STUDY ON THE AGE OF INFORMATION IN DIGITAL TWINS

STUDY ON THE AGE OF INFORMATION IN DIGITAL TWINS

By AMIRHOSSEIN AGHAEI, Electrical Engineering

A Thesis Submitted to the School of Graduate Studies in Partial Fulfillment of the Requirements for the Degree Master of Applied Science

McMaster University

MASTER OF APPLIED SCIENCE (2024)

Hamilton, Ontario, Canada (Electrical and Computer Engineering)

TITLE:	Study on the Age of Information in Digital Twins
AUTHOR:	Amirhossein Aghaei B.S., AmirKabir University, Tehran, Iran
SUPERVISOR:	Dr. Dongmei Zhao
NUMBER OF PAGES:	xiv, 82

Abstract

Digital Twins (DTs) have emerged as significant tools for real-time monitoring and control across various industries, offering dynamic digital replicas of physical systems (PSs). However, maintaining the freshness of information in DTs is challenged by communication delays, uncertain network conditions, and limited computational resources. These challenges can lead to increased Age of Information (AoI), reducing the effectiveness of DTs in time-sensitive applications where timely and accurate data is critical. This thesis addresses the optimization of PS-DT synchronization and DT response time to applications, aiming to minimize AoI while efficiently allocating communication and computational resources.

Firstly, we investigate the optimal DT response time when applications request real-time information. Considering uncertainties in wireless communication channels and the unpredictability of future AoI, we formulate the problem as a Markov Decision Process (MDP) with delayed rewards. To solve this, we employ reinforcement learning techniques, specifically combining Long Short-Term Memory (LSTM) networks with Dueling Double Deep Q-Networks (DDDQN). This approach enables the DT to decide whether to respond immediately to application requests or wait for fresher data from the PS, effectively balancing response timeliness and information freshness.

Secondly, we extend the optimization to multiple PS-DT pairs operating under

shared and constrained communication and computational resources. We model the system as a multi-agent environment where each PS-DT pair aims to keep the AoI at DT below a predefined threshold while minimizing power consumption during data transmission. The problem is formulated as a stochastic optimization task and addressed using a two-stage MDP framework. In the first stage, agents optimize transmission power considering channel interference in an orthogonal frequency-division multiple access (OFDMA) scheme. In the second stage, they request computational resources from the edge server (ES) with limited processing capacity. We utilize the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm within a centralized training and decentralized execution (CTDE) framework to solve the MDP efficiently.

Simulation results demonstrate that the proposed methods reduce the AoI at both DTs and applications, enhance resource utilization, and outperform existing algorithms in managing the trade-off between AoI and power consumption. The findings contribute to the efficient design and operation of DT systems in time-sensitive applications, ensuring timely updates and responses while optimizing resource allocation in constrained environments.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Dr. Dongmei Zhao, for her unwavering guidance and generous support throughout this journey. I am truly delighted to work under her supervision and have learned immensely from her, not only in technical aspects but also through her exemplary conduct and interactions. Without her profound insights, continuous support, and encouragement, this work would never have come to fruition.

To my beloved parents, Khadijeh and Reza, your unconditional support and unwavering belief in me have been my foundation. I learned to be a fighter during life's storms from my greatest supporter, my father, and I could never have completed this journey without my mother's encouragement and the immeasurable love she has given me throughout my life.

I would also like to extend my heartfelt thanks to my brother, Mojtaba, for his constant support and for always being there for me. His encouragement and understanding have been invaluable during the challenging times of this journey.

I am sincerely thankful to my labmates, Dr. Kiana Noroozi, Dr. Hong Chen, Mehrad Vaezi, and Mohammad Heydari, whose collaboration and camaraderie have been invaluable. Their support, insightful discussions, and shared dedication have made this research experience both productive and enjoyable. Working alongside

such talented and motivated individuals has greatly enriched my academic journey.

Finally, I am grateful to my friends Hamed Akhlaghi, Amir Eskandari, Nicky Anvari, and Alireza Daei Javad, who have consistently stayed in touch with me, bridging the distance and keeping us connected.

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviations

DT	Digital Twin
PS	Physical System
ES	Edge Server
MDP	Markov Decision Process
OFDMA	Orthogonal Frequency-Division Multiple Access
AoI	Age of Information
DDPG	Deep Deterministic Policy Gradient
TD3	Twin Delayed Deep Deterministic Policy Gradient
DDDQN	Dueling Double Deep Q-learning
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning

CTDE	Centralized Training with Decentralized Execution
ML	Machine Learning
CPU	Central Processing Unit
IIoT	Industrial Internet of Things
FSMC	Finite State Markov Channel

Declaration of Authership

I, Amirhossein Aghaei, declare that this thesis titled, and the work presented in it are my own.

Chapter 1

Introduction

1.1 Digital Twins and Definitions

A digital twin (DT) is a virtual representation of a physical system (PS) that mirrors the system's state, behavior, and performance in real-time. The "state" of a PS encompasses all relevant data about its current and historical conditions. To derive actionable insights, this state data must be processed to extract the system's status or feature information.

For instance, state data from a plant might include temperature, humidity, and noise levels from various locations. When analyzed together, this data could indicate whether the plant's current status is "normal" or "abnormal." When this status information is transmitted from the PS to its DT for processing and analysis, the timeliness and accuracy of representing the PS's current status become critical quality metrics for the DT.

To maintain the timeliness and accuracy, continuous "synchronization" between the PS and its DT is needed. The upper part of figure 1.1 shows an example of

periodic PS-DT synchronization. In each period of W, the state data is transmitted from the PS to the server where the DT is hosted and then processed at the DT. In the same figure, $t1$, $t4$ and $t7$ are the start time of the three updates, the intervals (t_1, t_2), (t_4, t_5), and (t_7, t_8) represent the data transmission time, and (t_2, t_3), (t_5, t_6), and (t_8, t_9) correspond to the execution times for each of the three updates.

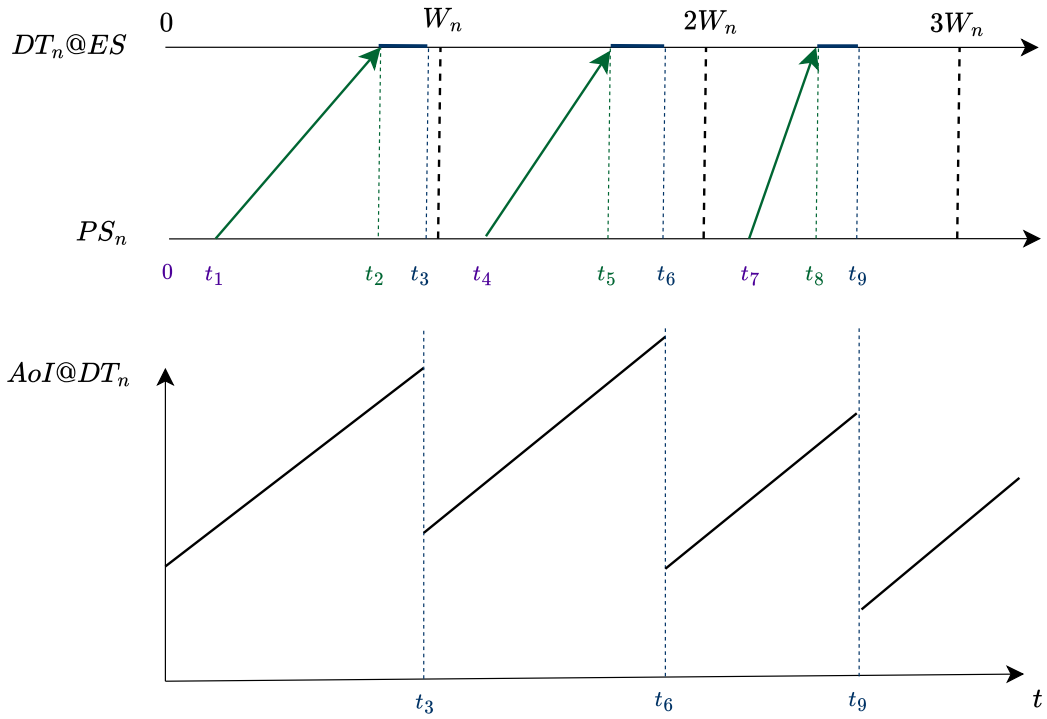

Figure 1.1: PS-DT update

The lower part of figure 1.1 shows the Age of Information (AoI) at the DT. The AoI is the time that has elapsed since the latest data that is used to produce the current information at the DT was generated at the PS. As illustrated in the figure, the AoI at the DT keeps increasing with a slope of 1 until the completion time of the next PS-DT synchronization update. Maintaining a low AoI at the DT requires minimizing both the data transmission delay from the PS to the DT and the processing time at the

DT. Furthermore, for applications that rely on the DT to retrieve feature information of a PS, additional delay occurs while fetching this information, during which the AoI of the feature data continues to increase.

1.2 Digital Twin Applications

DT technology has advanced significantly and is now utilized across various industries and applications. This section provides a brief overview of the initial purposes of DTs, their current implementations, and potential future applications.

DT technology initially emerged in manufacturing, where it was used to create virtual replicas of physical assets for monitoring and optimization purposes. Early applications focused on improving operational efficiency through simulations of equipment performance and predictive maintenance [14]. In these initial stages, the amount of data transferred from the physical part to its virtual replica was relatively limited, restricting the extent to which these virtual models could mirror real-world conditions.

Modern DTs have evolved into essential tools across various sectors, driving continuous and extensive data transfer to transform operations and outcomes. In healthcare, for instance, DTs combined with artificial intelligence enable personalized patient care by simulating treatment scenarios and predicting patient responses in real-time. This technology also extends to surgical simulations, where highly detailed virtual models of patients allow surgeons to practice and refine procedures before performing them on actual patients, leading to improved precision and outcomes [15], [4]. In the realm of smart cities, DTs are employed in traffic management and forecasting, where they integrate real-time sensor data with advanced simulations to optimize traffic flow, reduce congestion, and enhance accident response capabilities

[16], [45]. Additionally, in wireless networks, DTs are utilized to enhance network management and optimization. A critical aspect in this context is the AoI, which ensures that the data driving network decisions remain current and relevant. This real-time data integration is particularly vital for optimizing 5G networks and supporting connected infrastructure. For instance, DTs facilitate the visualization and prediction of 5G signal propagation within urban environments, contributing to more reliable and efficient network performance [40], [33], [29].

As DT use cases continue to evolve, future DTs are anticipated to achieve near-instantaneous synchronization with their physical systems (PS) through continuous real-time data streams. However, current challenges, such as delays in PS-DT synchronization and the sheer volume of data being transferred, hinder the full potential of DT applications. Advances in computing power will support the transfer of exponentially larger datasets between PS and DTs, enabling a more seamless interaction. Furthermore, the integration of artificial intelligence as a foundational component will empower DTs to autonomously analyze complex data, make informed decisions without human intervention, and continuously optimize their performance. These advancements are expected to drive the broader adoption of DTs across industries, unlocking new levels of efficiency and innovation.

These advancements will extend beyond traditional applications in manufacturing and smart healthcare to include the emerging concept of the Human Digital Twin (HDT). Building on insights from existing DT applications and incorporating technologies like Data Mining, Artificial Intelligence (particularly Deep Learning), and Human-Computer Interaction, HDTs will aim to manage the full lifecycle of human beings [32].

1.3 Age of Information at DTs

When a digital twin (DT) is used to monitor, control, or optimize its physical system (PS), maintaining a low Age of Information (AoI) at the DT ensures that it operates with the most up-to-date data, critical for accurate simulations and reliable decision-making. Conversely, a high AoI increases the risk of errors and undermines the reliability of real-time operations. The effectiveness of DT implementations depends heavily on the synchronization process between the PS and the DT, as highlighted in studies like [1] and [35]. This synchronization involves the continuous transmission of real-time data from the PS to the DT and subsequent processing at the DT to derive actionable insights. Consequently, robust networking support is vital to ensure timely data updates and maintain high-quality DT operations.

DTs can serve as intermediaries, interacting with third-party applications on behalf of their PSs. The integration of such applications introduces additional complexity and raises further considerations related to the AoI.

Time-sensitive applications utilizing DTs span diverse domains, including healthcare, manufacturing, and transportation. For example, in healthcare, real-time patient monitoring through DTs enables immediate responses to anomalies, potentially saving lives [17]. Similarly, in transportation, accident prevention in smart vehicles—highly reliant on data from PSs such as cameras—can be significantly enhanced through real-time synchronization between the vehicle and its DT. These use cases underscore the importance of maintaining low AoI to ensure timely and reliable data flow for critical decision-making across various industries.

To support time-sensitive applications, maintaining low AoI at both the DT and application levels is crucial to ensure that the feature information at the DT remains

current and aligned with application demands. Achieving this requires frequent, efficient updates between the PS and its DT, coupled with optimized server response times to applications. These measures ensure that the information provided by the DT remains relevant, up-to-date, and capable of meeting the stringent requirements of real-time decision-making.

1.4 Markov Decision Processes (MDPs) in Digital Twin Systems

In recent years, the integration of Markov Decision Processes (MDPs) into wireless networks and DT applications has gained considerable attention for addressing complex decision-making problems. MDPs are a mathematical framework from stochastic control processes, effective for modeling and solving sequential decision-making problems under uncertainty. Fundamentally, an MDP models a system that transitions through a sequence of states, with a decision-maker selecting actions at each state to guide the system's evolution. The defining feature of MDPs is the Markov property, which asserts that the future state depends solely on the current state and chosen action, independent of the system's history.

In the context of DTs, MDPs provide a structured approach to managing decisions in dynamic environments. The assumption of the Markov property simplifies modeling by positing that the DT's future state depends only on the current state of the PS and the server's action, eliminating dependence on prior states. This memoryless characteristic enables the modeling of dynamic changes in the PS and

wireless transmission environment as state transitions, facilitating adaptive and efficient decision-making. However, this also introduces significant complexity due to the need to address uncertainty and the vast state and action spaces that arise in practical applications.

An MDP includes four fundamental components: the state space, action space, transition probabilities, and reward function. The state space refers to the complete set of all possible states in which the system can exist. In the context of DTs, each state might represent a specific configuration of the physical system and its digital counterpart. The action space encompasses all possible actions or decisions that the decision-maker can take while in a particular state. These actions drive the system's transitions from one state to another. The transition probabilities define the likelihood of moving from one state to another given a specific action, encapsulating the inherent uncertainty and dynamics of the environment. Finally, the reward function assigns a numerical value to each state-action pair, representing the immediate benefit or cost of taking a particular action in a given state. This reward guides the decision-maker towards strategies that maximize cumulative rewards over time. By meticulously defining these components, MDPs provide a robust framework for modeling and optimizing decision-making processes in the inherently uncertain and dynamic environments typical of DT systems.

1.5 Challenges in Integrating MDPs for Digital Twin Systems

Integrating MDPs for resource allocation in PS-DT synchronization requires accounting for delayed feedback due to transmission and processing delays. Early decisions do not immediately reflect their impact on metrics such as the AoI or the accuracy of feature information at the DT, making delayed reward handling critical for effective decision-making. This delay can disrupt the learning process of the MDPs, which typically relies on immediate feedback to update its policy and improve decision-making over time. The temporal misalignment between actions and their consequences can lead to sub-optimal and erroneous decisions, as the MDP might update its policy based on outdated or irrelevant information. Furthermore, data transmissions for the PS-DT updates often require wireless transmissions. The uncertainty inherent in wireless communication can exacerbate these delays, making it difficult to accurately model the environment and predict outcomes. Addressing these challenges requires developing strategies that can tolerate or compensate for the feedback delay, such as by incorporating predictive models, adjusting the MDP's temporal resolution, or employing techniques that can mitigate the impact of outdated feedback on policy updates.

To address the challenge of delayed feedback in MDPs, in [18] novel algorithms have been developed that directly mitigate the impact of feedback delays on policy updates. These algorithms focus on minimizing regret, ensuring that decision-making remains effective even when feedback is received with significant delays. Another approach that addresses the challenge of delayed rewards in MDPs works by

redistributing rewards and transforming the learning task into a regression problem. This method simplifies Q-value estimation by focusing on immediate rewards and significantly improves learning efficiency, especially in environments with delayed feedback[2].

A significant challenge lies in the vast state and action spaces, which can grow exponentially with the number of PSs and other parameters. When employing machine learning algorithms to address this problem, the expansive state and action spaces necessitate extensive exploration and evaluation of possible scenarios. This increases computational complexity and significantly prolongs the convergence time needed to identify an optimal solution.

1.6 Contribution of Thesis

This thesis studies two important topics related to AoI at DTs, optimum DT response time in Chapter 3 and power efficient PS-DT update in Chapter 4.

