Available online at www.sciencedirect.com

ScienceDirect

Future Generation Computer Systems 00 (2022) 1–10

FGCS

AoI-Aware Energy Control and Computation Offloading for Industrial IoT

Jiwei Huang[a], Han Gao[a], Shaohua Wan[b], Ying Chen[c,*]

[a]Beijing Key Laboratory of Petroleum Data Mining, China University of Petroleum, Beijing 102249, China
[b]Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen 518110, China
[c]School of Computer Science, Beijing Information Science and Technology University, Beijing 100101, China

Abstract

In Industrial Internet of Things (IIoT), a large volume of data is collected periodically by IoT devices, and timely data routing and processing are important requirements. Age of Information (AoI), which is a metric to evaluate the freshness of status information in data processing, has become one of the most important objectives in IIoT. In this paper, considering limited communication, computation and energy resources on IoT devices, we jointly study the optimal AoI-aware energy control and computation offloading problem within a dynamic IIoT scenario with multiple IoT devices and multiple edge servers. Based on extensive analysis of real-life IoT dataset, Markovian queueing models are constructed to capture the dynamics of IoT devices and edge servers, and their corresponding analyses are provided. With the quantitative analytical results, we formulate a dynamic Markov decision problem with the objective of minimizing the long-term energy consumption while satisfying AoI constraints for real-time data processing. To solve the problem, we apply Deep Reinforcement Learning (DRL) techniques for adapting to large-scale dynamic IIoT environments, and design an intelligent Energy Control and Computation Offloading (ECCO) algorithm. Extensive simulation experiments are conducted based on real-world dataset, and the comparison results illustrate the superiority of our ECCO algorithm over both existing DRL and non-DRL algorithms.

Keywords: Age of Information (AoI), computation offloading, Industrial Internet of Things (IIoT), Deep Reinforcement Learning (DRL)

1. Introduction

Recently, Industrial Internet of Things (IIoT) has emerged as a novel technology that brings wireless communication and intelligent control to manufacturing [1]. With more and more industrial applications with the capability of data sensing, real-time processing and intelligent decision making being deployed, the requirement on real-time Quality of Service (QoS) has been growing rapidly, which brings significant challenges to the design and management of the underlying infrastructure.

In order to process the sensing data in a real-time manner, a novel architecture namely edge computing is introduced in most IIoT systems. It deploys edge servers with computing capacities at the edge of the network (base stations). Delay-sensitive requests can be determined to be offloaded to a nearby edge server via wireless channel [2], and thus the traffic at the core network and the delay for transmitting user data can be reduced especially for data-intensive industrial applications [3]. In edge computing, networking and computing are combined, and whether to offload the tasks to an edge server should be carefully decided according to QoS requirements, networking status, and energy supplies [4].

In IIoT, a wide range of smart devices continuously generate a large amount of status or event data. The IIoT system has to process the data, analyze the status, and then react accordingly. IIoT applications require low latency to ensure timely statistics and analysis of real-time data. Besides traditional QoS, a new metric namely Age of Information (AoI) has been proposed recently, which is to evaluate the freshness of the status information of physical processes [5, 6]. AoI introduces data semantics into traditional performance evaluation, and thus has become increasingly popular in the design and optimization of IoT systems. Quantitatively, it can be defined as the time elapsed from the status information being collected from the source node (IoT device) to the data being processed [7]. However, minimizing AoI is a challenging task in IIoT. Since IoT devices commonly have very limited computing and

*Corresponding author
Email addresses: huangjw@cup.edu.cn (Jiwei Huang), 2021211248@student.cup.edu.cn (Han Gao), shaohua.wan@uestc.edu.cn (Shaohua Wan), chenying@bistu.edu.cn (Ying Chen)

energy resources [8, 9, 10], while the computation capability of edge servers is also commonly bounded, how to dynamically allocate the limited resources and schedule the tasks according to the environments for obtaining a long-term minimization of AoI still remains largely unexplored.

In this paper, we make an attempt at filling this gap by introducing Deep Reinforcement Learning (DRL) to energy control and computation offloading for AoI optimization in IIoT. We carry out extensive analyses on dataset collected from real-world mobile network operators, and propose queueing models for both IoT devices and edge servers. With queueing theory, delays, AoI as well as energy consumption are quantitatively evaluated. Then, we formulate a dynamic optimization problem of AoI-aware energy control and computation offloading by Markov Decision Process (MDP). To solve the problem in a dynamic IIoT network environment, we apply DRL technique and design an intelligent Energy Control and Computation Offloading (ECCO) algorithm. Simulations are conducted based on real-life data, and the experimental results illustrate the superiority of our ECCO algorithm over both existing DRL and non-DRL algorithms.

The remainder of this paper is organized as follows. In Section 2, we survey the related work most pertinent to this paper. In Section 3, we propose the system model for capturing the dynamics of an IIoT system and conduct quantitative analyses. In Section 4, we formulate the dynamic optimization problem and describe our ECCO algorithm. In Section 5, we evaluate the performance of our approach experimentally. Finally, we conclude this paper in Section 6.

2. Related Work

AoI has been recently proposed to describe the freshness of status information, which is considered as a key metric in performance evaluation of information systems. Formally, AoI is defined as the time elapsed from the source node generating the status information to the target node successfully receiving the data. Kaul *et al.* [5] studied the AoI theoretically by proposing a single-server Markovian queueing model. In their mathematical analysis, the data packets were assume to be a Poisson random process, and data transmission was formulated as the service processes. With the analytical results, the frequency of status updating was discussed. Champati *et al.* [11] attempted to obtain the minimum AoI in a single-source and single-service queueing model when service time can be a general distribution. Recently, there are some existing works introducing AoI analysis or optimization to some specific IoT scenarios. Qian *et al.* [12] have derived a policy-independent lower bound on the long-term average AoI in a multi-channel IoT system and proposed a matching strategy to optimize the AoI. Lou *et al.* [13] investigated the relationship between system AoI and throughput in a multi-hop wireless network. Zhang *et al.* [14] studied the effect of edge caching on AoI, proposed a cache update strategy based on refresh windows, and derived the approximate expressions for the average AoI and latency.

Besides data transmission, some researchers paid their attention to the processing of the updated status information when studying the AoI issue. It has been accepted that the data processing time could be also part of the AoI in quantitative analysis. Arafa *et al.* [15] considered the service time for computing status updates information and showed that preemptive updates are beneficial for minimizing the AoI after it reaches a certain threshold. Song *et al.* [16] proposed a similar metric namely task age as their minimization objective, and designed an age-based task scheduling and computational offloading scheme in mobile edge computing systems.

