Learn to Schedule: Data Freshness-oriented Intelligent Scheduling in Industrial IoT

Jianhua Tang, *Senior Member, IEEE*, Fangfang Chen, Jiaping Li, and Zilong Liu, *Senior Member, IEEE*

***Abstract*—In the context of the Industrial Internet of Things (IIoT), developing an accurate and timely scheduling policy is essential. Recently, the Age of Incorrect Information (AoII) is proposed for measuring the timeliness and accuracy of certain status information for monitoring/controlling purposes. In this work, we investigate a multi-sensor state updating system in which AoII is used for quantifying information freshness. We aim to find an optimal scheduling policy to minimize the system-wide cost under bandwidth constraint. We first model the source status updates monitored by sensors as Markov chains and the scheduling problem as a constrained Markov decision process (CMDP). It is challenging to solve the formulated CMDP problem by conventional methods, due to the heterogeneity of source status updates in IIoT and the bandwidth constraint. As such, a framework with the aid of deep reinforcement learning, i.e., Order-Preserving Quantization-Based Constrained Reinforcement Learning Algorithm with Historical Adjustment (OPQ-RL_HA) is developed. Furthermore, by integrating it with the Asynchronous Advantage Actor-Critic (A3C) and the Deep Deterministic Policy Gradient (DDPG), two different algorithms are proposed, i.e., OPQ-A3C_HA and OPQ-DDPG_HA. With extensive numerical validation, it is demonstrated that the proposed algorithm has a lower average system-wide cost compared to the benchmark algorithms.**

***Index Terms*—Age of Incorrect Information, Data Freshness, Deep Reinforcement Learning, Constrained Markov Decision Process, Industrial Internet of Things**

I. Introduction

In recent years, the Industrial Internet of Things (IIoT) has become a crucial element of "Industrial 4.0", and it involves the deployment of numerous communication devices (such as sensors, robots, machines, etc) throughout a large area (such as factories, warehouses, nuclear plants, etc) for remote monitoring and controlling [2]. For instance, the status information of manufacturing needs to be timely captured by remote monitors to reduce product defect rates, whilst a real-time delivery of an alert for any disastrous event (such as fire, earthquake, flood, etc) can help save lives and protect property. However, the limited wireless resources pose an important challenge in these real-time monitoring systems. Therefore, maintaining the freshness of task status and performing effective scheduling under limited resources has become crucial. This is key to ensuring the efficient operation of the IIoT system.

In the past decade, a new metric, called Age-of-Information (AoI), has been studied to characterize the freshness in status-updating systems and applications. It is noted that AoI is different from traditional communication latency which corresponds to the time that an information packet travels on the transmission, propagation and queuing. Therefore, AoI can be high (indicating outdated status information), even when communication latency is low, for instance, if the system infrequently sends status packets [3]–[5].

However, AoI captures only the freshness of transmitted packets without considering the content's accuracy, potentially leading to unnecessary status updates. Therefore, the Age of Incorrect Information (AoII) was introduced as a new metric [6]. This metric considers both the accuracy and freshness of information from the perspective of the remote server. Formally, AoII is described as a penalty function whose value increases over time when the monitor is unable to accurately estimate the status of local source (i.e., there is a discrepancy between the remote server's data and the actual status of the source). Conversely, the value of the penalty function is zero when the monitor's estimation is accurate. Intuitively, AoII surpasses AoI in efficiency by mitigating unnecessary data updates. Therefore, AoII is more effective than AoI in resource-constrained scenarios.

After using AoII to describe whether the status of the remote server matches the status of the source, it becomes crucial to strategize on allocating limited wireless resources effectively to support industrial applications. Traditional scheduling methods often rely on simple priority queues or pre-defined rules, which may not be efficient and flexible in complex and dynamic industrial environments [7], [8]. As an advanced machine learning method, Reinforcement Learning (RL) can continuously learn from interactions with the environment and optimize the decision-making process to achieve optimal task scheduling in a constantly changing environment and still perform well in the face of unknown situations [9], [10]. Additionally, given resource limitations, RL's outputs might not consistently fulfill certain physical requirements. Therefore, the RL algorithm needs to be further developed in this context.

A. Related Works

AoI is mainly used to describe the timeliness of information in real-time systems. The research on AoI is mainly divided

J. Tang, F. Chen and J. Li are with the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou 511442, China, e-mails: jtang4@e.ntu.edu.sg, wifannychen59@mail.scut.edu.cn and wijiapingli@mail.scut.edu.cn. Z. Liu is with the School of Computer Science and Electronics Engineering, University of Essex, Colchester CO4 3SQ, UK, e-mail: zilong.liu@essex.ac.uk. The corresponding author is Jiaping Li.

This work of J. Tang was supported in part by the National Key R&D Program of China under Grant 2024YFE0200500, and in part by the Department of Science and Technology of Guangdong Province under Grant 2021QN02X491. The work of Z. Liu was supported in part by the UK Engineering and Physical Sciences Research Council under Grant EP/Y000986/1. Part of this paper [1] has been presented at the 2022 IEEE Global Communications Conference (GLOBECOM), Rio de Janeiro, Brazil, December 2022.

into the following three categories: 1) AoI design of update packages under different access conditions, such as the AoI-oriented UAV resource management problem when the update packets can be sampled and generated at any time by the source node [11]. 2) AoI optimization after the introduction of other technologies, such as the use of Mobile Edge Computing (MEC) technology to further reduce AoI [5]. 3) AoI deformation under different scenario requirements. For example, [12] introduces a metric known as the Age of Changed Information (AoCI), which decreases only when a newly received update packet at the remote server differs from the previous one, and increases otherwise. [13] introduces the Age of Loop Information (AoLI), which simultaneously reflects the passage of time and the changes in transmission rounds during the uplink and downlink processes in the IIoT.

In the same year, AoII was proposed [6] and the follow-up works have focused on exploring various methods to measure the AoII. [14] provides a general expression for AoII, which can establish AoII expressions for different application requirements. [15] derives the average AoII expression of the Hybrid Automatic Repeat request (HARQ) transmission scheme under the Finite Blocklength (FBL) regime. These studies promote the application of AoII in practical communication systems. Given its good balance between accuracy and information freshness of remote servers, AoII has been widely used to solve resource-constrained scheduling problems. [16] investigates the scheduling problem for minimizing AoII in slot-based systems under channels with random delays. [17] studies the optimal scheduling problem for minimizing the AoII in a symmetric/homogeneous binary information source system. [18] investigates a multi-status Markov source system with transmission power constraint. This is done by casting the scheduling problem into a Markov Decision Process (MDP) where the state space and action space are composed of AoII and scheduling action, respectively. However, the aforementioned studies on AoII have only considered single-source systems with homogeneous statuses. For a real IIoT system with multiple sensors/sources, heterogeneous status transitions should be studied [19]. [20] utilizes a threshold strategy to minimize AoII, thereby achieving optimal scheduling decisions in multi-user scenarios. However, this method strongly depends on the selection of the threshold. [21] uses AoII as a metric in semantic communication-based extended reality (XR) applications, employing an optimal exact linear search method to obtain the optimal strategy. However, the time cost of this search algorithm is relatively high. Besides, as the number of sensors increases, the scheduling action space grows exponentially (i.e., the curse of dimension), and it remains a big challenge to find the optimal scheduling policy.

Reinforcement learning can adapt to changes in the environment and unknown dynamics, so it is often used in large-scale complex communication problems in recent years. For example, in [22], reinforcement learning is used to achieve resource allocation that minimizes the AoII in a satellite-based IoT downlink non-orthogonal multiple access (NOMA) system. In [23], a reinforcement learning algorithm is adopted to select the optimal cooperative nodes in the underwater IoT scenario. In reinforcement learning, asynchronous Advantage Actor-Critic (A3C) and Deep Deterministic Policy Gradient (DDPG) have been widely used to quickly solve complex problems in recent years [24]. However, these two algorithms cannot be used to solve constrained problems directly. To ensure that the output of RL satisfies the constraints, [25] directly adds a safety layer to the policy, which can correct each output action. While this method ensures constraint adherence, it could potentially hinder RL performance. [26] proposes the Constrained-Rectified Policy Optimization method (CRPO), which updates the policy alternatingly between objective improvement and constraint satisfaction. [27] introduces the Semi-Infinite Constrained Policy Optimization (SI-CPO) method, which performs a single step of policy optimization along the direction of minimizing the value of reward corresponding to the violated constraint, if the constraint violation exceeds the tolerance. However, these methods suffer from low learning efficiency, highlighting the needs for further exploration in efficiently solving CMDP problems with RL.

Building on our previous work [1], which focused on the unconstrained AoII minimization problem and uses A3C to obtain the optimal scheduling policy, in this paper, we take the bandwidth constraint into account and propose more sophisticated algorithmic solutions.

B. Our Contributions

In this work, we use a multi-sensor monitoring system where the sources need to deal with heterogeneous status data and the updating system is subject to a bandwidth constraint. Considering packet loss, we use ACK/NACK feedback to reflect whether the remote server has successfully received the status packet. We adopt AoII as the metric to quantify the precision and freshness of remote server for the status of a local source. Our goal is to determine the optimal scheduling policy to minimize the system-wide cost. The main contributions of this paper are summarized as follows:

- In solving the scheduling problem of minimizing system-wide cost under bandwidth constraints, to ensure the flexibility of scheduling decisions and the balance of sensor usage, we propose the Constrained Reinforcement Learning with Historical Adjustment (RL_HA) framework. This framework utilizes historical decision outcomes to adjust the scheduling priorities of sensors.
- Due to the complexity of the CMDP, we use an order preserving quantization (OPQ) method to expand the range of optional actions and further enhance the decision-making ability of the model. Combing the OPQ method with Constrained A3C Algorithm and Constrained DDPG Algorithm respectively, we obtain two solution algorithms for the scheduling problem, i.e., OPQ-A3C_HA and OPQ-DDPG_HA.
- Numerical results show that our proposed algorithms can achieve excellent performance (average system-wide cost) in both status-homogeneous and status-heterogeneous systems, and also outperform the baseline algorithm.

The rest of this paper is organized as follows. The system model and problem formulation are introduced in section II.

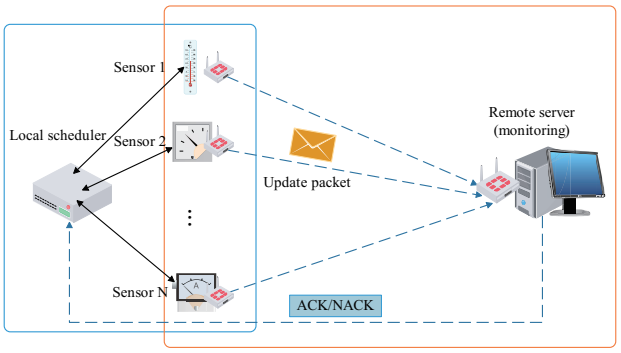


Fig. 1. System model. The left box refers to the wired link between local scheduler and sensors, and the right box represents the wireless link between sensors and remote server.