In Chapter 3, we investigate the optimal response time for a DT when it receives a request from an application server to retrieve feature information about its PS. The DT faces a choice: respond immediately or wait until its information is next refreshed. This decision is complicated by the uncertainty in network conditions, which makes it challenging to predict the AoI at the DT in the future. This chapter examines the trade-off between the cost of delaying the response and the benefit of providing more accurate information, all while accounting for the stochastic nature of the communication channel between the PS and the DT. The problem is formulated as a MDP with delayed feedback. To address the delayed feedback, the reward structure is adjusted using a Long Short-Term Memory (LSTM) network to better

capture temporal dependencies. The optimal policies are then derived using Dueling Double Deep Q-learning (DDDQN). Numerical results demonstrate that the proposed solutions achieve near-optimal performance, effectively balancing response timeliness and data accuracy. This approach highlights the potential of advanced reinforcement learning methods to address challenges in DT systems operating under uncertain and dynamic network conditions.

In Chapter 4, we address network resource allocation for supporting the PS-DT update process. We consider a system where multiple PSs maintain their respective DTs at a shared edge server (ES) via a set of communication channels. The objective is to minimize the transmission power of the PSs while ensuring that the AoI at the DTs remains below a predefined target threshold. The problem is formulated as a MDP and decomposed into two stages: communication resource allocation (stage one) and computation resource allocation (stage two). Stage one leverages multi-agent reinforcement learning (TD3) for distributed decision-making, while stage two employs deterministic algorithms for computation resource assignments. This two-tiered approach enhances both scalability and decision-making efficiency.

S

Chapter 2

Literature Review

2.1 DTs and Network Management

DTs have been used to enhance network performance in wireless systems by creating virtual replicas of physical network components for real-time monitoring and optimization. For instance, DTs are developed in [29] to model 5G network signal propagation in three-dimensional urban environments, where the physical systems include the 5G infrastructure and urban terrain features affecting signal propagation. The DTs facilitate real-time signal optimization, improving network coverage and capacity. Similarly, DTs are used in [7] to emulate network components in real 5G networks for continuous prototyping, testing, and self-optimization. The physical systems in this case encompass automotive drive testing setups, smart factory configurations, and network planning for communications service providers. The use of DTs enhances network reliability, reduces network downtime, and improves resource allocation efficiency.

In the context of IoT networks, DTs are leveraged for dynamic resource management in edge networks supporting IoT applications in [44]. By modelling both the edge computing nodes and IoT devices, the DTs enable dynamic allocation of network resources that optimize resource utilization and improve service quality. DTs are utilized in [38] to create a virtual representation of the propagation environment in disadvantaged networks. Integrating the DTs with software-defined networking and machine learning techniques helps improve network topology management, leading to reduced path lengths and enhanced performance for IoT devices. DTs are created in [26] for wireless sensor networks equipped with distributed antenna arrays to optimize energy efficiency in large-scale MIMO systems.

2.2 AoI at DTs

For DTs to effectively support various applications, one of their essential performance metric is the AoI, which quantifies the information freshness at the DTs and is directly affected by network resource availability during the PS-DT synchronization. Supporting DTs consumes a considerable amount of network resources. Strategically placing DTs [40] [21] at network servers and allocating network resources [33] and [46] helps maintain low AoI at the DTs and improve the performance of the applications that need to fetch the PS information from their DTs. The work in [33] focuses on efficient network resource allocation to minimize AoI in the state representations of electric devices within DTs, while [46] introduced a distributed fault-tolerant communication algorithm that minimizes the expected peak AoI, ensuring up-to-date information is maintained even in resource-constrained and fault-prone environments. Further emphasizing the significance of AoI, Tang et al. [36] introduced an improved AoI

metric for DT synchronization in the Internet of Vehicles (IoV), which dynamically adjusts upload intervals and optimizes deployment strategies to enhance the freshness of vehicle state information while minimizing system costs and latency.

The importance of real-time synchronization between the PS and its DT has been emphasized in the literature, such as [37, 3, 27], while little work has been done toward maintaining tight synchronization between the two. From the networking perspective, such synchronization requires both short PS-DT communication delay and short data processing time, which translates to high bandwidth and computing power. The minimum AoI at the DT is bounded by the sum of minimum communication delay and data processing delay. Most existing work on DTs assumes that a DT for a given PS, such as individual devices in an IoT network [25], mobile device and edge servers of a mobile edge computing system [9], and vehicles and road side units of vehicular networks [49, 8], is already available to provide the information of the PS to an application. It should be noted that the AoI at the DT is an important factor that affects both the the application quality and the network resource management. In [39], DT placement was studied by optimizing the host server of the DTs of the PSs based on required AoI at the application and the network delay.

Research on joint communication and computation resource management in relation to AoI targets for DTs remains limited. Considering the dynamic nature of network conditions, the substantial resource demands of practical DTs, and the importance of maintaining information freshness, this area of study holds considerable significance. This work addresses this research gap by exploring the interplay between joint communication and computation resource allocation and the AoI performance of DTs.

2.3 DTs and ML

Digital Twins (DTs) are closely related to machine learning (ML), which has been used to build and maintain up-to-date DT models, and predict future states of the PSs. and optimize the performance of DT-based systems and applications. Advanced ML techniques within DT frameworks are used to optimize operations in dynamic environments. For example, Markov Decision Processes (MDPs) is used together with DTs of manufacturing machines and human operators for optimizing device assignment in manufacturing tasks in [12], and DRL is applied with DTs of dynamic and stochastic robotic construction environments in [20] for task prioritization in robotic construction." In the Industrial Internet of Things (IIoT), DRL within DT frameworks is used to optimize resource allocation and improve system performance. Reference [10] develops a DRL-based stochastic computation offloading scheme in DT networks using an asynchronous actor-critic algorithm. Similarly, [47] proposes a DRL-assisted federated learning framework in DT-empowered IIoT environments, enhancing model training efficiency and accuracy by selecting high-efficiency devices and addressing device heterogeneity. Reference [13] integrates expert knowledge, reinforcement learning, and DT technology for the self-optimization of 5G networks. By constructing a DT of the current network, optimization decisions can be simulated and evaluated, leading to enhanced performance without risking disruptions to the real network during training.

Chapter 3

Optimum DT Response Time for Time-Sensitive Applications

Real-time synchronization between a PS and its DT is crucial for ensuring that accurate status information is relayed from the DT to any application relying on the PS's state. However, delays in data communication and processing inherently introduce latency between the PS and its DT. Furthermore, unpredictable network conditions exacerbate these delays, making it challenging to forecast the AoI at the DT over time.

When an application sends a request to the DT, the DT faces a decision: respond immediately or wait until its information is refreshed during the next synchronization cycle. This chapter explores the optimal response time for the DT by weighing the costs of delaying responses against the benefits of providing more accurate and updated information. Additionally, the study incorporates the impact of variable and random communication channel conditions between the PS and DT. The analysis balances these factors to identify the most efficient response strategy, improving

decision-making under uncertain and dynamic network environments.

3.1 Introduction

As a digital replica of a physical system, a DT plays a vital role by interacting with third-party applications on behalf of its physical counterpart. For instance, if multiple applications need status information about a farm field (the physical system, or PS), sensed data such as temperature and humidity can be sent from the PS to these applications. With a DT of the farm field maintained at a server, the sensed data is first transmitted to the DT server, which processes the information to determine the field's status as either "normal" or "abnormal." Applications can then contact the DT directly to retrieve this status information.

This approach offers several advantages. First, the sensed data is sent only from the PS to the DT, regardless of how many applications require access to that information. Second, servers hosting DTs typically possess superior computing power and storage capabilities compared to the physical system itself. This allows the DT to process raw data into application-friendly insights, such as stating, "the plants are in normal condition."

To ensure that a DT reflects the real-time status of its physical counterpart, data sensed at the PS should be transmitted periodically to the DT and processed there. However, delays inherent in data communication and processing introduce latency between the PS and its DT. Additionally, unpredictable wireless channel conditions contribute to uncertainty, making it difficult to forecast the AoI at the DT over time. Consequently, when an application sends a request to the DT, it becomes challenging to determine whether the DT should respond immediately or wait until its next

scheduled information refresh.

This chapter focuses on addressing the uncertainty introduced by wireless communication channels, as sensed data are typically transmitted through these wireless pathways. The problem is formulated as a Markov Decision Process (MDP), with the objective of maximizing a reward that balances the DT response delay and the AoI experienced by the application upon receiving the response from the DT. The remainder of this chapter is organized as follows. The remainder of this chapter is structured as follows: Section 3.2 describes the system model and formulates the problem, which is subsequently expanded into a Markov Decision Process (MDP) in Section 3.3. Section 3.4 presents the proposed solution algorithms, including the reward redistribution mechanism and the Dueling Double Deep Q-learning (DDDQN) approach for policy optimization. In Section 3.5, we provide simulation results and analysis to demonstrate the effectiveness of the proposed methods. Finally, Section 3.6 concludes the chapter with a summary of the key findings and insights gained from the study.

3.2 System Model and Problem Formulation

We consider a PS-DT-application system, where the applications fetch information of the PSs through their respective DTs and each DT is synchronized with its own PS periodically. We focus on one PS and its DT. The PS updates the DT by sending its current status data to the DT every W seconds through a wireless channel.

We consider the periodic synchronization updates as shown in figure 1.1. Applications requiring state information of the PS send their requests to the DT. Upon the arrival of an application request, the DT may respond it immediately or delay the

response until the next data processing completion time. Either way, the DT always sends the most recent PS status information to the application. In figure 1.1, if an application request arrives at the DT at time t_a with t_a slightly less than t_6, then the optimum response time is to wait and send the response back to the application after t_6. On the other hand, if t_a is slightly larger than t_6, then the optimum response time is to send the response immediately. Both decisions are based on the amount of waiting time for the application to receive the PS state information and the AoI of the received information. In order to focus the research on the effect of uncertainty caused by wireless channels, we consider that the amount of time for sending the information from the DT to the application is a constant and small. When the application server is connected to the DT server through a wired network, in which case the DT-application communication delay is normally much less than the PS-DT communication delay. In addition, the amount of data delivered from the DT to the application is much less than that from the PS to the DT. The optimum response time, however, is not obvious for an arbitrary application request arrival time, due to the unknown next data processing finish time, which affects both the delay of DT response time and the updated AoI at the time.

We consider a finite-state Markov chain channel. Let τ be the duration of the time slot and N be the total number of channel states. Define P_{nm}^{CH} as the transition probability of the channel from state n in one slot to state m in the next slot. At state n, the data transmission rate is R_n in bits per second.

Let b_t be the number of remaining data bits to be transmitted from the PS to the

DT at the beginning of time slot t for the current PS-DT update period. We have

$$b_{t+1} = (b_t - R(t)\tau)^+, \tag{3.2.1}$$

where $(x)^+ = x$ if $x \geq 0$ and $(x)^+ = 0$ if $x < 0$, and $R(t) \in \{R_n|_{n=1}^N\}$ is the data transmission rate at time slot t.

Define c_t as the number of remaining CPU cycles to be processed at the beginning of time slot t. We have

$$c_{t+1} = (c_t - f_c\tau)^+. \tag{3.2.2}$$

At the beginning of any time slot t, b_t and c_t satisfy the following relationship

$$\begin{aligned}
&\text{phase 0:} \quad t = kW, \quad b_t = B_{\max},\ c_t = C_{\max},\\
&\text{phase 1:} \quad t \neq kW, \quad 0 \leq b_t < B_{\max},\ c_t = C_{\max},\\
&\text{phase 2:} \quad t \neq kW, \quad b_t = 0,\ 0 < c_t < C_{\max},\\
&\text{phase 3:} \quad t \neq kW, \quad b_t = 0,\ c_t = 0,
\end{aligned} \tag{3.2.3}$$

where k is any non-negative integer. Phase 0 (p0) is the start of an update period. In phase 2 (p2), state data of the PS is being transmitted to the DT server through the wireless channel. In phase 2 (p2), the DT is processing the data. In phase 3 (p3), the PS-DT update is completed for the current period.

At the beginning of time slot t, the AoI at the DT is given as

$$A_t^{\mathrm{DT}} = \begin{cases} (t \bmod W) + W, & \text{if DT in p0, p1 or p2} \\ (t \bmod W), & \text{if DT in p3} \end{cases} \tag{3.2.4}$$

where $(t \bmod W)$ finds the remainder of t/W.

Let A_t^{App} be the AoI of the application at the beginning of time slot t. The mean AoI at the application is given as

$$\overline{\mathrm{AoI}} = \frac{1}{D_{\max}} \int_{t=t_a}^{t_a+D_{\max}} A_t^{\mathrm{App}} dt. \tag{3.2.5}$$

Note that if the DT responds at time t, then $A_t^{\mathrm{App}} = A_t^{\mathrm{DT}}$; otherwise, $A_t^{\mathrm{App}} = A_{t-1}^{\mathrm{App}} + 1$.

3.3 MDP Formulation

The above problem can be formulated as a Markov decision process.

3.3.1 State space

Define s_t as the system state at the beginning of time slot t. We have $s_t = [g_t, b_t, c_t, o_t, d_t]$, where o_t is the AoI at the application server at time t, $b_t \in [0, B_{\max}]$ is the remaining number of bits to be transmitted, $c_t \in [0, C_{\max}]$ is the remaining number of CPU cycles to be executed, $g_t \in \{1, 2, \ldots, N\}$ is the channel state at time t, and $d_t \in \{0, 1, \ldots, D_{\max}\}$ is the amount of delay since the request arrival.

Note from (3.2.3) that b_t and c_t are closely related. Instead of having b_t and c_t as two different elements in s_t, we define a new element u_t with $u_t = b_t + c_t \in [0, U_{\max}]$

with $U_{\max} = B_{\max} + C_{\max}$. In this way, we have

$$\begin{aligned}
&\text{p0}, \quad b_t = B_{\max}, c_t = C_{\max}, u_t = U_{\max}, \\
&\text{p1}, \quad 0 \leq b_t < B_{\max}, c_t = C_{\max}, C_{\max} \leq u_t < U_{\max}, \\
&\text{p2}, \quad b_t = 0, 0 < c_t < C_{\max}, 0 < u_t < C_{\max}, \\
&\text{p3}, \quad b_t = c_t = 0, u_t = 0.
\end{aligned}$$

With this, the state is redefined as $s_t = [g_t, u_t, o_t, d_t]$.

3.3.2 Action space

Let a_t be the action at time t, then $a_t = \{0, 1\}$ with $a_t = 1$ if the DT sends the response at time t and $a_t = 0$ otherwise.

Define t_w as the amount of time from the application request arrival to the next PS-DT update completion, then $t_a + t_w$ is the PS-DT update completion time. The optimum DT response time is either at t_a or $t_a + t_w$. Therefore, our goal is to decide whether $a_{t_a} = 1$. If $a_{t_a} = 1$, $a_t = 0$ for all $t \in [t_a + 1, t_a + D_{\max}]$; otherwise, $a_{t_a+t_w} = 1$, and $a_t = 0$ for all $t \in [t_a, t_a + D_{\max}]$ and $t \neq t_a + t_w$.

3.3.3 State transition probabilities

In this subsection, we find the transition probability $\Pr(s_{t+1}|s_t, a_t)$. We drop the subscript t from all variables for concise expressions.

$$\Pr(s'|s,a) = \Pr(g', u', o', d'|g, u, o, d, a) \tag{3.3.1}$$

$$= \Pr(g'|g)\Pr(u', o', d'|g, u, o, d, a) \tag{3.3.2}$$

$$= \Pr(g'|g)\Pr(u'|g, u, o)\Pr(o'|u, o, a)\Pr(d'|d, a) \tag{3.3.3}$$

where (3.3.2) is due to the fact that the channel state changes are independent of all other elements in the state, and (3.3.3) is owing to unrelated amount of DT response delay to the data transmitting and processing at each time slot. Each of the probabilities in (3.3.3), will be derived in the remaining of this subsection.

