Thesis for the Degree of Master

Energy - Delay Optimization in Mobile Edge Computing Federation System

제어 모바일 에지 컴퓨팅 연합 시스템에서 에너지 지연 최적화 연구

December 2023

Department of Information and Communication Convergence

Graduate School of Soongsil University

Do Mai Huong

Thesis for the Degree of Master

Energy – Delay Optimization in Mobile Edge Computing Federation System

제어 모바일 에지 컴퓨팅 연합 시스템에서 에너지 지연 최적화 연구

December 2023

Department of Information and Communication Convergence

Graduate School of Soongsil University

Do Mai Huong

Thesis for the Degree of Master

Energy – Delay Optimization in Mobile Edge Computing Federation System

A thesis supervisor: 유명식

Thesis submitted in partial fulfillment of the requirements for the Degree of Master

December 2023

Department of Information and Communication Convergence

Graduate School of Soongsil University

Do Mai Huong

To approve the submitted thesis for the

Degree of Master by Do Mai Huong

Thesis Committee

Chair	김영한	(signature)
Member	정윤원	(signature)
Member	유명식	(signature)

December 2023

Graduate School of Soongsil University

ACKNOWLEDGEMENT

Initially, I'd like to extend my heartfelt gratitude to my advisor, Prof. MyungSik Yoo, for his guidance in my master's study and research, for his motivation, his willingness whenever I needed help, and his profound knowledge. My thesis will not be possible without him.

Second, I also extend my appreciation to the other members of my thesis committee for their supportive comments and encouragement.

In particular, I want to thank the "Pho" team for bringing me the great experience in Korea. They are pleasant and willing to assist me at any moment. I am also grateful to all our lab-mates for supporting me during the time I studied at Soongsil University.

Finally, I would like to give appreciation to my family for their spiritual encouragement during the time I was studying in Korea.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Energy – Delay Optimization in Mobile Edge Computing Federation

DO MAI HUONG

Department of Information and Communication Convergence

Graduate School of Soongsil University

Industrial IoT (IIoT) has seen remarkable growth with the advent of smart devices and 5G technology, yet faces limitations in device capabilities. To address resource constraints, computation offloading has emerged as a viable solution. Mobile edge computing (MEC) alleviates device workload, and MEC federation enhances resource utilization but encounters practical challenges. This thesis delves into decision offloading and computing resource allocation within the MEC federation system. Our proposition introduces a task offloading framework designed to minimize overall delay-energy expenses, while accounting for resource constraints within the MEC federation. This challenge is reformulated as a Markov Decision Process (MDP) and addressed through a resource allocation and task offloading algorithm that leverages the Deep Deterministic Policy Gradient (DDPG) technique and integrates the Prioritized Experience Replay (PER) buffer.. Simulation results demonstrate that DDPG-PER-IS effectively reduces the delay-energy cost.

국문초록

제어 모바일 에지 컴퓨팅 연합 시스템에서 에너지 지연 최적화 연구

무이후엉도

정보통신융합학과

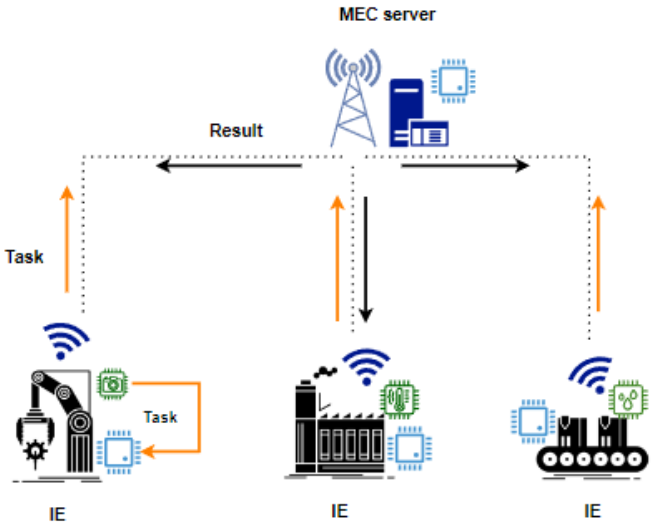
숭실대학교 대학원

산업용 IoT 는 스마트 기기와 5G 기술의 등장으로 괄목할 만한 성장을 이루었지만 기기 성능에는 한계가 있습니다. 리소스 제약을 해결하기 위해 계산 오프로딩이 실행 가능한 솔루션으로 등장했습니다. 모바일 에지 컴퓨팅는 장치 작업 부하를 완화하고 MEC 페더레이션은 리소스 활용도를 향상시키지만 실질적인 문제에 직면합니다. 본 논문에서는 모바일 에지 컴퓨팅는 연합 시스템 내에서의 의사결정 오프로딩과 컴퓨팅 자원 할당에 대해 탐구합니다. 모바일 에지 컴퓨팅는 연합의 자원 상황을 고려하여 총 지연 에너지 비용을 최소화하는 것을 목표로 하는 작업 오프로딩 프레임워크를 제안합니다. 이 문제는 Markov Decision Process 로 변환되고 Deep Deterministic Policy Gradient 및 Prioritized Experience Replay 버퍼를 기반으로 하는 자원 할당 및 작업 오프로딩 알고리즘을 사용하여 해결됩니다. 시뮬레이션 결과는 DDPG-PER.

CHAPTER 1 INTRODUCTION

The Internet of Things (IoT) has rapidly expanded, particularly in Industrial IoT (IIoT), offering enhanced productivity and cost-efficiency in industries [1], [2]. IIoT projects are prevalent, with projections anticipating a global investment of 992 billion by 2025. Despite its potential benefits, IIoT faces challenges due to data volume and device limitations. To address these constraints and cater to diverse application needs, developing technologies that optimize IIoT system performance is crucial.

