cdots, \alpha_{|\mathcal{I}|}]$ as the binary offloading decision variable of task node i and the offloading decision vector, respectively. We have $\alpha_i = 0$ if task node i executes its task by local computing, and $\alpha_i = 1$ otherwise. Let p_i and p_b be the transmit power of the ith task node which is selected prior to the subcarrier allocation according to a power allocation method and the transmit power of the busy node, respectively. Denote $\mathbf{p} = [p_1, \cdots, p_{|\mathcal{I}|}]$ and z_m as the power allocation vector and the additive white Gaussian noise (AWGN) with variance σ_m^2, respectively. Hence, the received signal at the mth helper node is given by

$$y_m = \sum_{i\in\mathcal{I}_m} \eta_{i,m}\alpha_i\sqrt{p_i}\sqrt{d_{i,m}}h_{i,m}x_i + \sqrt{p_b}\sqrt{d_{b,m}}h_{b,m}x_b + z_m, \quad \forall m \in \mathcal{M}, \quad (3)$$

where $d_{i,m}$ and $h_{i,m} \sim \mathcal{CN}(0,1)$ denote the path loss and small scale fading of the channel between the ith task node and the mth helper node, respectively. Similarly, $d_{b,m}$ and $h_{b,m} \sim \mathcal{CN}(0,1)$ represent the path loss and small scale fading of the channel between the busy node and the mth helper node, respectively. Let x_i and x_b be the transmit symbol of task node i and the transmit symbol of the busy node, respectively. In addition, x_i and x_b satisfy $\mathsf{E}[|x_i|^2] = 1$ and $\mathsf{E}[|x_b|^2] = 1$, respectively.

Based on (3), the signal-to-interference-plus-noise ratio (SINR) of symbol x_i observed by the mth helper node can be expressed as

$$\gamma_{i,m} = \frac{\eta_{i,m}\alpha_i p_i d_{i,m}|h_{i,m}|^2}{\sum_{k\neq i}\eta_{k,m}\alpha_k p_k d_{k,m}|h_{k,m}|^2 + p_{b,m}d_{b,m}|h_{b,m}|^2 + \sigma_m^2}. \quad (4)$$

According to the principle of the SIC technique [25], we assume that the task nodes in set $\mathcal{I}_m$ are ordered based on the following criterion: $d_{1,m}|h_{1,m}|^2 \leq d_{2,m}|h_{2,m}|^2 \leq \cdots \leq d_{|\mathcal{I}_m|,m}|h_{|\mathcal{I}_m|,m}|^2$. SIC is carried out at the mth helper node to facilitate NOMA transmission. Specifically, the mth helper node first decodes symbol $x_{|\mathcal{I}_m|}$ transmitted by task node $|\mathcal{I}_m|$, and then decodes symbol $x_{|\mathcal{I}_m|-1}$ after cancelling the interference due to symbol $x_{|\mathcal{I}_m|}$, followed by symbol $x_{|\mathcal{I}_m|-2}$, symbol $x_{|\mathcal{I}_m|-3}$, and so on, until symbol x_1. Therefore, the SINR of symbol x_i observed by the mth helper node is given by

$$\gamma_{i,m} = \frac{\eta_{i,m}\alpha_i p_i d_{i,m}|h_{i,m}|^2}{\zeta_{i,m}^{\text{in}} + \zeta_m^{\text{out}} + \sigma_m^2}, \quad (5)$$

where $\zeta_{i,m}^{\text{in}} = \sum_{k=1}^{i-1}\eta_{k,m}\alpha_k p_k d_{k,m}|h_{k,m}|^2$ represents the interference from the superimposed symbols that occupies the mth subcarrier, and $\zeta_m^{\text{out}} = p_{b,m}d_{b,m}|h_{b,m}|^2$ represents the interference from the busy node. To ensure that the helper node can decode the ith task node's signal, the SINR of symbol x_i should satisfy: $\gamma_{i,m} > \gamma_{i,m}^{\text{th}}, \forall i \in \mathcal{I}_m$, where $\gamma_{i,m}^{\text{th}}$ is the SINR threshold required to successfully decode the received symbol of task node i.

Based on the SINR in (5), the achievable data rate of the ith task node corresponding to the mth subcarrier is given by

$$r_{i,m} = \log_2\left(1 + \frac{\eta_{i,m}\alpha_i p_i d_{i,m}|h_{i,m}|^2}{\zeta_{i,m}^{\text{in}} + \zeta_m^{\text{out}} + \sigma_m^2}\right). \quad (6)$$

B. Computation Model

1) Local Computing: Local computing refers to the case that task node i computes its task R_i locally. Let $D_i^{(l)}$ and $f_i^{(l)}$ denote the local computation delay for computing task R_i and the computation capacity (i.e., CPU cycles per second) of task node i, respectively. It is worth noting that the task nodes can have different computational capacities. As the delay only includes the processing delay of local CPUs, we derive the local computation delay $D_i^{(l)}$ as

$$D_i^{(l)} = \frac{C_i}{f_i^{(l)}}, \quad \forall i \in \mathcal{I}, \quad (7)$$

where C_i is the total number of CPU cycles required to accomplish the computation task R_i and is positively related to the task size. At the same time, we denote z_i and $E_i^{(l)}$ as the energy consumption per CPU cycle and the corresponding energy consumption for computing task R_i, respectively. Hence, we have

$$E_i^{(l)} = z_i C_i, \quad \forall i \in \mathcal{I}. \quad (8)$$

We define the cost function as the weighted sum of the delay and energy consumption, and denote w_i^t and w_i^e as the weights of delay and energy consumption, respectively, without loss of generality, we have

$$w_i^t + w_i^e = 1, \quad 0 \leq w_i^t \leq 1, \ 0 \leq w_i^e \leq 1. \quad (9)$$

Then, the total cost of local computing can be obtained by combining (7) and (8) and is given by

$$\Omega_i^{(l)} = w_i^t D_i^{(l)} + w_i^e E_i^{(l)}. \quad (10)$$

2) Task Offloading: Task offloading refers to the case that the task is offloaded to be executed by a nearby helper node. Specifically, if a task is to be processed by the helper node, the task node needs to transmit the task through the shared wireless channel. The transmission delay of task R_i is given by

$$D_{i,t}^{(o)} = \frac{B_i}{r_{i,m}}, \quad \forall i \in \mathcal{I}, \quad (11)$$

where B_i is the size of task R_i. The corresponding energy cost is given by

$$E_{i,t}^{(o)} = p_i D_{i,t}^{(o)} = \frac{p_i B_i}{r_{i,m}}. \quad (12)$$

After receiving the computation tasks, the helper node allocates the computation resource for each task. We denote f_i, $\mathbf{f}$, and F as the allocated computation resource (i.e., CPU cycles per second) to complete task R_i, the computation resource allocation vector, and the entire computation resource of the helper node, respectively. Hence, the delay due to the task computation in the helper node is given by

$$D_{i,p}^{(o)} = \frac{C_i}{f_i}. \quad (13)$$

Since the total amount of assigned computation resource cannot exceed the entire computation resource of the helper node, we have $\sum_{i\in\mathcal{I}_m,\forall m}\alpha_i f_i \leq F$. After task offloading, task node i stays in the idle state and we define the power consumption of the idle state as p_i^i. Hence, the corresponding energy consumption is

$$E_{i,p}^{(o)} = p_i^i D_{i,p}^{(o)} = \frac{p_i^i C_i}{f_i}. \quad (14)$$

As in [6], [33]–[35], we neglect the delay and energy consumption required for the helper node to send the computation outcome back to the task node. After combining (11)-(14), the total cost of task node i through task offloading is given by

$$\Omega_{i,m}^{(o)} = w_i^t \left(\frac{B_i}{r_{i,m}} + \frac{C_i}{f_i} \right) + w_i^e \left(\frac{p_i B_i}{r_{i,m}} + \frac{p_i^i C_i}{f_i} \right). \quad (15)$$

Based on (10) and (15), the weighted sum cost of all task nodes in terms of delay and energy costs can be expressed as

$$\Omega_{\text{all}} = \sum_{m \in \mathcal{M}} \sum_{i \in \mathcal{I}} (1-\alpha_i)\Omega_i^{(l)} + \alpha_i \Omega_{i,m}^{(o)}. \quad (16)$$