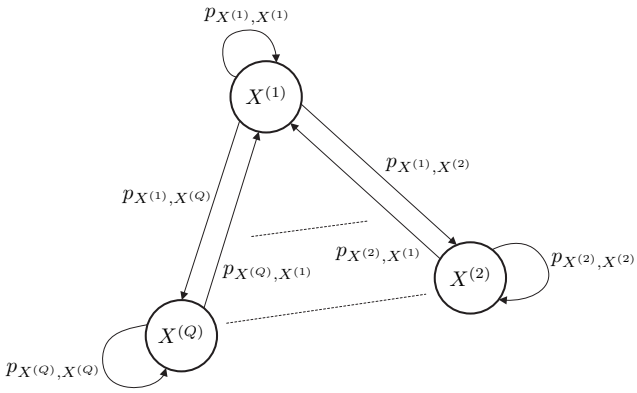


Fig. 2. The transition probability.

In section III, we propose improved algorithms based on reinforcement learning to solve the CMDP. In section IV, the effectiveness of our algorithms is validated, and the two algorithms are compared through numerical results. Finally, section V draws the conclusions.

II. System Model and Problem Formulation

To enhance the processing capability of large-scale systems, we consider the system as shown in Fig. 1. This system includes N sensors to monitor changes in the status of the object/source and transmit update packets to a local scheduler in a simultaneous manner. Assuming that the local scheduler has sufficient capacity, and it determines which (and when) sensors should send update packets to the remote server through wireless networks to maintain the sensing system lightweight and efficient[1]. Besides, the considered system model can also map to the case that the local scheduler directly sends all statuses to the remote server.

On one hand, due to the presence of numerous power sources with ample electricity in actual factories, we consider connecting sensors and local schedulers via wired links and disregard the communication delay between them. On the other hand, wireless links (from sensors to remote server) are prone to errors, if the remote server receives update packets successfully, an ACK feedback is sent back to the local scheduler; Or, the remote server sends an NACK feedback.

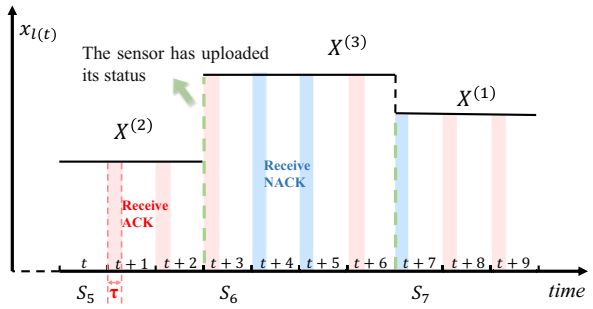
A. Single Sensor Scenario

To illustrate our model, we first consider a single-sensor scenario. For simplicity, let us consider time-slot index t ranging from 1 to T, where T refers to the total number of time-slots. Then, we use the generate-at-will model [28] that the sensor samples the source at the beginning of each time slot, while the local scheduler acquires the source's status. Furthermore, we assume that the local source status only changes at the beginning of certain time-slots. The information process that is sampled by the sensor is classified into Q discrete statuses, i.e., $X^{(1)}, X^{(2)}, ..., X^{(Q)}$, and at each time-slot t, the status indicator is denoted by $x_l(t) \in \{X^{(1)}, X^{(2)}, ..., X^{(Q)}\}$. Moreover, as shown in Fig. 2, we assume $x_l(t)$ as a discrete Markov chain with Q-status and one-step transition probability $p_{X^{(i)},X^{(j)}} = \mathbf{Pr}\left(x_l(t+1) = X^{(j)} | x_l(t) = X^{(i)}\right)$. Meanwhile, let $x_r(t)$ represents the remote server's estimation at time-slot t.

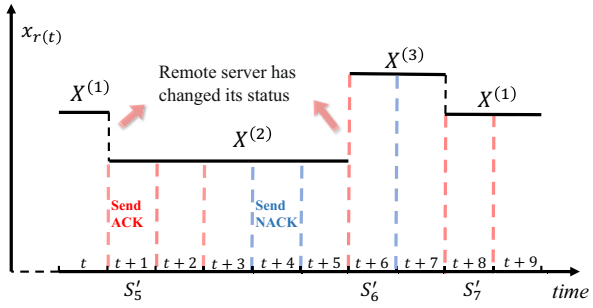
Under the above context setting, we aim to minimize AoII (instead of AoI). S_k and S'_k are defined as the time-slot of the k-th status change of the local and remote server, respectively. In Fig. 3, we can observe that at time-slot S_5, the remote server differs from the source status, resulting in the value of AoII increasing to 1. At time-slot S'_5, the remote server matches the source status, so the value of AoII drops to zero. At time-slot S_6, the remote server becomes incorrect again, causing the AoII value to increase over time. Without loss of generality, we consider a step-wise AoII model, where at any time-slot S_k, $k \in \{0, 1, 2, ...\}$, if the remote server is incorrect, the value of AoII increases by 1. Furthermore, the value of AoII resets to zero immediately when the remote server becomes correct, i.e., $x_l(t) = x_r(t)$.

To simplify the model, we make the following three assumptions: 1) at the commencement of a time-slot, the latest update packet is sent by the sensor, 2) one time-slot is taken up by each transmission of a update [29], and the transmission time of ACK/NACK can be disregarded. 3) at the end of a time-slot, the update packet is received by the remote server. Upon successful reception of the status packet by the remote server, the source status is estimated based on the latest update packet. The probability of successful reception of the update packet is denoted by ε_s. Therefore, the probability of failed transmission is calculated as $\varepsilon_f = 1 - \varepsilon_s$. Assuming that the remote server is correct at the beginning of S'_k, it may be because the source status packet is transmitted correctly, or the source status just changes at this moment, which is consistent with the status of the remote server. Therefore, the remote server incorrectly estimates the status of the source between the start of slot S_k and the start of slot S'_k, and the AoII denoted by $\Delta(t)$ increases over time. Besides, the remote server achieves accurate estimation at the beginning of slot S'_k, and the AoII drops back down to zero.

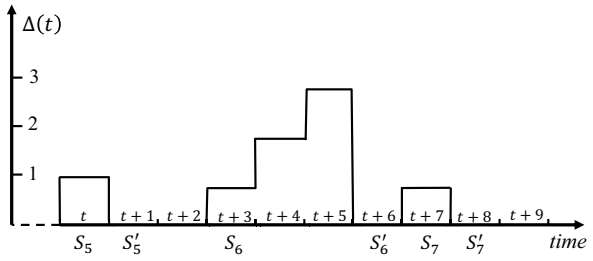
[1]The purpose of introducing a local scheduler is to simultaneously obtain the status of both the sensors side and the remote server side, in order to characterize the AoII (see Equation (1)).



(a) The local observation $x_l(t)$. The pink and blue shades indicate the events of receiving an ACK and a NACK, respectively.



(b) The remote server $x_r(t)$. The pink and blue dashed lines indicate the events of sending an ACK and a NACK, respectively.



(c) The evolution of AoII.

Fig. 3. An example of AoII.

Assuming the ACK/NACK transmission time is τ, $\tau \ll t$, at the end of the slot τ, a recent ACK/NACK feedback is received by the local scheduler receives. At time-slot t, the local scheduler knows if the remote server is receiving the packet accurately at the end of slot τ_t, i.e. if ACK, $x_r(t) = x_l(\tau_t)$. the local scheduler can determine whether the remote server is providing correct estimates by comparing the value of $x_l(t)$ and $x_l(\tau_t)$. At the beginning of the time-slot, we define the indices of the latest accurate remote server as $m_r(t) = \max\{k|S'_k \leq t\}$, and let $m_l(t) = \max\{k|S_k \leq t\}$ to denote the indices of the latest incorrect remote server. For example, if time-slot t is between S_6 and S'_6, $m_r(t) = 5$, $m_l(t) = 6$, indicating that the remote server does not estimate the status of source correctly; if time-slot t is between S'_6 and S_7, $m_r(t) = 6$, $m_l(t) = 6$ and the remote server's estimation is correct. Consequently, the AoII at time-slot t can be formulated as[2]:

$$\Delta(t) = (t - S_{m_l(t)} + 1)\mathbb{1}_{\{m_r(t) \neq m_l(t)\}}, \tag{1}$$

[2]The timeliness of AoI is only reflected in the time between the sensor sending the packet and the remote server receiving it. AoII adds the consideration of whether the status of the sensor and remote server are consistent.

where $\mathbb{1}$ denotes the indicator function. We can use the variable a_t to indicate whether the sensor is selected to send status updates ($a_t = 1$) or not ($a_t = 0$). Due to the nature of the wireless channel, a single transmission attempt may require more than one time-slot to be successful.

The AoII, source status, and remote server status make up the system state, which is represented by the vector $\boldsymbol{s}_t = (\Delta(t), x_l(t), x_r(t))$. At each time-slot, the local scheduler can decide whether to transmit by setting the variable a_t to either 1 or 0. The system state transition probability is defined as $\mathbf{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_t) = \mathbf{Pr}(\boldsymbol{s}_{t+1} = \boldsymbol{s}'|\boldsymbol{s}_t = \boldsymbol{s}, a_t = 0 \text{ or } 1)$, which indicates that taking an action in state $\boldsymbol{s}$ at time t will result in a new state $\boldsymbol{s}'$ at time $t+1$. We use x'_l to denote the source status at slot $t+1$ and divide the transition probability into three cases as follows by omitting (t) in $\Delta(t)$, $x_l(t)$ and $x_r(t)$:

1) Case 1: At slot t, the sensor is not scheduled to send an update packet, $a_t = 0$. At slot $t+1$, the remote server remains unchanged.

- If $x'_l = x_r$, the AoII at slot $t+1$ will be 0 and $\mathbf{Pr}((0, x_r, x_r)|(\Delta, x_l, x_r), 0) = p_{x_l, x_r}$.
- If $x'_l \neq x_r$, the AoII at slot $t+1$ will be $\Delta+1$ and $\mathbf{Pr}((\Delta+1, x'_l, x_r)|(\Delta, x_l, x_r), 0) = p_{x_l, x'_l}$.

2) Case 2: At slot t, the sensor is scheduled to send an update packet, $a_t = 1$. However, there is a probability ε_f that the destination will not receive the packet successfully.

- If $x'_l = x_r$, the AoII at slot $t+1$ will be 0 and $\mathbf{Pr}((0, x_r, x_r)|(\Delta, x_l, x_r), 1) = \varepsilon_f p_{x_l, x_r}$.
- If $x'_l \neq x_r$, the AoII at slot $t+1$ will be $\Delta+1$ and $\mathbf{Pr}((\Delta+1, x'_l, x_r)|(\Delta, x_l, x_r), 1) = \varepsilon_f p_{x_l, x'_l}$.

3) Case 3: At slot t, the sensor is scheduled to send an update packet, $a_t = 1$. And there is a probability ε_s that the destination will receive the packet successfully.

- If $x'_l = x_l$, the AoII at slot $t+1$ will be 0 and $\mathbf{Pr}((0, x_l, x_l)|(\Delta, x_l, x_r), 1) = \varepsilon_s p_{x_l, x_l}$.
- If $x'_l \neq x_l$, the AoII at slot $t+1$ will be $\Delta+1$ and $\mathbf{Pr}((\Delta+1, x'_l, x_l)|(\Delta, x_l, x_r), 1) = \varepsilon_s p_{x_l, x'_l}$.