For the channel state changes,

$$\Pr(g'|g) = P^{\mathrm{CH}}_{gg'},\ \forall g, g' \in \{1, 2, \ldots, N\}. \tag{3.3.4}$$

For the DT response delay, if the DT responds at a given time, the delay in the next time slot is reset to 0; otherwise, it is increased by 1. Therefore, we have

$$\Pr(d'|d,a) = \begin{cases} 1, & \text{if } \ d' = d+1,\ a = 0 \\ 1, & \text{if } \ d' = 0,\ a = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3.3.5}$$

The state transition probability during data transmission and processing can be

found according to (3.2.1)- (3.2.3) as follows:

$$\Pr(u'|g,u,o) = \begin{cases} 1, \text{ if } u=0, u'=0,\ (o \bmod W) < W-1 \\ 1, \text{ if } C_{\max} < u \leq U_{\max},\ u' = (u - R_g\tau)^+ \\ 1, \text{ if } 0 < u \leq C_{\max},\ u' = (u - f_c\tau)^+ \\ 0, \text{ otherwise} \end{cases} \tag{3.3.6}$$

For AoI at the application,

- when $a = 0$ and $0 < u \leq f_c\tau$,

$$\Pr(o'|u,o,a) = 1, \text{ if } o' = (o \bmod W) + 1. \tag{3.3.7}$$

- when $a = 0$, $u = 0$ or $u > f_c\tau$,

$$\Pr(o'|u,o,a) = 1, \text{ if } o' = o + 1 \tag{3.3.8}$$

- when $a = 1$ and $u = 0$,

$$\Pr(o'|u,o,a) = 1, \text{ if } o' = (o \bmod W) + 1. \tag{3.3.9}$$

- when $a = 1$ and $u \neq 0$,

$$\Pr(o'|u,o,a) = 1, \text{ if } o' = (o \bmod W) + W + 1. \tag{3.3.10}$$

- for all other cases, $\Pr(o'|u,o,a) = 0$.

3.3.4 Reward function

The reward function is defined as

$$r_{a_{t_a}} = \int_{t=t_a}^{t_a+D_{\max}} o_t dt. \tag{3.3.11}$$

If $a_{t_a} = 1$, $o_{t_a} = o_{t_a}^{\mathrm{DT}}$, and $o_t = o_{t_a} + (t - t_a)$ for all $t \in [t_a, t_a + D_{\max}]$. Therefore, (3.3.11) becomes

$$r_1 = \frac{1}{2}(2o_{t_a}^{\mathrm{DT}} + D_{\max})D_{\max}. \tag{3.3.12}$$

Similarly, if $a_{t_a+t_w} = 1$ (i.e., $a_{t_a} = 0$), (3.3.11) becomes

$$r_0 = \frac{1}{2}[(2o_{t_a}^{\mathrm{APP}} + t_w)t_w + (2o_{t_a+t_w}^{\mathrm{DT}} + D_{\max} - t_w)(D_{\max} - t_w)]. \tag{3.3.13}$$

With this, a revised reward can be defined as $r = r_1 - r_0$. By substituting (3.3.12) and (3.3.13) into r and simplifying, we have

$$r = (o_{t_a}^{\mathrm{DT}} - o_{t_a+t_w}^{\mathrm{DT}} + t_w)D_{\max} - (o_{t_a}^{\mathrm{APP}} - o_{t_a+t_w}^{\mathrm{DT}} + t_w)t_w \tag{3.3.14}$$

Upon the application request arrives at the DT, if $r \geq 0$, $a_{t_a} = 1$ is the optimal action; otherwise, $a_{t_a} = 0$. This is an MDP with delayed reward, since t_w is unknown at t_a.

3.4 Solutions

The MDP problem addressed in this work is characterized by delayed rewards. Specifically, the reward for a particular action taken upon request arrival is not disclosed until a new update is completed at DT. Addressing delayed reward MDPs introduces several significant challenges. Firstly, unlike immediate reward scenarios where a single state-action pair determines the reward, delayed rewards depend on a sequence of state-action pairs. The system's dynamics can lead to diverse future state-action sequences before the delayed reward is received, resulting in sparse rewards. This sparsity complicates the agent's ability to accurately predict the expected reward for its current actions. Secondly, determining the optimal policy becomes time-consuming as agents must comprehend the long-term consequences of their actions, which may not be immediately evident. This necessity for understanding extended temporal dependencies can hinder the efficiency and effectiveness of policy learning.

Given these challenges, traditional Dynamic Programming (DP) approaches become impractical for several reasons. DP requires exhaustive computation over the entire state-action space, which is computationally infeasible for large or continuous state spaces typical in real-world applications. Additionally, DP necessitates precise knowledge of transition probabilities and reward functions. In environments with delayed and sparse rewards, accurately modeling these elements can be overly complex or even unattainable, limiting the applicability of DP. Furthermore, the iterative nature of DP makes it slow to adapt to changes in the environment or to incorporate new experiences. In dynamic or stochastic settings where timely policy updates are crucial, DP's rigidity hampers its effectiveness. Given these limitations, alternative

approaches that can efficiently handle large state spaces, operate with partial knowledge of the environment, and address the complexities introduced by delayed rewards are essential. Reinforcement Learning (RL) techniques, particularly those designed to manage temporal credit assignment and sparse reward structures, offer promising solutions to these challenges.

Several approaches have been developed to address the challenges posed by delayed reward MDP problems. One such approach is a sequence modeling technique presented in [23], which focuses on temporal credit assignment in episodic reinforcement learning. This approach employs deep neural networks to decompose the episodic return back to each time-step in the trajectory, allowing the agent to learn the optimal policy more effectively and reducing the overall learning time. In [31], the temporal credit assignment problem in reinforcement learning is solved by utilizing InferNet, which is a Neural Network-based algorithm that is designed to learn how to infer immediate rewards from delayed and noisy feedback to the earlier states. This approach is also able to achieve faster and higher-scoring solutions to MDP problems.

Authors in [2] introduce RUDDER (Reward Redistribution for Delayed Rewards), a sophisticated reinforcement learning (RL) solution designed to address the inherent challenges of delayed rewards in Markov Decision Processes (MDPs). RUDDER constructs return-equivalent Sequence-Markov Decision Processes (SDPs) through a mechanism called reward redistribution. The primary objective is to transform the original MDP into an SDP where the expected future rewards are zero, thereby simplifying the estimation of Q-values to merely computing the mean of immediate rewards. This transformation is particularly advantageous in scenarios requiring swift

decision-making, such as responding promptly to application requests. Unlike traditional methods that merely decompose delayed rewards across time steps without ensuring zero expected future rewards, RUDDER redistributes rewards from the end of a sequence back through the entire state-action sequence. This approach not only aims to nullify future rewards but also significantly accelerates the learning of optimal policies, even if the reward redistribution is not perfectly optimal.

Central to the RUDDER's methodology is the use of a Long Short-Term Memory (LSTM) network as the return decomposition function. The LSTM predicts the entire sequence return based on the state-action sequence, enabling the redistribution of rewards in a manner that maintains return equivalence between the original SDP (ρ') and the transformed SDP (ρ). This process involves generating LSTM training samples, securing episodes with previously unseen delayed rewards, and training the LSTM to accurately predict the sequence-wide return at each time step. The reward redistribution is achieved by calculating the differences between consecutive return predictions, effectively spreading the final reward across all relevant state-action pairs in the sequence. Additionally, RUDDER employs a second-order Markov reward distribution, where the redistributed reward at each time step depends not only on the current state-action pair but also on the preceding pair. This second-order dependency ensures a more accurate attribution of rewards, mitigating the "explaining away" problem where early actions might otherwise have their contributions overshadowed by later ones. By defining a difference function $\Delta(s_{t-1}, a_{t-1}, s_t, a_t)$, RUDDER ensures that each state-action pair's contribution is precisely captured, maintaining the return equivalence and preserving the optimal policies of the original MDP.

RUDDER operates through a structured three-phase approach to efficiently implement reward redistribution and return decomposition. The first phase, Safe Exploration, generates high-quality training samples by avoiding episodes that result in low Q-values, which could indicate the agent is trapped in suboptimal states. The second phase involves populating a Lessons Replay Buffer with episodes that exhibit unexpected delayed rewards, identified by significant prediction errors from the LSTM. These episodes are prioritized during training to enhance the LSTM's predictive capabilities. In the final phase, LSTM Training and Return Decomposition, the LSTM is trained to predict the expected return for entire sequences at each time step. The differences between consecutive return predictions are then used to assign redistributed rewards to each state-action pair, ensuring accurate attribution of rewards and addressing potential biases. By incorporating a second-order Markov reward distribution into the reward redistribution process, RUDDER not only preserves the optimal policies of the original MDP but also enhances the robustness and efficiency of the learning process. Empirical studies have demonstrated that RUDDER significantly accelerates the learning of optimal policies, particularly in environments with sparse and delayed rewards, making it a powerful tool for complex, long-horizon decision-making tasks.

In our study, we have implemented a variation to the approach described in [2]. The algorithm proposed in [2] simultaneously updates the Q-values of (state,action) pairs while learning an LSTM network, and at the start of each episode, it selects the optimal policy based on the updated Q-values. In our research, we have redistributed the reward for episodes with the same action upon request arrival separately by training distinct LSTM networks. We have outlined the process of redistributing the

reward for episodes involving a wait action upon request arrival in Algorithm 1. This algorithm involves iterating through episodes where a wait decision was made upon request arrival and storing the transitions for each episode in the replay buffer. We then train the LSTM network for reward redistribution at a certain frequency using the stored transitions from the replay buffer.

Algorithm 1 Reward redistribution for waiting at request arrival

for `Episode` = 1 to Number of episodes **do**
  $\mathtt{s}_0 \leftarrow$ getting the system state with request arrival
  state-list $\leftarrow [\mathtt{s}_0]$, action-list $\leftarrow []$, reward-list $\leftarrow []$
  `episodeIter` = 0
  Response-Flag = False
  DT-Update = False
  for `episodeIter` = 1 to $D_{\max}$ **do**
    if `episodeIter` = 1 **then**
      `a` = `wait`
    else if DT-Update = True and Response-Flag = False **then**
      `a` = `send`
      Response-Flag = True
    else
      `a` = `wait`
    end if
    $\mathtt{s}_{\mathtt{next}}, \mathtt{r}_{\mathtt{next}} \leftarrow$ Observe next state and reward
    if `u` in $\mathtt{s}_{\mathtt{next}}$ = 0 and DT-Update = False **then**
      DT-Update = True
    end if
    Add $\mathtt{s}_{\mathtt{next}}$ to state-list
    Add `a` to action-list
    Add $\mathtt{r}_{\mathtt{next}}$ to reward-list
  end for
  Add [state-list, action-list, reward-list] to the Reply Buffer
  if Episode % TrainingFrequency = 0 **then**
    Train the LSTM network
  end if
end for

The aforementioned procedure is replicated for episodes with respond immediately action upon a request arrival. As a result, there are two distinct LSTM networks that

can redistribute rewards for episodes with waiting and sending back upon request arrival.

Now for finding optimal policies with redistributed rewards, a viable option is Q-learning which uses iterative updates to estimate the action value function and build policies based on a Q-table as follows [34].

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{3.4.1}$$

However, Q-learning is not suitable for problems with large state and action spaces due to the requirement of generating a Q-value table for every feasible state-action combination. Deep Q-learning [28] is presented as a potential solution to this problem, as it uses a neural network to approximate the Q-function, allowing for generalization across similar states and actions. However, overestimation can still be a problem in Deep Q-learning. Double Deep Q-learning [42] by utilizing two separate networks to select the best action and estimating its value is an approach for solving overestimation problem of Deep Q-learning. Dueling Double Deep Q-learning (DDDQN) introduced in [43] is more effective than Double DQN. DDDQN uses two estimators, one for calculating the state value function, and the other for calculating the state-dependent action advantage function. The outputs of these estimators are then combined to determine the Q-values. We adopt DDDQN in this work.

Now after constructing LSTM networks according to algorithm 1 and redistributing rewards for different episodes, we train the DDDQN agent using Algorithm 2. Once the DDDQN agent is fully trained, we can determine the optimal policy for each state by iterating through all possible states that may have request arrivals and providing them as input to the DDDQN agent. This results in an response scheduler

that can make decisions based on the optimal policies.

3.5 Simulation Result and Analysis

In this section, we demonstrate performance of algorithms 1 and 2. Default parameter values are listed in Table 3.1. Every episode length in algorithms 1 and 2 is $D_{\max}$, which starts from request arrival and lasts for $D_{\max}$ seconds. That is, each episode is for a single request arrival. We assume that application requests arrive at the beginning of each update interval, a scenario representing the highest uncertainty regarding the update completion time and thus requiring non-trivial decisions for determining the DT's response time.

We conduct 100 episodes using the proposed DT response scheduler, calculating the mean AoI at the application for each episode. We also perform this analysis assuming that the decision is always to respond immediately upon the request arrival ("Respond immediately"), and compare the results.

Figures. 3.1-3.3 show the results with the channel-related parameters on the top of each figure. In each of the figures, the top sub-figure shows the average AoI of using the proposed response scheduler and the "Respond immediately" for each episode (i.e., for each application request), and the bottom sub-figure shows the difference of the average AoI using the two response decision methods. When the difference is positive, the proposed response scheduler achieves better (i.e., lower) average AoI at the application than the "Respond immediately" method.

All the figures show that the "Respond immediately" solution results in much higher average AoI at the application. In each of the three figures, there are some episodes (i.e., application requests) that have zero difference in Average AoI, in which

case responding immediately is the optimal policy; in each of figures. 3.1 and 3.3, there are 2 out of 100 episodes that have negative difference in Average AoI, in which case the proposed response scheduler did not reach the optimum decision; and for all other episodes, the proposed response scheduler results in lower average AoI than the "Respond immediately" method.

Furthermore, in figure 3.1, the scheduler tends to make more "send" decisions (i.e., responding immediately upon a request arrival) than "wait" decisions, since the poor transmission channel condition results in a higher probability of long waiting time until the next PS-DT update completion. In contrast, in figure 3.3, the scheduler tends to make more "wait" decisions (i.e., responding until the next PS-DT update completion) than "send" decisions because of the better channel conditions. As a result, there are more "zero"s in figure 3.1(b) than in figure 3.3(b).

In addition, when "Respond immediately" is not the optimum decision, the proposed response decision method helps improve the average AoI at the application by at most 60 milliseconds (or W), and this maximum improvement (or average AoI reduction) is independent of the channel condition. The exact average AoI reduction varies among the episodes, depending on the system state at the request arrival time.

3.6 Summary

In this chapter, we studied the optimal response time for a DT to provide real-time information from a PS to applications, considering the uncertainties in network conditions and the cost associated with delaying the response. The problem was modeled as a Markov Decision Process (MDP) with delayed feedback and solved through a

Table 3.1: Simulation Parameters

Parameter	**Value**
$B_{\max}$	100 K bits
$C_{\max}$	10 M CPU cycles
$D_{\max}$	50 ms
f_c	1G cycles/s
W	60 ms
N	2
R_2	10 M bits/s
R_1	2 M bits/s
τ	1 ms
γ	0.99
α	3×10^{-4}

combination of LSTM networks and Dueling Double Deep Q-Network (D3QN). Simulation results highlighted the differences in average AoI between our proposed AoI scheduler algorithm and the strategy of sending DT data immediately upon request, varying under different transmission channel conditions. Our findings show that our proposed approach achieves near-optimal DT response time, which leads to improved information accuracy at the applications under dynamic and uncertain communication scenarios.

