February 2023
Master's Degree Thesis

Reinforcement Learning-Based Offloading in Unmanned Aerial Vehicle-Aided Edge Computing Systems

Graduate School of Chosun University
Department of Computer Engineering
S. M. Asiful Huda

Reinforcement Learning-Based Offloading in Unmanned Aerial Vehicle-Aided Edge Computing Systems

무인 비행체 활용 에지 컴퓨팅 시스템에서의 강화 학습 기반 오프로딩

February 24, 2023

Graduate School of Chosun University

Department of Computer Engineering

S. M. Asiful Huda

Reinforcement Learning-Based Offloading in Unmanned Aerial Vehicle-Aided Edge Computing Systems

Advisor: Prof. Sangman Moh, Ph.D.

A thesis submitted in partial fulfillment of the requirements for a master's degree

October 2022

Graduate School of Chosun University

Department of Computer Engineering

S. M. Asiful Huda

This is to certify that the master’s thesis of

S. M. Asiful Huda

has been approved by examining committee for the thesis requirement for the master’s degree.

후다 아시풀의

석사학위논문을 인준함

위원장	조선대학교	교수	신석주	(인)
위 원	조선대학교	교수	강문수	(인)
위 원	조선대학교	교수	모상만	(인)

2022년 12월

조선대학교 대학원

Table of contents

List of Figures

List of Tables

Abstract

Reinforcement Learning-Based Offloading in Unmanned Aerial Vehicle-Aided Edge Computing System

S. M. Asiful Huda
Advisor: Prof. Sangman Moh, Ph.D.
Department of Computer Engineering
Graduate School of Chosun University

Unmanned aerial vehicles (UAVs) have recently shown an ever-increasing trend in many areas of civil applications as well as military applications such as surveillance, reconnaissance, augmented reality, etc. Due to ease of flexibility in terms of mobility and cost, UAVs can be deployed anywhere to provide seamless connectivity where terrestrial infrastructure is not available or damaged. However, both performing communication and executing computation-intensive tasks become huge burden for a UAV to perform a mission because of the battery lifetime being very limited. To mitigate this issue, mobile edge computing (MEC) is considered as a reliable and effective platform which can provide additional computational support at the edge of the network. Incorporating an MEC server enables to offload the computationally intensive tasks to be offloaded from UAVs to the edge server. This reduces the transmission delay significantly compared to a cloud server. However, based on the dynamic characteristics of different tasks, offloading decision is a major issue. Furthermore, most tasks have a stringent deadline that the task needs to be executed.

UAV swarm-enabled MEC systems can be effectively leveraged to address this problem. Most of the existing studies consider the assumption that a single-UAV system has sufficient communication and computation capacity to perform any mission which is very unlikely. In this thesis, we propose a deep reinforcement learning based computation offloading (DRLCO) scheme using double deep Q-learning for surveillance applications. DRLCO minimizes the total weighted cost by jointly considering task execution delay and energy consumption. The DRLCO technique can effectively address the dynamic environment and based on the task characteristics, effective offloading decisions are made. The performance of DRLCO is evaluated through computer simulation and compared with conventional offloading schemes. The simulation results show that proposed DRLCO mechanism can outperform the conventional offloading techniques in terms of total offloading cost, task execution delay and energy consumption.

요 약

무인 비행체 활용 에지 컴퓨팅 시스템에서의 강화 학습 기반 오프로딩

후다 아시풀
지도교수: 모상만
컴퓨터공학과
조선대학교 대학원

무인 비행체(UAV)는 최근 감시, 증강 현실, 가상 현실 등과 같은 군사 응용 분야뿐만 아니라 민간 응용 분야의 많은 영역에서 지속적으로 증가하는 추세를 보이고 있다. 높은 이동성과 낮은 비용으로 인해 UAV 는 지상 기반 시설을 이용할 수 없거나 손상된 곳에서 원활한 연결을 제공하기 위해 어디에나 배치될 수 있다. 그러나 제한된 배터리 용량 때문에, 통신과 계산 집약적 작업을 함께 수행하는 것은 UAV 임무 수행에 큰 부담이 된다. 이 문제를 완화하기 위한 모바일 에지 컴퓨팅(MEC)은 네트워크 에지에서 추가 컴퓨팅 지원을 제공할 수 있는 안정적이고 효과적인 플랫폼으로 간주된다. MEC 서버를 활용하면 계산 집약적인 작업을 UAV 에서 에지 서버로 오프로드하여 클라우드 서버에 비해 전송 지연을 크게 줄일 수 있다. 그러나 서로 다른 작업의 동적 특성에 따라 오프로드 결정이 주요 문제가 된다. 또한 대부분의 작업에는 엄격한 실행 마감 시간이 있다.

UAV 군집 지원 MEC 시스템을 효과적으로 활용하여 이 문제를 해결할 수 있다. 대부분의 기존 연구는 단일 UAV 시스템이 임무를

수행하기에 충분한 통신 및 계산 용량을 가지고 있다는 가정한다. 본 논문에서는 보안 감시 응용을 위한 심층 강화 학습 기반 계산 오프로딩(DRLCO) 기법을 제안한다. DRLCO 기법은 이중 심층 Q-학습을 사용하고 태스크 실행 시간과 에너지 소모량을 함께 려함으로써 총 가중치 비용을 최소화한다. DRLCO 기법은 동적 환경을 효과적으로 해결할 수 있으며 작업 특성에 따라 효과적인 오프로드 결정이 내려진다. 컴퓨터 시뮬레이션을 통해 DRLCO 성능을 평가하고 기존 오프로드 방식들과 비교한다. 시뮬레이션 결과에 의하면, 제안한 DRLCO 메커니즘이 총 오프로드 비용, 작업 실행 지연 및 에너지 소비 측면 등에서 기존 오프로드 기법들보다 우수하다.

1. Introduction

1.1 Overview

The increasing growth of smart devices has resulted in a dramatic change in society, which now heavily relies on cellular technologies. Online streaming services and social-networking sites have become popular and useful across all demographics. The resultant data increase has created intense burdens for mobile service providers. Without proper measures for storing and processing such workloads, cellular networks will become even more congested, resulting in deteriorated quality and slower download speeds. Hence, additional computational resources are necessary for mobile devices. Furthermore, the long-term evolution of 5G technology has inspired a wide range of services that require high computational tasks, for which the designated devices are ill-equipped to handle [1].

Mobile edge computing (MEC) is a promising solution that leverages cloud servers deployed in support of mobile devices to mitigate computational workloads via process offloading. In 2009, the first edge-computing concept (i.e., cloudlet) was proposed. Cloudlets allow mobile users to take advantage of cloud services, but they require users to swap between Wi-Fi and cellular networks during use [2].

Tasks can also be executed locally on mobile devices by leveraging the concept of an *ad hoc* cloud [3]. It enables multiple user devices to combine their computational resources to process tasks. Notably, offloading to an edge server directly improves the quality of experience and battery lifetime [4]. In 2012, Cisco proposed the concept of computation offloading [5]. Hence, any mobile device having constrained resources can wirelessly pass processing tasks to other devices having sufficient resources. Those other devices then complete the tasks and transmit the results back to the mobile devices. Unfortunately, this method continues to fall short of expectations, owing to the characteristics of wireless networks in rural and mountainous areas. Furthermore, cases of emergency response should always take precedence. Thus, even in the most ideal environments, maintaining the quality of experience and energy efficiency while avoiding communication delays is difficult.

Unmanned aerial vehicle (UAV)-enabled MEC servers have emerged as promising candidates for handling computationally intensive task loads in areas lacking ground infrastructure. Due to the high mobility and low cost, both academia and industry have shown interest in UAV research. There has been an enormous amount of studies that emphasizes on the effective communication and localization techniques for the UAV network [6]–[14]. In Ref. [6], the authors designed a routing protocol where UAVs are divided into cluster and presented a comparative study. The authors in [7], [8] proposed routing protocol using reinforcement learning (RL). UAVs can send raw images from affected areas to any base station (BS) for fast mobility [15]. It was recently noted that UAVs can also act as MEC servers or MEC relays. UAV-mounted MEC relay services use line-of-sight (LoS) links to transmit computational tasks to MEC servers situated on the ground as they hover over areas where it is otherwise challenging to set up a cloud or edge computing-based solutions.

