Master of Science

Deep RL-based Ellipsoidal Path Planning for MEC enabled UAVs to Minimize Data Transmission Latency and Energy Consumption of Mobile Devices

The Graduate School of the University of Ulsan

Department of Electrical, Electronic and Computer Engineering

RABEYA SADIA

Deep RL-based Ellipsoidal Path Planning for MEC enabled UAVs to Minimize Data Transmission Latency and Energy Consumption of Mobile Devices

Supervisor: Prof. Seokhoon Yoon

A Dissertation

Submitted to
the Graduate School of the University of Ulsan
In Partial Fulfillment of the Requirements
for the Degree of

Master of Science

by

Rabeya Sadia

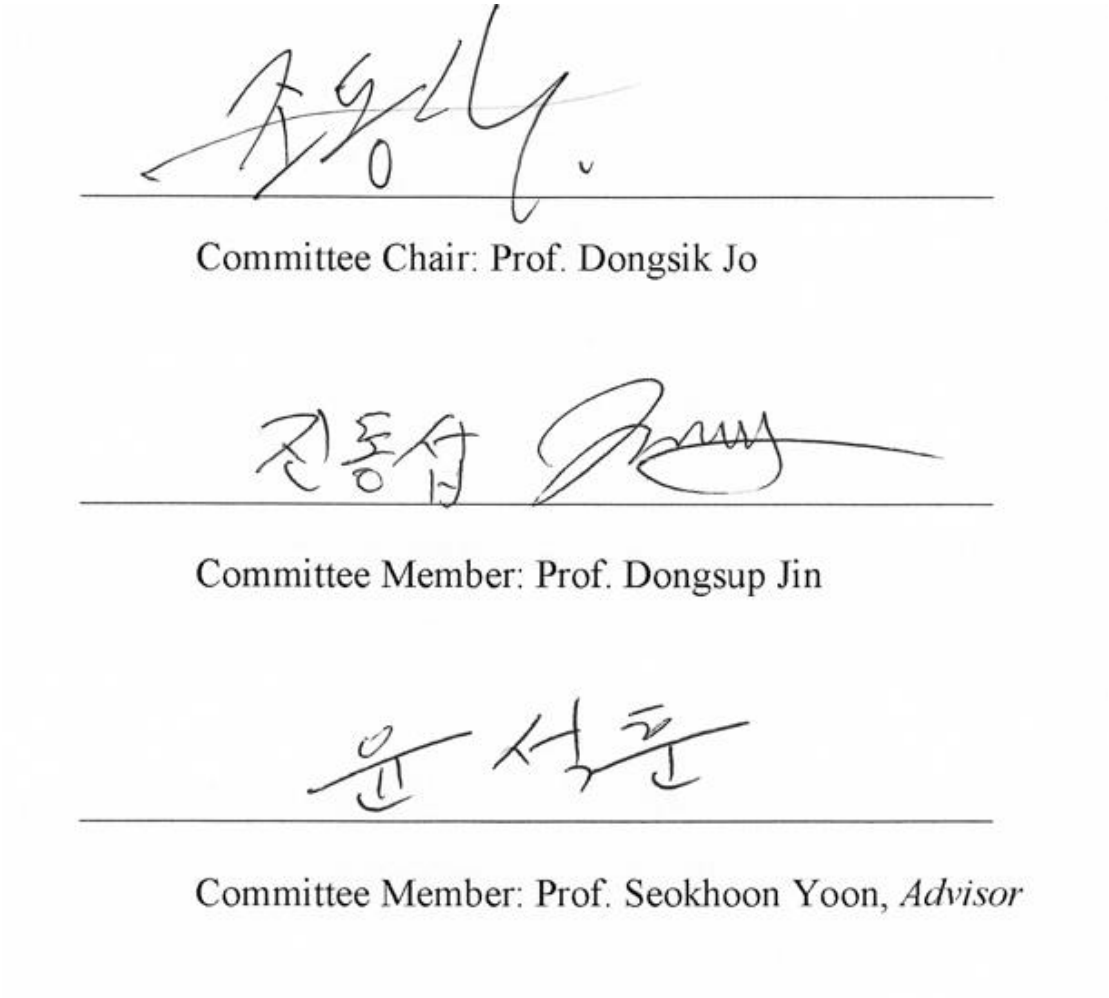
Department of Electrical, Electronic and Computer Engineering

University of Ulsan, Korea

December 2023

Deep RL-based Ellipsoidal Path Planning for MEC enabled UAVs to Minimize Data Transmission Latency and Energy Consumption of Mobile Devices

This certifies that the thesis of Rabeya Sadia is approved by:



Committee Chair: Prof. Dongsik Jo

Committee Member: Prof. Dongsup Jin

Committee Member: Prof. Seokhoon Yoon, *Advisor*

Department of Electrical, Electronic and Computer Engineering

Ulsan, Korea

December 2023

Dedicated to

My beloved husband, parents, parents-in-law, sisters, and brothers, …

Acknowledgments

First, I would like to express my sincere gratitude to my advisor, Prof. Seokhoon Yoon of the department of Computer Engineering, University of Ulsan, for his dedicated guidance and continuous support throughout my study. His critical comments and recommendations not only made my research easier but also enriched my knowledge and sharpened my analytical thinking skills. Without his support and guidance, I would not have been able to make my academic journey successful at the University of Ulsan.

A special thanks goes out to the committee members who reviewed my thesis and provided valuable comments that allowed me to improve it. I wish to convey my appreciation to my fellow labmates at the Advanced Mobile Networks and Intelligence Systems Laboratory for the valuable discussions and great times we've experienced together over the last two years.

Lastly, I want to express my gratitude to my family, who encouraged me to pursue a higher education and supported me during my time in Korea.

Abstract

Due to the flexible deployment and movement capability, unmanned aerial vehicles (UAVs) are being utilized as flying mobile edge computing (MEC) platforms, offering real-time computational resources and low-latency data processing for a wide range of applications. The aim of this article is to explore a multi-UAV-assisted MEC system where multiple UAVs move using ellipsoidal trajectory to provide MEC services to resource-constrained mobile devices in temporary hotspot areas such as festival areas that have delay-sensitive applications with varying task offloading requests. Depending on the position, size, shape, and orientation of the ellipsoidal trajectories, the coverage area, energy consumption of mobile devices, and task transmission latency changes. Moreover, the varying user densities and task offloading request rates make the problem more challenging. Thus, we formulate an optimization problem that finds the center position, major radius, minor radius, and rotation angle of the ellipsoidal trajectory of UAV-assisted MEC servers with the objective of minimizing transmission latency and energy consumption of mobile devices while taking into account the required data transmission rate, task transmission time, energy consumption and several system-related constraints. Then, we have transformed this optimization problem into a Markov decision process and propose a deep Q-learning-based ellipsoidal trajectory optimization (DETO) algorithm to resolve it. The results from our simulations demonstrate that our approach outperforms other baselines, particularly in terms of reducing latency and the energy required for mobile devices to transmit data.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Mobile Edge Computing

Recent technological advancements and the widespread adoption of smart devices have given rise to a plethora of innovative applications, including augmented reality (AR), virtual reality (VR), online gaming, infrastructure monitoring, and automatic navigation, which are complex, computationally intensive, and highly energy demanding [1–4]. Despite being equipped with advanced computation and communication technology, smart devices or mobile devices (MDs) face restrictions in terms of energy resources, computational power, and memory [5,6]. These constraints pose a significant challenge for processing compute-intensive and time-sensitive applications in MDs. Mobile edge computing (MEC) has become increasingly popular as a high-performance server technology that brings computing capabilities and services closer to the edge of the network, often within close reach of MDs [7–10]. By offloading resource-intensive computations to MEC servers, MDs can alleviate the burden of handling computationally intensive tasks. Consequently, this approach not only boosts system performance overall while using less energy, but it also prolongs the MDs' battery life [11, 12].

Even though MEC provides useful services, it's important to take into account the latency that is brought on by the physical distance between MDs and MEC servers [13]. Moreover, Task offloading to the MEC server is challenging during congested events or festivals. During festivals, more people access the network, which results in congestion and a decrease in the network's resource capacity. The increased user activity makes it more challenging to establish a reliable connection between the MDs and the MEC servers. Unmanned aerial vehicles(UAVs) equipped with MEC servers are being investigated as a potential solution to

these problems.

1.2 Deployment of UAV enabled MEC servers

UAV-enabled MEC networks have recently gained significant interest, capturing the attention of both academia and industry. They have demonstrated their value in various applications, including search and rescue operations, environmental monitoring, public safety, military endeavors, infrastructure management, tracking, surveillance, and more. These UAV-enabled MEC networks have emerged as promising solutions, particularly in scenarios marked by high congestion, such as festival events.

Their fast deployment, adaptive mobility, and higher likelihood of establishing Line-of-Sight (LoS) communication pathways all contribute to their suitability for such scenarios. UAV enabled MEC servers become more effective at providing real-time analysis and accelerating response times when placed close to MDs, which makes them important for time-sensitive applications [14–18].

Moreover, to provide better services, UAV enabled MEC servers can dynamically adjust their positions and coverage based on user density and geographical location. Additionally, they establish direct linkages with MDs, ensuring stable connections throughout their operations. Thus, the deployment of UAVs is expected to substantially enhance the performance of MEC servers.

1.3 Motivation

A lot of research is done on UAV location optimization [19–33] focusing on two types of deployment scenarios: static deployment and mobile deployment in 2D or 3D environments. UAVs serve fixed or predefined areas for specific purposes using static deployment. Mobile UAV deployment involves UAVs with the ability to change locations and relocate as necessary.

However, none of the existing research has examined the advantages of utilizing ellipsoidal movement by multiple UAVs. UAVs with ellipsoidal movement are able to dynamically adjust their position, trajectory, and shape within a predetermined area. This adaptability is essential in scenarios with varying user densities. By adjusting their positions in real-time, UAVs can successfully increase coverage to provide MEC services to the associated MDs. Consequently, the number of MDs served by UAVs also increases. When a UAV

moves in an elliptical pattern, the distance between associated MDs and UAVs is reduced, which enhances the data transmission rate. Therefore, it takes less time to transfer a task to the UAV and makes execution faster. In contrast to the common concept noticed in many task-offloading studies [32–34], which frequently simplify user scenarios by allocating a single task per user, our research addresses the task-offloading request rate by considering scenarios where MDs may have multiple tasks or no tasks. When tasks are offloaded by MDs, task transmission consumes the energy of MDs [35–38]. This energy consumption is even further influenced by the position of both UAVs and MDs. Because of increased energy consumption, the battery life of MDs runs out more quickly, which may affect the network's total longevity. Thus, we need to deploy UAVs in a way that minimizes the transmission energy consumption between MDs and UAV-MEC servers. However, focusing exclusively on energy efficiency can slow down data transmission, which might hinder rapid communication and can increase total task transmission latency. Conversely, focusing only on minimizing latency leads to high energy consumption and draining device batteries. Therefore, in this paper, we aim to optimize both the transmission latency and energy consumption of MDs. Moreover, a single UAV's constrained computing and energy capacities can actually limit its performance in task-offloading scenarios. Employing multiple UAVs may be more advantageous and suitable when the number of users is high, or the coverage area is extensive. However, it can be difficult to choose a globally optimal strategy in multi-UAV systems without exact and comprehensive environmental knowledge.

The advancements in machine learning [39], especially the integration of deep reinforcement learning (DRL) with deep neural networks (DNNs) [40] and reinforcement learning (RL) [41], have gained attention in solving the complex optimization problem for UAV-assisted MEC systems. Moreover, the complex states of a UAV-assisted MEC system are captured through the DNNs in DRL, and in each step, the expected reward of the next state is combined with the immediate reward of the current state, which directs the agent's decision-making towards potential actions in each step [42]. Thus, DRL can effectively resolve the problem under consideration by iteratively improving its learning.

Therefore, this article proposes a DRL model named deep Q-network (DQN) based ellipsoidal trajectory optimization (DETO) algorithm for UAVs in a UAV-assisted MEC system. The major contributions of this article are summarized as follows:

- A UAV-MEC system is investigated, where multiple UAVs are used as flying MEC servers to provide computing services to the associated MDs. Each MD has computation-

intensive tasks to offload to the UAV-MEC server. The primary objective is to reduce the weighted sum of total data transmission latency and energy consumption of MDs by optimizing the position, size, and shape of ellipsoidal trajectories for the UAV-MEC servers.

- A Markov decision process (MDP) is formulated for this optimization problem. Then, a DQN-based ellipsoidal trajectory optimization (DETO) is proposed to optimize the center position, major radius, minor radius, and rotation angle of the ellipsoidal trajectories of the UAV-MEC servers. Additionally, DETO also optimizes the associations between MDs and UAV-MEC servers.