Recently, there have been several works that tried to generalize the basic concept of AoI to be applied in data semantic modeling and design patterns for IoT. Chiariotti *et al.* [17] proposed the concept of Age of Information Query (QAoI), which is defined as the freshness of information at a query instance. Such measurement is quite valuable in pull-based communication scenarios (e.g., satellite communication) where the communication channel is time-varying, and the channel error rate is high. Maatouk *et al.* [18] presented a metric called Age of Incorrect Information (AoII), which measures the aging degree of destination information since the latest update of the source data. They focused on the content of the data, which is a promising area of the AoI study in data semantics. Moskowitz *et al.* [19] proposed Value of Information (VoI), to measure the usefulness of information for a certain specific goal. They studied and evaluated the value of data, according to which more computational or storage resources should be allocated to high-value data items instead of redundant or low-utility data items.

In IoT, the power supply of the IoT devices is commonly from the batteries (rechargeable or non-rechargeable). With such limited energy resources, the efficient energy consumption is a critical concern. A few existing works tried to optimize the energy efficiency in communication or computing when studying AoI in IoT. Ceran *et al.* [20] derived an optimal channel transmission strategy for energy limited IoT devices, with the objective of minimizing the average AoI of the system. Kuang *et al.* [21] studied the AoI optimization for computation-intensive applications, where mobile edge computing was introduced for processing the data at the edge of the network. Optimal Schemes of computation offloading and network resource management were proposed.

With the rapid development and growing popularity of IoT, traditional optimization algorithms face challenging in extremely high complexity and search space [22, 23]. DRL has been introduced for solving problems with large state or/and action space. Xu *et al.* [24] applied deep reinforcement learning in task offloading for UAV-assisted IoT systems. Chen *et al.* [25] proposed a multi-task DRL algorithm of resource allocation for reducing service latency. Lee *et al.* [26] applied DRL to resource allocation in dynamic wireless networks. Chen *et al.* [27] proposed the DRL algorithm to solve the problem of the collaborative and dynamic task offloading for IoT devices. Chen *et al.* [28] transformed the optimal resource management problem into a Markov decision process (MDP) and designed a dynamic resource management algorithm based on DRL to solve this problem in mobile edge computing scenarios. Abd-Elmagid *et al.* [29] used DRL to jointly optimize the wireless energy transfer and scheduling of update packet transmissions

with the objective of minimizing the long-term average AoI.

Though these approaches can model and optimize AoI, the AoI issue in IIoT remains largely unexplored. In IIoT, multiple IoT devices connect to the edge servers with bursty workload. With limited energy resources on the devices, how to efficiently transmit or/and process sensing data while keeping information updated especially in large-scale dynamic IIoT environment is a challenging task. Our ECCO approach, to be described next, is designed to fill this gap.

3. System Model and Problem Formulation

3.1. Scenario

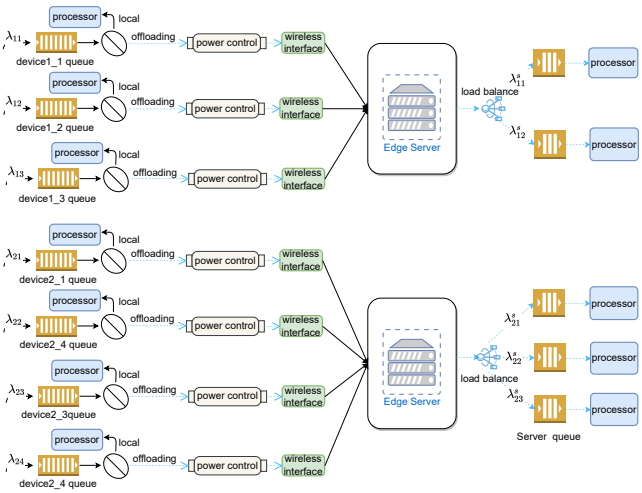

Figure 1. Scenario of an IIoT system.

In this paper, we consider an IIoT scenario where there are multiple IoT devices and edge servers, shown as Fig. 1. Each IoT device connects to a certain edge server through the wireless network. We consider that the communication between IoT device and edge server is performed via a frequency division multiple access (FDMA) scheme and the channels are pre-assigned [30]. Therefore, there is no interference between different IoT devices. We assume that there is no overlapping area among the base stations, and thus the network management and task scheduling can be treated in a parallel way. Each group consists of a set of $\mathcal{N} = \{1, 2, ..., N\}$ IoT devices and one edge server where there are a set of $C = \{1, 2, ..., c\}$ virtual machines deployed for data processing.

Each IoT device collects sensing data and generates a series of system status updates. This paper considers a discrete time slot system where time is divided into equal-length time slots. Upon arrival of the requests from an IoT device, our objective is to dynamically choose its local processor or an edge server to handle the status updates according to the network environment at each time slot t. For the local processing, certain amount of energy has to be consumed on the IoT device; while for edge processing, the IoT device has to transmit its data to the edge server at the base station and request the edge server for processing, which also takes energy consumption, communication delay and processing time. The trade-off should be carefully addressed in order to obtain optimal energy efficiency while keeping the data fresh.

3.2. System Model

3.2.1. Workload model

Tasks are initiated from IoT devices in an IIoT system. It has been commonly accepted in existing related literature that the task arrivals can be approximated by a Poisson process [31]. We also analyze a dataset collected by Shanghai Telecom in China, which contains over 7.2 million records of user service behaviors on 3,233 based stations in the year of 2014 [32, 33]. We find from the real-life dataset that the interarrival times conform to exponential distribution. We defer readers to Section 5 for detailed analytical results.

At the beginning of time slot t, the probability that IoT device i chooses to offload its requests to the edge server is denoted by $P_i^S(t)$, where $0 \le P_i^S(t) \le 1$. With the property of Poisson process, we know that the task arrivals at the edge server also follow a Poisson process with an average rate of $P_i^S(t)\lambda_i$. Meanwhile, the arrival rate of status updates being processed at the local IoT device can be expressed by $(1 - P_i^S(t))\lambda_i$.

The average data size of requests generated from the i-th IoT device is expressed by θ_i. Similar to existing works, we assume that the size of workload (data) conforms to an exponential distribution.

