This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

Edge Intelligence-Driven Joint Offloading and Resource Allocation for Future 6G Industrial Internet of Things

Yongkang Gong, *Student Member, IEEE,* Haipeng Yao, *Senior Member, IEEE,*
Jingjing Wang, *Senior Member, IEEE,* Maozhen Li, *Member, IEEE,* and Song Guo, *Fellow, IEEE*

***Abstract*—The sixth generation mobile networks (6G) will undergo an unprecedented transformation to revolutionize the wireless system evolution from connected things to connected intelligence, where future 6G Industrial Internet of Things (IIoT) covers a range of industrial nodes such as sensors, controllers, and actuators. Additionally, data scattered around the industrial environments can be collected for the sake of enabling intelligent operations. In our work, the promising multi-access edge computing (MEC) service is introduced into the IIoT system to execute the task scheduling and resource allocation for the sake of various compelling applications. Moreover, we define the objective function as the weighted sum of delay and energy consumption. Next, a novel deep reinforcement learning (DRL)-based network structure is proposed to jointly optimize task offloading and resource allocation. More specifically, the task offloading is decomposed via the new isotone action generation technique (2AGT) and adaptive action aggregation update strategy (3AUS) based on the proposed DRL framework, and the initial problem can be transformed into a convex optimization problem to solve the resource allocation for each IIoT device. Additionally, we periodically renovate the offloading policy in the DRL framework so that our proposed DRL-based decision-making algorithm can better accommodate time-varying environments. Numerous experimental results demonstrate our proposed DRL-based network structure for each IIoT device can obtain quasi-optimal system performance compared with some conventional baseline algorithms.**

***Index Terms*—The sixth generation mobile networks (6G), edge intelligence, Industrial Internet of Things (IIoT), task offloading, resource management.**

Y. Gong and H. Yao are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China. Email: {yongkanggong@bupt.edu.cn, yaohaipeng@bupt.edu.cn}.

J. Wang is with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China. Email: drwangjj@buaa.edu.cn.

M. Li is with the Department of Electronic and Computer Engineering, Brunel University, London, Uxbridge UB8 3PH, U.K. Email: maozhen.li@brunel.ac.uk.

S. Guo is with the Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Hong Kong. E-mail: song.guo@polyu.edu.hk.

This work is partially supported by the National Key Research and Development Plan under grant 2018YFB1800805, partially supported by the Director Foundation Project of National Engineering Laboratory for Public Safety Risk Perception and Control by Big Data (PSRPC), Future Intelligent Networking and Intelligent Transportation Joint Laboratory (BUPT-CTTIC), National Nature Science Foundation of China (Grant Nos. 61922050), General Research Fund of the Research Grants Council of Hong Kong (PolyU 152221/19E) and the National Natural Science Foundation of China (Grant61872310). This work of Dr. Wang was supported by the Young Elite Scientist Sponsorship Program by CAST (Grant No. 2020QNRC001). (Corresponding author: Haipeng Yao)

I. Introduction

RECENTLY, the efforts and initiatives from standard bodies have started to conceptualize the sixth generation mobile networks (6G) [1] and 6G may become an unparalleled transformation to revolutionize the wireless communication systems. Furthermore, intelligent industrial Internet of things (IIoT) in 6G [2] has received considerable attention from both academic and industrial field. In the last decade, the industrial standards and infrastructures have evolved substantially due to the amalgamation of Internet of things (IoT) [3] paradigm with some industrial units and equipment. Sometimes, IIoT is also known as the "Internet of really important stuff, the objects, and machines that powers our life". There may be massive devices connected by the IoT at the end of 2020 [4]. Furthermore, the connected IIoT infrastructures (e.g., actuators, vehicles, and industrial controllers) generate a humongous amount of data that require real-time analysis and evaluations with heterogenous characteristics in terms of size and modes [5]. Specifically, IIoT connects different kinds of industrial assets in industrial environments to enable intelligent operations, such as industrial monitoring, automation and intelligent control [6]. However, the proliferation of the number of IIoT devices and the ever-increasing computation-intensive applications including augmented reality (AR), real-time online gaming and ultra-high-definition (UHD) pose great challenges on data processing, architecture rigidity and resource allocation. To address the aforementioned challenges, it is important to analyse data. Consequently, joint task offloading and resource management have attracted the significant focus from IIoT systems [7] - [8].

Generally, mobile cloud computing (MCC) provides a proper paradigm that wireless devices execute the computation offloading in the cloud server [9] - [10]. Thus, some wireless devices choose to offload application tasks [11] to the remote cloud server to ameliorate computational velocity, spectrum efficiency and energy efficiency, which can be utilized widely in the past decade. Nevertheless, due to remote distance between the cloud server and local wireless devices, the long propagation delay, limited channel capacity and task queuing delay make it hard to process latency-critical and computation-intensive application tasks relying on centralized methods.

To address the complex problems, the concept of edge intelligence [12] is conceived for offering powerful computational processing and massive data acquisition at the edge

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

networks. Specifically, the edge server is closer to wireless devices, and hence the offloading scheme for computing tasks can enormously decrease transmission delay and save backhaul bandwidth between cloud servers and wireless devices [13]. At the same time, artificial intelligence (AI) is a promising trend for extracting information from large-scale data and for making efficient resource scheduling strategies in complex environment. By integrating AI into edge networks, the radio networks with service and resource awareness can dynamically adapt to the resource orchestration, which can be viewed as a beneficial remedy for data processing and resource allocation issues [14] in complex IIoT environment. To elaborate a little further, multi-access edge computing (MEC) servers can help alleviate latency, reduce energy consumption and guarantee the quality of experience (QoE).

In this paper, we intend to solve the above problems. First of all, we establish a novel IIoT network model which contains multiple IIoT devices followed with the MEC server. Each IIoT device has many application tasks to be executed. During the executed process, each device chooses to execute the task locally or dynamically offload to the MEC server. We propose and solve the network model to obtain better offloading decision and resource allocation schemes, which can enable the whole system to obtain lower total cost of energy consumption and task delay for each IIoT device receiving different application tasks. Furthermore, we compare our proposed deep reinforcement learning (DRL)-based network structure with some benchmarks under the fixed offloading modes and conventional intelligent offloading schemes. To the best of our knowledge and belief, there are few related works considering the IIoT model integrated with edge intelligence. With multiple application tasks to acquire better system performance, our works contain offloading decision and resource allocation utilizing AI methods.

In summary, the main contributions of our paper can be summarized as follows.

- We construct a novel IIoT system model including the MEC server for different application tasks in 6G. Driven by edge intelligence, we jointly optimize the offloading decision-making and transmission resource allocation for each device, which aims at minimizing the system's total cost in terms of energy consumption and delay.
- A DRL-aided algorithm is proposed for the sake of generating offloading actions under different application tasks. Moreover, isotone action generation technique (2AGT) is utilized to quantize the actions. To stabilize the generated policy and reduce over-fitting, experience replay as well as stochastic sampling are introduced for the sake of retraining the DRL agent, followed by an adaptive action aggregation update strategy (3AUS) for reducing the computational complexity.
- Simulation results are shown to demonstrate the effectiveness of our proposed DRL-based algorithm in the IIoT system with the MEC server. Numerical results show that our proposed approach plays a key role both in improving convergent performance and achieving high utility compared with other baseline schemes.

The structure of this paper is given as follows. In Section II, the related works are reviewed. Section III discusses the network model of our proposed IIoT system enhanced by edge intelligence. In Section IV, we propose the DRL-based network structure followed with 2AGT and 3AUS schemes. Extensive simulations and experimental results are illustrated in Section V and we compare the proposed DRL-based strategy with some benchmarks. Finally, Conclusions are given in Section VI.

II. Related Works

A. IIoT and Mobile Cloud System

MCC is regarded as a powerful computation server that promotes some IIoT devices (e.g., actuators, sensors, and controllers) to deliver task data to cloud providers. To adequately explore the potential of cloud servers, some researchers propose a range of solutions about offloading schemes and resource allocation for wireless devices. Satyanarayanan *et al*. [15] proposed a virtual machine-based structure to efficiently process task requests from wireless devices. Liu *et al*. [16] considered an task scheduling and routing strategy. Further, the authors in [17] - [18] implemented two cloud system models and structures and gave some code offloading methods.