- The Geolife dataset [43] providing user position data is used for simulation. Extensive simulations are done to assess the effectiveness of the algorithm. According to the simulation findings, the proposed model performs better than other baseline approaches, including the greedy algorithm and genetic algorithm (GA), in terms of different numbers of MDs, different numbers of task request rates, and different numbers of UAVs.

1.4 Dissertation Organization

The subsequent sections of the thesis are organized as follows: The literature that is related to the location optimization of UAV is presented in Chapter 2. Chapter 3 provides background information on the topics used in the problem formulation and algorithms. Chapter 4 consists of system model and problem formulation. The proposed Algorithm is described in detail in chapter 5. Chapter 6 describes the simulation results. The thesis is concluded in Chapter 7.

Chapter 2

Related Works

There has been a lot of recent research on the difficulties that come with deploying UAVs while taking into account a number of other factors. For instance, optimizing response time and bandwidth efficiency [19], maximizing ground radio coverage [20], enhancing coverage performance efficiency while minimizing transmission power [24], maximizing the system throughput [26], improving spectral efficiency [30], optimizing the quality of experience (QoE) [31]. Next, we provide a brief introduction to these aspects.

2.1 Optimal placement of static UAVs

Earlier studies have focused on the deployment of fixed-position UAVs in a number of scenarios, including both single-UAV and multi-UAV configurations. In [19], authors introduced an evolutionary algorithm for disaster areas, optimizing the terrestrial and UAV base station positions to reduce their count, enhance response times, and ensure sufficient bandwidth. In [20], an analytical method is introduced to adjust the height of the UAV for enhanced wireless communication in remote or disaster-affected areas by maximizing ground radio coverage. The letter [21] suggests an effective 3D placement technique for UAV-BSs that optimizes the connected ground user's number while minimizing the UAV's transmission power. In [22], an optimal UAV deployment strategy is suggested for high-speed wireless networks that increase the number of served high priority GNs. The main objective of these studies is to address the problem of developing a single UAV in order to increase its coverage. However, as the need for wireless connectivity grows, it becomes more and more important to investigate the deployment of multiple UAVs while trying to achieve even wider coverage and improve the efficiency of wireless networks. In [23], the challenge of establishing

wireless connectivity in infrastructure-limited terrestrial networks is addressed using mobile base stations (MBSs) mounted on UAVs, aiming to ensure communication coverage for distributed ground terminals (GTs) by minimizing MBS count through a polynomial-time spiral arrangement algorithm. In [24], the efficient placement of UAVs as wireless base stations is investigated to improve ground user coverage where downlink coverage probability is used as a function of antenna gain and UAV height. A novel framework utilizing circular packing theory is suggested for optimizing the UAV's 3D locations, aiming to enhance coverage performance while reducing transmit power. The paper [25] focuses on enhancing 5G and beyond 5G network performance by strategically placing multiple UAV-BSs in 3D space to optimize user coverage while accounting for various QoS requirements through heuristic algorithms. For high-rate wireless communication systems, the paper in [26] suggests a 3-D multi-UAV deployment strategy in an iterative way. The goal is to maximize system throughput while considering co-channel interference and QoS requirements. The above-mentioned studies fail to consider the multiple UAVs' high level of mobility and flexible movement design.

2.2 Optimal placement of mobile UAVs

The paper [27] proposes a novel cyclical multiple-access method using a base station-equipped mobile UAV to enhance wireless connectivity and communication quality for distributed GTs through scheduled interactions. In the paper [28], the author presents a mobile relaying technique using high-speed UAVs to enhance communication between a source and destination by optimizing the trajectory and power allocation of the UAV iteratively. However, the above-mentioned studies considered a single UAV and designed a simple trajectory for it. The study in [29] focuses on energy-efficient UAV-to-GT communication, proposing a circular trajectory with the center position at GT, optimizing the UAV's trajectory to find a balance between communication throughput and energy consumption. However, this paper focuses on optimizing the speed and flight radius of a single UAV to achieve an optimal circular trajectory.

2.3 Optimal placement of UAVs using RL techniques

In recent decades, the application of machine learning, especially reinforcement learning, has gained significant attention for solving UAV deployment problems by enabling adap-

tive and optimized operations in dynamic environments. The paper [30] uses BS-equipped UAVs to enhance cellular network capacity during high-traffic scenarios in 5G networks and optimize the 3D UAV-BS locations based on user requirements to maximize spectral efficiency. In [31], a novel framework addressing UAV deployment and movement for QoE optimization is introduced, incorporating genetic algorithm-based cell division, Q-learning-based deployment, and movement strategies to enhance cumulative user satisfaction. These papers concentrate on improving the positions of UAVs in a multi-UAV scenario, with a focus on communication-related issues. Additionally, the UAV-assisted MEC system has been explored by investigating the flexible movement of UAVs in [32], [33]. Using a cooperative multi-agent DRL framework, paper [32] investigates collaborative task offloading in a multi-UAV multi-EC MEC system, optimizing trajectories, communication, and computation to reduce delays and energy consumption for execution. The paper [33] introduces a flying MEC (F-MEC) architecture that offloads tasks to UAVs, aiming to achieve energy efficiency with DRL and convex optimization-based trajectory control techniques.

2.4 Minimizing Latency and Energy Consumption for AR/VR applications

Recently, there has been a growing interest in using MEC to enhance and support AR/VR applications. The article [44] addresses the challenges of ultralow delay and high energy consumption in IoT-based AR applications, proposing an intelligent task offloading and resource allocation algorithm using a multiagent deep deterministic policy gradient (MADDPG) framework in dynamic MEC environments. In a multi-user wireless network intended for AR applications, the paper [45] focuses on applying MEC in 5G for Ultra Reliable Low Latency Communication (URLLC) applications. It addresses the joint optimization of resource allocation and task offloading to minimize energy consumption for user terminals, utilizing a deep reinforcement learning algorithm known as multi-agent deep deterministic policy gradient (MADDPG). A MEC model is introduced in [46] for VR applications in 5G, addressing limitations in device resources. It proposes an efficient multi-player, multi-task computation offloading model, formulated as an integer optimization problem to minimize network latency and energy consumption. A wireless VR-enabled medical treatment (WVMT) system is proposed in [47], merging VR with the Internet of Medical Things (IoMT) and emphasizing multiaccess edge computing (MEC). To address computation efficiency and security, it introduces a blockchain-enabled task offloading scheme for VR viewport rendering tasks to edge access points (EAPs). A collective reinforcement learning (CRL) algorithm dynamically

allocates resources, enhancing VR QoE. The paper [48] explores MEC as an efficient solution for delivering VR videos over wireless networks. It introduces a communications-constrained MEC framework that optimizes communication resource consumption by utilizing computation and caching resources on the mobile VR device. The proposed task scheduling strategy aims to minimize communication resource consumption under a delay constraint, considering tradeoffs among communication, computing, and caching.

In the above-mentioned location optimization studies, the primary objective is to optimize the horizontal location and altitude of UAVs, while the concept of ellipsoidal movement for UAVs remains unexplored. To address this gap, our research explores the potential of UAVs to dynamically adjust their trajectories in ellipsoidal paths, thereby enhancing the connection between MDs and UAVs. Each MD may have multiple tasks rather than just one because of the variance in task offloading request rates. Furthermore, we have developed a multi-objective framework aimed at minimizing the weighted sum of total transmission latency and energy consumption that differs from a single-objective perspective. We use user positions taken from the Geolife trajectory dataset rather than random user placements. This method offers precise location data, which enhances the optimization of ellipsoidal trajectories for UAVs, bringing them closer to real-world scenarios.

Chapter 3

Background

3.1 Optimization

Optimization techniques are encountered in wide range of applications, including engineering [49] and manufacturing [50], operations research and logistics [51], and finance and economics [52]. The aim of optimization is to identify the optimal solution for a problem, taking into account a predefined set of constraints. In the context of computer science, this often entails finding the solution that minimizes resource usage, such as time or memory.

The main characteristics of optimization problems:

- Objective function: A mathematical formula is used to determine the performance, value or cost related to a specific set of variables. The optimization problem requires either maximization or minimization of this function.
- Decision Variables: They are the input variables that need to be controlled to have an impact on the result. The optimal solution can be found via the optimization algorithm by determining the values of these variables.
- Constraints: Constraints denote conditions that restrict the acceptable values of decision variables and must be met by the solution. There are two types of constraints: equality constraints and inequality constraints.
- Feasible Region: The set of all decision variable values that are feasible and satisfy the constraints is known as the feasible area. It represents the area in which the best solution will be found by the optimization method.

- Optimal Solution: The set of decision variable values that either maximizes or minimizes the objective function while satisfying all constraints constitutes the optimal solution. It is possible to have multiple acceptable solutions in some cases.

The general form of an optimization problem can be expressed as:

$$\text{Maximize} \quad f(\mathbf{x}) \tag{3.1}$$

$$\text{Subject to} \quad g_i(\mathbf{x}) \leq 0, \quad i = 1, 2, \ldots, m \tag{3.2}$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, 2, \ldots, p \tag{3.3}$$

$$x_i \text{ variables}, \quad i = 1, 2, \ldots, n \tag{3.4}$$

Here, equation (3.1) represents the objective function. Equation (3.2) represents a set of inequality constraints. Each inequality constraint with the form $g_i(x) \leq 0$ specifies a requirement that the solution x must meet. Equation (3.3) represents a set of equality constraints. The optimization problem's decision variables are represented by equation (3.4). To optimize the objective function, we need to determine these values.

3.1.1 Linear Optimization

The soviet mathematician Leonid Kantorovich and the american economist Wassily Leontief made the first significant attempts to apply the linear programming method in the fields of manufacturing schedules and economics, respectively, in the late 1930s. However, their work was ignored for many years. Linear programming was widely utilized during World War II to manage transportation, scheduling, and resource allocation subject to limitations like cost and availability. With the advent of the simplex approach by american mathematician George Dantzig in 1947, which significantly simplified the solution of linear programming problems and contributed significantly to the method's acceptance.
It's a strategy for solving problems when the objective function and the constraints are linear functions of the decision variables. Equalities or inequalities may be used in the constraint equations. Decision variables, an objective function, constraints, and non-negative restrictions are all components of a linear optimization problem.

$$\text{Maximize} \quad 2x + 3y \tag{3.5}$$

$$\text{Subject to} \quad x + y \leq 5 \tag{3.6}$$

$$2x + 3y \leq 12 \tag{3.7}$$

$$x, y \geq 0 \tag{3.8}$$

Here, equation (3.5) is the objective of the optimization problem. Equation (3.6) and (3.7) represent a set of inequality constraints. These constraints specify the requirements that the solution (x, y) must meet. Equation (3.8) indicate that the decision variables x and y must be non-negative.

3.1.2 Integer Linear Optimization

Integer linear programming is a branch of mathematical optimization that is also known as integer linear optimization, or ILP for short. The objective function and constraints maintain linearity in ILP, but one distinguishing feature is that only integer values are allowed for decision variables. ILP essentially seeks to identify an optimal solution inside a specified problem domain, where some variables are constrained to integer values, often non-negative integers.

$$\text{Maximize} \quad \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{3.9}$$

$$\text{Subject to} \quad A\mathbf{x} \leq \mathbf{b} \tag{3.10}$$

$$\mathbf{x} \geq \mathbf{0} \tag{3.11}$$

$$\mathbf{x} \in \mathbb{Z}^n \tag{3.12}$$

Here, equation (3.9) is the objective function that need to be maximized. Equation (3.10) represents a set of inequality constraints that need to be satisfied by solution x. Equation (3.11) indicate that the decision variables x must be non-negative. Equation (3.12) indicate that the decision variables x can only have integer values.

3.1.3 Mixed-Integer Nonlinear Optimization

Making discrete decisions while taking into account nonlinear system dynamics that affect the final design or plan's quality is required in a variety of optimal decision-making scenarios in scientific, engineering, and public sector contexts. Mixed-integer nonlinear programming (MINLP) problems involve the difficult task of maximizing discrete variable sets while taking care of the complexities of controlling nonlinear functions. The nonlinear programming (NLP) and mixed-integer linear programming (MILP) are both subsets of MINLP, which is one of the most comprehensive modeling approaches in optimization.

$$\text{Minimize} \quad f(x) \tag{3.13}$$

$$\text{Subject to} \quad c(x) \leq 0 \tag{3.14}$$

$$x \in X \tag{3.15}$$

$$x_i \in \mathbb{Z}, \quad \forall i \in I \tag{3.16}$$

Here, equation (3.13) is the objective function that need to be minimized. Equation (3.14) represents a set of inequality constraints that need to be satisfied by solution x. Equation (3.15) states that the decision variables x must be a part of the set X. Equation (3.16) imply that either all or part of the decision-making variables, x_i, must have integer values.