III. Problem Formulation and Analysis

A. Problem Formulation

In this section, we formulate a joint task, power, and computation resource allocation problem with an objective to minimize the total cost in terms of the delay and energy consumption, taking into account the communication and computation constraints. Based on the system model presented in Section II, the formulated problem is given by

$$\begin{aligned}
(\text{P1}) \ \min_{\mathbf{a},\mathbf{p},\mathbf{f},\boldsymbol{\eta}} \ & \sum_{\text{m} \in \mathcal{M}} \sum_{\text{i} \in \mathcal{I}} (1-\alpha_\text{i})\Omega_\text{i}^{(\text{l})} + \alpha_\text{i} \Omega_{\text{i,m}}^{(\text{o})} && (17a)\\
\text{s.t. } & \alpha_\text{i} \in \{0,1\}, \quad \forall \text{i} \in \mathcal{I}, && (17b)\\
& (1-\alpha_i) D_i^{(l)} + \alpha_i D_i^{(o)} \le \psi_i, \quad \forall i \in \mathcal{I}, && (17c)\\
& 0 \le f_i \le \alpha_i F, \quad \forall i \in \mathcal{I}, && (17d)\\
& \sum_i \alpha_i f_i \le F, \quad \forall i \in \mathcal{I}_m, && (17e)\\
& \gamma_{i,m} > \gamma_{i,m}^{\text{thr}}, \quad \forall i \in \mathcal{I}, && (17f)\\
& \sum_m \eta_{i,m} \le 1, \quad \forall i \in \mathcal{I}, && (17g)\\
& \sum_i \eta_{i,m} \le |\mathcal{I}_m|, \quad \forall m \in \mathcal{M}, && (17h)\\
& \eta_{i,m} \in \{0,1\}, \quad \forall i \in \mathcal{I}, \quad \forall m \in \mathcal{M}. && (17i)
\end{aligned}$$

Constraint (17b) represents that each task node chooses to compute its own task either by local computing or by task offloading. Constraint (17c) guarantees that the delay cost should not exceed the maximum tolerable delay ψ_i. Since the computational capacity of each helper node is F, constraints (17d) and (17e) guarantee that the allocated computation resource for task node i and the sum of allocated computation resource for all task nodes cannot exceed F, respectively. Constraint (17i) represents the subcarrier pairing result, where task node i is either allocated subcarrier m or some other subcarrier. Constraint (17g) guarantees that each task node can be allocated at most one subcarrier. To reduce the implementation complexity and limit the interference on each subcarrier, constraint (17h) ensures that the maximum number of task nodes allocated to a subcarrier is not greater that $|\mathcal{I}_m|$.

B. Problem Analysis

Without loss of generality, we assume that the arriving tasks at the task node i are independent and identically distributed (i.i.d.) in different time slots. By exploiting the fact that the task scheduling constraints and subcarrier allocation constraints are separable, we propose an iterative algorithm in each time slot, which alternatively optimizes the task offloading at each task node assuming a fixed subcarrier allocation policy, and the subcarrier allocation assuming a given task scheduling policy.

Since the task offloading vector $\mathbf{a}$ and subcarrier allocation vector $\boldsymbol{\eta}$ are binary variables, finding the optimal solution for Problem (P1) is challenging. Moreover, the feasible set and the objective function of Problem (P1) are not convex, so it is NP hard. Due to dynamic channel conditions and dynamic computational cost for time-variant IIoT systems, instead of solving the NP hard Problem (P1) by utilizing conventional optimization methods, we solve the task scheduling and subcarrier allocation problem in an online learning fashion. During the online learning process, we propose an iterative algorithm to jointly optimize the subcarrier allocation and task scheduling in each time episode. In the iterative algorithm, we first solve the task scheduling subproblem by optimizing the power allocation $\mathbf{p}$, computational resource allocation $\mathbf{f}$. Subsequently, we solve the subcarrier allocation subproblem by proposing a subcarrier matching method to find the cost minimized variable $\boldsymbol{\eta}$. Based on the power, computational resource, and subcarrier allocations, we propose reinforcement learning (RL) algorithm to find optimized task offloading decisions $\mathbf{a}$.

IV. Online Task Scheduling and Subcarrier Allocation

A. Reinforcement Learning Settings

1) Key Elements for Online Learning: We formulate the task scheduling problem in (20) as an online learning problem to obtain the offloading decisions that minimize the total cost. The online learning method has three key elements, including state, action, and reward, as discussed as follows.

- State: In each time slot, the agent collects system state S, which is defined as the sum cost of the task offloading, i.e., $S = \sum_{i \in \mathcal{I}_m} \alpha_i \Omega_{i,m}^{(o)}$.
- Action: The agent selects the action according to the predefined strategy. The action is the offloading decision of task nodes. We denote $\mathbf{a}_m$ as the decision vector of task nodes at subcarrier m, i.e., $\mathbf{a}_m = [\alpha_1, \alpha_2, \cdots, \alpha_{|\mathcal{I}_m|}]$.
- Reward: After performing action $\mathbf{a}_m$, the agent gets a reward $R(S, \mathbf{a}_m)$ in state S, and the goal of online learning is to get the maximum reward. Therefore, the reward function is negatively correlated to the size of the sum cost Ω_{all}. Consequently, we define the immediate reward as

$$R(S, \mathbf{a}_m) = -\Omega_{\text{all}}(S, \mathbf{a}_m), \quad (18)$$

where $\Omega_{\text{all}}(S, \mathbf{a}_m)$ denotes the actual sum cost of current state S.

2) Action Selection: Q-learning [36] is a well-known model-free reinforcement learning (RL) algorithm that finds an estimate of the optimal action-value function. The action-value

Q can be regarded as a long-term reward, which is defined as

$$Q(S,\mathbf{a}_m)=R(S,\mathbf{a}_m)+\vartheta\mathsf{E}[\sum_{t=0}^{\infty}\vartheta^t R_t|S=S'],\qquad(19)$$

where R_t, S', and ϑ, $0\le\vartheta<1$, are the immediate reward at time slot t, the random next state on executing action $\mathbf{a}_m$ in state S and the discount factor, respectively. According to the Q-learning algorithm, the ϵ-greedy algorithm is utilized for the action selection [36]. It should be noted that threshold ϵ and random number ϕ are predetermined for ϵ-greedy algorithm, and ϕ is generated each time alot. Moreover, the ϵ-greedy algorithm consists of exploration and exploitation phases, which depends on the value of ϕ. In all, we describe the the details of action selection as below:

- If $\phi>\epsilon$, then the agent select the action according to exploitation, i.e., the agent selects the best action $\mathbf{a}'_m=\arg\max_{\mathbf{a}}Q(S,\mathbf{a}_m)$ according to the Q-values stored in the Q-matrix.
- Otherwise, the agent selects the action according to exploration, i.e., the agent selects a random action instead of utilizing local optimization.

3) Minimization Cost: In this subsection, we solve the task scheduling subproblem to obtain the power and computation resources allocation. We denote $\boldsymbol{\eta}^*$ as a given subcarrier assignment, task scheduling can be independently performed for each helper node. To simplify the notation, the task scheduling subproblem for the task nodes at given subcarrier m is given by

$$\min_{\mathbf{a},\mathbf{p},\mathbf{f}}\sum_{i\in\mathcal{I}_m}(1-\alpha_i)\Omega_i^{(l)}+\alpha_i\Omega_{i,m}^{(o)}\qquad(20a)$$

$$\text{s.t. }(17b),(17c),(17d),(17e),(17f).\qquad(20b)$$

For each time slot with selected action, we need to optimize the power and computation resource to obtain the system state. By denoting α_i^* as the given action for task node i, the power and computational resource allocation subproblem is given by

$$\min_{\mathbf{p},\mathbf{f}}\sum_{i\in\mathcal{I}_m}\alpha_i^*\Omega_{i,m}^{(o)}\qquad(21a)$$

$$\text{s.t. }(17e),(17f).\qquad(21b)$$

By analyzing subproblem (21), we respectively define $\Gamma(p_i,f_i)$ and $\Theta(p_i,f_i)$ as

$$\Gamma(p_i,f_i)=\alpha_i^*w_i^t(\frac{B_i}{\kappa_i\log_2\gamma_{i,m}+\upsilon_i}+\frac{C_i}{f_i}),\qquad(22)$$

and

$$\Theta(p_i,f_i)=\alpha_i^*w_i^e(\frac{p_iB_i}{\kappa_i\log_2\gamma_{i,m}+\upsilon_i}+\frac{p_i^iC_i}{f_i}),\qquad(23)$$

where $\kappa_i=\frac{\gamma'_{i,m}}{1+\gamma'_{i,m}}$, $\upsilon_i=\log_2(1+\gamma'_{i,m})-\frac{\gamma'_{i,m}}{1+\gamma'_{i,m}}\log_2\gamma'_{i,m}$, and $\gamma'_{i,m}$ is a random chosen variable. Then, we have the following theorem.