B. Multi-sensor Scenario and Problem Formulation

This subsection presents a generalized scenario of the system that involves N sensors. Let $\boldsymbol{c} = \rho[c_1, c_2, ..., c_N]^T$ be the weighted transmission cost vector, where ρ denotes the weight and $c_n \geq 0$ is the transmission cost of sensor n. The scheduling decision is denoted by $\boldsymbol{a}_t = [\boldsymbol{a}_t^{(1)}, \boldsymbol{a}_t^{(2)}, ..., \boldsymbol{a}_t^{(N)}]^T$, where $\boldsymbol{a}_t^{(n)} \in \{0, 1\}$ decides whether sensor n sends a status packet at slot t. In addition, to be more applicable to modern industrial scenarios, we assume that each sensor should occupy a certain amount of bandwidth to transmit data. We denote the bandwidth requirement vector as $\boldsymbol{b} = [b_1, b_2, ..., b_N]^T$, and the total available bandwidth of the system to transmit the update packet in each slot is M. In this work, we desire to obtain an optimal schedule policy to minimize the system-wide cost [30], i.e., long-term weighted sum of AoII and

transmission cost under the bandwidth constraint. The problem can be mathematically expressed as:

$$\boldsymbol{Q1}: \min_{\pi\in\Pi}\lim_{T\to\infty}\frac{1}{T}\mathbb{E}_{\pi}\left(\sum_{t=1}^{T}\left(\|\boldsymbol{\Delta}(t)\|_1+\boldsymbol{c}^T\boldsymbol{a}_t\right)\right), \quad (2)$$

$$s.t. \quad \boldsymbol{a}_t\cdot\boldsymbol{b}\le M,\ \forall t, \quad (3)$$

$$\boldsymbol{a}_t^{(n)}\in\{0,1\},\ \forall t,\ \forall n, \quad (4)$$

where Π is the class of non-anticipated policies, i.e. the scheduling decisions are made based on the historical and current data, $\mathbb{E}_{\pi}[\cdot]$ represents the expectation with respect to policy π and $\|\cdot\|_1$ denotes the ℓ^1 norm of a vector. The vector $\boldsymbol{\Delta}(t)=[\Delta_1(t),\Delta_2(t),...,\Delta_N(t)]^T\in\mathbb{N}^N$ denotes the AoII of all sensors at the beginning of time-slot t. To simplify the problem, we assume that all sensors and the receiver are synchronized initially, i.e, $\boldsymbol{\Delta}(0)=\boldsymbol{0}$.

Due to the bandwidth constraint in (3), there are many possible scheduling decisions in each time-slot, and it is not possible to obtained the optimal scheduling policy through exhausted search method. Additionally, traditional techniques, such as dynamic programming [31], have difficulty in addressing the scheduling problem due to the status-heterogeneous transitions for each source [19]. In recent years, reinforcement learning has become increasingly popular, which is also regarded as a potential solution approach to this problem. However, due to the communication resource constraints, it is challenging to schedule tasks efficiently in this environment even with the existing reinforcement learning algorithms, such as A3C and DDPG. In sections III, we improve the existing reinforcement learning algorithms and propose two novel approaches to solve problem $\boldsymbol{Q1}$ separately.

III. Proposed scheduling algorithm based on RL

In deep reinforcement learning, neural networks are used to approximate the value function without a fixed form. Deep Q-networks (DQN), double DQN, asynchronous advantage actor-critic (A3C), and deep deterministic policy gradient (DDPG) are advanced algorithms that may effectively tackle complicated sequential decision-making problems [32]. Nevertheless, they may not be directly applicable to our specific problem, which involves a large sensor network with bandwidth constraints, due to the extensive action space. Therefore, we propose an Order-Preserving Quantization-Based Constrained Reinforcement Learning Algorithm with Historical Adjustment (OPQ-RL_HA) framework to address these challenges.

Different from the traditional reinforcement learning algorithm, the core part of the Constrained Reinforcement Learning Algorithm with Historical Adjustment (RL_HA) includes the following two parts: 1) optimization based on historical decisions, and 2) scheduling decisions based on greedy algorithm. The implementation process of RL_HA is as follows: First, an initial scheduling decision is obtained using a reinforcement learning algorithm. Then, the initial decision is adjusted based on the historical scheduling conditions of each sensor. Next, a scheduling decision that satisfies the bandwidth constraints is obtained using a greedy algorithm. Finally, the model is updated according to the scheduling decisions. In addition, to address the trade-off between exploration and exploitation, we introduce an Order-Preserving Quantization (OPQ) [33] method to generate multiple actions.

This section introduces the CMDP established based on $\boldsymbol{Q1}$. It then presents the main principles of the reinforcement learning algorithm adopted in this study. Following that, a historical adjustment optimization method is proposed. Finally, the section describes the method for optimizing the model using the order-preserving quantization technique.

A. Constrained Markov Decision Process

We reformulate problem $\boldsymbol{Q1}$ as a constrained Markov decision process, i.e., $\langle\mathcal{S},\mathcal{A},\mathcal{P},r,c\rangle$. Where $\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, r and c are network state space, action space, state transition probability, reward function and cost function, respectively. At time-slot t, the state of the environment that the agent senses is denoted as $\boldsymbol{S}_t$. The agent selects an initial action as the output using the decision policy π. Next, by utilizing the historical adjustment and the OPQ method, the final output action $\boldsymbol{a}_t$ is chosen to satisfy the constrained cost. In the next time-slot, the change of the environment to a new state $\boldsymbol{S}_{t+1}$, and the agent receives a reward r_t based on $\boldsymbol{a}_t$. The interaction between the agent and environment continues until one of the following conditions is met: 1) the environment enters a terminal state; 2) the slot t surpasses the predefined upper bound.

1) State Space: For multi-sensor systems, in time-slot t, the state of the n-th sensor (stored in the local scheduler) includes the status of AoII, source and remote server, i.e., $\boldsymbol{s}_t^{(n)}=\{\|\boldsymbol{\Delta}(t)\|_1,\boldsymbol{x}_l^{(n)}(t),\boldsymbol{x}_r^{(n)}(t)\}$, $\boldsymbol{s}_t^{(n)}\in\mathcal{S}$. Assuming that the system has N sensors, the state of the system is $\boldsymbol{S}_t=[\boldsymbol{s}_t^{(1)};\boldsymbol{s}_t^{(2)};...;\boldsymbol{s}_t^{(N)}]$.

2) Action Space: At time-slot t, according to $\boldsymbol{s}_t^{(n)}$ and historical scheduling situation, the scheduling action $\boldsymbol{a}_t^{(n)}$ of the n-th sensor is obtained, $\boldsymbol{a}_t^{(n)}\in\mathcal{A}$. For N sensors, $\boldsymbol{a}_t=[\boldsymbol{a}_t^{(1)},\boldsymbol{a}_t^{(2)},...,\boldsymbol{a}_t^{(N)}]^T$.

3) State Transition Probability: As previously discussed in II-A, the state transition probability of the n-th sensor is $\mathbf{P}^{(n)}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,a_t)\in\mathcal{P}$. $\mathbf{P}=[\boldsymbol{P}^{(1)};\boldsymbol{P}^{(2)};...;\boldsymbol{P}^{(N)}]$ is defined as the state transition probability of a system with N sensors.

4) Reward Function: When $\boldsymbol{S}_t$ and $\boldsymbol{a}_t$ are given, the one-step reward can be expressed as follows:

$$r_t=-\left(\|\boldsymbol{\Delta}(t)\|_1+\boldsymbol{c}^T\boldsymbol{a}_t\right), \quad (5)$$

where the negative sign is adopted, because that $\boldsymbol{Q1}$ is a minimization problem. Besides, the formula for calculating the cumulated reward of time-slot t can be expressed as $R_t=\sum_{k=0}^{\infty}\gamma^k r_{t+k}$, where $\gamma\in(0,1]$ (the discount factor) denotes the weight of future rewards.

5) Cost Function: Immediate cost $c_t=\boldsymbol{a}_t\cdot\boldsymbol{b}$ is incurred after the execution of action $\boldsymbol{a}_t$ at time-slot t. According to (3), we can get the cost function that satisfies the constraints as $c_t\le M$.

In large-scale industrial scenarios, it is very difficult to obtain scheduling actions that satisfy constraints by conventional methods, such as dynamic programming, genetic algorithms, etc [34]. Therefore, we use reinforcement learning algorithms to find the optimal scheduling policy that maximizes the

expectation of cumulative rewards. In addition, historical adjustment is used to adjust and optimize the scheduling decisions, and the OPQ method is introduced to improve the generalization performance of the model.

B. Reinforcement Learning Algorithm

In this section, we briefly introduce A3C and DDPG algorithms, which perform well in task scheduling problems [35], [36]. Both of them employ the actor-critic framework, where the actor is responsible for generating the action and the critic evaluates the value of the current policy. The difference is that A3C learns a randomness strategy, improving the explorability of the model by training multiple agents in parallel with different copies of the environment. DDPG learns a deterministic strategy and realizes improved policy exploration by adding a noise component. It employs a training method that involves randomly sampling historical experiences to enhance model stability. Therefore, within the same application scenario, the two algorithms may yield different performances. Furthermore, by using deep neural networks (DNN), actor-critic methods are able to learn and adapt to complex, high-dimensional environments more effectively, thereby improving the generalization performance of the model.

1) A3C: A3C enables asynchronous multiple agents to interact with their environment in parallel and implement different exploration policies. Each agent evaluates and optimizes its policy according to the value function. In state $\boldsymbol{S}_t$, given the policy π, we define the state and state-action value function as $V(\boldsymbol{S}_t;\boldsymbol{\theta}_v)$ and $Q(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta})$, respectively. where $\boldsymbol{\theta_v}$ and $\boldsymbol{\theta}$ represent the parameters of critic and actor respectively. $V(\boldsymbol{S}_t;\boldsymbol{\theta}_v)$ represents the average return, and $Q(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta})$ is the expected return after taking action $\boldsymbol{a}_t$. To reduce the variance in the learning process and enhance strategy effectiveness, the advantage function $A(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta},\boldsymbol{\theta}_v)$ replaces $Q(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta})$ [37]. Assuming that the critic network is updated after every d actions, and denoting the beginning steps of each update by t_b, the expression for $A(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta},\boldsymbol{\theta}_v)$ is as follows:

$$\begin{aligned} A(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta},\boldsymbol{\theta}_v) =& Q(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta}) - V(\boldsymbol{S}_t;\boldsymbol{\theta}_v) \\ =& \sum_{i=0}^{t_b+d-t-1} \gamma_a^i r_{t+i} + \gamma_a^{t_b+d-t} V(\boldsymbol{S}_{t_b+d};\boldsymbol{\theta}_v) \\ & - V(\boldsymbol{S}_t;\boldsymbol{\theta}_v), \end{aligned} \tag{6}$$

where $\gamma_a \in (0,1]$ is the discount factor of the actor. t varies from t_b to t_b+d-1 for each parameter update.