B. Offloading Schemes in MEC

Currently, there are still some technical challenges to be solved. First of all, when wireless devices obtain application tasks, whether to offload application tasks needs to be studied carefully. When global tasks are executed in the MEC server, this may cause huge congestion on the uplink wireless channels [19] and lead to severe delay. To substantially explore and exploit the computation offloading in IIoT systems, we should consider the task scheduling [20] and uplink resource management [21] for the sake of reducing the total task delay and energy consumption.

To elaborate a little further, Kumar *et al*. [22] proposed a general rule in offloading decision for minimizing energy consumption, where uplink communication links were assumed to have a fixed transmission rate. An optimal binary offloading scheme was proposed in [23] by Barbarossa *et al*. relying on convex optimization technologies, where this defined the power-rate function with the aid of Shannon-Hartley formulas. At the same time, Chen *et al*. [24] illustrated a distributed offloading scheme in the context of multi-user systems based on game theory, which solved the multi-user computation offloading on account of large iterations. Furthermore, Bi *et al*. [25] proposed a novel iteration update algorithm to solve the computation resource allocation with binary offloading. Additionally, by slacking the binary variable to real value, Zhang *et al*. [26] proposed a novel joint offloading and resource allocation optimization (JORAO) algorithm to minimize the total cost of mobile users, which iteratively reduced the energy consumption and completion time. Besides, a separable semi-definite relaxing scheme was proposed in [27] by Chen *et al*. to minimize overall energy consumption. However, computational complexity of aforementioned algorithms is extremely high for real-time computation offloading and resource allocation,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

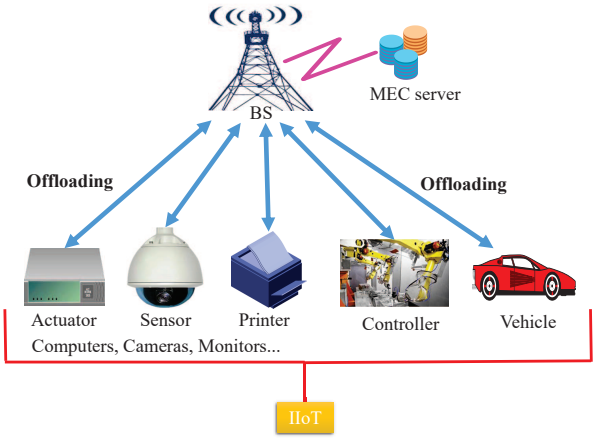

Fig. 1: The service scenario for IIoT.

and it is difficult to process some latency-critical application tasks.

C. Edge Intelligence: AI-enabled Strategy

To tackle this problem, AI-enabled strategies have been considered as efficient solutions [28]. Specifically, Zhang *et al*. [29] presented an optimal offloading algorithm in an intermittently connected cloudlet system based on markov decision process (MDP). Besides, Liu *et al*. [30] proposed a delay-optimal task scheduling policy, where application tasks were scheduled relying on the state of task buffer, local processing unit and transmission unit. However, it was difficult to construct a precise MDP model in the actual world. Without a precise model, a reinforcement learning (RL) algorithm was proposed in [31] by Huang *et al*. to replace MDP, whereas it was complicated to store state-action value tabular in high-dimensional space. Besides, due to the expansion of network size and massive application tasks, the RL agent is limited for solving high-dimensional problems, which can cause the curse of dimensionality. Fortunately, with the development of deep neural network (DNN) [32], [33], [34], some DRL algorithms combining DNN with RL are regarded as efficient solutions for real-time control decisions [35], [36], [37], [38] and they can be performed to study characteristics from high-dimensional data and solve the curse of dimensionality.

III. System Model

Fig. 1 shows the network model that consists of multiple IIoT devices (IIoTD) and one BS with the MEC server. In this network model, IIoTD can be defined by $\mathbb{N} = \{1, 2, ...N\}$. Besides, the MEC server and the BS are connected with a wired connection (e.g., optical fiber), in which transmission delay between them can be ignored significantly [39]. Each IIoTD has large numbers of application tasks to be processed locally or offloaded to the MEC server with BS. Without loss of generality, assuming that there are Z independent tasks, denoted by $\mathbb{Z} = \{1, 2, ...Z\}$, and the computation of each task could not be split for partial offloading or partial computing. That is to say, the task can only be computed in local IIoTD or offloaded to the MEC server but not both. We assume that $l_{n,z}$ is the task size including programming codes and general parameters, and $l_{n,z}$ is the z^{th} task of the n^{th} IIoTD. These parameters are related to features of the task and they can be estimated through task types. Each IIoTD n can choose whether to offload its own computation-intensive task z to the MEC server or not. We define the offloading decision vector A, which can be given by

$$A = [a_{1,1}, a_{1,2}, ...a_{n,z}, ...a_{N,Z}] \ , \tag{1}$$

where $n \in \{1, 2, ...N\}$ and $z \in \{1, 2, ...Z\}$ represent the IIoTD and task, respectively. $a_{n,z}$ represents the offloading decision and it belongs to $\{0, 1\}$. In detail, $a_{n,z} = 1$ represents the IIoTD n chooses to offload the task z to the MEC server, and $a_{n,z} = 0$ means that IIoTD n decides to carry out the task z locally. In this way, we can take advantage of parallel computing of IIoTD and MEC servers, which results in a decrease of total delay and energy consumption. We consider $B_{n,z}$ as the optimized wireless channel bandwidth of the z^{th} task of the n^{th} IIoTD. Due to the fact that there only exists one BS, so the interval interference between the BS could be overlooked [40]. Simultaneously, we consider that the assigned channel of application tasks from each agent is orthogonal each other in the IIoT system model through orthogonal frequency division multiple access (OFDMA). As there is only one BS coverage area and nature characteristics of OFDM, we ignore the co-channel frequency and adjacent channel interference. In terms of Shannon's theory [41], the achievable uplink transmission rate for each task z of the n^{th} IIoTD can be obtained by

$$c_{n,z} = B_{n,z} \log_2 \left(1 + \frac{P^n_{tran} |h_{n,z}|^2}{\sigma^2}\right) \ , \tag{2}$$

where P^n_{tran} means the transmission power from the IIoTD, and $h_{n,z}$ represents the channel gain which follows rayleigh flat fading under the allocated channel bandwidth. σ^2 is noise.

A. Computation Offloading Mode

The MEC server starts to process the task $l_{n,z}$ after it has fully received the IIoTD's task and feeds back information after the entire task z is computed [27]. Because the data size of the feedback message is small in general [42], the feedback energy consumption and delay can be neglected. Subsequently, we formulate transmission time and processing time. Specifically, for the task z of the IIoTD n, the transmission time caused by the uplink channel can be described as

$$T^{n,z}_{\text{tran}} = \frac{a_{n,z} l_{n,z}}{c_{n,z}}. \tag{3}$$

Similarly, the transmission energy consumption for the task z of the IIoTD n can be denoted by

$$E^{n,z}_{\text{tran}} = T^{n,z}_{tran} P^n_{tran}. \tag{4}$$

The computation time at the MEC server via the BS can be represented by

$$T^{n,z}_{pro} = \frac{a_{n,z} l_{n,z} e_{n,z}}{F^{total}_{server}}, \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

where $e_{n,z}$ represents the number of required CPU cycles, and F_{server}^{total} denotes the computational power. At the same time, we assume that MEC server is pretty powerful and can process all received application tasks concurrently. Furthermore, we model the computational processing energy consumption as the linear function $l_{n,z}$ [43], and it can be written as

$$E_{com}^{n,z} = \beta l_{n,z}, \tag{6}$$

where β can be defined as the task weight factor relative to the computational energy consumption from the MEC server and the unit of β is joule per bit. It depends on the application task size and diverse MEC servers. So the total delay can be denoted as