3.2 Optimization Algorithms

In the real world, we frequently have to deal with limitation resources, time constraints, and limited financial resources. Therefore, using optimization techniques is always necessary to ensure the greatest possible allocation of these priceless assets. Additionally, selecting appropriate optimization methods requires careful consideration in order to solve particular problems effectively. But for engineers, coming up with optimization strategies can be a difficult task because of the inherent complexity of optimization difficulties.

Numerous approaches that come within the stochastic or deterministic categories have been used to handle optimization problems [53]. Deterministic algorithms consistently provide the same output with no sign of randomness when given the same input. The downhill and hill climbing algorithms are two instances of deterministic algorithms. While using the

same input, stochastic algorithms, on the other hand, incorporate a random component and often produce different results each time they are used.

The two subcategories of stochastic algorithms are heuristics and meta-heuristics. To discover solutions, heuristics are specialized approaches that depend on a process of trial and error. An example of a heuristic algorithm is the nearest neighbor method. Contrarily, meta-heuristics, which are an improvement above heuristics [54], integrate randomization with local search methods. The genetic algorithm is a noteworthy example of a meta-heuristic that was inspired by a natural process.

Next we will discuss the genetic algorithm which is used in this work to find the center position, size and shape of the ellipsoidal trajectory of UAV-MEC server to compare the result with DETO.

3.2.1 Genetic Algorithm

The Genetic Algorithm (GA) has been used to solve a variety of practical problems in the fields of design, optimization, and problem-solving. J. Holland originally proposed this population-based method, which is based on the ideas guiding the evolution of biological systems. A GA's framework includes the following steps in the process:

- Selection: Parents are selected from the population depending on their fitness, which is strongly related to the particular optimization problem.
- Crossover: By transferring genetic information between the chosen parent solutions, new offspring are created.
- Mutation: Newly generated offspring go through mutations, where randomly chosen genes are changed, which encourages diversity and exploration.

Each individual or solution is represented as a chromosome in a GA, frequently encoded as a string in binary or decimal form. The iterative process comprises choosing parents, producing offspring through crossover and mutation, and keeping the fittest offspring depending on their fitness. The selection process for the next generation is governed by the "survival of the fittest" principle.

3.3 Reinforcement Learning

A computational method for comprehending and automating goal-directed learning and decision-making is called reinforcement learning (RL). By putting a high focus on an agent's capacity to learn directly from its interactions with the environment, without the necessity for model-perfect supervision, it sets itself apart from other computational techniques. RL, in our opinion, is the first area to genuinely address the computing difficulties that come from learning from interactions with the environment in order to accomplish long-term objectives.

The interaction between a learning agent and its environment is defined by a formal framework in reinforcement learning in terms of states, actions, and rewards. This framework aims to offer a straightforward method of expressing key aspects of the artificial intelligence challenge. A sense of cause and effect, a sense of ambiguity and nondeterminism, and the presence of clear goals are some of these characteristics.

Most RL techniques rely on the ideas of value and value functions as their main components. In order to efficiently search the space of policies, the value functions are essential. RL techniques use value functions to set them apart from evolutionary techniques that search in policy space directly and are led by scalar assessments of entire policies.

The mathematical framework called the Markov Decision Process (MDP) is used to characterize an environment in RL. The four tuples that make up MDP are (S, A, R, P). Where, S denotes the state set in the environment. At each time step t, state S_t is observed by the agent, where $S_t \in S$. A denotes the set of possible actions for the agent. The agent determines what action to take at each time step t based on the received state S_t, where $A_t \in A(S_t)$. $A(S_t)$ denotes a set of potential actions in the state S_t. R denotes the expected reward that will be obtained as a result of taking action A_t to move from state S_t to state S_{t+1}. P represents the transition probability that at time step t while taking action $A_t = a$ staying at state $S_t = s$ will result to state $S_{t+1} = s'$ in the next time step $t + 1$ is given by:

$$P(s' \mid s, a) = \Pr(S_{t+1} = s' \mid S_t = s, A_t = a) \tag{3.17}$$

3.3.1 Q Learning

One of the traditional RL is the Q-learning algorithm [55] that uses model-free RL to determine the value of a given action in a given state. It can handle issues with stochastic

transitions and rewards without the need for modifications because it does not require a model of the environment. Chris Watkins first mentioned Q-Learning in his Ph.D. thesis at Cambridge University (1989). The method was created as a solution to the problem of RL, where a decision-making agent learns by interaction with its environment. Watkins' research builds on earlier work in control theory and dynamic programming.

Even though initially, Q-learning was used in the fields of process control, chemical process, industrial process automatic control, and in the field of airplane control [56–58]. Currently, Q-learning is mostly utilized in network management to improve routing and reception processing in network communication [59]. The development of AlphaGo has sparked intense game theory research [60]. Trial and error reinforcement learning shares several traits with the characteristics of the human learning process. As a result, Q-learning is excelling in the robotics industry. Particularly in drones, humanoid robotics, and autonomous vehicles [61].

The classic expression for optimal Q-value function is :

$$Q^*(s,a) = \mathbb{E}[r + \gamma \cdot \max_{a'} Q^*(s',a')] \tag{3.18}$$

This means that the total return from state and action is equal to the sum of the immediate reward and the discounted reward obtained by employing optimal strategy. Here, γ denotes the discount factor which is used to account for the decreased value of future reward in comparison to immediate reward. This is because future rewards are frequently considered to be less significant than immediate rewards, hence they need to be discounted. However, Q-learning mainly involves applying the Bellman optimality equation to iteratively update the Q-values.

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a')\right] \tag{3.19}$$

It is seen that Q_i converges to $Q*$ when $i \to \infty$. Q-learning uses greedy policy $a = max_a Q(s,a;\theta)$ for selecting actions based on the most recent Q-value estimations.

3.3.2 Deep Q Learning

Q-learning is a simple and powerful algorithm that maintains a table to store each combination of state and action and has limited capacity for storing information. Thus, it is

impracticable to describe the action-value function (Q-function) as a table with values for each combination of state and action in many real-world scenarios. This is due to the fact that the state and action spaces can be huge or even continuous, making it impossible to store and compute values for all potential state-action pairs. Furthermore, this approach cannot be applied to unseen states since the agent must visit each state and carry out each action in order to update each state-action pair. Although under some circumstances Q-Learning is guaranteed to converge to an ideal policy, in reality convergence may be delayed, especially in complex contexts. Deep-Mind [62] introduces DQN to overcome this issue by utilizing function approximation through neural network (NN) to estimate the Q-values for individual state-action combinations. The state is provided as the input, and the Q-value for every action that could be taken is created as the output. The difference of Q-learning and deep Q-learning presented in Figure 3-1

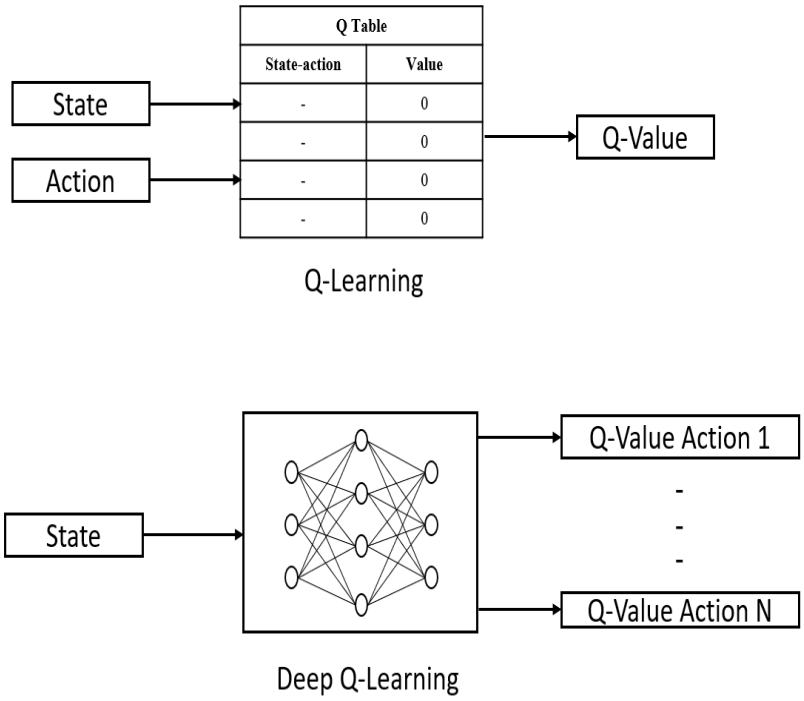


Figure 3-1: The difference of Q-learning and deep Q-learning

Chapter 4

System Model and Problem Formulation

4.1 System Model

We consider a square region consisting of a set of MDs, represented by $\mathbb{D} = \{1, 2, 3, \ldots, M\}$, and a set of UAVs, denoted by $\mathbb{U} = \{1, 2, 3, \ldots, N\}$. Other notations used in the system model are in Table 4.1.

We assume the current location of MDs is known, and each MD has computation-intensive tasks to execute. The MDs themselves, however, are unable to execute these computation-intensive tasks locally because of limited resources in processing power, memory, or energy. To address this, UAVs are utilized to enhance the delivery of MEC services to ground MDs, allowing them to use more powerful computing resources for efficient task execution. Each MD i has task request rate λ_{i} ($\lambda_{\text{min}} \leq \lambda_i \leq \lambda_{\text{max}}$), and the size of the input data for each task is denoted l_{i} ($l_{\text{min}} \leq l_i \leq l_{\text{max}}$), where $\lambda_{\text{min}}, \lambda_{\text{max}}$ are the minimum and maximum task request values, and $l_{\text{min}}, l_{\text{max}}$ are the minimum and maximum task data sizes. MDs transfer all their computational task requests to UAV-MEC servers for execution. The UAV-MEC servers hover over a target area in an elliptical pattern while providing services to the connected MDs. In particular, to prevent trajectories from colliding, a careful deployment plan is essential. We choose each UAV-MEC server's initial center position for the ellipsoidal trajectory from the cluster center. Thus, a specific sub-area within the total coverage region is allocated to each UAV, which helps in providing MEC services to the ground MDs located in that specific sub-area in an efficient manner.

Table 4.1: List Of Notations

Notation	Description
$M, \mathbb{D}$	Number and set of MDs
$N, \mathbb{U}$	Number and set of UAVs
$A_{i,j}$	User association between the ith MD and the jth UAV-MEC server
λ_i	Task request rate of MD i
l_i	Input data size (in byte) of each task of MD i
$\lambda_{min}, \lambda_{max}$	Minimal, maximal values of task request
l_{min}, l_{max}	Minimal, maximal values for the task data size
$[x_i, y_i, 0]$	Coordinates of the ith MD
$[x_j(t), y_j(t), h]$	Coordinates of the jth UAV-MEC server
T	Cycle period of UAV-MEC server's ellipsoidal trajectory
$C_{x,j}$	X-coordinate center of the jth UAV-MEC server's ellipsoidal path
$C_{y,j}$	Y-coordinate center of the jth UAV-MEC server's ellipsoidal path
$R_{x,j}$	Major radius of the jth UAV-MEC server's ellipsoidal path
$R_{y,j}$	Minor radius of the jth UAV-MEC server's ellipsoidal path
θ_j	Rotation angle in the jth UAV-MEC server's ellipsoidal path
$r_{i,j}$	Ground distance
$d_{i,j}(t)$	3D distance between MD i and UAV-MEC server j
PL_{LoS}	Path loss of line-of-sight link
PL_{NLoS}	Path loss of non-line-of-sight link
$P(LoS)$	Probability of line-of-sight link
$P(NLoS)$	Probability of non-line-of-sight link
a, b	Propagation environment constants
c	Speed of light
f	Carrier frequency
η_{LoS}, η_{NLoS}	Additional loss for LoS and NLoS propagation modes
ω	Elevation angle between MD and UAV-MEC server
$PL_{i,j}(t)$	Path loss between MD i and UAV-MEC server j
h	Height of the UAV-MEC server
P_t	Transmit power of the MD
σ^2	Noise power
B	Channel bandwidth
$SNR_{i,j}(t)$	Signal-to-noise ratio between MD i and UAV-MEC server j
$R_{i,j}$	Average data transmission rate
$T_{i,j}^{Trans}$	Data transmission latency between MD i and UAV-MEC server j
$E_{i,j}^{Trans}$	Energy consumption for transferring data from MD i to UAV-MEC server j
T_j^{total}	Total data transmission latency of MDs connected to UAV-MEC server j
E_j^{total}	Total transmission energy consumption of MDs connected to UAV-MEC server j for offloading task request per-second
U_j	Weighted sum of transmission latency and transmission energy consumption
ρ	Relative weights of transmission latency and transmission energy consumption of MDs connected to UAV-MEC server j

Additionally, each MD can establish a connection with one UAV-MEC server at a time to offload its task request. Then, one can have

$$A_{ij} \in \{0, 1\}, \quad \forall i \in \mathbb{M}, \forall j \in N. \tag{4.1}$$

where, $A_{ij} = 1$ implies the ith MD is associated with the jth UAV; otherwise, $A_{ij} = 0$.