3.2.2. Computation model

a) Local computing model: Some lightweight workload can be handled locally at the IoT device. Since there is no data communications, we only have to calculate the processing time spent with the status information from the IoT devices. With the workload model, the IoT device can be formulated by an $M/M/1$ queueing model. With queueing theory, the average response time $T_i^A(t)$ for locally processed requests on IoT device i is expressed by (1), where μ_i^A is the service rate indicating the computing capability of IoT device i.

$$T_i^A(t) = \frac{1}{\mu_i^A - (1 - P_i^S(t))\lambda_i}, \quad i \in \mathcal{N}. \tag{1}$$

b) Edge computing model: Multiple IoT devices connect to the edge server, and thus their requests may be converged at the edge node. We have known that each IoT device submits its requests to the edge server following a Poisson distribution. The superposition of all the Poisson processes from the IoT devices is also a Poisson process, with the arrival rate shown by,

$$\lambda_{total}^A(t) = \sum_{i=1}^{N} P_i^S(t)\lambda_i. \tag{2}$$

For avoiding queue congestion which may result in unexpected performance degradation, there might be an access control scheme limiting the maximum workload on the edge server for stabling the queue status. We let λ_{max}^S denote the maximum

arrival rate at the edge server. Therefore, the arrival rate at the edge server can be expressed as follows.

$$\lambda_p^S(t) = \begin{cases} \lambda_{total}^A(t), & \lambda_{total}^A(t) \le \lambda_{max}^S \\ \lambda_{max}^S, & \lambda_{total}^A(t) > \lambda_{max}^S, \end{cases} \tag{3}$$

It should be noted that the edge server has c homogeneous virtual machines, which can handle multiple status updates requests from IoT devices simultaneously. We assume that the service rate of each virtual machine is denoted by μ^S. Therefore, the edge server can be modeled by an $M/M/c$ queue. Its utilization can be calculated by the following equation.

$$\rho^S(t) = \frac{\lambda_p^S(t)}{c\mu^S}. \tag{4}$$

The average waiting time for processing the requests contains queueing time and service time. The average queueing time for each status updates in the edge server is:

$$T_w(t) = \frac{G(t)}{c\mu^S - \lambda_p^S(t)}, \tag{5}$$

where,

$$G(t) = \frac{\frac{(c\rho^S(t))^c}{c!}\frac{1}{1-\rho^S(t)}}{\sum_{k=0}^{c-1}\frac{(c\rho^S(t))^k}{k!} + \frac{(c\rho^S(t))^c}{c!}\frac{1}{1-\rho^S(t)}}. \tag{6}$$

Eq. (6) is also known as Erlang's formula.

The average service time in the edge server can be expressed as follows:

$$T_s = \frac{1}{\mu^S}. \tag{7}$$

From (5), (6), and (7), the average waiting time for computation offloading can be expressed as:

$$T_{wait}^S(t) = T_w(t) + T_s. \tag{8}$$

3.2.3. Communication model

If the IoT devices choose to offload the status updates information to the edge server, additional data transmission time is incurred. At time slot t, the transmission power of IoT device i is denoted by $P_i(t), 0 < P_i(t) < P_i^{max}$, where P_i^{max} is the maximum transmission power of IoT device i. The channel power gain between the i-th IoT device to the edge server is h_i. Then, according to Shannon Theorem which has been widely applied in performance evaluation of data transmission [7, 28], we can obtain the uplink data transmission rate for the IoT device i computation offloading as follows:

$$R_i(t) = \omega log_2(1 + \frac{P_i(t)h_i}{\sigma^2}), \tag{9}$$

where ω is the wireless channel bandwidth and σ^2 is average Gaussian white noise. Similar to existing works [34], in order to facilitate the analysis, we assume that the wireless channels are shared orthogonal among different IoT devices. From (9) we can obtain that the transmission time for IoT device i computation offloading is expressed as follows:

$$T_i^t(t) = \frac{(P_i^S(t)\lambda_i)\theta_i}{R_i(t)}, \tag{10}$$

Commonly, similar to most of the previous works [7, 35], the size of the calculating results is relatively small to the original data size, and thus the time for sending back the result and the corresponding energy consumption can be ignored.

3.2.4. AoI model

AoI has recently been proposed to describe the freshness of status information in data processing. Formally, in this paper, we define AoI as the time consumed from the generation of status information at an IoT device to the completion of data processing at either the IoT device or the edge server.

With the mathematical models presented above, specifically with (1), (8) and (10), the AoI at time slot t can be quantified as,

$$T_{AOI}(t) = T_i(t) + T_S(t), \tag{11}$$

where $T_i(t)$ is the time consumed by local processing to process and send status updates, $T_i(t)$ is expressed as follows:

$$T_i(t) = (1 - P_i^S(t))T_i^A(t) + P_i^S(t)T_i^t(t). \tag{12}$$

The second part of the AoI on the right-hand side of (11) is the time consumed by the edge server to process the status information, which can be expressed as:

$$T_S(t) = P_i^S(t)T_{wait}^S(t). \tag{13}$$

Thus, from (11) we can obtain the average AoI as follows:

$$T_{AOI} = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\{T_{AOI}(t)\} \tag{14}$$

$$\begin{aligned} &= \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\{(1 - P_i^S(t))\frac{1}{\mu_i^A - (1 - P_i^S(t))\lambda_i} \\ &+ P_i^S(t)(\frac{(P_i^S(t)\lambda_i)\theta_i}{\omega log_2(1 + \frac{P_i(t)h_i}{\sigma^2})} \\ &+ P_i^S(t)(\frac{\frac{\frac{(c\rho^S(t))^c}{c!}\frac{1}{1-\rho^S(t)}}{\sum_{k=0}^{c-1}\frac{(c\rho^S(t))^k}{k!} + \frac{(c\rho^S(t))^c}{c!}\frac{1}{1-\rho^S(t)}}}{c\mu^S - \lambda_p^S(t)(t)} + \frac{1}{\mu^S})\}. \end{aligned} \tag{15}$$

3.2.5. Energy model

The energy consumption on the i-th IoT device basically consists of two parts, including (1) the energy consumed for local processing of the status information, and (2) the power consumed by transmitting sensing data to the edge servers. For the first part, the energy consumption for data processing is closely related to the CPU frequency of the IoT device. It has been widely accepted that the energy consumption in time slot t can

be calculated by (16) where f is the CPU frequency of IoT devices and κ_i is the effective capacitance parameter of IoT device i.

$$E_i^A(t) = \kappa_i f^3 T_i^A(t), \tag{16}$$

For the second part, transmission power consumption can be calculated by the product of the power rate $P_i(t)$ and the transmission time $T_i^t(t)$. Formally, it can be expressed as:

$$E_i^S(t) = P_i(t)T_i^t(t) = P_i(t)\frac{P_i^S(t)\lambda_i\theta_i}{R_i(t)}. \tag{17}$$