Cloud computing has been utilized in IIoT systems to tackle challenges, but its remote server deployment often results in increased latency. Mobile Edge Computing (MEC) emerges as a solution by deploying servers closer to IoT devices, reducing latency for intensive tasks. Figure 1-1 illustrates an IIoT system with an MEC server, enabling data acquisition and processing. Tasks can be managed locally by IIoT devices or offloaded to the MEC server. Research suggests leveraging dedicated MEC servers for low-latency tasks; otherwise, tasks might be rerouted to remote cloud servers, increasing latency and underutilizing nearby MEC resources [3], [4].



[Figure 1-1] The IIoT-enabled MEC system.

The concept of MEC federation [5], revolutionizes IIoT systems by establishing direct connections among MEC servers, optimizing resource utilization and workload allocation. However, prior studies faced constraints in formulating optimization problems. Some overlooked the computing abilities of IIoT devices [6] [7], relying excessively on MEC servers. Other studies assumed infinite storage within MEC servers, presenting practical challenges in decision-making during high-workload scenarios. Optimizing resource allocation and task offloading requires considering IIoT device capabilities while accounting for storage and queuing limitations in MEC servers.

Managing energy consumption in IIoT devices is vital due to limited battery capacity. Studies aimed at optimizing both task processing latency and power usage in IIoT devices exist [8]. However, some research has limitations in modeling MEC

system energy consumption. While some studies focused on enhancing IIoT device energy efficiency, ignoring server power consumption, other studies emphasized server energy optimization but overlooked idle power consumption. Failure to optimize server power can result in financial setbacks. Therefore, it's vital to reduce energy usage in both IIoT devices and MEC servers while executing tasks to mitigate economic losses.

Various methods, such as traditional optimization-based [4] and machine learning (ML)-based [8] have been proposed to address this challenge. While heuristic techniques like particle swarm optimization offer low complexity, they often trade performance for high-dimensional systems. Machine learning, particularly deep reinforcement learning (DRL), has gained traction in optimizing MEC resources. Techniques like deep Q-learning and DDPG have been employed to manage resources in MEC-enabled IIoT networks efficiently. DDPG, designed for continuous optimization problems, utilizes prioritized experience replay (PER) [9] and importance sampling (IS) to enhance performance and mitigate overfitting.

In this research, we tackle crucial challenges by delving into the performance and resource optimization complexities within an MEC federation system integrated into an IIoT network. Our primary aim is to simultaneously tackle the issues of reducing latency and optimizing energy consumption across the entire IIoT system. Our approach involves considering the resources that are accessible or present within both the IEs and servers., incorporating computational capabilities, communication infrastructure, storage capacity, and power provisions across both

IEs and servers. These factors serve as inputs for the optimization problems we address. To tackle these issues effectively, we introduce a resource allocation and task offloading framework based on DDPG, enhanced with a PER buffer. Our study makes the following key contributions:

- We introduce an innovative framework in IIoT systems with MEC federation to allocate resources and offload tasks, prioritizing end-to-end delay and energy efficiency, supported by a comprehensive objective function and queuing model.
- We convert our MINLP-based optimization problem into a MDP and employ the DDPG-PER algorithm as a Deep Reinforcement Learning (DRL) approach within the MEC Federation controller to address this challenge.
- We conduct thorough experiments, showcasing that our MEC federation system and the DDPG-PER method outperform other strategies, demonstrating superior effectiveness in IIoT systems..

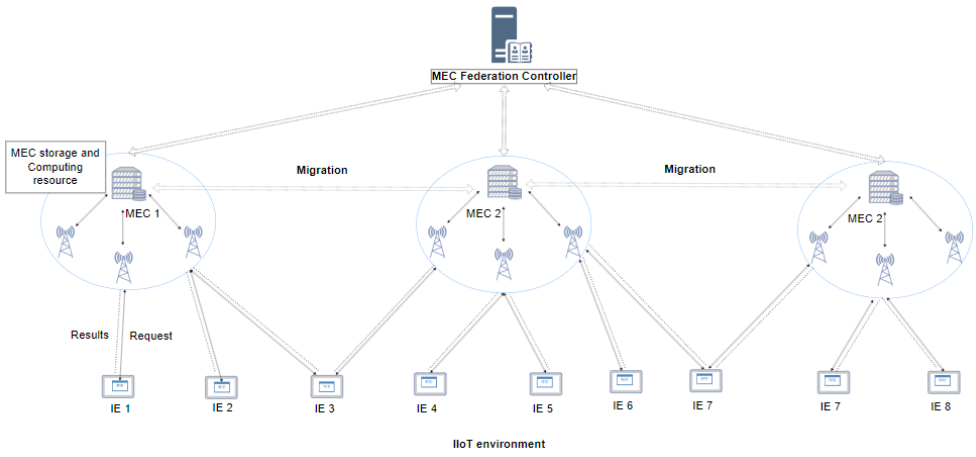
The structure of this thesis is outlined as follows:

- Chapter 2 outlines the proposed system model and optimization problem.
- Chapter 3 details the resource allocation algorithm based on DDPG-PER used to address this problem.
- Chapter 4 offers a presentation of the simulation results.
- Finally, Chapter 5 summaries the findings and contributions of this thesis.

CHAPTER 2 SYSTEM AND PROBLEM STATEMENT

2.1. System overview

In Figure 2-1, an MEC-driven IIoT network displays IEs within fixed zones, connecting wirelessly to multiple strategically positioned MEC servers. These servers, part of an interconnected MEC federation, facilitate resource sharing and information exchange. A central MEC federation controller manages data aggregation and optimal resource allocation. IEs offload computation-intensive tasks to local MEC servers; however, resource constraints may prompt task transfer to remote MEC servers, orchestrated by the MEC federation controller.



[Figure 2-1] The MEC-based IIoT system architecture.