4.1.1 Equations of the UAVs Ellipsoidal Movement

On the horizontal plane, an ellipse's equation, according to [63], can be presented as

$$\frac{(x - C_x)^2}{R_x^2} + \frac{(y - C_y)^2}{R_y^2} = 1. \tag{4.2}$$

where the ellipse's center is located at (C_x, C_y); R_x and R_y, respectively, are the radii of the X and Y axes. If the ellipse rotates at a particular angle, θ, the equation can be written as

$$\begin{aligned} &\frac{((x - C_x)\cos(\theta) + (y - C_y)\sin(\theta))^2}{R_x^2} \\ &+ \frac{((x - C_x)\sin(\theta) - (y - C_y)\cos(\theta))^2}{R_y^2} = 1. \end{aligned} \tag{4.3}$$

Equation (4.3) can be written as a parametric equation of a rotated ellipse as follows

$$\begin{aligned} x(\delta) &= C_x + R_x\cos(\delta)\cos(\theta) - R_y\sin(\delta)\sin(\theta), \\ y(\delta) &= C_y + R_x\cos(\delta)\sin(\theta) + R_y\sin(\delta)\cos(\theta). \end{aligned} \tag{4.4}$$

This parametric equation is used to calculate the UAV's X and Y location points on the rotated ellipsoidal path using parameter δ. Specifically, for any given time t, $\delta(t) = \frac{2\pi}{T}(t)$ calculates the UAV's position along the ellipsoidal path [64], where 2π and T represent the complete circumference of the ellipse and the total time for UAV to complete one cycle over the ellipse, respectively.

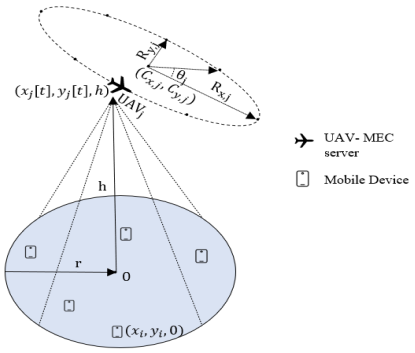


Figure 4-1: The xy-plane position of MDs and a UAV with an ellipsoidal trajectory.

In our considered scenario, we assume the jth UAV-MEC server following an ellipsoidal trajectory on the horizontal plane with major radius $R_{x,j}$ and minor radius $R_{y,j}$, as well as

rotation angle θ_j (Figure 4-1). The center position of this ellipsoidal trajectory is represented by $(C_{x,j}, C_{y,j})$. The UAV-MEC server maintains a constant altitude h while flying, similar to [23,65]. Within cycle period T, the UAV-MEC server completes one cycle of this ellipsoidal trajectory in order to assist the associated MDs. We partition cycle period T into P equally-sized time intervals. Thus, in each time slot t, the coordinates of the jth UAV-MEC server are denoted $[x_j(t), y_j(t), h]$, where $x_j(t), y_j(t)$, and h represent the X, Y, and Z coordinates, respectively, of UAV-MEC server j in time slot t along the ellipsoidal path. By using equation (4.4), the X, Y location of the jth UAV-MEC server in time slot t can be written as

$$\begin{aligned} x_j(t) &= C_{x,j} + R_{x,j} \cos\left(\frac{2\pi}{T}t\right)\cos(\theta) \\ &\quad - R_{y,j} \sin\left(\frac{2\pi}{T}t\right)\sin(\theta), \\ y_j(t) &= C_{y,j} + R_{x,j} \cos\left(\frac{2\pi}{T}t\right)\sin(\theta) \\ &\quad + R_{y,j} \sin\left(\frac{2\pi}{T}t\right)\cos(\theta). \end{aligned} \tag{4.5}$$

4.1.2 Air-to-ground Path Loss Model

Now, if the ith MD wants to offload its task to the jth UAV-MEC server, where the position of MD i is denoted $(x_i, y_i, 0)$ in the 3D coordinate system, then the ground distance between MD i and UAV-MEC server j in time slot t is denoted $r_{i,j}(t)$. While the UAV-MEC server moves along the ellipsoidal path, the 3D distance between MD i and UAV-MEC server j in time slot t is expressed as

$$d_{i,j}(t) = \sqrt{(x_i - x_j(t))^2 + (y_i - y_j(t))^2 + h^2}. \tag{4.6}$$

The wireless communications model between the UAV-MEC server and the ground MDs is constructed based on line-of-sight (LoS) and non-line-of-sight (NLoS) link propagation modes from which the air-to-ground path loss model is derived. From [66] and [67], the mathematical equations for the path loss of the LoS and NLoS propagation modes can be expressed as follows

$$\begin{aligned} PL_{\text{LoS}} &= 20\log d_{i,j}(t) + 20\log f + 20\log\frac{4\pi}{c} + \eta_{\text{LoS}}, \\ PL_{\text{NLoS}} &= 20\log d_{i,j}(t) + 20\log f + 20\log\frac{4\pi}{c} + \eta_{\text{NLoS}}, \end{aligned} \tag{4.7}$$

where c represents the light's propagation speed, and f represents the carrier frequency; η_{LoS} and η_{NLoS} are the environment factors specific to average the additional path losses for LoS and NLoS propagation modes, respectively. However, according to the International Telecommunication Union (ITU) guidelines for radio transmissions, the following parameters are essential for figuring out the geometric probability of LoS transmission in an urban setting:

- α represents the portion of the total land area occupied by buildings.
- β quantifies the average density of buildings per unit area, measured in terms of the number of buildings per square kilometer.
- γ is a scale parameter that illustrates the distribution of building heights using the Rayleigh probability density function.

Using these parameters, the LoS probability equation between MD i and UAV-MEC server j can be represented as

$$P(\text{LoS}) = \prod_{n=0}^{m} \left(1 - \exp\left(- \frac{\left(h_{\text{j}} - (n + \frac{1}{2}) \frac{(h_{\text{j}} - h_{\text{i}})}{(m+1)} \right)^2}{2\gamma^2} \right) \right), \tag{4.8}$$

with

$$m = \lfloor (h_{\text{j}} - h_{\text{i}}) tan\theta \sqrt{\alpha\beta - 1} \rfloor, \tag{4.9}$$

where the height of the transmitter (the UAV-MEC server) and the receiver (the MD) are denoted h_j and h_i, respectively. It is important to mention that the system's frequency has no effect on the geometric LoS formula, and this formula can be represented by using the simple modified sigmoid function (S-curve), which significantly simplifies the calculation of LoS probability.

Moreover, the probability function for the LoS link using the S-curve can be represented as

$$P(\text{LoS}) = \frac{1}{1 + a \exp(-b(\omega - a))}, \tag{4.10}$$

where the propagation environment determines constants a and b, which represent various scenarios, including suburban, urban, dense urban, and high-rise urban zones; $\omega = \arctan(\frac{h}{r_{i,j}(t)})$ is the elevation angle between MD i and UAV-MEC server j in time slot t along the ellipsoidal trajectory. The probability function for establishing the NLoS link is

then

$$P(\text{NLoS}) = 1 - P(\text{LoS}). \tag{4.11}$$

Therefore, the probabilistic mean path loss of the system in time slot t can be expressed as

$$PL_{i,j}(t)) = P(\text{LoS}) \times PL_{\text{LoS}} + P(\text{NLoS}) \times PL_{\text{NLoS}}. \tag{4.12}$$

Now, using equations (4.7), (4.10), and (4.11) in (4.12), and by applying basic algebraic operations we have the following [66, 68]:

$$\begin{aligned} PL_{i,j}(t) = & \frac{\eta_{\text{LoS}} - \eta_{\text{NLoS}}}{1 + a \exp\left(-b\left(\arctan\left(\frac{h}{r_{i,j}(t)}\right) - a\right)\right)} \\ & + 10\log(d_{i,j}(t)^2) + 20\log(f) + 20\log\left(\frac{4\pi}{c}\right) + \eta_{\text{NLoS}}, \end{aligned} \tag{4.13}$$

where $r_{i,j}(t) = \sqrt{(x_i - x_j(t))^2 + (y_i - y_j(t))^2}$. In our scenario, we assume the transmit power of each MD is the same. Thus, the signal-to-noise ratio between MD i and UAV-MEC server j in time slot t is

$$SNR_{i,j}(t) = \frac{P_i}{PL_{i,j}(t) \times \sigma^2}, \tag{4.14}$$

where P_i is the transmit power of MD i, and σ^2 is noise power. The ground-to-air (G2A) data transmission rate between MD i and UAV-MEC server j in time slot t is

$$R_{i,j}(t) = B\log_2(1 + \text{SNR}_{i,j}(t)). \tag{4.15}$$

The average data transmission rate for MD i throughout cycle period T is then defined as

$$R_{i,j} = \frac{1}{P}\sum_{t=1}^{P} R_{i,j}(t). \tag{4.16}$$

Assume that all task requests coming from MD i, which is connected to UAV-MEC server j based on maximum throughput, are offloaded through the G2A channel. Thus, the G2A data transmission latency, $T_{i,j}^{Trans}$, between MD i and UAV-MEC server j is determined by the task's data size, l_i, and the rate at which the data are transmitted, $R_{i,j}$. Then, $T_{i,j}^{Trans}$

can be stated as

$$T_{i,j}^{Trans} = \frac{l_i}{R_{i,j}}. \tag{4.17}$$

Therefore, total data transmission latency experienced by the MDs connected to UAV-MEC server j for offloading task requests per second can be obtained as follows

$$T_j^{total} = \sum_{i=1}^{M} A_{i,j} \lambda_i T_{i,j}^{Tran}, \tag{4.18}$$

where λ_i is the task offload request rate of MD i.

The energy consumption for task transmission between MD i and UAV-MEC server j can be obtained as follows

$$E_{i,j}^{Tran} = P_i T_{i,j}^{Tran}. \tag{4.19}$$

The total transmission energy consumption of all MDs connected to UAV-MEC server j for offloading task requests per second is

$$E_j^{total} = \sum_{i=1}^{M} A_{i,j} \lambda_i E_{i,j}^{Tran}. \tag{4.20}$$

4.2 Problem Formulation

The aim of this article is to minimize total transmission latency and total transmission energy consumption of MDs. Thus, the function can be written as

$$U_j = \rho T_j^{\text{total}} + (1-\rho) E_j^{\text{total}}, \tag{4.21}$$

where $\rho \in [0,1]$ is a coefficient that is used to define the relative weight of transmission latency and transmission energy consumption based on the nature of the application. Then, the optimization problem can be expressed as follows

$$\min_{\substack{C_{x,j}, C_{y,j}, \\ R_{x,j}, R_{y,j}, \theta_j}} \quad \sum_{j=1}^{N} U_j, \tag{4.22}$$

$$\text{s.t.} \quad \sum_{j=1}^{N} A_{i,j} = 1, \quad \forall i \in M, \tag{4.23}$$

$$X_{\min} \leq C_{x,j} \leq X_{\max}, \quad \forall j \in N, \tag{4.24}$$

$$Y_{\min} \leq C_{y,j} \leq Y_{\max}, \quad \forall j \in N, \tag{4.25}$$

$$R_{\min} \leq R_{x,j} \leq R_{\max}, \quad \forall j \in N, \tag{4.26}$$

$$R_{\min} \leq R_{y,j} \leq R_{\max}, \quad \forall j \in N, \tag{4.27}$$

$$\theta_{\min} \leq \theta_j \leq \theta_{\max}, \quad \forall j \in N, \tag{4.28}$$

$$R_{i,j} \geq R_{\text{th}}, \tag{4.29}$$

$$T_{i,j}^{\text{tran}} \leq T_{\text{th}}, \tag{4.30}$$

$$E_{i,j}^{\text{tran}} \leq E_{\text{th}}, \tag{4.31}$$

where equation (4.23) indicates one MD can connect to one UAV-MEC server at a time to offload its task. Equations (4.24) and (4.25) indicate the horizontal and vertical center position deployment constraints for the UAV-MEC server's ellipsoidal trajectory. Equations (4.26), (4.27), and (4.28) are the constraints for the major radius, minor radius, and rotation angle, respectively, which define the size and shape of the ellipsoidal trajectory. Constraint (4.29) indicates that the data transmission rate should be higher than the predefined threshold value to ensure the QoS for each MD. Equation (4.30) is the constraint for transmission latency, and (4.31) is the constraint for transmission energy consumption of each MD, which has to be less than the threshold value.

To address the problem formulated above, we propose a DQN-based solution in the next section to obtain nearly optimal results utilizing environmental knowledge.