In military applications, the vital task is to capture information from a given region and send it to the nearest BS to identify or track objects from a distance, which generally is very intricate and burdensome for any human. UAVs can also provide target-tracking geographical data-capture services. A UAV-aided wireless sensor network is another promising paradigm to enhance the energy efficiency of the sensor nodes in collecting data in various areas of commercial applications [16] However, transmission delays are hindrances because longer processing times result in late responses. Integrating MEC servers into UAVs effectively eradicates this problem. Furthermore, UAVs can simultaneously operate as computation servers and relay nodes [17], resulting in faster decision-making during emergency scenarios while increasing efficiency with potential real-time computational capabilities. Notably in such scenarios, UAVs can either process computational tasks locally, or they can offload them to other edge servers.

Despite being a promising approach, it has been less explored in the existing literature. For UAV-enabled MEC networks, the existing literature emphasizes localization [12], routing protocol [18], bandwidth allocation [19], path planning [20], area coverage [21], topology control [22], etc. These studies mostly focused on optimizing the rescue efficiency by minimizing the response delay and enhancing resource utilization. In a UAV-assisted surveillance system, UAVs perform the duty of covering a certain region and execute computationally intensive tasks in which both delay and energy consumption are crucial metrics [23]. In large-scale 3D areas, single-UAV systems often fail to successfully accomplish complex missions because of their limited energy capacity, although they provide sufficient coverage for a specific area. In such cases, a swarm of UAVs comprising many small and low-cost UAVs was observed to be effective in performing missions in large areas [21]. With the availability of a mobile edge server, the offloading computation task minimizes the task execution delay and energy consumption involved in offloading the task from the UAV.

In the UAV-MEC system, one of the most crucial decisions is where the task execution occurs (e.g., local execution, edge server). This may depend on various metrics such as the number of tasks, channel quality, UAV position, and edge nodes [24]. The UAV-enabled MEC system may fail to execute a task when the computational load increases significantly with a limited number of UAVs and insufficient computing resources. A promising solution is to utilize a UAV-enabled MEC system along with a base station (BS)-assisted MEC to enable the UAVs utilize the MEC services provided by the BS [25]. The offloading decision becomes more challenging under dynamic environmental conditions, where the characteristics of the network are highly dynamic. Thus, capturing accurate information to determine an offloading decision becomes difficult. Offloading decision-making has been explored broadly which mainly focuses on using traditional optimization methods and heuristic methods. [4], [26], [27]. However, because of the complicated constraint conditions related to practical environments, these algorithms cannot obtain significant results. For example, global optimization methods require the problem formulation to be as simple as possible to enable it to be decomposed into subproblems. Reinforcement learning (RL) is considered a viable solution for such complex and dynamic environments, as it can model large and complex environments. Incorporating deep learning with RL yields deep reinforcement learning (DRL) algorithms such as a deep Q-learning network (DQN), that is able to obtain significant results in the absence of previous environment knowledge [28]–[30].

Motivated by the above discussion, in our thesis, we consider a hierarchical UAV-enabled MEC architecture comprising a head UAV (H-UAV), a team of member UAVs (M-UAVs), and a BS-assisted MEC to enhance the task execution time and energy efficiency for surveillance application scenarios. The M-UAV senses the area and generates a computing task to be processed.

When the number of tasks increases significantly and the processing capacity of the local edge server is at a maximum, the M-UAV can offload the task to either the H-UAV or ground edge server. The decision to offload the task depends on various factors, such as the task type and computation capacity at the H-UAV.

1.2 Research Objective

The collaboration between the H-UAV and ground edge server enables a minimizing the energy consumption as well as task delay. Based on the above discussion, we formulate a weighted sum cost minimization problem by jointly considering task execution delay and energy consumption. In our study, we emphasize both on energy consumption and task execution delay because UAV have limited energy and tasks may have stringent deadline. The major contributions of this study are summarized as follows:

- We design a network consisting of a UAV swarm and a ground MEC server in which the swarm is divided into individual coalitions. In each coalition, the M-UAV senses a plane area and generates a computation-intensive task that can be processed directly at the aerial edge server or can utilize additional MEC resources from the BS.
- Considering the limited energy of the UAV, communication and task execution delay, we design the computation offloading decision-making problem of the M-UAV as a weighted cost minimization problem considering both the energy consumption of the UAV and the task execution time.
- We formulate the weighted cost minimization problem leveraging deep reinforcement learning (DRL) scheme. The state, action, and reward of

the DRL are designed. Each agent (M-UAV) acts with the environment and selects an action that provides a reward. The optimal offloading policy is achieved by maximizing the cumulative reward.

- To maximize the expected cumulative reward (by minimizing the weighted sum cost), we propose a DDQN-based decentralized DRLCO scheme in which each M-UAV is considered to be an agent and can make the offloading decision using local observation.
- We performed a comprehensive numerical simulation to verify the convergence of the proposed method and compare our results with those of other conventional schemes (local computing, edge execution, and DQN) with varying parameter configurations. The superiority of the proposed scheme was investigated using different performance metrics.

1.3 Thesis Layout

Rest of the thesis is organized as follows:

Existing offloading techniques are summarized next. The system model of our proposed work is described in Section 3. The proposed DRLCO technique is demonstrated in detail along with the devised algorithm in Section 4. The performance of the proposed algorithm is then evaluated through computer simulation and compared with conventional offloading schemes in Section 5. Finally, the study is concluded in Section 7.

2. Related Works

2.1 Existing Offloading Techniques in UAV-MEC

With the drastic increase of Internet of Things (IoT) devices and emerging computation-intensive and delay-sensitive applications (e.g., augmented reality, face recognition, virtual reality) MEC have become a promising trend which enables the end-users to offload the heavy computation-intensive tasks to enhance the overall system performance. With edge servers installed at the edge of the network, the end-users have the option to offload the task to the edge server rather than offloading to the cloud. Thus, the overall task execution delay can be reduced significantly. Furthermore, the tasks having different requirements demands to be executed considering the dynamic task characteristics. For example, tasks that have strict deadline with less computational requirement may be executed in the UAV than sending to the edge server. Similarly, tasks having less strict deadline and high computation can be well executed to the edge server. The major drawback with the introduction of next-generation wireless networks is going to be limited battery lifetime and computation resource of the mobile users for performing the task with low latency requirements, which are introduced by the services and applications such as virtual reality, telesurgery, autonomous driving, and UAVs [31]. UAV-MEC systems can comfortably handle such emergent situations where terrestrial MEC servers are out of service or overloaded [32]. Having said that, existing studies have greatly emphasized on the computation offloading aspects of UAV-enabled MEC to enhance the system efficiency by considering different practical scenarios in their study.

RL-based algorithms have been used widely in various fields of wireless communication, owing to the uncertain behavior of the communications environment. Because the network entities must act on those uncertain behaviors, RL is a near-perfect solution; it enables agents to take random actions to reach an optimal policy, especially in complex environments [33]. Existing studies show that RL algorithms can deal with massive amount of offloading request generated by mobile devices autonomically. In such a scenario, offloading decision is a very complex task because it involves several metrics such as availability of resource, resource demand, and current network status [34].