2.2. System model

We designate the quantity of IE servers as *D*, and MEC servers as *S*. Our study involves dividing into *T* time slots. Within *t*, an IE *d* has $I^d(t)$ tasks, as outlined in a prior study [3]. In detail, we represent the task as a set:

$$w_i^d(t) = \{a_i^d(t), b_i^d(t), c_i^d(t)\}, \quad (1)$$

here, $a_i^d(t), b_i^d(t)$and $c_i^d(t)$ signify the data, result size, and the workload needed to execute the data unit.

The determination of the offloading decision of the tasks is denoted by the variable $x_i^d(t)$. Specifically, $x_i^d(t) = 0$ signifies that the task is to be executed bby local device, while $x_i^d = s$ $(0 < s < S)$ shows that the task is being processed by the server *s*. The offloading strategy determines the task execution as follows:

- $x_i^d(t) = 0$, for local IE
- $x_i^d(t) = s$, for local server *s*,
- $x_i^d(t) = r \neq s$, for remote server *r*.

2.3. Delay model

2.3.1. Delay of computation

The delay incurred during task execution, whether locally on an equipment or on an offloaded server, is an inherent part of computation. The latency involved in processing the task on IE *d* ($T_i^d(t)$) can be computed using the following formula:

$$T_i^d(t) = \frac{a_i^d(t)c_i^d(t)}{f_i^d(t)}, \quad (2)$$

here, $f_i^d(t)$ represents the computational resource, which is allocated by device to execute the task $w_i^d(t)$ during *t*. The delay of the computational task $w_i^d(t)$ performed on server ***s*** is calculated by the formula:

$$T_i^s(t) = \frac{a_i^d(t)c_i^d(t)}{f_i^s(t)}, \quad (3)$$

where $f_i^s(t)$ is the computational resource, which is allocated by server ***s*** to process $w_i^d(t)$ within time slot ***t***.

2.3.2. Delay of transmission

This study focuses on a network utilizing two types of communication: wireless connections for communication between IEs and MEC servers, and a wired network linking MEC servers among themselves. For both types of connections, we define uplink and downlink channels. The uplink handles task data transmission, while the downlink manages the transmission of resulting data.

The gain of the channel wireless link between device and server is denoted by $G_{d,s}$ and calculated using the formula:

$$G_{d,s} = 127 + 30\log\left(dist_{d,s}\right), \tag{4}$$

where $dist_{\{d,s\}}$ represents the distance. Consequently, the data rate of uplink $R^U_{\{d,s\}}(t)$ and downlink $R^D_{\{d,s\}}(t)$ between device and server *s* during *t* are derived using Shannon's formula:

$$R^U_{\{d,s\}}(t) = B^U_{\{d,s\}}(t)\log_2\left(1 + \frac{P_{d(t)}\, G_{\{d,s\}}}{N_0 B^U_{\{d,s\}}(t)}\right), \tag{5}$$

$$R^D_{\{d,s\}}(t) = B^D_{\{d,s\}}(t)\log_2\left(1 + \frac{P_s(t) G_{\{d,s\}}}{N_0\, B^D_{\{d,s\}}(t)}\right), \tag{6}$$

here, $B^U_{\{d,s\}}(t)$ and $B^D_{\{d,s\}}(t)$ denote the bandwidths which are allocated to the up and down link between device and server *s*. $P_d(t)$ and $P_s(t)$ represent the transmission power, which are allocated by device and server *s*, while N_0 signifies Gaussian noise.

The data transmission rate of the wired network, on the other hand, correlates to the connection bandwidth. Therefore, the data rate of up and down link $W^{U}_{\{s,r\}}(t)$ and $W^{D}_{\{s,r\}}(t)$ in wired from server ***s*** to MEC r during ***t*** are simply:

$$W^{U}_{\{s,r\}}(t) = B^{U}_{\{s,r\}}(t), \tag{7}$$

$$W^{D}_{\{s,r\}}(t) = B^{D}_{\{s,r\}}(t), \tag{8}$$

where $B^{U}_{\{s,r\}}(t)$ and $B^{D}_{\{s,r\}}(t)$ are the uplink bandwidth and downlink bandwidth of link between server *s* and server *r*.

Uplinks from device to local server and between two servers are utilized for task data transmission. $T^{\{U,R\}\{d,s\}}(t)$ and $T^{\{U,W\}\{s,r\}}(t)$ are the transmission delay of the wireless (device to server) and wired (server to server) in the uplink, they are computed as follows:

$$T^{\{U,R\}\{d,s\}}(t) = \frac{a^{d}_{i(t)}}{R^{U\{d,s\}(t)}}, \tag{9}$$

$$T^{\{U,W\}\{s,r\}}(t) = \frac{a^{d}_{i(t)}}{W^{U\{s,r\}(t)}}, \tag{10}$$

where $a^{d}_{i}(t)$ represents the size of the task data.

Similar, the transmission delays of the two downlinks are:

$$T^{\{D,R\}\{d,s\}}(t) = \frac{b^{d}_{i}(t)}{R^{D\{d,s\}}(t)}, \tag{11}$$

$$T^{\{D,W\}\{s,r\}}(t) = \frac{b^{d}_{i}\,(t)}{W^{D\{s,r\}}(t)}, \tag{12}$$

where $b^{d}_{i}(t)$ represents the size of the task results.

2.3.3. Delay of queuing

The server utilizes a task buffer to store the new coming tasks that waiting to process. In this model, task arrivals follow a Poisson distribution, while $c = 4$ denoting the number of cores [10]. Considering λ_s as the tasks average arrival rate and β_s as the MEC server's service rate, the M/M/4/SC^s defines the probability P_i that i tasks exist within the buffer [11]. Consequently, the queue delay $T_s^q(t)$ as follows:

$$T_{\{s,i\}}^q(t) = \frac{\{SC\}^{\{s\}}(t) - \{SA\}^{\{s\}}(t)}{\lambda_s\left(1 - P_{\{SC^s\}}\right)}, \tag{13}$$

here, $SC^s(t)$ represents the capacity of server ***s*** at time slot ***t***, while $SA^s(t)$ denotes the available storage within the server at that time. The difference between $SC^s(t)$ and $SA^s(t)$, i.e., $SC^s(t) - SA^s(t)$, indicates the number of tasks awaiting execution in the queue. P_{SC}^s stands for the probability that a total of SC^s tasks are present in the queuing line.