Chapter 5

Algorithm

In this section, the proposed DQN-based ellipsoidal trajectory optimization (DETO) algorithm is discussed in detail, with a focus on the formulation of the state, action, and reward design aimed at addressing the optimization problem presented in Equation (4.22).

5.1 Markov Decision Process Formulation

Generally, RL problems can be represented as instances of MDP [69], in which the actions taken in the present state determine the future state. Therefore, the process of optimizing the center position, major radius, minor radius, and rotation angle of the ellipsoidal trajectories of UAV-MEC servers to minimize total transmission latency and total transmission energy consumption of MDs can be formulated as an MDP with tuple $< S, A, R >$. At each time step t, the agent observes a state $s \in S$, selects an action $a \in A$, transitions to the future state s', and obtains a reward $r \in R$. The design of state, action, and reward are presented as follows.

State: It consists of positional parameters of each UAV's ellipsoidal trajectory.

$$S = \{C_{x,j}, C_{y,j}, R_{x,j}, R_{y,j}, \theta_j, \forall j \in N\}. \tag{5.1}$$

where $C_{x,j}, C_{y,j}, R_{x,j}, R_{y,j}$, and θ_j denote the x-coordinate center, y-coordinate center, x-coordinate radius or major radius, y-coordinate radius or minor radius and rotation angle of the ellipsoidal trajectory of UAV-MEC server j, receptively.

Action: An action involves one of the following: moving the center position horizontally or vertically in a positive or negative direction, incrementing or decrementing the major or minor radius, changing the rotation angle to either clockwise or anticlockwise, or keeping

the current state unchanged.

Reward: An effective reward design acts as a map, directing the learning agent in the direction of the desired result. It gives the agent important feedback on the effectiveness of its actions. Thus, to effectively solve the center position, size, and shape of the ellipsoidal trajectories of the UAV-MEC servers, we have designed our reward function with the help of our objective function, which is defined as

$$r = \begin{cases} \frac{1}{\eta_1 \Phi}, & \text{if objective decreases} \\ & \text{in current state,} \\ \frac{1}{\eta_2 \Phi}, & \text{otherwise,} \end{cases} \tag{5.2}$$

where η_1, η_2 are constants. The values of η_1, η_2 are both within the range (0,1) and $\eta_2 > \eta_1$, and Φ represents the system objective in Equation(4.22).

5.2 DQN-based Ellipsoidal Trajectory Optimization

Considering the large state space and high dimensional action space in our UAV-assisted MEC system, we employ the DQN framework to address the MDP problem discussed earlier, called DQN-based ellipsoidal trajectory optimization (DETO). The framework is depicted in Figure 5-1 and consists of two NNs (the Q-network and the target Q-network) and a memory dataset to store the experiences.

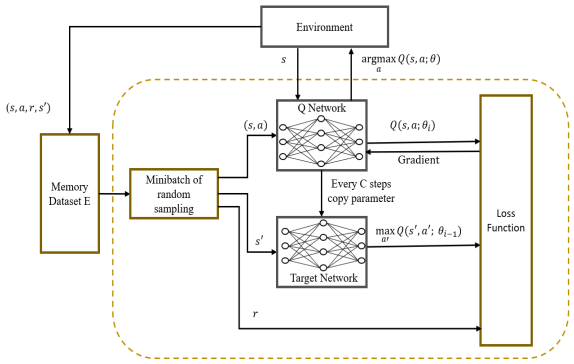


Figure 5-1: The DETO framework.

Each network has multiple layers that are connected to each other to correlate the state and action. The input of the NN is used to represent the current state of the environment and the probability assigned to each possible action is represented by the output of the networks. The memory dataset is depicted as a queue-like data structure.

Algorithm 1 DQN based ellipsoidal trajectory optimization (DETO)

Input: Initial center position, radius, rotation angle of the ellipsoidal trajectory of each UAV-MEC server

Output: Optimal center position, radius, rotation angle of the ellipsoidal trajectory to minimize the system objective

for simulation = 1 to N **do** $\rightarrow N$: Total simulation number
  Initialize memory dataset E with capability B
  Initialize the Q-network's parameter values θ
  Initialize the target Q-network's parameter values $\theta^{-} \leftarrow \theta$
  for episode = 1 to L **do** $\rightarrow L$: Total episode number
    Initial state
    for $t = 1$ to T **do** $\rightarrow T$: Max length of episode
      Current state s_t
      Select an action according to $\epsilon-$greedy algorithm
      Perform action a_t and observe the next state s_{t+1}
      Compute the association of MDs with the UAV-MEC server
      Obtain reward r_t using (5.2)
      Store current experience (s_t, a_t, r_t, s_{t+1}) in E
      Randomly sampling a batch of experiences (s_j, a_j, r_j, s_{j+1}) from E
      Compute target Q-value for the next state using target Q-network
      Apply a gradient descent step with regard to the network parameter values θ to reduce the squared difference between target Q-value and current Q-value using (5.3)
      Periodically, reset the target Q-network's parameter values from the Q-network: $\theta^{-} \leftarrow \theta$
      Set $s_t = s_{t+1}$
    end for
  end for
  Test the model and save the results
end for
choose the best result

The agent creates new transitions by interacting with the environment and stores them in memory. When the memory reaches its limit, it begins to remove the earliest transitions

to make way for the newest ones. Therefore, the most recent transitions are kept in memory while removing the oldest ones, which primarily helps in tracking the agent's most recent interactions with the environment.

The DETO training steps are presented in Algorithm 1. At the beginning of each simulation during the learning process, a number of elements, including the capacity of the memory dataset and the parameters of both the Q-network and the target network, are initialized. For the NN learning process, these parameter values serve as starting points. This initialization is performed through lines 2, 3, and 4 of the algorithm. Note that the DQN uses a dual-network strategy to estimate the current Q-value and target Q-value, where both Q-network and target Q-network have the same neural architecture. However, these networks have a different set of parameter values to make these estimations.

Each training episode begins with the agent starting from an initial state, and each episode runs for a given number of steps (lines 5, 6, and 7). The initial deployment positions for each UAV-MEC server are determined using the K-means clustering technique and are integrated into the initial state information. In each step of the training episode, the NN receives the current state (line 8) as an input value, and it returns the Q-values of every action. The agent then selects an action using an epsilon greedy algorithm (line 9).

Due to the lack of background knowledge about the environment, the agent emphasizes exploration at first. To do this, it occasionally selects random actions under the guidance of a probability parameter, ϵ. With the aid of this exploratory strategy, the agent discovers the dynamics of the environment, gathers important data, and stays away from locally optimal solutions. The agent eventually reduces the use of random actions as it becomes more familiar with the environment. This reduction acts as a switch towards exploitation, forcing the agent to choose actions that work best. This adjustment aids the agent in finding the ideal balance between utilizing what it already determined as best and attempting new things, enabling it to get the best results over time by combining both previously successful strategies and new information.

The agent takes action to change the center position, length of a radius, or rotation angle of the ellipsoidal path to determine the position, size, and shape of the ellipse. By performing the above actions, it observes the next state (line 10). This state transition leads to a change in the UAV-MEC server's position. Consequently, the associations between MDs and UAV-MEC servers also change. The association between each MD and UAV-MEC server is determined based on throughput. In particular, for each MD, the UAV-MEC server that

yields the highest throughput is selected. Subsequently, after computing the association between MDs and the UAV-MEC server, the agent obtains the reward (lines 11, 12).

To make the learning process stable and to enhance sample effectiveness, the agent stores each transition that includes state, action, reward, and the next state in memory dataset E (line 13). When training Q-network parameters, a batch of transitions from the memory dataset is randomly chosen by the agent (line 14). This sampling process is used to break the correlation between consecutive transitions, ensuring the training data are sufficiently diverse, which aids the network's ability to acquire useful weights that can effectively handle a variety of data values.

During training, the aim is to reduce the gap between the target Q-network's output, $r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$, and the Q-network's output, $Q(s, a; \theta_i)$, which is achieved by utilizing the mean squared error (MSE) loss function (lines 15, 16):

$$L(\theta_i) = \mathbb{E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)^2]. \tag{5.3}$$

The parameter values of the Q-network are updated using the Adam optimizer with an initial learning rate of 0.001. Then, the parameter values from the Q-network are copied periodically to the target Q-network (line 17). At the end of the training, we test the model and save the results with optimized parameters for the ellipsoidal trajectory (lines 20-21). Finally, from the outcomes of multiple simulations, we select the best result (lines 22-23).

Chapter 6

Performance Evaluation

The performance of the proposed DETO is evaluated by conducting simulations under different parameters and compared against the performance of GA and greedy algorithms.

In this section, we first present the experimental settings and performance metrics. Then, training efficiency of the DETO scheme and the descriptions of the compared algorithms are presented. Finally, the performance results under different scenarios are explained in detail.

6.1 Experimental settings

We analyze the efficiency of our suggested DETO approach in a UAV-enabled MEC network by conducting simulations. A sub-urban area is considered to simulate the deployment of UAV-MEC servers for 300 MDs distributed within three distinct hotspots. The K-means clustering technique is used to determine the initial center position for each UAV-MEC server based on the user distribution provided and the number of deployed UAV-MEC servers. The initial values of the major radius, minor radius, and the rotation angle for ellipsoidal movement of each UAV are randomly chosen within the ranges [60, 200] m and [0°, 360°], respectively. The value of ρ is set to 0.5, meaning that both transmission latency and transmission energy consumption are equally important. Table 6.1 summarizes the rest of the simulation parameters utilized in our experiments, with default values denoted in bold.

The Geolife mobility dataset [43] is used to determine the MDs' locations. The dataset contains GPS traces of 182 users in Beijing, China, which were collected during a five-year period by Microsoft Research Asia from April 2007 to August 2012. Our simulation covers a crowded region of 1000×1000 m^2 area in the city. The MD locations are then chosen

from the users' GPS points within the specified area by considering the time frame 12 pm to 1 pm during the year 2008.

Table 6.1: Experimental settings

Notation	Description	Value	Unit
Size of the area	-	1000x1000	m^2
Number of MDs	M	$\{150, 200, 250, \mathbf{300}, 350\}$	-
Number of UAVs	N	$\{1, 2, \mathbf{3}, 4\}$	-
Task offload request rate	λ	$[0.1, 1]$	task/s
Task size	l	$[1, 10]$	MB
Height of UAV	h	100 $[14, 70]$	m
Range of major radius	R_x	$[60, 200]$ $[65, 71]$	m
Range of minor radius	R_y	$[60, 200]$	m
Range of rotation angle	θ	[0, 360]	°
Path loss parameter	a	9.61	-
Path loss parameter	b	0.16	-
Path loss parameter	η_{LoS}	1	-
Path loss parameter	η_{NLoS}	20	-
Channel bandwidth	B	20	MHz
Transmit power of MD	P_t	20	dBm
Noise power	σ^2	-100	dBm
The carrier frequency	f	2	GHz
The speed of light	c	299792458	-
weight	ρ	0.5	-

For the DETO framework's neural network structures, we use two hidden layers, each with 64 neurons. These hidden layers have ReLU activation functions, whereas the output layer has a linear activation function. Table 6.2 represents the DETO framework's main hyperparameter representation, which is used to train the model. From five simulation runs, we select the best result, and then we calculate the average of the three best results from a total of 15 simulation runs to provide a more representative and fair evaluation. Four performance metrics are considered for evaluating the results:

- System objective: The System objective is the sum of total transmission latency and total transmission energy consumption of MDs associated with each UAV.

$$\sum_{j=1}^{N} U_j, \tag{6.1}$$

where

$$U_j = \rho T_j^{\text{total}} + (1-\rho) E_j^{\text{total}}, \tag{6.2}$$

$\rho \in [0,1]$ is a coefficient that is used to define the relative weight of transmission latency and transmission energy consumption based on the nature of the application.

- Total transmission Latency: The total transmission latency is the sum of the data transmission latency of each MD. The total transmission latency presented as

$$\sum_{j=1}^{N}\sum_{i=1}^{M} A_{i,j}\lambda_i T_{i,j}^{Tran}, \tag{6.3}$$

where λ_i is the task offload request rate of MD i, $A_{ij} = 1$ implies that ith MD is associated to jth UAV to offload it's task request, $T_{i,j}^{Tran}$ is the data transmission latency between MD i and UAV-MEC server j.

- Total transmission energy consumption: The total transmission energy consumption is the sum of the data transmission energy consumption of each MD. The total transmission energy consumption presented as

$$\sum_{j=1}^{N}\sum_{i=1}^{M} A_{i,j}\lambda_i E_{i,j}^{Tran}, \tag{6.4}$$

where $E_{i,j}^{Tran}$ is the data transmission energy consumption between MD i and UAV-MEC server j.