A hierarchical and cooperative coalition formation-based offloading algorithm was proposed in [35] wherein the authors placed UAVs as players in a game scenario acting in such a way that the overall payoffs were maximized. They presented a hierarchical structure in which the first level resulted in a coalition that could act cooperatively. However, to demonstrate the noncooperative manner of individual stations owned by different service providers, the second level comprised identical subgames. The authors showed that their algorithm reached an optimal state wherein the formation was stable, and the BSs followed a combined strategy to produce an offloading strategy. The authors proposed a game-theory-based RL approach that could find an optimal strategy for BSs via a Markov decision process. They demonstrated that BSs and UAVs can adapt mixed strategies, even when the players are not aware of the actions taken by other BSs. To evaluate performance, they compared their proposed strategy to other benchmark methods in terms of the numbers of users and payoffs achieved when reaching an optimal state.

A novel algorithm for offloading computational tasks to UAVs was presented in Ref. [36] which focuses on the cooperative behavior of UAVs, enabling them to offload tasks to other UAVs. The authors showed that UAVs can collaboratively work with others to maximize total network utility and to equalize and reduce computational and communication costs. To achieve this, they focused on limited computational resources and proposed a Markov decision process combined with a deep RL (DRL) to optimize target parameters, assuming that the UAV could also perform as a computational server alongside an edge server. This research showed that the system does not need to send all information to the central operator. This allows the implementation of such systems in areas where there is an immense task load generated from IoT devices.

A Space Air-Ground Integrated Network (SAGIN) network was demonstrated in Ref. [37] for offloading computational tasks, where UAVs provide additional computational support as edge servers. The authors considered a remote area with IoT devices on the ground performing heavy computationally intensive tasks, such as surveillance and monitoring. Because the area lacked cellular communications, the SAGIN network was equipped with complete caching, edge computing, and network provisioning capabilities. Their work helped to determine the allocation of resources alongside the scheduling of offloading tasks in a dynamic network. They further investigated the Markov decision process to better understand uncertain system dynamics. Utilizing the Markov decision process has a significant advantage in deciding upon uncertain system entities. To better handle the dynamicity of the network, the authors suggested an on-the-fly approach for DRL. They utilized common policy-gradient methods to act in the complex action space, and for fast convergence to the optimal, an actor-critic technique was adopted. This study revealed that such a system could generate optimal performance by jointly allocating resources in a virtual machine as well as via task assignment. The system minimized the total cost better than other benchmark methods.

A DRL-based hybrid load-balancing strategy was presented in Ref. [38] wherein the authors investigated the mitigation of limited computational resources by introducing UAVs as MEC nodes to improve computational capabilities. The authors suggested a DE-based UAV deployment algorithm that optimized task execution time. Initially, they considered that all UAVs were at randomly assigned positions. Then, they allocated the maximum load to each depending on its location. To tackle the intercommunication burdens among UAVs and IoT devices, they utilized a GAP, and an approximation-based algorithm was presented to determine the connections between IoT nodes and the UAV. To manage incoming tasks, they proposed a DRL algorithm for assigning tasks to the UAV. They demonstrated that it could adapt to dynamic system environments while controlling the effective allocation of network resources, enabling more effective handling of recently arriving tasks. The decision to either offload at a UAV or process the task locally created a binary offloading problem.

A novel cooperative technique for offloading tasks to other UAVs to tackle power consumption, total delay, and job loss was presented in Ref. [39]. Here, the authors considered a situation wherein there was a significant amount of latency caused by ground-generated data taking up a long time to reach the UAV. In the proposed system, the UAVs were equipped with a computational facility similar to an edge server. Owing to the excessive number of computational resources onboard and the limited power of the UAV, the authors introduced a system controller that could decide whether to turn the central controller on or off. Furthermore, the controller could offload tasks from the overloaded UAV to an underloaded one by leveraging RL to reach an optimal policy under uncertain and dynamic conditions in large and complex action space. The authors presented three offloading approaches using greedy and traversal-based algorithms as evaluation benchmarks. Their method minimized the total system cost for computing and reached the Nash equilibrium.

The authors of Ref. [40] attempted to reach an optimal offloading state by concentrating on total system costs. They considered a system having multiple wireless users and UAVs having limited capacities. Optimally, wireless users can offload tasks to UAVs. The authors considered that the UAVs could be recharged wirelessly using solar panels. Under this scenario, the authors formulated an optimization problem, wherein the focus was on minimizing the total cost of offloading by combinedly considering energy consumption, bandwidth costs, and total delay. Considering the large action space, the authors used the K-means algorithm to classify several types of computational tasks, resulting in the reduction of the dimensionality of the action space because a large action space can slow down the learning rate.

An optimized deep-Q-network (DQN) for DRL-based offloading, called double DQN (DDQN), which minimizes the total cost by solving the overestimation problem was proposed in [41]. The author compared the performance of the proposed method in terms of task arrival probability, cumulative reward delay cost, and the rate of arrival of renewable resources. They then compared their proposed method to four other benchmark techniques, wherein the location of offloading was considered. The proposed UACODRL technique significantly outperformed the other benchmarks in terms of average cumulative reward. Although the algorithm performed significantly well, with an increase in users, performance was negatively affected because it also increased the power consumption. Hence, in a large and dense network, system performance will suffer. The proposed scheme can be applied to heterogeneous wireless networks, and the UAVs can be charged wirelessly using a renewable energy supply.

The Multi-agent RL (MARL) approach was proposed in Ref. [42] where two RL agents performed two different tasks individually in a UAV-mounted edge-computing architecture. The novelty of this study lay in its consideration of the interdependency of the task and its dynamicity. At each time slot, when a task arrives at the queue, the two agents are responsible for deciding the target device for execution and the amount of bandwidth needed. The main goal is to minimize the average response time by determining an optimal policy that incorporates task assignment and bandwidth allocation.

In urban areas, UAVs are used primarily for tracking and monitoring city areas. From the perspective of object detection in surveillance and monitoring applications, the UAV's next action depends on the results obtained from object detection. Therefore, delays in task execution are crucial performance determinants [24]. To address this, a system that considers network load parameters was established to minimize the sum of energy expense and delay. The network comprises a UAV, an edge server, and an access point that connects the UAV to the server. The authors formulated the optimization problem by utilizing a semi-Markov process to minimize total costs. The authors defined the Markov process states as primary, termination, data transmission, and queuing types. The binary decision was made by the UAV, referring to either local computing or offloading. Performance was evaluated using parameters that depend on server load and wireless channel quality.

Simulations showed that the agents who were aware of system dynamics had advantages over agents lacking this information because they could not make decisions based on channel situation. Furthermore, the trajectory of the UAV is crucial, and the authors illuminated the average delays over the trajectory concerning system load. The study was concluded by demonstrating a good delay reduction. However, The system does not consider the UAV mobility model. In monitoring and tracking systems, the UAV will monitor certain areas wherein obstacles are unpredictable. These considerations make the study much more complex and require further attention.

Preventing eavesdropping and data transmission interception between UAVs and MECs is critical. The authors of Ref. [43] presented a method that considered this scenario. To mitigate the latency caused by the limited computational capacity of mobile devices, the authors aimed to reduce total energy consumption by allocating resources and ratios. To achieve this, the study proposed a DQN, which falls short of dynamically optimizing multiple performance metrics. The proposed method converged at ~8,000 iterations. To evaluate the effectiveness, they compared their method with two others: a local computing scheme and an offloading method based on total weighted cost, MEC processing capability, weight factoring, and total users. Simulations showed that the proposed method achieved a much lower cost when the number of users was set to five. It also demonstrated adaptivity when determining tradeoffs between communications and computational resources while reducing the eavesdropping threat.

