Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things

Xingxia Dai, Zhu Xiao, *Senior Member, IEEE*, Hongbo Jiang, *Senior Member, IEEE*, Mamoun Alazab, *Senior Member, IEEE*, John C. S. Lui, *Fellow, IEEE*, Schahram Dustdar, *Fellow, IEEE*, and Jiangchuan Liu, *Fellow, IEEE*

***Abstract*—Mobile edge computing (MEC) and device-to-device (D2D) offloading are two promising paradigms in the industrial Internet of Things (IIoT). In this article, we investigate task co-offloading, where computing-intensive industrial tasks can be offloaded to MEC servers via cellular links or nearby IIoT devices via D2D links. This co-offloading delivers small computation delay while avoiding network congestion. However, erratic movements, the selfish nature of devices and incomplete offloading information bring inherent challenges. Motivated by these, we propose a co-offloading framework, integrating migration cost and offloading willingness, in D2D-assisted MEC networks. Then, we investigate a learning-based task co-offloading algorithm, with the goal of minimal system cost (i.e., task delay and migration cost). The proposed algorithm enables IIoT devices to observe and learn the system cost from candidate edge nodes, thereby selecting the optimal edge node without requiring complete offloading information. Furthermore, we conduct simulations to verify the proposed co-offloading algorithm.**

***Index Terms*—Device-to-device (D2D) offloading, industrial Internet of Things (IIoT) devices, mobile edge computing (MEC), multiarmed bandit (MAB).**

Manuscript received 12 December 2021; revised 6 February 2022 and 28 February 2022; accepted 9 March 2022. Date of publication 15 March 2022; date of current version 8 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U20A20181, in part by the Key Research and Development Project of Hunan Province of China under Grant 2022GK2020, in part by Hunan Natural Science Foundation of China under Grant 2022JJ2059, and in part by the Funding Projects of Zhejiang under Grant 2021LC0AB05. Paper no. TII-21-5505. *(Corresponding authors: Zhu Xiao; Hongbo Jiang.)*

Xingxia Dai, Zhu Xiao, and Hongbo Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: xingxdai718@gmail.com; zhxiao@hnu.edu.cn; hongbojiang2004@gmail.com).

Mamoun Alazab is with the College of Engineering, IT, and Environment, Charles Darwin University, Darwin, NT 0810, Australia (e-mail: mamoun.alazab@cdu.edu.au).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Schahram Dustdar is with TU Wien, 1040 Vienna, Austria (e-mail: dustdar@infosys.tuwien.ac.at).

Jiangchuan Liu is with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: jcliu@cs.sfu.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3158974.

Digital Object Identifier 10.1109/TII.2022.3158974

I. Introduction

RECENT years have witnessed that industrial Internet of Things (IIoT) has drawn ever-increasing attention, boosting the development of smart factories. IIoT devices in smart factories are required to continuously and timely process tasks for effective industrial services, involving industrial control, smart transportation, and virtual reality (VR) [1]. However, taking VR as an instance, this service needs to consume enormous computing resources for rendering tasks, while constrained computing capabilities and limited battery lifetime of IIoT devices impede continuous local rendering [2]. As a current solution, these computing-intensive industrial tasks are offloaded to cloud servers to seek abundant computing resources [3]. Unfortunately, such a solution suffers from large transmission delay due to long network distance, and hence degrades service performance.

In response, mobile edge computing (MEC) provides a promising solution to sink cloud computing to the network edge, thereby shortening task transmission delay [4]. In MEC, edge servers offer computing resources for multiple IIoT devices within the communication coverage by creating a serial of virtual machines. As such, computing-intensive industrial tasks can be offloaded to MEC servers to pursue less computing delay in the meantime keeping low transmission delay. Nevertheless, both the communication capabilities and computing resources of MEC servers are limited. Once IIoT devices are out of the edge communication coverage, they are not supported by task offloading. Additionally, when an MEC delivers offloading services for numerous IIoT devices simultaneously, network congestion is typically inevitable, especially for tasks with a huge volume of data bits. This incurs large queue delay, even may cause task failure.

Facing these issues, device-to-device (D2D) offloading [5] serves as a promising solution to expand the communication coverage and address the network congestion problem. In D2D offloading, IIoT devices enable to establish short-range D2D communication links. According to the Cisco Annual Internet Report (2018–2023), the D2D (or the interchanged term "machine-to-machine") links are expected to add up to 14.7 billion by 2023 [6]. These links expand the network coverage since one IIoT device can communicate to another IIoT device out of MEC communication range. Furthermore, these links allow an IIoT device to offload computing-intensive tasks to its nearby IIoT devices with surplus computing resources. For these reasons, D2D offloading, serving as a supplement of MEC, effectively promotes industrial service performance.

In this article, we strive to propose a co-offloading scheme in D2D-assisted MEC networks. Different from previous works [7]–[20], which investigate MEC and D2D offloading separately. Our work concentrates on a joint MEC and D2D offloading scheme. Within our proposed scheme, IIoT devices are divided into two categories, i.e., service devices (SeDs) and user devices (UDs). SeDs are capable of processing tasks locally within the stipulated task deadline, exploring their underutilized computing resources and performing edge computing for UDs. As such, computing-intensive industrial tasks generated by UDs can be offloaded to MEC servers and SeDs based on industrial task features, i.e., computation demands and data bits. This co-offloading empowers to deliver small computation delay while avoiding network congestion, effectively avoiding suboptimal offloading performance.

However, when designing a co-offloading scheme in D2D-assisted MEC networks, we are facing several challenges. 1) *Erratic movement.* UDs move erratically that they may roam throughout communication coverage supported by different MEC servers and SeDs. In this case, offloading decisions following the movement are beneficial for small transmission delay, while additional migration cost arises in return, such as service interruption delay caused by service data roaming and service virtual machine set up [21]. 2) *Selfish nature.* SeDs tend to act in a selfish manner that they usually have a low will for computing resource sharing. The reason is that task offloading consumes considerable computing resources, while the primary concern of SeDs is to maintain their own performance given the limited computing resources. Therefore, it is challenging to measure the offloading willingness and stimulate computing resource sharing among IIoT devices [22]. 3) *Incomplete information.* Another key challenge for task co-offloading is the lack of complete information. Erratic movement and dynamic environment lead to the varying of network topology and channel state. Such information is intractable to model or predict before task offloading. Consequently, UDs are required to make co-offloading decisions based on incomplete offloading information [23].

To address the above-mentioned challenges, we design a co-offloading framework integrating MEC and D2D offloading, where computing-intensive industrial tasks are offloaded to MEC servers and SeDs based on task features and incomplete offloading information. This framework jointly considers migration cost and offloading willingness of SeDs with the goal of minimal system cost, i.e., the sum of task delay and migration cost. Our contributions are highlighted as follows.

1) We propose a co-offloading framework in D2D-assisted MEC networks, where MEC servers and SeDs enable offloading services for computing-intensive industrial tasks. In this framework, migration cost is considered to avoid frequent migration while keeping small transmission delay, detailed in Section III-B. Besides, we design a willingness metric, detailed in Section III-C, to quantify the possibility of D2D offloading and stimulate resource sharing among IIoT devices.
2) We investigate a learning-based industrial task co-offloading algorithm based on multiarmed bandit (MAB) theory, detailed in Section IV. This algorithm involves twice learning in the SeD willingness and the system cost. A UD first finds out the SeD with the largest willingness, and the selected SeD serves as a candidate edge node to perform edge computing. Then, the UD learns task delay and migration cost from the candidate edge nodes, targeting the edge node with minimal system cost. After several learning times, the optimal co-offloading decisions can be made without requiring complete offloading information.
3) We conduct extensive simulations, detailed in Section V, to validate the effectiveness of our proposed learning-based algorithm. The simulation results demonstrate the superiority of the proposed algorithm compared with other learning-based algorithms under various system parameters, such as task computation demands, data bits, and learning times.

The rest of this article is organized as follows. Section II presents the related works followed by the system model and problem formulation in Section III. Section IV is learning-based task co-offloading. In Section V, we conduct evaluations. Finally, Section VI concludes this article.

