POWER ALLOCATIONS FOR DELAY-CONSTRAINED TASK OFFLOADING IN CELL-FREE WIRELESS NETWORKS

POWER ALLOCATIONS FOR DELAY-CONSTRAINED TASK OFFLOADING IN CELL-FREE WIRELESS NETWORKS

By SUBAHA MAHMUDA, MSc

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 (2025)

Hamilton, Ontario, Canada (Electrical and Computer Engineering)

TITLE:	Power Allocations for Delay-Constrained Task Offloading in Cell-Free Wireless Networks
AUTHOR:	Subaha Mahmuda MSc (Electronics and Communication Engineering), Izmir Institute of Technology, Izmir, Turkey
SUPERVISOR:	Dongmei Zhao
NUMBER OF PAGES:	xiii, 53

Lay Abstract

Our mobile phones have small batteries and limited computing power. When faced with complex tasks that must be done quickly, our phones may not be able to finish the tasks on time. Instead, they offload these tasks wirelessly to the nearby cell towers, where mini data centers with powerful computers process the data and send back the results.

This thesis improves this offloading by developing a new learning method that helps devices decide the best way to send tasks under changing wireless conditions and limited battery power. Using advanced technology called Transformers, the system learns patterns over time to make smarter decisions.

Results show that this method uses less energy and meets deadlines better than existing approaches.

This work can help make wireless devices more reliable and energy-efficient when connecting to nearby servers for task processing.

Abstract

This thesis addresses the uplink resource allocation problem in a cell-free (CF) environment, where mobile devices (MDs) periodically offload application data for processing to a shared edge server (ES) co-located at the central unit (CU). These user applications require timely processing at the ES, while the uncertain and time-varying wireless transmission conditions, combined with the limited battery energy of the MDs, make this difficult.

To tackle this challenge, we propose a deep reinforcement learning framework based on the Deep Deterministic Policy Gradient (DDPG) algorithm, enhanced with Transformer encoders in both the actor and critic networks. The resulting Transformer-based DDPG (T-DDPG) framework captures spatial-temporal dependencies in dynamic wireless conditions to make more informed decisions.

Simulations are conducted under varying numbers of MDs and task sizes. The proposed T-DDPG consistently outperforms conventional DDPG, achieving both lower task completion time and energy consumption, while improving the rate of task completion within deadlines. These results highlight the effectiveness of spatial-temporal policy learning for real-time uplink scheduling in CF edge computing systems.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Dongmei Zhao, for her invaluable guidance, support and encouragement throughout my research. Her insightful feedback and patient mentorship have been essential to the completion of this thesis.

I also wish to thank to my husband, Md. Mutaherul Islam and my friend, Md. Masud Rana, for their valuable assistance in enhancing my technical understanding. Additionally, I appreciate the helpful discussions and moral support from my lab mates and friends during this journey.

Finally, I am grateful to my family for their unwavering encouragement and understanding.

This thesis would not have been possible without the assistance of all these individuals.

Table of Contents

List of Figures

List of Tables

Abbreviations

CF	Cell Free
MD	Mobile Device
CU	Central Unit
DDPG	Deep Deterministic Policy Gradient
T-DDPG	Transformer-based DDPG
MEC	Mobile Edge Computing
6G	Sixth-Generation
BS	Base Station
B5G	Beyond Fifth Generation
AP	Access Point
TDD	Time Division Duplexing
CSI	Channel State Information
RL	Reinforcement Learning

DT	Digital Twin
SINR	Signal-to-Interference Ratio
CPU	Central Processing Unit
MDP	Markov Decision Process
DQN	Deep Q-Network
FSMC	Finite State Markov Channel
AWGN	Additive White Gaussian Noise
IoT	Internet of Things

Declaration of Academic Achievement

I, Subaha Mahmuda, declare that the research in this thesis is my own work carried out under the supervision of Prof. Dongmei Zhao in the Department of Electrical and Computer Engineering at McMaster University.

I was responsible for designing and implementing the simulation environment, developing the Transformer-based DDPG framework, conducting experiments, analyzing results and writing this thesis.

All contributions are acknowledged where appropriate, and this work complies with McMaster University's Academic Integrity Policy.

Chapter 1

Introduction

The rapid advancement of mobile applications such as augmented reality, virtual reality, autonomous driving, and holographic communications has placed increasing demands on wireless communication systems. These applications require not only high data rates and ultra-low latency, but also significant computational resources, which often exceed the processing capabilities of conventional mobile devices [8, 4]. Sixth-generation (6G) networks are expected to address these challenges through innovations in both computing and communication technologies.

To meet these growing requirements, Mobile Edge Computing (MEC) can be a key enabler that brings computation closer to the network edge [10, 20]. At the same time, Cell-Free (CF) networks offer a novel architecture that eliminates the limitations of traditional cellular networks, enhancing reliability, spectral efficiency, and user fairness [12]. The integration of MEC and CF networks presents a promising approach to realize low-latency, energy-efficient, and scalable edge intelligence in 6G systems.

The remaining part of this chapter introduces the core concepts of MEC and CF networks. It also highlights the motivation for this research and provides an overview of the contributions and structure of the thesis.

1.1 Mobile Edge Computing (MEC)

MEC is a network architecture in which computation and storage resources are deployed at the edge of wireless networks to handle tasks offloaded from mobile devices. Compared to offloading all computation to remote cloud servers, MEC can significantly reduce transmission delays by bringing processing capabilities closer to end users [17, 15].

In the MEC framework, a user uploads task data to a MEC server via the uplink. The task is then executed at the edge server, and the result is sent back to the user via the downlink. This localized approach is well-suited for real-time applications where latency and reliability are critical. Compared to traditional cloud computing, MEC reduces service delay, alleviates core network congestion, and enhances user experience in computation-intensive services [13, 25].

However, the effectiveness of MEC systems is heavily dependent on the underlying wireless network's ability to support low-latency, high-throughput communication, which is a limiting factor in conventional cellular architectures.

1.2 Cell-Free Networks

In traditional cellular networks, communication performance is heavily influenced by the presence of cell boundaries. Each user connects to a dedicated base station (BS) within a predefined geographic area, which can lead to deteriorating signal quality, higher interference, and reduced data rates, particularly at the cell edges. As next-generation applications increasingly demand ultra-reliable, low-latency connectivity, conventional cell-based architectures become insufficient.

CF networks have emerged as a promising alternative for 6G systems [12]. In CF architectures, the concept of cell boundaries is eliminated. As illustrated in Fig. 1.1, a large number of geographically distributed Access Point (APs), all connected to and coordinated by a Central Unit (CU), collaboratively serve users. These APs operate on the same time-frequency resources and adopt a user-centric approach, ensuring uniformly high-quality service across the entire network.

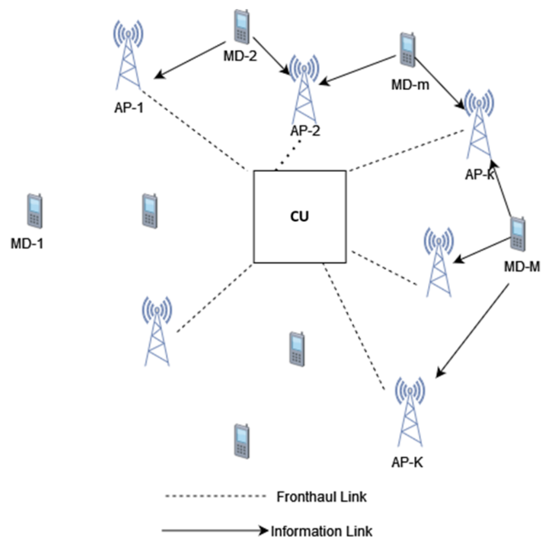

Figure 1.1: Architecture of Cell-Free Network

CF networks typically operate under Time Division Duplexing (TDD), where uplink and downlink transmissions share the same frequency band but are separated in time. This duplexing mode enables the use of channel reciprocity, allowing downlink channel state information (CSI) to be inferred from uplink measurements based on the pilot signals sent from the mobile devices (MDs).

In CF networks, the user-transmitted signals cannot be directly decoded at individual APs, as all APs concurrently serve and receive signals from all users. Instead, upon receiving the aggregated analog pilot and data signals, the APs perform appropriate processing, such as channel estimation or local combining, then quantize the resulting information and forward it to the CU via fronthaul links, where final decoding is carried out. Depending on where channel estimation is performed (at the APs or at the CU) and what is forwarded (e.g., raw quantized observations, quantized channel estimates, or locally combined data), three representative strategies are considered: Estimate Quantize, Quantize Estimate, and Decentralized Combining [1].

Unlike traditional cellular networks, where signal detection is handled independently at each base station, the CU in cell-free networks coordinates across all APs to achieve coherent reception. This centralized approach improves spectral efficiency and robustness, especially in environments with dense user deployments or non-uniform traffic loads [11]. Moreover, it supports scalable and adaptive coordination schemes such as user-centric clustering and dynamic AP selection [2], enabling efficient and flexible network management. Thus, the CU acts as the central processing hub responsible for channel estimation, signal detection, and coordination, which collectively enable the benefits of distributed access in cell-free wireless networks.

The integration of CF networks with MEC introduces new opportunities to enhance both communication and computation performance for end users. Meanwhile, it brings more challenges to timely and efficient computation offloading, since the data transmission environment of different MDs are more coupled in the CF networks.

1.3 Related Work

Edge computing has emerged as a crucial paradigm to meet the increasing demand for low-latency, computation-intensive mobile applications. Several complementary models, such as fog computing [3], cloudlets [16], and MEC [10] have been developed under the broader concept of multi-access edge computing. These paradigms enable task offloading from resource-constrained mobile devices to nearby servers, reducing latency and energy consumption. Among them, MEC has gained particular traction for its ability to support ultra-low latency services in 6G.