2.2 Comparison of Existing Offloading Algorithms in UAV-MEC

The existing offloading techniques in UAV enabled MEC are compared based on the operational characteristics in Table 1. From the comparison, we observe that most offloading techniques consider binary offloading mechanism which is either the task is locally computed or is offloaded to the edge server. UAV played the role of edge server in most studies, however, some studies consider the option of relay node as well. Minimizing the energy consumption and total delay is the most significant metric, which is not surprising. Offloaded tasks deteriorate the battery storage of the UAV, and the inability to successfully execute tasks and return the results impacts the overall system. Consideration of successful task completion, task arrival, task deadline can be crucial to enhance the overall task completion performance. Additionally, in a multi-UAV network, because the UAVs have dynamic topologies owing to their high mobility, ensuring reliable, secure, and effective communications amongst nodes is a major challenge. Hence, most studies focused on either military or civilian applications in densely populated areas for joint metrics, offloading delays, effective UAV deployments, trajectories, mission completion times, and resource allocations to enhance overall system performance. For example, for crowd surveillance, video capturing, video processing, face recognition, and face matching, tremendous overhead is generated in a single UAV–MEC system. In such cases, collaborative methodologies may provide potential solutions because they enable cooperative task execution.

To leverage the computing facilities provided by the MEC server, an optimization approach for DL-based target-tracking UAVs, which is widely used in urban areas, was proposed. To perform this type of task effectively using the limited computing resources and battery lifetime, offloading a portion of the task to a MEC server is required. For this reason, the authors in Ref. [44] proposed a hierarchical architecture in which one portion of DL execution is executed at the UAV, and another portion is executed at the edge server. The study emphasized several crucial performance metrics, such as interference error rate, input data quality, and transmission bandwidth. To meet objectives, the authors considered a MEC system with multiple UAVs to track either a person or a vehicle. While executing the task, the lower level of the DL model was executed to save bandwidth for transmission. Subsequently, the next portion of the model was executed at the MEC to enhance the inference error rate. Offloading can either be binary or partial. The constraint under consideration fulfills both types.

This study also emphasized the availability of the wireless channel because its unavailability makes it impossible for the task to be offloaded. The offloading decision depends on image quality. For example, if it is good, object tracking is executed at the UAV, otherwise, it is executed at the MEC node. The proposed method was evaluated in terms of total cost and interference error rate according to the offloading ratio and total number of UAVs. Simulation results demonstrated that with total offloading, the tasks were executed at the MEC server, which supports the correctness of the proposed method.

The authors of Ref. [26] considered the allocation of data and trajectory optimization intending to minimize the total energy consumption in a hovering UAV-enabled MEC system to provide computational services to ground mobile terminals (MTs). Specifically, the authors considered different energy consumption rates of specific MTs when individual devices consumed more computational resources, and most of the computing occurred at the devices instead of the flying edge server. In the system model, the authors considered that all tasks were independent and could be divided into several portions to be offloaded.

To effectively optimize both the UAV trajectory and the allocation of bits, the authors divided the problem into two subproblems and formulated them as convex optimization problems. The authors proposed the JTDATO algorithm, which solves the individual subproblems while jointly optimizing both the trajectory and data allocation. To evaluate system performance, the authors presented an optimized trajectory alongside a data transfer and maximum energy consumption scheme. The proposed method-maintained speed while traversing trajectory when the distance between the UAV and the MTs was minimal. Therefore, more MTs can offload tasks to edge servers, thus conserving energy. To further investigate the performance of the proposed method, the authors also considered the random deployment of MTs. In this case, the energy consumption was much lower than baseline methods. However, the mobility of MTs can change the energy consumption scenario assumed in this study. High mobility adds more dynamicity to the system.

However, there are several challenges associated with collaborative UAV deployments. Communication links can drop between UAVs, owing to battery drainage, malfunctions, and terrain. Overlapping areas must be controlled to maximize coverage. Most studies utilized a traditional optimization method in which the overall network information is obtained by the UAV and utilized to make an offloading decision. A central controller was also assumed in some studies to collect the required network information, channel information, etc.

In contrast, in coalitional UAV systems, the decision is made by each member by interacting with others, enabling them to collaboratively decide upon an optimization plan in a distributed fashion [45].

Most studies consider tasks generated from IoT devices or mobile users considering the data-intensive task requirement. However, the task generated from a vehicular network in surveillance applications poses a different challenge because UAVs need an efficient mobility plan to support the extremely dynamic nature. Furthermore, the consideration of topology formation depending on the dynamic nature of the environment is overlooked by most studies. Thus, further research is needed to design robust offloading algorithms for such complex application scenarios.

Table 1. Operational characteristics of offloading techniques for UAV-MEC.

Algorithm	**Offloading type**	**UAV role**	**Performance metrics**	**Application scenario**	**Optimization objectives**
HGTRL	Binary	Edge server	Offloading delay, Energy consumption	Mobile device	Maximizes the long-term payoff (inversely proportional to delay and energy cost)
COUMEC	Binary	Edge server	Service drop rate, Network utility	Mobile device	Maximizes the total utility by deciding optimal offloading and resource allocation policies.
SAG-IOT	Binary, cloud processing	Edge server	Average total delay, Total cost	IoT user	Minimizes the total system cost in terms of delay, energy consumption of IoT user, edge, and server usage cost
DE-GAP-DRL	Binary	Edge server	Task Load balance, Average transmission cost	Ground IoT device	Minimizes the average slowdown of tasks in UAVs
GTCO	Binary	Edge server	Mean delay, Loss probability, Power consumption gain	Ground device	Maximizes an objective function defined in terms of power consumption, delay, and

					loss probability by offloaded and non-offloaded jobs
UACODRL	Binary	Edge server, load relay	Cumulative reward	Wireless user	Minimizes the weighted sum of the delay, energy consumption, and bandwidth cost
MARL	Binary	Edge server	Average mission response time, Communication, and processing time of the missions	Surveillance	Minimizes the mission response time
DP-DRL	Binary	Edge server	Offloading and local computing probability, Average delay	Building inspection	Minimizes the weighted sum of delay and energy expense
DDQN	Partial	Edge server, load relay	Total weighted cost, Latency, Energy consumption	IoT device	Minimizes the weighted cost of latency and energy consumption
HMTD	Partial	Edge server	Total cost, Inference error.	Target tracking	Minimizes the total cost.
UMEC	Partial	Edge server	Energy consumption, Computation load	Surveillance	Minimizes the energy consumption of UAVs
JTDATO	Hybrid	Edge server	UAV Trajectory Energy consumption among Mobile terminals	Mobile terminals	Minimizes the energy consumption of all mobile terminals
CCCP	Hybrid	Edge server	UAV trajectory, Energy consumption of UAV, Transmission power of all users	User device	Minimizes the total transmission energy consumption of all users

3. System Model

3.1. Motivation Scenario

In this thesis, we focus on the implementation of a UAV swarm-enabled MEC system consisting of a multi-UAV network and a ground BS-enabled MEC that provides computational support to the UAVs. The UAV swarm is assumed to perform a video surveillance task to avoid unexpected occurrences during the event. Owing to their high mobility, UAVs are deployed around the venue to capture videos using an onboard camera for face recognition tasks. Owing to such tasks being highly computation intensive and UAVs having a limited battery lifetime, in this thesis, we address the minimization of the energy consumption of the UAV and the task completion delay by exploiting the benefits of additional computation service provided by the ground BS-MEC server. Thus, this system can replace human involvement with reduced costs and faster response times.