Theorem 1: Solving subproblem (21) *is equivalent to solving the following problem*

$$\min_{\boldsymbol{\rho},\mathbf{f}}\sum_{i\in\mathcal{I}_m}\Gamma(\rho_i,f_i)+\Theta(\rho_i,f_i)\qquad(24a)$$

$$\text{s.t. }(17e),(17f),\qquad(24b)$$

where $\rho_i=\log_2 p_i$ *and* $\boldsymbol{\rho}=\{\rho_1,\cdots,\rho_{|\mathcal{I}|}\}$.

Proof: Please refer to Appendix -A. ■

Corollary 1: The equivalent task scheduling Problem (24) *is a convex optimization problem with respect to* ρ_i *and* f_i, $\forall i\in\mathcal{I}_m$.

Proof: Please refer to Appendix -B. ■

Based on Theorem 1, Problem (24) is a standard convex optimization problem, which can be solved by many algorithms (e.g., subgradient and interiorpoint methods) to obtain the optimal solution. In order to efficiently solve Problem (24), we iteratively update the solution of the power and computational resource allocations p_i and f_i. The obtained power and computational resource allocations for each iteration tighten the upper-bound in (34) until convergence. Therefore, the value of the total cost decreases after each iteration in the proposed power and computation resource allocation (PCRA) algorithm, as shown in Algorithm 1. Note that Algorithm 1 always converges and outputs the optimized power and computational resource allocation results p_i^* and f_i^*, $\forall i\in\mathcal{I}_m$.

Algorithm 1 PCRA Algorithm

```
Step 1: Initialize $n = 0$ and the maximum number of iterations $I_{\max}$.
Initialize power and computation resource allocation vectors, $\mathbf{p}$ and $\mathbf{f}$, and then calculate $\gamma'_{i,m}(0)$.
Step 2: Set the convergence threshold $\iota$.
while $|\gamma'_{i,m}(n)-\gamma'_{i,m}(n-1)|\ge\iota$ do
  $n = n + 1$.
  Set $\gamma'_{i,m}(n)=\gamma_{i,m}(n-1)$, and compute $\kappa_i$, $\upsilon_i$.
  Solve the optimization problem in (24) and obtain the solutions as $\boldsymbol{\rho}(n)$ and $\mathbf{f}(n)$.
  Update $\mathbf{p}$, where $\rho_i(n)=\log_2 p_i(n)$.
  Calculate $\gamma_{i,m}(n)$ based on $p_i(n)$.
end while
Step 3: Output: $p_i^*=p_i(n)$, $f_i^*=f_i(n)$, $\forall i\in\mathcal{I}_m$.
```

B. Subcarrier Allocation via Matching

After solving the task scheduling subproblem, we in this section aim to tackle the subcarrier allocation subproblem for given computation and power allocation decisions. We apply matching theory to solve the subcarrier allocation problem, which is motivated by its ability to tackle combinatorial optimization problems and result in a distributed solution. Moreover, matching theory allows each player, i.e., task node and subcarrier, to define its individual utility according to its local information. The matching subproblem is given by

$$\min_{\boldsymbol{\eta}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{I}}(1-\alpha_i^*)\Omega_i^{(l)}+\alpha_i^*\Omega_{i,m}^{(o)}\qquad(25a)$$

$$\text{s.t. }(17i),(17g),(17h).\qquad(25b)$$

Due to a large set of task nodes and subcarriers in a practical amount of time [37] and combinatorial nature of problem (25), finding the optimal solution is NP-hard. In order to solve Problem (25) with low-complexity, we transfer Problem (25) into an equivalent matching problem. Subsequently, we discuss the details of the matching algorithm in the following subsections.

1) Matching Game Formulation: In order to proceed with the two-sided matching game, we first introduce some definitions. We consider the two sets of subcarriers and helper nodes as two sets of players. The matching is defined as an assignment of subcarriers in $\mathcal{M}$ to task nodes in $\mathcal{I}$. Since the subcarrier assignment game is a two-sided one-to-many matching process between the sets of subcarriers and task nodes, our design corresponds to a one-to-many matching given by the tuple $(\mathcal{I}, \mathcal{M}, \succ_{\mathcal{I}}, \succ_{\mathcal{M}})$, where $\succ_{\mathcal{I}} \triangleq \{\succ_i\}_{i\in\mathcal{I}}$ and $\succ_{\mathcal{M}} \triangleq \{\succ_m\}_{m\in\mathcal{M}}$ represent the set of preference relations of task nodes and subcarriers, respectively. Then, we define the matching as follows.

Definition 1: A matching μ is defined by a function on the set $\mathcal{I} \cup \mathcal{M}$ such that $\mathcal{I}_m = \mu(m)$ and $m = \mu(i)$, $i \in \mathcal{I}_m$.
Here, $\mu(i) = \emptyset$ means that task node i is not matched to any subcarrier. Similarly, if $\mu(m) = \emptyset$, then there are no task nodes matched to subcarrier m.

The players make their decisions based on their individual preferences (e.g., total cost). With these preferences, the players can rank one another. This process does not require the players to know each others' preferences, and hence can be implemented in a distributed manner. Therefore, the two sets of task nodes and subcarriers build their preferences based on the local information features (e.g., interference). For any task node $i \in \mathcal{I}$, the preference profile is based on the achievable total cost on subcarrier m. In other words, the preference function can be expressed as

$$U_i(m) = (1-\alpha_i^*)\Omega_i^{(l)} + \alpha_i^*\Omega_{i,m}^{(o)}. \tag{26}$$

It should be noted that each task node may change its preference order due to the changeable task nodes allocated with the same subcarrier, which can be solved with a strict rank order [38], [39].

The task nodes simply rank the subcarriers based on the total cost. Note that, in Problem (25) it is desirable to minimize the total cost for each task node. Specifically, we consider that each task node ranks all the subcarriers in $\mathcal{M}$ in a non-increasing order. Hereinafter, the preference of task node i is denoted as $\mathcal{P}_i$. Using the preference in (26), a subcarrier $m \in \mathcal{M}$ that corresponds to a smaller cost is preferred over another subcarrier $m' \in \mathcal{M}$ by a task node i, i.e., $m \succ_i m'$ and then subcarrier m is placed higher in its preference list. After building preference lists, task node i takes actions based on the local preference.

The goal of subcarrier allocation is to enhance the transmission quality of the task nodes. In fact, a group of task nodes create more interference using a subcarrier, which gives less utility to the system. To ensure that the interference generated by task nodes is less than a tolerable interference threshold $\zeta_{\max}^m$, subcarrier m prefers to choose a subset $\mathcal{I}_m$ of task nodes. Thus for the subcarriers, one can build preferences based on the interference in (5). The preference function is defined as follows

$$U_m(\mathcal{I}, \mu) = \max_v\{|\mathcal{I}_v| : \zeta_{\mathcal{I}_v,m} \le \zeta_{\max}^m\}. \tag{27}$$

In order to obtain the preference list, the required information includes power p_i of task node $i \in \mathcal{I}$, the predefined maximum interference threshold $\zeta_{\max}^m$ for each subcarrier, and the subcarrier power gain between task node i and busy node m as $d_{i,m}|h_{i,m}|^2$. With the subcarrier power gain $d_{i,m}|h_{i,m}|^2$, the interference $\zeta_{i,m}$ caused by each task node i can be calculated. Then, each subcarrier can rank all the task nodes in $\mathcal{I}$ according to the preference function in (27). Additionally, task nodes that violate the interference tolerance receive a zero utility and are ranked as the lowest in the preference list of subcarrier m.