2.4. Energy consumption model

2.4.1. Energy consumption of computation

The power which is used by device and server *s* when executing task $w_i^d(t)$ in each time ***t*** is represented by [12]:

$$p_i^{\{d\}}(t) = \xi\left[f_i^{\{d\}}(t)\right]^3, \tag{14}$$

$$p_i^{\{s\}}(t) = \xi\left[f_i^{\{s\}}(t)\right]^3, \tag{15}$$

here, ξ represents the effective parameters, which relies on the hardware architecture. Consequently, the energy of device or server *s* which consumed is as:

$$E_i^{\{d\}}(t) = p_i^{\{d\}}(t)T_i^{\{d\}}(t), \quad (16)$$

$$E_i^{\{s\}}(t) = p_i^{\{s\}}(t)T_i^{\{s\}}(t). \quad (17)$$

2.4.2. Energy for transmission and reception

Certainly, the transmission energy required by each IE or MEC server for offloading or migrating tasks is a crucial consideration. As a result, $E^{\{U,R\}\{d,s\}}(t)$ and $E^{\{U,W\}\{s,r\}}(t)$ denote the energy consumption which used for transmission in uplink between device to server and server to server. They are calculated as follows:

$$E^{\{U,R\}\{d,s\}}(t) = T^{\{U,R\}\{d,s\}}(t)P_d(t) + T_{\{d,s\}}^{\{U,R\}}(t)P_s(t), \quad (18)$$

$$E^{\{U,W\}\{s,r\}}(t) = T^{\{U,W\}\{s,r\}}(t)P_s(t) + T_{\{s,r\}}^{\{U,W\}}(t)P_r(t). \quad (19)$$

Meanwhile, IE or MEC servers require energy to remain in an active "on" state, capable of receiving tasks or results. $E^{\{D,R\}\{d,s\}}$, (t) and $E^{\{D,W\}\{s,r\}}(t)$ represent the energy consumption which used for transmission in the downlink between server to device and server to server. They are calculated as follows:

$$E^{\{D,R\}\{d,s\}}(t) = T^{\{D,R\}\{d,s\}}(t)P_d(t) + T_{\{d,s\}}^{\{D,R\}}(t)P_s(t), \quad (20)$$

$$E^{\{D,W\}\{s,r\}}(t) = T^{\{D,W\}\{s,r\}}(t)P_s(t) + T_{\{s,r\}}^{\{D,W\}}(t)P_r(t). \quad (21)$$

These equations represent the energy consumption required for both transmitting data and keeping the devices active in the "on" mode for receiving tasks or results across various connections in the IIoT system.

2.5. Energy - Delay cost

Our scenario adversely impacts system latency and power consumption. Hence, there is a critical need to optimize both latency and energy with the trade-off concurrently. The investigation involves considering three offloading strategies.

Rather than focusing solely on an "offload or not" strategy. Such an approach not only simplifies the choice of "offload or not" but also addresses the challenge of "offload to which one."

2.5.1. First case – on device

In this scenario, the tasks $w_i^d(t)$ are exclusively executed by device *d* and are not delegated to any server ($x_i^d(t) = 0$). The delay experienced by each device encompasses the computing delay required for all tasks, while energy refers to the energy utilized for task execution. The energy-delay cost associated with task $w_i^d(t)$ is as below:

$$C_i^d(t) = \alpha . A . T_i^d(t) + \beta . B . E_i^d(t), \quad (22)$$

here, α and β serve as constant weighting parameters representing the relative importance of delay and energy costs for the task, respectively. *A* and *B* denote values used for normalizing the value.

2.5.2. Second case – local server

The local server ***s*** is 'local' if it satisfies $dist_{\{d,s\}} \leq R$, where *R* represents radius of each MEC server covers.

In this scenario, the tasks are fully executed by local server. Specifically, $x_i^d(t) = s$, implying that ***s*** denotes the local server connected to IE *d* by directly wireless link. There are three distinct kinds of delays:

- Delay for transmission
- Delay for computation
- Queuing delay

According to above equations the overall delay for computing task $w_i^d(t)$ in *t* is as follows:

$$T_i^{\{loc\}}(t) = T_i^s(t) + T_{\{d,s\}}^{\{U,R\}}(t) + T_{\{d,s\}}^{\{D,R\}}(t) + T_s^q(t). \quad (23)$$

Energy consumption is categorized as:

- Transmission energy
- Reception energy
- Computation energy

Based on above equations, the total energy consumption is as:

$$E_i^{\{loc\}}(t) = E_i^s(t) + E_{\{d,s\}}^{\{U,R\}}(t) + E_{\{s,d\}}^{\{U,R\}}(t) + E_{\{d,s\}}^{\{D,R\}}(t) + E_{\{d,s\}}^{\{D,R\}}(t). \quad (24)$$

The energy-delay cost can be expressed as:

$$C_i^{\{loc\}}(t) = \alpha.A.T_i^{\{loc\}}(t) + \beta.B.E_i^{\{loc\}}(t). \quad (25)$$

2.5.3. Third case – remote server

A remote server processes the task when $x_{i(t)}^d = r \neq s$, where ***r*** represents a server, which is not directly linked to device *d*. This approach considers three kinds of delays:

- Transmission delay
- Computation delay
- Queuing delay

The delay experienced by the remote server r is formulated as:

$$T_i^{\{rem\}}(t) = T_{\{d,s\}}^{\{U,R\}}(t) + T_{\{s,r\}}^{\{U,W\}}(t) + T_r^q(t) + T_i^r(t) + T_{\{s,r\}}^{\{D,W\}}(t) + T_{\{d,s\}}^{\{D,R\}}(t). \quad (26)$$