3.2. Network Model and Assumptions

We consider a UAV swarm-enabled MEC system in which the UAVs are assumed to perform a surveillance mission in an urban area as shown in Figure 1. We envision a widely used application scenario dedicated to public venue use (e.g., stadium) based on the ETSI framework [46]. Typically, ETSI considers stadiums as a potential use case requiring MEC services

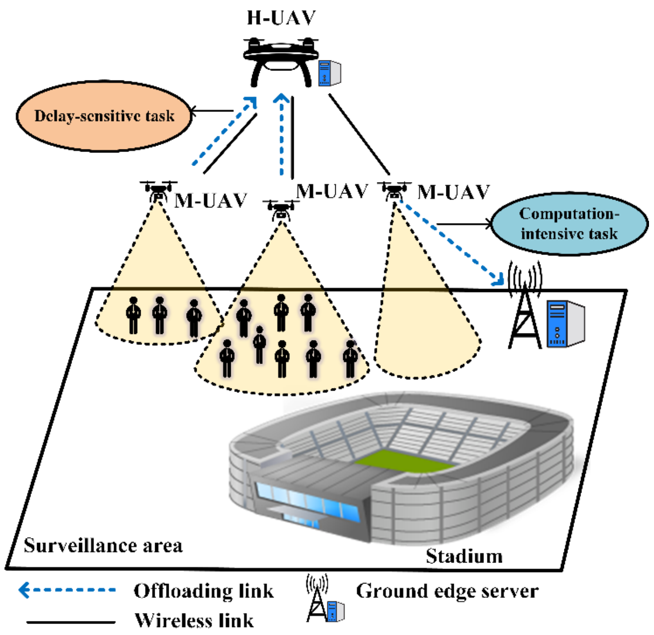


Figure 1. Practical use case of unmanned aerial vehicle (UAV) swarm-enabled mobile edge computing system for surveillance applications.

owing to the additional arrangements conducted during large sports events [47]. We assume that UAVs are divided into N coalitions, i.e., $\{Q_1, Q_2, \ldots, Q_N\}$. Each coalition consists of one coalition head and several coalition members. Each UAV in the coalition is equipped with a computing unit to perform computationally intensive tasks. Here, we consider that coalition heads have a high computing performance compared with M-UAVs. The M-UAVs function as sensing UAVs by sensing a particular area and capturing videos to process them using face recognition algorithms.

When the computing resources are exhausted and computing tasks are prolonged in the M-UAVs, the task can be either offloaded to the H-UAV or to the ground edge server to assist the M-UAVs depending on the task characteristics. The tasks are classified into two types: computationally intensive or delay tolerant task and delay sensitive task. If the task is delay sensitive and must be computed before a certain time period, the task is offloaded to the H-UAV to avoid transmission delay. Otherwise, if the task is delay tolerant and requires high computational power and more energy, the task is offloaded from the M-UAVs to the ground edge server via a wireless link that has a sufficient computing capacity. Our objective is to minimize the overall computational energy and transmission latency involved in this computation offloading scenario. We assume that the topology of the UAV has been optimized according to the task requirements; therefore, the M-UAVs remain in a quasi-static scenario during offloading the task [48].

We consider that the required energy of all UAVs is sufficient to perform the mission and wireless communication during a flight period of J, which is divided into T time slots equally. We denote the set of time slot as $\mathcal{F} = \{1, 2, \cdots, t, \cdots, T\}$. The M-UAVs and H-UAVs are considered to fly at different altitudes to avoid collisions. We represent the coalition members and coalition head as $\mathcal{M} = \{1,2, \cdots, M\}$ and $\mathcal{H} = \{1, 2, \cdots, H\}$, respectively, where M and H represents the number of M-UAVs and H-UAVs in a coalition Q_N.
We denote the horizontal coordinates of M-UAV j as $w_j^M = (x_j, y_j, h)$, where $j \in \mathcal{M}$. The H-UAV flies through a predefined trajectory to minimize the transmission delay. The horizontal coordinate of the H-UAV in time slot t is denoted by $w_t^H = (X_t, Y_t, H)$.

3.3. Communication Model

We consider that the communication channels between the M-UAVs and H-UAV are characterized by line-of-sight communication, wherein the channel quality heavily relies on the communication distance [49][50]. The distance between the H-UAV and M-UAV j in time slot t is denoted as

$$d_{j,t} = \sqrt{\left\|w_n^H - w_{j,t}^M\right\|^2}, \forall t \in \mathcal{F}, j \in \mathcal{M}. \tag{1}$$

We assume that the channel gain between the H-UAV and M-UAVs in coalition Q_N follows the free-space path loss model as follows:

$$h_{j,t} = \eta d_{j,t}^{-2} = \frac{\eta}{\left\|w_n^H - w_{j,t}^M\right\|^2}, \forall t \in \mathcal{F}, j \in \mathcal{M}, \tag{2}$$

where η denotes the channel gain, which is located at 1 m and relies on the antenna gain and carrier frequency.

We consider that between the time intervals, all the M-UAVs are served by the H-UAV by following frequency division multiple access (FDMA) [51]. The total bandwidth of system B is partitioned into M sub-bands without overlapping. In each time slot, each M-UAV is allocated $\frac{B}{M}$ subbands. Following this, as shown in [52], the signal-to-noise ratio is derived as follows:

$$\begin{aligned} SNR_{j,t} &= \frac{h_{j,t} P_{j,t}}{\wp B/M} \\ &= \frac{\eta P_{j,t}}{\dfrac{\wp B}{M\left\|\mathrm{w}_t^H - \mathrm{w}_{i,t}^M\right\|^2}}, \forall t \in \mathcal{F}, j \in \mathcal{M}, \end{aligned} \tag{3}$$

where $P_{j,t}$ denotes the transmit power of the j^{th} M-UAV, and $\wp$ indicates the spectrum density of the white Gaussian noise (WGN) in W/Hz at the H-UAV. Following Shannon's formula, the uplink data rate of M-UAV j in time slot t is given by

$$R_{j,t} = \frac{B}{M}\log_2\left(1 + \frac{\eta P_{j,t}}{\wp B/M\left\|\mathrm{w}_t^H - \mathrm{w}_{j,t}^M\right\|^2}\right), \forall t \in \mathcal{F}, j \in \mathcal{M}. \tag{4}$$

Similarly, when the j^{th} M-UAV offloads the task to the ground edge server, we assume that the location of the edge server is fixed, and the horizontal coordinate of the edge server is denoted as $w_{\mathrm{t}}^{EC} = (x_{EC}, y_{EC}, 0)$. Subsequently, the distance between the j^{th} M-UAV and edge server in time slot t is defined as

$$d_{EC,t} = \left\|\mathrm{w}_{j,t}^M - w_{\mathrm{t}}^{\mathrm{EC}}\right\|. \tag{5}$$

Because the UAV offloads computationally intensive tasks to the ground edge server, the channel gain between the j^{th} M-UAV and edge server in time slot t is denoted by

$$h_{EC,t} = \frac{g}{[d_{EC,t}]^2} \tag{6}$$

where g denotes the power gain and the reference distance is considered to be 1 meter. Therefore, the transmission data rate between the j^{th} M-UAV and edge server in time slot t is defined as

$$R_{EC,t} = B_u log_2\left(1 + \frac{h_{EC,t}\, P_{j,t}}{\mu^2}\right), \forall t \in \mathcal{F}, j \in \mathcal{M} \tag{7}$$

Where B_u is the bandwidth pre-assigned to the edge server and μ is the noise power.

3.4. Task Computation Model

In this paper, we assume that each M-UAV j in a coalition Q_N has a sensing task that can be computed locally in the j^{th} M-UAV or either in the H-UAV or the ground edge server that is co-located with the BS at time slot t, where $t \in \mathcal{F}$. Let a_j^1 define the computation offloading decision of the M-UAV, where $a_j^1 = 1$ indicates that the M-UAV offloads the task, whereas $a_j^1 = 0$ indicates that the task is computed locally. We define each task as $K_j^t = (L_j^t, d_j^t)$, where L_j^t indicates the CPU cycles required to perform task of the j^{th} M-UAV, and d_j^t indicates the data size that must be computed at time slot t. Next, we derive the computation cost in terms of the task delay as well as energy consumption for local and edge computing.