II. Related Works

In this section, we classify the existing task offloading into the following three categories.

One is the MEC-enabled task offloading, where computing-intensive tasks can be offloaded to MEC serves (e.g., base stations, access points, and roadside units) for low computation delay and small local energy consumption [7]–[14]. Ma *et al.* [7] presented a framework combining service caching and workload scheduling in MEC. To achieve minimal service delay and outsourcing traffic, an iterative algorithm based on Gibbs sampling is proposed. Gao *et al.* [8] jointly optimize the access network selection and service placement in MEC. Each user selects an access point for service offloading with the goal of small service delay and switching cost. Yang *et al.* [9] investigated a task offloading strategy in MEC, where edge servers are heterogeneous and users have different locations. The strategy is analyzed based on a Markov decision process (MDP) and realizes small offloading time. Chen *et al.* [10] developed a task processing and caching solution in MEC. By rationally adjusting communication, computation, and caching settings, mobile devices enable to acquire minimal mean delay. Liu *et al.* [11] considered a task offloading problem in MEC, aiming at the long-term minimal task delay and energy consumption. This optimization problem is achieved by finding out the optimal offloading strategies, CPU frequency and transmission power. Xu *et al.* [12] proposeed trust-aware task offloading in MEC-enabled internet of vehicles. To achieve minimal service response time, an improving strength Pareto evolutionary algorithm is derived. Lai *et al.* [13] addressed user allocation problems in MEC, where a vendor determines, which MEC servers to serve for a specific user. This problem is formulated as a potential game based on user's quality of service. Correspondingly, a game-theoretic approach is proposed, and its performance is theoretically and empirically evaluated. Zhou *et al.* [14] offloaded computing-intensive tasks to nearby BSs for task processing on the internet of health things (IoHT), The goal of this work is to minimize the total long-term energy consumption of IoHT devices under task delay and reliability requirements.

Another is the D2D-assisted task offloading, where computing-intensive tasks can be offloaded to other devices with underutilized computing resources to maintain service performance [15]–[20]. Cheng *et al.* [15] advocated a D2D-assisted computation offloading framework in cognitive radio networks. In this framework, the primary user offloads computation tasks to the secondary user by D2D communications. To minimize energy consumption, offloading decisions and transmission power are jointly optimized under the limitation of deadline and power control. Budhiraja *et al.* [16] propose a Tactile Internet-driven delay assessment in a D2D-enabled communication framework. In this framework, Tactile Internet communication and a pricing-based 3-D matching method are derived, targeting improving transmission speed and throughput of cell edge users. Zhou *et al.* [17] investigated offloading willingness of mobile nodes in D2D offloading. An inventive-driven method is proposed to simulate mobile nodes to participate in D2D offloading. On this basis, an integer nonlinear programming problem is formulated to maximize the saving energy. Hamdi *et al.* [18] studied a joint task assignment and power control problem in D2D offloading with the assistance of energy harvesting technology. This work assumes that energy consumption conducted by D2D links can be compensated by the harvested energy. Guided by this assumption, energy efficiency is maximized under the constraint of task delay and energy causality. Saleem *et al.* [19] studied a D2D-enabled cooperative MEC network, where resource-limited devices enable to offload their tasks to the devices with surplus resources, with the goal of minimal total task execution delay. This optimization is achieved by adjusting task assignment and power allocation. Peng *et al.* [20] considered a joint multiuser cooperative partial offloading, transmission scheduling, and computation allocating problem in D2D-assisted MEC. In this article, idle mobile terminals serve as relay nodes for active mobile terminals, targeting minimal task response latency, and energy consumption.

The last one is the co-offloading integrating MEC and D2D offloading, where MEC servers and devices with excessive computing resources are both served as candidate edge nodes for task processing. IIoT devices offload computing-intensive tasks to these edge nodes for small computation delay and local energy consumption while keeping low queue delay [24]–[29]. Wang *et al.* [24] investigated a co-offloading problem in D2D enabled MEC networks by integrating task traffic and computation. Mobile devices make offloading decisions based on the locations and offloading willingness. Tang *et al.* [25] presented a framework integrating MEC and cache-enabled D2D communications with the goal of minimal energy cost. File popularity and user preference are paid particular attention in this framework. A reinforcement learning-based method is proposed to determine file popularity and user preference. Sun *et al.* [26] formalized a resource management problem in D2D-aided MEC networks. One device can offload tasks to MEC servers or other devices with surplus resources under the energy harvesting system. By jointly optimizing computation offloading decisions, energy transmission power, and CPU processing speed, the optimal solution with maximal long-term utility energy efficiency can be realized. He *et al.* [27] considered D2D communications with MEC, where each device can offload its task to an edge node by cellular link or a nearby device with abundant resources via D2D link. The goal of this work is to maximize the number of devices supported by the cellular networks under a series of constraints in both communication and computation. Tan *et al.* [28] optimized offloading decisions, collaboration decisions, and resource allocation in the multiuser collaborative MEC network. In this network, delay-sensitive tasks can be processed locally, offloaded to the nearby mobile devices or MEC servers, aiming to the minimal total energy consumption of mobile users meanwhile guaranteeing the task delay constraint. Li and Cai [29] investigated an online truthful mechanism integrating resource allocation, where a requester can offload its tasks to the collaborators and BS. Based on data bits, task delay, and task preference, the authors derive a social welfare-maximization optimization problem by jointly considering collaborator selections, resource allocation, time scheduling decisions, and pricing policy designs.

As these works mentioned, D2D-assisted MEC offloading facilitates small task delay and local energy consumption. However, this co-offloading also produces additional migration cost due to erratic movement, and derives offloading willingness dilemma due to the selfish nature of devices. What's worse, intractable networks hamper the acquisition of transmission rate, and hence co-offloading has to rely on incomplete offloading information. However, most of the existing works overlook these issues, thus the system performance is easily trapped by suboptimal co-offloading strategies. Different from these works, we study industrial task co-offloading in D2D-assisted MEC networks by jointly considering migration cost, offloading willingness and incomplete offloading information. To the best of our knowledge, this co-offloading has not been studied before.

III. System Models and Problem Formulation

Fig. 1 presents the proposed system model, involving two entities, i.e., MEC servers and IIoT devices. The IIoT devices are classified by UDs and SeDs. For a UD, there are multiple edge nodes, i.e., MEC servers and SeDs, enabling offloading services for it. In this work, we discretize the timeline into T time slots. At each time slot, a UD generates one computing-intensive industrial task and offloads its task to a single edge node. For simplify, we assume that tasks generated by the same UD across time slots hold sequential relationship and are executed in order [30]. Besides, we list the key notations as Table I for better readability.

A. Co-Offloading Model

UDs enable to offload tasks to MEC servers and SeDs for processing, which is referred to as task co-offloading in this article. The co-offloading decision of task n at time slot t is

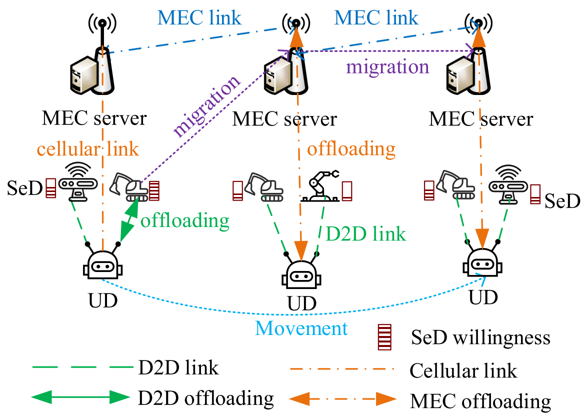

Fig. 1. Industrial task co-offloading in D2D-assisted MEC networks. Computing-intensive industrial tasks are offloaded to MEC servers and SeDs by joint considering SeD willingness and migration cost, targeting minimal system cost, i.e., the sum of task delay and migration cost.