A major research thrust in MEC lies in optimizing task offloading strategies and resource allocation, to jointly minimize latency and energy consumption. For example, several studies have explored energy-efficient offloading under latency constraints [23], [24], while others have aimed to reduce computation delay within energy budgets [14]. Partial offloading strategies for single-user (e.g.,[21]) and multi-user (e.g.,[22])-MEC systems have been studied, focusing on the joint optimization of communication and computation resources.

More sophisticated models incorporate stochastic task arrivals and time-varying channels. For instance, dynamic offloading and resource allocation approaches using Lyapunov optimization have been proposed to balance power consumption and service delay [9], [10]. However, these centralized optimization-based methods are often limited by high signaling overheads and require full knowledge of network dynamics, making them less practical for highly dynamic environments with strict real-time requirements.

In recent years, there has been growing interest in integrating MEC with CF systems to further improve service reliability and latency. Unlike traditional cellular architectures, CF systems eliminate cell boundaries and allow all APs to cooperate in serving users, thereby enhancing spectral and energy efficiency [12]. This architectural shift opens up new possibilities for user-centric edge computing systems.

The potential of combining CF systems with MEC has been investigated in recent works, e.g.[6], which study the joint optimization of task partitioning and computational resource allocation to minimize energy consumption and execution latency. In [5] authors also explore the concept of successful edge computing probability (SECP) under stochastic network models, utilizing tools from queuing theory and stochastic geometry. However, their work assumes fixed transmission powers and static resource allocation, which may limit system adaptivity in practical and dynamic wireless environments.

Building upon this, [7] proposes a user-centric, MEC-enabled CF-framework that jointly optimizes uplink transmit power and computational resources. Their formulation aims to minimize user transmit power while ensuring fair spectral efficiency. This line of research highlights the emerging convergence of distributed radio access and edge computing to support low-latency, energy-efficient services in next-generation wireless networks.

Despite these advancements, the field is still in its early stages, particularly when it comes to scalable, learning-based methods for network resource allocation in CF-MEC systems. While existing studies have made progress, there remains considerable scope for improvement, motivating the development for adaptive frameworks capable of learning and reacting to dynamic environments.

1.4 Motivation of the Work

Recent advances in combining edge computing with distributed wireless access architectures have shown promise in enhancing service reliability and reducing latency. However, most of the existing research in this area relies on centralized optimization techniques that require complete and often static knowledge of the network state. These methods typically assume fixed user positions, deterministic channel conditions, or predefined task models, which limit their applicability in real-world deployments where user dynamics and task patterns are unpredictable.

Additionally, traditional optimization-based approaches are computationally intensive and difficult to scale with increasing network size or user density. The complexity of jointly optimizing network resource allocations for multiple users in a distributed environment such as CF networks makes real-time decision making particularly challenging.

To address these challenges, reinforcement learning (RL) offers a data-driven alternative that can adaptively learn optimal policies from interaction with the environment. Actor-critic algorithms such as Deep Deterministic Policy Gradient (DDPG) are well-suited for continuous control problems like uplink power allocation and resource scheduling. However, conventional RL models may struggle to generalize in highly dynamic environments without incorporating temporal structure or attention to spatial dependencies.

In this context, integrating Transformer architectures into the DDPG framework presents a promising direction. Transformer encoders can effectively capture spatio-temporal correlations in the observed state sequences, enabling better generalization and more informed decision-making. This motivates the development of a

Transformer-based DDPG (T-DDPG) algorithm to address the uplink resource allocation problem in CF-MEC systems, with the goal of minimizing task completion time while ensuring reliable and adaptive performance under varying conditions.

1.5 Our Contribution

This thesis focuses on the uplink resource allocation problem in a CF environment with mobile users, aiming to minimize total energy consumption of mobile devices, subject to their maximum transmission power and the task completion deadlines. We present DDPG and T-DDPG-based resource allocation algorithms specifically designed for random and time-varying channel conditions. These algorithms support low-latency task offloading, crucial for real-time edge computing.

The proposed approach forms a foundational step toward enabling time-sensitive and computation-intensive tasks for the 6G era.

Chapter 2

System Model and Problem Formulation

A cell-free network is considered as shown in Fig.2.1, where a number of MDs are connected to the CU through the APs. Let $\mathcal{M}$ and $\mathcal{K}$, respectively, be the sets of MDs and APs.

We consider, a time-division duplexing (TDD) based system. Each channel coherence interval, also referred to as a time interval for simplicity, is divided into three phases: τ_p for uplink pilot transmission, τ_u for uplink data transmission, and τ_d for downlink data transmission. Define $\tau_c = \tau_p + \tau_u + \tau_d$. Let $t = 1, 2, \ldots$ denote the index of the time intervals. For each link between MD m and AP k, the signals are subject to both large-scale fading $G_{m,k}$ and small-scale fading $h_{m,k,t}$. We assume that the small-scale fading remains constant within each time interval and changes one interval to the next [12].

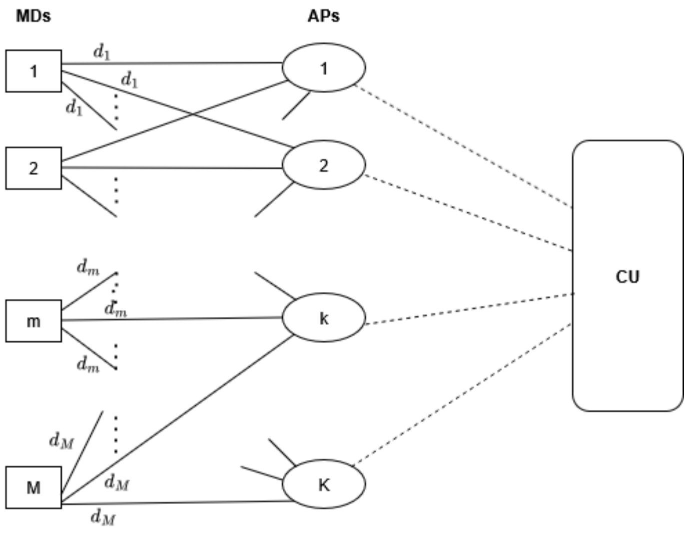

Figure 2.1: System Model

2.1 Data Transmission from the MDs

We consider an application scenario, where each MD should periodically send its application data to the edge server which is co-located at the CU for processing. One example of such applications is that the MD needs to maintain its own Digital Twin (DT) that is hosted at the edge server. Let $\bar{d}_m^{\mathrm{MD}}$ be the total amount of data that MD m should send to the edge server every time interval, and this requires $\bar{c}_m$ execution cycles at the edge server to process in order to extract features required to maintain the DT. t_d is defined as the deadline to finish processing the data from MD m at the server. We assume that the pilot transmissions result in perfect channel estimation [12]. Although this is an idealized assumption, it provides a valuable basis for future studies on the impact of imperfect channel estimation on mobile device power allocation.

Let $P_{m,t}$ be the transmission power of MD m at time interval t, where $P_{m,t} \in$

$[0, P_{m,\max}]$ with $P_{m,\max}$ the maximum transmission power of MD m.

The transmitted signals are quantized at the APs and forwarded by the APs to the CU, where the signals from individual MDs are decoded. With matched filter, the received signal-to-interference ratio (SINR) of the signal is given as below [18]:

$$\gamma_{m,t} = \frac{P_{m,t}\left[\sum_{k\in\mathcal{K}} G_{m,k}|h_{m,k,t}|^2\right]^2}{\displaystyle\sum_{n\in\mathcal{M},n\neq m} P_{n,t}\left|\sum_{k\in\mathcal{K}}\sqrt{G_{n,k}G_{m,k}}h^*_{n,k,t}h_{m,k,t}\right|^2 + \sigma^2} \tag{2.1.1}$$

Note that all simultaneous transmissions cause interference and contribute to the denominator along with the noise power σ^2 in (2.1.1). With this, the instantaneous data transmission rate of MD m at time t is given as

$$R_{m,t} = B\log_2(1+\gamma_{m,t}) \tag{2.1.2}$$

2.2 Data Receiving at the CU

Let $d^{\text{CU}}_{m,t}$ be the number of bits that the CU has received from MD m at the beginning of time interval t. We have $d^{\text{CU}}_{m,1} = 0$ and

$$d^{\text{CU}}_{m,t+1} = \min\{(d^{\text{CU}}_{m,t} + R_{m,t}\tau_u), \bar{d}^{\text{MD}}_m\} \tag{2.2.1}$$

2.3 Data Processing at the CU

Let $\bar{c}_m$ denote the maximum number of CPU cycles required to complete the processing of application data from MD m, and $c_{m,t}$ represent the remaining number of CPU cycles to be processed at the CU for MD m at the beginning of the tth time interval. We have,

$$c_{m,t+1} = \begin{cases} \bar{c}_m, & \text{if } d^{\text{CU}}_{m,t} < \bar{d}^{\text{MD}}_m \\ \left(c_{m,t} - f^{\text{CU}}_{m,t}\tau_c\right)^+, & \text{otherwise} \end{cases} \tag{2.3.1}$$

where $f^{\text{CU}}_{m,t}$ is the allocated CPU speed of the CU to process the data from MD m at time interval t and

$$\sum_m f^{\text{CU}}_{m,t} \leq f^{\text{CU}}_{\max}, \tag{2.3.2}$$

and $f^{\text{CU}}_{\max}$ is the maximum computational capacity of the CU.
We assume that, at any given time, the CU's total CPU resources are equally shared among all MDs requiring computation. That is,

$$f^{\text{CU}}_{m,t} = \frac{c_{m,t}}{\sum_{\forall m \in \mathcal{M}'} c_{m,t}} f^{\text{CU}}_{\max} \tag{2.3.3}$$

where $\mathcal{M}' = \{m | m \in \mathcal{M}, c_{m,t} > 0, d^{\text{CU}}_{m,t} = \bar{d}^{\text{MD}}_m\}$.
That means, the CU will not start processing the data until it receives all data from the MD.