According to (6), given the policy π, the loss function and parameter update formula for the actor are as follows:

$$\begin{aligned} R^{(a)}(\boldsymbol{\theta}) = \sum_{t=t_b}^{t_b+d-1} \Big(& A(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta},\boldsymbol{\theta}_v)\log\pi(\boldsymbol{a}_t|\boldsymbol{S}_t;\boldsymbol{\theta}) \\ & + \beta H(\pi(\boldsymbol{a}_t|\boldsymbol{S}_t;\boldsymbol{\theta})) \Big), \end{aligned} \tag{7}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta'}} R^{(a)}(\boldsymbol{\theta'}), \tag{8}$$

where β is the weight of the entropy regularization term, $H(\pi(\boldsymbol{a}_t|\boldsymbol{S}_t;\boldsymbol{\theta}))$ denotes the entropy of the policy π. The inclusion of entropy in the policy aids in preventing suboptimal solutions by enhancing the model's exploration capabilities. Additionally, α represents the the learning rate of the actor, and $\boldsymbol{\theta}'$ denotes the parameters of local actors[3].

For the critic, its loss function and parameters are updated as follows:

$$L^{(a)}(\boldsymbol{\theta}_v) = \sum_{t=t_b}^{t_b+d-1} A^2(\boldsymbol{S}_t,\boldsymbol{a}_t;\boldsymbol{\theta},\boldsymbol{\theta}_v), \tag{9}$$

$$\boldsymbol{\theta}_v \leftarrow \boldsymbol{\theta}_v + \alpha_v \nabla_{\boldsymbol{\theta}'_v} L^{(a)}(\boldsymbol{\theta}'_v), \tag{10}$$

where α_v represents the learning rate of the critic, and $\boldsymbol{\theta}'_v$ denotes the parameters of local critics

2) DDPG: DDPG is also based on the Actor-Critic framework and comprises both Actor and Critic networks. In addition, to improve the stability of the learning process, each network has its corresponding target network. Unlike A3C, DDPG is a single-threaded algorithm, and obtains deterministic actions through the actor. At slot t, given the policy π, to enhance the ability to explore, an exploration policy $\widetilde{\pi}$ is constructed by adding noise sampled from a noise process $\mathcal{N}$ to the actor policy π. And we use an Ornstein-Uhlenbeck process to generate temporally correlated exploration [38]. Assuming that under the policy $\widetilde{\pi}$, the obtained decision action is $\widetilde{\boldsymbol{a}}_t$.

Critic evaluates the value of state-action pairs in DDPG, i.e. $Q(\boldsymbol{S}_t,\widetilde{\boldsymbol{a}}_t;\boldsymbol{\theta}_q)$, where $\boldsymbol{\theta}_q$ is the parameter of critic network. To avoid the negative impact of data correlation on training accuracy, after the interaction between the agent and the environment, the transition information $(\boldsymbol{S}_t,\widetilde{\boldsymbol{a}}_t,r_t,\boldsymbol{S}_{t+1})$ is stored in the experience replay pool $\boldsymbol{D}$. During each training process, a batch of size L of the transition information is sampled from $\boldsymbol{D}$ to train the network.

In the DDPG algorithm, the actor's goal is to find actions that maximize future returns. Therefore, the loss function and parameter update for actor are as follows:

$$R^{(d)}(\boldsymbol{\theta}_a) = \frac{1}{L}\sum_{i=1}^{L} Q(\boldsymbol{S}_{r(i)},\boldsymbol{a}_{r(i)};\boldsymbol{\theta}_q), \tag{11}$$

$$\boldsymbol{\theta}_a \leftarrow \boldsymbol{\theta}_a + \alpha_a \nabla_{\boldsymbol{\theta}_a} R^{(d)}(\boldsymbol{\theta}_a), \tag{12}$$

where $\boldsymbol{\theta}_a$ are the parameters of the actor, $r(i)$ is the time slot corresponding to the i-batch of transmission information.

To provide a stable learning goal, the target value is defined as $y_t = r_t + \gamma_d Q'(\boldsymbol{S}_{t+1},\widetilde{\boldsymbol{a}}'_{t+1};\boldsymbol{\theta}'_q)$. Where γ_d is the discount factor of the critic, $\widetilde{\boldsymbol{a}}'_{r(i)}$ is the decision action obtained under policy $\widetilde{\pi}'(\boldsymbol{S}_{r(i)};\boldsymbol{\theta}'_a)$. $Q'(\cdot)$ and $\widetilde{\pi}'(\cdot)$ are the target networks with the weights $\boldsymbol{\theta}'_q$ and $\boldsymbol{\theta}'_a$, respectively[4]. The loss function and parameter updates in the critic are as follows:

$$L^{(d)}(\boldsymbol{\theta}_q) = \frac{1}{L}\sum_{i=1}^{L}(y_i - Q(\boldsymbol{S}_{r(i)},\boldsymbol{a}_{r(i)};\boldsymbol{\theta}_q))^2, \tag{13}$$

$$\boldsymbol{\theta}_q \leftarrow \boldsymbol{\theta}_q + \alpha_q \nabla_{\boldsymbol{\theta}_q} L^{(d)}(\boldsymbol{\theta}_q). \tag{14}$$

[3]The A3C algorithm updates the global network parameters asynchronously in real time based on the interaction information provided by local agents. Then, synchronize the parameters to the local agent.

[4]The target network, $Q'(\cdot)$ and $\widetilde{\pi}'(\cdot)$, have the same structures as the original critic network $Q(\cdot)$ and actor network $\widetilde{\pi}(\cdot)$.

The parameters of the target network do not completely copy the parameters of the original network in each learning, but gradually track the parameters of the original network through slow updates. DDPG employs a soft update to modify the target network's parameters:

$$\begin{aligned} \boldsymbol{\theta}'_a &\leftarrow \tau\boldsymbol{\theta}_a + (1-\tau)\boldsymbol{\theta}'_a, \\ \boldsymbol{\theta}'_q &\leftarrow \tau\boldsymbol{\theta}_q + (1-\tau)\boldsymbol{\theta}'_q. \end{aligned} \tag{15}$$

where the weight τ presents the rate of change and $\tau \ll 1$ to ensure smooth and gradual updates to the target network's parameters.

C. Constrained Reinforcement Learning with Historical Adjustment

In section III-B, the actor's output is not directly suitable as the final decision action. While A3C produces action probabilities, DDPG generates continuous actions. The following section introduces the application of historical adjustment in reinforcement learning to derive discrete action outputs.

To avoid overuse of certain sensors and ensure the balance of data collection, we assume that the first l scheduling count is $\boldsymbol{d}(l) = [d(l)^{(1)}, d(l)^{(2)}, ..., d(l)^{(N)}]^T$, where the scheduling count $d(l)^{(n)} \in \mathbb{N}$ for each sensor. At time-slot t, we believe that the sensors with smaller $d(l)$ are more likely to be scheduled in the next moment. To characterize this tendency, we apply an exponential function $0.5^{d(l)}$ such that $d(l)$ maps to $(0, 1]$. This exponentially decaying mapping not only reflects the preference for sensors that have not been scheduled for a long time, but also provides a smooth way to adjust the scheduling priority of sensors. Besides, the choice of l should be moderate, allowing it to take into account enough historical information to ensure long-term equilibrium, while being flexible enough to adapt to recent environmental changes.

However, simply selecting the output action through the method described above does not guarantee that the action will always satisfy the bandwidth constraint. To address this issue, we adopt a greedy algorithm. Specifically, from the set of potential decision actions selected by the threshold-based method, this algorithm aims to preserve as much as possible the characteristics of the original decision strategy obtained through RL. This approach enables the model to more effectively learn the features of the constraints, thereby improving overall performance.

1) A3C_HA: Assuming the probability of output under policy π is $\hat{\boldsymbol{a}}_t$, and we can define the optimization probability obtained by using the historical adjustment as a function below:

$$\begin{aligned} &\hat{\pi}(\cdot;\boldsymbol{\theta}):\ \mathcal{S} \rightarrow \mathbb{R}^N, \\ &\hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta}) = \lambda\hat{\boldsymbol{a}}_t + (1-\lambda)0.5^{\boldsymbol{d}(l)}, \end{aligned} \tag{16}$$

where $\hat{\boldsymbol{a}}_t = \pi(\boldsymbol{S_t};\boldsymbol{\theta})$, $\pi(\cdot;\boldsymbol{\theta})$ is a function with parameters $\boldsymbol{\theta}$, mapping from the state space $\mathcal{S}$ to the N-dimension real number space $\mathbb{R}^N$. λ balances the actor's output and the exponential mapping of the first l scheduling results. If $\lambda = 0$, the optimization probability is only related to the historical scheduling decision information. At this time, A3C does not work, and it is difficult to meet the optimization goals. By contrast, if $\lambda = 1$, the optimization probability is only related to the output of A3C, which is likely to lead to overuse of certain sensors. Therefore, in order to let both A3C and the historical adjustment play approximate weight in the optimization problem, we set $\lambda = 0.5$.

After sampling the local source, sensors can then be scheduled to send status packets based on optimization probability. At time-slot t, the scheduling action can be denoted as $\bar{\boldsymbol{a}}_t$:

$$g:\ \hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta}) \rightarrow \{\bar{\boldsymbol{a}}_t | \bar{\boldsymbol{a}}_t \in \{0,1\}^N\}, \tag{17}$$

where g uses the threshold method to map the set $\mathbb{R}^N$ to the set $\{0,1\}^N$. Nevertheless, since the bandwidth constraint (3), the action $\bar{\boldsymbol{a}}_t$ may be not valid.

We denote the set Ψ as $\hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta}) \odot \bar{\boldsymbol{a}}_t$ (the probability that the sensor is scheduled), where $\odot$ stands for the Hadamard product. We adopt a heuristic method to send status packets while satisfying the bandwidth constraint: if the total bandwidth of sensors to be scheduled at time-slot t is less than M, the actual scheduling action is just $\bar{\boldsymbol{a}}_t$; otherwise, set the non-zero minimum element of Ψ to 0 until the bandwidth constraint is satisfied, and the corresponding scheduling action at this time is denoted by $\check{\boldsymbol{a}}_t = [\check{\boldsymbol{a}}_t^{(1)}, ..., \check{\boldsymbol{a}}_t^{(N)}]$. Besides, $\check{\boldsymbol{a}}_t^{(n)} = 1$ if $n \in \Psi$, otherwise $\check{\boldsymbol{a}}_t^{(n)} = 0$. Hence, the actual scheduling action can be expressed as:

$$\begin{aligned} &h: \{0,1\}^N \rightarrow \dashv, \\ &\boldsymbol{a}_t = h(\bar{\boldsymbol{a}}_t) = \begin{cases} \bar{\boldsymbol{a}}_t, & \bar{\boldsymbol{a}}_t \cdot \boldsymbol{b} \leq M, \\ \check{\boldsymbol{a}}_t, & \bar{\boldsymbol{a}}_t \cdot \boldsymbol{b} > M. \end{cases} \end{aligned} \tag{18}$$

For instance, the Optimization probability vector is $\hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta}) = [0.5, 0.6, 0.2, 0.7, 0.3]$, the bandwidth vector is $\boldsymbol{b} = [9.5, 7.0, 8.4, 8.5, 5.7]$ and bandwidth constraints $M = 16$. After selecting according to $\hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta})$, we can get action vector $\bar{\boldsymbol{a}}_t = [1, 1, 0, 1, 0]$, which violates the bandwidth constraint. By calculating $\hat{\pi}(\boldsymbol{S_t};\boldsymbol{\theta}) \odot \bar{\boldsymbol{a}}_t$, we can obtain $\Psi = \{0.5, 0.6, 0, 0.7, 0\}$. Through heuristic algorithm, we can get $\boldsymbol{a}_t = \check{\boldsymbol{a}}_t = [0, 1, 0, 1, 0]$. In consequence, only the second and the fourth sensors can update their status information.