TABLE I
SUMMARY OF THE KEY NOTATIONS

Notations	Definitions
$\mathbf{x}_n^t$	The co-offloading decision of UD n at time slot t
$\mathcal{M}$	The edge node set
$\mathcal{M}_{MEC}$	The MEC server set
$\mathcal{M}_{SeD}$	The SeD set
b_n^t	The data bits of task n at time slot t
B	The wireless bandwidth
δ^2	The noise power
$r_{n,m}^{t,mec}$	The transmission rate between UD n and MEC m
$T_{n,m,mec}^{t,trans}$	The communication delay for task n transmitting to MEC m
$l_{m,n}^{t,mec}$	The distance between MEC m and UD n
R_{mec}	The communication range of an MEC server
$T_{n,m,mec}^{t,queue}$	The queue delay of MEC m for task n processing
c_n^t	The computation demands of task n at time slot t
$f_{m,n}^{t,mec}$	The allocated computing resources of MEC m for task n
$T_{n,m,mec}^{t,compute}$	The computing delay of task n in MEC offloading
$r_{n,m}^{t,d2d}$	The communication rate between UD n and SeD m
$T_{n,m,d2d}^{t,trans}$	The communication delay for task n transmitting to SeD m
$T_{n,m,d2d}^{t,compute}$	The computing delay of task n in D2D offloading
$W_{m,n}^t$	The offloading willingness of SeD m for processing task n
G_n^t	The migration cost of task n at time slot t

denoted as $\mathbf{x}_n^t = \{x_m^{t,n}, m \in \mathcal{M}\}$, where $x_m^{t,n} \in \{0,1\}$, $\mathcal{M}$ is an edge node set consisting of MEC set $\mathcal{M}_{\text{MEC}}$ and SeD set $\mathcal{M}_{\text{SeD}}$, i.e., $\mathcal{M} = \mathcal{M}_{\text{MEC}} \cup \mathcal{M}_{\text{SeD}}$. We assume that the optimal edge node will not change at each time slot, but can be changed across time slots.

1) MEC Offloading: When $x_m^{t,n} = 1, m \in \mathcal{M}_{\text{MEC}}$, task n is offloaded to MEC server m at time slot t. In MEC offloading, each UD is assigned one subchannel of cellular links for transmitting the offloaded task to the target MEC server. We define $h_{n,m}^{t,\text{mec}}$ and $p_{n,m}^{t,\text{mec}}$ as the channel power gain and transmission power between UD n and MEC server m at time slot t. Additionally, we assume the wireless bandwidth and noise power keep fixed during T time slots in MEC offloading, denoted as B_{mec} and δ^2_{mec}, respectively. As such, the communication rate between UD n and MEC server m at time slot t can be expressed as

$$r_{n,m}^{t,\text{mec}} = B_{\text{mec}} \log_2 \left(1 + \frac{p_{n,m}^{t,\text{mec}} \mid h_{n,m}^{t,\text{mec}} \mid^2}{\delta^2_{\text{mec}}}\right), m \in \mathcal{M}_{\text{MEC}}. \quad (1)$$

Correspondingly, we obtain the communication delay when task n is transmitted to MEC m at time slot t, expressed as

$$T_{n,m,\text{mec}}^{t,\text{trans}} = \frac{b_n^t}{r_{n,m}^{t,\text{mec}}}, m \in \mathcal{M}_{\text{MEC}} \quad (2)$$

where b_n^t represents the data bits of task n at time slot t. Since a UD moves erratic, the UD may be incapable of communicating with MEC server m directly. In this case, task n is required to be propagated to the target MEC m through edges-relay in local area network, and thus network propagation delay of $t_{n,m}^{t,\text{pro}}$ is produced. Guided by this, we define the task transmission delay as

$$T_{n,m,\text{mec}}^{t,\text{trans}} = \begin{cases} b_n^t / r_{n,m}^{t,\text{mec}}, & l_{m,n}^{t,\text{mec}} \leq R_{\text{mec}} \\ b_n^t / r_{n,m}^{t,\text{mec}} + t_{n,m}^{t,\text{pro}}, & l_{m,n}^{t,\text{mec}} > R_{\text{mec}} \end{cases} \quad (3)$$

where $l_{m,n}^{t,\text{mec}}$ is the distance between MEC m and UD n at time slot t. The parameter R_{mec} is a fixed value, denoting the communication range of an MEC server.

Then, we use an $M/M/1$ queuing model to characterize the queue delay caused by network congestion in MEC offloading

$$T_{n,m,\text{mec}}^{t,\text{queue}} = \frac{t_{\text{exp}}}{1 - t_{\text{exp}} c_0}, m \in \mathcal{M}_{\text{MEC}} \quad (4)$$

where t_{exp} is the expected delay for sending and receiving data without network congestion, and c_0 is the normalized computation tasks processed by MEC cooperation [31].

After that, task n can be processed by MEC m with the allocated computing resources. Since an MEC server serves for multiple UDs, we use $f_{m,n}^{t,\text{mec}}$ to denote the allocated computing resources of MEC m for processing task n. Based on this, we calculate the computing delay of task n as

$$T_{n,m,\text{mec}}^{t,\text{compute}} = \frac{c_n^t}{f_{m,n}^{t,\text{mec}}}, m \in \mathcal{M}_{\text{MEC}} \quad (5)$$

where c_n^t is the computation demands of task n. Overall, the total delay of task n for MEC offloading is formulated as

$$T_{\text{mec}}^{t,n} = T_{n,m,\text{mec}}^{t,\text{trans}} + T_{n,m,\text{mec}}^{t,\text{queue}} + T_{n,m,\text{mec}}^{t,\text{compute}}, m \in \mathcal{M}_{\text{MEC}}. \quad (6)$$

2) D2D Offloading: When $x_m^{t,n} = 1, m \in \mathcal{M}_{\text{SeD}}$, task n is offloaded from a UD to SeD m via the D2D link at time slot t. We define $h_{n,m}^{t,d2d}$ and $p_{n,m}^{t,d2d}$ as the channel power gain and transmission power between UD n and SeD m at time slot t. Given fixed bandwidth of B_{d2d} and noise power of δ^2_{d2d} in D2D offloading, we obtain the communication rate between UD n and SeD m at time slot t, expressed as

$$r_{n,m}^{t,d2d} = B_{\text{d2d}} \log_2 \left(1 + \frac{p_{n,m}^{t,d2d} \mid h_{n,m}^{t,d2d} \mid^2}{\delta^2_{\text{d2d}}}\right), m \in \mathcal{M}_{\text{SeD}}. \quad (7)$$

Correspondingly, when task n is transmitted to the target SeD m, the communication delay is expressed as

$$T_{n,m,d2d}^{t,\text{trans}} = \frac{b_n^t}{r_{n,m}^{t,d2d}}, m \in \mathcal{M}_{\text{SeD}}. \quad (8)$$

Since the communication coverage conducted by D2D links is small, we assume that each SeD delivers offloading service for at most one UD to avoid queue delay [19]. Then, we use $f_{m,n}^{t,d2d}, m \in \mathcal{M}_{\text{SeD}}$, to denote the computing resources of SeD m for processing task n. Therefore, the computation delay is expressed as follows:

$$T_{n,m,d2d}^{t,\text{compute}} = \frac{c_n^t}{f_{m,n}^{t,d2d}}, m \in \mathcal{M}_{\text{SeD}}. \tag{9}$$

It is noted that the computation results need to be feedback from the target edge node $m \in \mathcal{M}$ to the UD. If direct communication is not available for result feedback, the results will be transmitted to the UD via edges-relay. Because the output result size is much smaller compared with the input task, we neglect the result feedback delay in both MEC and D2D offloading [32]. As such, the total delay of task n at time slot t for D2D offloading is expressed as

$$T_{d2d}^{t,n} = T_{n,m,d2d}^{t,\text{trans}} + T_{n,m,d2d}^{t,\text{compute}}, m \in \mathcal{M}_{\text{SeD}}. \tag{10}$$

B. SeD Willingness Model

SeDs tend to act in a selfish manner that they usually have a low will for computing resource sharing. The reason is that processing tasks offloaded by UDs consume considerable computing resources, while the primary concern of SeDs is to maintain their own service performance given the limited computing resources. Under the selfish nature, massive computing resources of SeDs are wasted, while a variety of tasks generated by UDs will suffer from the prolonged delay due to lacking computing resources. In addition, varying roles of IIoT devices indicate that a SeD could serve as a UD at the next time slot and seek task offloading services. Therefore, SeDs acting for their interests will impact their own performance and ultimately degrade the whole system performance.