2.4 Objective

Let t_m^{FIN} be the finishing time of data processing at the CU for MD m. We have

$$c_{m,t} > 0, \text{ if } t < t_m^{\text{FIN}} \tag{2.4.1}$$

$$c_{m,t} = 0, \text{ otherwise} \tag{2.4.2}$$

Our objective of this resource allocation problem is to minimize the energy consumption of the MDs, subject to the maximum transmission power of individual MDs and the task completion deadline. The above problem can be formulated as,

$$\min_{P_{m,t} \forall m,t} \sum_{t} \sum_{m \in \mathcal{M}} P_{m,t} \tag{2.4.3}$$

$$\text{s.t. } P_{m,t} \leq P_{m,\max}, \quad \forall m \in \mathcal{M}, \ \forall t \tag{2.4.4}$$

$$t_m^{\text{FIN}} \leq t_d, \quad \forall m \in \mathcal{M} \tag{2.4.5}$$

Markov Decision Process (MDP) is a good approximation of the discussed problem, as the system's evolution at each time step depends only on the current state and the selected action (i.e., the transmission power of the MDs). The state captures all relevant information, including the instantaneous channel conditions, currently received amount of data at the CU, and computation status. The wireless channel varies over time following probabilistic dynamics, allowing future channel behavior to be predicted based on current observations. The next state depends solely on the current state and action, satisfying the Markov property and enabling sequential decision-making through reinforcement learning. Next, we will first reformulate the above optimization problem into an MDP problem. We will then solve it using RL methods.

Chapter 3

Methodology

This chapter first reformulates the optimization problem in Section 2.4 into an MDP problem, and then presents the RL algorithms for solving the problem.

3.1 MDP Formulation

To effectively capture the stochastic and time-varying nature of wireless environments and computation demands, we reformulate the original optimization problem as an MDP. This framework enables learning a sequential decision policy that adapts to changes in channel conditions, task arrival patterns, and resource constraints. Unlike traditional model-based optimization techniques, the MDP-based approach allows the system to learn directly from environment interactions, without requiring full knowledge of system dynamics. This is particularly advantageous in practical scenarios where optimal decisions depend not only on current observations but also on anticipated future variations. An MDP problem includes the state space, the action space, and the state transitions.

3.1.1 State Space

The system state s_t at the begining of the time interval t, can be represented as $s_t = [h_{m,k,t}, d_{m,t}^{\text{CU}}, c_{m,t}, \ \forall\ m \in \mathcal{M}$ and $k \in \mathcal{K}]$, where $h_{m,k,t}$ is the channel state between MD m and AP k at t, $d_{m,t}^{\text{CU}} \in [0, \bar{d}_m^{MD}]$ is the total number of received bits in the CU at t from MD m, and $c_{m,t} \in [0, \bar{c}_m]$ is the remaining number of CPU cycles to be executed for MD m at t.

We have the following observations:

- For a given m, at any time interval t, $h_{m,k,t}$ is independent of $d_{m,t}^{\text{CU}}$ and $c_{m,t}$.
- For a given m, at the beginning of time interval t, if $d_{m,t}^{\text{CU}} = 0$, then $c_{m,t} = \bar{c}_m$. This happens, when an MD just started transmitting its data to the CU.
- For a given m, if $0 < d_{m,t}^{\text{CU}} < \bar{d}_m^{\text{MD}}$, then $c_{m,t} = \bar{c}_m$.This happens during the time MD m is uploading the data to the CU.
- For a given m, if $d_{m,t}^{\text{CU}} = \bar{d}_m^{MD}$, then $c_{m,t} \leq \bar{c}_m$.This happens when MD m has finished uploading the data to the CU and the CU should start (if $c_{m,t} = \bar{c}_m$) or has already started (if $c_{m,t} < \bar{c}_m$) processing the offloaded data.

3.1.2 Action Space

Let a_t be the action at time t, then $a_t = [P_{m,t}, \forall m]$, where $P_{m,t} \in [0, P_{m,\max}]$

3.1.3 State Transitions

Channel states. We consider the finite state Markov channel model, where each channel has a finite number of states, and there is a fixed and known probability for the channel to transit from one state to another, and the transition probabilities are independent of other elements in the state space. Converting the continuous channel gains into a finite number of channel states will be presented in Chapter 4.

Received data at CU:

$$d_{m,t+1}^{\mathrm{CU}} = \min\{(d_{m,t}^{\mathrm{CU}} + R_{m,t}\tau_u), \bar{d}_m^{\mathrm{MD}}\} \tag{3.1.1}$$

Remaining computation:

$$c_{m,t+1} = \begin{cases} \bar{c}_m, & \text{if } d_{m,t}^{\mathrm{CU}} < \bar{d}_m^{\mathrm{MD}} \\ \left(c_{m,t} - f_{m,t}^{\mathrm{CU}}\tau_c\right)^+, & \text{otherwise} \end{cases} \tag{3.1.2}$$

The defined MDP captures the sequential nature of the transmission power allocation problem under stochastic wireless channel conditions, which vary over time due to random fading. These dynamics are inherently difficult to model or optimize using conventional approaches. RL, however, allows an agent to learn optimal decision policies by interacting with the environment, without requiring complete knowledge of the underlying channel model.

Since power allocation decisions are made repeatedly and each action (e.g., a chosen transmission power level) affects future system performance metrics such as task completion time and energy consumption, RL is particularly well-suited for optimizing long-term cumulative rewards.

Given the continuous and high-dimensional nature of the action space in this problem, traditional discrete-action methods such as Deep Q-Networks (DQN) are not appropriate. To address this, we adopt the DDPG algorithm, which is specifically designed for continuous action spaces. To further improve performance in temporally dynamic and partially observable environments, we enhance this framework by integrating a Transformer-based encoder, resulting in the T-DDPG algorithm.

Next, we present the reward function, conventional DDPG framework, detailing its core components, including the actor–critic architecture, experience replay, target networks, and update mechanisms. We then introduce the T-DDPG extension, which incorporates a Transformer encoder into both the actor and critic networks to better capture temporal and spatial dependencies in the input sequences. An overview is provided to highlight the architectural differences and performance gains achieved through this enhancement.

3.2 Proposed Reward Function

At any given time t, the immediate reward r_t is the sum of all individual rewards $r_{m,t}$ across all MDs.

$$r_t = \sum_m r_{m,t} \tag{3.2.1}$$

where,

$$r_{m,t} = \begin{cases} \beta \cdot \dfrac{t_d - t}{t_d} - \dfrac{P_{m,t}}{P_{m,\max}}, & c_{m,t} > 0 \text{ and } t \leq t_d \\ \beta \cdot \dfrac{t_d - t}{t_d} - \alpha' - \dfrac{P_{m,t}}{P_{m,\max}}, & c_{m,t} > 0 \text{ and } t > t_d \\ 0, & c_{m,t} \leq 0 \end{cases} \tag{3.2.2}$$

The reward in (3.2.2) is designed to balance task completion time with power consumption. The first term $\beta\frac{t_d-t}{t_d}$ encourages early task completion by providing higher rewards at earlier time steps, where β controls the relative importance of finishing quickly. The second term $-\frac{P_{m,t}}{P_{m,\max}}$ penalizes excessive transmit power. If the task is not finished before the deadline, an additional penalty $-\alpha'$ is applied, ensuring that meeting the deadline is prioritized over saving power. Finally, when the task is already completed ($c_{m,t} \leq 0$), the reward is set to zero. In conclusion, this reward allows adjusting the trade-off between timeliness and energy efficiency.

3.3 Deep Deterministic Policy Gradient (DDPG)

To handle continuous action spaces, we employ DDPG as our base algorithm. DDPG is an off-policy actor-critic method, as illustrated in Figure 3.1. It is built on an actor–critic architecture, where the actor network learns a policy that generates continuous deterministic actions based on the current environment state, and the critic network learns a value function that evaluates these actions by estimating the corresponding Q-values. The critic takes both the state and action as input, while the actor uses only the state. To enhance training stability and data efficiency, DDPG employs target networks for both actor and critic, along with an experience replay buffer. Below we elaborate the DDPG algorithm given in Algorithm 1.

Initially, the target networks are synchronized with their evaluation counterparts:

$$\theta^{Q'} = \theta^{Q}, \quad \theta^{\mu'} = \theta^{\mu}$$

where θ^{Q} and θ^{μ}, respectively, denote the parameters of the critic and actor networks.