According to (16) and (17), the average energy consumption in the IIoT system can be expressed as follows:

$$E_{ave} = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\{(1-P_i^S(t))E_i^A + P_i^S(t)E_i^S\} \tag{18}$$

$$= \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T}\{(1-P_i^S(t))\left(\frac{\kappa_i f^3}{\mu_i^A - (1-P_i^S(t))\lambda_i}\right) + P_i^S(t)(P_i(t)\frac{P_i^S(t)\lambda_i\theta_i}{\omega log_2(1+\frac{P_i(t)h_i}{\sigma^2})})\} \tag{19}$$

3.3. Problem Formulation

In IIoT, our goal of network resource management and computation offloading is to minimize the energy consumption on the IoT devices while guaranteeing the freshness of the sensing data. The network environment as well as task arrivals are time-varying, however the objective is a long-term cumulative minimization. Consequently, we formulate an optimization problem as follows.

$$\mathbf{P1}: \min_{P_i^S(t),P_i(t)} E_{ave}, \tag{20}$$

$$s.t.\ T_{AOI}(t) \le \varepsilon, \forall i \in \mathcal{N}, \tag{21}$$

$$0 \le P_i^S(t) \le 1, \forall i \in \mathcal{N}, \tag{22}$$

$$0 \le P_i(t) \le P_i^{max}, \forall i \in \mathcal{N}, \tag{23}$$

$$(1-P_i^S(t))\lambda_i < \mu_i^A, \forall i \in \mathcal{N}, \tag{24}$$

$$\lambda_p^S(t) < c\mu^S, \forall i \in \mathcal{N}. \tag{25}$$

The objective function defined by (20) is minimizing the long-term cumulative average energy consumption. Eq. (21) indicates that the AoI should be bounded in an acceptable range for keeping the sensing data fresh. Constraint (22) represents that the offloading probability between local processing and edge processing should be in the scope between 0 and 1, while (23) bounds the maximum transmission power of each IoT device. Eqs. (24) and (25) guarantee the queue stability of each IoT device and the edge server, respectively.

4. DRL-Based Algorithm Design

4.1. MDP Formulation

The problem **P1** presented in the previous section is a dynamic optimization problem. To solve the problem according to the time-varying IIoT networking environments, we reformulate **P1** by a Markov Decision Process (MDP). It can be denoted by a four-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}\}$, where $\mathcal{S}$ represents the system states, $\mathcal{A}$ indicates the action space, $\mathcal{R}$ is the reward functions, and $\mathcal{T} = p(S(t+1)|S(t), A(t))$ is the transition probability between states.

4.1.1. State Space

At each time slot t, the system state $S(t) = (\mathbf{\Theta}_s(t), \mathbf{\Theta}_e(t))$ is defined as a two-tuple indicating the queueing states of the IoT devices and the edge server respectively. The specific definitions are shown as follows.

- $\mathbf{\Theta}_s(t)$ is defined as the number of requests waiting to be processed at each IoT device, which can be expressed by (26) where $\mathrm{Q}_{s_i}(t)$ denotes the number of tasks in the buffer of IoT device i.

$$\mathbf{\Theta}_s(t) = \{\mathrm{Q}_{s_1}(t), \mathrm{Q}_{s_2}(t), ..., \mathrm{Q}_{s_i}(t), ..., \mathrm{Q}_{s_N}(t)\}, \tag{26}$$

- $\mathbf{\Theta}_e(t)$ is the queueing state of the edge server, expressed as:

$$\mathbf{\Theta}_e(t) = \{\mathrm{Q}_{e_1}(t), \mathrm{Q}_{e_2}(t), ..., \mathrm{Q}_{e_j}(t), ..., \mathrm{Q}_{e_c}(t)\}, \tag{27}$$

where $\mathrm{Q}_{e_j}(t)$ is the number of requests buffered at the j-th VM on the edge server.

4.1.2. Action Space

At each time slot t, the agent performs action $A(t) = (\gamma_t, \delta_t, \epsilon_t)$ to compute the offloading decision of IoT devices based on the current state $S(t)$ in the system.

- γ_t represents the offloading probability of IoT devices, which can be expressed as:

$$\gamma_t = \{P_1^S(t), P_2^S(t), ..., P_i^S(t), ..., P_N^S(t)\}. \tag{28}$$

- δ_t represents the transmission power of IoT devices, which can be expressed as:

$$\delta_t = \{P_1(t), P_2(t), ..., P_i(t), ..., P_N(t)\}. \tag{29}$$

- ϵ_t represents the scheduling policies of IoT devices, which can be expressed as:

$$\epsilon_t = \{W_1(t), W_2(t), ..., W_i(t), ..., W_N(t)\}. \tag{30}$$

Therefore, the action space $\mathcal{A}(t)$ can be expressed as follows:

$$\begin{aligned} A(t) = \{&(P_1^S(t), P_2^S(t), ..., P_i^S(t), ..., P_N^S(t)),\\ &(P_1(t), P_2(t), ..., P_i(t), ..., P_N(t)),\\ &(W_1(t), W_2(t), ..., W_i(t), ..., W_N(t))\}. \end{aligned} \tag{31}$$