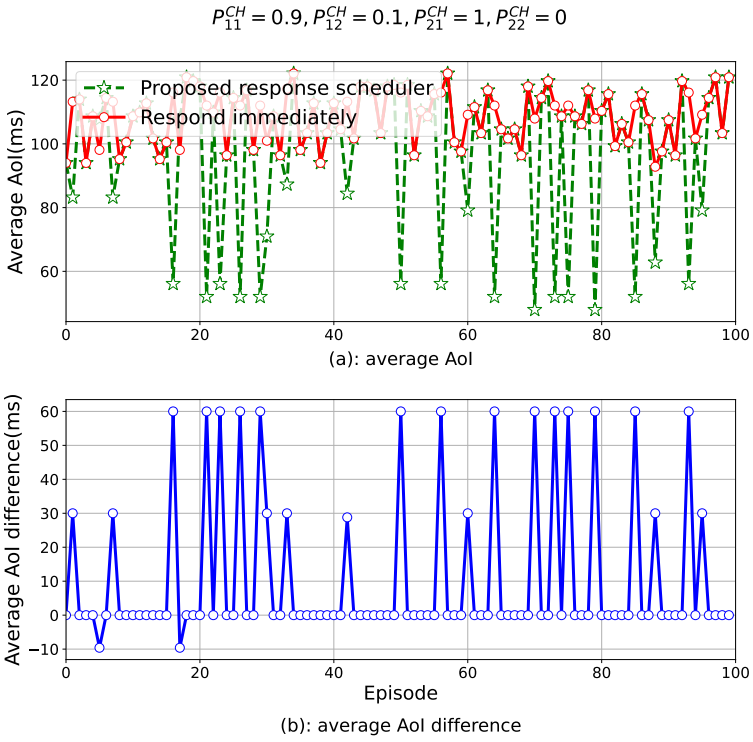

Figure 3.1: AoI performance: poor channel condition

Algorithm 2 Finding optimal policies with DDDQN

for `Episode` = 1 to Number of episodes **do**
  $\mathtt{s}_0 \leftarrow$ getting the state with request arrival
  LSTM1 $\leftarrow$ Load LSTM network for wait decision
  LSTM2 $\leftarrow$ Load LSTM network for send decision
  state-list $\leftarrow [\mathtt{s}_0]$, action-list $\leftarrow []$, reward-list $\leftarrow []$
  `episodeIter` = 0
  Response-Flag = False
  DT-Update = False
  for `episodeIter` = 1 to $D_{\max}$ **do**
    if `episodeIter` = 1 **then**
      `a` $\leftarrow$ Choose action with DDDQN agent
    else if DT-Update = True, action-list[0] = wait and Response-Flag = False **then**
      `a` = `send`
      Response-Flag = True
    else
      `a` = `wait`
    end if
    $\mathtt{s}_{\mathtt{next}}, \mathtt{r}_{\mathtt{next}} \leftarrow$ Observe next state and reward
    if `u` in $\mathtt{s}_{\mathtt{next}} = 0$ and DT-Update = False **then**
      DT-Update = True
    end if
    Add $\mathtt{s}_{\mathtt{next}}$ to state-list
    Add `a` to action-list
    Add $\mathtt{r}_{\mathtt{next}}$ to reward-list
  end for
  if action-list[0] = wait **then**
    reward-list $\leftarrow$ Redistribute reward with LSTM1
  else
    reward-list $\leftarrow$ Redistribute reward with LSTM2
  end if
  for iter = 1 to $\mathtt{D}_{\max}$ **do**
    Store iter-th transition of `Episode` in the DDDQN Buffer
    Train the DDDQN agent
  end for
end for

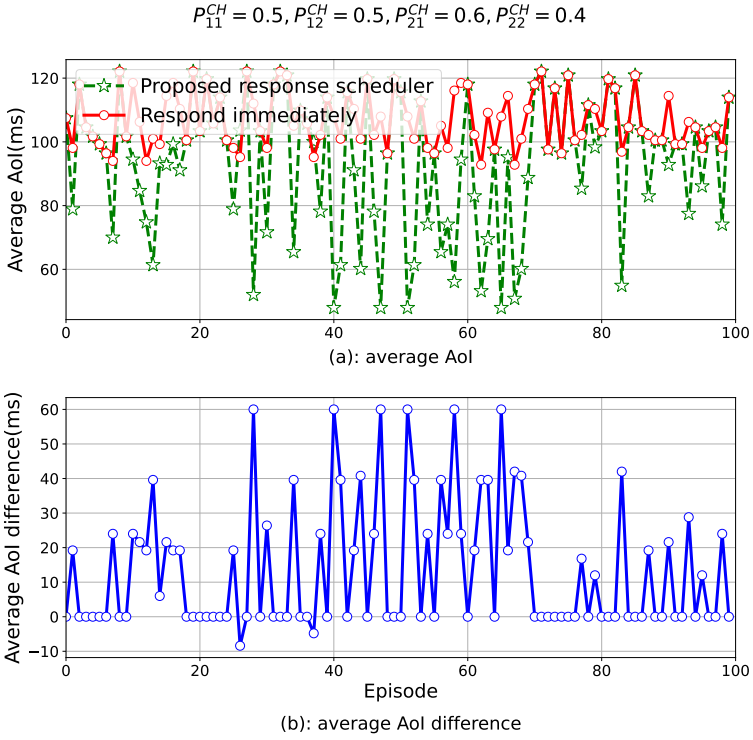

Figure 3.2: AoI performance: medium channel condition

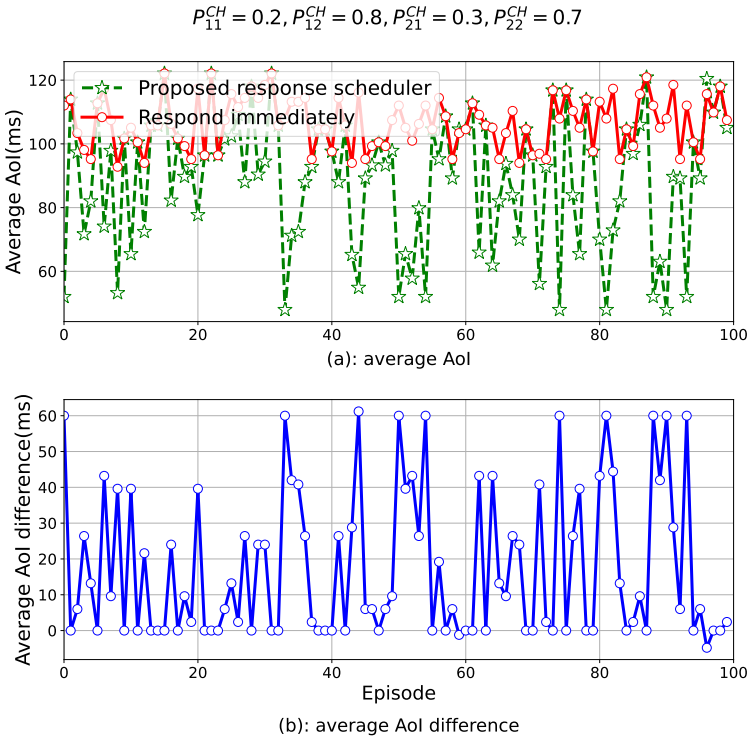

Figure 3.3: AoI performance: good channel condition

Chapter 4

Power Efficient Networking Support for Digital Twins with Age of Information Targets

This chapter investigates the joint allocation of communication and computation resources in scenarios where multiple PSs maintain their DTs at a shared edge server (ES) connected through a set of communication channels. The primary objective is to minimize the transmission power of the PSs while ensuring that the AoI at their corresponding DTs remains below a predefined target. This chapter integrates optimization strategies that balance efficient resource use with timely and reliable data delivery to maintain system performance.

4.1 Introduction

The rapid growth of Digital Twin (DT) technology is transforming various industries such as manufacturing, healthcare, transportation, and smart cities, where real-time monitoring and control of physical systems (PSs) are critical to ensuring efficiency, reliability, and safety. A DT is a digital replica of a physical asset, process, or system and is continuously updated to reflect the current status of its PS. By creating a virtual mirror of the physical entity, DTs enable organizations to simulate scenarios, predict system behavior, and make informed decisions. These capabilities are particularly useful for tasks such as predictive maintenance [41], operational optimization, and disaster recovery [48], as DTs provide insights that go far beyond traditional methods of monitoring and control.

A major advantage of DT technology is the ability to offer actionable insights through continuous synchronization between a DT and its PS [1], [35]. This real-time connection is needed to ensure that a DT is always an accurate reflection of its physical counterpart, thereby improving decision-making and allowing operators to address potential issues before they become critical. However, synchronization between a PS and its DT requires networking support. The PS-DT synchronization requires state-related data at the PS to be periodically delivered to the server that hosts the DT and then processed at the server to extract the status features of the PS. Given the limited amount of network resources, information reflected at the DT is always delayed. This delay includes both data transmission delay over the network and data processing delay at the server. The age of information (AoI) at the DT is the time that has elapsed since the latest data used to produce the current status information at the DT was originally generated at the PS. Maintaining low AoI at

DTs is important for applications that rely on the DTs for time-critical tasks, such as automation control and traffic navigation.

DTs are more suitable to be hosted at the network edge in order to reduce the communication delay in the PS-DT synchronization. However, the limited amount of computational resources at the network edge makes it challenging to meet the growing demands for digital twins. In addition, hosting more DTs at the network edge increases the communication load at the edge, requiring more efficient communication resource allocations and joint communication and computation resource allocations among the PS-DT pairs.

This chapter studies a scenario where multiple PSs maintain their DTs at a shared edge server (ES) through a set of radio communication channels. The channels experience random fading, which causes uncertainty in data transmission time in PS-DT synchronization. The work studies how to optimize the transmission power of the PSs in order to maintain the AoI at the DTs to be below a certain limit while minimizing the average power consumption of the PSs. The joint communication and computation management problem is formulated as a constrained Markov decision process. It is then decomposed into a two-stage decision-making problem, stage one for communication resource allocations and stage two for computation resource allocations. While the stage-two decisions are solved by using deterministic algorithms, the stage-one decisions are reformulated as a multi-agent reinforcement learning problem and solved using Twin Delayed Deep Deterministic Policy Gradient (TD3). Using a deterministic algorithm to make the stage-two decisions not only helps reduce the complexity in making the stage-one decisions, but also improves the efficiency for the RL algorithms designed for stage-one resource allocations to learn the delayed reward

and minimize the AoI violation later on. The multi-agent RL framework for making the transmission power decisions of individual PSs in stage one enhances the solution scalability. The integration of the TD3 algorithm within a centralized training with decentralized execution (CTDE) framework enables each physical system (PS) to efficiently learn optimal power allocation strategies, enhances coordination and learning efficiency by leveraging global information during training, and supports independent and scalable decision-making during execution. Simulation results show that the proposed solution outperforms existing solutions in terms of both AoI at the DT and average power consumption of the PSs.

The remainder of this chapter is structured as follows: Section 4.2 describes the system model and formulates the resource allocation problem, which is reformulated into an MDP in Section 4.3. Section 4.4 proposes the algorithms for making stage-two decisions and Section 4.5 presents the algorithms for making stage-one decisions. In Section 4.6, we present simulation results to demonstrate the effectiveness of the proposed solution.

4.2 System Description

We consider a system comprising multiple PSs and an edge server co-located with a cellular base station (BS). Denote $\mathcal{N} = \{1, 2, \ldots, N\}$ as the set of the PSs with N the total number of PSs. Each PS maintains a DT at the ES. We use $n \in \mathcal{N}$ to index the nth PS, the nth DT, or the n-th PS-DT pair, when there is no ambiguity. In order to keep the information at the DTs up-to-date, each PS-DT pair should perform periodic synchronization. Let W_n be the synchronization period of the nth PS-DT pair. During each update, the PS sends its current state data through the

cellular network to the ES. We assume that the transmission delay between the BS and the ES can be neglected. Let j denote the jth update period of a given PS-DT pair, where $j = 1, 2, \ldots$. The start time of the jth update for the nth PS-DT pair is given as $(j-1)W_n + t_{s,n}$, where $t_{s,n}$ is a time offset that allows us to take into consideration the fact that different PSs may start transmitting their synchronization data at a different time. The state data of the PS is processed at the ES so that the updated feature information of the PS can be extracted. For each synchronization update of the nth PS-DT pair, let $B_{\max,n}$ and $C_{\max,n}$, respectively, be the amount of data to be transmitted from the PS to the DT and amount of computation load required to process the data at the ES.

4.2.1 Communication model

The orthogonal frequency-division multiple access (OFDMA) is adopted for the PSs to communicate with the BS for their data transmissions. Denote $\mathcal{K} = \{1, 2, \cdots, K\}$ the set of the channels with K the total number of channels, each of which has a bandwidth of B Hz. The system time is divided into equal size time slots of length τ so that the channel condition can be assumed to be constant within each time slot. We consider finite state Markov channels [30] with H states and use $h_{n,k,t}$ to represent the link gain of PS n at frequency channel k at time t. That is, $h_{h,k,t} \in \mathcal{H} = \{g_1, g_2, \cdots, g_H\}$. That is, at any time slot t, if the channel gain $h_{n,k,t} = g$, then the probability that $h_{n,k,t+1} = g^{'}$ is $P_{g,g'}$ with $\sum_{g' \in \mathcal{H}} P_{g,g'} = 1$, $\forall g \in \mathcal{H}$.

Let $p_{n,k,t}$ be the transmission power of PS n in channel k at time t. The total

amount of data that can be transmitted by PS n at time slot t is given by:

$$r_{n,t} = \tau B \sum_{k} \log \left(1 + \frac{p_{n,k,t} h_{n,k,t}}{\sum_{m \neq n} p_{m,k,t} h_{m,k,t} + N_0 B} \right), \tag{4.2.1}$$

where N_0 is the power spectrum density of additive white Gaussian noise. The total transmission power of PS n at time t is

$$p_{n,t} = \sum_{k=1}^{K} p_{n,k,t}. \tag{4.2.2}$$

4.2.2 Computation model

Let f_C denote the computation capacity of the ES, measured as the total number of available CPU cycles per time slot. If $f_{n,t}$ is the number of CPU cycles allocated to the nth PS-DT update at time slot t. The following condition should hold

$$\sum_{n=1}^{N} f_{n,t} \leq f_C. \tag{4.2.3}$$

4.2.3 Problem formulation

Our objective is to minimize the long-term average power of the PSs, subject to the total amount of computation resource at the ES and that the AoI at DT n does not

exceed a maximum value $D_{\max,n}$. The optimization problem is given as

$$\text{P0:} \min_{\mathbf{p},\mathbf{f}} \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N}\sum_{n=1}^{N}\sum_{k} p_{n,k,t} \tag{4.2.4}$$

$$\text{s.t.} \sum_{n\in\mathcal{N}} f_{n,t} \le f_C,\ t = 1, 2, \ldots \tag{4.2.5}$$

$$o_{n,t} \le D_{\max,n},\ \forall n \in \mathcal{N} \text{ and } t = 1, 2, \ldots. \tag{4.2.6}$$

where $\mathbf{p} = [p_{n,k,t}, \forall n, k, t]$ and $\mathbf{f} = [f_{n,t}, \forall n, t]$. The problem cannot be solved directly since it requires future channel condition information, which is unknown when the channels experience random fading. In the next section, we reformulate the problem into a Markov decision process so that sequential decisions on transmission power and computation resource allocations can be made based on the current system state.

4.3 MDP Formulation

In this section, we first define the state space, action space, and state transitions. We then analyze the complexity of the MDP and reformulate it to achieve reduced complexity.

4.3.1 State space

Define $\mathbf{S_t}$ as the system state at the beginning of time slot t. We have:

$$\mathbf{S_t} = [\mathbf{s_{n,t}}, \forall n \in \mathcal{N}] \tag{4.3.1}$$

where $\mathbf{s_{n,t}}$ is the state of PS-DT n at time slot t given as

$$\mathbf{s_{n,t}} = [\mathbf{h_{n,t}}, o_{n,t}, b_{n,t}, c_{n,t}] \tag{4.3.2}$$

and $\mathbf{h_{n,t}} = [h_{n,k,t}, \forall k \in K]$ represents the link gains of n-th PS, $o_{n,t}$ is the AoI at the n-th DT, $b_{n,t}$ is the remaining number of bits to be transmitted from the PS to the DT in the current update, and $c_{n,t}$ is the remaining number of CPU cycles to be processed for the current update.

4.3.2 Action space

At each time slot t, the action $\mathbf{a}_t$ consists of actions for transmission power of individual PSs at different channels and computation resource allocations among the DTs. That is,

$$\mathbf{a}_t = [\mathbf{a}_t^p, \mathbf{a}_t^f] = [\mathbf{a}_{n,t}^p, \mathbf{a}_{n,t}^f, \forall n \in \mathcal{N}] \tag{4.3.3}$$

where $\mathbf{a}_t^p = [p_{n,k,t}, n \in \mathcal{N}, k \in \mathcal{K}]$ and $\mathbf{a}_t^f = [f_{n,t}, n \in \mathcal{N}]$, respectively, are the actions of transmission power and computation resource at time slot t, and $\mathbf{a}_{n,t}^p = [p_{n,k,t}, k \in \mathcal{K}]$ and $\mathbf{a}_{n,t}^f = [f_{n,t}]$, respectively, are the actions of transmission power and computation resource for PS-DT pair n at time slot t.

4.3.3 State transitions

To find the state transitions from time slot t to time slot $t+1$, we find the transitions of each element in the state.

For the channel gains, the transition probabilities between different states follow

the FSMC model and are independent of the other elements in the state space.

The transitions of the remaining number of bits to be transmitted from PS n to DT n are given as

$$b_{n,t+1} = \begin{cases} B_{\max,n}, & \text{if } t = jW_n + t_{s,n} - 1, \\ (b_{n,t} - r_{n,t}(\mathbf{a}_t^p))^+, & \text{otherwise} \end{cases} \tag{4.3.4}$$

where j is any positive integer and $r_{n,t}(\mathbf{a}_t^p)$ is the total amount of data that PS n can transmit when the transmission power of all the PSs is given by $\mathbf{a}_t^p$ and can be calculated using (4.2.1).