Based on this preference function, subcarrier m chooses a subset of task nodes under the condition that the interference caused by this subset is less than the tolerable interference threshold $\zeta_{\max}^m$. Thus, we obtain the subset of task nodes with the largest number of elements among all the feasible subsets in $\mathcal{I}$ based on the preference function. It is worth noting that subcarrier m prefers to choose the subset of task nodes causing the lowest interference. Denote $\succ_m$ as the preference relation for any subcarrier m, which can be defined as follows

$$(\mathcal{I}', \mu) \succ_m (\hat{\mathcal{I}}, \mu') \Leftrightarrow U_m(\mathcal{I}', \mu) > U_m(\hat{\mathcal{I}}, \mu'), \forall \mathcal{I}', \hat{\mathcal{I}} \subset \mathcal{I}, \tag{28}$$

where $\mathcal{I}' \neq \hat{\mathcal{I}}$, and $\mathcal{I}' = \mu(m)$, $\hat{\mathcal{I}} = \mu'(m)$.

2) Subcarrier Allocation Algorithm: According to the matching algorithm in [40], we propose a matching game algorithm, which exploits the local information and the preference of each player to solve the subcarrier allocation problem. The matching game algorithm is able to find a stable subcarrier allocation, which is defined as follows.

Definition 2: A matching μ is stable if there exists no blocking pair (i, m), $\forall\, i, i' \in \mathcal{I}$, $m \in \mathcal{M}$, satisfies

- $\zeta_{\text{res}}^m \ge \zeta_{i,m}$, $i \succ_m \emptyset$, $m \succ_i \mu(i)$, $\mu(m) \notin \mathcal{R}_i$,
- $\zeta_{\text{res}}^m < \zeta_{i,m}$, $\zeta_{\text{res}}^m + \sum_{i'\in\mu(m)} \zeta_{i',m} \ge \zeta_{i,m}$, $i \succ_m i'$, $m \succ_i \mu(i)$, $\mu(m) \notin \mathcal{R}_i$,

where $\zeta_{\text{res}}^m = \zeta_{\max}^m - \sum_{i\in\mu(m)} \zeta_{i,m}$ and $\mathcal{R}_i$ represent the remaining interference tolerance on subcarrier m and the conflict set of task node i, respectvely.
The conflict set of task node i represents all the interfering task nodes such that the required SINR for task node i is below a threshold, which can be expressed as

$$\mathcal{R}_i = \left\{ i' \in \mathcal{I} : \frac{p_i d_{i,m}|h_{i,m}|^2}{p_{i'} d_{i',i}|h_{i',i}|^2} \le \zeta_{i,m} \right\}. \tag{29}$$

The main purpose of constituting the conflict set is to restrict the subcarrier reuse. Under the condition that $\zeta_{\text{res}}^m < \zeta_{i,m}$, the quota of subcarrier m has no room. For a blocking pair, task node i and subcarrier m deviate from their assigned matching. According to the definition of the blocking pair [40], either subcarrier m has sufficiently large interference tolerance ζ_{res}^m and is willing to accept task node i or its quota has no room but it is able to match task node i by removing the existing accepted task nodes ranked lower than task node i. Thus, stable matching makes sure that there exists no blocking pairs.

For subcarrier m and new subcarrier m', we note that a stable solution in our matching game ensures that no

Algorithm 2 Subcarrier Allocation Algorithm

```
1: Step 1: Network Initialization:
2: Preferences Specification: $t' = 1$, $\mu_{t'} \triangleq \{\mu(i)_{t'}, \mu(m)_{t'}\}_{i\in\mathcal{I}, m\in\mathcal{M}} = \varnothing$, $\Xi_{m,t'} = \varnothing$, $\mathcal{P}_{i,t'} = \mathcal{P}_i$, $\mathcal{P}_{m,t'} = \mathcal{P}_m$, $\zeta_{\max}^m$, $\forall m, i$.
3: Step 2: Subcarriers Matching:
4: repeat
5:   $t' = t' + 1$.
6:   while $i \notin \mu(m)_{t'}$, and $\mathcal{P}_{i,t'} \neq \emptyset$ do
7:     if $\zeta_{\max}^m < \zeta_i^m$ then
8:       if $i \succ_m \mu(m)_{t'}$ then
9:         $\mu(m)_{t'} \leftarrow \mu(m)_{t'} \setminus i'$.
10:        $\mu(m)_{t'} \leftarrow i$.
11:        $\hat{\mathcal{P}}_{m,t'} = \{i' \in \mu(m)_{t'} | i \succ_m i'\}$.
12:      else
13:        $\tilde{\mathcal{P}}_{m,t'} = \{i \in \mathcal{I} | \mu(m)_{t'} \succ_m i\}$.
14:      end if
15:    else
16:      $\bar{\mathcal{P}}_{m,t'} = \{i \in \mathcal{I} | \zeta_{\max}^m \geq \zeta_i^m\}$.
17:      $\Xi_{m,t'} = \{\hat{\mathcal{P}}_{m,t'}\} \cup \{\tilde{\mathcal{P}}_{m,t'}\}$.
18:      for $u \in \Xi_{m,t'}$ do
19:        $\mathcal{P}_{u,t'} \leftarrow \mathcal{P}_{u,t'} \setminus \{m\}$.
20:        $\mathcal{P}_{m,t'} \leftarrow \mathcal{P}_{m,t'} \setminus \{u\}$.
21:      end for
22:    end if
23:  end while
24: until $\mu_{t'} = \mu_{t'-1}$, convergence to a stable matching.
```

matched task node would benefit from deviating from their assigned subcarrier. As such, the matching algorithm output of minimizing the objective of the optimization Problem (25) is the subcarrier allocation vector $\boldsymbol{\eta}$ for all task nodes. According to the well-known deferred acceptance algorithm [38], the presented matching Algorithm 2 is guaranteed to converge to a stable allocation, which will be shown in the next subsection.

3) Stability and Convergence: A basic matching algorithm for a matching problem consists of the initialization phase, the matching phase, and the subcarrier allocation phase. For the initialization phase in Step 1 of Algorithm 2, the task nodes and subcarriers build their preferences based on the local information including the cost and interference that each player acquired. Using the initial preferences $\mathcal{P}_{i,1}$, each task node without subcarrier allocation prefers to select its cost minimized subcarrier m, while the central controller calculates interference $\zeta_{i,m}$.

In Step 2 of Algorithm 2, under the condition of violating the interference tolerance, task node i is removed. Otherwise, the preference list of subcarrier m will be updated according to the preference ranking. Subsequently, for updating their preference profiles, the set $\Xi_{m,t'}$ representing all the rejected task nodes at iteration t' is removed by both sides. As such, the iterative subcarrier allocation is carried out until a stable match is found for both the task nodes and subcarriers. This process is terminated under two conditions, one is that all the task nodes maintaining the interference tolerance level

Algorithm 3 JSATS Algorithm

```
Step 1: Initialize time step $t = 0$ and the maximum number of iterations $N_{\max}$.
Initialize subcarrier allocation result, $\boldsymbol{\eta}_0$.
Step 2: Set the convergence threshold $\iota$.
while $\|\mathbf{p}_t - \mathbf{p}_{t-1}\| \geq \iota$ do
  $t = t + 1$.
  Task Scheduling: Run Algorithm 1 to obtain $\mathbf{p}_t$ and $\mathbf{f}_t$.
  Subcarrier Allocation: Run Algorithm 2 to obtain $\boldsymbol{\eta}_t$ based on the power allocation $\mathbf{p}_t$.
end while
Step 3: Obtain the joint subcarrier allocation and task scheduling results.
```

have been assigned to subcarriers, and the other is that no more subcarriers are remained to realize subcarrier matching. Finally, the matched task nodes offload their tasks on the matched subcarriers. By analyzing the subcarriers matching algorithm, we have the following theorem.

Theorem 2: The subcarriers allocation matching algorithm converges to a pairwise stable matching.

Proof: Please refer to Appendix -C. ■

C. Joint Subcarrier Allocation and Task Scheduling

Based on the power and computation resources allocation algorithm and subcarrier allocation algorithm, it is worth considering how to jointly consider subcarrier allocation and task scheduling. Thus, an iterative joint subcarrier allocation and task scheduling algorithm (JSATS) is presented in the following.

According to the optimization problem in (17), the SIC order for each task node depends on the subcarrier allocation. Thus, the power allocation cannot be achieved if we do not know the subcarrier allocation. To propose the optimization algorithm with a low complexity, we first solve the task scheduling problem by optimizing the task, power, and computation resource allocations based on the random given initial subcarrier allocation results. After the convergence of Algorithm 1, the subcarrier can be allocated via matching Algorithm 2 based on the power allocation $\mathbf{p}$, where the preference of task nodes is replaced by the achievable data rate as (6). The iterative joint subcarrier allocation and task scheduling algorithm can be shown as Algorithm 3.