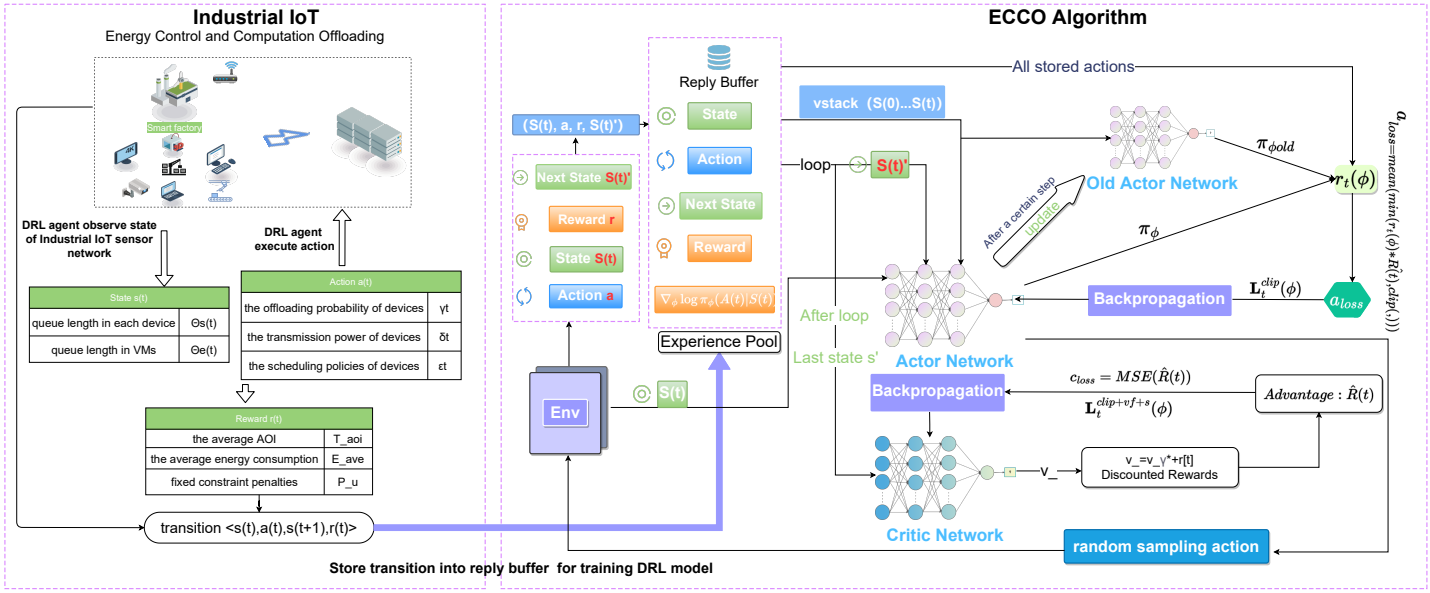

Figure 2. The overview of ECCO

4.1.3. Reward Function

Within the time slot t, the agent calculates the reward $R(t)$ after the action $A(t)$ is performed at the state $S(t)$. Considering that our goal is to minimize the energy consumption of the system under the average AoI constraint, we define the reward function as follows:

$$R(t) = \sum_{i \in N} \mathbb{E}\left[E_{ave}(t) + T_{AOI}(t) + \Psi(t)\right]. \tag{32}$$

In (32), besides the energy consumption $E_{ave}(t)$, we put the AoI value in the reward function for minimization in our MDP. Meanwhile, we define a penalty function $\Psi(t)$ to guarantee the constrains in **P1** being satisfied. Specifically, if any constraint is unsatisfied, a positive large value will be added to our reward function.

4.2. Deep Reinforcement Learning Algorithm

From the MDP formulation presented in the previous subsection, we find that our action space is three-dimensional and continuous. With the increase of the number of IoT devices in the system, the action space grows exponentially. Traditional MDP algorithms such as value iteration or policy iteration or even Approximate Dynamic Programming (ADP) techniques may suffer from space explosion problem in solving the MDP. To attack this challenge, we apply deep reinforcement learning for finding the optimal policy. Based on Proximal Policy Optimization (PPO) algorithm which is a popular policy gradient method that can be used for environments with continuous action spaces, we design an Energy Control and Computation Offloading (ECCO) algorithm, shown by Fig. 2.

The basic idea of our approach is to introduce the techniques of deep neural network for solving the original MDP problem. We define a data structure namely "Env" to capture the basic formulation of the MDP problem, where we let $S(t)$ denote the current state, r express the reward function, a represent the action, and $S(t)'$ denote the next state after the action has taken. The neural networks use their experience to examine the actions in the action space, and inference the optimal policy for maximizing the expected long-term reward value.

In our approach, two types of deep neural networks are involved, namely actor and critic. The actor maps the observation to an action and the critic gives an expectation of the rewards of the agent for the given state. Specifically, we deploy two actor networks for updating the policy and we want to ensure that the update is not too large.

Policy gradient is applied for the agent to learn, the goal of which is to optimize the parameter ϕ to obtain the optimal policy. We take the gradient estimator shown as below:

$$\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\phi \log \pi_\phi(A(t)|S(t))\hat{R}(t)\right], \tag{33}$$

where π_ϕ is a stochastic parameterized policy with parameter ϕ, and $\hat{R}(t)$ is an estimation of the advantage function at time slot t.

The learning ratio is a critical concern in policy gradient. In order to control the learning ratio in an acceptable range, we introduce a metric to evaluation the difference between the old and new policies, which is formally defined as (34).

$$r_t(\phi) = \frac{\pi_\phi(A(t)|S(t))}{\pi_{\phi old}(A(t)|S(t))} \tag{34}$$

The actor network bounds the update between the policies within an acceptable ratio of ϵ. To this end, we define the loss function of the actor network with a clip function with ratio of ϵ, as follows:

$$\mathbf{L}_t^{\text{clip}}(\phi) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\phi)\hat{R}(t), clip(r_t(\phi), 1-\epsilon, 1+\epsilon)\hat{R}(t)\right)\right], \tag{35}$$

where $\hat{\mathbb{E}}_t$ represents the empirical average of a sequence of samples. The clip function is designed to control the sampling range of our reinforcement learning process. By introducing a hyperparameter ϵ, the update of the neural network parameters in each epoch can be bounded in an acceptable interval $[1-\epsilon, 1+\epsilon]$.

After the actor is updated, we take out the data from the experience pool for calculation to update the critic. To do so, we first calculate the advantage function using the following formula:

$$\hat{R}(t) = \varphi_t + (\gamma)\varphi_{t+1} + \cdots + \cdots + (\gamma)^{T-t+1}\varphi_{T-1}, \quad (36)$$

where,

$$\varphi_t = R(t) + \gamma \mathbf{V}(S(t+1)) - \mathbf{V}(S(t)). \quad (37)$$

Here, $\mathbf{V}(S(t+1))$ $\mathbf{V}(S(t))$ can be obtained from the critic network, and $R(t)$ is the reward function. Then, we use mean squared error (MSE) as the Loss function to train the Critic network. Since we are taking a network architecture where actor and critic share parameters, we have to use a loss function that combines the policy agent and the value function error term as follows:

$$\mathbf{L}_t^{\text{clip+vf+s}}(\phi) = \mathbb{E}_t\left[\mathbf{L}_t^{\text{clip}}(\phi) - c_1\mathbf{L}_t^{vf}(\phi) + c_2\mathbf{S}[\pi_\phi](S(t))\right], \quad (38)$$

where $\mathbf{S}[\pi_\phi](S(t))$ represents the entropy bonus illustrating the possibility of the critic network globally exploring new policies, $\mathbf{L}_t^{vf}(\phi)$ is the MSE estimator of the advantage function of the critic network, and c_1, c_2 are hyperparameters.

The details of the ECCO are given in Algorithm 1.