The transitions of the remaining number of bits to be transmitted from PS n to DT n are given as

$$c_{n,t+1} = \begin{cases} C_{\max,n}, & \text{if } b_{n,t} > 0, \\ (c_{n,t} - f_{n,t})^+, & \text{otherwise} \end{cases} \tag{4.3.5}$$

The AoI at the DT keeps increasing with a slope of 1, i.e., increasing by one time slot in each time slot, except upon the completion of a new update when it drops to the amount of elapsed time since the start transmission time of the current update.

$$o_{n,t+1} = \begin{cases} (t + 1 - t_{s,n}) \mod W_n, & \text{if } b_{n,t} = 0, 0 < c_{n,t} \leq f_{n,t} \\ o_{n,t} + 1, & \text{otherwise.} \end{cases} \tag{4.3.6}$$

The complexity for solving the MDP problem increases exponentially as the number of PS-DT pairs, the number of channels and channel states, and other system parameters increase due to the exponential increase of the state and action spaces. Next, we seek

solution methods with lower complexity.

4.3.4 Reformulation for reduced complexity

Based on (4.3.1) and (4.3.2), the total number of states in the state space is $[H^K O_{\max}(B_{\max}+1)(C_{\max}+1)]^N$, where we assume $o_{n,t} \leq O_{\max}$, $b_{n,t} \leq B_{\max}$, and $c_{n,t} \leq C_{\max}$ for all $n \in \mathcal{N}$ and $t > 0$. The size of the state space can be prohibitively large for solving the MDP using most practical algorithms.

However, we notice that the changes to $b_{n,t}$ and $c_{n,t}$ represent the two stages of the synchronization update for the n-th PS-DT pair. In stage one, $b_{n,t}$ decreases from $B_{\max,n}$ to 0 while $c_{n,t}$ keeps constant at $C_{\max,n}$; and in stage two, $c_{n,t}$ decreases from $C_{\max,n}$ to 0 while $b_{n,t}$ is fixed at 0. Therefore, we define a new state element, remaining update load, to combine the two values as

$$u_{n,t} = b_{n,t} + c_{n,t} \tag{4.3.7}$$

with $u_{n,t} \in [0, U_{\max}]$ and $U_{\max} = B_{\max,n} + C_{\max,n}$. When $C_{\max,n} \leq u_{n,t} < U_{\max,n}$, the update for the n-th PS-DT is in stage one; and when $0 < u_{n,t} < C_{\max,n}$ the update is in stage two. The state in (4.3.2) is redefined as

$$\mathbf{s_{n,t}} = [\mathbf{h_{n,t}}, o_{n,t}, u_{n,t}] \tag{4.3.8}$$

Combining the state elements $b_{n,t}$ and $c_{n,t}$ into $u_{n,t}$ helps reduce the state space of the MDP. The size of the state space becomes $[H^K O_{\max}(B_{\max} + C_{\max} + 1)]^N$.

Accordingly, the transitions of the remaining update load are given as

$$u_{n,t+1} = \begin{cases} U_{\max,n}, & \text{if } t = jW_n + t_{s,n} - 1, \\ (u_{n,t} - r_{n,t}(\mathbf{a}_t^p))^+, & \text{if } C_{\max,n} < u_{n,t} \leq U_{\max,n}, \\ & \text{and } t \neq jW_n + t_{s,n} - 1, \\ (u_{n,t} - f_{n,t})^+, & \text{if } 0 < u_{n,t} \leq C_{\max,n}, \\ & \text{and } t \neq jW_n + t_{s,n} - 1, \\ 0, & \text{otherwise} \end{cases} \tag{4.3.9}$$

The four different cases in (4.3.9) represent the remaining update load at the initial time, time in stage one, time in stage 2, and time when the current update is completed.

From (4.3.3) it is seen that for each PS-DT pair, the action at time t includes two components, the transmission power of the PS at all channels and the computation resource at the ES. When we consider the power and computation allocations as two consecutive stages, we only need to make the decisions for either transmission power or computation, but not both. The two-stage decision making for the nth PS-DT pair is shown in Algorithm 3. In stage one, i.e., $C_{\max,n} < u_{n,t} \leq U_{\max,n}$, the PS is transmitting state data to the ES, the decisions should be made for the transmission power only, and the computation resource required in this stage is zero. In stage two, i.e., $0 < u_{n,t} \leq C_{\max,n}$, the required transmission power for the PS is zero since the data transmission is complete, and the amount of computation resource is to be determined. When the ES has finished processing the data for the PS-DT update, i.e., $u_{n,t} = 0$, both the PS transmission power and the amount of allocated computation resource are zero.

Algorithm 3 Two-Stage decisions for nth PS-DT pair at time t

if $C_{\max,n} < u_{n,t} \leq U_{\max,n}$ **then**
    Determine $p_{n,k,t}, \forall k \in K$
    $f_{n,t} = 0$
else if $0 < u_{n,t} \leq C_{\max,n}$ **then**
    $p_{n,k,t} = 0, \forall k \in K$
    Determine $f_{n,t}$
else
    $p_{n,k,t} = 0, \forall k \in K$
    $f_{n,t} = 0$
end if
$\mathbf{a}^p_{n,t} \leftarrow [p_{n,k,t}, \forall k \in K]$ and $\mathbf{a}^f_{n,t} = [f_{n,t}]$

Next, we design the two-stage resource allocation decisions based on the different natures of the communication and computation resource. For making the transmission power decisions in stage one, the achievable data transmission rates for individual PSs are not only determined by the transmission power of all the PSs, but also the time-varying channel gains and the mutual interference conditions. On the other hand, for stage two, the total amount of computation resource is fixed and the computation of all the DTs is performed at the ES. Therefore, stage-two decisions are well-suited for implementation through a centralized algorithm at the ES, whereas stage-one decisions are better aligned with a distributed or hybrid algorithm involving both the PSs and the ES.

Based on this, we first design deterministic algorithms to make centralized decisions at the ES for allocating computation resources in stage two. Subsequently, we develop a reinforcement learning (RL) method to make power allocation decisions for stage one. The centralized decisions for stage two ensure that the hard constraint in (4.2.5) is always satisfied, whereas the RL-based solution for stage one does not consistently guarantee that the AoI remains below the maximum threshold.

4.4 Stage-Two Decisions

We propose three algorithms, Algorithms 4, 5, and 6, to determine the computation resource allocation. In each algorithm, the ES iteratively selects the next PS-DT pair based on a specific criterion and allocates computation resources to it. The allocation is determined as either the amount required to complete the computation in the current time slot or the total remaining available computation resources, whichever is smaller. This process is repeated until either all remaining computation loads are processed within the current time slot or the available computation resources are fully allocated. In all three algorithms, $\mathcal{E} = \{n \in \mathcal{N}, 0 < u_{n,t} < C_{\max,n}\}$.

In Algorithm 4, the criterion for selecting the next DT is based on the AoI deviation, which is the difference between the maximum AoI ($D_{\max,m}$) and the current AoI ($o_{m,t}$). The DT with the minimum AoI deviation is selected to have a high priority to use the remaining amount of CPU resources.

In Algorithm 5, in addition to considering the AoI deviation, $u_{m,t}/\Delta f$, which is the amount of time to finish the remaining computation based on the currently available ES computation resource for the mth PS-DT update, is also considered. Define $D_{\max,m} - o_{m,t} - \frac{u_{m,t}}{\Delta f}$ as the residual margin of the AoI for DT m, which reflects the remaining "buffer" before the AoI exceeds the allowable maximum when considering in the remaining processing delay.

In Algorithm 6, a higher priority is given to the DT that is more urgent to finish the current update. An urgency factor is defined as $\frac{u_{m,t}}{D_{\max,m}-o_{m,t}}$, if $D_{\max,m} \neq o_{m,t}$. The urgency factor reflects the relationship between the remaining processing load and the time margin available before the AoI exceeds the limit. The urgency factor is defined as $u_{m,t}$ if $D_{\max,m} = o_{m,t}$. Computation resources are first allocated to the DTs

whose AoI already exceeds their maximum tolerable values, i.e., DTs in set $\mathcal{D}_1$. After all these DTs have received the computation resources to complete their computation within the current time slot, if there is still remaining computation resource, the DTs with the current AoI equal to their $D_{\max,m}$ are considered, i.e., DTs in set $\mathcal{D}_2$, followed by the the DTs with the current AoI small than their $D_{\max,m}$, i.e., DTs in set $\mathcal{D}_3$. Within each set, a higher priority is given to the DTs with a larger urgency factor.

Algorithm 4 Stage-two decisions based on AoI deviation

Input: Execution buffer $\mathcal{E}$, total CPU capacity f_C
Output: Allocated CPU resources $[f_{n,t}, \forall n \in \mathcal{N}]$
Initialize $f_{n,t} = 0\ \forall n \in \mathcal{N}$, $\Delta f = f_C$
while $\Delta f > 0$ **and** $\mathcal{E} \neq \emptyset$ **do**
    $n = \arg\min_{m \in \mathcal{E}} (D_{\max,m} - o_{m,t})$
    $f_{n,t} = \min(u_{n,t}, \Delta f)$
    $\Delta f \leftarrow \Delta f - f_{n,t}$
    $\mathcal{E} \leftarrow \mathcal{E} \setminus \{n\}$
end while
Return $[f_{n,t}, \forall n \in \mathcal{N}]$

Algorithm 5 Stage-two decisions based on AoI residual margin

Input: Execution buffer $\mathcal{E}$, total CPU capacity f_C
Output: Allocated CPU resources $[f_{n,t}, \forall n \in \mathcal{N}]$
Initialize $f_{n,t} = 0\ \forall n \in \mathcal{N}$, $\Delta f = f_C$
while $\Delta f > 0$ **and** $\mathcal{E} \neq \emptyset$ **do**
    $n = \arg\min_{m \in \mathcal{E}} \left(D_{\max,m} - o_{m,t} - \frac{u_{m,t}}{\Delta f}\right)$
    $f_{n,t} = \min(u_{n,t}, \Delta f)$
    $\Delta f \leftarrow \Delta f - f_{n,t}$
    $\mathcal{E} \leftarrow \mathcal{E} \setminus \{n\}$
end while
Return $[f_{n,t}, \forall n \in \mathcal{N}]$

Algorithm 6 Stage-two decisions based on urgency factor

Input: Execution buffer $\mathcal{E}$, total CPU capacity f_C
Output: Allocated CPU resources $[f_{n,t}, \forall n \in \mathcal{N}]$
Initialize $f_{n,t} = 0 \ \forall n \in \mathcal{N}$
while $\Delta f > 0$ **and** $\mathcal{E} \neq \emptyset$ **do**
    $\mathcal{D}_1 = \{m \in \mathcal{E} \mid o_{m,t} > D_{\max,m}\}$
    $\mathcal{D}_2 = \{m \in \mathcal{E} \mid o_{m,t} = D_{\max,m}\}$
    $\mathcal{D}_3 = \{m \in \mathcal{E} \mid o_{m,t} < D_{\max,m}\}$
    if $\mathcal{D}_1 \neq \emptyset$ **then**
        $n = \arg\max_{m \in \mathcal{S}_1} \left(\frac{u_{m,t}}{D_{\max,m} - o_{m,t}} \right)$
    else if $\mathcal{D}_2 \neq \emptyset$ **then**
        $n = \arg\max_{m \in \mathcal{S}_2} (u_{m,t})$
    else if $\mathcal{D}_3 \neq \emptyset$ **then**
        $n = \arg\max_{m \in \mathcal{S}_3} \left(\frac{u_{m,t}}{D_{\max,m} - o_{m,t}} \right)$
    end if
    $f_{n,t} = \min(u_{n,t}, \Delta f)$
    $\Delta f \leftarrow \Delta f - f_{n,t}$
    $\mathcal{E} \leftarrow \mathcal{E} \setminus \{n\}$
end while
Return $[f_{n,t}, \forall n \in \mathcal{N}]$

4.5 Stage-One Decisions

At any given time, joint power allocation decisions should be made for all PSs in stage one, and the sequential decisions along the time should also take into consideration the AoI targets and the average power consumption of the PSs. Therefore, the solution should be scalable as the number of PS-DT pairs, the number of channels and channel states increase, and the algorithms should be of low complexity. For this reason, we adopt a multi-agent reinforcement learning approach. There is one local agent at each PS and one central agent at the ES. The local agents at individual PSs make decisions on the their own transmission power, and the central agent interacts with the local agents to collect the interference and competition conditions for the shared

resources.

4.5.1 Choices of RL algorithms

Selecting RL algorithms for the stage-one decisions involves several key considerations. First, since the PS transmission power is a continuous variable, the chosen algorithms must be capable of handling continuous action spaces. Second, the sequential decisions over time, including power allocation in stage one and computation resource allocation in stage two, jointly influence the PS-DT update completion time and the resulting AoI dynamics. Consequently, the agent cannot foresee early on whether the AoI at the DT will eventually exceed $D_{\max,n}$ in later time slots. Therefore, the algorithms must effectively manage the challenge of delayed rewards to ensure reliable performance.

We employed the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [11], which is well-suited for handling continuous action spaces and addresses the limitations of the traditional Deep Deterministic Policy Gradient (DDPG) algorithm [22]. DDPG is prone to overestimation bias and unstable training dynamics when dealing with continuous action spaces. Unlike DDPG, which utilizes a single critic network to estimate the action-value function, TD3 employs twin critic networks. By taking the minimum of the two Q-value estimates, TD3 effectively reduces overestimation bias, leading to more accurate value function approximations and improved training stability.

Additionally, TD3 incorporates delayed policy updates and target policy smoothing regularization, both of which enhance training stability and reduce the exploitation of noisy value estimates. Delayed policy updates allow the algorithm to update

the policy network less frequently, giving the critic networks adequate time to converge. This ensures more accurate evaluations of long-term expected returns, even in the absence of immediate feedback. By leveraging past experiences—where the outcomes of actions are realized after a delay—TD3 enhances the agent's decision-making ability, enabling it to optimize performance over time more effectively. These features make TD3 particularly well-suited for environments with complex, delayed reward structures.

4.5.2 CTDE framework

To implement the TD3 algorithm in our multi-agent setting, we adopt a Centralized Training with Decentralized Execution (CTDE) framework [24], which allows agents located at individual PSs to leverage global information during training while maintaining independence during execution. This approach enhances coordination and learning efficiency among the local agents. The training procedure for our proposed method is given in Algorithms 7 and 8.

As illustrated in Figure 4.1, there is a local agent at each PS that includes one actor network, denoted as μ_n^{θ}, and one target actor network, $\mu_n^{\theta'}$, where n represents the nth PS and θ and θ' are the parameters of the corresponding networks. There are two critic networks at the ES, $Q_1^{\phi_1}$ and $Q_2^{\phi_2}$ and two target critic networks $Q_1^{\phi_1'}$ and $Q_2^{\phi_2'}$, where ϕ_1, ϕ_2, ϕ_1' and ϕ_2' represent the parameters of the corresponding networks. These networks work together to refine decision-making. The actor networks at each PS map observations into action decisions, while the critic networks at the ES evaluate these actions by producing value estimates. To enhance training stability, each actor and critic network is paired with its corresponding target network. These target

networks are delayed copies that offer more consistent reference values, which ensures smoother and more reliable updates to the parameters. For updating the policy of the local agents, the parameters of the actor and target actor networks at all local agents are sent to the central agent for updates. Leveraging the CTDE framework, the centralized critics maintained at the ES are shared across all the local agents, allowing for the incorporation of global information during training.

As outlined in Algorithm 7, each PS ps initializes its state $\mathbf{s}_{ps,1}$ and the actor network parameter θ_{ps}. At each time slot, the action is determined based on the current stage of the PS-DT update. If the update is in the transmission stage (i.e., stage one), the PS determines its transmission power based on its current state and learned policy plus exploration noise ξ (line 5). If the update is in the computation stage (i.e., stage two), the computation resource is allocated by using the algorithms designed in Section 4.4 (line 7). These actions are executed in the environment, and the transitions are stored in the replay buffer $\mathcal{B}$ (lines 11-13). After each episode of length $L_{\max}$, the states of all PSs are reinitialized (lines 14-17), where $L_{\max}$ is chosen to be longer than the PS-DT update period to ensure sufficient data accumulation during policy training, enabling agents to learn from a diverse range of observations. Each PS's policy is trained periodically with a training period I_{train}. At the end of each training period, the actor networks for each PS and the centralized critic networks are updated using mini-batches sampled from the replay buffer (lines 18-20) and the details of these updates are given in Algorithm 8 and explained below.