$$T_n^s = \sum_{z=1}^{Z} \left(T_{tran}^{n,z} + T_{pro}^{n,z}\right). \tag{7}$$

In addition, the total energy consumption at the computation offloading mode for each IIoTD n is formulated as

$$E_n^s = \sum_{z=1}^{Z} (E_{tran}^{n,z} + E_{com}^{n,z}). \tag{8}$$

B. Local Computing Mode

Next, we formulate the case that each IIoTD decides to execute its task locally. Specifically, the processing energy consumption b_n^l can be represented as

$$b_n^l = k\left(f_n^l\right)^2, \tag{9}$$

where f_n^l is the CPU-cycle frequency for each IIoTD, and k is a constant interrelated to the hardware performance. So the local processing energy consumption for task z of each IIoTD n can be given by

$$E_{local}^{n,z} = (1 - a_{n,z})\, l_{n,z} b_n^l. \tag{10}$$

At the same time, the local processing time can be defined by

$$T_{local}^{n,z} = \frac{l_{n,z} e_{n,z} (1 - a_{n,z})}{f_n^l}. \tag{11}$$

Thus, given the task offloading choice $a_{n,z}$, the total local processing delay for each IIoTD n can be denoted as

$$T_n^l = \sum_{z=1}^{Z} T_{local}^{n,z}. \tag{12}$$

Meanwhile, the total local processing energy consumption can be depicted as

$$E_n^l = \sum_{z=1}^{Z} E_{local}^{n,z}. \tag{13}$$

C. Problem Formulation

In this section, we formulate the joint offloading decision and wireless transmission rate allocation for IIoT system with MEC server as a multi-objective optimization problem. To minimize the total delay and energy consumption, the total cost function $V(L, A, C)$ can be defined as the weighted sum of the task delay and energy consumption, where $L = [l_{1,1}, l_{1,2}, ...l_{n,z}, ...l_{N,Z}]$ and $C = [c_{1,1}, c_{1,2,...}c_{n,z}, ...c_{N,Z}]$. Hence, the total cost function can be denoted as

$$V(L, A, C) = \{\lambda \sum_{n=1}^{N} (T_n^l + T_n^s) + \sum_{n=1}^{N} (E_n^l + E_n^s)\}\ , \tag{14}$$

where λ represents the weight on delay relative to total energy consumption. The unit of λ is joule per second and we can adjust λ to attach different importance to delay and energy consumption for various application tasks. In short, the optimization objective can be expressed as

$$\min_{\{L,A,C\}} V(L, A, C), \tag{15}$$

$$s.t. \quad a_{n,z} \in \{0, 1\}, \tag{16}$$

$$\sum_{n=1}^{N} \sum_{z=1}^{Z} c_{n,z} \leq C_{total}, \tag{17}$$

$$c_{n,z} > 0. \tag{18}$$

Then, we minimize the total cost function $V(L, A, C)$ via choosing the optimal offloading decision vector A and allocating uplink wireless transmission rate vector C under different application tasks L. In fact, L is one state constant vector in the optimization problem and it is diverse in different time slots for network environments. Furthermore, we regard it as the input state vector in the optimization problem. At the same time, decision vector A and transmission rate vector C are considered as two optimized variables. Next, there exist some constraints about minimizing the total cost function $V(L, A, C)$. (16) means offloading decision belongs to 0 or 1, which represents the task is executed locally or the task is offloaded to the MEC server. As the total wireless channel bandwidth is limited for all IIoTD, (17) means that the sum of the achievable transmission rate allocated for each task z must not exceed the maximum C_{total}. Additionally, wireless transmission rate $c_{n,z}$ is closely related to the required channel bandwidth $B_{n,z}$, so optimizing the transmission rate $c_{n,z}$ is equivalent to solving the optimal channel bandwidth $B_{n,z}$. (18) indicates the achievable transmission rate should be positive since each IIoTD is supposed to access the BS with MEC server to guarantee the QoE. In the next section, we show an effective and efficient algorithm based on DRL to resolve this problem. The detailed parameter settings are described in TABLE I.

IV. System Optimization

In this section, we firstly introduce a novel DRL-based framework to solve our proposed problems. Then we present the detailed process of how to generate required offloading policy through the DRL framework and give the 2AGT to approximate offloading action $a_{n,z}$. Next, after obtaining the initial offloading action, we transform our proposed initial problem into a convex objective problem and calculate the optimal offloading action by parallel computing in terms of substantial application tasks. Besides, we provide a 3AUS

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

TABLE I: Summary of Notations

Notation	Description
L	application task vector
A	offloading decision vector
C	transmission rate vector
N	the number of IIoTD
Z	the number of independent application tasks for each IIoTD
$l_{n,z}$	the task size for z^{th} task of n^{th} IIoTD
$a_{n,z}$	the offloading decision for z^{th} task of n^{th} IIoTD
$B_{n,z}$	the optimized wireless channel bandwidth for z^{th} task of n^{th} IIoTD
P^n_{tran}	transmission power from the IIoTD
$h_{n,z}$	channel gain
σ^2	the variance of AWGN channel
$c_{n,z}$	uplink transmission data rate for z^{th} task of n^{th} IIoTD
F^{total}_{server}	MEC server computational capacity
$e_{n,z}$	the number of cycles required to process each task bit
β	the task weight factor relative to computational energy consumption
k	a constant interrelated to the hardware performance
f^l_n	CPU-cycle frequency for each IIoTD
λ	the delay weight factor relative to the total energy consumption
π	the DRL agent policy function
w	neural network parameter
K	a distance parameter related to the activation function
M	the number of action aggregations
P	incremental constant in optimal action aggregations

method for setting the number of action aggregations parameter. At the same time, we introduce a DRL network parameter update policy to strengthen the network stability and reduce the over-fitting. The detailed procedures are just shown as follows.

A. Deep Reinforcement Learning Framework

As shown in Fig. 2, the DRL agent brings the control strategy via interacting with the network environment (e.g., the application tasks) without a precise transition probability and adjusts own behavior depending on the outcomes of actions

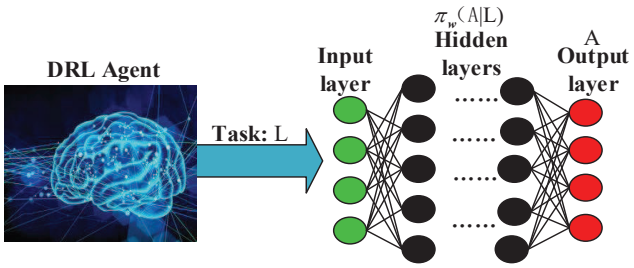

Fig. 3: The detailed agent internal structure.

[44] in order to maximize the discounted reward functions. So DRL means a new exemplification through trial-and-error and delayed incentive mechanism to achieve an optimal behavior policy [45]. While obtaining the application tasks, our DRL network framework contains offloading decision generation, 2AGT optimization and 3AUS parameters settings. At the same time, the network framework can compute the convex optimization problem in order to select current optimal action aggregation. Each detailed part of the novel DRL agent network framework structure can be illustrated as follows.

B. Offloading Policy Generation

(1) Offloading decision: For the proposed MINPP, our purpose is to generate the offloading action by the DRL agent interacting with the environment (i.e., the application tasks from IIoTD). Specifically, given DRL agent's initial policy function π, we input the application task L to DRL agent, and it can be defined as

$$\pi : \pi_w(A|L) \ . \tag{19}$$

The detailed internal network structure of the DRL agent is presented in Fig. 3. In this agent network structure, the network structure between different hidden layers employs full connection and we can see that the offloading action depends on the policy function $\pi_w(A|L)$ which is implanted DNN parameter w, i.e., the weights that connect neural neurons between different network layers. In addition, the output layers generate the offloading decision A based on the current policy function $\pi_w(A|L)$. Next, we describe an isotone optimization method in terms of the generated offloading action.