At this point, three steps are taken in turn to return a scheduling action and the functions can be composited as:

$$F_{\boldsymbol{\theta}} : h \circ g \circ \hat{\pi}(\cdot;\boldsymbol{\theta}), \tag{19}$$

where $\circ$ denotes the composition operation.

2) DDPG_HA: The actions generated by the DDPG are continuous, and $\widetilde{\boldsymbol{a}}_t \in \{0,1\}$, which can be regarded as the priority level to be scheduled. According to (16)-(18), first, the scheduling priority level is optimized using the history adjustment. Then, the discretized output is obtained by using the threshold method, such as dividing $\widetilde{\boldsymbol{a}}_t$ into two intervals representing 0 and 1 respectively. Finally, the final output is obtained by using the heuristic algorithm.

D. Order-Preserving Quantization Based Constrained Reinforcement Learning with Historical Adjustment

In section III-C, it may be challenging for deep learning methods to encompass all valid actions. As a supplementary mechanism, OPQ can bolster the model's decision-making

capabilities by broadening the spectrum of available actions through similarity search. Consequently, we introduced the OPQ method in RL_HA.

1) OPQ-A3C_HA: In A3C_HA, the exploration space of RL_HA mainly relies on the output of the actor. According to equation (16), each entry of the optimization probability is located in the interval of $[0, 1]$, and we first quantize $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ to K scheduling actions with binary entry to expand its search space, where K is an integer. The quantization function can be expressed as:

$$g_k: \ \hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta}) \rightarrow \{\bar{\boldsymbol{a}}_{t,k} | \bar{\boldsymbol{a}}_{t,k} \in \{0, 1\}^N\}, \ k = 1, ..., K. \quad (20)$$

When choosing a large K, the solution quality becomes better with higher computational complexity. In general, the value of K is no more than $N + 1$. Furthermore, K possible scheduling actions can be generated according to Property 1 of OPQ method [33].

Property 1. *The ordering of the entries of quantized actions are the same as that of $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$, which can be represented as:*

$$\bar{\boldsymbol{a}}_{t,k}^{(i)} \geq \bar{\boldsymbol{a}}_{t,k}^{(j)}, \ \textit{when} \ \hat{\boldsymbol{a}}_t^{(i)} \geq \hat{\boldsymbol{a}}_t^{(j)}, \ \forall i, j \in \{1, ...N\}, \quad (21)$$

where $\hat{\boldsymbol{a}}_t^{(i)}$ and $\bar{\boldsymbol{a}}_{t,k}^{(i)}$ are the i-th entry of $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ and the i-th entry of the k-th quantized action, respectively.

Next, we first quantize action $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ to K (we adopt $K = N + 1$) binary actions without bandwidth constraint:

1) Given the threshold 0.5, the entry of $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ which is less than or equal to 0.5 is quantized to 0; otherwise, the entry is set to 1. That is, $\forall i \in \{1, ..., N\}$,

$$\bar{\boldsymbol{a}}_{t,1}^{(i)} = \begin{cases} 1, \ \hat{\boldsymbol{a}}_t^{(i)} \geq 0.5, \\ 0, \ \hat{\boldsymbol{a}}_t^{(i)} < 0.5, \end{cases} \quad (22)$$

2) Given the threshold $\hat{\boldsymbol{a}}_t^{(k-1)}$ (where $k = 2, .., N + 1$), if the entry of $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ is less than the threshold, or the entry equals the threshold and the threshold is greater than 0.5, it is set to 0; otherwise, the entry is approximated to 1. That is, $\forall i \in \{1, ..., N\}$,

$$\bar{\boldsymbol{a}}_{t,k}^{(i)} = \begin{cases} 1, \ \hat{\boldsymbol{a}}_t^{(i)} > \hat{\boldsymbol{a}}_t^{(k-1)}, \\ 1, \ \hat{\boldsymbol{a}}_t^{(i)} = \hat{\boldsymbol{a}}_t^{(k-1)} \text{and } \hat{\boldsymbol{a}}_t^{(k-1)} \leq 0.5, \\ 0, \ \hat{\boldsymbol{a}}_t^{(i)} = \hat{\boldsymbol{a}}_t^{(k-1)} \text{and } \hat{\boldsymbol{a}}_t^{(k-1)} > 0.5, \\ 0, \ \hat{\boldsymbol{a}}_t^{(i)} < \hat{\boldsymbol{a}}_t^{(k-1)}. \end{cases} \quad (23)$$

We now give an example to explain the OPQ method. Assume that action $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta}) = [0.3, 0.3, 0.2, 0.7, 0.8]^T$, then the $K = N + 1 = 6$ quantized actions are $\bar{\boldsymbol{a}}_{t,1} = [0, 0, 0, 1, 1]^T$, $\bar{\boldsymbol{a}}_{t,2} = [1, 1, 0, 1, 1]^T$, $\bar{\boldsymbol{a}}_{t,3} = [1, 1, 0, 1, 1]^T$, $\bar{\boldsymbol{a}}_{t,4} = [1, 1, 1, 1, 1]^T$, $\bar{\boldsymbol{a}}_{t,5} = [0, 0, 0, 0, 1]^T$, $\bar{\boldsymbol{a}}_{t,6} = [0, 0, 0, 0, 0]^T$, respectively.

Since sensors with a total bandwidth greater than M cannot be scheduled to send their update information at the same time, the actions generated by the OPQ method may not meet constraint (3), i.e., $\bar{\boldsymbol{a}}_{t,k} \notin \mathcal{A}$. Next, we restrict the scheduling actions from the OPQ method to meet the bandwidth constraint.

Property 1 indicates that the ordering of entries of the quantized actions remains unaltered after quantization. If the actions contravene constraint (3), i.e., $\bar{\boldsymbol{a}}_{t,k} \cdot \boldsymbol{b} > M$, those sensors meeting the bandwidth constraint in action $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ will be scheduled to send update packets and the other sensors will keep silence. Here, the action selection in $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ that satisfies the bandwidth constraint is consistent with the heuristic in section III-C1. Since the OPQ method may not meet the bandwidth constraint, we denote the indexed collection that meets the bandwidth constraint in action $\hat{\pi}(\boldsymbol{S_t}; \boldsymbol{\theta})$ as Ω. and $\boldsymbol{a}_{t,0}$ as the corresponding scheduling action.

As a result, the actions that schedule more than M bandwidth are restricted to meet the bandwidth constraint, and the regenerating function of scheduling actions can be formulated below:

$$\begin{aligned} &h_k: \ \{0, 1\}^N \rightarrow \mathcal{A}, \\ &\boldsymbol{a}_{t,k} = h_k(\bar{\boldsymbol{a}}_{t,k}) = \begin{cases} \bar{\boldsymbol{a}}_{t,k}, & \bar{\boldsymbol{a}}_{t,k} \cdot \boldsymbol{b} \leq M, \\ \boldsymbol{a}_{t,0}, & \bar{\boldsymbol{a}}_{t,k} \cdot \boldsymbol{b} > M, \end{cases} \end{aligned} \quad (24)$$

where $k \in \{1, 2, \cdots, K\}$, and h_k's are the mapping functions from set $\{0, 1\}^N$ to the feasible region $\dashv$ of problem $\boldsymbol{Q1}$. The scheduling actions transform to $\boldsymbol{a}_{t,0}$, if they do not meet the constraint. In addition, $\boldsymbol{a}_{t,0} = [\boldsymbol{a}_{t,0}^{(1)}, ..., \boldsymbol{a}_{t,0}^{(N)}]^T$ where $\boldsymbol{a}_{t,0}^{(n)} = 1$ if $n \in \Omega$, otherwise $\boldsymbol{a}_{t,0}^{(n)} = 0$.

For example, the action probability is $\pi'(\boldsymbol{S_t}; \boldsymbol{\theta}) = [0.3, 0.3, 0.2, 0.7, 0.8]^T$, the bandwidth is $\boldsymbol{b} = [9.5, 7.0, 8.4, 8.5, 5.7]^T$ and bandwidth constraints $M = 16$. Through the heuristic algorithm, $\boldsymbol{a}_{t,0} = [0, 0, 0, 1, 1]^T$, the possible quantized actions that meet the bandwidth constraint are $\boldsymbol{a}_{t,1} = [0, 0, 0, 1, 1]^T$, $\boldsymbol{a}_{t,2} = [0, 0, 0, 1, 1]^T$, $\boldsymbol{a}_{t,3} = [0, 0, 0, 1, 1]^T$, $\boldsymbol{a}_{t,4} = [0, 0, 0, 1, 1]^T$, $\boldsymbol{a}_{t,5} = [0, 0, 0, 0, 1]^T$, $\boldsymbol{a}_{t,6} = [0, 0, 0, 0, 0]^T$. Moreover, there are 3 different scheduling decisions here, i.e., $\boldsymbol{a}_{t,1}$, $\boldsymbol{a}_{t,5}$, $\boldsymbol{a}_{t,6}$.

At each time-slot, the agent adopts only one action to interact with the environment. Hence, the further step is required to select the optimal action from $\boldsymbol{a}_{t,1}$, ..., $\boldsymbol{a}_{t,K}$ as:

$$\begin{aligned} &v_t(\cdot; \boldsymbol{\theta}, \boldsymbol{\theta}_v): \{\boldsymbol{a}_{t,k}\} \rightarrow \boldsymbol{a}_t^*, \\ &\boldsymbol{a}_t^* = v_t(\boldsymbol{a}_{t,1}, ..., \boldsymbol{a}_{t,K}; \boldsymbol{\theta}, \boldsymbol{\theta}_v) \\ &\quad = \arg\max_{\boldsymbol{a} \in \{\boldsymbol{a}_{t,k}\}} A(\boldsymbol{S}_t, \boldsymbol{a}; \boldsymbol{\theta}, \boldsymbol{\theta}_v), \end{aligned} \quad (25)$$

where v_t is a function that maps K scheduling actions to action $\boldsymbol{a}_t^*$ with highest A which is estimated by the critic network.

So far, the four steps are taken in turn to return a scheduling action and we call the composite function of g_k and h_k as constrained OPQ (COPQ) method. Furthermore, the actual scheduling function can be composited as:

$$F_{\boldsymbol{\theta}, \boldsymbol{\theta}_v}: \ v_t(\cdot; \boldsymbol{\theta}, \boldsymbol{\theta}_v) \circ h_k \circ g_k \circ \hat{\pi}(\cdot; \boldsymbol{\theta}). \quad (26)$$

2) OPQ-DDPG_HA: The output of actor is discretized by the threshold method in DDPG_HA. However, in complex spaces, the mapping error from continuous space to discrete space may be large [39]. Therefore, based on Wolpertinger strategy, we use the OPQ method to quickly find the discrete action closest to the continuous action. The specific method is the similar as the one in Section III-D1.