1) Local computing: We denote the computation capability, i.e., the clock frequency of the CPU chip, of M-UAV j for task J_j^t as $f_{j,K}$. The local execution time of task K on M-UAV j is given by

$$T^{loc,exe} = \frac{L_j^t}{f_{j,K}} \tag{8}$$

while the energy consumption of M-UAV j for executing task K_j^t is given by

$$E_{j,m}^{loc,exe} = kL_j^t f_{j,K}^2 \tag{9}$$

where k indicates the switched capacitance of a specific chip architecture of the device. In accordance with previous studies, we consider that $k = 10^{-28}$ [53]. Consequently, the total cost for executing task K_j^t locally is defined by the sum of the local execution time and energy consumption during execution. That is,

$$U_{j,K}^{local} = \alpha_1^l T^{loc,exe} + \beta_2^l E_{j,m}^{loc,exe}, \tag{10}$$

where ϑ_1 and ϑ_2 are the weights that control the importance of the latency and energy consumption.

2) Task offloading: With an increasing number of tasks, the computation capacity of the M-UAV is exhausted owing to the shortage of computing resources. Thus, the tasks generated after a certain time are continuously rejected. Hence, whether an M-UAV should offload its tasks has a significant impact on the performance of the overall network. In our paper, we consider that the M-UAV can offload the task to either the H-UAV or the ground edge server, depending on the task characteristics. As discussed earlier, we consider that tasks can be classified into two different categories: delay sensitive and energy sensitive tasks. The intuition for such a consideration is that deep-learning-based image recognition techniques involve many phases, such as noise removal, pre-processing, resizing, training, and classification. Thus, the tasks are of various types, with varying sizes and computational requirements. Therefore, for any task K_j^t, $a_j^2 \in \{0, 1\}$ represents the action that the j^{th} M-UAV can perform. When the task is delay-sensitive, it is offloaded to the H-UAV, i.e., $a_j^2 = 1$; hence, the total delay consists of the transmission and computation delays at the H-UAV.

Thus, the transmission delay for offloading the task to the H-UAV is given by

$$T_{K_j^t}^{tx} = \frac{d_j^t}{R_{j,t}}. \tag{11}$$

Because the H-UAV has a higher computational capacity than the M-UAV, each H-UAV can execute the offloaded task locally using the computing unit onboard. Here, we assume that the H-UAV can run several virtual machines to compute tasks from different M-UAVs [54][55]. Thus, we consider tasks from different M-UAVs are executed simultaneously. Hence, we ignore the computation capacity allocation in this paper [48][17]. Therefore, when the H-UAV functions as a server, the computation delay of the H-UAV is given by

$$T^{H,exe} = \frac{L_j^t}{f_{h,K}} \tag{12}$$

where $f_{h,K}$ denotes the clock frequency of the H-UAV on task K. Meanwhile, the energy consumption during task execution can be calculated by

$$E^{H,exe} = k_H L_j^t f_{h,K}^2 \tag{13}$$

where k_H represents the effective switched capacitance of the H-UAV related to the chip architecture; we set $k_H = 10^{-28}$ [53]. The total completion time of the task K_j^t is defined by the sum of transmission delay from M-UAV j to the H-UAV and the execution delay at the H-UAV.

$$T_{j,K}^{H-UAV} = T^{H,exe} + T_{K_j^t}^{tra}. \tag{14}$$

We are now ready to define the total cost for executing the task on H-UAV, which is given by

$$U_{j,K}^{H-UAV} = \alpha_1^h T_{j,K}^{H-UAV} + \beta_2^h E^{H,exe}. \tag{15}$$

However, because deep learning techniques often require extensive computation (e.g., matching face images from existing datasets), tasks can be computationally intensive and require more time, executing such tasks at the H-UAV may degrade the overall network performance. Thus, we consider that the M-UAV can also offload the task to the edge server for edge execution with a strong computation capacity, i.e., $a_j^2 = 0$. In this case, execution occurs in three phases: (i) Task transmission stage, (ii) edge computing stage, and (iii) result transmission stage.

In the first phase, we derive the total cost by considering the energy consumption during the transmission and execution of the task. The transmission time and energy consumption of the task K_j^t at the edge server are given by

$$T_{K_j^t}^{edge,tx} = \frac{d_l^t}{R_{EC,t}}. \tag{16}$$

and

$$E_{K_j^t}^{edge,tx} = P_{j,t}\, T_{K_j^t}^{edge,tx}. \tag{17}$$

For edge execution, the computing task execution time is denoted by

$$T^{edge,exe} = \frac{L_j^t}{f_e}, \tag{18}$$

where f_e denotes the CPU frequency of the ground edge server.

We assume that the frequency remains constant during task execution. Owing to the high computational capacity and sufficient power of the ground edge server, the edge server can easily complete the offloaded task. Corresponding with other studies, e.g., [56] and [57], we omit the result receiving delay because the size of the returned data is exceedingly small. We also ignore the edge energy consumption in the total offloading cost calculation because the energy consumption of the ground edge server is negligible. Thus, the total duration for completing the execution of the task K_j^t in the edge server is given by

$$T_{j,K}^{edge} = T_{J_l^t}^{edge,tx} + T^{edge,exe}. \tag{19}$$

The total cost of edge execution is given by

$$U_{j,K}^{edge} = \alpha_1^e T_{j,K}^{edge} + \beta_2^e E_{J_l^t}^{edge,tx}. \tag{20}$$

Because of the computation capacity among the three computation nodes, based on where the task is being executed, we introduce different weights to enable diversity in the delay and the energy consumption calculation of the three cases. This means α_1^l is different from α_1^h and α_1^e. The same applies for β_2^l, β_2^h, and β_2^e.

3.5. Problem Formulation

In this thesis, our aim is to minimize the normalized weighted cost by considering both task execution delay and energy consumption. When all tasks are locally computed, an M-UAV cannot complete all tasks owing to its limited energy and computational capacity. In addition, latency is another crucial metric in an edge-computing environment that can significantly deteriorate performance. Thus, an optimal task allocation strategy (M-UAV, edge server) is required by considering execution time and energy consumption. Therefore, we jointly consider the energy consumption and execution delay during transmission and computing. Because we have a multi-objective optimization problem, we consider a popular multi-objective optimization method: the linear weighted sum method [51][58]. This method combines multiple objectives into a single objective function.

First, the execution time and energy consumption are normalized such that they are within the same range. To normalize, we divide the task execution delay and energy consumption by the delay and energy calculated from local computing. Since energy consumption and delay have different units, we have normalized execution delay and energy consumption to a range of the same size by dividing with the maximum energy and maximum execution delay that is obtained from local computing. Thus, we convert the two different metrics into the same number range. Subsequently, we apply different weights for both energy consumption and delay. These weights enable us to configure the video analysis based on task requirements. Based on the above discussion, the objective function for a sequence of tasks $\mathcal{R}$ can be formulated as follows:

$$U_j = \sum_{K=1}^{\mathcal{B}} U_{j,K} = \sum_{i=0}^{I} a_j^1 a_j^2 U_{\mathrm{j,K}}^{H-UAV} + \overset{1}{a_j}\left(1 - a_j^2\right) U_{j,K}^{edge} + \left(1 - a_j^1\right) U_{\mathrm{j,K}}^{local}\Big), \qquad (21)$$

where $\mathcal{B}$ indicates the total size of set $\mathcal{R}$. Thus, the optimization problem can be formally derived as

$$\min_{A} \sum_j U_j. \qquad (22)$$

s.t $\quad a_j^1 a_j^2 T_{j,K}^{H-UAV} + a_j^1\left(1 - a_j^2\right) T_{j,K}^{edge} + \left(1 - a_j^1\right) T^{loc,exec} \leq T_K^{max}, \forall K = 1, \cdots, \mathcal{B} \qquad (23)$

where A = $\{ a_j^1, a_j^2 \,|\, j \in \mathcal{M}, K \in \mathcal{R}\}$. This constraint states that all tasks must be completed by the total completion time, T_K^{max}. Because we use integer constraints (e.g., $a_j^1 a_j^2$), this is a non-convex problem, and using traditional optimization techniques is not useful owing to dynamic task requirements. Thus, we design a multi-agent reinforcement learning algorithm that can provide a simpler and more efficient solution in less time.