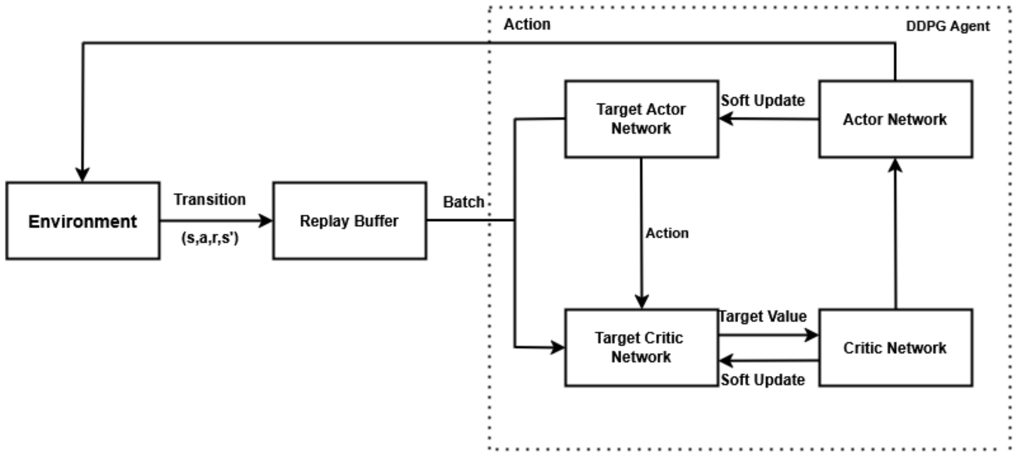

Figure 3.1: Structure of DDPG algorithm

At each time step t, the agent observes a state s_t and selects an action $a_t = \mu(s_t \mid \theta^\mu) + \mathcal{N}_t$, where $\mathcal{N}_t$ denotes exploration noise. The resulting transition (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer (Algorithm 1- line 12). Once populated, mini-batches are drawn to train the networks. Let $\mathcal{B} \subset \mathcal{D}$ denote a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) sampled from the replay buffer $\mathcal{D}$.
For critic updates, the target Q-value for each sample $i \in \mathcal{B}$ is computed as (Algorithm 1- line 18) :

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \tag{3.3.1}$$

where- $\gamma \in [0,1]$ is the discount factor, and the critic loss is:

$$L = \frac{1}{N} \sum_{i \in \mathcal{B}} \left(Q(s_i, a_i|\theta^Q) - y_i \right)^2 \tag{3.3.2}$$

The actor network is updated by maximizing the expected return $J(\theta^\mu)$, defined as:

$$J(\theta^\mu) = \mathbb{E}_{s \sim \rho^\mu} \left[Q(s, \mu(s|\theta^\mu)) \right], \tag{3.3.3}$$

where ρ^μ is the state distribution induced by the policy μ. The deterministic policy gradient is approximated using mini-batches (Algorithm 1- line 19):

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_{i \in \mathcal{B}} \nabla_a Q(s_i, a|\theta^Q)\big|_{a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s_i|\theta^\mu) \tag{3.3.4}$$

To ensure stable learning, the target networks are updated using soft updates (algorithm 1- line 20):

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}, \quad \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{3.3.5}$$

where $\tau \in (0, 1)$ denotes the soft update coefficient.

Algorithm 1 Conventional Deep Deterministic Policy Gradient (DDPG)

```
1:  Initialize actor learning rate $\alpha_a$, critic learning rate $\alpha_c$
2:  Set discount factor $\gamma$, soft update rate $\tau$
3:  Initialize actor network $\mu(s|\theta^{\mu})$, critic network $Q(s, a|\theta^{Q})$
4:  Initialize target networks $\mu'(s|\theta^{\mu'})$, $Q'(s, a|\theta^{Q'})$ with weights $\theta^{\mu'} \leftarrow \theta^{\mu}$, $\theta^{Q'} \leftarrow \theta^{Q}$
5:  Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
6:  for each episode do
7:    Initialize environment and receive initial state $s_0$
8:    Set $t \leftarrow 0$
9:    while not done do
10:     Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ according to the current policy and exploration noise
11:     Execute action $a_t$, observe reward $r_t$, next state $s_{t+1}$, and done flag $d_t$
12:     Store transition $(s_t, a_t, r_t, s_{t+1}, d_t)$ into $\mathcal{D}$
13:     if enough samples in $\mathcal{D}$ then
14:       Sample mini-batch $\mathcal{B} \subset \mathcal{D}$, where $\mathcal{B} = \{(s_i, a_i, r_i, s_{i+1}, d_i)\}_{i=1}^{N}$
15:       for each $i \in \mathcal{B}$ do
16:         Compute target Q-value:
                $$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$$
17:       end for
18:       Update critic using (3.3.2)
19:       Update actor using (3.3.4)
20:       Soft update target networks:
                $$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \quad \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$
21:     end if
22:     $s_t \leftarrow s_{t+1}$
23:     $t \leftarrow t + 1$
24:   end while
25: end for
```

3.4 Transformer-Based DDPG (T-DDPG)

While the basic DDPG algorithm effectively addresses continuous action spaces, it primarily relies on the current state to make decisions, which may limit its performance in partially observable or highly dynamic environments. To overcome this limitation, we extend DDPG by incorporating Transformer encoders into both the actor and critic networks. This enhanced architecture, termed T-DDPG, is designed to capture spatial and temporal dependencies by processing sequences of past states rather than relying solely on the current observation. The architecture of the T-DDPG algorithm is given in Figure 3.2.

Unlike the basic DDPG, the actor network in T-DDPG, $\mu(s_{1:T}|\theta^{\mu})$, receives a sequence of past states $s_{1:T}$, the sequence of states are first embedded and passed through positional encodings. The Transformer encoder processes this sequence to extract spatial-temporal aware features, capturing both temporal dependencies across past states and spatial dependencies across nodes or links. These enriched features are then used to generate the deterministic action. Similarly, the critic network $Q(s_{1:T}, a|\theta^{Q})$ utilizes the same encoded sequence and concatenates it with the action to estimate the Q-value.

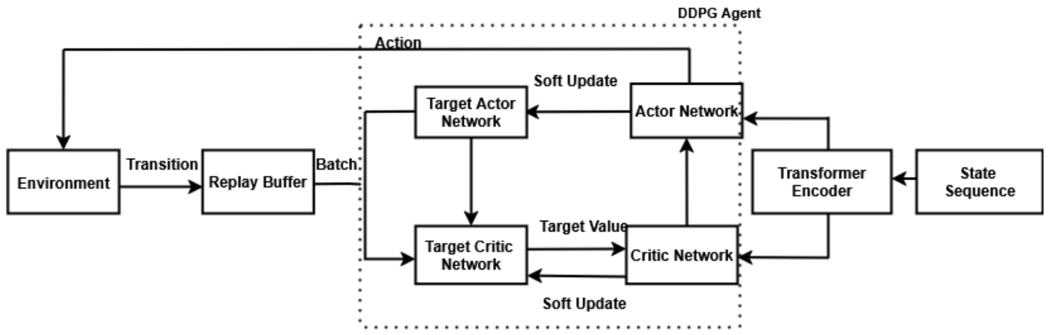

Figure 3.2: Architecture of T-DDPG algorithm

At each time step t, the environment state sequence $s_{1:T}$ is updated via a sliding window. The agent selects an action using the actor:

$$a_t = \mu(s_{1:T}|\theta^{\mu}) + \mathcal{N}_t$$

where $\mathcal{N}_t$ is a temporally correlated exploration noise. The environment returns the reward r_t, next state s_{t+1}, and termination flag d_t, and the transition tuple $(s_{1:T}, a_t, r_t, s'_{1:T}, d_t)$ is stored in a replay buffer (Algorithm 2- line 17).

During training, a batch of transitions is sampled from the buffer. The target Q-value for each sample is computed as:

$$y_i = r_i + \gamma Q'(s'_{1:T,i}, \mu'(s'_{1:T,i}))$$

where Q' and μ' are the target critic and actor networks. The critic is updated by minimizing the mean squared error loss as in (3.3.2) (Algorithm 2- line 25). The actor is updated by maximizing the expected Q-value as mentioned in (3.3.4) (Algorithm 2- line 26). Soft target updates are performed after each training step as in the DDPG framework (Algorithm 2- line 27).

By modeling sequences with self-attention, T-DDPG can focus on the most relevant historical information. This sequence-aware design enhances the agent's ability to reason over delayed effects, track temporal patterns, and generate more robust decisions in dynamic wireless environments where instantaneous observations may be insufficient.

Algorithm 2 Transformer-based Deep Deterministic Policy Gradient (Transformer-DDPG)

```
1:   Initialize actor learning rate $\alpha_a$, critic learning rate $\alpha_c$
2:   Set discount factor $\gamma$, soft update rate $\tau$
3:   Initialize Transformer-based actor $\mu(s_{1:T}|\theta^{\mu})$, critic $Q(s_{1:T}, a|\theta^{Q})$
4:   Initialize target networks $\mu', Q'$ with $\theta^{\mu'} \leftarrow \theta^{\mu}, \theta^{Q'} \leftarrow \theta^{Q}$
5:   Initialize replay buffer $\mathcal{D} \leftarrow \emptyset$
6: **for** each episode **do**
7:   Initialize environment and state sequence $s_{1:T}$
8:   Set $t \leftarrow 0$
9:   **while** not done **do**
10:    **TransformerEncoder Forward Pass:**
11:        Embed state: $x \leftarrow \text{Linear}(s_{1:T}) + PE[:, :T, :]$
12:        Encode: $z \leftarrow \text{Transformer}(x)$
13:        Select last time step: $z \leftarrow z[:, -1, :]$
14:    **Actor Forward:** $a_t \leftarrow \text{sigmoid}(Wz + b) \cdot \text{max\_action} + \mathcal{N}_t$
15:    Execute action $a_t$, observe $r_t, s_{t+1}, d_t$
16:    Update sequence: $s'_{1:T} \leftarrow \text{shift-append}(s_{1:T}, s_{t+1})$
17:    Store transition $(s_{1:T}, a_t, r_t, s'_{1:T}, d_t) \in \mathcal{D}$
18:    **if** $|\mathcal{D}| \geq N$ **then**
19:      Sample mini-batch $\mathcal{B} = \{(s_i, a_i, r_i, s'_i, d_i)\}_{i=1}^{N}$
20:      **for** each $i \in \mathcal{B}$ **do**
21:        **Critic Target:**
22:          $a'_i \leftarrow \mu'(s'_i)$
23:          $y_i \leftarrow r_i + \gamma Q'(s'_i, a'_i)$
24:      **end for**
25:      Update critic using (3.3.2)
26:      Update actor using (3.3.4)
27:      Soft update target networks:
             $$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}, \quad \theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$
28:    **end if**
29:    $s_{1:T} \leftarrow s'_{1:T}$
30:    $t \leftarrow t + 1$
31:  **end while**
32: **end for**
```

3.5 Comparison Between DDPG and T-DDPG

The key distinction between DDPG and T-DDPG lies in how temporal dependencies are treated within the actor and critic networks.

In DDPG, decisions are made based solely on the current state and action, processed through feedforward layers. This approach lacks temporal modeling and processes each transition independently, limiting performance in environments where the history of states influences optimal actions.