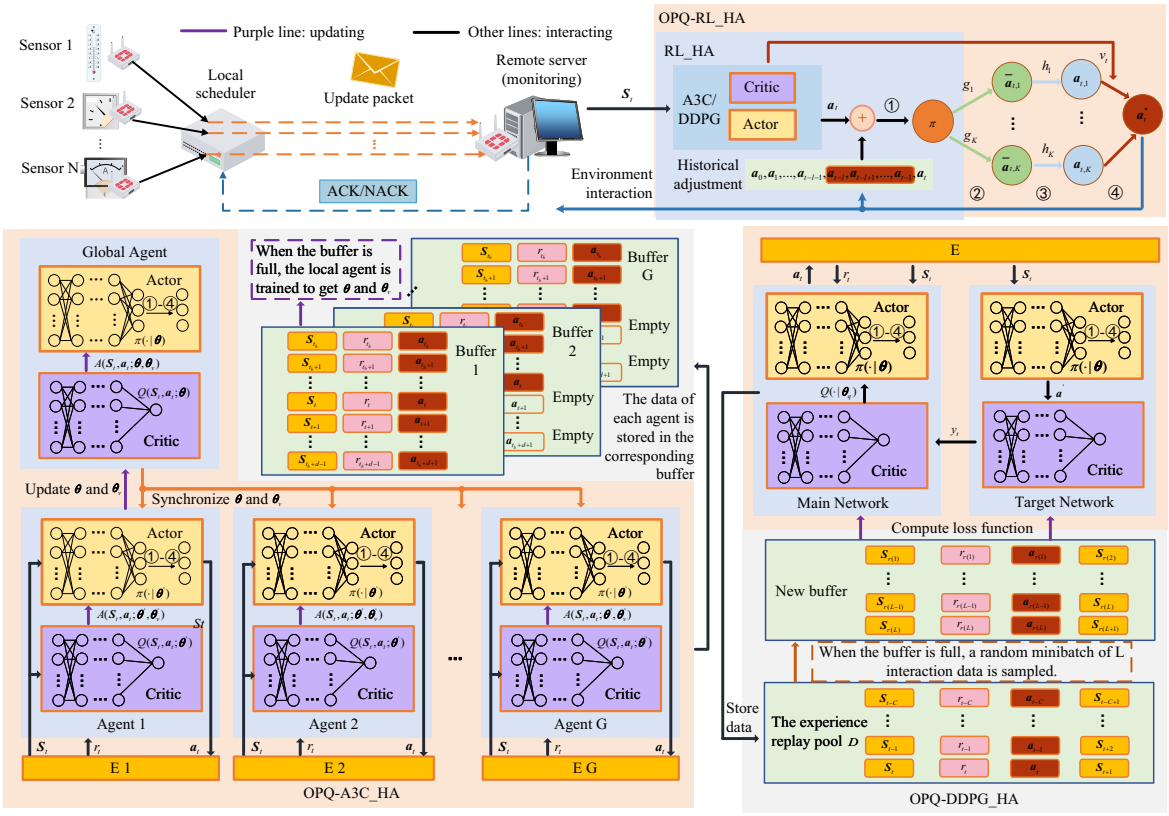


Fig. 4. The structure of scheduling policy and agent updating in the proposed OPQ-RL_HA.

The structure of scheduling policy to solve problem $\boldsymbol{Q1}$ and the updating of agent are illustrated in Fig. 4. For OPQ-A3C_HA, we can divide it into the following two parts: 1) action generation and interaction, 2) the parameters updating of actor and critic networks.

E. Convergence Analysis

Theorem 1. *Convergence of A3C. Under the Assumptions 2.2 and 4.1-4.5 in [40], the policy parameter θ obtained from (8) converges almost surely to a point in the set of asymptotically stable equilibria. When the error between the state-action value function Q and its linear approximation is small, the convergence point will correspond to a small neighborhood of a local optimum of the long-term average reward.*

In OPQ-A3C_HA, the output of the actor in A3C is primarily enhanced to achieve better decision-making results while meeting bandwidth constraints. Under the assumptions of Theorem 1, the proposed OPQ-A3C_HA is also convergent.

Theorem 2. *Convergence of DQN. Under the Assumptions 4.2 and 4.3 in [41], the state-action value function Q^{π_E} corresponding to the E-th iteration of training of the sparse neural network formed by ReLU activation functions can approximate the optimal state-action value function Q^* within a specific error range (the sum of statistical error and algorithmic error). Among them, the statistical error diminishes as the sample size in each iteration grows, whereas the algorithmic error decays to zero geometrically as the number of iterations increases.*

Combining with Theorem 1, the optimal state-action value function Q^* is substituted into (12) to obtain the optimal result of policy parameter θ_a, and thus verifies the convergence of OPQ-DDPG_HA.

IV. Numerical Analysis

In this section, we evaluate the effectiveness of the proposed OPQ-A3C_HA and OPQ-DDPG_HA algorithms in constrained homogeneous/heterogeneous systems by comparing them with the AoI-optimal, Probabilistic Ranking based AoII (PR-AoII), A3C_HA, DDPG_HA, A3C, DDPG algorithms.

In our work, AoI minimization is employed as the benchmark method (referring to as the *AoI-optimal algorithm*). By definition, a packet with timestamp u is said to have the AoI of

$$\delta_n(t) = t - u, \tag{27}$$

at time $t \geq u$, where the investigation is carried out with no bandwidth constraint. Therefore, the scheduling issue for a single sensor can be separated from the problem of minimization of the AoI for N sensors. If sensor n is scheduled at time-slot t, and $a_t^{(n)} = 1$, the value of AoI in the subsequent time-slot is either 1 with probability$\varepsilon_s^{(n)}$ or $\delta_n(t) + 1$ with probability $1 - \varepsilon_s^{(n)}$, therefore, we denote $W_1 = \varepsilon_s^{(n)} + (\delta_n(t) + 1)(1 - \varepsilon_s^{(n)}) + c_n$ as the one-step cost.

TABLE I
NETWORK PARAMETERS.

A3C		**DDPG**	
Actor	**Critic**	**Actor**	**Critic**
Input (3N, None)			
Dense_64	Dense_64	Dense_64	Dense_64
Dense_64	Dense_32	Dense_64	Dense_32
Dense_16	Dense_16	Dense_32	Dense_16
Dense_N (Sigmoid)[1]	Dense_1 (Linear)	Dense_N (Tanh)	Dense_1(Linear)
		Lambda($* \cdot 0.5 + 0.5$)	
$t_d = 5$		$\tau = 0.05$; $L = 16$; C^{buf}[2]$= 10000$	
E_{max}[3]$= 1000$; $\gamma = 0.9$; t_{max}[4]$= 100$			

[1] If there is no description $(\cdot)$, use ReLU activation function.
[2] The capacity of a buffer.
[3] The maximum episode.
[4] Truncate $\boldsymbol{Q1}$'s time-slot to t_{max} to make sure it can be processed.

On the other hand, if $a_t^{(n)} = 0$, let $W_0 = \delta_n(t) + 1$ represent the one-step cost. By comparing W_1 and W_0, we have

$$W_1 - W_0 = w_n - \delta_n(t) p_s^{(n)}, \tag{28}$$

which is a non-increasing function of $\delta_n(t)$. Therefore, if $\delta_n(t) \geq \frac{c_n}{\varepsilon_s^{(n)}}$, the scheduling decision $a_t^{(n)} = 1$ is preferable to $a_t^{(n)} = 0$. That is, if AoI is too large, the sensor should send its update packets. This scheme is referred to as the AoI-optimal method and serves as the baseline in our work.

Furthermore, we prioritize the scheduling of sensors based on their scheduling count. Assuming that sensors with fewer scheduling occurrences have a higher probability of being scheduled, we sort the scheduling probabilities in descending order. The sensors with top probability that also meet the bandwidth constraints will be scheduled. The above describes the PR-AoII benchmark method used in this section.

As shown in Table I, all networks have the same input layer, which includes 3N neurons that hold source statuses, remote servers, and AoII of N sensors. The outputs of the actor-network are N results (actions), and the critic network produces one result (Q value).

In addition, we consider high-bandwidth sensors with different bandwidths, which are distributed randomly in size from 5 to 10 MB/s, with a bandwidth limit of 37.5 MB/s [42]. In addition, we fixed random seeds to compare the results of different experiments.

In the following, we first analyze a system with homogeneous source statuses without bandwidth constraint, for both single sensor and multiple sensors scenarios, and we can derive theoretical results [6]. Next, we introduce the homogeneous statuses multi-sensor system with bandwidth constraints. Finally, we explore the multi-sensor system for heterogeneous statuses, whose theoretical inferences cannot be drawn.

A. Status-homogeneous Single Sensor System

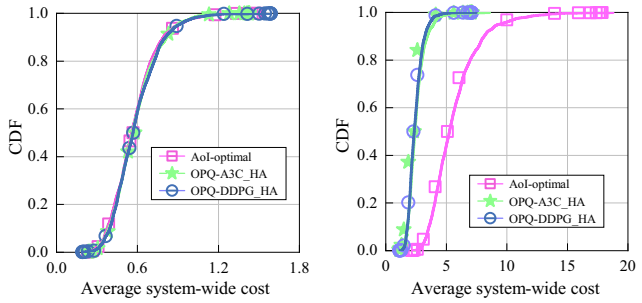
We take into account the status-homogeneous[5] single sensor system, which has 4 statuses (0 to 3), and a transmission cost $\boldsymbol{c}$ (zero vector). Below are provided the two alternative status transition probability matrices R_1 and R_2:

$$R_1 = \begin{bmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.7 & 0.1 \\ 0.1 & 0.1 & 0.1 & 0.7 \end{bmatrix} \text{ and } R_2 = \begin{bmatrix} 0.1 & 0.3 & 0.3 & 0.3 \\ 0.3 & 0.1 & 0.3 & 0.3 \\ 0.3 & 0.3 & 0.1 & 0.3 \\ 0.3 & 0.3 & 0.3 & 0.1 \end{bmatrix},$$

where the transition probability in R_1 are 0.7 and 0.1, respectively (transition probability represents the probability of each status transferring to another status and itself). Since $0.7 > 0.1$, selecting R_1 is more likely to result in a transformation to itself from the current status. And the opposite is held for R_2 (i.e., $0.1 \leq 0.3$). According to [6, Theorem 1], if the transition probability matrix is R_1, the transmitter on the sensor side should send update packets at the beginning of each time-slot. This includes two cases: when the remote server's estimate is accurate, there is no need to send packets (the AoII value is zero); but when the estimate of remote server is inaccurate, packets need to be sent (the AoII value grows with time). On the other hand, if R_2 is chosen as the transition probability matrix, the transmitter on the sensor side will not send any status update packets. The above is used as the optimal transmission strategy for this work. In this section, we assume the initial status of the source and the remote server are both 0, $\varepsilon_s = 0.8$, $l = 3$ for any sensors, and there is no bandwidth constraint.

The state-action transition sequence in which the agent moves from some initial state to the final reward state is known as an episode, and the episode is the most foundational

[5] In this case, the transition probability matrix is symmetric.



(a) Under $R1$. (b) Under $R2$.