Algorithm 1 Energy Control and Computation Offloading (ECCO) Algorithm

Input: Training episode number T_{epi}; training step number T_{step}; model update frequency $\mathbf{U}$; decay factor ζ; model update number t_epochs;

Output: Offloading probability of IoT devices γ_t; transmission power of IoT devices δ_t; scheduling policy of IoT devices ϵ_t.

```
Initialize random seed, experience replay pool D, system environment, network parameters
Set time_steps = 0
for episode = 1, ..., T_epi do
    Reset the environment to get the initial state S(0)
    Set reward = 0, energy = 0
    for step = 1, ..., T_step do
        Obtain action A(t) from actor network
        Obtain the next state S(t+1) with A(t), calculate reward R(t)
        Store ⟨S(t), A(t), S(t+1), R(t)⟩ into D
        if step mod U = 0 then
            for i = 1, ..., t_epochs do
                Randomly sample s data from D
                Calculate the decay reward discount_reward = r + ζ * discount_reward
                r ⇐ discount_reward
                Calculate the advantage function using Eq. (36)
                Calculate optimization objectives using Eq. (38)
                Update network parameters φ
            end for
            Save the updated network parameters Π_old ⇐ Π_φ
        end if
    end for
end for
```

5. Performance Evaluation

5.1. Experimental Setup

5.1.1. Dataset

We evaluate the efficiency of our algorithm through extensive experimental simulations. The workload in our experiments is generated according to real-life trace data, and the edge servers (with base stations) are placed following the geographical distribution of base stations in the city of Shanghai, China. The dataset we use in our experiment is obtained from Shanghai Telecom [32, 33], which contains more than 7.2 million records of Internet access trace data through 3,233 base stations generated by 9,481 mobile users within a time period of 6 months.

Before conducting our experiments, we first analyze the trace data to validate our workload model presented in Section 3. We extract the timestamps of the user requests from the dataset, and analyze the interarrival times by plotting the probability density function (PDF) and cumulative distribution function (CDF). The analytical results are shown by Fig. 3 and Fig. 4. Intuitively, we find the interarrival times conform to an exponential distribution. Furthermore, we apply non-linear least squares estimator to generate a fitted function of the data distribution. Coefficient of determination, also well-known as R^2, is used to statistically quantify how well the fitted function describes the data. For both CDF and PDF, we find that exponential distribution is a perfect approximation with R^2 values above 0.99. Therefore, it is reasonable for us to use Poisson process to construct the workload model.

5.1.2. Experimental Parameters

In our simulation experiments, we assume that there are 4 virtual machines deployed on each edge server. The tasks are generated according to the Telecom dataset, and the size of each task (size of updated sensing data) is a random variable following a uniform distribution in the range from 1MB to 5MB. We set that the CPU frequency of IoT device is 1GHz while the CPU frequency of the edge server is 4.4GHz. The wireless channel bandwidth is set to be 1MHz. The detailed parameter settings about the system are shown in Table 1.

5.2. Experimental Results

In the following experiments, we compare our ECCO algorithm with three existing DRL algorithms which have been widely used in solving dynamic optimization problems with continuous action space, listed as follows.

- **DDPG**: Deep Deterministic Policy Gradient (DDPG) is a popular DRL algorithm whose main idea is to use the critic network to find the possibly optimal decision (POD),

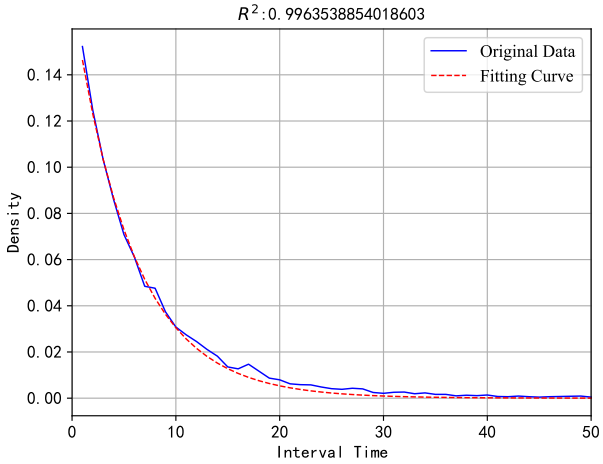

Figure 3. PDF of the interarrival time of an IoT device

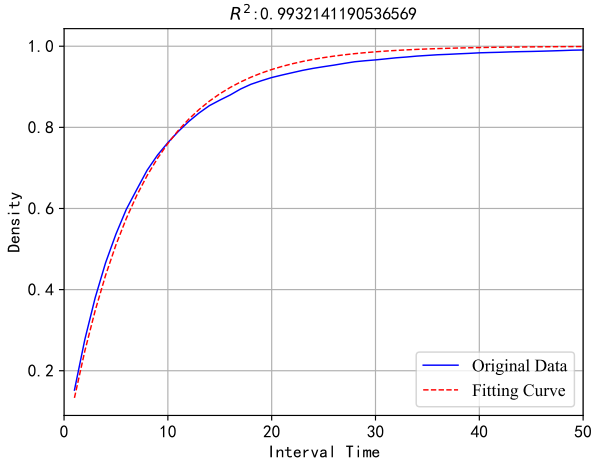

Figure 4. CDF of the interarrival time of an IoT device

and subsequently use the POD to optimize the policy actor network. The generation of the policy is completely dependent on the critic network without caring about the actual benefits.

- **SAC**: Soft Actor-Critic (SAC) is an off-policy DRL algorithm with a stochastic actor. The actor network in the SAC algorithm aims to maximize the gain and the entropy at the same time. By combining off-policy updates with a stable stochastic actor-critic formulation, SAC is stable in taking optimal actions in dynamic environments.

- **TD3**: Twin Delayed Deep Deterministic Policy Gradient (TD3) is designed to solve the overestimation problem on Q-values in DDPG. It uses two sets of networks to represent different Q values, and suppresses persistent overestimation by choosing the smallest one as the updated target. It also adds noise to the output action **A** of the actor target network to improve the algorithm stability.