Motivated by the challenges mentioned above, a promising solution is to stimulate computing resource sharing among IIoT devices. For a SeD $m \in \mathcal{M}_{\text{SeD}}$, we present a function to depict its offloading willingness for task n at time slot t

$$W_{m,n}^t = \gamma I_n^t f_{m,n}^{t,d2d} \sqrt{\sigma_m^t}, m \in \mathcal{M}_{\text{SeD}} \tag{11}$$

where γ is a normalized coefficient to make the willingness range from 0 to 1. I_n^t is an incentive factor, recording the times that the UD serves as a SeD and process tasks offloaded by other devices up to time slot t; $f_{m,n}^{t,d2d}$ and σ_m^t are performance factors, paying attention to SeD performance. We define $\sigma_m^t = c_m^t(c - c_m^t)$, where c_m^t represent the task computation demands of SeD m at time slot t, $c \in (c_m^t, 2c_m^t)$. When SeD m is with heavy computation demands, σ_m^t will drop sharply and hence go against large willingness. Based on the willingness function shown in (11), the willingness of SeD m at time slot t is jointly determined by the incentive factor, i.e., I_n^t, and the performance factors, i.e., $f_{m,n}^{t,d2d}$ and σ_m^t. For example, suppose that 1) UD n has a large incentive value I_n^t, indicating this UD had played as the *SeD role* and shared computing resource for other IIoT devices; 2) the requested SeD m has small own computation demands and large computing resources. Then, the SeD will perform a large willingness for processing tasks offloaded by UD n.

In a nutshell, the proposed willingness function is beneficial for effective D2D offloading via resolving the "selfishness dilemma." On the one hand, recall I_n^t in (11), UD n broadcasts its incentive value to SeDs. A large I_n^t would reap a higher willingness W_m^t, and hence increasing the chance that SeDs deliver offloading services for UD n. Furthermore, this incentive factor will in turn stimulates SeDs to share their resource, as SeDs could turn into UDs and would strive to a large I_n^t for seeking task offloading services. On the other hand, $f_{m,n}^{t,d2d}$ and σ_m^t in the willingness function ensure performance-guaranteed offloading for the SeDs, since a SeD with large computing resources and low computation demands tends to contribute its computing resources in D2D networks.

C. Migration Cost Model

Since a UD may roam throughout several areas supported by different edge nodes, dynamic service migration should be considered to maintain effective task co-offloading [33]. For example, a UD offloads its task to an MEC at time slot t via the direct cellular link between them. At the next time slot, the UD roams to the communication coverage served by another edge node. In this case, the UD holds two co-offloading decisions. One is that the UD offloads its task to the same edge node selected at time slot t for keeping task processing continuity. As a result, long transmission delay may incur as the UD movements extend network distance and the propagation delay caused by edges-relay needs to be considered. The other is that the UD makes its co-offloading decision following UD's movements and the task will be processed by another edge node different from the former MEC. Guided by this co-offloading decision, the transmission network distance is greatly reduced, while additional migration overhead incurs, such as service interruption delay and service virtual machine set up. Overall, co-offloading decisions following UD's movements are beneficial for small transmission delay, while additional migration overhead arises in return. It is therefore important to investigate the impact of dynamic service migration in task co-offloading. To this end, we introduce a parameter c_{mig} to denote the migration cost between different edge nodes. The migration cost for UD generating task n at time slot t is expressed as

$$G_n^t = c_{\text{mig}} \mathbb{I}\{x_i^{t-1,n-1} = 1, x_j^{t,n} = 1, i \neq j\}, i, j \in \mathcal{M} \tag{12}$$

where $\mathbb{I}\{x\}$ is an indicator function. When event x is true, $\mathbb{I}\{x\} = 1$; if event x is false, $\mathbb{I}\{x\} = 0$.

D. Problem Formulation

Task delay reflects service performance directly, especially for time-critical industrial tasks. For that reason, our objective is to minimize task delay in this article. To achieve small delay, frequent service migration is typically inevitable for the sake of less network distance and transmission delay. However, frequent migration produces additional data roaming and virtual machine set-up delay, leading to degraded offloading performance. As such, we define the optimization objective as the system cost of task n conducted by edge node m processing at time slot t, expressed as

$$\Xi_{m,n}^{t} = \alpha T_n^t + (1-\alpha) G_n^t, m \in \mathcal{M} \tag{13}$$

which integrates task delay and service migration cost. The weighting factor $\alpha \in [0,1]$ is introduced to balance these two conflicting objectives. $T_n^t = \mathbf{x}_n^t (T_{\text{mec}}^{t,n} + T_{\text{d2d}}^{t,n}), m \in \mathcal{M}$, indicating the delay of task n. Mathematically, we formulate the co-offloading problem as

$$\min_{\mathbf{x}_n^t} \frac{1}{T} \sum_{t=1}^{T} \sum_{n \in \mathcal{N}} \Xi_{m,n}^{t}, m \in \mathcal{M} \tag{14}$$

$$s.t.\, x_m^{t,n} \in \{0,1\} \tag{15a}$$

$$\sum_{m=1}^{M} x_m^{t,n} = 1 \tag{15b}$$

$$f_{m,n}^t \leq F^m \tag{15c}$$

where constraint (15a) denotes that offloading decisions are bounded by integer zero and one, constraint (15b) implies that each task is required to be offloaded to a single edge node for task continuity, and constraint (15c) indicates that the allocated computing resources cannot exceed the maximum computing capabilities of the edge node.

To solve the formulated problem, global offloading information, including the task computation demands, allocated computing resources, and channel states, are required. Unfortunately, computation demands change with varying task requests, and computing capacities of edge nodes are heterogeneous. What's worse, the transmission rate between a UD and edge nodes is hard to model or predict due to UD movements and varying network topology. These challenges call for a learning-based optimization approach that can efficiently perform industrial task co-offloading without relying on overall offloading information.

IV. Learning-Based Task Co-Offloading

In this section, we investigate a learning-based approach to find out the optimal offloading decision. Following that, we propose a learning-based task co-offloading algorithm and analyze the learning regret.

A. Learning-Based Co-Offloading Approach Based on MAB

When a UD only holds incomplete offloading information, i.e., lacking information of the allocated computing resources and channel states, finding out the optimal edge node for task processing directly is difficult. Under the circumstances, the UD needs to observe and learn edge performance while offloading their tasks. As such, we propose a learning-based task co-offloading approach based on MAB theory, with the goal of minimal task delay and service migration cost.

MAB focuses on the exploration-exploitation dilemma in reinforcement learning [34]. The gambler does not have any prior information about k arms in a k-armed bandit problem, choosing an arm per time slot and correspondingly obtaining a reward. The purpose of the game is to maximize the reward value via "learning while choosing." In this classical situation, the gambler wants to choose a new arm to pursue a higher reward (i.e., exploration), yet he is unwilling to undertake the related risk (the reward of the new arm is less than the former one). For that reason, the arm with the largest reward currently may be selected (i.e., exploitation). Apparently, MAB theory enables to be adopted in our industrial task co-offloading problem, where the candidate edge nodes are considered as arms with different reward values. In our work, the reward is determined by task delay and service migration cost as shown in (14). The UD may select the current optimal edge node or search for new edge nodes for possible better rewards.

Despite the above analysis, we emphasize there are three main differences of our proposed industrial task co-offloading approach compared with the classical MAB. 1) The optimization objective. Different from MAB, our goal is to minimize task delay and migration cost rather than seeking the largest value. 2) States of arms. Candidate edge nodes are varying due to UD movements and changing computation demands, while "arms" keep fixed in classical MAB. 3) Learning process. The proposed learning-based approach involves twice learning at each iteration instead of once learning in classical MAB.