Moreover, energy consumption is categorized into:

- Transmission and reception energy
- Computation energy

The energy consumption in the case tasks executed by the remote server r considers various energy components:

$$E_i^{\{rem\}}(t) = E_{\{d,s\}}^{\{U,R\}}(t) + E_{\{s,r\}}^{\{U,W\}}(t) + E_i^r(t) + E_{\{s,r\}}^{\{D,W\}}(t) + E_{\{d,s\}}^{\{D,R\}}(t). \quad (27)$$

So, the energy-delay cost $C_i^{\{remote\}}(t)$ is formulated as:

$$C_i^{\{rem\}}(t) = \alpha.A.T_i^{\{rem\}}(t) + \beta.B.E_i^{\{rem\}}(t). \quad (28)$$

2.6. Problem formulation

Based on the outlined scenarios, the energy-delay cost function, expressed as $\mathbb{C}_i^d(t)$, is defined as follows:

$$\mathbb{C}_i^d(t) = \begin{cases} C_i^d(t), & x_i^d(t) = 0 \\ C_i^{loc}(t), & x_i^d(t) = s \\ C_i^{rem}(t), & x_i^d(t) = r \neq s \end{cases}$$

The primary target of this research aims to get the minimum of the average energy-delay cost across the entire system spanning T time slots, incorporating both devices and servers, while satisfy the resource and stringent delay constraints. This objective function is formulated as an optimization problem:

$$\min_{x_i^d(t)} \sum_{t=1}^{T} \sum_{d=1}^{D} \sum_{i=1}^{I^d} \frac{1}{T} \frac{1}{D} \frac{1}{I^d} \mathbb{C}_i^d(t) \quad (29)$$

The objective function works under the list of constraints below:

Constraint 1: Offloading strategy decision is only one option in one time

$$0 \leq x_i^d(t) \leq S$$

Constraint 2: Limited transmission power

$$p_{min}(t) \leq p_d(t) \leq p_{max}(t), p_{min}(t) \leq p_s(t) \leq p_{max}(t)$$

Constraint 3: Limited bandwidth

$$\sum_{d=1}^{D} B_{d,s}^{Up} \leq B_{max}^{Up}, \sum_{d=1}^{D} B_{d,s}^{Down} \leq B_{max}^{Down}$$

Constraint 4: Limited storage (caching) resource

$$SA^s(t) \geq \sum_{d=1}^{D} S^d(t); S^d(t) \geq 0$$

Constraint 5: Limited computing resource

$$FA^s(t) \geq \sum_{d=1}^{D} f_i^s(t); f_i^s(t) \geq 0$$

$$FA^d(t) \geq \sum_{d=1}^{D} f_i^d(t); f_i^d(t) \geq 0$$

Constraint 6: Limited battery of IE [13]

$$EA^d(t) \geq \sum_{d=1}^{D} E_i^d(t);$$

$$C_{i(t)}^{\{loc\}} = \alpha \cdot A \cdot T_{i(t)}^{\{loc\}} + \beta \cdot B \cdot E_{i(t)}^{\{loc\}}$$

In the subsequent section, we will demonstrate how the optimization problem defined above exhibits characteristics akin to those of a MINLP problem.

CHAPTER 3 METHODOLOGY

The framework of resource allocation and tasks offloading strategies in an IIoT network's MEC federation is presented in this section. Initially modeled as a MDP in optimization problem above, DDPG emerges as a suitable solution due to its adaptability to dynamic states and large data dimensions. However, DDPG's complexity during training prompts the proposal of a hybrid solution combining DDPG with PER. This hybrid approach aims to enhance training efficiency and mitigate complexity issues. Our framework utilizes DDPG-PER to optimize energy and latency in an MEC-enabled IIoT network.

3.1. Markov Decision Process

To implement DRL to tackle the optimization problem, we define the key components of a MDP, encompassing five crucial elements: the state space S, reward function R, action space A, discount factor γ, state transition probability P. As each time slot t commences, the agent is tasked with selecting an action $a_t \in A$ based on the current system state $s_t \in S$. Following this action, the environment undergoes an update to the subsequent state $s_{\{t+1\}}$ of the system, providing a reward as feedback to gauge the efficacy of the chosen action. Through this continual interaction between the agent and the environment, the agent can formulate a policy that serves as a mapping from states to actions.

- State space:

  The system state at time slot t is represented as:

$$s_t = \{K(t), D(t), E(t), F(t), S(t)\},$$

with,

$K(t)$: location of tasks,

$D(t)$: size of tasks,

$E(t)$: available power,

$F(t)$: available computing resources,

$S(t)$: available storage resources.

- Action space:

  The agent makes the decision follows below set:

$$a_t = \{P(t), f(t), B(t), s(t), X(t)\},$$

  where,

  $X(t)$: indicates all task offloading decisions,

  $P(t)$: allocated power resources,

  $f(t)$: allocated computing resources,

  $B(t)$: allocated bandwidth resources,

  $s(t)$: allocated storage resources.

- Reward function:

  The reward r_t reflects the immediate benefit obtained upon taking action a_t within the state s_t. In line with the goal of minimizing the energy-delay cost, the reward r_t at $\boldsymbol{t}$ is structured to represent the negative value of the average cost of IE tasks.

$$r_t = -\sum_{d=1}^{D}\sum_{i=1}^{I^d}\frac{1}{D}\frac{1}{I^d}\mathbb{C}_i^d(t). \quad (30)$$

3.2. DDPG – PER framework

The DDPG framework involves four neural networks: the actor network (θ^μ) aims to refine the policy $\mu(\theta^\mu)$ to choose the best actions for specific states. The critic network (θ^Q) assesses action quality through Q-values. Furthermore, delayed target networks, $\theta^{\mu'}$ and $\theta^{Q'}$ are employed to maintain stable estimated targets, thereby improving training stability and learning effectiveness.