D. Reinforcement Learning Algorithm

1) Q-Network Update: Based on the above analytic results, we can obtain the joint subcarrier allocation and task scheduling results via Algorithm 3 in each time episode. Next, we introduce our RL algorithm. Denote $\mathcal{S}$ as a set of states and τ $(\tau = 1, \cdots, K_\tau)$ as the decision times such that $S_\tau \in \mathcal{S}$, meaning that the agent's state at time τ is S_τ, which can be obtained by Algorithm 3. At the same time, the reward function is based on the observation after performing action $\mathcal{A}_{m,\tau}$ by the agent when in state S_τ. Then, function Q is updated at the next decision time $\tau + 1$, which is given by

$$\begin{aligned} &Q(S_{\tau+1}, \mathbf{a}_{\tau+1}) \\ &\quad = Q(S_\tau, \mathbf{a}_\tau) + \delta \left(TQ(S_{\tau-1}, \mathbf{a}_{\tau-1}) - Q(S_\tau, \mathbf{a}_\tau)\right) \\ &\qquad + (1-\delta)\left(TQ(S_\tau, \mathbf{a}_\tau) - TQ(S_{\tau-1}, \mathbf{a}_{\tau-1})\right), \end{aligned} \tag{30}$$

where δ and T are the learning step and the Bellman optimality operator, respectively. According to [36], the Bellman optimality operator is defined as

$$TQ(S, \mathbf{a}_m) = R(S, \mathbf{a}_m) + \vartheta \sum_{S' \in \mathcal{S}} P(S'|S, \mathbf{a}_m) \max_{\mathbf{a}'_m} Q(S', \mathbf{a}'_m), \tag{31}$$

where $P(S'|S, \mathbf{a}_m)$ denotes the probability distribution of the next state S'. At each step τ and for state-action pair $(S, \mathbf{a}_m)$, the online speedy Q-learning algorithm works as follows

Step 1: Based on $P(\cdot|S, \mathbf{a}_m)$, it generates the next state S'_τ.

Step 2: After applying the Bellman optimality operator to the Q function, it estimates and calculates $TQ(S_{\tau-1}, \mathbf{a}_{\tau-1,m})$ and $TQ(S_\tau, \mathbf{a}_{\tau,m})$ for state-action pair $(S, \mathbf{a}_m)$.

Step 3: It updates the action-value function of $(S, \mathbf{a}_m)$ based on the update rule of (30) and generates $Q(S_{\tau+1}, \mathbf{a}_{\tau+1,m})$.

Moreover, in order to apply the speedy Q-learning algorithm, we let δ_τ decay linearly with time, i.e., $\delta_\tau = 1/(\tau+1)$. As a result, we can rewrite the update rule of (30) as follows

$$\begin{aligned} Q(S_{\tau+1}, \mathbf{a}_{\tau+1,m}) = &(1-\delta_\tau) Q(S_\tau, \mathbf{a}_{\tau,m}) \\ &+ \delta_\tau \phi[Q_\tau, Q_{\tau-1}](S, \mathbf{a}_m), \end{aligned} \tag{32}$$

where $\phi[Q_\tau, Q_{\tau-1}](S, \mathbf{a}_m) = \tau TQ(S_\tau, \mathbf{a}_{\tau,m}) - (\tau - 1) TQ(S_{\tau-1}, \mathbf{a}_{\tau-1,m})$.

According to the speedy Q-learning algorithm, the Q-learning based task scheduling algorithm is presented in Algorithm 4. The algorithm consists of three main parts: action selection, next state observation, and reward calculation, and Q-network updating. Then, for each time step, the task node selects $\mathbf{a}_{m,\tau}$ according to the predefined strategy, and observes the next state $S_{\tau+1}$. The reward R_τ is calculated based on (18). Meanwhile, the Q values are stored in the Q-network, where the replay memory $\mathcal{J}$ is an essential part of Algorithm 4. After updating, the algorithm outputs $\mathbf{a}_{m,\tau}$. Note that the action selection and the update steps are two main parts of the algorithm.

2) Convergence Analysis: The convergence of Algorithm 4 is discussed in the following theorem.

Theorem 3: The online-learning based task scheduling algorithm converges to the optimal policy $\mathbf{a}^*_m$ under the conditions that $\sum_\tau \delta_\tau(S, \mathbf{a}_m) = \infty$ and $\sum_\tau \delta^2_\tau(S, \mathbf{a}_m) \leq \infty$, where $\delta_\tau(S, \mathbf{a}_m)$ represents the learning rate during time slot T_τ.

For the efficient Q-learning based task allocation algorithm, [41] has provided the asymptotic convergence rate for Q-learning and proved that the optimal policy can be gradually achieved under the stationary markov decision process (MDP) system with a sufficiently small learning rate [41]. In summary, the Q-learning based task scheduling Algorithm 4 is convergent, and we will demonstrate the availability and effectiveness of the learning algorithm in Section V via simulations.

Algorithm 4 Q-Learning Algorithm

```
1: Initialize $Q(S, \mathbf{a}_m)$.
2: for episode=1,N do
3:   Choose a random state $S$.
4:   for $\tau = 1,\ K_\tau$ do
5:     Initialize $\phi$
6:     if $\phi > \epsilon$ then
7:       Set $\mathbf{a}_{m,\tau} = \arg\max_{\mathbf{a}_m} Q(S_\tau, \mathbf{a}_m)$, and compute $\kappa_i$, $\upsilon_i$.
8:       Solve the convex optimization problem in (24) and obtain the solutions as $\boldsymbol{\rho}(n)$ and $\mathbf{f}(n)$ based on Algorithm 1.
9:       Joint subcarrier allocation and task scheduling based on Algorithm 3.
10:    else
11:      The action is selected by exploration, select random $\mathbf{a}_{m,\tau}$ from all possible actions of state $S$, and compute $\kappa_i$, $\upsilon_i$.
12:      Solve the convex optimization problem in (24) and obtain the solutions as $\boldsymbol{\rho}(n)$ and $\mathbf{f}(n)$ based on Algorithm 1.
13:      Joint subcarrier allocation and task scheduling based on Algorithm 3.
14:    end if
15:    Execute chosen $\mathbf{a}_{m,\tau}$, and observe $S_{\tau+1}$.
16:    Calculate $R_\tau$ by (18).
17:    Store transition $(S_\tau, \mathbf{a}_\tau, R_\tau, S_{\tau+1})$ in $\mathcal{J}$.
18:    Sample random $\mathcal{J}_r \subset \mathcal{J}$.
19:    for $(S^r_\tau, \mathbf{a}^r_\tau, R^r_\tau, S^r_{\tau+1})$ in $\mathcal{J}_r$ do
20:      Calculate the output of $Q$-learning network with weight $\theta$: $Q(S^r_\tau, \mathbf{a}^r_\tau; \theta)$.
21:      Set $y_{\tau,r} = (1-\delta_\tau)Q(S^r_\tau, \mathbf{a}^r_\tau; \theta) + \delta_\tau \tau TQ(S^r_\tau, \mathbf{a}^r_\tau; \theta) - (\tau-1)TQ(S^r_{\tau-1}, \mathbf{a}^r_{\tau-1}; \theta)$.
22:      Calculate $e_{\tau,r} = |y_{\tau,r} - Q(S^r_\tau, \mathbf{a}^r_\tau; \theta)|$.
23:    end for
24:    Set $\theta = \arg\min_\theta E[e^2_{\tau,r}]$.
25:    Output $\mathbf{a}_{m,\tau}$.
26:  end for
27: end for
```

V. Performance Evaluation

In this section, we evaluate the performance of the proposed online-learning based task scheduling and resource allocation algorithm in NOMA-based FC networks by simulations.

A. Parameter Settings

Unless specified otherwise, the simulation parameters are set as follows. There are 10 helper nodes with abundant resources located near the task nodes. The variance of AWGN is set to 10^{-9} W. For each task node and helper node, the CPU computation capacity are randomly selected from

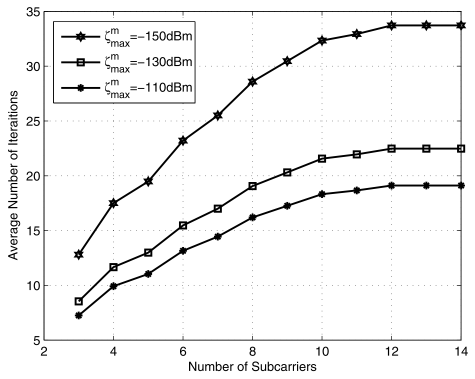


Fig. 2. Average number of iterations versus the number of subcarriers for different interference tolerance levels.