(2) 2AGT optimization: Supposing that we obtain the application task L_s in s^{th} step. The offloading decision A_s can denoted as

$$A_s = \{0 \le a^s_{i,j} \le 1 | i \in [1,2,...N], j \in [1,2,...Z]\} \ , \tag{20}$$

Then we start to introduce an isotone action method. Inspired by the quantitative technology from signal coding [46], we transform the offloading decision A_s to massive action sets for the sake of obtaining optimal action A^*_s. The number of action aggregations is $1 \le M \le NZ+1$. A_{s1} can be derived just as follows:

$$A_{s1} = \{a^{s1}_{ij}\} = \begin{cases} 1, & \text{if } a^s_{ij} \ge K, \\ 0, & \text{if } a^s_{ij} < K, \end{cases} \tag{21}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

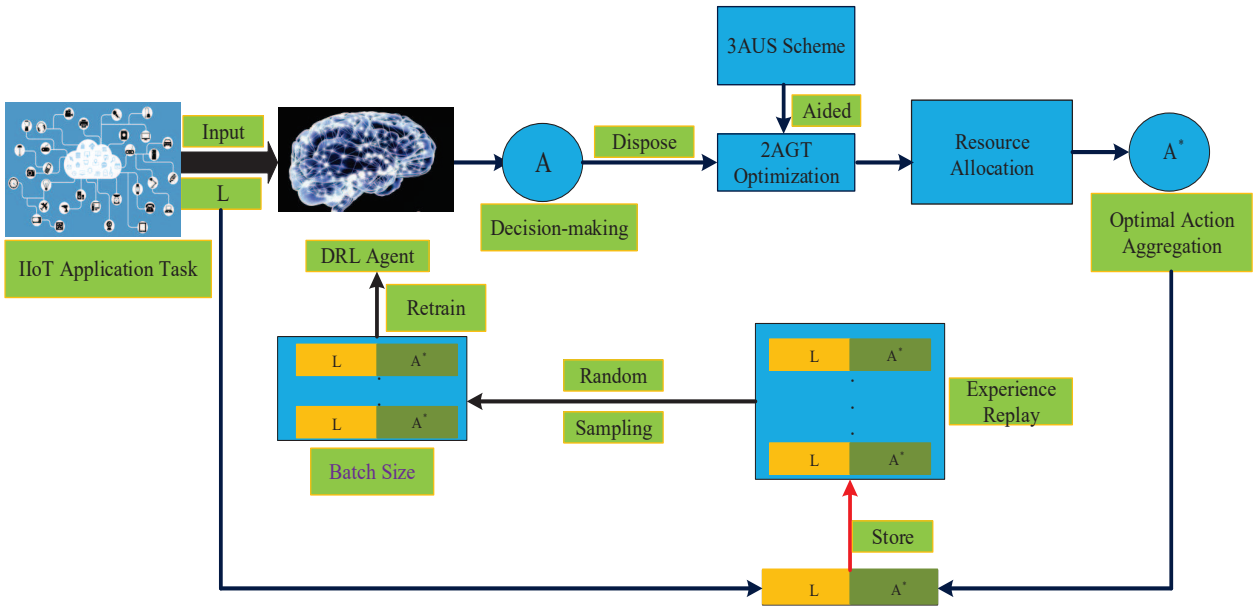

Fig. 2: The proposed DRL network structure.

where K is equal to a discriminant value for offloading decision, and we set it as 0.5 in order to quantify the offloading decision equally. Subsequently, the agent generates other $M-1$ offloading aggregations respect to the distance parameter K and reshuffle the originally generated actions, which can be defined by $|a^s_{11}-K| \leq |a^s_{12}-K| ... \leq |a^s_{NZ}-K|$. Then, the remaining $M-1$ offloading aggregations are recalculated according to 2AGT, which can be denoted as two cases.

Case 1: When $a^s_{ij} > a^s_{m-1m-1}$ or $a^s_{ij} = a^s_{m-1m-1}$, $a_{m-1m-1} < K$, the quantitative action value a^{sm}_{ij} can be represented as

$$a^{sm}_{ij} = 1. \tag{22}$$

Case 2: When $a^s_{ij} < a^s_{m-1m-1}$ or $a^s_{ij} = a^s_{m-1m-1}$, $a_{m-1m-1} \geq K$, the quantitative action value a^{sm}_{ij} can be represented as

$$a^{sm}_{ij} = 0. \tag{23}$$

In terms of above two cases, the total quantitative action aggregations can be calculated as

$$A_{sm} = \{a^{sm}_{ij}\}, \tag{24}$$

where $m = 2, 3, 4...M$, and we can see that there are NZ offloading actions for all application tasks. In addition, we can generate at most $NZ+1$ action aggregations. Next, by solving the convex optimization problem, the optimal offloading action A^*_{sm} can be denoted as:

$$A^*_{\mathrm{sm}} = \arg\min_{A_{sm}} V^*\left(L_s, A_{sm}, C\right), \tag{25}$$

In the next section, we discuss how to adjust the action aggregations parameter.

(3) 3AUS parameter setting: Intuitively, by setting more action aggregations M, a lower total cost function can be calculated followed with higher computational complexity. Instead, setting a proper M may reduce the potential computational complexity without losing the system performance. According to the rolling horizon control (RHC) theory [47], we can update the action aggregations parameter per δ steps. Specifically, when the step s is the integer times of the δ, the DRL agent can choose to renew the aggregations parameter. When s=1, RHC parameters are

$$M_s = NZ + 1. \tag{26}$$

When s mod δ=0, the update parameter is

$$M_s = \min(\max(m^*_{s-\delta+1}, m^*_{s-\delta+2}, ...m^*_{s-1}) + P, NZ+1), \tag{27}$$

where $m^*_{s-\delta+1}$ represents the index of optimal action aggregations. P is a constant in order to allow the number of aggregations to increase during the update period and if it doesn't reach the update steps δ for other steps, it can be the same as the previous value.

(4) Convex optimization function: According to the 2AGT and 3AUS schemes, we can transfer the initial problem into convex objective [48], as illustrated in Fig. 2. After obtaining the value of offloading action aggregations, the original problem is

$$\min_{\{L,C\}} V(L, C), \tag{28}$$

$$s.t. \quad \sum_{n=1}^{N}\sum_{z=1}^{Z} c_{n,z} \leq C_{total}, \tag{29}$$

$$c_{n,z} > 0. \tag{30}$$

Evidently, it is a convex optimization problem and we can solve the Karush-Kuhn-Tucker (KKT) conditions to obtain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

current optimal cost function.

Proof: To demonstrate the optimization problem (30) as the convex problem, we need to prove the $V(L,\ C)$, constraint (31), and (32) as convex function, respectively. Firstly, as L is the time-varying state vector, so the function $V(L,\ C)$ is only related to the optimization variable C. Next, combined with (31) and (32), as $\sum_{n=1}^{N}\sum_{z=1}^{Z} c_{n,z} - C_{total}$ and $-c_{n,z}$ are affine functions, it must be the convex function. Additionally, after obtaining the offloading decision-making, the optimization objective $V(L,\ C)$ can be simply reformulated as

$$V(L,\ C) = \lambda\left(\sum_{n=1}^{N}\sum_{z=1}^{Z}\frac{a_{n,z}l_{n,z}}{c_{n,z}}\right) + \sum_{n=1}^{N}\sum_{z=1}^{Z}\frac{a_{n,z}l_{n,z}P_{tran}^{n}}{c_{n,z}}. \tag{31}$$

Lemma 1: For two convex functions $f_1(x)$ and $f_2(x)$, the summation of $f_1(x)$ and $f_2(x)$ is still convex function.