In Algorithm 8, states and actions from all the local agents are sampled (line 3) for Q-value calculations, enabling the centralized critics to have a comprehensive view of the system. Then, for each PS i in the set of all PSs $\mathcal{N}$, the algorithm checks

Table 4.1: Notation for TD3

Variable	**Definition**
$T_{\max}$	Maximum simulation steps
$L_{\max}$	Episode length
I_{train}	Training frequency
ξ	Exploration noise distribution
C_{ep}	Episode iteration
μ_n^{θ}	Actor network for nth PS-DT
$\mu_n^{\theta'_n}$	Target Actor Network for nth PS-DT
$\mathcal{B}$	Replay buffer
$Q_1^{\phi_1}$, $Q_2^{\phi_2}$	Central critic networks
$Q_1^{\phi'}$, $Q_2^{\phi'}$	Target central critic networks
ω	Actor Update frequency
γ	Discount factor
β	Batch size
δ	Target network update rate

whether the remaining computation load u_{i,t_j+1} is within the maximum computation capacity $C_{\max,i}$ (lines 5). If it is, the algorithm allocates computation resources using one of the deterministic algorithms (Algorithms 4, 5, or 6) to ensure efficient processing. If the remaining computation load exceeds $C_{\max,i}$, the algorithm determines the transmission power p_{i,k,t_j+1} for each channel k by querying the target actor network $\mu_i^{\theta'_i}(\mathbf{s}_{i,t_j+1})$ and adding exploration noise ξ to encourage exploration of new power allocations (line 8). These target actions for transmission power and computation resources are then aggregated into comprehensive action vectors $\mathbf{a}_{t_j+1}^p$ and $\mathbf{a}_{t_j+1}^f$, respectively, forming the complete action vector $\mathbf{a}_{t_j+1}$ (line 13). When calculating Q-values for updating the actor and critic networks (lines 14 and 18), only the first component of the action space, which is the transmission power allocation, is utilized. The second component, representing the allocated computation resource $f_{n,t}$ from the ES, is assigned deterministically by the ES and is not learned by the policy (line 6).

This deterministic allocation influences the environment by determining the next

Algorithm 7 Training Procedure of Problem with TD3-CTDE

Require: $T_{\max}$, $L_{\max}$, I_{train}, $E_{\max}$, ξ
Ensure: Trained policies for all PSs
1: Initialize $\mathbf{s}_{ps,1}$ and θ_{ps} $\forall ps \in \mathcal{N}$, C_{ep}, $\mathcal{B}$, ϕ_1, ϕ_2,
2: **for** $t = 1$ to $T_{\max}$ **do**
3: **for** $ps \in \mathcal{N}$ **do**
4: **if** $C_{\max,ps} < u_{ps,t} \leq U_{\max,ps}$ **then**
5: $[p_{ps,k,t}, \forall k \in \mathcal{K}] \leftarrow \mu_{ps}^{\theta_{ps}}(\mathbf{s}_{ps,t}) + \xi$
6: **else**
7: $f_{ps,t} \leftarrow$ return of Algorithms 4, 5, or 6
8: **end if**
9: **end for**
10: $\mathbf{a}_t^p = [p_{ps,k,t}, \forall k \in \mathcal{K}, ps \in \mathcal{N}]$, $\mathbf{a}_t^f = [f_{ps,t}, ps \in \mathcal{N}]$,
11: Execute actions $\mathbf{a}_t = [\mathbf{a}_t^p, \mathbf{a}_t^f]$
12: Observe next state $\mathbf{s}_{t+1}$ and reward $\mathbf{r}_t$
13: Store transition $(\mathbf{s}_{ps,t}, \mathbf{a}_{ps,t}, \mathbf{s}_{ps,t+1}, \mathbf{r}_{ps,t})$ in $\mathcal{B}, \forall ps \in \mathcal{N}$
14: **if** $C_{\text{ep}} = L_{\max}$ **then**
15: Reinitialize $\mathbf{s}_{ps,t}$ for all $ps \in \mathcal{N}$
16: $C_{\text{ep}} \leftarrow 0$
17: **end if**
18: **if** $t \bmod I_{\text{train}} = 0$ **then**
19: Update θ_{ps}, $\forall ps \in \mathcal{N}$, and ϕ_1, ϕ_2 using algorithm 8
20: **end if**
21: $C_{\text{ep}} \leftarrow C_{\text{ep}} + 1$
22: **end for**

state but does not factor into the policy optimization. Consequently, the training process focuses solely on optimizing transmission power allocations within an environment where computation resources are predefined by the ES. To simplify the policy and reduce decision complexity, actions related to computation states are set to zero during Q-value calculations. This allows the agent to concentrate on scenarios that require transmission power allocation. The algorithm employs two critic networks to estimate Q-values, mitigating overestimation bias by considering the minimum of the two critics (line 14). This dual-critic approach enhances the stability and reliability

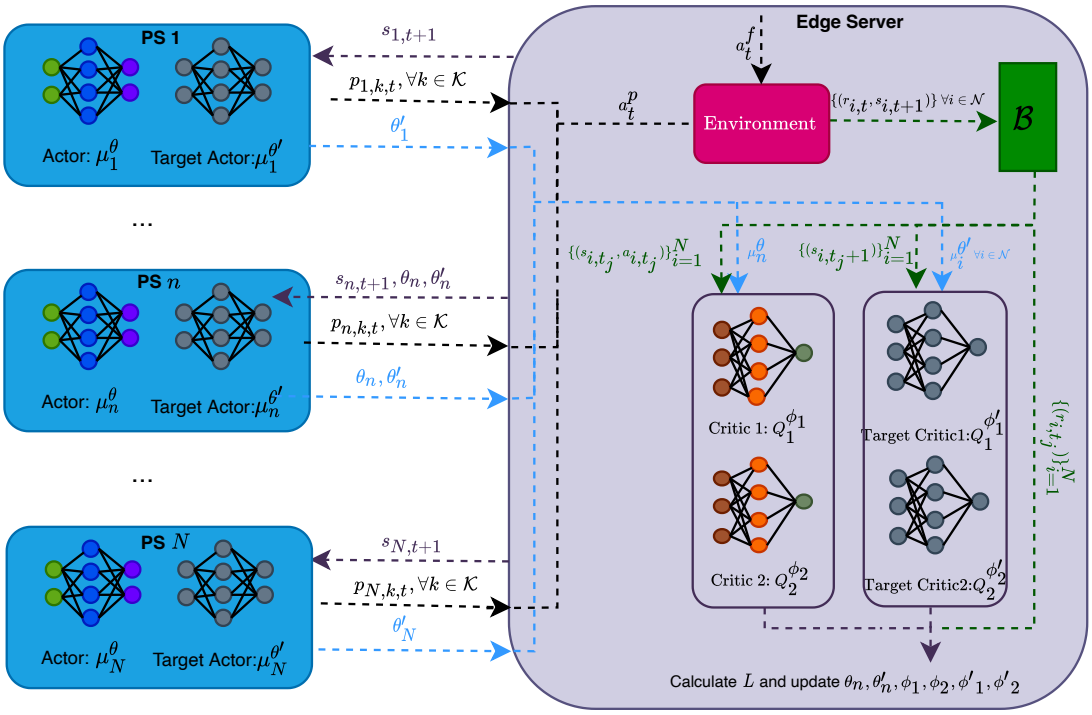

Figure 4.1: Structure of TD3-CTDE algorithm

of policy updates by reducing the likelihood of overly optimistic value estimates. Additionally, the actor network is updated less frequently than the critic networks, as specified by the actor update frequency parameter (ω) (line 19). At the end of each training iteration, the trained actor networks are sent back to PSn for decentralized execution. This iterative process continues until the maximum number of simulation steps ($T_{\max}$) is reached. Through this process, each PS learns an optimal power scheduling policy in a dynamic, multi-agent environment.

4.5.3 Reward function

For the agent located at PS n, referred to as the nth agent, its reward function at time slot t is defined as

$$r_{n,t} = -\sum_{k} p_{n,k,t} - \eta \sum_{\forall i \in \mathcal{N}} \lambda_{i,t}(o_{i,t} - D_{\max,i}), \tag{4.5.1}$$

where η is a coefficient that balances the weights between the transmission power and the AoI violation, and $\lambda_{i,t}$ is given as

$$\lambda_{i,t} = \begin{cases} 1, & \text{if } i = n \\ 1 - 0.9 \cdot \mathcal{I}_{\{o_{i,t} < D_{\max,i}\}}, & \text{otherwise} \end{cases} \tag{4.5.2}$$

with $\mathcal{I}_{\{x\}}$ as the indicator function which is 1 when the relationship x is true and 0 otherwise.

The reward function takes the following factors into consideration. First, the agent tries to reduce its own total transmission power at all the channels. This is consistent with the objective of minimizing the average transmission power of the PSs. Second, when $i = n$, $\lambda_{i,t} = 1$, a smaller $o_{n,t}$ helps the agent achieve a higher reward. In addition, the agent also coordinates with the agents located at other PSs due to the shared channel and computation resources. The level of coordination depends on the AoI conditions of the other DTs. More specifically, for $i \neq n$, if $o_{i,t} < D_{\max,i}$, then $\lambda_{i,t} = 0.1$, in which case agent n can take more selfish actions; and when $o_{i,t} \geq D_{\max,i}$, $\lambda_{i,t} = 1$, and agent n's action is more influenced by the AoI at other DTs.

Table 4.2: Default simulation Parameters

Parameter	**Value**
N	6
B	5 MHz
K	1
$C_{\max,n}$	100 M cycles
$B_{\max,n}$	15 M bits
$D_{\max,n}$	120 ms
$P_{\max,n}$	1 W
f_C	50
$T_{\max}$	10^6
$L_{\max}$	200
I_{train}	5
β	256
δ	0.005
γ	0.99
η	0.1
ω	5
ξ	0.2

4.6 Simulation Results

In this section, we present computer simulation results to demonstrate the performance of the proposed algorithms. For the FSMC channels, each channel has two states, G and B. The link gains of the G and B states are -74 dB and -94 dB, respectively, the transition probability from the G state to the B state is 0.25 and that from the B state to the G state is 0.75. Default simulation parameters are listed in Table **??**, where the values for channel bandwidth, maximum transmission power, and total amounts of transmitted data computation load for each PS-DT update were based on the settings in [6], [5], [10].

We first compare the three algorithms proposed in Section 4.4 for making stage-two decisions, when each works together with the TD3-CTDE algorithms proposed in Section 4.5 for stage-one decisions. Figures 4.2 and 4.3 show that in general, both

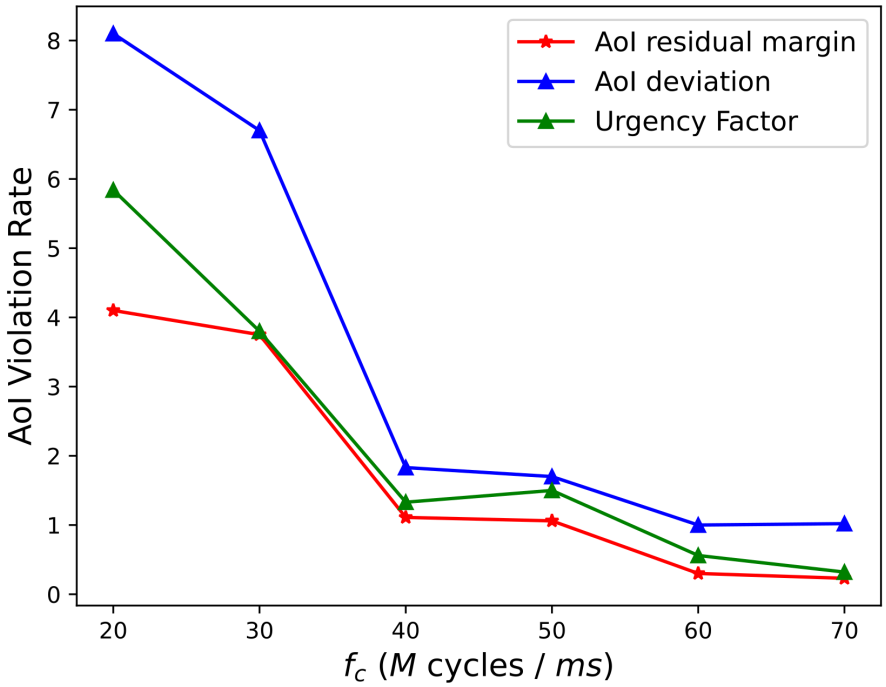

Figure 4.2: Comparison of stage-two algorithms: average AoI violation rate versus ES computation capacity, $C_{\max} = 100$ M cycles

the AoI violation rate and the average PS power consumption decrease with the ES computation capacity. The figures show that among the three computation resource allocation algorithms, the one based on the residual margins achieves both the lowest AoI violation rate and the lowest average PS power consumption. This is due to the fact that the residual margins take into consideration both the current AoI deviation and possible AoI increase, while each of the other algorithms considers only one of these two aspects. Therefore, for generating the remaining results, we will adopt the AoI Residual Margin algorithm to make the stage-two decisions.

It is also interesting to see from figure 4.3 that for all the three algorithms, the average PS transmission power keeps almost constant for a wide range of f_C values.

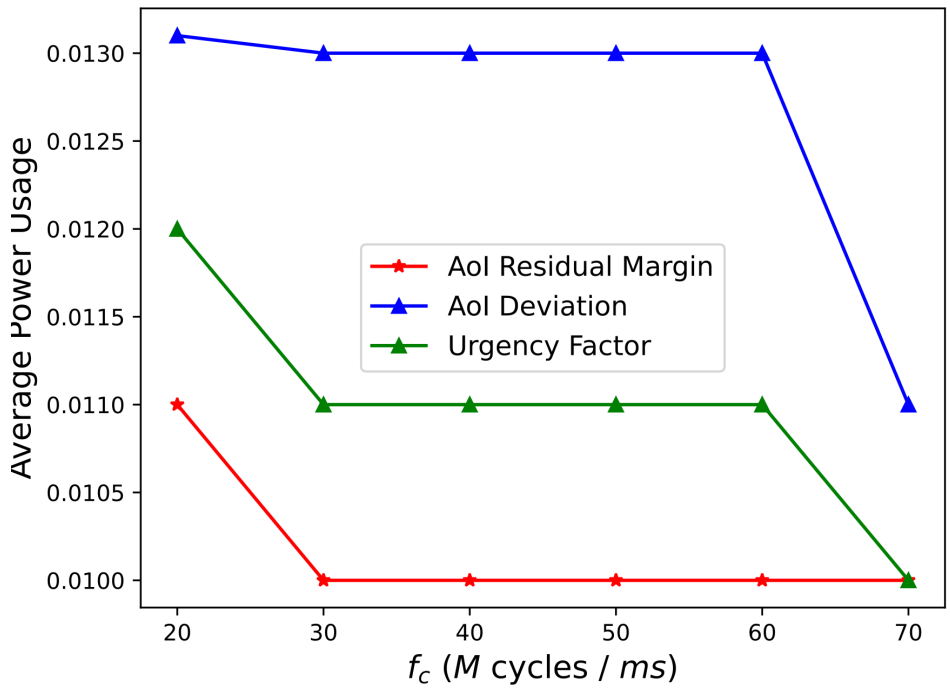

Figure 4.3: Comparison of stage-two algorithms: average power consumption versus ES computation capacity, $C_{\max} = 100$ M cycles

Since the power allocation is primarily determined in the first stage to ensure timely data delivery, faster processing at the second stage does not necessarily translate into lower transmission power. In other words, the two-stage decision-making approach inherently separates the communication and computation decisions, allowing the system to maintain stable power usage even as f_C varies. However, when the computation resource is insufficient and will likely cause longer delay in stage two, the stage-one decisions do allow higher transmission power in order to speed up the data transmission and reduce the overall synchronization update time. Furthermore, when the ES capacity is very high, the algorithm in stage-one effectively learns the short computation delay, which helps the PS reduce their transmission power while

still maintaining low AoI violation rate.

We further compare the three computation resource allocation algorithms changing the number of PSs and the results are shown in figures 4.4 and 4.5. The figures show that the AoI residual margin algorithm clearly stands out as it keeps both the average power usage and the AoI violation rate lower than the other two approaches. In comparison, the algorithms based on the urgency factor and the AoI deviation not only result in higher AoI violation rate but also much higher average power consumption. This indicates that by considering both the current AoI conditions and the potential future AoI changes, the AoI Residual Margin algorithm is the best in terms of both power efficiency of the PSs and the AoI at the DT.