- Total Throughput: The total throughput is the sum of the average data transmission rate of each MD. The total throughput presented as

$$\sum_{j=1}^{N}\sum_{i=1}^{M} R_{i,j}. \tag{6.5}$$

Table 6.2: Hyperparameters of the DETO Model

Parameter	**Value**
Number of episodes	100
Time steps	80
Learning rate	0.001
Discount factor	0.95
ϵ with starting value	1
ϵ with final value	0.1
Epsilon decay	.9997
Memory size	2500
Batch size	32
Optimizer	Adam optimizer

6.2 Training Efficiency of the DETO Scheme

In this subsection, we examine the effectiveness of the DETO optimization method's training. DETO's training progress is shown in Fig. 6-1. As the number of training episodes increases, we can see that the total number of rewards continues to rise progressively from the beginning. With a total of 100 episodes, total rewards experience a notably significant increase after the 40th episode. This occurs because the agent initially explores new positions at random using the ϵ probability, and these new positions may not yield optimal performance. Moreover, experience replay memory initially has fewer feasible solutions. However, as the neural networks begin training, experience replay memory gathers more useful solutions. Thus, UAVs are able to execute actions iteratively and can learn from mistakes to increase the reward, which ultimately leads to a better system objective. Even though the curve shows fluctuations and lacks convergence, it continually moves upward.

In Fig. 6-2 and Fig. 6-3, we plot the UAVs' ellipsoidal trajectories and the corresponding rewards during the algorithm's iterative operations after the end of the DETO model's training, respectively. With advancement of the learning process, UAVs gradually optimize

their ellipsoidal movements, minimizing transmission latency and energy consumption. The distribution of MDs and the deployment positions for UAV ellipsoidal movements are shown in Fig. 6-2. Fig. 6-3 illustrates the system's improvement over time as the reward increases in each step. Remarkably, within the first 25 steps, the reward stabilizes and stays relatively constant. This implies that after an adequate number of steps, the system achieves a balanced state to reduce the system's overall transmission delay and energy consumption in providing MEC services to the MDs with maximum throughput.

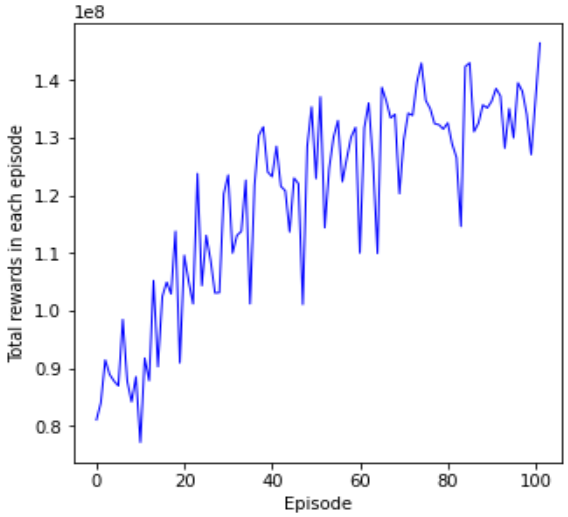


Figure 6-1: Training curve of DETO.

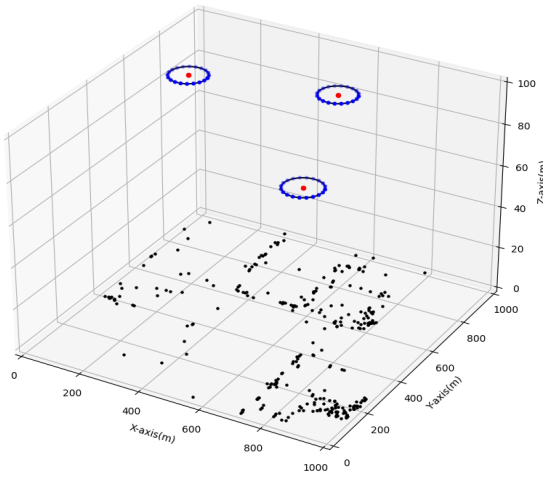


Figure 6-2: Deployment positions of UAVs ellipsoidal movement.

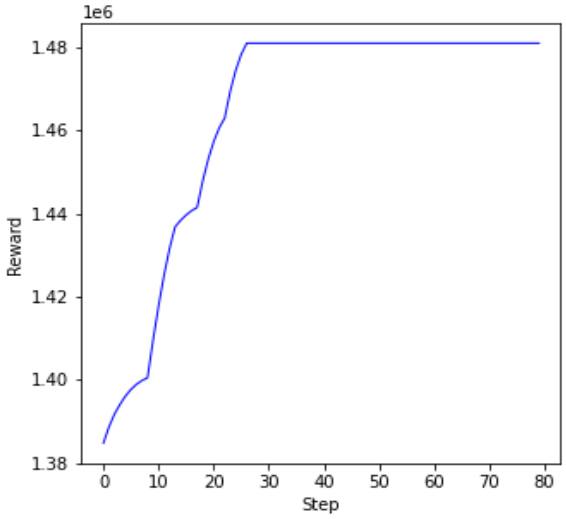


Figure 6-3: Reward during the algorithm's iterative operations.

6.3 Description of the Compared Algorithms

The proposed DETO method is compared with a GA solution and a greedy solution describes below.

- GA-based Ellipsoidal Trajectory Optimization (GAETO): A rank-based genetic algorithm is employed to find a solution for ellipsoidal trajectory optimization, where each solution is a vector representation of real numbers. The initial population is generated randomly, and the fitness function is calculated using the reciprocal value of the system objective for each individual solution. Lower fitness values are given to higher ranks and have a higher chance of being selected as parents. Then, new solutions are generated by applying crossover and mutation, and the best solution is recorded. This procedure repeats for a given number of generations.

- Greedy: In this setup, each UAV's center position is chosen from the cluster center. The major radius of each UAV is equal to one-half the distance along the x-axis between its center and its farthest associated user, while the minor radius is equal to one-half the distance along the y-axis between its center and its farthest associated user. The rotation angle is set to 0°.

6.4 Performance Comparison between DETO and Other Baselines

In this section, the performance of the DETO method is compared with a GA solution and a greedy solution under various conditions, including different numbers of MDs, different numbers of task request rates, and different numbers of UAVs.

6.4.1 Effects of Different Numbers of MDs

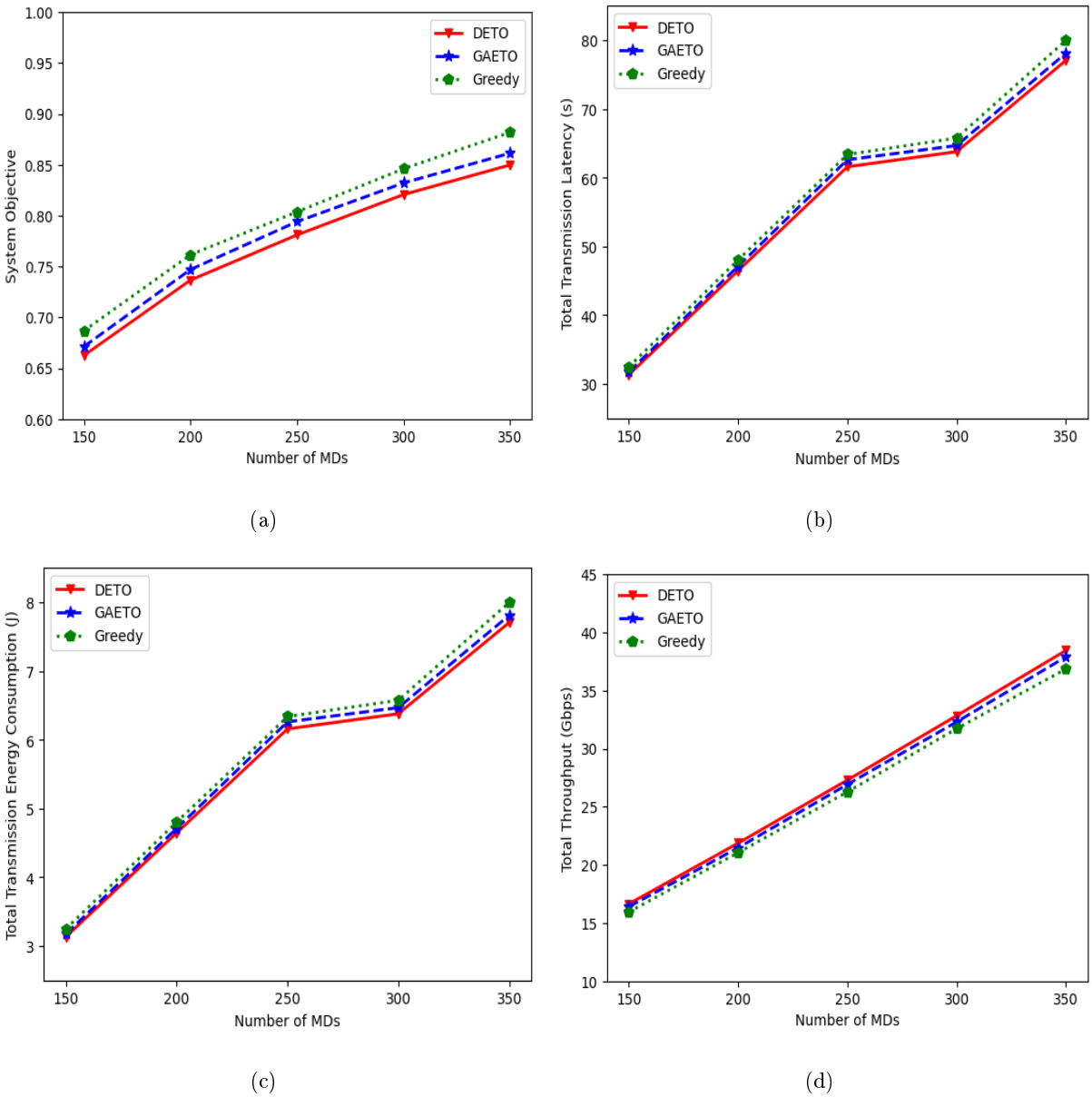


Figure 6-4: Effects of different numbers of MDs: (a) effects on the system objective, (b) effects on the total transmission latency, (c) effects on total transmission energy consumption, and (d) effects on total throughput.

Fig. 6-4 shows the system objective, total transmission latency, total transmission energy consumption, and total throughput of the three methods. With an increase in the number of MDs, all four figures show an increasing trend. This occurs because when the MDs increase in number, the tasks that need to be offloaded also increase, which results in higher system objectives. Additionally, as the number of MDs increases, there is a change in their distribution. This can cause the distance between MDs and UAV-MEC servers to increase, which leads to longer latency and more energy consumption during transmissions, further contributing to the rise in the system objective.

From Fig. 6-4(a), we see that the system objective increases gradually when the MDs increase in number from 150 to 200, from 200 to 250, from 250 to 300, and from 300 to 350, respectively, and DETO performs better than the other two algorithms, achieving a lower system objective. For example, the value of the system objective with DETO is 0.82 when the number of MDs is 300. On the other hand, the objective values from GAETO and the greedy approach are 0.83 and 0.84, respectively. Moreover, the gap between the system objectives increases with the increase in MDs, and the gap is especially wider when MDs number 350 because more tasks need to be offloaded, and GAETO and greedy provide suboptimal solutions.

From Fig. 6-4(b) and Fig. 6-4(c), it can be observed that both total transmission latency and total transmission energy consumption for all three algorithms experience a relatively sharp increase as the number of MDs increases from 150 to 200 and from 200 to 250. This increase occurs due to the increased task requests associated with the growing number of MDs, resulting in longer transmission latency and greater transmission energy consumption. However, the rise is slightly slower when MDs increase from 250 to 300. This is because when the number of MDs is 300, the majority of the users have a task request rate in the 0.1 to 0.5 range. Having lower task requests results in lower transmission latency and transmission energy consumption. It again follows a sharp increase with higher task requests when MDs increase from 300 to 350. DETO's performance is always better than GAETO and the greedy algorithm. For example, when the number of MDs is 300, total transmission latency with DETO is 63.76 s, which is 1.37% and 2.99% lower than GAETO and the greedy algorithms, respectively.

Fig. 6-4(d) illustrates the effect of the different numbers of MDs on total throughput of the system. With an increase in the number of MDs, total throughput experiences a sharp rise. This occurs because the channel bandwidth for each MD is assumed to be fixed.

Therefore, when the number of MDs increases, the system's total throughput also increases. In terms of DETO, total throughput increases by 31.39%, 24.80%, 20.27%, and 17.09% when MDs increase in number from 150 to 200, from 200 to 250, from 250 to 300, and from 300 to 350, respectively. Moreover, the performance of DETO is 4.33%, 3.92%, 3.87%, 3.51%, and 4.33% higher than the greedy approach for all numbers of MDs.