Proof: For any convex function $f(x)$,

$$f\left(\lambda x_1 + (1-\lambda)x_2\right) \le \lambda f\left(x_1\right) + (1-\lambda) f\left(x_2\right). \tag{32}$$

Let $g\left(x\right) = f_1(x) + f_2(x)$, where $f_1(x)$ and $f_2(x)$ are two convex functions. Hence,

$$g\left(\lambda x_1 + (1-\lambda)x_2\right) = f_1(\lambda x_1 + (1-\lambda)x_2) \tag{33}$$
$$+ f_2(\lambda x_1 + (1-\lambda)x_2) \tag{34}$$
$$\le \lambda f_1(x_1) + (1-\lambda)f_1(x_2) + \lambda f_2(x_1) + (1-\lambda)f_2(x_2) \tag{35}$$
$$= \lambda(f_1(x_1) + f_2(x_1)) + (1-\lambda)(f_1(x_2) + f_2(x_2)) \tag{36}$$
$$= \lambda g(x_1) + (1-\lambda)g(x_2), \tag{37}$$

which demonstrates that the summation of $f_1(x)$ and $f_2(x)$ is still convex function. As $a_{n,z}$, $l_{n,z}$ and P_{tran}^{n} are known for $V(L,\ C)$, $V(L,\ C)$ can be regarded as the summation of multiple convex functions for optimization variable $c_{n,z}$. Hence, $V(L,\ C)$ can be proved as a convex function in terms of Lemma 1. Finally, the formulated optimization problem is convex. Then, we choose an action aggregation A_{sm}^{*} from the optimal total cost function to start to update network parameters. In the next section, we show how to update the network parameters.

C. Network Parameters Update

After the agent obtains the optimal action aggregation A_{sm}^{*}, the agent can update the network parameters (i.e., the offloading policy $\pi_w(A|L)$). In detail, since the experience of the DRL agent is interrelated, randomly selecting a batch of training samples from replay memory can decrease the interrelation among agent experience and this may help the DRL agent utilize comprehensive experience in order to learn better. So we adopt the experience replay technology [49] to update the network parameters by using the stored data pairs (L_s, A_{sm}^{*}). Firstly, we keep an empty memory structure. Then the structure supplies new data pairs, and once the memory structure is full, the newly generated data pairs can displace the old. The DRL agent randomly selects several generated data pairs (L_s, A_{sm}^{*}) in s^{th} step from memory structure to reduce the over-fitting, which can be characterized by total steps S_t. We define the cross-entropy just as follows:

$$O(w_s) = -\frac{1}{|S_t|}\sum_{s}\left[(A_{sm}^{*})^{T}\log(\pi_{ws}(A_s|L_s)\right. \\ \left. +(1-A_{sm}^{*})^{T}\log(1-(\pi_{ws}(A_s|L_s)))\right], \tag{38}$$

where $|S_t|$ represents the total number of sampling steps, and the superscript T means the transpose operator. In our simulations, we update our network parameters each ε while collecting enough new data pairs. Meanwhile, the DRL agent only updates from the most recent data pairs, which are produced by a new offloading strategy. The detailed algorithm procedure is described in Algorithm 1, where the computational complexity of the proposed algorithm can be derived as $O\left(SL + SMNZ + \frac{S}{K}\right)$.

Algorithm 1 The proposed DRL algorithm

Input:
Each task L_s;
Initially the neural network parameter w;
Output:
The agent outputs A_{sm}^{*};
The cost function $V(L, A, C)$;

for $\{1, 2, ...S\}$ **do**
  Obtain the offloading decision A_s via 2AGT and DRL network structure;
  Choose appropriate M_s in terms of 3AUS;
  if s mod δ =0 **then**
    Start to choose the new the number of action aggregations from 3AUS;
  end if
  Extend action sets A_s into $\{A_{s1}, A_{s2}, ...A_{sm}\}$;
  for $\{1, 2, ...|M_s|\}$ **do**
    Calculate $V(L_s, A_s, C)$ for all $\{A_{s1}, A_{s2}, ...A_{sm}\}$;
    Select $A_{sm}^{*} = \underset{\{A_{s1}, A_{s2}, ... A_{sm}\}}{\arg\min} V(L_s, A_s, C)$;
  end for
  Add the action pairs $\{L_s, A_{sm}^{*}\}$ into buffer pool;
  if s mod ε =0 **then**
    Stochastically selecting K tuples $\{L_s, A_{sz}^{*}\}$ and update the DRL agent.
  end if
end for

V. Simulation Results and Performance Analysis

A. Experimental Settings

In this section, the number of IIoTD can be denoted as $N = 10$ and there are $Z = 5$ tasks to be performed. Simultaneously, the channel bandwidth and transmission power are 100Mbps and 0.2W, respectively. The task size $l_{n,z}$ obeys the uniform distribution between $(5MB, 35MB)$ and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

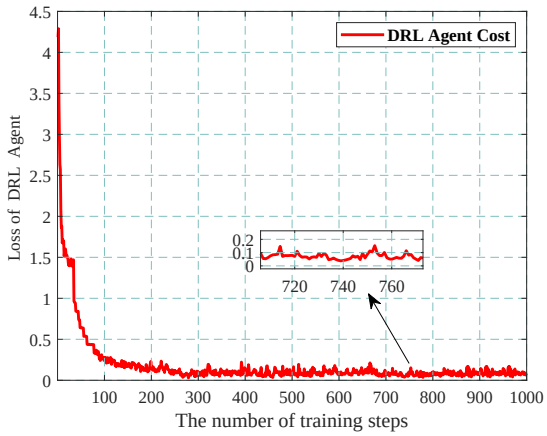

Fig. 4: The loss function in terms of learning rate=0.01.

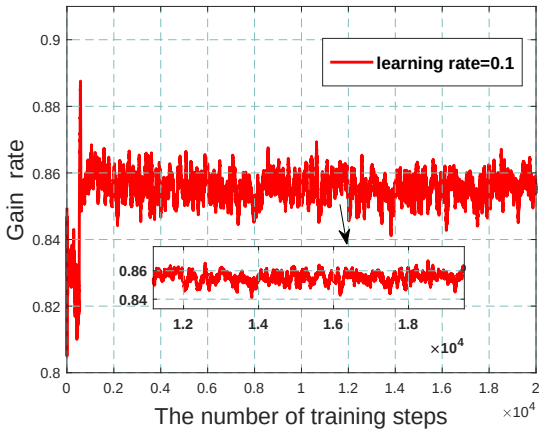

Fig. 5: The DRL gain rate with learning rate=0.1.

the CPU-cycle frequency follows the uniform distribution between $(0.6e8cycle/s, 2.5e8cycle/s)$ [50]. Further, $e_{n,z}$ is uniformly distributed between $(1000cycle/bit, 3000cycle/bit)$ and F_{server}^{total} is $6.5e9$ cycle/s. We set $\sigma^2 = 1 * 10^{-9}$. We utilize the pycharm community edition as the programming environment for constructing DNN with tensorflow, and the number of hidden layers is 3 where DNN uses full connection.

B. Convergent Performance Analysis

In this simulation, we input the application tasks L_s into the DRL agent in each step s, where the sample complexity includes 30000 training samples and 10000 test samples. After proper time intervals, the DRL agent is retrained again in order to improve its convergent performance. Finally, we obtain a quasi-optimal gain rate and total system cost subject to enumerating actions.

(1) The DRL cost function: As shown in Fig. 4, when the number of training steps increases, the error function of predicted value and optimal value gradually decreases to the minimum value. In fact, when in approximately 250 steps, the cost of the DRL agent is close to 0, which validates that our proposed algorithms have fast convergent speed. At the same time, while receiving different application tasks, the DRL agent has a stronger generalization ability and reduces over-fitting, which demonstrates the effectiveness of buffer pool and stochastically selecting.

(2) Gain Rate: As illustrated in Fig. 5, we can see that the agent will not converge to optimal solutions despite enough training steps. In other words, once more than 250 steps, the agent can not obtain the optimal offloading action and the gain rate is less than 0.9. It means that the system utility is lower. Further, we have to choose proper network parameters and achieve tradeoff between performance and computational complexity. Next, we show the system gain rate with different network parameters settings.