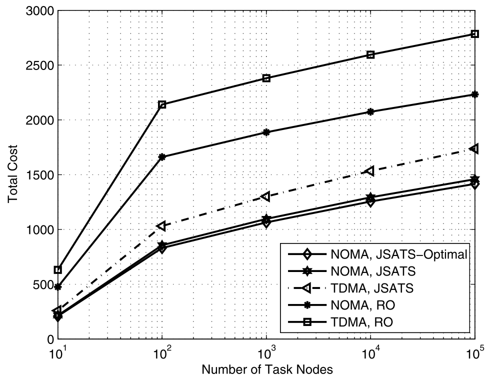


Fig. 3. The total cost versus the number of task nodes for both NOMA and TDMA.

the set $\{0.1, 0.2, \cdots, 1.0\}$ GHz, the Random Access Memory (RAM) size is 2 GB, and the local computation energy per CPU cycle z_i follows a uniform distribution in the range of $(0, 20 \times 10^{-11})$ J/cycle. For the computing task, we consider the robot mapping application similar to [6], [42], where the task size for the computation offloading is $B_i = 5000$ KB, $\forall i \in \mathcal{I}$, the SINR threshold is 1.5 dB, and the required number of CPU cycles per bit follows the uniform distribution in $[500, 1500]$ cycles/bit. The power consumption of the idle state $p_i^i = 60$ mW.

B. Convergence of the Proposed Resource Allocation Algorithm

Fig. 2 evaluates the average number of iterations versus the number of subcarriers for three different interference tolerance thresholds. It can be observed that for a loose interference tolerance threshold (e.g., $\zeta^m_{\max} = -110$ dBm), the proposed subcarrier assignment algorithm has a remarkable convergence time. This is due to the fact that most of the subcarriers are accepted at their initial proposals when the tolerance threshold level is loose. Meanwhile, the average number of iterations increases with the number of available subcarriers. This is because the blocking probability increases with the number of subcarriers. On the other hand, for a tighter interference tolerance threshold (e.g., $\zeta^m_{\max} = -130$ dBm), the average number of iterations increases as more helper nodes will be initially rejected due to the tighter interference constraint.

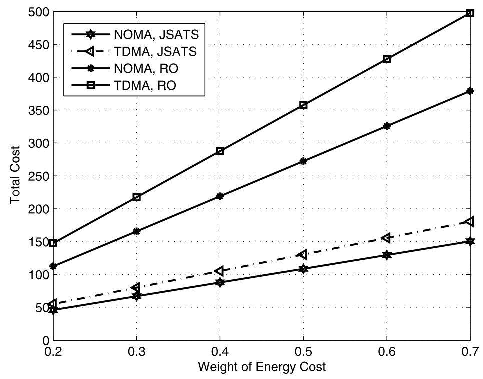


Fig. 4. The total cost versus the weight of energy cost, i.e., w_i^e, when the total number of task nodes is 10.

C. Performance of Optimal Task Scheduling Algorithm

Fig. 3 shows the total cost of task scheduling system versus different number of task nodes. We compare the performance of NOMA and TDMA strategies under different conditions. JSATS, JSATS-Optimal and random offloading (RO) represent the proposed task scheduling, proposed task scheduling with optimal subcarrier matching and random task scheduling, respectively. From this figure, we observe that the stable matching with JSATS based algorithm converges to the near optimal solution with JSATS-Optimal algorithm. We observe that JSATS always outperforms RO for different number of helper nodes. Furthermore, compared with the TDMA task scheduling strategy, we observe that the NOMA strategy can reduce the total cost with the same task scheduling strategy, which coincides with the analytic results. Meanwhile, it is worth noting that with the increase of the number of task nodes, the total cost is increasing. This is due to the fact that the computation resource is limited at the helper nodes, and the task should be computed locally with the increasing number of task nodes. In addition, it can also be observed that the total cost of NOMA is larger than that of the TDMA while the optimal task scheduling strategy JSATS is utilized in TDMA system, which is due to the dominance of the task scheduling cost in comparison to transmission strategy.

For the comparison of different task scheduling strategies in terms of the total cost, we also conduct the numerical studies to investigate the impact of the weight of energy cost. Fig. 4 shows the total cost versus the weight of energy cost of each task node. We observe that the total cost increases with w_i^e, which is due to the dominance of the energy cost. Similarly, it can be seen from Fig. 4 that the JSATS strategy leads to a lower total cost than the RO strategy for different number of helper nodes.

In Fig. 5, we plot the total cost of the NOMA and TDMA scheduling strategies versus the size of the computation task at each task node. It can be observed that the JSATS strategy in NOMA system always performs better for different sizes of computation task. The total cost increases with the size of the computation task. This is mainly due to the fact that the task size usually plays a important role in scheduling strategy, and this in turn is due to the fact that with a larger task size, the task node tends to offload a larger quantity of tasks to the helper nodes in order to decrease the total cost.

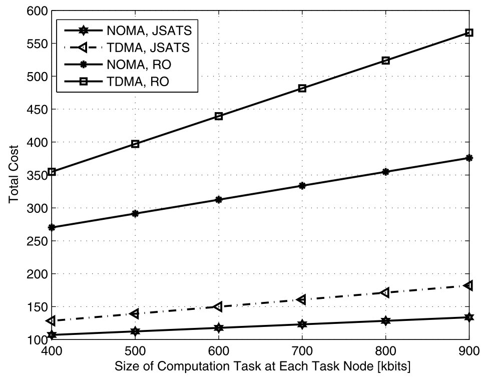


Fig. 5. The total cost versus the size of computation task at each task node when the number of task nodes is 10.

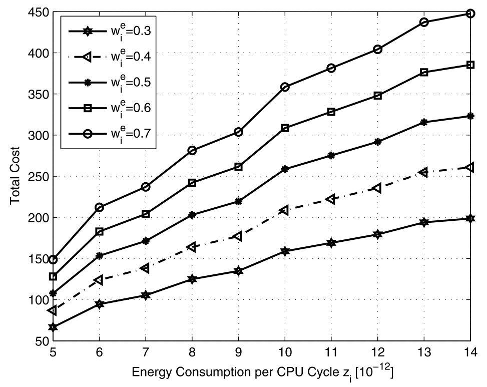


Fig. 6. The total cost versus the energy consumption per CPU cycle z_i, $\forall i \in \mathcal{I}$, when the number of task nodes is 10.

The total cost with an increasing energy consumption per CPU cycle z_i is shown in Fig. 6. Here we only consider the JSATS strategy in NOMA system for comparison, and the energy consumption per CPU cycle for each task node has the same value. From Fig. 6, we can observe that the total cost increases as the energy consumption per CPU cycle grows. We can also observe that the total cost is larger at larger region of weight for energy cost with different energy consumptions per CPU cycle. This is because as the weight for the energy cost increases, the energy cost increases at large region of energy consumption per CPU cycle, which leads to a larger total cost. This phenomenon is similar to the effect of dominance place of energy cost.

VI. Conclusion

In this paper, we proposed an NOMA-based FC framework for IIoT systems, where multiple task nodes perform task offloading via NOMA to nearby helper nodes. The FC networks share the spectrum of licensed cellular networks in the underlay mode. We formulated an optimization problem to minimize the sum cost of delay and energy consumption for all task nodes, which is non-convex and NP-hard. In order to tackle this challenging problem and obtain the optimal task scheduling and subcarrier allocation strategy, we presented an online learning-based optimization framework to optimize the subcarrier allocation and task scheduling. The simulation results showed that the proposed scheme achieves better performance than the baselines under various system settings. For future work, we will consider an arbitrary delay requirement in the proposed framework.

Appendix

A. Proof of Theorem 1

Since the upper-bound of the transmission rate can be written as a linear function of $\log_2 \gamma_{i,m}$ (see [43] for details), we obtain the following inequality at a chosen variable $\gamma'_{i,m}$ as

$$\log_2(1+\gamma_{i,m}) \geq \kappa_i \log_2 \gamma_{i,m} + \upsilon_i, \tag{33}$$

where κ_i and υ_i are respectively defined as $\kappa_i = \frac{\gamma'_{i,m}}{1+\gamma'_{i,m}}$, $\upsilon_i = \log_2(1+\gamma'_{i,m}) - \frac{\gamma'_{i,m}}{1+\gamma'_{i,m}} \log_2 \gamma'_{i,m}$. It is not hard to obtain that $\log_2(1+\gamma_{i,m}) = \kappa_i \log_2 \gamma_{i,m} + \upsilon_i$ is satisfied when $\gamma_{i,m} = \gamma'_{i,m}$.