Table 1. IIoT Parameter Settings

Notations	Definitions	Value
C	number of VMs	4
θ	task size	1MB~5MB
f	the computational frequency of IoT devices	1GHz
μ_S	the computational frequency of edge server	4.4GHz
ω	bandwidth of IoT device	1MHz
p_i^{max}	the max transmission power of IoT device	100mW
σ^2	Gaussian white noise	-100dBm

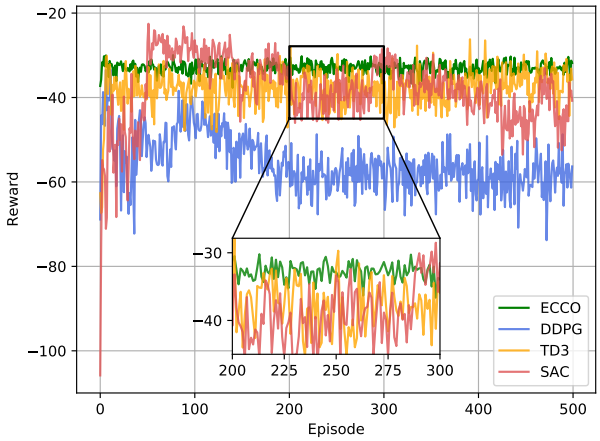

Figure 5. Reward comparisons of different DRL algorithms during training

We train our ECCO algorithm as well as the other three DRL algorithms for 500 iterations, and examine the experimental results to compare the convergence rate and performance of the four approaches. Fig. 5 shows the reward values in the first 500 epochs. We should note that, in order to facilitate our experiments, we define the reward function as the negative summation of the energy consumption, AoI and the penalty and thus convert the original problem into a traditional reward maximization problem. We obtain from the experimental results that our ECCO algorithm and the TD3 algorithm gradually converge during the first 20 epochs, and ECCO is slightly faster in convergence rate. The DDPG algorithm tends to converge with a slower speed, while the SAC algorithm does not converge in 500 iterations. Among all the four algorithms, our ECCO obtains the highest reward on average and its performance is the most stable.

Fig. 6 illustrates the average energy consumption of the four algorithms during the first 500 training iterations. We have the similar conclusion. The SAC algorithm is not able to stabilize in 500 epochs, while the DDPG algorithm and TD3 converge at similar speed but have a higher average energy consumption.

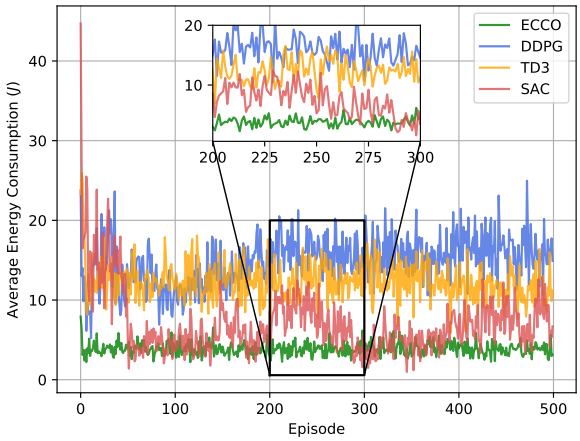

Figure 6. Energy consumption comparisons of different DRL algorithms during training

Our ECCO algorithm performs best in both convergence rate and energy consumption minimization.

Then, we tune the parameters to see their impact to the obtained policy. In order to further validate the performance of our approach, we further introduce a non-DRL random algorithm for comparison where tasks are randomly scheduled to either local IoT device or the edge server. Fig. 7 illustrates the experimental results when we tune the parameter of maximum transmission power P_i^{max}. Here, the dots and solid lines indicate the average values, while the shade areas represents the variation in different experimental tryouts. With the increase of P_i^{max}, the constraint on the networking communication power is relaxed, and hence the IoT device may choose higher transmitting rate in order to reduce AoI, resulting in the overall energy consumption going up. Among all the five schemes, random algorithm performs the worst because it is not intelligent to make optimal decisions. Although SAC is good at reducing the energy consumption, it is the most unstable. Our ECCO algorithm performs the best in terms of energy minimization and its stability is also acceptable.

Fig. 8 shows the average energy consumption of the IoT devices with different AoI constraints. With the increase of ε, less penalty will be paid to the growth of AoI, and thus the IoT device may choose to process or transit data with lower speed for energy reduction. Therefore, the average energy consumption decreases in most experiments. Also, the experimental results prove that our ECCO algorithm always has the best performance in terms of energy reduction.

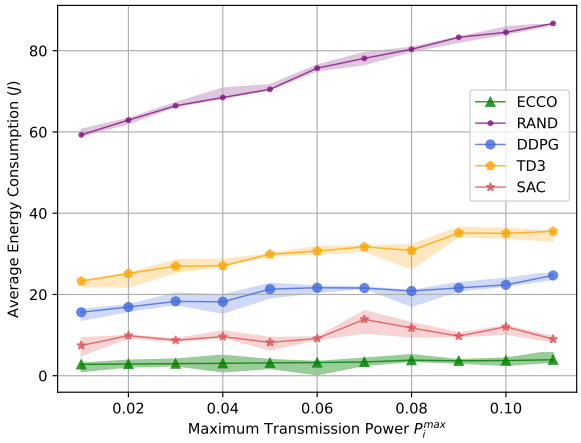

Figure 7. Average energy consumption with different transmission power

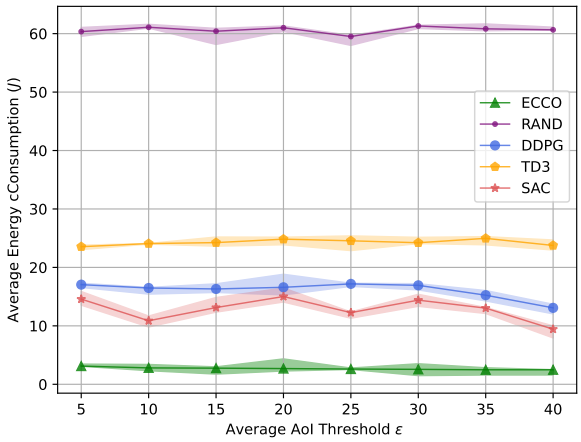

Figure 8. Average energy consumption with different AoI threshold

6. Conclusion

In this paper, we study the optimal AoI-aware network resource management and computation offloading problem within a dynamic IIoT scenario with multiple IoT devices and multiple edge servers. Considering the IoT devices with limited networking, computing and energy resources, we construct a dynamic optimization problem for minimizing the long-term av-

erage computation energy consumption under the average AoI constraint. Mathematical models are presented for performance evaluation, and a DRL-based algorithm is designed for dynamic optimization. We conduct simulation experiments based on realistic telecommunication data, and the comparison results validate the efficacy of our approach.

7. Acknowledgement

This work is supported by Beijing Nova Program (No. Z201 100006820082), National Natural Science Foundation of China (Nos. 61972414 and 61902029), and Beijing Natural Science Foundation (No. 4202066).

References

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, M. Gidlund, Industrial internet of things: Challenges, opportunities, and directions, IEEE transactions on industrial informatics 14 (11) (2018) 4724–4734.

[2] Y. Chen, F. Zhao, X. Chen, Y. Wu, Efficient multi-vehicle task offloading for mobile edge computing in 6G networks, IEEE Transactions on Vehicular Technology 71 (5) (2022) 4584–4595.

[3] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656.