The DDPG-based framework includes four networks, corresponding four trainable parameters:

- θ^μ: presents actor network – uses Deterministic Policy Gradient function, proposes an action given a state by policy.
- θ^Q: present critic network – uses Deep Q network, predicts if the action is good or bad given a state and an action.
- $\theta^{\mu'}$: target actor network, the time-delayed copies of original actor network, that slowly tracks and improves the learned network.
- $\theta^{Q'}$: target critic network, the time-delayed copies of original critic network, that slowly tracks and improves the learned network.

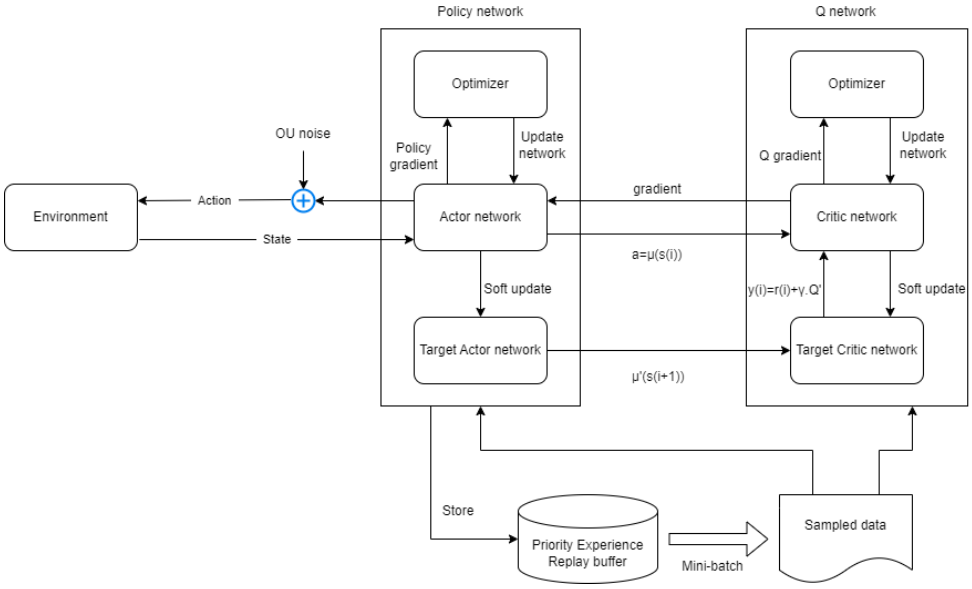
The process includes three phases:

- Update critic network, find the Q-value, the parameter measures how good the action. Based on Bellman's equation, the Q-value is:

$$Q^{\mu}(s_t, a_t) = E[r_t + \gamma Q(s_{t+1}, \mu_{t+1})]. \quad (31)$$

- Update actor network, optimize the policy and find the action.
- Update two target networks.

Details of whole process is shown in Figure 3-1.



[Figure 3-2] The DDPG-PER framework.

The output of actor network and critic network are:

$$\mu(s_t|\theta^{\mu}) \approx \mu^{*(s_t)}, \quad (32)$$

$$Q(s_t, a_t|\theta^{Q}) \approx Q(s_t, a_t). \quad (33)$$

Update Q-value by one step od gradient descent using:

$$L(\theta^{Q}) = \frac{1}{M}\delta^2, \quad (34)$$

with

$$\delta = Q(s_i, a_i|\theta^{Q}) - y_i. \quad (35)$$

Update policy by one step of gradient ascent using [14], [15]:

$$\nabla_{\{\theta^{\mu}\}} J_{\{\mu*\}} = \frac{1}{M} \sum_{i=1}^{M} \nabla_{\{a_i\}} \left[Q^{\mu}(s_i, a_i | \theta^{Q}) \nabla_{\{\theta^{\mu} \mu(s_i | \theta^{\mu})\}} \right]. \tag{36}$$

Update target network with:

$$\theta^{\{\mu'\}} \leftarrow (1 - \delta)\theta^{\{\mu'\}} + \delta\theta^{\mu}, \tag{37}$$

$$\theta^{\{Q'\}} \leftarrow (1 - \delta)\theta^{\{Q'\}} + \delta\theta^{Q}. \tag{38}$$

On the other hand, PER in DDPG assigns accurate values to experiences using TD-error metrics, avoiding overfitting through stochastic prioritization and correcting state visitation bias with importance-sampling weights. The probability of sampling experience j is determined as:

$$P(j) = \frac{D_j^u}{\sum_i D_i^u}, \tag{39}$$

with D_j is the priority of experience j.

The IS weight stabilizes training by reducing gradient magnitude, computed as detailed in \cite{mahmood2014weighted}:

$$w_j = \frac{1}{V^{\{v\}}.P(j)^v}. \tag{40}$$

The loss function is reformulated as below:

$$w_j \nabla_{\left\{\theta_1^Q\right\}} L\left(\theta_1^Q\right) = w_j \delta_j \nabla_{\left\{\theta_1^Q\right\}}.Q\left(s, a \middle| \theta_1^Q\right). \tag{41}$$

CHAPTER 4 EXPERIMENTS

4.1. Simulation requirement

The simulation runs on a PC with the configuration as below:

- Windows 11 (64-bit)
- 2.60 GHz Intel(R) Core(TM) i5-11400F
- RAM 16 GB

The simulation environment uses Anaconda and Microsoft Visual Studio IDE, details of the package used are as follows:

- Python 3.7
- Tkinter 8.6
- Tensorflow 2.6

4.2. Simulation setting

This study investigated an IIoT network empowered by Mobile Edge Computing wherein multiple MEC servers collaborate. The system comprises ten industrial zones, each with a radius of 500 meters, with a single MEC server covering each zone. IIoT devices are randomly positioned within these areas and connect exclusively to the local MEC server of their respective region. The system's connections comprise wireless links connecting the IE to the local MEC server, along with wired connections among the MEC servers. Within the DDPG-PER model, the objective is to optimize task offloading and resource allocation to minimize energy-delay costs. The model consists of four 5-layer networks that

continually update during system operation, adapting to changing environments. Using exponential moving average (EMA) models, training parameters are adjusted based on parameter trends. Both batch and mini-batch sizes are set to 32, and training commences as the memory fills with samples, which are explored to form the dataset used for learning. Based on similar simulations [16] main parameters are shown in Table 4-1.