Based on the inequality in (33), the upper bound of the objective function in (21) can be written as

$$\begin{aligned}\alpha_i^* \Omega_i^o &= \alpha_i^* w_i^t \left(\frac{B_i}{r_{i,m}} + \frac{C_i}{f_i}\right) + \alpha_i^* w_i^e \left(\frac{p_i B_i}{r_{i,m}} + \frac{p_i^i C_i}{f_i}\right) \\ &\leq \Gamma(p_i, f_i) + \Theta(p_i, f_i),\end{aligned} \tag{34}$$

where $\Gamma(p_i, f_i)$ and $\Theta(p_i, f_i)$ are respectively defined as

$$\Gamma(p_i, f_i) = \alpha_i^* w_i^t \left(\frac{B_i}{\kappa_i \log_2 \gamma_{i,m} + \upsilon_i} + \frac{C_i}{f_i}\right), \tag{35}$$

and

$$\Theta(p_i, f_i) = \alpha_i^* w_i^e \left(\frac{p_i B_i}{\kappa_i \log_2 \gamma_{i,m} + \upsilon_i} + \frac{p_i^i C_i}{f_i}\right). \tag{36}$$

We define $\rho_i = \log_2 p_i$ and denote $\boldsymbol{\rho} = \{\rho_1, \cdots, \rho_{|\mathcal{I}|}\}$. The power and computation resource allocation that minimize the total cost can be found by solving the following problem

$$\min_{\boldsymbol{\rho}, \mathbf{f}} \sum_{i \in \mathcal{I}_m} \Gamma(\rho_i, f_i) + \Theta(\rho_i, f_i) \tag{37a}$$

$$\text{s.t. } (17\text{e}), (17\text{f}). \tag{37b}$$

B. Proof of Corollary 1

Based on the definitions of $\Gamma(\rho_i, f_i)$ and $\Theta(\rho_i, f_i)$, we have

$$\Gamma(\rho_i, f_i) = \alpha_i^* w_i^t \left(\frac{B_i}{\kappa_i \rho_i + \kappa_i \Phi_{i,m} + \upsilon_i} + \frac{C_i}{f_i}\right), \tag{38}$$

$$\Theta(\rho_i, f_i) = \alpha_i^* w_i^e \left(\frac{2^{\rho_i} B_i}{\kappa_i \rho_i + \kappa_i \Phi_{i,m} + \upsilon_i} + \frac{p_i^i C_i}{f_i}\right), \tag{39}$$

where $\Phi_{i,m} = \log_2 \frac{d_{i,m}|h_{i,m}|^2}{\zeta_{i,m}^{\text{in}} + \zeta_m^{\text{out}} + \sigma_m^2}$. By taking the derivatives of $\Gamma(\rho_i, f_i)$ and $\Theta(\rho_i, f_i)$ with respect to ρ_i and f_i, we have

$$\frac{\partial^2 \Gamma(\rho_i, f_i)}{\partial \rho_i^2} = \frac{2\kappa_i^2 \alpha_i^* w_i^t B_i}{(\kappa_i \rho_i + \kappa_i \Phi_{i,m} + \upsilon_i)^3} \geq 0, \tag{40a}$$

$$\frac{\partial^2 \Gamma(\rho_i, f_i)}{\partial f_i^2} = \frac{2\alpha_i^* w_i^t C_i}{f_i^3} \geq 0, \tag{40b}$$

$$\frac{\partial^2 \Theta(\rho_i, f_i)}{\partial f_i^2} = \frac{2\alpha_i^* w_i^e p_i^i C_i}{f_i^3} \geq 0, \tag{40c}$$

$$\begin{aligned}\frac{\partial^2\Theta(\rho_i,f_i)}{\partial\rho_i^2} &= \alpha_i^* w_i^e 2^{\rho_i} B_i \left[\frac{2\kappa_i^2}{(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)^3}\right.\\ &\quad \left. -\frac{2\kappa_i\ln 2}{(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)^2}+\frac{\ln^2 2}{\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i}\right]\\ &= \frac{\alpha_i^* w_i^e 2^{\rho_i} B_i}{(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)^3}\left[2\kappa_i^2+\ln^2 2(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)^2\right.\\ &\quad \left. -2\kappa_i\ln 2(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)\right]\\ &= \frac{\alpha_i^* w_i^e 2^{\rho_i} B_i}{(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)^3}\left[(\ln 2(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)\right.\\ &\quad \left. -\sqrt{2}\kappa_i\Big)^2+\kappa_i\ln 2(\kappa_i\rho_i+\kappa_i\Phi_{i,m}+\upsilon_i)(2\sqrt{2}-2)\right]\geq 0.\end{aligned} \tag{41}$$

Since (40a), (40b), (40c), and (41) are positive, we obtain the Hessian

$$\mathbf{H}_i=\mathrm{diag}\left(\frac{\partial^2\Gamma(\rho_\mathrm{i},\mathrm{f_i})}{\partial\rho_\mathrm{i}^2}+\frac{\partial^2\Theta(\rho_\mathrm{i},\mathrm{f_i})}{\partial\rho_\mathrm{i}^2},\frac{\partial^2\Gamma(\rho_\mathrm{i},\mathrm{f_i})}{\partial\mathrm{f_i}^2}+\frac{\partial^2\Theta(\rho_\mathrm{i},\mathrm{f_i})}{\partial\mathrm{f_i}^2}\right). \tag{42}$$

For all vectors $\mathbf{b}=[b_1,b_2]^T$, matrix $\mathbf{H}_i$ is positive semidefinite, i.e.,

$$\mathbf{b}^T\mathbf{H}_i\mathbf{b}=b_1^2\left[\frac{\partial^2\Gamma(\rho_i,f_i)}{\partial\rho_i^2}+\frac{\partial^2\Theta(\rho_i,f_i)}{\partial\rho_i^2}\right]+b_2^2\left[\frac{\partial^2\Theta(\rho_i,f_i)}{\partial f_i^2}\right]\geq 0. \tag{43}$$

Therefore, $\sum_{i\in\mathcal{I}_m}\Gamma(\rho_i,f_i)+\Theta(\rho_i,f_i)$ is a convex function with respect to $\boldsymbol{\rho}$ and $\mathbf{f}$, and its Hessian is a positive semidefinite block diagonal matrix $\mathrm{diag}(\mathbf{H}_1,\mathbf{H}_2,\cdots,\mathbf{H}_{|\mathcal{I}_m|})$. We conclude that the problem in (24) is a standard convex optimization problem.

C. Proof of Theorem 2

This theorem can be proved by contradiction. Suppose that there exist a task node $\bar{i}$ that $\bar{i}\notin\mu(i)$ such that $\zeta_{\bar{i},m}<\zeta_{\mathrm{res}}^m$ and $\bar{i}\succ_m i$. Then, task node $\bar{i}$ is in the preference list of the subcarrier m. Therefore, μ is not stable. Due to limited spectrum resource, the total cost has a lower bound. Therefore, the matching process converges to a stable matching within a limited number of iterations.

References

[1] K. Wang, Y. Zhou, Y. Yang, X. Yuan, and X. Luo, "Task offloading in NOMA-based fog computing networks: A deep Q-learning approach," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Honolulu, HI, USA, Dec. 2019, pp. 1–6.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A survey on enabling technologies, protocols, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2347–2376, 4th Quart., 2015.

[3] C.-M. Tseng, S. C.-K. Chau, and X. Liu, "Improving viability of electric taxis by taxi service strategy optimization: A big data study of New York City," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 817–829, Mar. 2019.

[4] Q. Du, H. Song, and X. Zhu, "Social-feature enabled communications among devices toward the smart IoT community," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 130–137, Jan. 2019.

[5] A. J. Jara, M. A. Zamora-Izquierdo, and A. F. Skarmeta, "Interconnection framework for mHealth and remote monitoring based on the Internet of Things," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 47–65, Sep. 2013.

[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[7] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[8] N. Chen, Y. Yang, T. Zhang, X. Luo, and J. Zao, "Fog as a service technology," *IEEE Commun. Mag.*, vol. 56, no. 11, pp. 95–101, Nov. 2018.