Conversely, T-DDPG leverages Transformer encoders to process sequences of past states, allowing it to learn from context and capture complex dependencies over time. The self-attention mechanism provides a dynamic weighting of prior information, which enhances both action generation and Q-value estimation.

Figure 3.1 through Figure 3.2 visually illustrate these differences. T-DDPG's architecture enables richer policy learning by incorporating structured temporal features, offering clear advantages in environments with evolving dynamics, such as edge computing networks with variable channel conditions and computation loads.

Chapter 4

Simulation Setup and Evaluation

This chapter presents the simulation setup and the results of evaluating the proposed T-DDPG framework under different channel models. First, the simulation environment and key parameters are described. Then, performance outcomes are analyzed separately for the 2-state and 3-state finite state Markov channel (FSMC) cases.

4.1 Simulation Setup

We consider a wireless communication system where M MDs and K APs are randomly deployed within a $500\,\mathrm{m} \times 500\,\mathrm{m}$ cell-free network area, with their positions sampled from a uniform distribution. The Euclidean distance between the mth MD and the kth AP, denoted by $d_{m,k}$, is used to compute the path loss. For simplicity, we consider only the path loss component of the large-scale fading, denoted as $G_{m,k}$. The path loss is modeled as:

$$G_{m,k} = G^0_{m,k} \cdot \left(\frac{d_{m,k}}{d_0}\right)^{-\alpha} \tag{4.1.1}$$

where $G^0_{m,k}$ is the path loss at a reference distance d_0, and α denotes the path loss exponent. The overall channel gain is given as,

$$H_{m,k,t} = G_{m,k} \cdot (h_{m,k,t})^2 \tag{4.1.2}$$

where $h_{m,k,t}$ denotes the small-scale fading component, modeled using Rician distribution.

The values of $H_{m,k,t}$ may have high variability, especially in environments with diverse node placements and dynamic fading conditions. As a result, directly using continuous-valued channel gains can lead to training instability. To address this and enhance both training stability and generalization capability, the continuous channel gains are discretized into a fixed number of representative discrete levels using a linear quantization scheme, and the original wireless channels are modeled as a FSMC. To model the original channel into an N-states FSMC, $N - 1$ thresholds are needed to divide the entire range of channel gain values into N intervals, each of which is mapped to one channel state. In addition, transition probabilities for the channel to switch from one state to another should also be found. Details of these are available in [19].

The default simulation parameters and simulation hyperparameters are summarized below in Tables 4.1 - 4.3. Some of the parameters in Table 4.1 have been borrowed from [18].

Table 4.1: Default Simulation Parameters

Notation	Parameter	Value
f_{max}^{CU}	Maximum computational capacity of the CU	$100GHz$
N_{cpb}	Number of cycles to process one bit task	500 cycles/bit
B	Total system bandwidth	$5MHz$
$P_{m,max}$	Maximum transmission power	$0.5W$
$\bar{d}_m^{\mathrm{MD}}$	Task size	$1Mbits$
t_d	Task completion deadline	$4ms$
α'	Penalty parameter	5
β	Scaling factor	1.0
d_0	Reference distance for pathloss	$10m$
α	pathloss exponent	3
σ^2	Power of the AWGN	$2 \times 10^{-14}W$

Table 4.2: Simulation Hyperparameter for DDPG Training

Hyperparameter	Value
Dicount factor,γ	0.99
Actor learning rate	0.0001
Critic learning rate	0.0001
Soft update rate	0.005
Batch size	64
Replay buffer size	10^6
Exploration noise	0.1
Actor-Critic layer architecture	2 layers, 256 neurons/layer

Table 4.3: Simulation Hyperparameter for Transformer

Hyperparameter	Value
Number of Transformer encoder layer	3
Size of embedding layer	256
Number of heads in multi head attention	4
Dimension of the feed forward network	512

4.2 Simulation Results: 2-state FSMC Case

First, the wireless channel is modeled as a two-state FSMC. To evaluate the effectiveness of the proposed reward function in equ. (3.2.2), it is compared against three alternative reward functions. At any given time t, the reward of MD m can be defined as $r_{m,t}$.

Reward Function 1:

$$r_{m,t} = \begin{cases} -1, & c_{m,t} \leq 0 \text{ and } t \leq t_d \\ -10 \cdot \dfrac{t - t_d}{t_d}, & c_{m,t} \leq 0 \text{ and } t > t_d \\ -.01, & \text{otherwise} \end{cases} \tag{4.2.1}$$

This reward applies a fixed small penalty -1 for completing the task before the deadline, and a harsher penalty that increases linearly with delay if completion occurs after the deadline. This reward function introduces a discontinuity at the deadline, where the reward abruptly shifts from a constant value (-1) to a linearly decreasing function. This sudden change introduces noise in the attention mechanism of sequence-based models like Transformer-DDPG, hindering their ability to learn temporal dependencies.

Reward Function 2:

$$r_{m,t} = \begin{cases} -t - \frac{P_{m,t}}{P_{m,\max}}, & c_{m,t} > 0 \\ 0, & c_{m,t} \leq 0 \end{cases} \tag{4.2.2}$$

This reward penalizes the agent proportionally to the elapsed time and the normalized transmit power as long as the task remains unfinished, while assigning zero reward upon completion. Such a formulation naturally encourages earlier completion, since prolonging execution leads to increasingly negative returns. However, the absence of an explicit deadline-aware component limits its effectiveness in time-sensitive scenarios.

Reward Function 3:

$$r_{m,t} = \begin{cases} 10 \cdot \dfrac{t_d - t}{t_d} - \frac{P_{m,t}}{P_{m,\max}}, & c_{m,t} > 0 \text{ and } t \leq t_d \\ -1.0 - \frac{P_{m,t}}{P_{m,\max}}, & c_{m,t} > 0 \text{ and } t > t_d \\ 0, & c_{m,t} \leq 0 \end{cases} \tag{4.2.3}$$

This reward encourages early task completion by providing a higher positive return when tasks are completed well before the deadline and gradually reducing the reward as time progresses. Once the deadline is missed, it imposes a fixed negative penalty along with the power cost, thereby explicitly discouraging late completions.

Compared to the above three reward functions, the proposed reward offers a smooth and continuous formulation, maintaining linear decay across time and ensuring a gradual transition at the deadline. This provides more stable gradient signals, improving learning in sequence-aware architectures.

Comparison between the proposed reward function and the above three rewards is shown in Figure 4.1. The results show that, for all four reward functions, the average task completion time increases with the number of MDs due to increased traffic load in the network. For each reward function, T-DDPG outperforms DDPG. In addition, for both DDPG and T-DDPG, the proposed reward function consistently achieves lower task completion times than the other three reward functions.

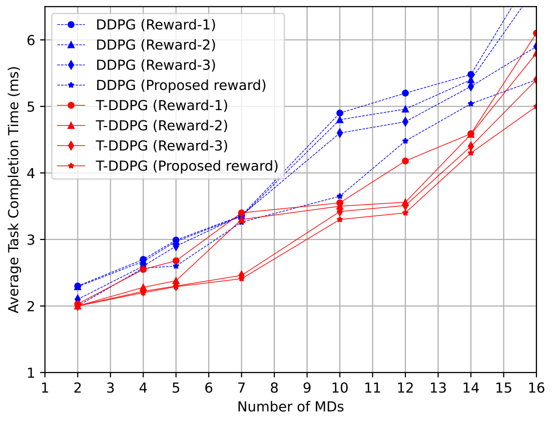

Figure 4.1: Average task completion time of DDPG & T-DDPG using different rewards (number of APs = number of MDs)

For the remaining part of this chapter, we only show the results based on the proposed reward function. The learning curves of both the DDPG and T-DDPG agents are illustrated in the Fig. 4.2 and Fig. 4.3, respectively. The DDPG agent stabilizes after approximately 400 episodes, whereas T-DDPG converges within around 150 episodes, indicating that T-DDPG achieves faster convergence.

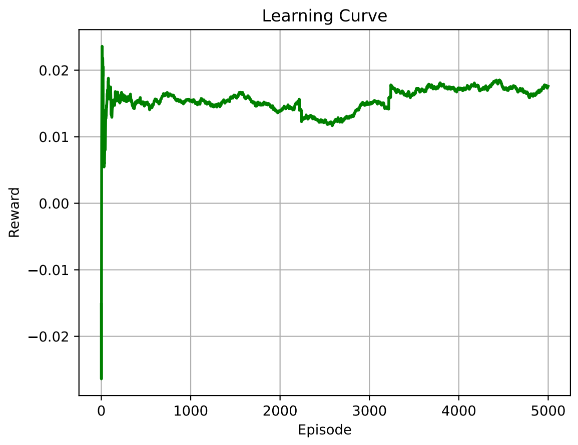

Figure 4.2: Learning curve of DDPG agent

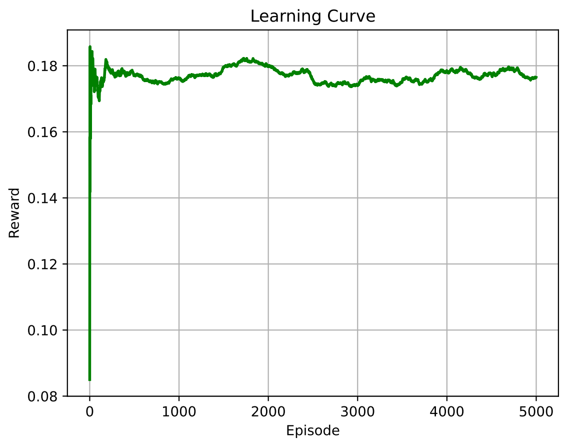

Figure 4.3: Learning curve of T-DDPG agent

Fig. 4.4 shows the performance comparison between the DDPG and T-DDPG agent in terms of average task completion time over the number of MDs. As expected, both agents experience increased task completion time as MD density rises, due to increased computational load. Meanwhile, T-DDPG consistently achieves lower average completion times than DDPG. This demonstrates T-DDPG's ability to better capture temporal dependencies and dynamic interference patterns via the Transformer architecture, allowing it to make more informed power allocation decisions.