As shown in Fig. 6, we set some different learning rates to illustrate the relationship between gain rate and training steps. For better comparison, the contrast scheme is the optimal solution in terms of extensive search method. Generally, if

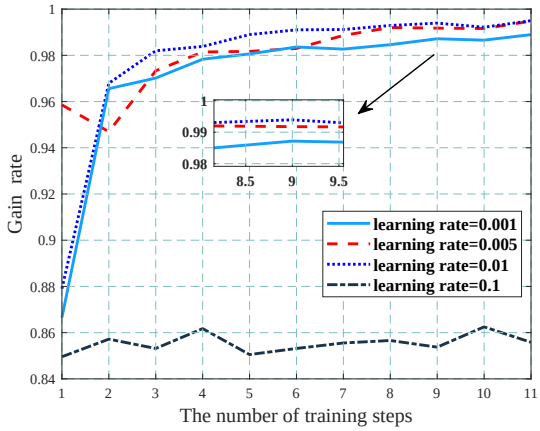

Fig. 6: The interrelation between gain rate and learning rates.

the learning rate is higher, the convergent speed of the DRL agent can be faster. However, the figure shows that when the learning rate is higher, the gain rate can not be optimal. If we set a higher learning rate, the DRL agent may obtain a local optimal policy rather than the global optimal. Accordingly, we must select a proper learning rate in terms of a specific network environment.

In Fig. 7, we study the effect of different batch sizes on gain rate. In addition, we can see that a small batch size (e.g., size=32) can not utilize all data pairs in the memory size, which leads to slow convergent speed. However, if the selected batch size is large enough (e.g., size=256), the agent can frequently use the old data pairs and may reduce the system performance. Hence, we must choose the proper batch size according to the environment states.

Fig. 8 shows the interrelation between gain rate and different replay experience sizes. At the same time, we set the batch size as 128. We can see that the gain rate is close to 1 when the number of replay experience size is 512 and 1024. Further, as the number of replay experience size is 1024, its convergent speed is faster than others. However, the DRL agent can not converge to optimal solutions once the replay experience size

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

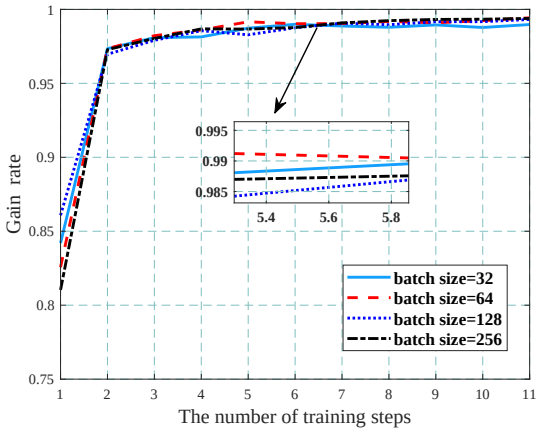

Fig. 7: The interrelation between gain rate and batch sizes.

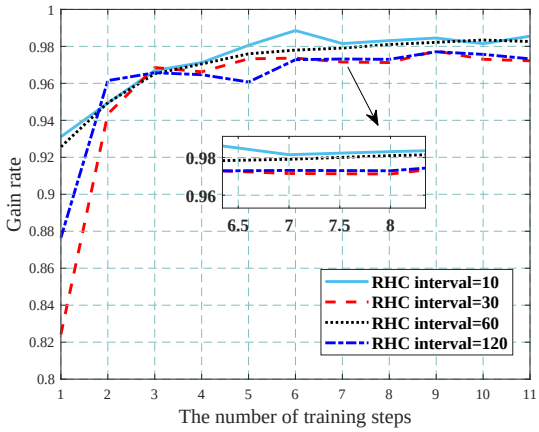

Fig. 9: The interrelation between gain rate and RHC intervals.

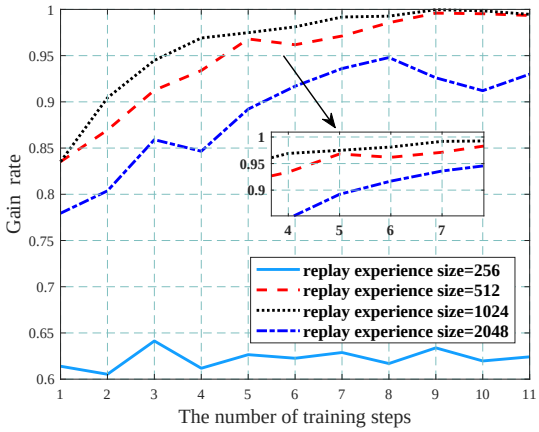

Fig. 8: The interrelation between gain rate and replay experience sizes.

is 256, since the selected training data pairs are interrelated and lead to a local optimal solution. In addition, when the replay experience size is 2048, the convergent speed of the DRL agent is slow as it can not fully utilize the collected data pairs to reduce the error loss. Hence, we are supposed to choose the proper replay experience size in terms of different application tasks.

C. 3AUS Parameter Interval

In Fig. 9, it shows the interrelation between gain rate and RHC intervals. If the update interval is proper, the aggregation parameter M_s can be renewed frequently, which means the DRL agent decreases its computational complexity with a small aggregation parameter. As the number of RHC interval increases, the gain rate is gradually descendent since the big RHC interval causes the higher computational complexity for the total cost function, which means that the DRL agent must renew the number of action sets with a relatively small RHC interval instead of the big. Hence, we are supposed to choose a proper RHC interval while maintaining system performance.

D. System Performance

In this section, we compare our proposed DRL-based 2AGT method and 3AUS strategy with some benchmarks under a variety of system settings. Simulation results demonstrate the effectiveness of our proposed DRL-based algorithm.

(1) CPU clock speed: As illustrated in Fig. 10, it shows the total system cost of the IIoT model considering the MEC server's CPU clock speed. From Fig. 10, our proposed method is a quasi-optimal solution compared with extensive search algorithm. Also, there is a small gap between our proposed method and extensive search. At the same time, the full local represents that all application tasks from IIoTD are executed in local devices, while the full offloading indicates that all tasks are offloaded to the MEC server to reduce energy consumption and delay. We can see that our proposed method outperforms the full offloading and local execution. Further, the full local execution is not able to change with the CPU speed clock. This is because the local execution can not depend on the MEC server resources [51], whereas the full offloading mode decreases the total system cost with the increase of the CPU clock speed. More importantly, we compare our proposed method with two conventional DRL-based algorithms, i.e., conventional deep Q network (C-DQN) and deep deterministic policy gradient (DDPG), which show that our proposed method can further reduce the total system cost in contrast with C-DQN and DDPG. Hence, we can utilize the MEC server's powerful computational resources to help handle the application tasks in terms of our proposed method.

(2) Delay weight factor: In Fig. 11, we plot the total system cost in terms of delay weight factor λ. The delay weight factor reveals the weight on delay relative to total energy consumption in terms of different application tasks from IIoTD. In general, delay weight is more significant than energy consumption for some current IIoT application tasks [52]. Specifically, with the increase of delay weight factor, we compare our proposed method with other offloading strategies including extensive search, full local, full offloading, intelligent C-DQN and DDPG for the total system cost.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

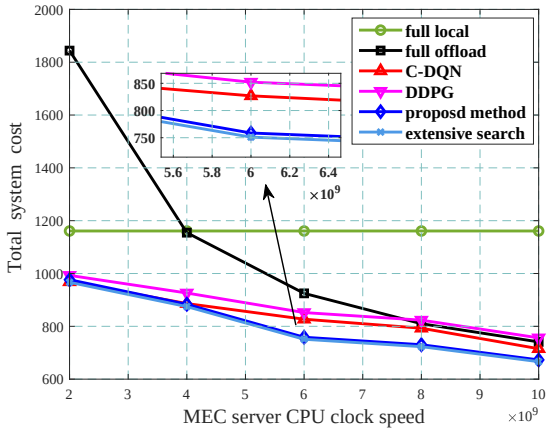

Fig. 10: The total system cost versus MEC server speed.