[9] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 14–20, Aug. 2017.

[10] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4702–4711, Oct. 2018.

[11] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4590–4602, Oct. 2018.

[12] Y. Yang, X. Luo, X. Chu, and M. T. Zhou, *Fog-Enabled Intelligent IoT Systems*. Cham, Switzerland: Springer, 2019.

[13] Y. Yang, "Multi-tier computing networks for intelligent IoT," *Nature Electron.*, vol. 2, no. 1, pp. 4–5, Jan. 2019.

[14] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[15] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4076–4087, Oct. 2018.

[16] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11399–11403, Nov. 2019.

[17] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MOBIHOC)*, Hilton Head Island, SC, USA, Jun. 2012, pp. 145–154.

[18] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, Apr. 2014, pp. 1060–1068.

[19] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-task assignment for crowdsensing in mobile social networks," in *Proc. IEEE INFOCOM*, Hong Kong, Apr. 2015, pp. 2227–2235.

[20] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE ICC*, Kuala Lumpur, Malaysia, May 2016, pp. 1–6.

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

[22] Z. Ding, Z. Yang, P. Fan, and H. V. Poor, "On the performance of non-orthogonal multiple access in 5G systems with randomly deployed users," *IEEE Signal Process. Lett.*, vol. 21, no. 12, pp. 1501–1505, Dec. 2014.

[23] Z. Ding, P. Fan, and H. V. Poor, "Impact of user pairing on 5G nonorthogonal multiple-access downlink transmissions," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6010–6023, Aug. 2016.

[24] L. Dai, B. Wang, Y. Yuan, S. Han, C.-L. I, and Z. Wang, "Non-orthogonal multiple access for 5G: Solutions, challenges, opportunities, and future research trends," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 74–81, Sep. 2016.

[25] D. Tse and P. Viswanath, *Fundamentals of Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[26] M. Mohseni, R. Zhang, and J. M. Cioffi, "Optimized transmission for fading multiple-access and broadcast channels with multiple antennas," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1627–1639, Aug. 2006.

[27] Z. Chen, Z. Ding, X. Dai, and R. Zhang, "An optimization perspective of the superiority of NOMA compared to conventional OMA," *IEEE Trans. Signal Process.*, vol. 65, no. 19, pp. 5191–5202, Oct. 2017.

[28] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12137–12151, Dec. 2018.

[29] H. Zhang, Y. Qiu, K. Long, G. K. Karagiannidis, X. Wang, and A. Nallanathan, "Resource allocation in NOMA-based fog radio access networks," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 110–115, Jun. 2018.

[30] X. Wen, H. Zhang, H. Zhang, and F. Fang, "Interference pricing resource allocation and user-subchannel matching for NOMA hierarchy fog networks," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 3, pp. 467–479, Jun. 2019.

[31] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, "Optimal task assignment and power allocation for NOMA mobile-edge computing networks," 2019, *arXiv:1904.12389*. [Online]. Available: http://arxiv.org/abs/1904.12389

[32] *Study on Downlink Multiuser Superposition Transmission for LTE*, document RP-150496, TSG RAN Meeting 67, 3rd Generation Partnership Project (3GPP), Mar. 2015.

[33] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, Jan. 1998.

[34] G. Huerta-Canepa and D. Lee, "An adaptable application offloading scheme based on application behavior," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, Okinawa, Japan, Apr. 2008, pp. 387–392.

[35] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[36] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Netw.*, vol. 21, no. 4, pp. 682–697, May 2008.

[37] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[38] A. E. Roth, "Deferred acceptance algorithms: History, theory, practice, and open questions," *Int. J. Game Theory*, vol. 36, nos. 3–4, pp. 537–569, Mar. 2008.

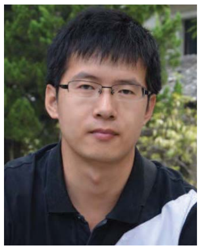
[39] D. F. Manlove, *Algorithmics Of Matching Under Preferences*. Singapore: World Scientific, 2013.

[40] D. Gale and L. Shapley, "College admissions and the stability of marriage," *Amer. Math. Monthly*, vol. 69, no. 1, pp. 9–15, Jan. 1962.

[41] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[42] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE ISCC*, Cappadocia, Turkey, Jul. 2012, pp. 59–66.

[43] J. Papandriopoulos and J. S. Evans, "Low-complexity distributed algorithms for spectrum balancing in multi-user DSL networks," in *Proc. IEEE ICC*, Istanbul, Turkey, Jun. 2006, pp. 3270–3275.

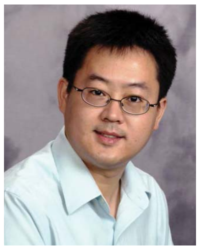


Kunlun Wang (Member, IEEE) received the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016. From 2016 to 2017, he was with Huawei Technologies Company, Ltd., where he was involved in energy efficiency algorithm design. From 2017 to 2019, he was with the Key Lab of Wireless Sensor Network and Communication, SIMIT, Chinese Academy of Sciences, Shanghai. Since March 2019, he has been a Research Assistant Professor with the School of Information Science and Technology, ShanghaiTech University. He is also with the Shanghai Institute of Fog Computing Technology (SHIFT). His current research interests include energy efficient communications, fog computing networks, resource allocation, and optimization algorithm.

Yong Zhou (Member, IEEE) received the B.Sc. and M.Eng. degrees from Shandong University, Jinan, China, in 2008 and 2011, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2015. From November 2015 to January 2018, he worked as a Post-Doctoral Research Fellow at the Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC, Canada. He is currently an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. His research interests include Internet of Things, edge computing, and reconfigurable intelligent surface.

Zening Liu (Student Member, IEEE) received the B.S. degree in electrical science and technology from Wuhan University, Wuhan, China, in 2015. He is currently pursuing the Ph.D. degree with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. He is also a visiting Ph.D. student at Arizona State University, Tempe, AZ, USA, from September 2019 to March 2020. His current research interests include task scheduling, resource allocation, and edge intelligence in edge/fog computing.

Ziyu Shao (Senior Member, IEEE) received the B.S. and M.Eng. degrees from Peking University, and the Ph.D. degree from The Chinese University of Hong Kong. He was a visiting Post-Doctoral Researcher at the EE Department, Princeton University. He was also a Visiting Professor at the EECS Department, UC Berkeley. He is currently an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. His current research interests lie in learning and optimization for intelligent networks, including fog/edge computing and cloud computing.



Xiliang Luo (Senior Member, IEEE) received the B.Sc. degree in physics from Peking University, Beijing, China, in 2001, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Minnesota, Minneapolis, MN, USA, in 2003 and 2006, respectively.

After finishing his Ph.D. studies, he joined Qualcomm Research and carried out cutting edge research at different posts as a Senior Engineer (2006), a Staff Engineer (2010), and then a Senior Staff Engineer (2013), where he was involved in the system designs, analyses, and standardization of 4G LTE. He was a Designer of various enhancements to Qualcomm's LTE solutions and led the designs of Qualcomms LTE modem for heterogeneous networks from initial concept to final completion. Since 2014, he has been with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. He has authored or coauthored over 100 research papers in top journals and conferences. He is also the co-inventor of over 70 US and international patents. His current research interests include signal processing, communications, and machine learning.

Yang Yang (Fellow, IEEE) received the B.Eng. and M.Eng. degrees in radio engineering from Southeast University, Nanjing, China, in 1996 and 1999, respectively, and the Ph.D. degree in information engineering from The Chinese University of Hong Kong, Hong Kong, in 2002.

He is currently a Full Professor at ShanghaiTech University, China, serving as the Executive Dean of the School of Creativity and Art and the Co-Director of the Shanghai Institute of Fog Computing Technology (SHIFT). Before joining ShanghaiTech University, he has held faculty positions at The Chinese University of Hong Kong, Brunel University, U.K., University College London (UCL), U.K., and SIMIT, CAS, China. His current research interests include fog computing networks, service-oriented collaborative intelligence, wireless sensor networks, IoT applications, and advanced testbeds and experiments. He has published more than 200 papers and filed more than 80 technical patents in these research areas. He is the Chair of the Steering Committee of Asia-Pacific Conference on Communications (APCC) since January 2019. In addition, he is the General Co-Chair of the IEEE DSP 2018 Conference and the TPC Vice-Chair of the IEEE ICC 2019 Conference.