Fig. 4.5 shows the total energy consumption of both agents under increasing MD load. The DDPG agent shows a sharper increase in energy usage with more MDs, reaching up to around 0.048 J for 16 devices. In contrast, the T-DDPG agent maintains significantly lower energy consumption throughout, rising more gradually and consuming around 0.02 J at 16 MDs. This improvement highlights T-DDPG's superior resource efficiency, as it optimizes transmission power more effectively by utilizing historical channel and traffic information through its attention-based mechanism.

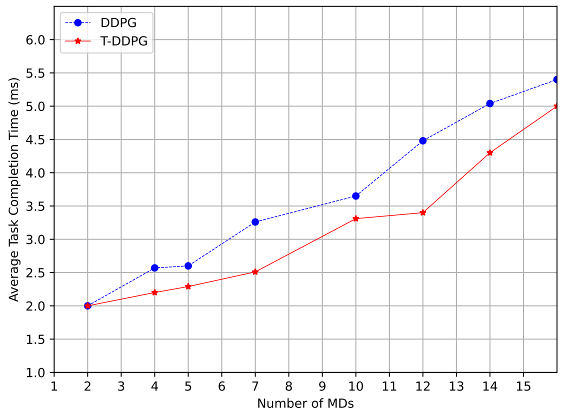

Figure 4.4: Average task completion time of DDPG and T-DDPG in 2-state channel condition (number of APs = number of MDs)

The percentage of task completion within the deadline on MD counts is shown in Fig. 4.6. This is a crucial performance metric for time sensitive applications. While

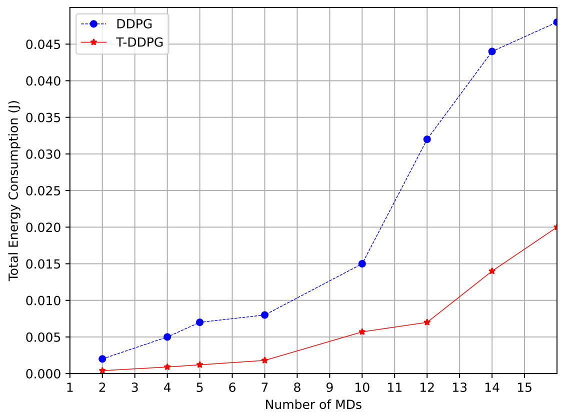

Figure 4.5: Total energy consumption of DDPG and T-DDPG in 2-state channel condition (number of APs = number of MDs)

DDPG begins with a high completion rate at low MD counts, its performance degrades rapidly with increasing device density, 2.5% at 14 MDs and falling below 1% at 16 MDs. In contrast, T-DDPG maintains a robust completion rate even in congested scenarios, achieving over 65% at 14 MDs and 20.5% deadline adherence at 16 MDs. This significant improvement reflects the Transformer's capacity to anticipate and adapt to time-varying system dynamics, ultimately enabling better compliance with delay constraints under high load conditions.

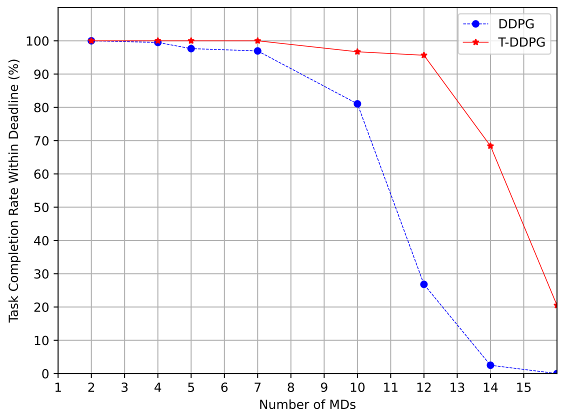

Figure 4.6: Percentage of task completion within deadline of DDPG and T-DDPG in 2-state channel condition (number of APs = number of MDs)

Now, for a network comprising 10 MDs and 10 APs, the performance of the DDPG and T-DDPG algorithms is evaluated in terms of average task completion time, total energy consumption and the percentage of deadline-compliant tasks under varying numbers of transmitted bits in Figs. 4.7 to 4.9. In Fig. 4.7, as the number of transmitted bits increases from 10 to 50 Mbits, the average task completion time rises for both agents. However, T-DDPG achieves up to a 34.9% reduction in task completion time compared to DDPG at 50 Mbits, demonstrating its superior ability to handle larger data transmissions.

Fig. 4.8 presents the total energy consumption with respect to the number of transmitted bits. As the transmitted data increases, both DDPG and T-DDPG agents exhibit higher energy consumption. However, T-DDPG achieves 88.9% lower energy usage than DDPG at maximum bit loads, highlighting its more efficient power allocation strategy.

The task completion rate within the deadline under varying transmission bit for DDPG and the T-DDPG agent is illustrated in Fig. 4.9. As the number of transmitted bits increases, the completion rate declines for both agents. For DDPG, the rate drops sharply to 0% at 20 Mbits and remains at 0% beyond that. In contrast, T-DDPG maintains significantly higher completion rates, achieving 93.6% at 20 Mbits and 70.4% at 30 Mbits, demonstrating its robustness in meeting quality-of-service (QoS) requirements under increased data loads.

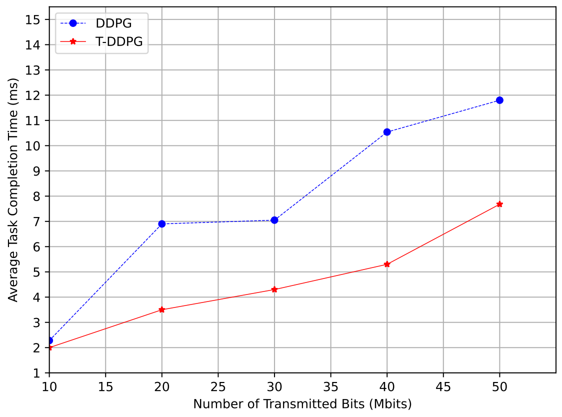

Figure 4.7: Average task completion time vs. transmitted bits in 2-state channel condition for DDPG and T-DDPG

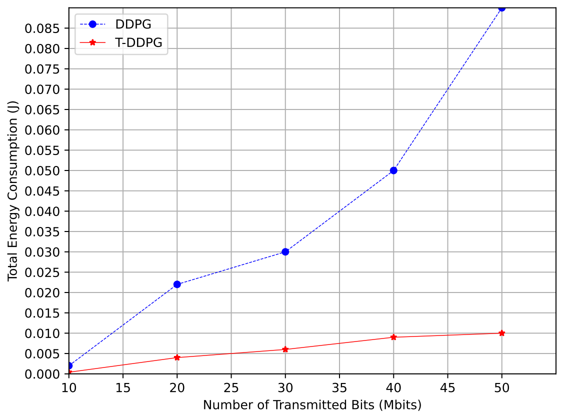

Figure 4.8: Total energy consumption vs. transmitted bits in 2-state channel condition for DDPG and T-DDPG

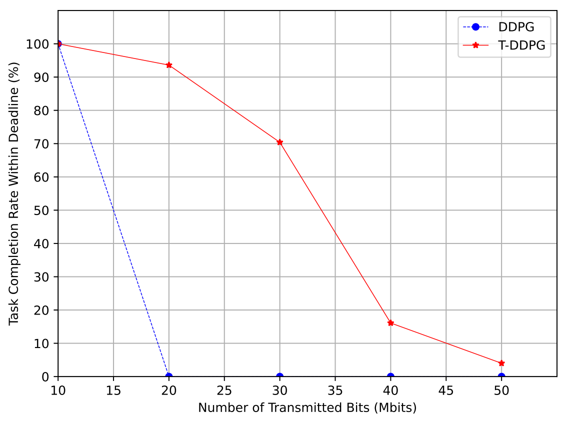

Figure 4.9: Task completion rate within deadline vs. transmitted bits in 2-state channel condition for DDPG and T-DDPG

4.2.1 Impact of Reward Parameters (β, α') on Performance

Fig. 4.10 and Fig. 4.11 show the influence of the reward parameters β and α' on the trade-off between energy efficiency and task completion rate. For $\beta = 1, \alpha' = 5$, energy consumption remains relatively low, but DDPG exhibits a sharp decline in completion success under high load. In contrast, T-DDPG maintains above 95% success up to 12 MDs with only modestly higher energy, highlighting its ability to capture temporal and spatial dynamics. Increasing the penalty to $\alpha' = 12$ further improves deadline satisfaction for both methods, with T-DDPG sustaining over 80% success at 14 MDs, though at the cost of 20-30% higher energy.

At the lower shaping weight $\beta = 0.25$, overall success rates are weaker, as the reduced time-shaping incentive limits early task completion. Nevertheless, T-DDPG still consistently outperforms DDPG across all cases. In summary, larger α' values improve completion rate but increase energy usage, while a stronger shaping factor β provides a more balanced trade-off without requiring excessively high penalties.