[Table 4-2] Setting parameters.

Parameters	Value
Number of servers	10
Number of devices	{10, 20, 30, 40}
Wireless bandwidth device – server	[0, 20] MHz
Wired full bandwidth server – server	[0, 150] MBps
Buffer size	[0, 25] GB
CPU capacity of device	[0, 5] GHz
CPU capacity of server	[0, 25] GHz
Task size	1 MB
Transmission power range	1 MB
Workload requirement of 1 bit task	737.5 cycles
Time executed requirement of 1 bit task	2.10-8 s
Batch size	32
Mini batch size	32
Replay memory size	10000
Total episode	100

4.3. Testing cases

Base on the type of MEC federation systems:

- MEC Federation greedy: only after the most recent location is overloaded may tasks be completed by IEs, local MEC, or distant MEC.
- MEC federation optimal: tasks can be executed by IEs, local MEC or remote MEC to get the optimal energy-delay cost.

Base on the resource allocation & task offloading decision-making methods:

- DDPG-PER: making the best decision regarding resource allocation and task offloading to get the optimal point.
- Random: the resources for both IEs and MEC servers are assigned at random, and the task offloading choice is made by the DDPG-PER.
- Uniform: following the allocation of resources for both IEs and MEC servers, the task offloading choice specified by the DDPG-PER is made.

Besides, we also change the variables of the number of IEs to evaluate the effect when increasing the number of IEs: 10, 20, 30, 40.

4.3.1. Delay performance comparison

Table 4-3 illustrates the comparison of average execution delays across various algorithms. As the number of IEs in the system increases, task completion latency rises across all methods due to resource exhaustion. Specifically, in a system with 10 IEs utilizing DDPG-PER, task latency is 0.86 seconds, slightly lower than the latency observed in greedy MECF at 0.90 seconds. Similar patterns are evident in URA and RRA methods. MEC federation is instrumental in addressing task

management challenges in IIoT systems, with optimal MECF showcasing the most efficient strategy to minimize latency.

[Table 4-4] Average of delays (s)

Number of devices		40	30	20	10
RRA	Greedy	5.26	3.35	2.4	1.47
	Optimal	4.52	2.78	2.32	1.43
URA	Greedy	4.98	3.56	2.14	1.27
	Optimal	4.46	3.4	2.11	1.14
DDPG - PER	Greedy	3.85	2.94	1.94	0.9
	Optimal	**3.37**	**2.68**	**1.73**	**0.86**

Furthermore, DDPG-PER consistently better than other methods in optimizing delay. As indicated in Table 4-5, the proposed method consistently exhibits reduced delays across settings compared to RRA and URA. In the case of 40 IEs, where tasks are solely managed within IEs, the latencies for RRA, URA, and DDPG-PER are 18.5s, 17.98s, and 17.28s, respectively. When local MEC servers handle ongoing MEC tasks, the minimum latency for RRA, URA, and DDPG-PER drops to 5.58s, 5.51s, and 4.12s, respectively. When using optimal strategy, the average task latency for RRA and URA is 4.52s and 4.46s, respectively, whereas that of DDPG-PER stands at 3.37s. This underscores the superior effectiveness of system when using DDPG-PER in minimizing latency.

4.3.2. Energy performance comparison

In addition to studying system latency, we evaluate power consumption optimization performance, as depicted in Table 4-6. Similar to latency, the quantity of IEs within the system plays a significant role in determining energy consumption. For example, in a scenario involving an IIoT configuration containing 10 IEs implementing RRA, the average power utilization for tasks using Optimal MECF and Greedy MECF stands at 40.1 J and 42.8 J, respectively. This pattern remains consistent even as the quantity of IEs grows., indicating that Greedy MECF consumes more energy per task compared to Optimal MECF. Notably, when considering RRA and URA alongside MEC federation, the combined energy usage surpasses that of DDPG-PER, highlighting the inefficiency of these methods in the MEC federation system.

[Table 4-7] Average energy performance (J)

Number of devices		40	30	20	10
RRA	Greedy	173.8	131.2	81.5	42.8
	Optimal	159.9	124.8	75.2	40.1
URA	Greedy	135.8	104.3	66.3	31.2
	Optimal	123	95.9	59.2	30.5
DDPG - PER	Greedy	108.8	85.4	58	27.3
	Optimal	**105.2**	**81.1**	**51.3**	**25.7**

4.3.3. Energy – Delay cost performance comparison

The utilization of DDPG-PER in Optimal MECF stands out for delivering optimal performance by concurrently optimizing both latency and power consumption. The tasks managed by Optimal MECF consistently demonstrate the lowest average energy-delay cost, showcasing the effectiveness of DDPG-PER in achieving efficiency across latency and power consumption metrics simultaneously, as shown in Table 4-8. This advantage is obtained by using the computing capabilities and available resources of MEC servers throughout the whole network, rather than just processing using a greedy technique. The average energy-delay cost for each job completed by Greedy MECF and Optimal MECF in a system with 10 IEs using RRA is 140 and 134. Similarly to RRA, URA's resource allocation for the 10-IE system resulted in expenses of 114 and 105 when activities were only executed with Greedy MECF and Optimal MECF, respectively.