It is noted that the proposed approach is easy to implement in the real world since network information (i.e., channel states, transmission rate), and computing resources information (i.e., computation demands and allocated computing resources), are not required prior.

B. Learning-Based Co-Offloading Algorithm

We propose a learning-based task co-offloading algorithm as shown in Algorithm IV-A, which consists of two stages, i.e., SeD choosing (lines $2 \sim 13$) and co-offloading decision making (lines $14 \sim 27$). The former stage selects a SeD with the largest willingness, and the latter finds out the edge node with minimal task delay and service migration cost.

In the SeD choosing stage, we first initialize SeD set by selecting each SeD once at least. We define $N_{m_1}^t, m_1 \in \mathcal{M}_{\text{SeD}}$, to denote the selected times of SeD m_1 up to time slot t. After each SeD has been selected at least once, we find out the SeD π_1 with maximum index value based on t times learning

$$\pi_1^t = \arg\max \left\{ \bar{W}_{m_1,n}^{t-1} + \sqrt{\frac{2 \ln t}{N_{m_1}^{t-1}}} \right\}, m_1 \in \mathcal{M}_{\text{SeD}} \tag{16}$$

where $\bar{W}_{m_1,n}^{t-1}$ represents the empirical (sample-mean) estimation of SeD willingness up to time slot $t-1$, and $\sqrt{\frac{2 \ln t}{N_m^{t-1}}}$ is the confidence bound used to realize an exploration-exploitation tradeoff in SeD choosing. Then, SeD π_1 is selected at time slot t, and hence the selected times of SeD π_1 plus one. Correspondingly, we update the average willingness of SeD π_1 as follows:

$$\bar{W}_{\pi_1}^t = \frac{N_{\pi_1}^{t-1} \bar{W}_{\pi_1,n}^{t-1} + W_{\pi_1,n}^t}{N_{\pi_1}^t}, \pi_1 \in \mathcal{M}_{\text{SeD}}. \tag{17}$$

In the co-offloading decision making stage, we first update the candidate edge node set by adding SeD π_1 to the MEC set to form a new set of $\mathcal{M}_{\text{cand}}$. If edge node $m \in \mathcal{M}_{\text{cand}}$ has not

Algorithm 1: Learning-based Task Co-offloading.

Input : b_n^t, c_n^t and α.
Output: The optimal task offloading decision $x_*^{t,n}$.
1 **for** $t = 1$ ***to*** $t = T$ **do**
2 **Stage 1: SeD choosing**
3 **for** $m_1 = 1$ ***to*** $m_1 = M_{SeD}$ **do**
4 **if** *m_1 has not be selected once after t-th learning* **then**
5 SeD m_1 is selected at time slot t.
6 $N_{m_1}^t = 1$.
7 **end**
8 **else**
9 Find out the SeD with maximum index value based on Equation (16).
10 **end**
11 Updata the selected times: $N_{\pi_1}^t = N_{\pi_1}^{t-1} + 1$.
12 Updata the avarage williness value based on Equation (17).
13 **end**
14 $\mathcal{M}_{cand} = \mathcal{M}_{MEC} \leftarrow$ SeD π_1.
15 **Stage 2: Co-offloading decision making**
16 **for** $m_2 = 1$ ***to*** $m_2 = M_{cand}$ **do**
17 **if** *m_2 has not be selected once after t-th learning* **then**
18 Edge node m_2 is selected at time slot t.
19 $J_{m_2}^t = 1$.
20 **end**
21 **else**
22 Find out the edge node with minimal task delay and service migration cost based on Equation (18).
23 **end**
24 Updata the selected times: $J_{\pi_2}^t = J_{\pi_2}^{t-1} + 1$.
25 Updata the avarage task delay and service migration cost based on Equation (19).
26 **end**
27 **end**

been selected up to tth learning, it will be selected at time slot t. We use $J_{m_2}^t$ and $\bar{\Xi}_{m_2,n}^t$ to denote the chosen number and the empirical (sample-mean) estimation of edge node $m \in \mathcal{M}_{\text{cand}}$ at time slot t, respectively. On this basis, we define the index-based decision making function as

$$\pi_2^t = \arg\min \left\{ \bar{\Xi}_{m_2,n}^{t-1} - \sqrt{\frac{2\ln t}{J_{m_2}^{t-1}}} \right\}, m_2 \in \mathcal{M}_{\text{cand}}. \tag{18}$$

Correspondingly, we update the average task delay and service migration cost of edge node π_2 as follows:

$$\bar{\Xi}_{\pi_2}^t = \frac{J_{\pi_2}^{t-1}\bar{\Xi}_{\pi_2}^{t-1} + \Xi_{\pi_2}^t}{J_{\pi_2}^t}, \pi_2 \in \mathcal{M}_{\text{cand}}. \tag{19}$$

It is noted that the proposed learning-based task co-offloading algorithm is required to learn twice. First, the SeD with the highest willingness is picked out. Then, we combine the selected SeD and MECs to form a new candidate edge node set. The second learning aims to find out the edge node from the candidate edge node set with the goal of minimal task delay and service migration cost.

Computational Complexity: Line 9 shows a maximum willingness seeking problem, occupying $\mathcal{O}(M_{\text{SeD}})$, where $M_{\text{SeD}} = \mid \mathcal{M}_{\text{SeD}} \mid$ denotes the number of SeDs. Line 22 represents a minimal system cost seeking problem, the computational complexity is $\mathcal{O}(M_{\text{cand}})$, where $M_{\text{cand}} = \mid \mathcal{M}_{\text{cand}} \mid$ is the number of candidate edge nodes. The update behaviors, such as lines 11, 12, 24, and 25 have a computational complexity of $\mathcal{O}(1)$. Therefore, we conclude that the computational complexity of our proposed algorithm is $\mathcal{O}(M_{\text{SeD}} + M_{\text{cand}})$ for processing a single task. Based on this, we obtain the total computational complexity is $\mathcal{O}(N(M_{\text{SeD}} + M_{\text{cand}}))$, where N indicates the total offloaded tasks.

C. Regret Analysis

In this section, we analyze the learning regret conducted by Algorithm IV-A. Learning regret is commonly used to measure the performance loss in MAB algorithm [35]. Different from some classical MAB algorithms, such as UCB1, our proposed algorithm contains twice learning at each iteration, i.e., SeD choosing and co-offloading decision making. As such, the learning regret refers to *willingness choosing regret (WR)* and *decision making regret (DR)* in this article. On this basis, we define the learning regret at time slot t as the expected performance difference between our proposed Algorithm IV-A and the optimization algorithm with global offloading information

$$R^t = \mathbb{E}(\underbrace{(W_{*_1,n}^t - W_{m_1,n}^t)}_{WR} + \underbrace{(\Xi_{m_2,n}^t - \Xi_{*_2,n}^t)}_{DR}) \tag{20}$$

where $m_1 \in \mathcal{M}_{\text{SeD}}$ and $m_2 \in \mathcal{M}_{\text{cand}}$. The variable $W_{*_1,n}^t$ denotes the largest willingness of SeD $*_1 \in \mathcal{M}_{\text{SeD}}$ for processing task n, and $\Xi_{*_2,n}^t$ is the minimal system cost of the edge node $*_2 \in \mathcal{M}_{\text{cand}}$ for processing task n. Additionally, we introduce ν_{m_1} as the willingness expectation of SeD m_1, and μ_{m_2} as the system cost expectation of edge node m_2. Additionally, we define the expectation of optimal willingness and system cost as: $\nu^{*_1} = \max \nu_{m_1}, m_1 \in \mathcal{M}_{\text{SeD}}, \mu^{*_2} = \min \mu_{m_2}, m_2 \in \mathcal{M}_{\text{cand}}$.