6.4.2 Effects of Different Numbers of Task Request

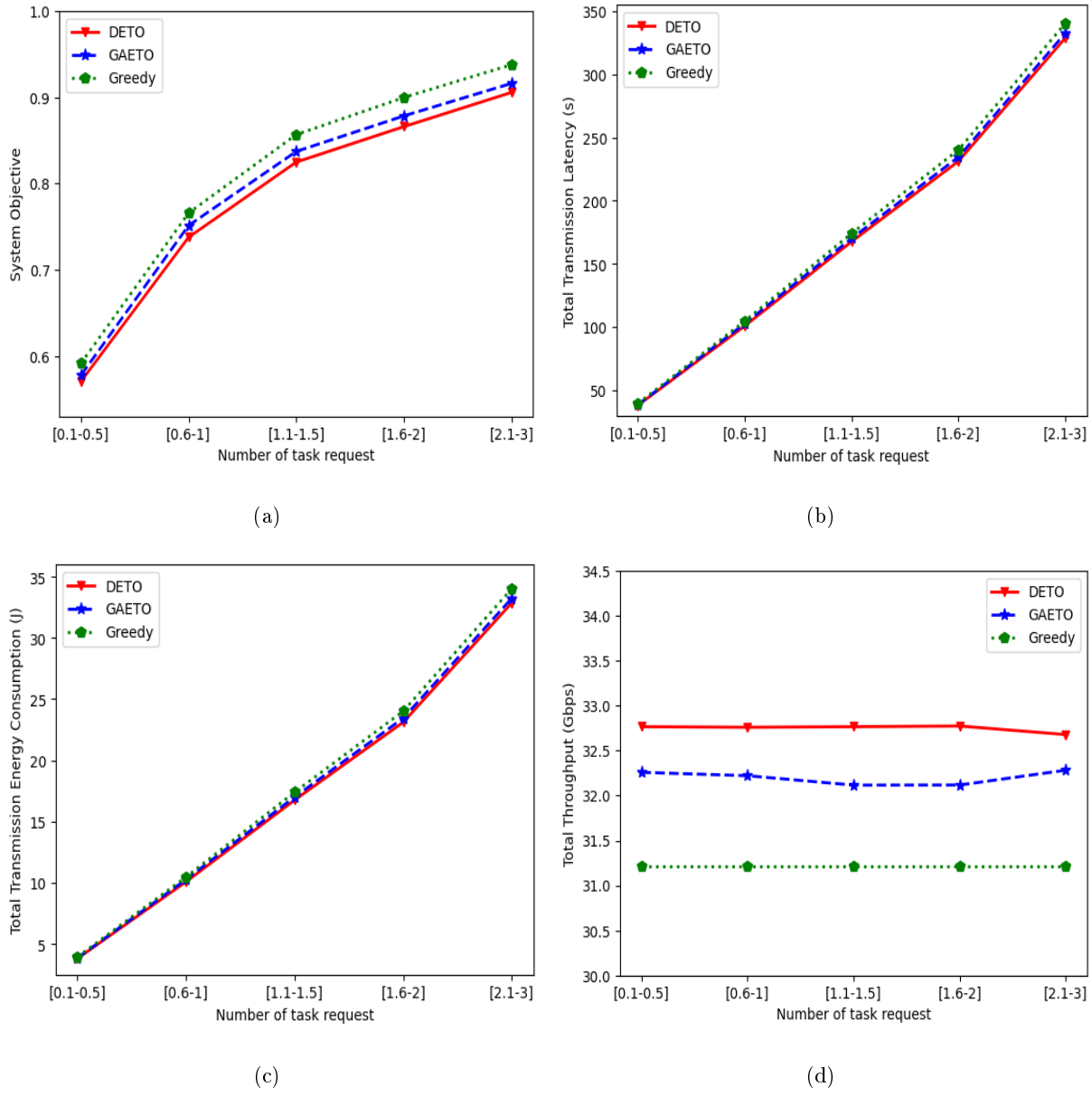


Figure 6-5: Effects of different numbers of task requests: (a) effects on the system objective, (b) effects on the total transmission latency, (c) effects on total transmission energy consumption, and (d) effects on total throughput.

Fig. 6-5 shows the effects of varying numbers of task requests. With the increasing number of task requests, the total task size concurrently expands. Consequently, more data bits need to be offloaded to UAV-MEC servers from the MDs. When MDs transmit a large volume of data bits to the UAV-MEC servers, it leads to higher transmission latency and increased energy consumption. Therefore, the system objective, total transmission latency, and total transmission energy consumption shown in Fig. 6-5(a), Fig. 6-5(b), and Fig. 6-5(c), respectively, demonstrate an increase consistent with the growth in task requests. In Fig. 6-5(a), it is observed that DETO always performs better than the GA and greedy approaches, obtaining a lower system objective across different task request ranges, whereas the system objective obtained by the greedy algorithm is the highest among them. For example, when the number of task requests is in the range of 0.6 to 1, the system objectives obtained by DETO and GAETO are 0.737 and 0.751, respectively, while the system objective for the greedy approach is 0.765.

The displayed curves in Fig. 6-5(b) and Fig. 6-5(c) show a steady and gradual increase as the number of task requests increased. This is a result of the growing need to offload more tasks, which requires more time and energy for transmission. For example, in Fig. 6-5(b), transmission latency obtained by DETO increases from 231.17 s to 328.66 s when task request changes from the [1.6-2] range to the [2-3] range. In comparison, GAETO increases from 234.49 s to 332.45 s, and the greedy approach increases from 240.19 s to 340.33 s, respectively. Similarly, in Fig. 6-5(c), transmission energy consumption increases for DETO, GAETO, and the greedy approach, ranging from 23.11 J to 32.86 J, 23.44 J to 33.24 J and 24.01 J to 34.03 J, respectively.

Fig. 6-5(d) illustrates the system's total throughput, with DETO and GA showing a slight up-and-down trend, while greedy remains unchanged. This is because throughput is influenced by the positions of MDs and the positions of the UAVs along the ellipsoidal paths. Throughput follows an increasing or decreasing trend when the number of MDs increases or decreases. However, in this scenario, the positions remain unchanged for a fixed number of MDs, even though the number of task requests varies. Thus, the solutions obtained by DETO and GAETO show slight fluctuations in total throughput. Notably, the DETO-based approach consistently shows higher throughput than GAETO and greedy. Moreover, the results imply that using the NN to approximate the Q-values enables the DQN-based solution to achieve better results for the considered problem. By maximizing the total reward and locally optimizing the DQN result, the agent can learn a strategy to

identify the ellipsoidal trajectory's parameters that provide a lower system objective. On the other hand, GA-based solutions partially depend on random processes, which may result in early convergence if genetic diversity is lost, resulting in suboptimal solutions. Moreover, because of the large solution space, the GA produces a considerable number of infeasible solutions in each iteration, whereas greedy uses fixed parameters to solve the considered problem. Therefore, finding a good solution is hard using GA and greedy approaches.

6.4.3 Effects of Different Numbers of UAVs

The performance of DETO is evaluated across a range of UAV counts, varying from one to four, by considering the system objective, total latency, total transmission energy consumption, and total throughput. Fig. 6-6(a) illustrates a decrease in the system objective as the number of UAVs rises. As UAVs increase in number from one to two, then to three, and to four, the system objective of DETO decreases by 7.60%, 13.13%, and 4.53%, respectively. Similarly, GAETO decreases by 6.60%, 13.27%, and 4.60%, respectively, while the greedy approach records reductions of 5.97%, 13.21%, and 4.88%, respectively. Additionally, at the relevant UAV count intervals, DETO consistently outperforms the greedy technique by 2.33%, 4.03%, 3.94%, and 3.58%, respectively.

For different numbers of UAVs, Fig. 6-6(b) and Fig. 6-6(c) compare the total transmission latency and transmission energy consumption for all algorithms; significantly, DETO performs better than the alternatives. Transmission latency and transmission energy consumption of MDs tend to reduce as the number of UAVs rises because increased accessibility of UAVs enables more efficient task offloading. The average distance between MDs and UAVs reduces as the number of UAVs rises. Due to the shorter travel distance, data transfer takes less time and energy, resulting in reduced latency and lower energy consumption for MDs. For example, when the number of UAVs increases from three to four, transmission latency of DETO, GAETO, and greedy decreases from 73.16 s to 69.84 s, from 74.53 s to 71.09 s, and from 76.16 s to 72.44 s, respectively. Similarly, transmission energy consumption of DETO, GAETO, and greedy decreases from 7.31 J to 6.98 J, from 7.45 J to 7.10 J, and from 7.61 J to 7.24 J, respectively.

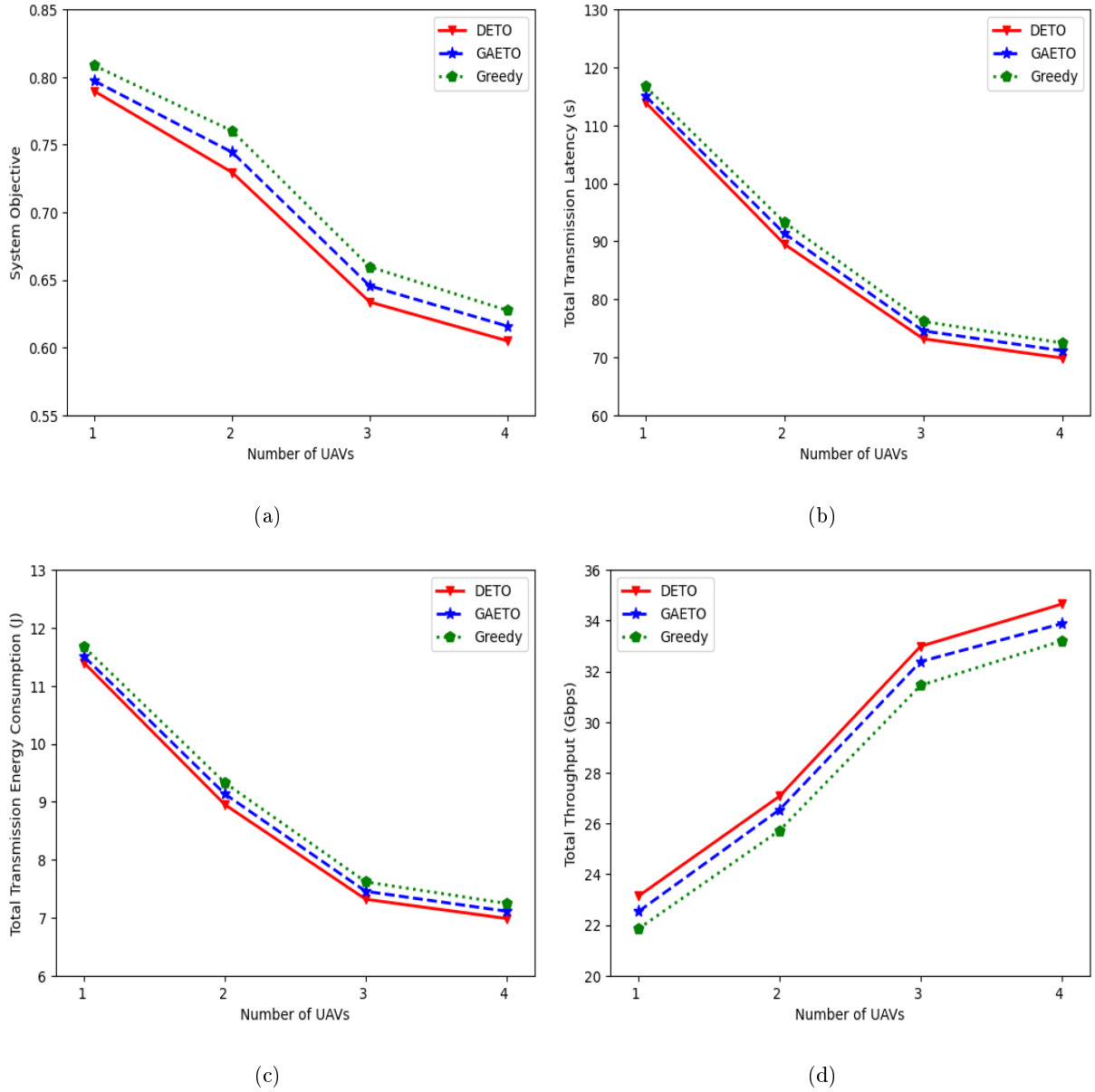


Figure 6-6: Effects of different numbers of UAVs: (a) effects on the system objective, (b) effects on the total transmission latency, (c) effects on total transmission energy consumption, and (d) effects on total throughput.

As the number of UAVs increases, Fig. 6-6(d) illustrates the corresponding growth in total throughput. This increase is because data transmission speeds can be boosted by reducing the distance between sending and receiving points with more UAVs. With the rise of UAVs, the performance of DETO improved by 17.02%, 21.84%, and 5.03%, respectively. Notably, the degree of improvement is slightly lower when the number of UAVs changes from three to four. This happens because the distances between MDs and UAVs do not change noticeably when there are three UAVs in the environment compared to when there

are four UAVs. The DETO solution achieves higher throughput compared to GAETO and the greedy approach with the increasing number of UAVs. For example, total throughput of DETO is 2.27% and 4.36% higher than that of GAETO and greedy approaches, respectively, with four UAVs.