[Table 4-9] Average Energy – Delay cost

Number of devices		40	30	20	10
RRA	Greedy MECF	524.2388	356.029	241.572	140.2746
	Optimal MECF	461.8816	312.6484	229.6496	134.7994
URA	Greedy MECF	465.2844	340.4128	208.5212	114.9506
	Optimal MECF	418.1908	321.104	198.9338	105.8772
DDPG - PER	Greedy MECF	363.827	280.3332	186.7612	87.066
	Optimal MECF	**329.2286**	**259.0544**	**166.0894**	**82.7788**

Additionally, across various experimental setups, DDPG-PER consistently demonstrates superior performance compared to other methodologies. When managing a network of 40 IIoT devices within the MEC federation system, resource allocation through RRA and URA results in average expenses of 461 and 418, respectively, whereas utilizing DDPG-PER incurs costs of 329. Correspondingly, with 20 IEs present, the energy-delay expenses for RRA, URA, and DDPG-PER systems are 229, 198, and 166, respectively. The limitations observed in RRA and URA stem from their constraints in resource allocation methods, including RRA's reliance on randomization and URA's tendency toward equal distribution. The experimental findings show that DDPG-PER can successfully manage resources by employing deep learning models to address complex optimization issues. As a result, the suggested DDPG-PER technique for task offloading and resource allocation in a MEC federation system is a perfect option for concurrently improving the IIoT network's latency and energy consumption.

CHAPTER 5 CONCLUSION

In this research, we introduced an innovative MEC federation paradigm for IIoT networks, allowing IEs to offload demanding tasks not only to local but also to remote MEC servers, optimizing resource utilization effectively. Unlike previous studies, our framework encompasses practical considerations for optimal decision-making, enhancing its real-world applicability. Yet, efficiently allocating resources to reduce system latency in dynamically changing conditions remains challenging. Using DDPG-PER, we defined this problem as an MDP with realistic limitations and suggested a task offloading and resource allocation framework for MEC federation in IIoT systems. Extensive testing demonstrated that the federation MEC method outperformed standalone IE or dedicated MEC server solutions. Moreover, our investigations highlighted DDPG-PER's superiority in reducing both power consumption and system delay compared to other resource allocation strategies. In future iterations, device mobility could be a consideration when optimizing IIoT system resources based on these findings.

REFERENCES

[1] Saeed, Mohammed Aledhari and Rehma Razzak and Basheer Qolomany and Ala I. Al-Fuqaha and Fahad, "Biomedical IoT: Enabling Technologies, Architectural Elements, Challenges, and Future Directions," *IEEE access : practical innovations, open solutions,* vol. 10, pp. 31306 - 31339, 2022.

[2] Aung, Nyothiri and Dhelim, Sahraoui and Chen, Liming and Lakas, Abderrahmane and Zhang, Wenyin and Ning, Huansheng and Chaib, Souleyman and Kechadi, Mohand Tahar, "VeSoNet: Traffic-Aware Content Caching for Vehicular Social Networks Using Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems,* pp. 1-12, 2023.

[3] Fang Fu and Zhicai Zhang and Fei Yu and Qiao Yan, "An actor-critic reinforcement learning-based resource management in mobile edge computing systems}," *International Journal of Machine Learning and Cybernetics,* vol. 11, pp. 1875-1889, 2020.

[4] Bebortta, Sujit and Senapati, Dilip and Panigrahi, Chhabi Rani and Pati, Bibudhendu, "Adaptive Performance Modeling Framework for QoS-Aware Offloading in MEC-Based IIoT Systems," *IEEE Internet of Things Journal,* vol. 9, pp. 10162-10171, 2022.

[5] Naouri, Abdenacer and Wu, Hangxing and Nouri, Nabil Abdelkader and Dhelim, Sahraoui and Ning, Huansheng, “A Novel Framework for Mobile-

Edge Computing by Optimizing Task Offloading," *IEEE Internet of Things Journal*, vol. 8, pp. 13065-13076, 2021.

[6] Zhang, Weishan and Guo, Wuwu and Liu, Xin and Liu, Yan and Zhou, Jiehan and Li, Bo and Lu, Qinghua and Yang, Su, "LSTM-based analysis of industrial IoT equipment," *IEEE Access*, vol. 6, pp. 23551--23560, 2018.

[7] Zhao, Long and Matsuo, Igor Brandao Machado and Zhou, Yuhao and Lee, Wei-Jen, "Design of an industrial IoT-based monitoring system for power substations," *IEEE Transactions on Industry Applications*, vol. 55, pp. 5666--5674, 2019.

[8] Xiaolan Liu and Jiadong Yu and Jian Wang and Yue Gao, "Resource Allocation With Edge Computing in IoT Networks via Machine Learning," *IEEE Internet of Things Journal*, vol. 7, pp. 3415-3426, 2020.

[9] Hou, Yuenan and Liu, Lifeng and Wei, Qing and Xu, Xudong and Chen, Chunlin, "A novel DDPG method with prioritized experience replay," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.

[10] Bertsekas, Dimitri and Gallager, Robert, Data networks, Athena Scientific, 2021.

[11] Jain, Sushant and Smith, J MacGregor, "Open finite queueing networks with M/M/C/K parallel servers," *Computers & operations research*, vol. 21, pp. 297--317, 1994.

[12] Sarwar, Abul, Cmos power consumption and cpd calculation, Texas Instruments Dallas, TX, USA, 1997.

[13] Onur Karatalay and Ioannis N. Psaromiligkos and Benot Champagne, "Energy-Efficient D2D-Aided Fog Computing under Probabilistic Time Constraints," *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 01-07, 2021.

[14] Kingma, Diederik P and Ba, Jimmy, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[15] David Silver and Guy Lever and Nicolas Manfred Otto Heess and Thomas Degris and Daan Wierstra and Martin A. Riedmiller, "Deterministic Policy Gradient Algorithms," in *International Conference on Machine Learning*, 2014.

[16] S. Yang, "A joint optimization scheme for task offloading and resource allocation based on edge computing in 5G communication networks," *Computer Communications,* vol. 160, pp. 759--768, 2020.