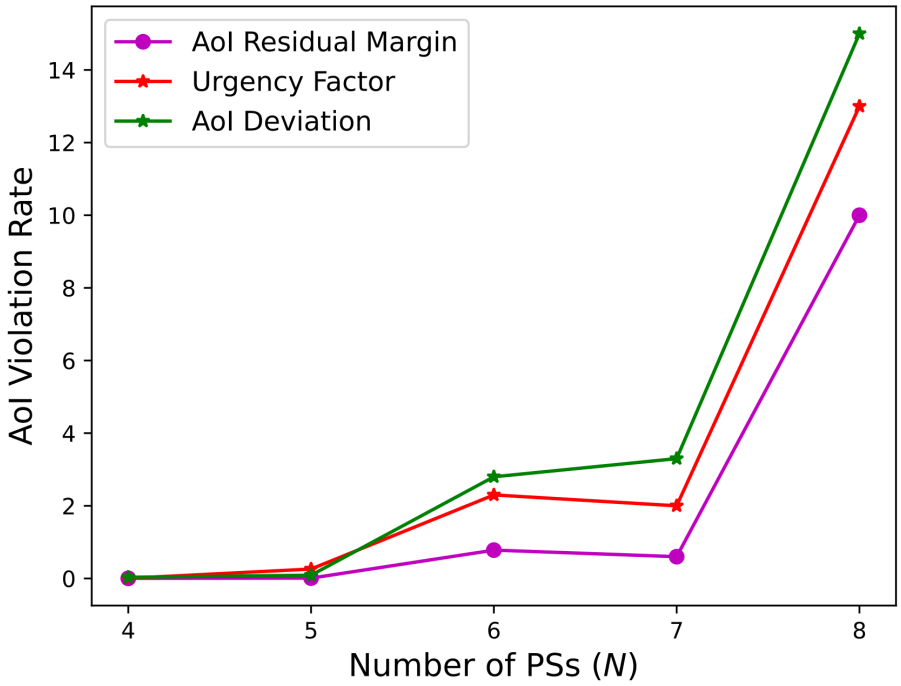

Figure 4.4: Comparison of stage-two algorithms: average AoI violation rate versus number of PSs, $C_{\max} = 100$ M cycles

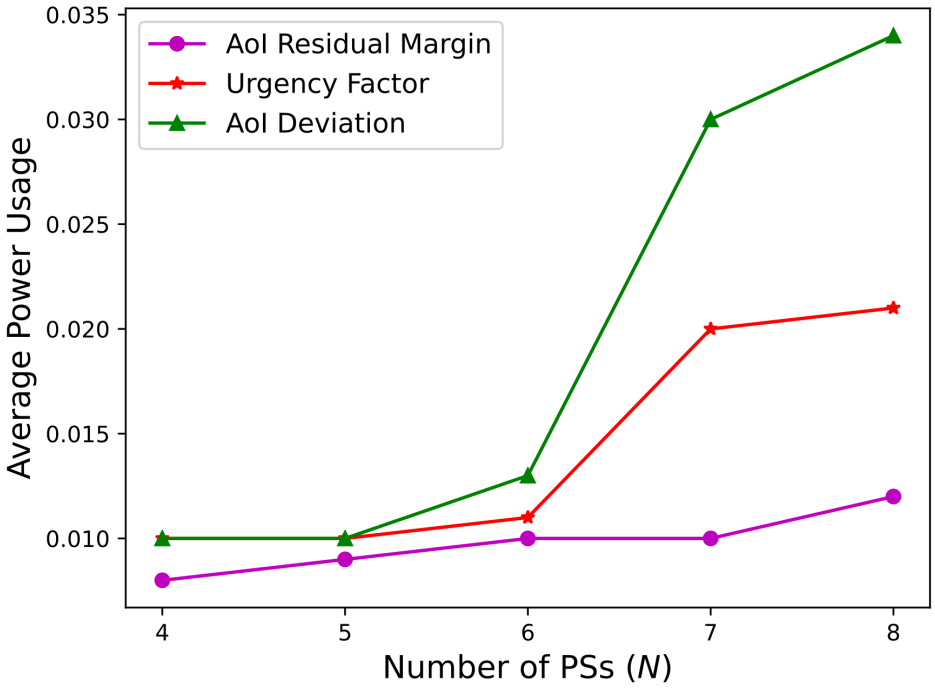

Figure 4.5: Comparison of stage-two algorithms: average power consumption versus number of PSs, $C_{\max} = 100$ M cycles

Next, we compare the proposed stage-one solution with three other power allocation solutions in the literature. The AoI Residual Margin algorithm is adopted for the stage-two resource allocations. The first one is the Multi-Agent Dueling Double Deep Q-Learning (MADDDQN) approach [42], [43], which is a fully decentralized multi-agent algorithm. In MADDDQN, each PS operates independently, making decisions without relying on a centralized critic network to evaluate the agents' actions. The action space is binary: for each channel, each PS either transmits at maximum power or refrains from transmitting entirely. The next solution is Time Division Channel Access (TDCA), where PSs in stage one take turns transmitting on a time-slot basis until all transmissions are completed. During its assigned time slot, each PS transmits

simultaneously on all channels using maximum transmission power. The last solution is based on a cooperative game designed in [19], and the objective is to encourage more PSs to transmit simultaneously and improve the total data transmission rate of all the PSs.

Figures 4.6 and 4.7, respectively, show the AoI violation rate and the average transmission power of the PSs. The figures show that comparing with the above three solution, the proposed TD3-CTDE solution leads to much lower average power consumption of the PSs while maintaining much lower AoI violation rate. The only exception is that when $N = 8$, the TDCA solution achieves a lower AoI violation rate than the proposed one, but the former requires much higher average transmission power.

The MADDDQN solution performs worse than our method in AoI violation rate and PS power consumption due to its lack of coordination mechanisms in a shared environment. In addition, its binary action space is neither effective in keeping the AoI below the limit nor efficient in power resource utilization. The Cooperative Game Theory approach considers only the immediate data transmission rates at the current time slot and overlooks the temporal dynamics of AoI and the future impact of the current actions. The TDCA solution achieves the lowest AoI violation rate due to interference-free transmissions, resulting in higher data rates. However, it inefficiently consumes power by requiring PSs to always transmit at maximum power. In contrast, our proposed solution dynamically allocates power to each PS in each time slot, optimizing power usage while accounting for AoI evolution over time. This approach achieves comparable performance in reducing AoI violations to TDCA while significantly lowering average power consumption.

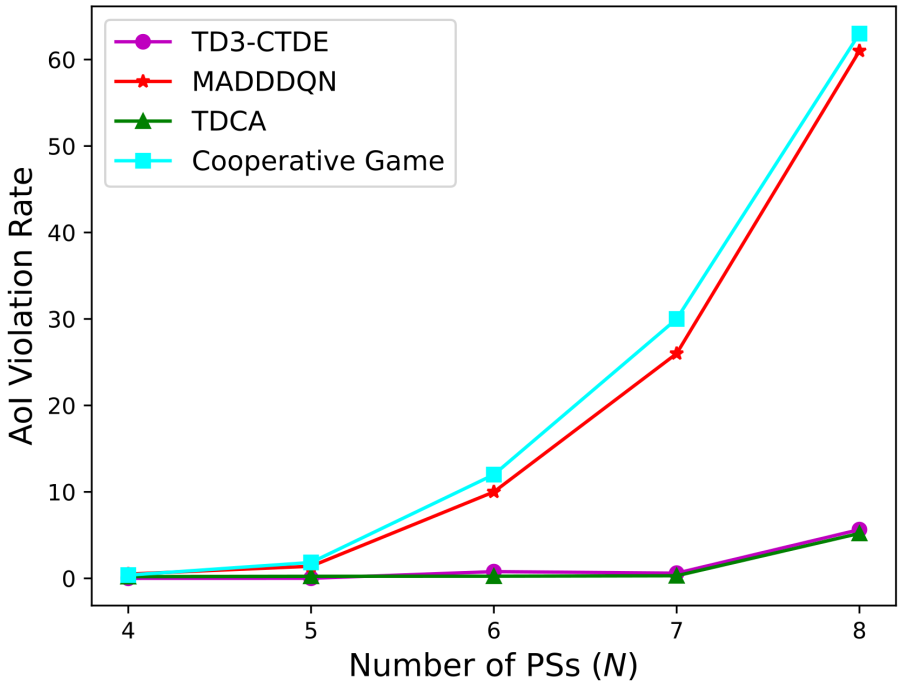

Figure 4.6: Comparison of stage-one algorithms: average AoI violation rate versus number of PSs

Figures 4.8 and 4.9 further compare the above solutions by varying the ES computation capacity. The figures show that the proposed solution achieves a slightly higher AoI violation rate than TDCA but significantly outperforms the other two solutions in this metric. Additionally, it demonstrates substantially lower average power consumption compared to the other solutions. This highlights the proposed method as the best tradeoff, effectively balancing low average PS power consumption with maintaining a low AoI violation rate. The results also reveal that while the AoI violation rate decreases markedly with increased ES computation capacity, the average PS power consumption is slightly decreased. This demonstrates that the joint effect of communication and computation resource on the DT quality. However,

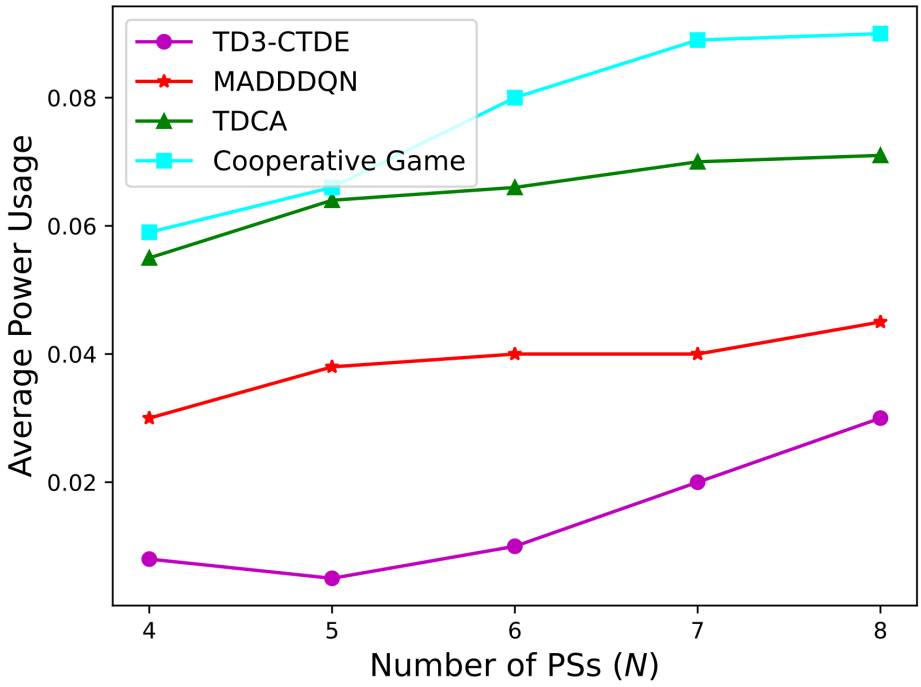

Figure 4.7: Comparison of stage-one algorithms: average power consumption versus number of PSs

the fact that the average PS power consumption is only slightly affected by the ES capacity further demonstrates the good performance of the proposed AoI Residual Margin algorithm for the stage-two decisions.

Next, we increase the total number of wireless transmission channels to $K = 3$. Figures 4.10 and 4.11, respectively, show the average AoI violation rate and power consumption of different solutions. Overall, the proposed TD3-CTDE solution and TDCA achieve much lower AoI violation rate than the other two solutions. When the number of PSs is large, TD3-CTDE results in slightly higher AoI violation rate than TDCA. However, TDCA consumes much higher PS power than TD3-CTDE. The Cooperative Game solution achieves apporoximately the same AoI violation rate as

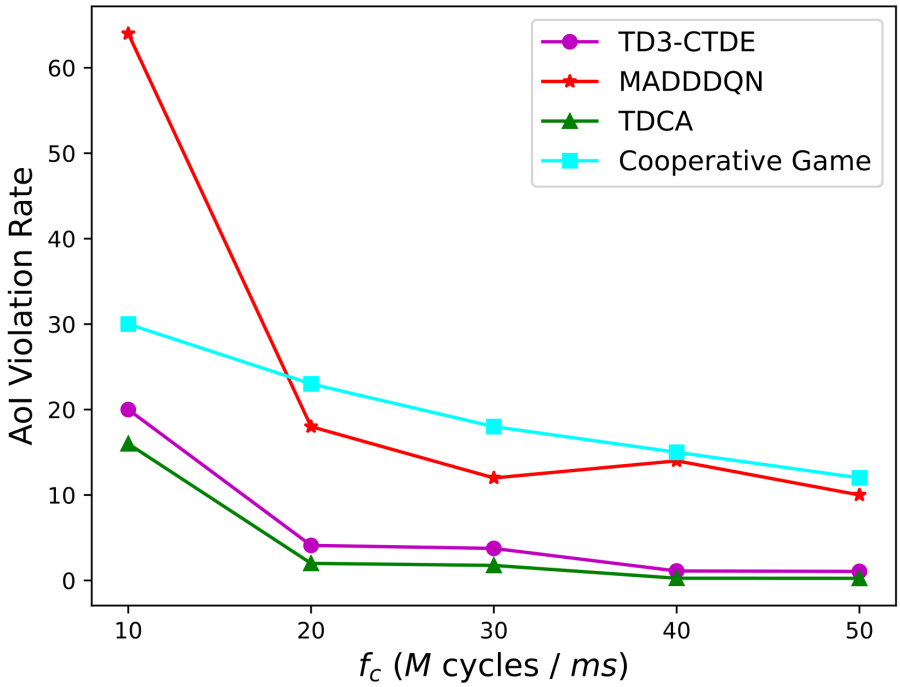

Figure 4.8: Comparison of stage-one algorithms: average AoI violation rate versus ES computation capacity

TD3-CTDE and DDDQN when the number of PSs is small, but the AoI violation rate increases significantly when the number of PSs is larger than 6. In addition, both the Cooperative Game and DDDQN solutions result in much higher power consumption of the PSs. Overall, the observations are consistent with that from figures 4.6 and 4.7.

4.7 Summary

We have studied the problem of network resource allocation for multiple physical systems updating their DTs via shared communication channels and a shared ES. Our proposed solution involves a two-stage process to optimize both the transmission

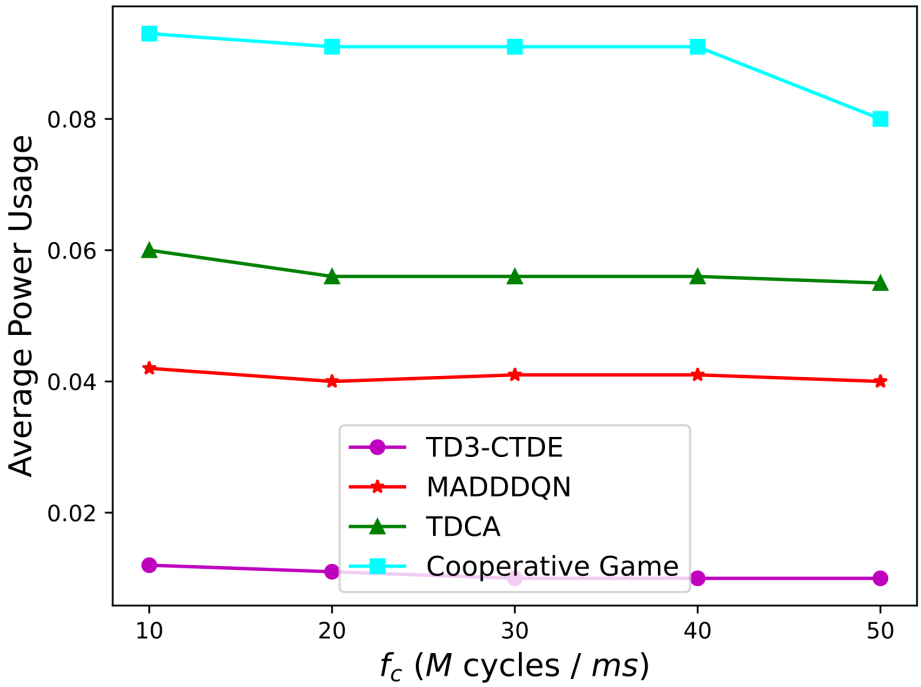

Figure 4.9: Comparison of stage-one algorithms: average power consumption versus ES computation capacity

power of individual physical systems and the computation resources at the ES. In stage one, a TD3-based multi-agent reinforcement learning algorithm is employed to allocate transmission power. This approach makes sequential decisions by considering the evolution of the AoI and ensuring compliance with AoI limits. In stage two, a set of centralized and deterministic algorithms is used to allocate computation resources among DTs efficiently. Compared to existing methods, our proposed solution strikes the optimal balance between minimizing the average transmission power of physical systems and maintaining the AoI at the DTs within acceptable limits, thereby enhancing overall system performance.