3.6. RL Framework

In traditional RL, the problem is modeled as an MDP. In this section, we first discuss why the offloading problem can be solved using DRL. Subsequently, according to the RL framework, we provide the definitions of the state space, action space, and reward function of our stated problem. The state space is defined by the metrics used to determine the optimal action from all available actions in the action space. The ultimate objective of action selection is to maximize the reward function, which is designed based on the objective of our study. In DRL, neural networks are used to simulate the optimization function to obtain the optimal result by training the network parameters. Currently, conventional optimization methods (e.g., heuristic methods and convex optimization) are widely used to determine the optimal offloading decision in MEC environments. However, with an increase in optimization variables and slightly complicated constraints, these algorithms cannot obtain optimal results. For example, in convex optimization-based techniques, the first step is to divide the global optimization problem into subproblems. To apply heuristic-based solutions, we must simplify the problem sufficiently, and the MDP requires input data in a state transition matrix. In DRL, a neural network enables DRL algorithms to obtain near-optimal results when the state space becomes large and complex. Thus, DRL overcomes the limitations of traditional optimization techniques in computation offloading decision-making problems.

1) State space s_j^t: To make the offloading decision, each agent in the network is provided with a set of input metrics that they consider while making the offloading decision. Let $s_j^t = \{D, c, f, \text{and } dt\}$ denote the state space. The meanings of these symbols are provided by

- D: size of the task
- C: cycles needed to complete the task
- f: computational capability of the H-UAV
- dt: task type $\in \{0,1\}$ (energy-sensitive or delay-sensitive)

2) Action space a_j^t: Each M-UAV selects a particular action after the tasks are generated in time slot $t \in \mathcal{F}$ in the coalition. The M-UAV can select an action from local information. In this paper, we consider binary offloading, which can either be executed at the H-UAV or offloaded to the ground edge server.

3) Reward function r_j^t: After selecting a certain action, the agent gains a reward that reflects the performance of the selected action by reinforcing the action performed by the agent. Designing a good reward function is crucial for network performance, because it can reduce the convergence time of the algorithm. In our UAV swarm enabled MEC system, the main objective is to complete the execution of the task with minimum task execution delay and energy consumption within the task deadline. According to the above discussion, the reward function is designed as follows:

The aim of each agent is to maximize the total reward. Because action selection depends on the unique characteristics of the network dynamics in which the agent is interacting, defining the reward function directly from the obtained utility would affect the learning process. Thus, we consider the difference between the utility value and the value obtained in the previous time step. The intuition for such a design is that a positive difference indicates an enhancement in the obtained reward value and, thus, that action should be emphasized. We assume that each agent has the same reward function. Based on the discussion above, the reward function is defined as follows:

$$Z_{j,K} = -a_j^2 a_j^1 U_{\mathrm{j,K}}^{\mathrm{H-UAV}} - a_j^1\left(1 - a_j^2\right) U_{j,K}^{edge} - (1 - a_j^1) U_{\mathrm{j,K}}^{\mathrm{local}}, \tag{24}$$

where a higher cost results in a smaller reward, and vice versa. Thus, for each agent, the utility is

$$Z_j = \sum_{K=1}^{\mathcal{B}} Z_{j,K}. \tag{25}$$

Each agent aims to maximize this objective, which emphasizes the action that produces a better reward value. Because of the dynamic characteristics of the network in which the agent is interacting, the action evaluation can be ambiguous. Thus, the value obtained here cannot be directly utilized to define the reward [59]. Therefore, the difference between reward values in immediate timesteps are considered the reward in this paper. The intuition for such a consideration is that an increase in the reward denotes an improvement, such that a particular action should be emphasized. Based on the discussion above, the reward value is defined as follows:

$$u_t^j = \begin{cases} p, \ if \ Z_j^t - Z_j^{t-1} < 0 \\ q, \ if \ Z_j^t - Z_j^{t-1} > 0 \\ 0, \quad otherwise \end{cases}, \tag{26}$$

where Z_j^t is the cumulative reward gained at timeslot t. When the reward difference between two immediate timesteps is negative, it is considered an improvement because the reward is formulated as a negative value; thus, p is a positive value, whereas a positive difference between the rewards obtained from immediate timesteps yields q which is a negative reward. For simplicity, we consider that each agent shares the same reward.

4. Deep Reinforcement Learning based Computation Offloading Algorithm (DRLCO)

Based on the previous discussion, we designed a deep reinforcement learning based computation offloading (DRLCO) algorithm to solve this decision-making problem as shown in Figure 2. In each time slot, when new tasks are generated to be computed locally or on the MEC server, each agent (M-UAV) acts on the environment by observing the state and selecting a specific action. Subsequently, each agent receives an immediate reward. During the entire time, each agent aims to determine an optimal offloading decision to maximize the cumulative reward. Owing to the integer constraint to determine on which device the task will be processed, conventional optimization methods fail to obtain an optimal solution in such scenarios. In addition, deep learning techniques occasionally fail to map the relationship between input and output efficiently and require a longer training time in new environments [60].

To address these challenges, we propose a DRLCO algorithm that incorporates a neural network and Q-learning to obtain an optimal offloading decision. By combining the perceptive characteristics of the neural network and decision-making capability of RL, we can determine the optimal offloading decision in a dynamic environment. Because the tasks to be computed are not known beforehand, the DRLCO algorithm was designed to fall under the category of model-free RL. Algorithm 1 presents the proposed DRLCO algorithm.

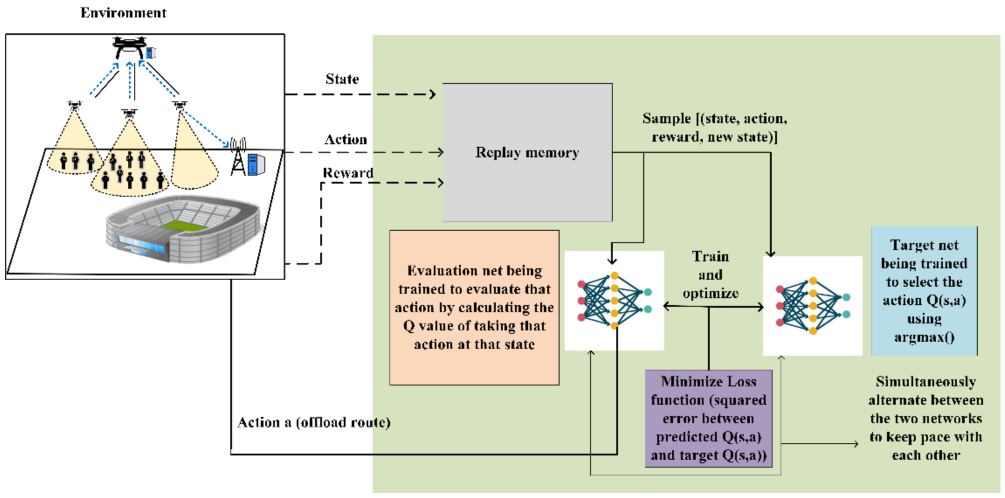

Environment
State
Action
Reward
Replay memory
Sample [(state, action, reward, new state)]
Evaluation net being trained to evaluate that action by calculating the Q value of taking that action at that state
Train and optimize
Target net being trained to select the action Q(s,a) using argmax()
Minimize Loss function (squared error between predicted Q(s,a) and target Q(s,a))
Simultaneously alternate between the two networks to keep pace with each other
Action a (offload route)

Figure 2. Block diagram of the proposed DRLCO scheme.