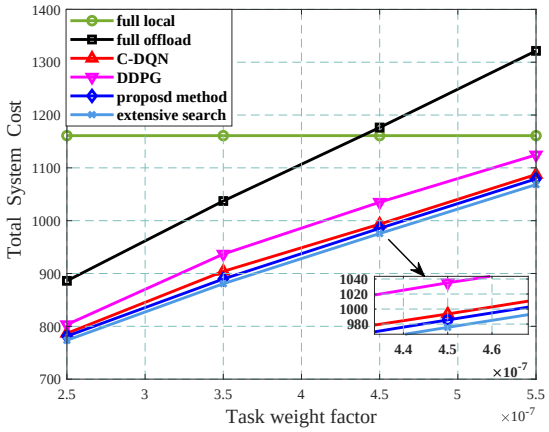

Fig. 12: The total system cost versus task weight factor of β.

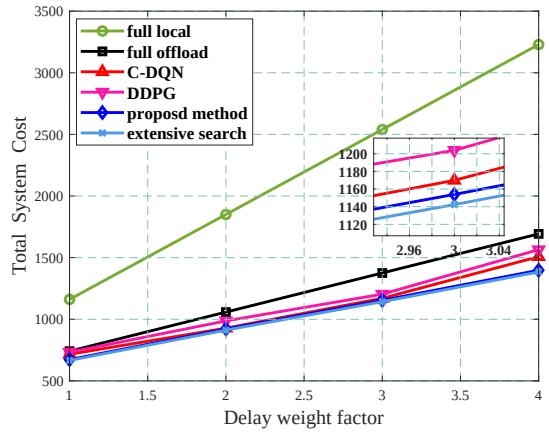

Fig. 11: The total system cost versus delay weight factor λ.

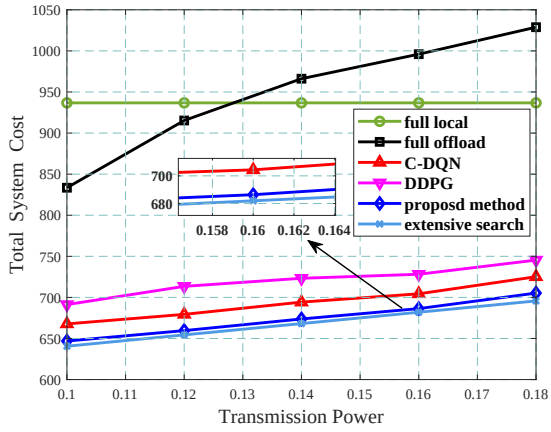

Fig. 13: The total system cost versus transmission power.

Under different delay weight factors, we can observe that our proposed method outperforms the full offloading and full local execution schemes in terms of the system cost. Further, our proposed method has lower total system cost than C-DQN and DDPG, which validates the progressiveness and intelligence of our proposed method. Finally, it is also close to extensive search, which means our proposed method can achieve quasi-optimal total system cost.

(3) Task weight factor: In Fig. 12, we show the interrelationship between the total system cost and task weight factor β with different strategies. In detail, the task weight factor suggests the task size is related to the computational energy consumption from the MEC sever. Different task weight factors mean the various application tasks from IIoTD. In general, the larger the task size is, the larger the task weight factor is. From Fig. 12, we can see that our proposed method is superior to the full offloading and full local schemes in terms of the total system cost. Further, our proposed method outperforms C-DQN and DDPG when task weight factor increases, which demonstrates the effectiveness and intelligence of our proposed method compared with C-DQN and DDPG. In addition, when the task weight factor increases, the total system cost is higher for all schemes. Finally, our proposed DRL-based algorithm is close to extensive search for total system cost, which represents that our proposed strategy can achieve a better sub-optimal solution and have superior intelligence.

(4) Transmission power: As shown in Fig. 13, we explore the relationship between different transmission power and total system cost. As DDPG only explores the optimal offloading-decision and transmission rate, it has higher computational cost. Additionally, C-DQN can better adapt to the network environments compared with DDPG. However, as we propose 2AGT optimization and 3AUS scheme to help dispose the computation offloading and resource allocation, the total system cost is lower than C-DQN. Additionally, our proposed DRL-based network structure is quasi-optimal compared with extensive search, which further demonstrates the effectiveness and reliability of our proposed method.

VI. Conclusions

In this paper, considering a large number of IIoT application tasks, we establish a novel IIoT model with edge intelligence service in 6G. In addition, we propose a novel DRL-based network structure followed with the 2AGT scheme and 3AUS strategy to jointly optimize the offloading decision and transmission resource allocation problems in the IIoT

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Citation information: DOI 10.1109/TNSE.2022.3141728,

system. Moreover, a 2AGT method is proposed to generate the offloading decision and update the number of action aggregation adaptively based on the 3AUS scheme. At the same time, we adopt the experience replay technique and randomly select a batch of samples in order to improve its convergent performance and reduce over-fitting. Finally, the simulation results validate that our proposed method can achieve better system performance compared with other benchmarks.

REFERENCES

[1] X. You, C.-X. Wang, J. Huang, X. Gao, Z. Zhang, M. Wang, Y. Huang, C. Zhang, Y. Jiang, J. Wang *et al.*, "Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts," *Science China Information Sciences*, vol. 64, no. 1, pp. 1–74, Nov. 2021.

[2] A. Mukherjee, P. Goswami, M. A. Khan, L. Manman, L. Yang, and P. Pillai, "Energy-efficient resource allocation strategy in massive IoT for industrial 6G applications," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5194–5201, Nov. 2020.

[3] Y. Gong, J. Wang, and T. Nie, "Deep reinforcement learning aided computation offloading and resource allocation for iot," in *2020 IEEE Computing, Communications and IoT Applications (ComComAp)*, Beijing, China, 2020, pp. 01–06.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, Aug. 2017.

[5] H. Yao, L. Wang, X. Wang, Z. Lu, and Y. Liu, "The space-terrestrial integrated network: An overview," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 178–185, Apr. 2018.

[6] H. Yao, T. Mai, J. Wang, Z. Ji, C. Jiang, and Y. Qian, "Resource trading in Blockchain-based industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3602–3609, Mar. 2019.

[7] T. Mai, H. Yao, N. Zhang, L. Xu, M. Guizani, and S. Guo, "Cloud mining pool aided Blockchain-enabled Internet of Things: An evolutionary game approach," *IEEE Transactions on Cloud Computing, (DOI: 10.1109/TCC.2021.3110965)*, 2021.

[8] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4242–4251, Oct. 2018.

[9] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb. 2013.

[10] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, Jul. 2019.

[11] P. Si, Y. He, H. Yao, R. Yang, and Y. Zhang, "DAVE: Offloading delay-tolerant data traffic to connected vehicle networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3941–3953, Apr. 2016.

[12] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, Jun. 2019.

[13] T. K. Rodrigues, K. Suto, H. Nishiyama, J. Liu, and N. Kato, "Machine learning meets computation and communication control in evolving edge and cloud: Challenges and future perspective," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 38–67, Sep. 2019.

[14] P. Yang, F. Lyu, W. Wu, N. Zhang, L. Yu, and X. Shen, "Edge coordinated query configuration for low-latency and accurate video analytics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4855–4864, Jul. 2020.

[15] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[16] B. Liu, W. Zhang, W. Chen, H. Huang, and S. Guo, "Online computation offloading and traffic routing for UAV swarms in edge-cloud computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 8777–8791, Aug. 2020.

[17] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*, San Francisco, CA, pp. 49–62.

[18] C. Wang, Y. Li, and D. Jin, "Mobility-assisted opportunistic computation offloading," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1779–1782, Oct. 2014.

[19] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[20] S. Guo, Y. Dai, S. Guo, X. Qiu, and F. Qi, "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile Blockchain," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5549–5561, May 2020.

[21] C. Liang, F. R. Yu, H. Yao, and Z. Han, "Virtual resource allocation in information-centric wireless networks with virtualization," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9902–9914, Dec. 2016.

[22] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.

[23] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

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

[25] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[26] J. Zhang, W. Xia, Y. Zhang, Q. Zou, B. Huang, F. Yan, and L. Shen, "Joint offloading and resource allocation optimization for mobile edge computing," in *IEEE Global Communications Conference (GLOBECOM)*, Singapore, 2017, pp. 1–6.