Fig. 5. Performance comparison of different algorithms.

learning unit in reinforcement learning. As for systems with transition probability matrices $R1$ and $R2$, the results of OPQ-A3C_HA and OPQ-DDPG_HA algorithms are illustrated in Fig. 5. The vertical axis represents the average system-wide cost to generate the cumulative distribution function (CDF). According to [6, eq. (15)] and [6, eq. (16)], the expectation of system-wide cost with transition probability matrices R_1 and R_2 is **0.58777** and **2.5**, respectively. Whether OPQ-A3C_HA algorithm or OPQ-DDPG_HA algorithm, the values of CDF being 0.5 (correspond to average system-wide cost) are **0.58777** (in Fig.5(a)) and **2.5** (in Fig. 5(b)), respectively. In other words, both OPQ-A3C_HA and OPQ-DDPG_HA can achieve the theoretically best results. Furthermore, as for transition probability matrix R_1 in Fig. 5(a), the AoI-optimal approach also obtains the average system-wide cost, with an expectation of 0.58777, since the optimal policy under transition probability matrix R_1 keeps updating. However, compared to OPQ-A3C_HA and OPQ-DDPG_HA, the results of AoI-optimal are significantly greater in Fig. 5(b).

B. Status-homogeneous Multi-sensor System

In this section, we consider a multi-sensor system with 30 sensors, and its transition probability matrix is R_1 (15 sensors) and R_2 (another 15 sensors). Assuming $\varepsilon_s = 0.8$ for any sensors. When $c_n = 0$, and there is no bandwidth constraint, consider the optimal scheduling policy in section IV-A, i.e., the transmitter only sends the update packets with R_1. Therefore, we can obtain the expectation of system-wide cost as 46.3166.

Assuming $c_n \in (0, 1]$, and considering bandwidth constraints, the average system-wide cost using different algorithms are presented in Fig. 6. It is obvious that using the A3C/DDPG algorithm alone results in slower convergence. After adding the history adjustment, the convergence performance of the algorithm is improved. With the consideration of OPQ, both the convergence performance and decision-making results of the algorithm are enhanced. Fig. 7 shows the values of AoII and transmission cost corresponding to CDF is 0.5 for each algorithm. Obviously, the AoI-optimal method results in the highest AoII and transmission cost. The AoII results obtained using the PR-AoII method are similar to those of DDPG and DDPG_HA. However, the PR-AoII method incurs higher transmission costs. In addition, DDPG sacrifices AoII to minimize transmission cost. In OPQ-DDPG_HA, there

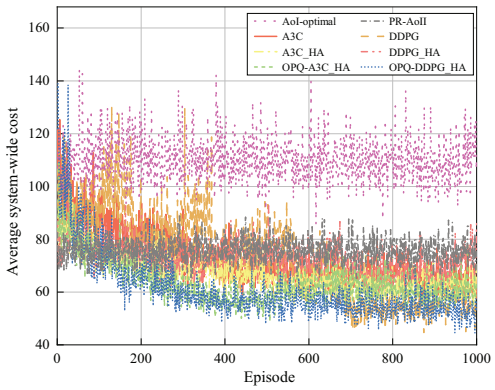


Fig. 6. Performance comparison under status-homogeneous system.

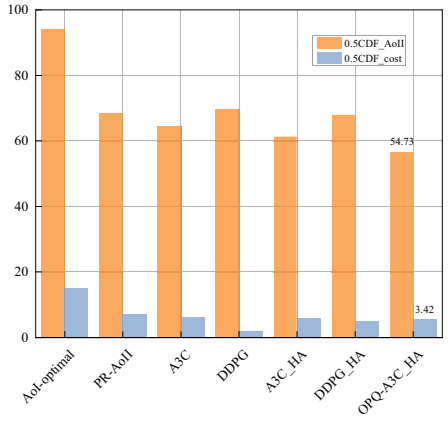


Fig. 7. The corresponding AoII and transmission cost values when the CDF of each algorithm is 0.5.

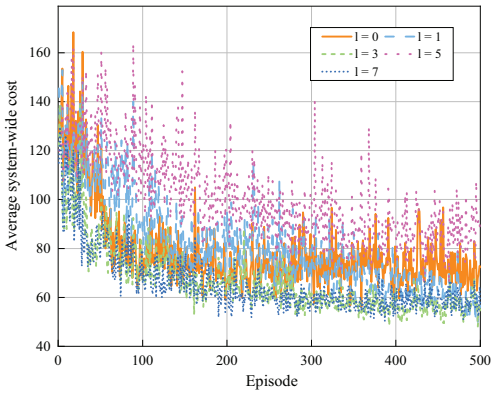


Fig. 8. Performance of OPQ-DDPG_HA under different scheduling counts.

is a 50% probability that AoII/transmission cost is below $54.73/3.42$, which shows the best performance.

To provide a clearer observation of the experimental results, in Fig. 8, we only display the average system-wide cost obtained using OPQ-DDPG_HA when l is 0, 1, 3, 5, and 7, respectively. In comparison to the scenario without historical adjustment, when l is too small, such as $l = 1$, the scheduling result is not good enough. However, when l is too large, one situation is that the algorithm is unstable. For example, when $l = 5$. Another case is that the performance of the algorithm

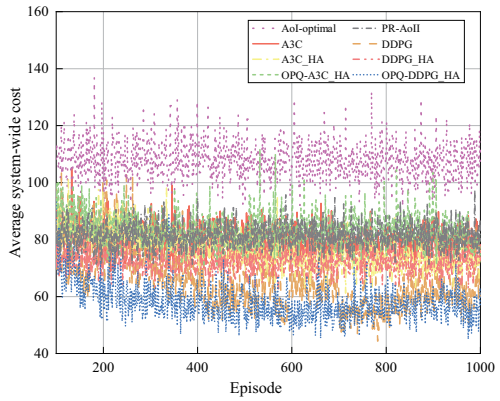


Fig. 9. Performance comparison under status-heterogeneous system.

is similar between each other, such as $l = 3$ and $l = 7$. In this case, choosing the adjustment with $l = 3$ can save computation resources.

C. Status-heterogeneous Multi-sensor System

The majority of source statuses in an actual IIoT system are heterogeneous. Therefore, it is essential to study a more general system. The two heterogeneous transition probability matrices below are what we leveraged in this section:

$$R_3 = \begin{bmatrix} 0.3 & 0.1 & 0.2 & 0.4 \\ 0.5 & 0.2 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ 0.6 & 0.1 & 0.2 & 0.1 \end{bmatrix} \text{ and } R_4 = \begin{bmatrix} 0.4 & 0.4 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.2 & 0.1 \\ 0.7 & 0.1 & 0.1 & 0.1 \end{bmatrix}.$$

The system has 30 sensors, including 8 sensors with transition probability matrix R_1, 8 sensors with transition probability matrix R_2, 7 sensors with transition probability matrix R_3, and 7 sensors with transition probability matrix R_4 respectively.

Fig. 9 shows the average system-wide cost of different algorithms in a multi-sensor heterogeneous system. Among them, the performance improvement of A3C using HA and OPQ methods is not obvious and is similar to the results of the PR-AoII method. OPQ-DDPG_HA has better decision-making results compared to A3C, and its convergence speed is faster than traditional DDPG. To further demonstrate the efficiency of OPQ-DDPG_HA, we compared four indicators (transmission cost, scheduling rate, bandwidth utilization, and AoII) of DDPG and OPQ-DDPG_HA when CDF is 0.5, as shown in Fig. 10. It is obvious that DDPG and OPQ-DDPG_HA have similar performances in terms of transmission cost, scheduling rate, and bandwidth utilization. The AoII value of OPQ-DDPG_HA is about 10.81% lower than that of DDPG, i.e., its convergence performance is better. Furthermore, the system-wide costs from the OPQ-DDPG_HA algorithms are much less than AoI-optimal method, which again demonstrates that AoII is a more efficient metric than AoI.

D. Comprehensive Comparison and Analysis

In order to provide a clearer exposition of the efficiency of the algorithm proposed in this paper for scheduling tasks, this section will focus on a comprehensive performance

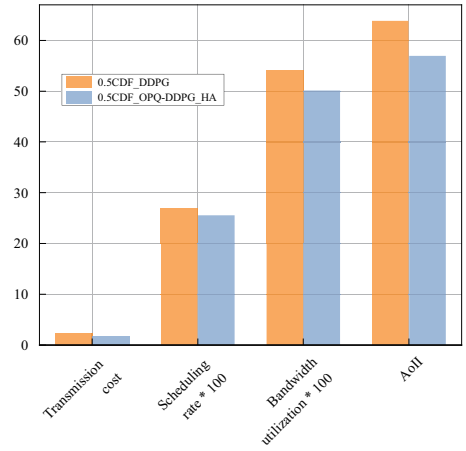


Fig. 10. Comparison of transmission cost, scheduling situation, occupied bandwidth, and AoII values of DDPG and OPQ-DDPG_HA when CDF is 0.5.

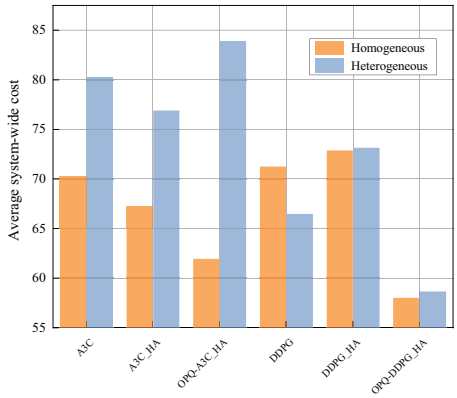


Fig. 11. Comparison of different algorithms in homogeneous and heterogeneous systems (with CDF at 0.5).

comparison among different algorithms in homogeneous and heterogeneous systems.

Fig. 11 illustrates the comparison of different algorithms in homogeneous and heterogeneous systems with the CDF of 0.5. Among them, when A3C considers the influence of the historical adjustment, the average system-wide cost decreases in both homogeneous and heterogeneous systems. However, with the addition of the OPQ method, it only yields improved results in the homogeneous system. Overall, A3C is more sensitive to HA. For the DDPG algorithm, HA has a negative impact, but the inclusion of OPQ leads to a significant reduction in average system-wide costs for DDPG in both homogeneous and heterogeneous systems, with the lowest values achieved. Therefore, the combined effect of HA and OPQ has a positive impact on DDPG. In summary, the performance of OPQ-DDPG_HA is shown as the best.

V. Conclusions

In this work, we have used DRL to address the problem of minimizing AoII in IIoT. First, we used CMDP to describe the multi-sensor scheduling problem, which is subject to bandwidth constraints. Subsequently, we proposed the OPQ-RL_HA framework to handle the issue of large state and

action spaces, and conducted experiments using A3C and DDPG under the framework. According to the numerical results, the proposed OPQ-DDPG_HA algorithm outperforms the OPQ-A3C_HA algorithm. Moreover, compared with the proposed approach, the traditional method of AoI-optimal is often ineffective since it disregards the statuses of the source and the remote server, which can lead to unnecessary updates.

REFERENCES

[1] J. Li, J. Tang, and Z. Liu, "Deep reinforcement learning for data freshness-oriented scheduling in industrial IoT," in *Proc. IEEE GLOBECOM*, Rio de Janeiro, Brazil, Dec. 2022, pp. 6271–6276.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[3] J. Li, J. Tang, and Z. Liu, "On the data freshness for industrial internet of things with mobile edge computing," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13 542–13 554, Aug. 2022.

[4] Q. Kuang, J. Gong, X. Chen, and X. Ma, "Analysis on computation-intensive status update in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4353–4366, Apr. 2020.

[5] H. Wu, H. Tian, S. Fan, and J. Ren, "Data age aware scheduling for wireless powered mobile-edge computing in industrial internet of things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 398–408, Jan. 2021.