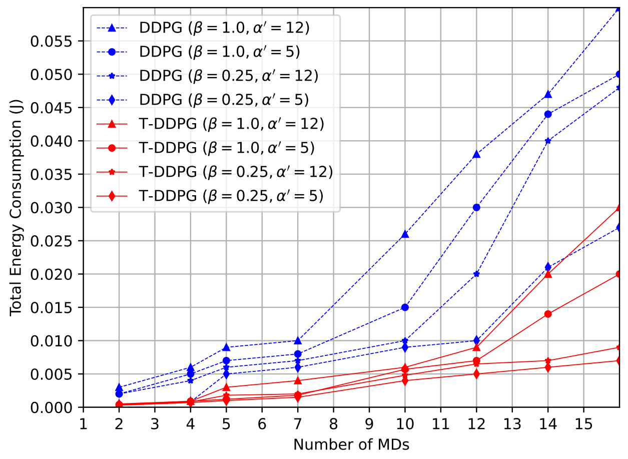

Figure 4.10: Impact of reward parameters on the total energy consumption of DDPG and T-DDPG in 2-state channel condition (number of APs = number of MDs)

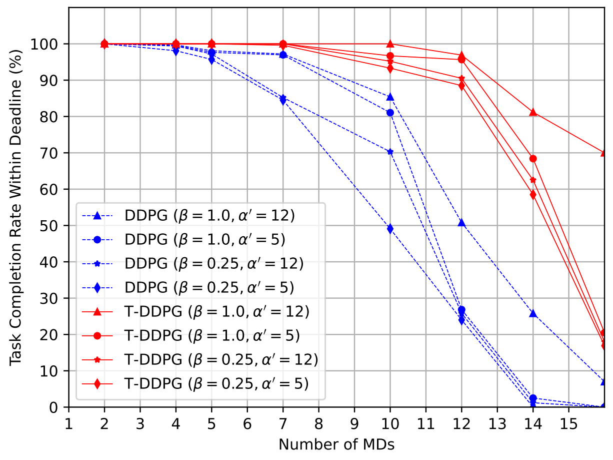

Figure 4.11: Impact of reward parameters on the percentage of task completion of DDPG and T-DDPG in 2-state channel condition (number of APs = number of MDs)

4.3 Simulation Results: 3-state FSMC Case

In the second step, the wireless channel is modeled as a three-state FSMC. To assess the behavior of the DDPG and T-DDPG agents in this setting, the same set of evaluation metrics used in the two-state FSMC case is considered.

The average task completion time over the MDs for 3-state channel is presented in Fig. 4.12. It can be seen that the rising trend for both agents are same as the Fig. 4.4. However, DDPG performs slightly worse than in the 2-state case due to the added variability, while T-DDPG performs better, benefiting from temporal attention to exploit richer channel state transitions.

Fig. 4.13 illustrates the total energy consumption under varying number of MDs for 3-state channel. Here, DDPG's energy consumption rises slightly compared to the 2-state case, especially for larger MD counts. In contrast, T-DDPG consumes even less energy in the 3-state case for most MD counts.

DDPG and T-DDPG's task completion rate within the deadline is shown in Fig. 4.14. It can be seen, DDPG's rate drops quickly as the number of MDs increases, falling below 30% at 12 MDs. On the other hand, T-DDPG maintains a much higher completion rate, above 98% up to 12 MDs and still over 80% at 14 MDs. This performance advantage is achieved while also consuming less energy than DDPG across all MD counts, shown in Fig. 4.13.

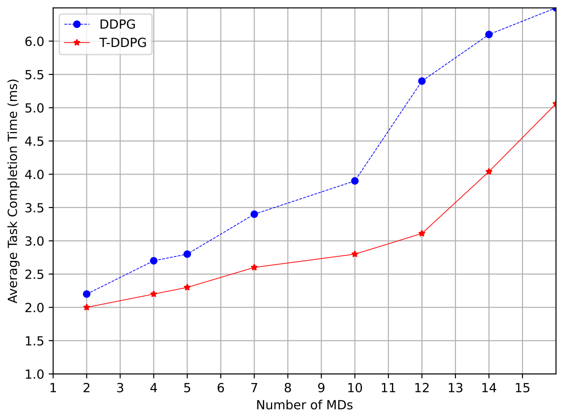

Figure 4.12: Average task completion time of DDPG and T-DDPG in 3-state channel condition (number of APs = number of MDs)

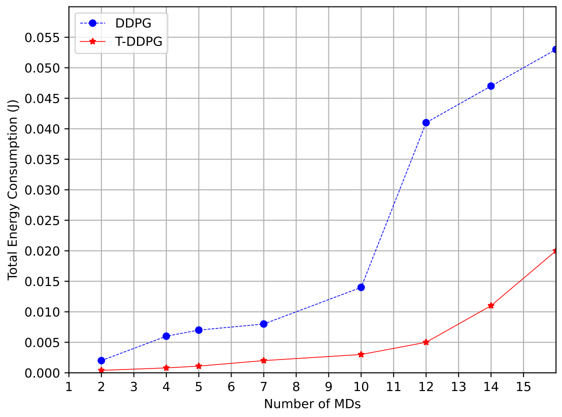

Figure 4.13: Total energy consumption of DDPG and T-DDPG in 3-state channel condition (number of APs = number of MDs)

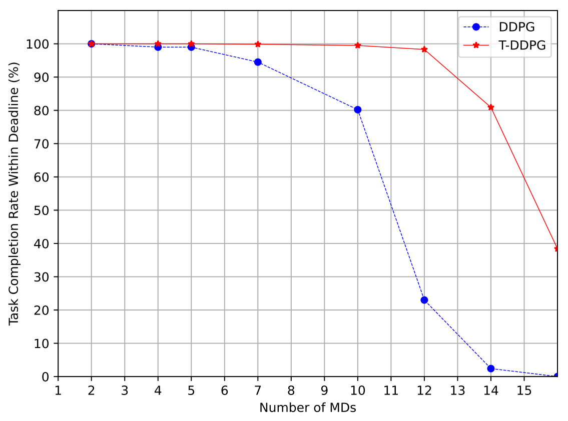

Figure 4.14: Percentage of task completion within deadline of DDPG and T-DDPG in 3-state channel condition (number of APs = number of MDs)

Figure 4.15 shows the average task completion time of DDPG and T-DDPG for different bit counts in a 3-state channel model with 10 MDs and 10 APs. The 3-state channel causes a slight increase in DDPG's completion time compared to the 2-state model (Fig. 4.7), due to greater channel variability. In contrast, T-DDPG maintains stable performance.

As shown in Fig. 4.16, DDPG's energy consumption increases with the number of transmitted bits and is higher in the 3-state channel compared to the 2-state model (Fig. 4.8). However, T-DDPG maintains low energy usage even in these conditions.

Fig. 4.17 shows that DDPG's task completion rate drops sharply from 100% at 10 Mbits to 0% beyond 20 Mbits. In contrast, T-DDPG maintains higher rates, 94.7% at 20 Mbits, 75.35% at 30 Mbits and still 24.75% at 40 Mbits. This indicates that temporal attention helps T-DDPG better handle increased channel complexity and larger data loads, ensuring more tasks complete within the deadline.

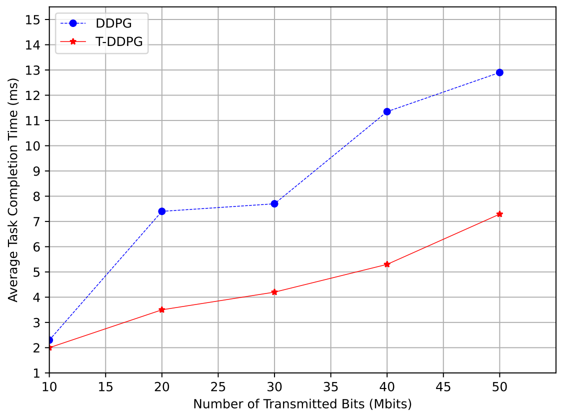

Figure 4.15: Average task completion time vs. transmitted bits in 3-state channel condition for DDPG and T-DDPG

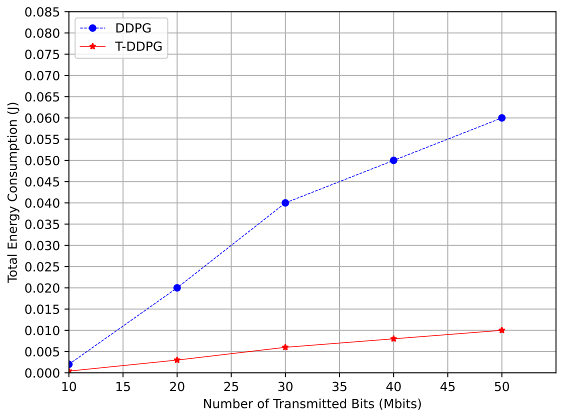

Figure 4.16: Total energy consumption vs. transmitted bits in 3-state channel condition for DDPG and T-DDPG

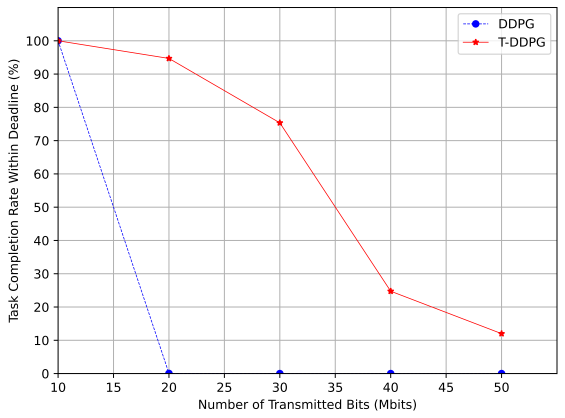

Figure 4.17: Task completion rate within deadline vs. transmitted bits in 3-state channel condition for DDPG and T-DDPG

4.3.1 Impact of Reward Parameters (β, α') on Performance

When extending the evaluation from a 2-state FSMC to a 3-state FSMC, both DDPG and T-DDPG exhibit consistent trends, but with notable differences which can be seen in Fig. 4.18 and Fig. 4.19. In a 3-state case, a slight increase in total energy consumption for both DDPG and T-DDPG can be noticed in Fig. 4.18. This is because the additional medium channel state leads the agents to allocate power more conservatively compared to the binary channel representation. However, T-DDPG consistently consumes less energy than DDPG, showing better adaptation across finer channel variations.