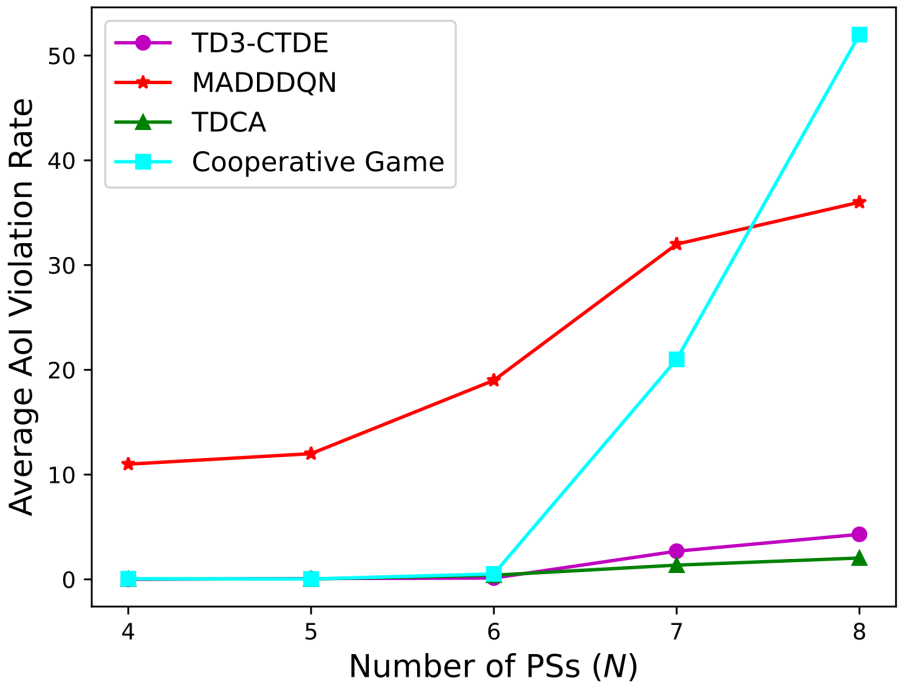

Figure 4.10: Comparison of stage-one algorithms: average AoI Violation Rate versus total number of PSs: $K = 3$, $B_{\max,n} = 60$ M bits

Algorithm 8 TD3-CTDE Algorithm for PS n

Require: θ_n, θ'_n for $n \in \mathcal{N}$, ϕ_1, ϕ_2, $\phi_1^{'}$, $\phi_2^{'}$, $\mathcal{B}$, ω, γ, β, δ, ξ
Ensure: θ_n, $\theta_n^{'}$, ϕ_1, ϕ_2

for $iter = 1$ **to** Training Iterations **do**
  for $j = 1$ **to** β **do**
    Randomly sample a transition from the replay buffer:

$$(\mathbf{s}_{t_j}, \mathbf{a}_{t_j}, \mathbf{r}_{t_j}, \mathbf{s}_{t_j+1})$$

    for $i \in \mathcal{N}$ **do** ▷ Compute target actions
      if $u_{i,t_j+1} \leq C_{\max,i}$ **then**
        $f_{i,t_j+1} \leftarrow$ output of Algorithms 4, 5, or 6
      else
        $[p_{i,k,t_j+1}, \forall k \in \mathcal{K}] \leftarrow \mu_i^{\theta'_i}(\mathbf{s}_{i,t_j+1}) + \xi$
      end if
    end for
    $\mathbf{a}^p_{t_j+1} = [p_{i,k,t_j+1}, \forall k \in \mathcal{K}, \forall i \in \mathcal{N}]$,
    $\mathbf{a}^f_{t_j+1} = [f_{i,t_j+1}, \forall i \in \mathcal{N}]$
    $\mathbf{a}_{t_j+1} = [\mathbf{a}^p_{t_j+1}, \mathbf{a}^f_{t_j+1}]$
    Compute minimum Q-value from target critic networks

$$Q_{\min} = \min\left\{Q_1^{\phi'_1}\left(\mathbf{s}_{t_j+1}, \mathbf{a}^p_{t_j+1}\right), Q_2^{\phi'_2}\left(\mathbf{s}_{t_j+1}, \mathbf{a}^p_{t_j+1}\right)\right\}$$

    Compute the target value

$$y_{n,t_j} = \begin{cases} r_{n,t_j}, & \text{if } u_{n,t_j} > 0, u_{n,t_j+1} = 0 \\ r_{n,t_j} + \gamma(1 - d_{n,t_j})Q_{\min}, & \text{otherwise} \end{cases}$$

  end for
  Replace $\mathbf{a}^p_{n,t_j}$ with $\mathbf{a}^p_{n,t_j} = \mu_n^{\theta_n}(\mathbf{s}_{n,t_j})$
  Update ϕ_1 and ϕ_2 using gradient descent to minimize L

$$L = \frac{1}{\beta}\sum_{j=1}^{\beta}\left[\left(y_{n,t_j} - Q_1^{\phi_1}(\mathbf{s}_{t_j}, \mathbf{a}^p_{t_j})\right)^2 + \left(y_{n,t_j} - Q_2^{\phi_2}(\mathbf{s}_{t_j}, \mathbf{a}_{t_j})\right)^2\right]$$

  if $iter \mod \omega = 0$ **then**
    Update θ_n using the sampled policy gradient.
    Update target networks using soft updates:

$$\phi'_1 \leftarrow \delta\phi_1 + (1 - \delta)\phi'_1$$

$$\phi'_2 \leftarrow \delta\phi_2 + (1 - \delta)\phi'_2$$

$$\theta'_n \leftarrow \delta\theta_n + (1 - \delta)\theta'_n$$

  end if
end for

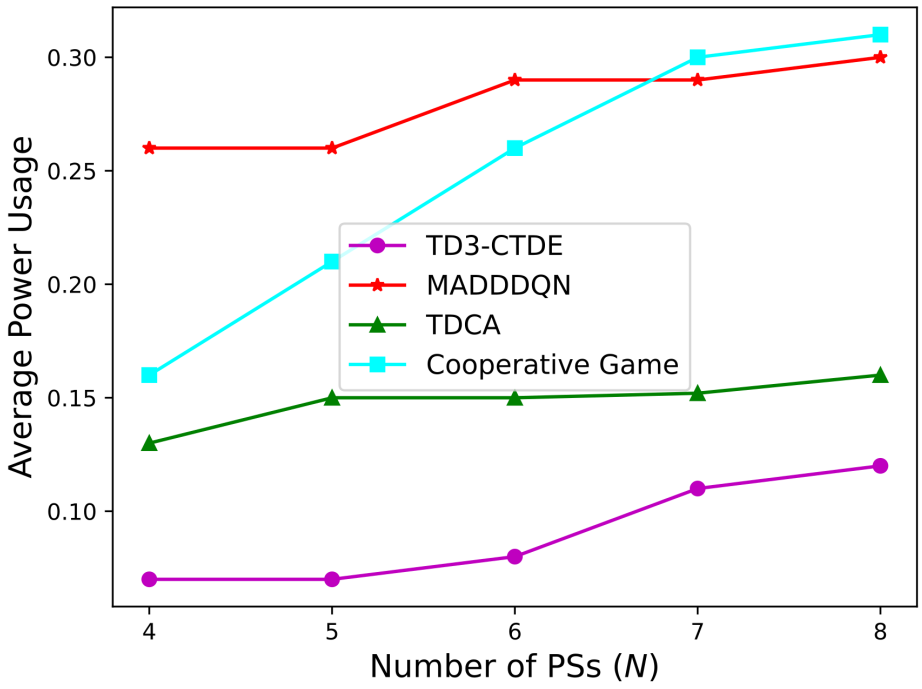

Figure 4.11: Comparison of stage-one algorithms: average Power Usage versus total number of PSs: $K = 3$, $B_{\max,n} = 60$ M bits

Chapter 5

Conclusion And Future Works

In conclusion, this thesis has advanced the understanding and practical handling of age of information (AoI) in digital twin systems by jointly considering communication and computation aspects under uncertain and dynamically changing conditions. We began by examining the problem of determining the optimal response time to application requests, taking into account the uncertainty in network performance and employing an integrated LSTM and Dueling Double Deep Q-learning approach. This method successfully reduced AoI compared to immediate response strategies, highlighting the value of intelligent, delay-aware decision making. Building on these insights, we then addressed more complex scenarios involving multiple physical systems, shared communication channels, and an edge server with limited computation resources. By proposing a two-stage solution—where a multi-agent, TD3-based reinforcement learning algorithm optimally allocates transmission power, and deterministic, centralized algorithms efficiently distribute computing capacity—we demonstrated the ability to achieve a desirable balance between minimal power consumption and maintaining AoI within specified limits. Together, these contributions illustrate

the importance of leveraging advanced RL techniques, predictive modeling, and integrated resource management strategies to ensure timely and energy-efficient updates from physical systems to their digital twins. Looking forward, this research can be extended by exploring multiple edge servers, jointly optimizing digital twin placement, and incorporating more complex constraints and uncertainty models to further enhance system responsiveness, scalability, and resilience.

Bibliography

[1] F. Akbarian, E. Fitzgerald, and M. Kihl. Synchronization in digital twins for industrial control systems. *arXiv preprint arXiv:2006.03447*, 2020.

[2] J. A. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] B. R. Barricelli, E. Casiraghi, and D. Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications. *IEEE Access*, 7:167653–167671, 2019. doi: 10.1109/ACCESS.2019.2953499.

[4] X. Cai, Z. Wang, S. Li, J. Pan, C. Li, and Y. Tai. Implementation of a virtual reality based digital-twin robotic minimally invasive surgery simulator. *Bioengineering*, 10(11):1302, 2023.

[5] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji. Computation offloading in beyond 5g networks: A distributed learning framework and applications. *IEEE Wireless Communications*, 28(2):56–62, 2021.

[6] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao. Deep reinforcement

learning-based dynamic resource management for mobile edge computing in industrial internet of things. *IEEE Transactions on Industrial Informatics*, 17(7): 4925–4934, 2020.

[7] S. Communications. Simplifying 5g with the network digital twin. white paper, Spirent Communications, 2019.

[8] Y. Dai and Y. Zhang. Adaptive digital twin for vehicular edge computing and networks. *Journal of Communications and Information Networks*, 7(1):48–59, 2022.

[9] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang. Deep reinforcement learning for stochastic computation offloading in digital twin networks. *IEEE Transactions on Industrial Informatics*, 17(7):4968–4977, 2020.

[10] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang. Deep reinforcement learning for stochastic computation offloading in digital twin networks. *IEEE Transactions on Industrial Informatics*, 17(7):4968–4977, 2020.

[11] S. Dankwa and W. Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *Proceedings of the 3rd international conference on vision, image and signal processing*, pages 1–5, 2019.

[12] G. De Giacomo, M. Favorito, F. Leotta, M. Mecella, and L. Silo. Digital twin composition in smart manufacturing via markov decision processes. *Computers in Industry*, 149:103916, 2023.

[13] J. Deng, Q. Zheng, G. Liu, J. Bai, K. Tian, C. Sun, Y. Yan, and Y. Liu. A digital twin approach for self-optimization of mobile networks. In *2021 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, 2021. doi: 10.1109/WCNCW49093.2021.9420037.

[14] M. Grieves. Digital twin: manufacturing excellence through virtual factory replication. *White paper*, 1(2014):1–7, 2014.

[15] R. Kaul, C. Ossai, A. R. M. Forkan, P. P. Jayaraman, J. Zelcer, S. Vaughan, and N. Wickramasinghe. The role of ai for developing digital twins in healthcare: The case of cancer care. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 13(1):e1480, 2023.

[16] B. Ketzler, V. Naserentin, F. Latino, C. Zangelidis, L. Thuvander, and A. Logg. Digital twins for cities: A state of the art review. *Built Environment*, 46(4): 547–573, 2020.

[17] M. Klimo, M. Kvassay, and N. Kvassayova. Digital twin and modelling a 3d human body in healthcare. In *2023 21st International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 307–312. IEEE, 2023.

[18] T. Lancewicki, A. Rosenberg, and Y. Mansour. Learning adversarial markov decision processes with delayed feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7281–7289, 2022.

[19] R. Langar, S. Secci, R. Boutaba, and G. Pujolle. An operations research game approach for resource and power allocation in cooperative femtocell networks. *IEEE Transactions on Mobile Computing*, 14(4):675–687, 2014.

[20] D. Lee, S. Lee, N. Masoud, M. Krishnan, and V. C. Li. Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction. *Advanced Engineering Informatics*, 53:101710, 2022.

[21] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Z. Xu, and W. Xu. Aoi-aware user service satisfaction enhancement in digital twin-empowered edge computing. *IEEE/ACM Transactions on Networking*, 2023.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL `https://api.semanticscholar.org/CorpusID:16326763`.

[23] Y. Liu, Y. Luo, Y. Zhong, X. Chen, Q. Liu, and J. Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017. URL `http://arxiv.org/abs/1706.02275`.

[25] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang. Communication-efficient federated learning and permissioned blockchain for digital twin edge networks. *IEEE Internet of Things Journal*, 8(4):2276–2288, 2021. doi: 10.1109/JIOT.2020.3015772.

[26] Z. Lv, L. Qiao, and R. Nowak. Energy-efficient resource allocation of wireless

energy transfer for the internet of everything in digital twins. *IEEE Communications Magazine*, 60(8):68–73, 2022.

[27] R. Minerva, G. M. Lee, and N. Crespi. Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models. *Proceedings of the IEEE*, 108(10):1785–1824, 2020.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[29] H. X. Nguyen, R. Trestian, D. To, and M. Tatipamula. Digital twin for 5g and beyond. *IEEE Communications Magazine*, 59(2):10–15, 2021.

[30] P. Sadeghi, R. A. Kennedy, P. B. Rapajic, and R. Shams. Finite-state markov modeling of fading channels-a survey of principles and applications. *IEEE Signal Processing Magazine*, 25(5):57–80, 2008.

[31] M. Sanz Ausin, H. Azizsoltani, S. Ju, Y. J. Kim, and M. Chi. Infernet for delayed reinforcement tasks: Addressing the temporal credit assignment problem. *arXiv e-prints*, pages arXiv–2105, 2021.

[32] W. Shengli. Is human digital twin possible? *Computer Methods and Programs in Biomedicine Update*, 1:100014, 2021. ISSN 2666-9900. doi: https://doi.org/10.1016/j.cmpbup.2021.100014. URL `https://www.sciencedirect.com/science/article/pii/S2666990021000136`.

[33] Y. Shu, Z. Wang, H. Liao, Z. Zhou, N. Nasser, and M. Imran. Age-of-information-aware digital twin assisted resource management for distributed energy scheduling. pages 5705–5710, 2022.

[34] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[35] B. Tan and A. Matta. The digital twin synchronization problem: Framework, formulations, and analysis. *IISE Transactions*, 56(6):652–665, 2024.

[36] L. Tang, Z. Cheng, J. Dai, H. Zhang, and Q. Chen. Joint optimization of vehicular sensing and vehicle digital twins deployment for dt-assisted iovs. *IEEE Transactions on Vehicular Technology*, 2024.

[37] F. Tao, H. Zhang, A. Liu, and A. Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, 15(4):2405–2415, April 2019.

[38] J. M. Taylor and H. R. Sharif. Leveraging digital twins to enhance performance of iot in disadvantaged networks. In *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pages 1303–1308. IEEE, 2020.

[39] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, and G. Karakostas. Digital twin placement for minimum application request delay with data age targets. *IEEE Internet of Things Journal*, pages 1–1, 2023. doi: 10.1109/JIOT.2023.3244424.

[40] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, and G. Karakostas. Digital twin placement for minimum application request delay with data age targets. *IEEE Internet of Things Journal*, 2023.

[41] R. van Dinter, B. Tekinerdogan, and C. Catal. Predictive maintenance using digital twins: A systematic literature review. *Information and Software Technology*, 151:107008, 2022.

[42] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[43] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[44] R. Xu, C.-W. Park, S. Khan, W. Jin, S. J. S. Moe, and D. H. Kim. Optimized task scheduling and virtual object management based on digital twin for distributed edge computing networks. *IEEE Access*, 2023.

[45] M. Yan, W. Gan, Y. Zhou, J. Wen, and W. Yao. Projection method for blockchain-enabled non-iterative decentralized management in integrated natural gas-electric systems and its application in digital twin modelling. *Applied Energy*, 311:118645, 2022.

[46] L. Yang, Y. Zou, S. Shen, P. Wang, D. Yu, and X. Cheng. A fault-tolerant communication algorithm for age-of-information optimization in ditens. *IEEE Transactions on Communications*, 2023.

[47] W. Yang, W. Xiang, Y. Yang, and P. Cheng. Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial iot. *IEEE Transactions on Industrial Informatics*, 19(2):1884–1893, 2022.

[48] D. Yu and Z. He. Digital twin-driven intelligence disaster prevention and mitigation for infrastructure: Advances, challenges, and opportunities. *Natural hazards*, 112(1):1–36, 2022.

[49] K. Zhang, J. Cao, and Y. Zhang. Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks. *IEEE Transactions on Industrial Informatics*, 18(2):1405–1413, 2022. doi: 10.1109/TII.2021.3088407.