[27] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *IEEE International Conference on Communications. (ICC)*, Kuala Lumpur, Malaysia, 2016, pp. 1–6.

[28] Z. Hong, H. Huang, S. Guo, W. Chen, and Z. Zheng, "QoS-aware cooperative computation offloading for robot swarms in cloud robotics," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 4027–4041, Apr. 2019.

[29] Y. Zhang, D. Niyato, and P. Wang, "Offloading in mobile cloudlet systems with intermittent connectivity," *IEEE Transactions on Mobile Computing*, vol. 14, no. 12, pp. 2516–2529, Dec. 2015.

[30] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, 2016, pp. 1451–1455.

[31] Y.-F. Huang, T.-H. Tan, N.-C. Wang, Y.-L. Chen, and Y.-L. Li, "Resource allocation for D2D communications with a novel distributed Q-learning algorithm in heterogeneous networks," in *International Conference on Machine Learning and Cybernetics (ICMLC)*, vol. 2, Chengdu, China, 2018, pp. 533–537.

[32] H. Yao, T. Mai, C. Jiang, L. Kuang, and S. Guo, "AI routers & network mind: A hybrid machine learning paradigm for packet routing," *IEEE Computational Intelligence Magazine*, vol. 14, no. 4, pp. 21–30, Nov. 2019.

[33] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, May 2017.

[34] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, Jan. 2020.

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[36] X. He, H. Lu, H. Huang, Y. Mao, K. Wang, and S. Guo, "QoE-based cooperative task offloading with deep reinforcement learning in mobile edge networks," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 111–117, Jun. 2020.

[37] C. Qiu, H. Yao, F. R. Yu, F. Xu, and C. Zhao, "Deep Q-learning aided networking, caching, and computing resources allocation in software-defined satellite-terrestrial networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 5871–5883, Jun. 2019.

[38] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Transactions on Big Data, (DOI: 10.1109/TBDA-TA.2019.2940675)*, 2019.
[39] S. R. Bickham, M. A. Marro, J. A. Derick, W.-L. Kuang, X. Feng, and Y. Hua, "Reduced cladding diameter fibers for high-density optical interconnects," *Journal of Lightwave Technology*, vol. 38, no. 2, pp. 297–302, Jan. 2019.
[40] H. Widiarti, S.-Y. Pyun, and D.-H. Cho, "Interference mitigation based on femtocells grouping in low duty operation," in *IEEE Vehicular Technology Conference (VTC)*, Ottawa, Canada, Sep, 2010, pp. 1–5.
[41] J. Phiri and T. J. Zhao, "Using Shannon's information theory and artificial neural networks to implement multimode authentication," in *IEEE International Conference on Communications and Intelligence Information Security (ICCIIS)*, Nanning, China, Oct, 2010, pp. 271–274.
[42] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, Mar. 2016.
[43] L. Huang, X. Feng, A. Feng, Y. Huang, and L. P. Qian, "Distributed deep learning-based offloading for mobile edge computing networks," *Springer Mobile networks and applications, (DOI: 10.1007/s11036-018-1177-x)*, Nov. 2018.
[44] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, Jun. 2020.
[45] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial internet of things: A dueling deep Q-learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4627–4639, Jun. 2018.
[46] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, Nov. 2019.
[47] K. J. Åström, *Introduction to stochastic control theory*. Courier Corporation, 2012.
[48] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.
[49] B. Luo, Y. Yang, and D. Liu, "Adaptive Q-learning for data-based optimal output regulation with experience replay," *IEEE transactions on cybernetics*, vol. 48, no. 12, pp. 3337–3348, Dec. 2018.
[50] Z. Zhao, R. Zhao, J. Xia, X. Lei, D. Li, C. Yuen, and L. Fan, "A novel framework of three-hierarchical offloading optimization for MEC in industrial IoT networks," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5424–5434, Aug. 2019.
[51] J. Wang, L. Zhao, J. Liu, and N. Kato, "Smart resource allocation for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on emerging topics in computing*, vol. 9, no. 3, pp. 1529–1541, Jul. 2019.
[52] J. Wan, S. Tang, Z. Shu, D. Li, S. Wang, M. Imran, and A. V. Vasilakos, "Software-defined Industrial Internet of Things in the context of industry 4.0," *IEEE Sensors Journal*, vol. 16, no. 20, pp. 7373–7380, Oct. 2016.

Yongkang Gong is pursuing his doctor degree in the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing. His research interests include network artificial intelligence, multi-access edge computing, space-air-ground integrated network, and multi-agent deep reinforcement learning.

Haipeng Yao (M'16, SM'20) is a Professor in Beijing University of Posts and Telecommunications. Haipeng Yao received his Ph.D. in the Department of Telecommunication Engineering at University of Beijing University of Posts and Telecommunications in 2011. His research interests include future network architecture, network artificial intelligence, networking, space-terrestrial integrated network, network resource allocation and dedicated networks. He has published more than 100 papers in prestigious peer-reviewed journals and conferences. Dr. Yao has served as an Editor of IEEE Network, IEEE Access, and a Guest Editor of IEEE Open Journal of the Computer Society and Springer Journal of Network and Systems Management. He has also served as a member of the technical program committee as well as the Symposium Chair for a number of international conferences, including IWCMC 2019 Symposium Chair, ACM TUR-C SIGSAC2020 Publication Chair.

Jingjing Wang (S'14-M'19-SM'21) received his B.S. degree in Electronic Information Engineering from Dalian University of Technology, Liaoning, China in 2014 and the Ph.D. degree in Information and Communication Engineering from Tsinghua University, Beijing, China in 2019, both with the highest honors. From 2017 to 2018, he visited the Next Generation Wireless Group chaired by Prof. Lajos Hanzo, University of Southampton, UK. Dr. Wang is currently an associate professor at School of Cyber Science and Technology, Beihang University. His research interests include AI enhanced next-generation wireless networks, swarm intelligence and confrontation. He has published over 100 IEEE Journal/Conference papers. Dr. Wang was a recipient of the Best Journal Paper Award of IEEE ComSoc Technical Committee on Green Communications & Computing in 2018, the Best Paper Award of IEEE ICC and IWCMC in 2019.

Maozhen Li is currently a Professor with the Department of Electronic and Computer Engineering, Brunel University London, London, U.K. His current research interests include high-performance computing, big data analytics, and knowledge and data engineering. Mr. Li is a Fellow of the BCS and IET.

Song Guo (Fellow, IEEE) is a Full Professor and Associate Head (Research Development) in the Department of Computing at The Hong Kong Polytechnic University. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. Prof. Guo is an IEEE Fellow (Computer Society), a Highly Cited Researcher (Clarivate Web of Science), and an ACM Distinguished Member. His research interests are mainly in the areas of big data, edge AI, mobile computing, and distributed systems. He co-authored 4 books, co-edited 7 books, and published over 500 papers in major journals and conferences. He is the recipient of the 2019 IEEE TCBD Best Conference Paper Award, 2018 IEEE TCGCC Best Magazine Paper Award, 2019 2017 IEEE Systems Journal Annual Best Paper Award, and other 8 Best Paper Awards from IEEE/ACM conferences. His work was also recognized by the 2016 Annual Best of Computing: Notable Books and Articles in Computing in ACM Computing Reviews. Prof. Guoąŕs research has been sponsored by RGC, NSFC, MOST, JSPS, industry, etc. Prof. Guo is the Editor-in-Chief of IEEE Open Journal of the Computer Society and the Chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee. He was an IEEE ComSoc Distinguished Lecturer and a member of IEEE ComSoc Board of Governors. He has also served for IEEE Computer Society on Fellow Evaluation Committee, Transactions Operations Committee, Editor-in-Chief Search Committee, etc. Prof. Guo has been named on editorial board of a number of prestigious international journals like IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Cloud Computing, IEEE Transactions on Emerging Topics in Computing, etc. He has also served as chairs of organizing and technical committees of many international conferences.