[4] S. G. Pease, R. Trueman, C. Davies, J. Grosberg, K. H. Yau, N. Kaur, P. Conway, A. West, An intelligent real-time cyber-physical toolset for energy and process prediction and optimisation in the future industrial internet of things, Future Generation Computer Systems 79 (2018) 815–829.

[5] S. Kaul, R. Yates, M. Gruteser, Real-time status: How often should one update?, in: 2012 Proceedings IEEE INFOCOM, IEEE, 2012, pp. 2731–2735.

[6] A. Kosta, N. Pappas, V. Angelakis, Age of information: A new concept, metric, and tool, Foundations and Trends in Networking 12 (3) (2017) 162–259.

[7] R. Li, Q. Ma, J. Gong, Z. Zhou, X. Chen, Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time iot applications, IEEE Internet of Things Journal 8 (19) (2021) 14471–14484.

[8] J. Huang, C. Zhang, J. Zhang, A multi-queue approach of energy efficient task scheduling for sensor hubs, Chinese Journal of Electronics 29 (2020) 242–247.

[9] J. Huang, B. Lv, Y. Wu, et al., Dynamic admission control and resource allocation for mobile edge computing enabled small cell network, IEEE Transactions on Vehicular Technology 71 (2) (2022) 1964–1973.

[10] Y. Chen, H. Xing, Z. Ma, et al., Cost-efficient edge caching for NOMA-enabled IoT services, China Communications.

[11] J. P. Champati, R. R. Avula, T. J. Oechtering, J. Gross, On the minimum achievable age of information for general service-time distributions, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 456–465.

[12] Z. Qian, F. Wu, J. Pan, K. Srinivasan, N. B. Shroff, Minimizing age of information in multi-channel time-sensitive information update systems, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 446–455.

[13] J. Lou, X. Yuan, S. Kompella, N.-F. Tzeng, AoI and throughput tradeoffs in routing-aware multi-hop wireless networks, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 476–485.

[14] S. Zhang, L. Wang, H. Luo, X. Ma, S. Zhou, AoI-delay tradeoff in mobile edge caching with freshness-aware content refreshing, IEEE Transactions on Wireless Communications 20 (8) (2021) 5329–5342.

[15] A. Arafa, R. D. Yates, H. V. Poor, Timely cloud computing: Preemption and waiting, in: 2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2019, pp. 528–535.

[16] X. Song, X. Qin, Y. Tao, B. Liu, P. Zhang, Age based task scheduling and computation offloading in mobile-edge computing systems, in: 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW), IEEE, 2019, pp. 1–6.

[17] F. Chiariotti, J. Holm, A. E. Kalør, B. Soret, S. K. Jensen, T. B. Pedersen, P. Popovski, Query age of information: Freshness in pull-based communication, IEEE Transactions on Communications 70 (3) (2022) 1606–1622. doi:10.1109/TCOMM.2022.3141786.

[18] A. Maatouk, S. Kriouile, M. Assaad, A. Ephremides, The age of incorrect information: A new performance metric for status updates, IEEE/ACM Transactions on Networking 28 (5) (2020) 2215–2228. doi:10.1109/TNET.2020.3005549.

[19] Chapter 9 - the value of information and the internet of things, in: W. Lawless, R. Mittu, D. Sofge, I. S. Moskowitz, S. Russell (Eds.), Artificial Intelligence for the Internet of Everything, Academic Press, 2019, pp. 145–169.

[20] E. T. Ceran, D. Gündüz, A. György, Reinforcement learning to minimize age of information with an energy harvesting sensor with harq and sensing cost, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019, pp. 656–661.

[21] Q. Kuang, J. Gong, X. Chen, X. Ma, Analysis on computation-intensive status update in mobile edge computing, IEEE Transactions on Vehicular Technology 69 (4) (2020) 4353–4366.

[22] J. Huang, Z. Tong, Z. Feng, Geographical POI recommendation for Internet of Things: A federated learning approach using matrix factorization, International Journal of Communication Systems doi:10.1002/dac.5161.

[23] Y. Chen, F. Zhao, Y. Lu, X. Chen, Dynamic task offloading for mobile edge computing with hybrid energy supply, Tsinghua Science and Technology doi:10.26599/TST.2021.9010050.

[24] J. Xu, D. Li, W. Gu, et al., UAV-assisted task offloading for IoT in smart buildings and environment via deep reinforcement learning, Building and Environment doi:10.1016/j.buildenv.2022.109218.

[25] J. Chen, S. Chen, Q. Wang, B. Cao, G. Feng, J. Hu, iraf: A deep reinforcement learning approach for collaborative mobile edge computing iot networks, IEEE Internet of Things Journal 6 (4) (2019) 7011–7024.

[26] H.-S. Lee, J.-Y. Kim, J.-W. Lee, Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach, IEEE Systems Journal 14 (2) (2019) 2589–2592.

[27] Y. Chen, W. Gu, K. Li, Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning, International Journal of Communication Systems doi:10.1002/dac.5154.

[28] C. Ying, L. Zhiyong, Z. Yongchao, el al., Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things, IEEE Transactions on Industrial Informatics 17 (7) (2021) 4925–4934.

[29] M. A. Abd-Elmagid, H. S. Dhillon, N. Pappas, A reinforcement learning framework for optimizing age of information in rf-powered communication systems, IEEE Transactions on Communications 68 (8) (2020) 4747–4760.

[30] X. Lan, Y. Zhang, L. Cai, Q. Chen, Adaptive transmission design for rechargeable wireless sensor network with a mobile sink, IEEE Internet of Things Journal 7 (9) (2020) 9011–9025.

[31] M. Moltafet, M. Leinonen, M. Codreanu, On the age of information in multi-source queueing models, IEEE Transactions on Communications 68 (8) (2020) 5003–5017. doi:10.1109/TCOMM.2020.2997414.

[32] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, X. Shen, Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach, IEEE Transactions on Mobile Computing 20 (3) (2019) 939–951.

[33] Y. Guo, S. Wang, A. Zhou, J. Xu, J. Yuan, C.-H. Hsu, User allocation-aware edge cloud placement in mobile edge computing, Software: Practice and Experience 50 (5) (2020) 489–502.

[34] J. Huang, M. Wang, Y. Wu, Y. Chen, X. Shen, Distributed offloading in overlapping areas of mobile edge computing for Internet of Things, IEEE Internet of Things Journal 9 (15) (2022) 13837–13847.

[35] X. He, S. Wang, X. Wang, S. Xu, J. Ren, Age-based scheduling for monitoring and control applications in mobile edge computing systems, in: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 2022, pp. 1009–1018. doi:10.1109/INFOCOM48880.2022.9796654.