In addition, the 3-state FSMC significantly improves the task completion rate compared to the 2-state case, especially for the T-DDPG, shown in Fig. 4.19. With an extra state capturing intermediate channel quality, the agent can better balance transmission scheduling, avoiding unnecessary deadline violations. T-DDPG consistently maintains a higher success rate, while DDPG degrades faster as MDs increase.

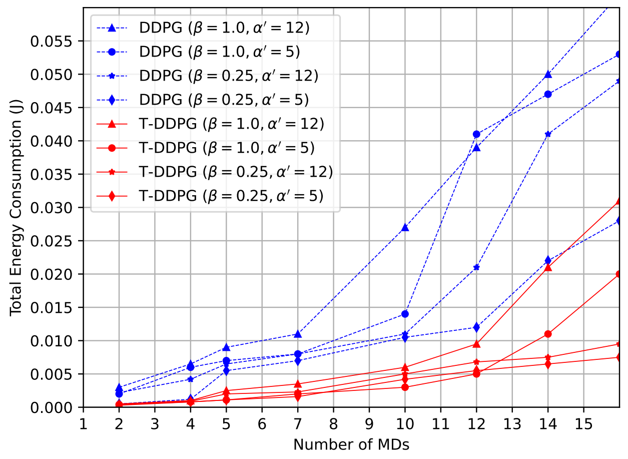

Figure 4.18: Impact of reward parameters on the total energy consumption of DDPG and T-DDPG in 3-state channel condition (number of APs = number of MDs)

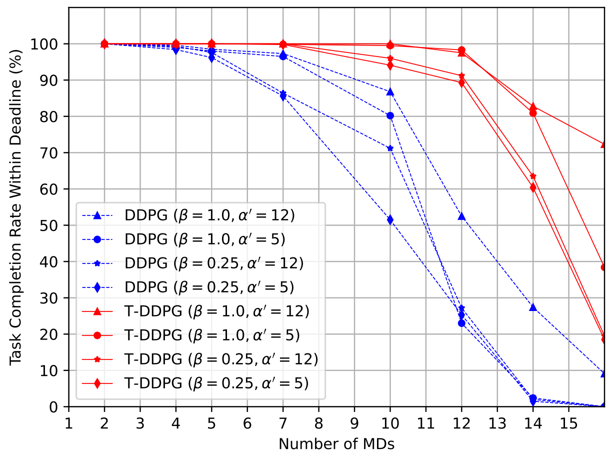

Figure 4.19: Impact of reward parameters on the percentage of task completion of DDPG and T-DDPG in 3-state channel condition (number of APs = number of MDs)

This chapter introduced a simulation framework to evaluate the performance of DDPG and T-DDPG for time constrained task completion in CF networks. The model incorporated distance dependent path loss, small-scale Rician fading, and quantized channel gains under various channel state scenarios.

The simulation environment reflected practical wireless deployment conditions, including transmission power limits, task generation processes, and dynamic channel variations. Relevant system and learning parameters were summarized, and performance was assessed using average task completion time, total energy consumption, and the percentage of tasks completed within the deadline.

Simulation results demonstrated that T-DDPG consistently outperformed DDPG across different channel conditions. The use of quantized channel gain modeling reduced learning complexity, enhanced training stability and accelerated convergence. While the 2-state and 3-state FSMC models provided clear insights into system performance, the framework can also be extended to higher-state models. Increasing the number of channel states enables a finer representation of fading dynamics, potentially improving decision accuracy and prediction reliability. However, this comes at the expense of higher system complexity, including more elaborate transition probability estimation and increased computational overhead, which may limit scalability in real-time scenarios.

Together, these results confirm that the T-DDPG agent surpasses standard DDPG in energy efficiency and deadline adherence, particularly in dense and dynamic network environments, making it a promising approach for real-time wireless resource allocation. Overall, the findings highlight the effectiveness of Transformer-based reinforcement learning for wireless resource allocation and underscore the importance of realistic channel modeling in developing robust and scalable learning agents for dynamic communication environments.

Chapter 5

Conclusion

This thesis presented a Transformer-enhanced DDPG (T-DDPG) framework for resource allocation in CF-MEC networks, aiming to reduce the energy consumption of mobile devices, subject to the maximum transmission power of the devices and the task completion deadlines in dynamic wireless environments. Traditional DDPG struggles to exploit temporal and spatial patterns, which we addressed by integrating spatial-temporal Transformers into both the actor and critic networks. This allows the agent to better leverage historical context and adapt to time-varying channel conditions, modeled using path loss and Rician fading.

We discretized the channel dynamics using a FSMC model to simulate different levels of channel granularity. A custom simulation environment was developed to assess performance under different numbers of MDs and task sizes. Results showed that T-DDPG outperformed DDPG across all metrics, achieving higher task success rates and significantly reducing energy consumption, especially under heavier traffic and finer-grained channel models.

These results demonstrate the suitability of T-DDPG for real-world applications like Digital Twins, augmented reality and mission-critical IoT, where real-time and reliable uplink scheduling is essential. Nevertheless, certain limitations persist. The proposed framework relies on offline training using simulated trajectories, which may not fully capture the challenges of real-time deployment. Additionally, the computational complexity of the Transformer architecture could pose scalability challenges in large-scale networks.

Future research may focus on incorporating online learning capabilities, extending the framework to multi-agent settings, and employing adaptive or learnable channel models to improve both performance and practical applicability.

In summary, this thesis shows that combining deep reinforcement learning with spatial-temporal Transformers enables robust and efficient resource allocation in CF-enabled edge networks, making it a promising direction for next-generation wireless systems.

Bibliography

[1] M. Bashar, H. Q. Ngo, K. Cumanan, A. G. Burr, P. Xiao, E. Björnson, and E. G. Larsson. Uplink spectral and energy efficiency of cell-free massive mimo with optimal uniform quantization. *IEEE Transactions on Communications*, 69 (1):223–245, 2020.

[2] E. Björnson, G. Interdonato, and E. G. Larsson. Scalable cell-free massive mimo systems. *IEEE Transactions on Communications*, 68(7):4247–4261, 2020.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[4] C. De Alwis, A. Kalla, Q.-V. Pham, P. Kumar, K. Dev, W.-J. Hwang, and M. Liyanage. Survey on 6g frontiers: Trends, applications, requirements, technologies and future research. *IEEE Open Journal of the Communications Society*, 2:836–886, 2021.

[5] G. Femenias and F. Riera-Palou. Performance analysis of edge computing-enabled cell-free massive mimo wireless networks. *IEEE Access*, 8:139227–139245, 2020. doi: 10.1109/ACCESS.2020.3011134.

[6] G. Femenias and F. Riera-Palou. Mobile edge computing aided cell-free massive mimo networks. *IEEE Transactions on Mobile Computing*, 23(2):1246–1261, 2022.

[7] G. Interdonato and S. Buzzi. Joint optimization of uplink power and computational resources in mobile edge computing-enabled cell-free massive mimo. *IEEE Transactions on Communications*, 2023.

[8] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten. The road towards 6g: A comprehensive survey. *IEEE Open Journal of the Communications Society*, 2: 334–366, 2021.

[9] Y. Mao, J. Zhang, S. Song, and K. B. Letaief. Power-delay tradeoff in multi-user mobile-edge computing systems. In *2016 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[10] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials*, 19(4):2322–2358, 2017.

[11] E. Nayebi, A. Ashikhmin, T. L. Marzetta, and H. Yang. Performance of cell-free massive mimo systems with mmse and lsfd receivers. *Asilomar Conference on Signals, Systems, and Computers*, 2016.

[12] H. Q. Ngo, A. Ashikhmin, H. Yang, E. G. Larsson, and T. L. Marzetta. Cell-free massive mimo versus small cells. *IEEE Transactions on Wireless Communications*, 16(3):1834–1850, 2017.

[13] T. H. Noor, S. Zeadally, A. Alfazi, and Q. Z. Sheng. Mobile cloud computing: Challenges and future research directions. *Journal of Network and Computer Applications*, 115:70–85, 2018.

[14] J. Ren, G. Yu, Y. Cai, and Y. He. Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 17(8):5506–5519, 2018.

[15] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li. A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys (CSUR)*, 52 (6):1–36, 2019.

[16] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[17] C. Sun, X. Wu, X. Li, Q. Fan, J. Wen, and V. C. Leung. Cooperative computation offloading for multi-access edge computing in 6g mobile networks via soft actor critic. *IEEE Transactions on Network Science and Engineering*, 2021.

[18] F. D. Tilahun, A. T. Abebe, and C. G. Kang. Multi-agent reinforcement learning for distributed resource allocation in cell-free massive mimo-enabled mobile edge computing network. *IEEE Transactions on Vehicular Technology*, 2023.

[19] H. S. Wang and N. Moayeri. Finite-state markov channel: A useful model for radio communication channels. *IEEE Transactions on Vehicular Technology*, 44 (1):163–171, Feb. 1995. doi: 10.1109/25.350282.

[20] S. Wang, J. Xu, N. Zhang, and Y. Liu. A survey on service migration in mobile edge computing. *IEEE Access*, 6:23511–23528, 2018.

[21] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, 64(10):4268–4282, 2016.

[22] K. Yang, Y. Zhao, J. Liu, and K. Wang. A multiuser computation offloading game for mobile cloud computing. *IEEE Transactions on Cloud Computing*, 4 (4):437–448, 2016.

[23] C. You, K. Huang, H. Chae, and B.-H. Kim. Energy-efficient resource allocation for mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 16(3):1397–1411, 2016.

[24] M. Zeng, W. Hao, O. A. Dobre, and H. V. Poor. Delay minimization for massive mimo assisted mobile edge computing. *IEEE Transactions on Vehicular Technology*, 69(6):6788–6792, 2020.

[25] J. Zheng, Y. Cai, Y. Wu, and X. Shen. Dynamic computation offloading for mobile cloud computing: A stochastic game-theoretic approach. *IEEE Transactions on Mobile Computing*, 18(4):771–786, 2018.