Based on this, the learning regret up to time slot T can be transferred to the following equivalent expression:

$$\text{WR}^T = \sum_{\nu_{m_1} < \nu^{*_1}} (\nu^{*_1} - \nu_{m_1})\mathbb{E}(N_{m_1}^T), m_1 \in \mathcal{M}_{\text{SeD}} \tag{21}$$

$$\text{DR}^T = \sum_{\mu_{m_2} < \mu^{*_2}} (\mu_{m_2} - \mu^{*_2})\mathbb{E}(J_{m_2}^T), m_2 \in \mathcal{M}_{\text{cand}}. \tag{22}$$

Then, we present the total learning regret conducted by Algorithm IV-A by combining WR^T and DR^T, expressed as

$$R^T \leq \sum_{\nu_{m_1} < \nu^{*_1}} \sum_{\mu_{m_2} < \mu^{*_2}} \left(\frac{8\ln T}{(\nu^{*_1} - \nu_{m_1})^2} + \frac{8\ln T}{(\mu^{*_2} - \mu_{m_2})^2} + 2 + \frac{2\pi^2}{3} \right) \tag{23}$$
$$m_1 \in \mathcal{M}_{\text{SeD}}, m_2 \in \mathcal{M}_{\text{cand}}.$$

TABLE II
PARAMETER SETTINGS

Parameter description	Value
The number of time slots	50
The number of MECs	5
The number of SeDs	5
The migration cost	50 ms
The communication radius of each MEC	300 m
The D2D communication radius	30 m
The data bits of each industrial task	[0.2, 4] MB
The computation demands of each industrial task	[0.2, 4] GHZ
The computing capabilities of each MEC	[6, 15] GHZ
The available computing resources of each MEC	[0.2, 0.5] F^m
The computing capabilities of each SeD	[1, 3] GHZ
The transmission rates of cellular network	[2, 3] Mb/s
The transmission rates of D2D communication	[0.8, 1,5] Mb/s

Proof: $\forall\tau$ be a positive integer, we obtain the upper bound of $N^T_{m_1}$ based on [34], expressed as follows:

$$\tau + \sum_{t=1}^{\infty} \sum_{N^t_{*_1}=1}^{t-1} \sum_{N^t_{m_1}=1}^{t-1}$$

$$\mathbb{I}\left\{\bar{W}^t_{*_1,n} + \sqrt{\frac{2\ln t}{N^t_{*_1}}} \le \bar{W}^t_{m_1,n} + \sqrt{\frac{2\ln t}{N^t_{m_1}}}\right\}. \tag{24}$$

We apply *Chernoff–Hoeffding bound* to (24), and we obtain the following two inequalities:

$$\mathbb{P}\left\{\bar{W}^t_{*_1,n} \le \nu^{*_1} - \sqrt{\frac{2\ln t}{N^t_{*_1}}}\right\} \le t^{-4} \tag{25}$$

$$\mathbb{P}\left\{\bar{W}^t_{m_1,n} \ge \nu_i + \sqrt{\frac{2\ln t}{N^t_{m_1}}}\right\} \le t^{-4} \tag{26}$$

Summarizing the above analysis, we have

$$\mathrm{WR}^T = \sum_{\nu_{m_1}<\nu^{*_1}} \left(\frac{8\ln T}{(\nu^{*_1}-\nu_{m_1})^2} + 1 + \frac{\pi^2}{3}\right), m_1 \in \mathcal{M}_{\mathrm{SeD}}. \tag{27}$$

Similar to the analysis of WR^t, we obtain DR^T

$$\mathrm{DR}^T = \sum_{\mu_{m_2}<\mu^{*_2}} \left(\frac{8\ln T}{(\mu^{*_2}-\mu_{m_2})^2} + 1 + \frac{\pi^2}{3}\right), m_2 \in \mathcal{M}_{\mathrm{cand}}. \tag{28}$$

By adding (27) and (28), we can obtain the total learning regret as shown in (23).

V. PERFORMANCE EVALUATION

In this section, we conduct simulations to evaluate the proposed learning-based task co-offloading algorithm.

A. Simulation Setup

We conduct simulations on a desktop computer with an 11th Gen Intel(R) Core(TM) i7-11700F @2.50 GHz, 16 GB memory and Win10 OS. The parameter settings are listed, shown in Tabel II.

We compare the proposed learning-based task co-offloading algorithm with the following methods: 1) Task co-offloading with global information (GI): Without learning, this co-offloading algorithm enables task offloading directly based on global offline information [36]. 2) Task offloading in MEC networks (MEC): Computing-intensive tasks can only be offloaded to MEC servers under this scheme [7]. 3) Task offloading in D2D networks (D2D): Computing-intensive tasks can only be offloaded to the nearby SeDs with surplus computing resources via D2D communication [16].

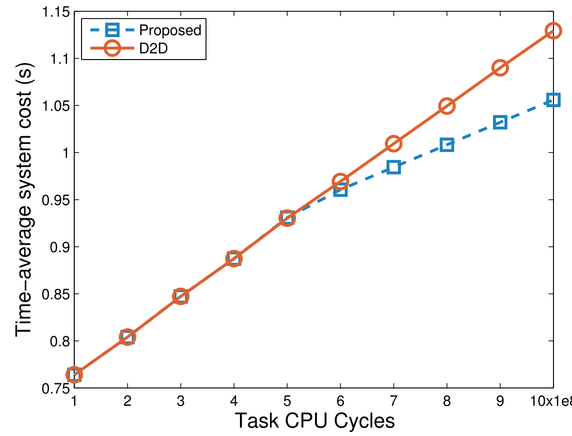

Fig. 2. Performance under different task computation demands.

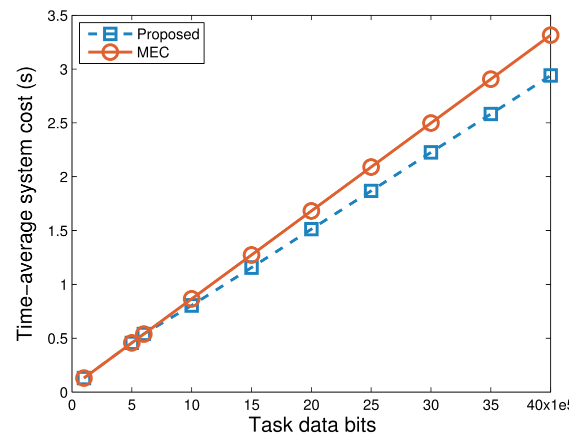

Fig. 3. Performance under different task data bits.

B. Comparison Analysis

1) Comparison Analysis on Task Computation Demands: Fig. 2 shows the comparison between the proposed algorithm and the D2D algorithm under different task computation demands. When the computation demands are small, these two algorithms have similar performance. As computation demands grow, the proposed algorithm incurs less system cost than that of D2D. This result demonstrates that tasks with large computation demands are inclined to offload to MEC servers to seek less computation delay.

2) Comparison Analysis on Task Data Bits: Fig. 3 shows the comparison between the proposed algorithm and the MEC algorithm under different task data bits. From the results, we can find that a task is offloaded to the MEC server when the task data bits are small. In this case, the proposed algorithm and

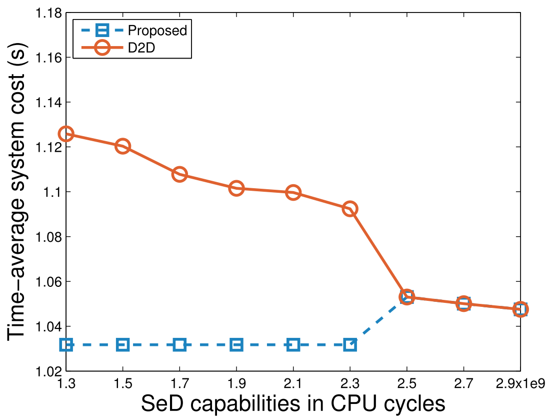

Fig. 4. Performance under different SeV capabilities in CPU cycles.