Chapter 7

Concluding Remarks

In this article, we explored a multi-UAV-assisted MEC system to provide MEC services within a festival area, serving MDs that have delay-sensitive and computation-intensive AR/VR applications with diverse task offloading requests. We formulated an optimization problem with the objective of minimizing the weighted sum of total transmission latency and total transmission energy consumption of MDs. This was achieved by optimizing the center position, major radius, minor radius, and rotation angle of the UAV's ellipsoidal trajectory to determine the position, size, and shape of the ellipsoidal movement. A DRL framework called DETO has been developed to address the challenges associated with a large state space and high dimensional action space while considering constraints aimed at throughput maximization, as well as transmission latency and transmission energy consumption minimization. Two other approaches were used to compare the performance of DETO in diverse situations, including different numbers of MDs, different numbers of task request rates, and different numbers of UAVs. Results from simulations illustrated that the proposed DETO method performed better than the compared algorithms due to its ability to learn from experiences and progressively improve on solutions.

For future work, we aim to enhance the existing framework by considering additional optimizing variables, including the height of the UAV, taking into account the dynamic mobility of MDs within the festival area. In addition to optimizing task transmission latency and energy consumption, we will extend our objectives, to include the minimization of the total task completion time (including task transmission and processing time when offloaded to the UAV) and the overall system's energy consumption. We plan to apply a more advanced RL method and to investigate the effect of varying ρ on the system's objective.

Bibliography

[1] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Task offloading and resource allocation in mobile-edge computing system," in *2018 27th wireless and optical communication conference (WOCC)*. IEEE, 2018, pp. 1–4.

[2] I. A. Elgendy, S. Meshoul, and M. Hammad, "Joint task offloading, resource allocation, and load-balancing optimization in multi-uav-aided mec systems," *Applied Sciences*, vol. 13, no. 4, p. 2625, 2023.

[3] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of things applications: A systematic review," *Computer Networks*, vol. 148, pp. 241–261, 2019.

[4] A. Paramonov, A. Muthanna, O. I. Aboulola, I. A. Elgendy, R. Alharbey, E. Tonkikh, and A. Koucheryavy, "Beyond 5g network architecture study: fractal properties of access network," *Applied Sciences*, vol. 10, no. 20, p. 7191, 2020.

[5] F. Z. Fagroud, L. Ajallouda, E. H. B. Lahmar, H. Toumi, A. Zellou, and S. El Filali, "A brief survey on internet of things (iot)," in *International Conference on Digital Technologies and Applications*. Springer, 2021, pp. 335–344.

[6] Y. G. Kim, J. Kong, and S. W. Chung, "A survey on recent os-level energy management techniques for mobile processing units," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 10, pp. 2388–2401, 2018.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[8] 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*, vol. 19, no. 4, pp. 2322–2358, 2017.

[9] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[10] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Communications Surveys & Tutorials*, 2022.

[11] 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)*, vol. 52, no. 6, pp. 1–36, 2019.

[12] J. Yang, Q. Yuan, S. Chen, H. He, X. Jiang, and X. Tan, "Cooperative task offloading for mobile edge computing based on multi-agent deep reinforcement learning," *IEEE Transactions on Network and Service Management*, 2023.

[13] Z. Zhou, J. Feng, L. Tan, Y. He, and J. Gong, "An air-ground integration approach for mobile edge computing in iot," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 40–47, 2018.

[14] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in uav-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[15] J. Xia, P. Wang, B. Li, and Z. Fei, "Intelligent task offloading and collaborative computation in multi-uav-enabled mobile edge computing," *China Communications*, vol. 19, no. 4, pp. 244–256, 2022.

[16] G. Wu, Y. Miao, Y. Zhang, and A. Barnawi, "Energy efficient for uav-enabled mobile edge computing networks: Intelligent task prediction and offloading," *Computer Communications*, vol. 150, pp. 556–562, 2020.

[17] W.-T. Li, M. Zhao, Y.-H. Wu, J.-J. Yu, L.-Y. Bao, H. Yang, and D. Liu, "Collaborative offloading for uav-enabled time-sensitive mec networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, pp. 1–17, 2021.

[18] J. Tian, D. Wang, H. Zhang, and D. Wu, "Service satisfaction-oriented task offloading and uav scheduling in uav-enabled mec networks," *IEEE Transactions on Wireless Communications*, 2023.

[19] J. Košmerl and A. Vilhar, "Base stations placement optimization in wireless networks for emergency communications," in *2014 IEEE international conference on communications workshops (ICC)*. IEEE, 2014, pp. 200–205.

[20] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[21] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu, "3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage," *IEEE Wireless Communications Letters*, vol. 6, no. 4, pp. 434–437, 2017.

[22] I. Moon, L. T. Dung, and T. Kim, "Optimal 3d placement of uav-bs for maximum coverage subject to user priorities and distributions," *Electronics*, vol. 11, no. 7, p. 1036, 2022.

[23] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim, "Placement optimization of uav-mounted mobile base stations," *IEEE Communications Letters*, vol. 21, no. 3, pp. 604–607, 2016.

[24] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Efficient deployment of multiple unmanned aerial vehicles for optimal wireless coverage," *IEEE Communications Letters*, vol. 20, no. 8, pp. 1647–1650, 2016.

[25] N. Adam, C. Tapparello, W. Heinzelman, and H. Yanikomeroglu, "Placement optimization of multiple uav base stations," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–7.

[26] I. Valiulahi and C. Masouros, "Multi-uav deployment for throughput maximization in the presence of co-channel interference," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3605–3618, 2020.

[27] J. Lyu, Y. Zeng, and R. Zhang, "Cyclical multiple access in uav-aided communications: A throughput-delay tradeoff," *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 600–603, 2016.

[28] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for uav-enabled mobile relaying systems," *IEEE Transactions on communications*, vol. 64, no. 12, pp. 4983–4996, 2016.

[29] Y. Zeng and R. Zhang, "Energy-efficient uav communication with trajectory optimization," *IEEE Transactions on wireless communications*, vol. 16, no. 6, pp. 3747–3760, 2017.

[30] P. Yu, J. Guo, Y. Huo, X. Shi, J. Wu, and Y. Ding, "Three-dimensional aerial base station location for sudden traffic with deep reinforcement learning in 5g mmwave networks," *International Journal of Distributed Sensor Networks*, vol. 16, no. 5, p. 1550147720926374, 2020.

[31] X. Liu, Y. Liu, and Y. Chen, "Reinforcement learning in multiple-uav networks: Deployment and movement design," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8036–8049, 2019.

[32] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-agent deep reinforcement learning for task offloading in uav-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[33] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and A. Nallanathan, "Deep reinforcement learning based dynamic trajectory control for uav-assisted mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 10, pp. 3536–3550, 2021.

[34] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2019.

[35] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "Offloading using traditional optimization and machine learning in federated cloud-edge-fog systems: A survey," *IEEE Communications Surveys & Tutorials*, 2023.

[36] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang, "Selective offloading in mobile edge computing for the green internet of things," *IEEE network*, vol. 32, no. 1, pp. 54–60, 2018.

[37] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "etime: Energy-efficient transmission between cloud and mobile devices," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 195–199.

[38] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE, 2013, pp. 494–502.

[39] J. Wang, C. Jiang, H. Zhang, Y. Ren, K.-C. Chen, and L. Hanzo, "Thirty years of machine learning: The road to pareto-optimal wireless networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1472–1514, 2020.

[40] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[41] R. S. Sutton, A. G. Barto *et al.*, "Introduction to reinforcement learning. vol. 135," 1998.

[42] S. Akter, T.-N. Dao, and S. Yoon, "Time-constrained task allocation and worker routing in mobile crowd-sensing using a decomposition technique and deep q-learning," *IEEE Access*, vol. 9, pp. 95 808–95 822, 2021.

[43] Y. Zheng, X. Xie, W.-Y. Ma *et al.*, "Geolife: A collaborative social networking service among user, location and trajectory." *IEEE Data Eng. Bull.*, vol. 33, no. 2, pp. 32–39, 2010.

[44] X. Chen and G. Liu, "Energy-efficient task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge networks," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 843–10 856, 2021.

[45] ——, "Joint optimization of task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge network," in *2020 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2020, pp. 76–82.

[46] A. Alshahrani, I. A. Elgendy, A. Muthanna, A. M. Alghamdi, and A. Alshamrani, "Efficient multi-player computation offloading for vr edge-cloud computing systems," *Applied Sciences*, vol. 10, no. 16, p. 5515, 2020.

[47] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless vr-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15 749–15 761, 2021.

[48] X. Yang, Z. Chen, K. Li, Y. Sun, N. Liu, W. Xie, and Y. Zhao, "Communication-constrained mobile edge computing systems for wireless virtual reality: Scheduling and tradeoff," *IEEE Access*, vol. 6, pp. 16 665–16 677, 2018.

[49] W. Stadler, *Multicriteria Optimization in Engineering and in the Sciences.* Springer Science & Business Media, 1988, vol. 37.

[50] R. V. Rao, *Advanced modeling and optimization of manufacturing processes: international research and development.* Springer, 2011.

[51] A. Sbihi and R. W. Eglese, "Combinatorial optimization and green logistics," *Annals of Operations Research*, vol. 175, pp. 159–175, 2010.

[52] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE transactions on evolutionary computation*, vol. 17, no. 3, pp. 321–344, 2012.

[53] M. Agarwal and G. M. S. Srivastava, "A cuckoo search algorithm-based task scheduling in cloud computing," in *Advances in Computer and Computational Sciences: Proceedings of ICCCCS 2016, Volume 2.* Springer, 2018, pp. 293–299.

[54] X.-S. Yang, *Nature-inspired metaheuristic algorithms.* Luniver press, 2010.

[55] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[56] E. Pan, P. Petsagkourakis, M. Mowbray, D. Zhang, and A. del Rio-Chanona, "Constrained q-learning for batch process optimization," *IFAC-PapersOnLine*, vol. 54, no. 3, pp. 492–497, 2021.

[57] C. A. Coker, *Motor learning and control for practitioners.* Routledge, 2021.

[58] D. J. Richter, L. Natonski, X. Shen, and R. A. Calix, "Attitude control for fixed-wing aircraft using q-learning," in *International Conference on Intelligent Human Computer Interaction.* Springer, 2021, pp. 647–658.

[59] R. Maivizhi and P. Yogesh, "Q-learning based routing for in-network aggregation in wireless sensor networks," *Wireless Networks*, vol. 27, pp. 2231–2250, 2021.

[60] W. Park, S. Kim, K. L. Kim, and J. Kim, "Alphago's decision making," *Journal of Applied Logics—IFCoLog Journal of Logics and their Applications*, vol. 6, no. 1, 2019.

[61] J. Zhang, "Ai based algorithms of path planning, navigation and control for mobile ground robots and uavs," *arXiv preprint arXiv:2110.00910*, 2021.

[62] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[63] L. M. Belmonte, A. S. García, R. Morales, J. L. de la Vara, F. López de la Rosa, and A. Fernández-Caballero, "Feeling of safety and comfort towards a socially assistive unmanned aerial vehicle that monitors people in a virtual home," *Sensors*, vol. 21, no. 3, p. 908, 2021.

[64] Y. Chen, N. Li, X. Zhong, and W. Xie, "Joint trajectory and scheduling optimization for the mobile uav aerial base station: a fairness version," *Applied Sciences*, vol. 9, no. 15, p. 3101, 2019.

[65] X. Chen, Z. Yang, N. Zhao, Y. Chen, J. Wang, Z. Ding, and F. R. Yu, "Secure transmission via power allocation in noma-uav networks with circular trajectory," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 033–10 045, 2020.

[66] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 569–572, 2014.

[67] A. Al-Hourani, S. Kandeepan, and A. Jamalipour, "Modeling air-to-ground path loss for low altitude platforms in urban environments," in *2014 IEEE global communications conference*. IEEE, 2014, pp. 2898–2904.

[68] X. Li, L. Zhou, Y. Sun, and B. Ulziinyam, "Multi-task offloading scheme for uav-enabled fog computing networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, pp. 1–16, 2020.

[69] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[70] L. Lyu, F. Zeng, Z. Xiao, C. Zhang, H. Jiang, and V. Havyarimana, "Computation bits maximization in uav-enabled mobile-edge computing system," *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10 640–10 651, 2021.

[71] J. Zhang, Y. Zeng, and R. Zhang, "Spectrum and energy efficiency maximization in uav-enabled mobile relaying," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.