[6] A. Maatouk, S. Kriouile, M. Assaad, and A. Ephremides, "The age of incorrect information: A new performance metric for status updates," *IEEE/ACM Trans. Netw.*, vol. 28, no. 5, pp. 2215–2228, Oct. 2020.

[7] K. C. Serdaroglu and S. Baydere, "An efficient multipriority data packet traffic scheduling approach for fog of things," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 525–534, Jan. 2022.

[8] Z. Yu, C. Hu, J. Wu, X. Sun, V. Braverman, M. Chowdhury, Z. Liu, and X. Jin, "Programmable packet scheduling with a single queue," in *ACM SIGCOMM 2021*, New York, NY, USA, Aug. 2021, pp. 179–193.

[9] M. Akbari, M. R. Abedi, R. Joda, M. Pourghasemian, N. Mokari, and M. Erol-Kantarci, "Age of information aware vnf scheduling in industrial iot using deep reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2487–2500, Aug. 2021.

[10] M. Chen, D. Gündüz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor, "Distributed learning in wireless networks: Recent progress and future challenges," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[11] E. T. Ceran, D. Gndz, and A. Gyrgy, "Average age of information with hybrid arq under a resource constraint," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Barcelona, Spain, Apr. 2018, pp. 1–6.

[12] W. Lin, X. Wang, C. Xu, X. Sun, and X. Chen, "Average age of changed information in the internet of things," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, Korea, May 2020, pp. 1–6.

[13] F. Chen and J. Tang, "Co-design of communication and control based on age of loop information in industrial iot," in *2023 IEEE 23rd Int. Conf. on Commun. Technol. (ICCT)*, Wuxi, China, Oct. 2023, pp. 896–902.

[14] A. Maatouk, M. Assaad, and A. Ephremides, "The age of incorrect information: An enabler of semantics-empowered communication," *IEEE Trans. Wireless Commun.*, vol. 22, no. 4, pp. 2621–2635, Apr. 2023.

[15] S. Meng, S. Wu, A. Li, and Q. Zhang, "Toward goal-oriented semantic communications: Aoii analysis of coded status update system under fbl regime," *IEEE J. Sel. Areas Inf. Theory*, vol. 4, pp. 718–733, Dec. 2023.

[16] Y. Chen and A. Ephremides, "Minimizing age of incorrect information over a channel with random delay," *IEEE/ACM Trans. Networking*, pp. 1–13, Apr. 2024.

[17] C. Kam, S. Kompella, and A. Ephremides, "Age of incorrect information for remote estimation of a binary markov source," in *Proc. IEEE INFOCOM Workshops*, Colorado, USA, Jul. 2020, pp. 1–6.

[18] Y. Chen and A. Ephremides, "Minimizing age of incorrect information for unreliable channel with power constraint," in *Proc. IEEE GLOBECOM*, Madrid, Spain, Dec. 2021, pp. 1–6.

[19] F. Peng, Z. Jiang, S. Zhou, Z. Niu, and S. Zhang, "Sensing and communication co-design for status update in multiaccess wireless networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 3, pp. 1779 – 1792, Mar. 2023.

[20] M. Ayik, E. T. Ceran, and E. Uysal, "Optimization of AoII and QAoII in multi-user links," in *Proc. IEEE INFOCOM 2023-IEEE Conf. on Comput. Commun. Workshops (INFOCOM WKSHPS)*, [Online], Apr. 2023, pp. 1–6.

[21] J. Chen, J. Wang, C. Jiang, and J. Wang, "Age of incorrect information in semantic communications for NOMA aided XR applications," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 5, pp. 1093–1105, Sep. 2023.

[22] H. Hong, J. Jiao, T. Yang, Y. Wang, R. Lu, and Q. Zhang, "Age of incorrect information minimization for semantic-empowered NOMA system in S-IoT," *IEEE Trans. Wireless Commun.*, vol. 23, no. 6, pp. 6639–6652, Jun. 2024.

[23] I. Ahmad, R. Narmeen, Z. Kaleem, A. Almadhor, Y. Alkhrijah, P.-H. Ho, and C. Yuen, "Machine-learning-based optimal cooperating node selection for internet of underwater things," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 22 471–22 482, Jun. 2024.

[24] X. Lu, L. Xiao, P. Li, X. Ji, C. Xu, S. Yu, and W. Zhuang, "Reinforcement learning-based physical cross-layer security and privacy in 6G," *IEEE Commun. Surv. Tutorials*, vol. 25, no. 1, pp. 425–466, Nov. 2022.

[25] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa, "Safe exploration in continuous action spaces," 2018. [Online]. Available: arXiv:1801.08757

[26] T. Xu, Y. Liang, and G. Lan, "Crpo: A new approach for safe reinforcement learning with convergence guarantee," in *Proc. ICML*, [Online], Jul 2021, pp. 11 480–11 491.

[27] L. Zhang, Y. Peng, W. Yang, and Z. Zhang, "Semi-infinitely constrained markov decision processes and provably efficient reinforcement learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–14, Jan. 2024.

[28] I. Kadota, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Minimizing the age of information in broadcast wireless networks," in *Proc. Annu. Allerton Conf. Commun. Control. Comput.*, Monticello, IL, USA, Sep. 2016, pp. 844–851.

[29] H. Tang, J. Wang, L. Song, and J. Song, "Minimizing age of information with power constraints: Multi-user opportunistic scheduling in multi-state time-varying channels," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 854–868, May 2020.

[30] X. Wu, X. Li, J. Li, P. C. Ching, and H. V. Poor, "Deep reinforcement learning for IoT networks: Age of information and energy cost tradeoff," in *Proc. IEEE GLOBECOM*, Taipei, Taiwan, Dec. 2020, pp. 1–6.

[31] D. P. Bertsekas, *Dynamic Programming and Optimal Control, 2nd ed.* Belmont, MA, USA: Athena Sci., 2000.

[32] D. Mehta, "State-of-the-art reinforcement learning algorithms," *International Journal of Engineering Research and Technology*, vol. 8, no. 1, pp. 717–722, Dec. 2020.

[33] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2581–2593, Nov. 2020.

[34] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Trans. Evol. Comput.*, vol. 28, no. 1, pp. 147–167, Feb. 2024.

[35] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks," *IEEE Trans. Mob. Comput.*, vol. 21, no. 3, pp. 940–954, Mar. 2022.

[36] S. Liu, Y. Yu, X. Lian, Y. Feng, C. She, P. L. Yeoh, L. Guo, B. Vucetic, and Y. Li, "Dependent task scheduling and offloading for minimizing deadline violation ratio in mobile edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 2, pp. 538–554, Feb. 2023.

[37] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, New York, New York, USA, Jun. 2016, pp. 1928–1937.

[38] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971

[39] Z. Yin, Z. Wang, J. Li, M. Ding, W. Chen, and S. Jin, "Decentralized federated reinforcement learning for user-centric dynamic tfdd control," *IEEE J. Sel. Top. Signal Process.*, vol. 17, no. 1, pp. 40–53, Jan. 2023.

[40] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. ICML*, Vienna, Austria, Jul. 2018, pp. 5872–5881.

[41] J. Fan, Z. Wang, Y. Xie, and Z. Yang, "A theoretical analysis of deep q-learning," in *Learning for Dynamics & Control*, Berkeley, California, Feb. 2020, pp. 486–489.

[42] S. Saponara, F. Giannetti, and B. Neri, "Design exploration for millimeter-wave short-range industrial wireless communications," in *Proc. IECON 2016-42nd ANN Conf. IEEE Ind. Electron. Soc.*, Florence, Italy, Otc. 2016, pp. 6038–6043.

Jianhua Tang (S'11-M'15-SM'24) received the B.E. degree in communications engineering from Northeastern University, China, in 2010, and the Ph.D. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2015. He was a Post-Doctoral Research Fellow with the Singapore University of Technology and Design from 2015 to 2016, and a Research Assistant Professor with the Department of Electrical and Computer Engineering, Seoul National University, South Korea, from 2016 to 2018. He is currently an Associate Professor with the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, China. His research interests include wireless communications, edge computing, network slicing and industrial Internet of Things.

He was honored with the 2020 IEEE Communications Society Stephen O. Rice Prize and the 2023 IEEE ICCT Best Paper Award. He is currently serving as an Editor for the IEEE WIRELESS COMMUNICATIONS LETTERS and DIGITAL COMMUNICATIONS AND NETWORKS.

Fanfang Chen received the Bachelor and Master degree from the School of Information Science and Engineering, Xinjiang University, Urumchi, China, in 2019 and 2022 respectively. She is currently pursuing towards the Ph.D. degree with the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, China. Her research interests include co-design of communication and control, semantic communication and industrial Internet of Things.

Jiaping Li received the Bachelor's degree from the School of Mechanical Engineering, Dongguan University of Technology, Dongguan, China, in 2018, and the Master's degree from the Shien-Ming Wu School of Intelligent Engineering, South China University of Technology, Guangzhou, China, in 2022. He is currently working for a semiconductor equipment company and engaged in robotics research. His research interests include age of information, reinforcement learning and robotic motion control.

Zilong Liu has been with the School of Computer Science and Electronic Engineering, University of Essex, since December 2019, first as a Lecturer and then a Senior Lecturer (since October 2023). He received his PhD (2014) from School of Electrical and Electronic Engineering, Nanyang Technological University (NTU, Singapore), Master Degree (2007) in the Department of Electronic Engineering from Tsinghua University (China), and Bachelor Degree (2004) in the School of Electronics and Information Engineering from Huazhong University of Science and Technology (HUST, China). He was also a Visiting PhD student to Hong Kong University of Science and Technology and the University of Melbourne. From Jan. 2018 to Nov. 2019, he was a Senior Research Fellow at the Institute for Communication Systems (ICS), Home of the 5G Innovation Centre (5GIC), University of Surrey, during which he studied the air-interface design of 5G communication networks. Prior to his career in UK, he spent nine and half years in NTU, first as a Research Associate (Jul. 2008 to Oct. 2014) and then a Research Fellow (Nov. 2014 to Dec. 2017). His PhD thesis "Perfect- and Quasi- Complementary Sequences", focusing on fundamental limits, algebraic constructions, and applications of complementary sequences in wireless communications, has settled a few long-standing open problems in the field.

His research lies in the interplay of coding, signal processing, and communications, with a major objective of bridging theory and practice. Recently, he has developed an interest in advanced 6G V2X communication, sensing, and localization technologies for future connected autonomous vehicles as well as machine learning for enhanced communications and networking.

He is a Senior Member of IEEE and an Associate Editor of IEEE Transactions on Vehicular Technology, IEEE Transactions on Neural Networks and Learning Systems, IEEE Wireless Communications Letters, IEEE Access, and Advances in Mathematics of Communications. He was the Hosting General Co-Chair of the 12th Sequences and Their Applications (SETA'2024, https://seta-2024.github.io/) and the 10th IEEE International Workshop on Signal Design and its Applications in Communications (iwsda2022.github.io). He was named the 2024 Outstanding Mid-Career Researcher in the Faculty of Science and Health, the University of Essex, in May 2024. He was a Consultant to the Japanese government on 6G assisted autonomous driving in January 2023. He was a Track Co-Chair on Networking and MAC in IEEE PIMRC'2023. Details of his research can be found at: https://sites.google.com/site/zilongliu2357.