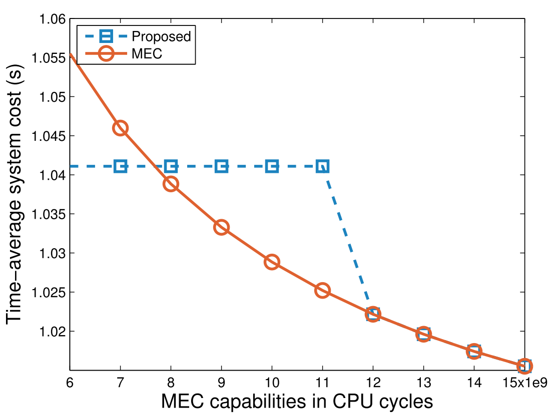

Fig. 5. Performance under different MEC capabilities in CPU cycles.

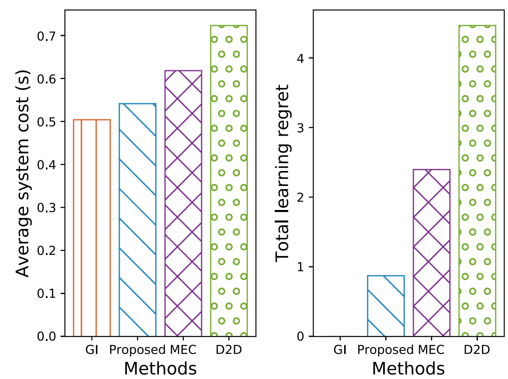

Fig. 6. Performance under different methods.

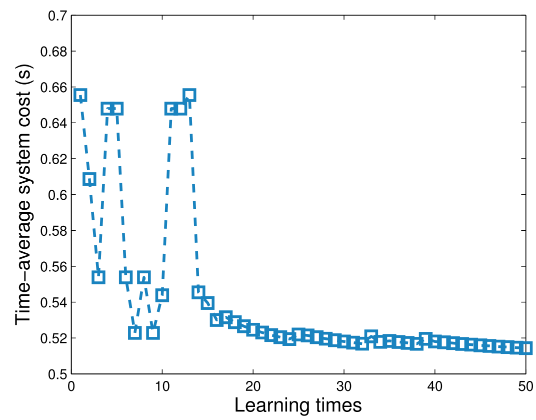

Fig. 7. Performance under different learning times.

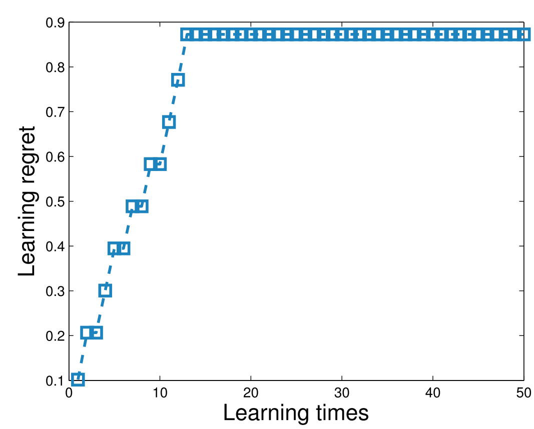

Fig. 8. Regret under different learning times.

MEC algorithm have the same system cost. However, when data bits enlarges, MEC offloading inevitably incurs excessive queue delay, and therefore our proposed algorithm intends to seek D2D offloading to avoid network congestion.

3) Comparison Analysis on SeV Capabilities: Fig. 4 shows the comparison between the proposed algorithm and the D2D algorithm under different SeV capabilities in CPU cycles. When SeV capabilities are small, UDs tend to offload tasks to MEC servers rather than SeDs. As such, the proposed co-offloading algorithm performs much better than the D2D algorithm. Because SeD capabilities do no affect MEC offloading, the performance of the co-offloading algorithm keeps fixed in this case. As SeD capabilities grow, D2D offloading achieves less computation delay and thus reduces system cost. When the SeD capabilities add up to 2.5 GHz, UDs will offload tasks to SeDs in our simulations.

4) Comparison Analysis on MEC Capabilities: Fig. 5 shows the comparison between the proposed algorithm and the MEC algorithm under different MEC capabilities in CPU cycles. For the MEC algorithm, its system cost decreases with MEC capabilities growing. Since the proposed algorithm enables co-offloading, it seeks D2D offloading when MEC capabilities are low; while it pursues MEC offloading when MEC capabilities enlarge. Correspondingly, the proposed algorithm's curve is invariable in D2D offloading and keeps the same as the MEC algorithm with increasing MEC capabilities.

5) Comparison Analysis on Different Methods: Fig. 6 shows the performance comparison between different methods. Since GI has global offline offloading information, GI maintains the optimal offloading decisions and produces no learning regret. Compared with MEC and D2D, our proposed co-offloading algorithm shows superiority in system cost and learning regret. This is because the proposed co-offloading scheme enables to adjust offloading decisions dynamically based on task features in computation demands and data bits.

6) Comparison Analysis on Learning Times: Figs. 7 and 8 show the impact of learning times on system performance and learning regret. Our proposed algorithm is implemented via "offloading while learning." The optimal edge node cannot be obtained before task offloading, and the offloading efficiency of each edge node is learned based on its system cost. From the results presented in Fig. 7, we find that when learning times are small, the proposed algorithm suffers from bad performance. As learning times grow, the optimal edge node can be selected. Correspondingly, the learning regret converges to a fixed value after finding out the optimal edge node as shown in Fig. 8. In our simulation, our proposed learning-based algorithm gradually converges after 12 times of learning.

VI. Conclusion

In this article, we investigated industrial task co-offloading in D2D-assisted MEC networks. Specifically, computing-intensive industrial tasks can be jointly served by MEC and D2D offloading, thereby achieving small computation delay meanwhile resolving the network congestion problem. To this end, we perform a co-offloading framework in D2D-assisted MEC networks. In this framework, we consider migration cost caused by erratic movements of IIoT devices, and design a willingness metric to stimulate resource sharing among IIoT devices. Beyond that, we investigate a learning-based task co-offloading algorithm for minimal task delay and migration cost. The proposed algorithm enables IIoT devices to observe and learn the system cost from the candidate edge nodes, thereby selecting the optimal edge node. Since the proposed algorithm does not require complete offloading information, it is easily implemented in the real world. Furthermore, we carry out simulations to evaluate the performance of the proposed algorithm. The results demonstrate that the proposed algorithm conducts a better performance compared with that of other algorithms, in various parameters, such as task computation demands, data bits, and learning times. In the future, we will extend our work by taking privacy protection into consideration for co-offloading in D2D-assisted MEC networks.

References

[1] Z. Xu *et al.*, "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2066–2075.

[2] Y. Siriwardhana, P. Porambage, M. Liyanage, and M. Ylianttila, "A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects," *IEEE Commun. Surv. Tut.*, vol. 23, no. 2, pp. 1160–1192, Apr./Jun. 2021.

[3] A. Rahman, J. Jin, A. L. Cricenti, A. Rahman, and A. Kulkarni, "Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 2500–2511, May 2019.

[4] H. Jiang, X. Dai, Z. Xiao, and A. K. Iyengar, "Joint task offloading and resource allocation for energy-constrained mobile edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3150432.

[5] D. Chatzopoulos, C. Bermejo, E. U. Haq, Y. Li, and P. Hui, "D2D task offloading: A dataset-based Q and A," *IEEE Commun. Mag.*, vol. 57, no. 2, pp. 102–107, Feb. 2019.

[6] San Jose, USA CA, White Paper C11-741490-01, "Cisco Annual Internet Report (2018-2023)," Website, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

[7] X. Ma, A. Zhou, S. Zhang, and S. Wang, "Cooperative service caching and workload scheduling in mobile edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 2076–2085.

[8] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3064847.

[9] G. Yang, L. Hou, X. He, D. He, S. Chan, and M. Guizani, "Offloading time optimization via markov decision process in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2483–2493, Feb. 2021.

[10] C.-L. Chen, C. G. Brinton, and V. Aggarwal, "Latency minimization for mobile edge computing networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3117511.

[11] T. Liu, Y. Zhang, Y. Zhu, W. Tong, and Y. Yang, "Online computation offloading and resource scheduling in mobile-edge computing," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6649–6664, Apr. 2021.

[12] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1787–1796, Mar. 2021.

[13] P. Lai *et al.*, "Quality of experience-aware user allocation in edge computing systems: A potential game," in *Proc. IEEE 40th Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 223–233.

[14] Z. Zhou *et al.*, "Learning-based URLLC-Aware task offloading for internet of health things," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 396–410, Feb. 2021.

[15] Y. Cheng, C. Liang, Q. Chen, and R. Yu, "Energy-efficient D2D-Assisted computation offloading in NOMA-Enabled cognitive networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13441–13446, Dec. 2021.

[16] I. Budhiraja, S. Tyagi, S. Tanwar, N. Kumar, and J. J. P. C. Rodrigues, "DIYA: Tactile internet driven delay assessment NOMA-Based scheme for communication," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6354–6366, Dec. 2019.

[17] H. Zhou, T. Wu, H. Zhang, and J. Wu, "Incentive-driven deep reinforcement learning for content caching and D2D offloading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2445–2460, Aug. 2021.

[18] M. Hamdi, A. B. Hamed, D. Yuan, and M. Zaied, "Energy-efficient joint task assignment and power control in energy harvesting D2D offloading communications," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3110319.

[19] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[20] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-Assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[21] Z. Ning *et al.*, "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, Jan. 2021.

[22] Y. Yang, C. Long, J. Wu, S. Peng, and B. Li, "D2D-Enabled mobile-edge computation offloading for multiuser IoT network," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12490–12504, Aug. 2021.

[23] Y. Sun, S. Zhou, and Z. Niu, "Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1138–1151, Feb. 2021.

[24] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 976–986, Feb. 2019.

[25] J. Tang *et al.*, "Energy minimization in D2D-Assisted cache-enabled Internet of Things: A. deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5412–5423, Aug. 2020.

[26] M. Sun, X. Xu, Y. Huang, Q. Wu, X. Tao, and P. Zhang, "Resource management for computation offloading in D2D-Aided wireless powered mobile-edge computing networks," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 8005–8020, May 2021.

[27] Y. He, J. Ren, G. Yu, and Y. Cai, "D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 18, no. 3, pp. 1750–1763, Mar. 2019.

[28] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960–1972, Mar. 2022.

[29] G. Li and J. Cai, "An online incentive mechanism for collaborative task offloading in mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 624–636, Jan. 2020.

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

[31] Z. Xiao *et al.*, "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052, Mar. 2020.

[32] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in *Proc. IEEE Conf. Comput. Commun.*, 2019, pp. 2467–2475.

[33] O. Tao, X. Chen, Z. Zhou, L. Li, and X. Tan, "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2021.3106746.

[34] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002.

[35] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang, and C. Pan, "Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4051–4063, Jul. 2021.

[36] C. Liu, K. Liu, S. Guo, R. Xie, V. C. S. Lee, and S. H. Son, "Adaptive offloading for time-critical tasks in heterogeneous Internet of Vehicles," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7999–8011, Sep. 2020.

Xingxia Dai received the B.S. degree in communication engineering from Xiangtan University, Xiangtan, China, in 2018. She is currently working toward the Ph.D. degree in computer science and technology with Hunan University, Changsha, China.

Her current research interests include internet of vehicles and mobile edge computing.

Zhu Xiao (Senior Member, IEEE) received the M.S. and Ph.D. degrees in communication and information system from Xidian University, China, in 2007 and 2009, respectively.

From 2010 to 2012, he was a Research Fellow with the Department of Computer Science and Technology, University of Bedfordshire, Bedfordshire, U.K. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Hunan, China. His research interests include mobile communications, wireless localization, Internet of Vehicles, and trajectory data mining.

Hongbo Jiang (Senior Member, IEEE) received the Ph.D. degree in computer science from Case Western Reserve University, Cleveland, OH, USA, in 2008.

He is currently a Full Professor with the College of Computer Science and Electronic Engineering, Hunan University, Zhuzhou, Hunan, China. He ever was a Professor with the Huazhong University of Science and Technology. His research interests include computer networking, especially algorithms and protocols for wireless and mobile networks.

Prof. Jiang was the Editor for the IEEE/ACM TRANSACTIONS ON NETWORKING, an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, and an Associate Technical Editor for the IEEE COMMUNICATIONS MAGAZINE. He is an elected Fellow of IET, Fellow of BCS.

Mamoun Alazab (Senior Member, IEEE) received the Ph.D. degree in computer science from the School of Science, Information Technology, and Engineering, Federation University, Ballarat, Australia, in 2012.

He is currently an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Australia. He is a Cyber Security Researcher and a Practitioner with industry and academic experience. He has more than 200 research papers in many international journals and conferences. His research is multidisciplinary that focuses on cyber security and digital forensics of computer systems with a focus on cybercrime detection and prevention.

Dr. Alazab is the Founding Chair of the IEEE Northern Territory (NT) Subsection.

John C. S. Lui (Fellow, IEEE) was born in Hong Kong. He received the Ph.D. degree in computer science from the University of California, Los Angeles, CA, USA, in 1992.

He is currently the Choh-Ming Li Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK), Hong Kong. He was the Chairman of the department from 2005 to 2011. His current research interests are in communication networks, network/system security (e.g., cloud security, mobile security, etc.), network economics, network sciences (e.g., online social networks, information spreading, etc.), cloud computing, large-scale distributed systems, and performance evaluation theory.

Prof. Lui is an elected member of the IFIP WG 7.3, a Fellow of the Association for Computing Machinery (ACM), a Senior Research Fellow of the Croucher Foundation, and was the Chair of the ACM SIGMETRICS. He has been serving in the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, Performance Evaluation, and the International Journal of Network Security. He was a recipient of various departmental teaching awards and the CUHK Vice-Chancellors Exemplary Teaching Award. He is also a co-recipient of the Best Paper Award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP NOMS 2006, and SIMPLEX 2013

Schahram Dustdar (Fellow, IEEE) received the Ph.D. degree in business informatics from the University of Linz, Linz, Austria, in 1992.

He is currently a Full Professor of computer science (informatics) with a focus on internet technologies heading the Distributed Systems Group, TU Wien, Wein, Austria. He has been the Chairman of the Informatics Section of the Academia Europaea, since December 2016.

Prof. Dustdar has been a member of the IEEE Conference Activities Committee (CAC), since 2016, the Section Committee of Informatics of the Academia Europaea, since 2015, and the Academia Europaea: The Academy of Europe, Informatics Section, since 2013. He was a recipient of the ACM Distinguished Scientist Award in 2009 and the IBM Faculty Award in 2012. He is an Associate Editor for the IEEE TRANSACTIONS ON SERVICES COMPUTING, *ACM Transactions on the Web*, and *ACM Transactions on Internet Technology*. He is on the Editorial Board of IEEE.

Jiangchuan Liu (Fellow, IEEE) received the B.Eng. degree (*cum laude*) from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2003, both in computer science.

He is currently a University Professor with the School of Computing Science, Simon Fraser University, British Columbia, Canada. In the past he worked as an Assistant Professor with The Chinese University of Hong Kong, Hong Kong, and as a Research Fellow with Microsoft Research Asia. His research interests include multimedia systems and networks, cloud and edge computing, social networking, online gaming, and Internet of things/RFID/backscatter.

Prof. Liu is a Fellow of The Canadian Academy of Engineering, and an NSERC E.W.R. Steacie Memorial Fellow. He was an EMCEndowed Visiting Chair Professor of Tsinghua University (2013–2016). He is a corecipient of the inaugural Test of Time Paper Award of IEEE INFOCOM (2015), ACM SIGMM TOMCCAP Nicolas D. Georganas Best Paper Award (2013), and ACM Multimedia Best Paper Award (2012). He has served on the editorial boards of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON BIG DATA, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, AND IEEE INTERNET OF THINGS JOURNAL. He is a Steering Committee member of IEEE Transactions on Mobile Computing and Steering Committee Chair of IEEE/ACM IWQoS (2015-2017). He is TPC Co-Chair of IEEE INFOCOM'2021.