Two identical neural networks, whose Q functions are denoted as $Q_j^t(s_j^t, a_j^t|\varphi_j^e)$ and $Q_j^e(s_j^t, a_j^t|\varphi_j^e)$, are used to construct the overall structure of the DRLCO, which we call the target network φ_j^t and evaluation network φ_j^e. To enhance the training efficiency and early convergence, we use the experience replay $\chi_j(t)$ to store past samples. The sample of experiences is defined as $\{s_j^t, a_j^t, u_t^j, \text{and } s'^t_j\}$. The intuition for using experience replay memory is that consecutive samples are highly correlated, which may affect the learning process and result in sample inefficiency. To break this correlation, we use a mini batch of random samples to train the model, which is stored in the replay memory. As memory fills up, old experiences are removed to create space for newer ones. Each agent (M-UAV) leverages replay memory to determine the optimal mapping between the state and action. We consider that the agent selects a certain action a_j^t using the following ϵ-greedy policy:

$$a_j^t = \begin{Bmatrix} random, & \epsilon \\ arg\ \max\ Q_j^e(s_j^t, a|\varphi_j^e), & 1-\epsilon \end{Bmatrix}. \tag{27}$$

In DRLCO, two neural networks aim to determine the optimal action, as stated previously. In a typical DQN, the max operator is used to select and evaluate an action that results in the overestimation of values, causing overoptimistic value estimates [61]. Therefore, to address this challenge, we use two identical networks to separate the task of selecting and evaluating an action, which enables the evaluation of the greedy action taken. Based on this concept in DRLCO, the action that produces the highest Q-value is first selected by the target network. Subsequently, the evaluation network evaluates the selected action by calculating the Q-value of taking that action in that state. Thus, the value of the policy is evaluated evenly using the two neural networks. The expected cumulative reward of the target network can be derived as

$$Rew_j = u_t^j + \mu_k Q_j^t(s'^t_j, \arg\max Q_j^e(s'^t_j, a_j^t|\varphi_j^e)|\varphi_j^t). \tag{28}$$

where μ_k denotes the discount factor for controlling further rewards. Therefore, the loss between $Q_j^t(s_j^t, a_j^t|\varphi_j^e)$ and $Q_j^e(s_j^t, a_j^t|\varphi_j^e)$ is calculated as follows:

$$Loss_j(Q_j^e, Q_j^t) = \mathrm{E}_{(s_j^t, a_j^t, u_t^j, s'^t_j)\sim\chi_j}[Rew_j - Q_j^e(s'^t_j, a_j^t|\varphi_j^e))]^2. \tag{29}$$

The two networks are alternated simultaneously to ensure stability in the training performance. The target network parameters are updated using the evaluation network until convergence. Thus, each agent can learn the optimal offloading policy in a distributed manner according to its own information. The DRLCO algorithm, as provided in Algorithm 1, has two major parts:

Algorithm 1 The algorithm is run in the M-UAV

```
Input: Task feature $\{D, C, f, dt\}$
Output: Optimal offloading location for a given input
1: Initialize parameters of target and evaluation networks for all M-UAV $\in \mathcal{M}$;
2: Initialize replay memory $\chi_j$ for each agent M-UAV $\in \mathcal{M}$;
3:   for episode = 1 to $episode^{max}$ do
4:    Reset entire environment for each M-UAV $\in \mathcal{M}$;
5:    for $j$ = 1 to $M$ do
6:       M-UAV acts on the dynamic environment;
7:         for t = 1 to $T$ do
8:               M-UAV observes the state $s_j^t$ parameters consisting of task size $D$, required cycles to execute the task $C$, computational capability of the H-UAV $f$, task type $dt$
9:               M-UAV selects action $a_j^t$ regarding offloading the task to the H-UAV or the ground edge server following $\epsilon$-greedy policy;
10:              M-UAV obtains the reward $u_t^j$, next state $s'^t_j$;
11:              Add sample $\{s_j^t, a_j^t, u_t^j, s'^t_j\}$ into replay memory $\chi_j$ when replay memory is not full.
                 if samples are sufficient in $\chi_j$, do
10:                  Select a mini batch from replay memory $\chi_j$;
11:                  Calculate cumulative reward using (28);
12:                  Calculate loss using (29);
13:                  Update the evaluation network
14:            Alternate the parameters from evaluation network to target network $(\boldsymbol{\varphi}_j^t \leftarrow \boldsymbol{\varphi}_j^e)$
15:              end if
16:        end for
15:    end for
16:  end for
17: return offloading location for given input
```

collecting data samples from the network environment and training based on the collected data. The parameters of the target network and parameters with initial weights are defined initially along with the replay memory size (lines

1−2). The number of episodes is then defined, and agent begins interacting with the network environment to collect data (lines 3−11). In this data-collection phase, each agent (M-UAV) observes the state, selects an action, receives the reward, and receives a new state (lines 8−10). The replay memory stores the experience gained. Subsequently, in the training phase, a random mini batch from the replay memory is sampled to train the agent and calculate the loss (lines 12−14).

4.1. Complexity Analysis

In this section, we study the proposed DRLCO scheme in terms of its computational complexity. Each M-UAV j, where $j \in \mathcal{M}$, has three types of neural networks: input layer, fully connected (FC) layers, and output layer. Each agent has a total of $V + 2$ layers with one input layer, one output layer, and V FC layers. We denote the total amount of training sample as H and the total number of epochs as F. i_q and i_n represents the input layer dimension and neuron number in layer q, respectively, where $q \geq 2$ and $q \in V$. Hence, the time complexity of the proposed DRLCO is $O(H * F * i_q * (V - 1)^{i_n})$. Because there is only one output layer, and $O(i_q)$ is the complexity of the total number of activation functions, the complexity of these two metrics has been ignored with regard to the overall complexity.

5. Performance Evaluation

In this section, we evaluate the performance of the proposed DRLCO scheme and compare it with that of conventional schemes through a simulation study.

5.1. Experimental Setup

We verified the correctness of our proposed DRLCO using Python and TensorFlow 1.15 on a computer with an AMD Ryzen 5 processor with a processor speed of 3.60 GHz and RAM of 8 GB. We considered five H-UAVs and 15 M-UAVs in our UAV swarm-enabled edge-computing scenario in an area of 1000 m × 1000 m. To simulate the DRLCO algorithm, we set that for each M-UAV, the neural network of the DRLCO algorithm consisted of one input layer and output layer with two FC layers. The sizes of the two hidden layers were 400 and 350. We used ReLU in the hidden layers, as the activation function and AdamOptimizer to optimize the loss function. For training, we set the maximum number of iterations to 2000. The size of the replay memory was 50000. To emphasize future rewards, we set the discount factor value to 0.9. Because $\alpha = 0.001$ yielded a higher reward and stable training, in the simulation environment, we set the value of the learning rate as $\alpha = 0.001$. The weights of energy consumption and task execution delay were set as $\alpha_1^l = \alpha_1^h = \alpha_1^e = \beta_2^l = \beta_2^h = \beta_2^e = 0.5$ to obtain equivalent importance on evaluating the total cost. The computational task size varied randomly between 2 to 20 MB. The computational capacities of the M-UAV, H-UAV, and ground edge server were 1.5, 15, and 20 GHz, respectively. The major simulation parameters are listed in Table 2.

Table 2. Simulation parameters.

Parameter	**Value**
Number of H-UAVs	5
Number of M-UAVs	15
Bandwidth (B)	20 MHz
Bandwidth at the edge server (B_u)	0.5 MHz
Transmission power ($P_{j,t}$)	20 dBm
Channel power gain (η)	1.42×10^{-4}
Power spectrum density ($\wp$)	-174 dBm
Path loss exponent ($\mathcal{g}$)	-50 dB
Noise power (μ)	-100 dB
Computation task size of each M-UAV	2 – 20 Mb
Required CPU to complete input task	0 – 1.5 GHz
Computation capacity of M-UAV ($f_{j,K}$)	1.5 GHz
Computation capacity of H-UAV ($f_{h,K}$)	15 GHz
Computation capacity of ground edge server (f_e)	20 GHz
Effective switched capacitance of M-UAV and H-UAV (k, k_H)	10^{-28}
Weights ($\alpha_1^l = \alpha_1^h = \alpha_1^e = \beta_2^l = \beta_2^h = \beta_2^e$)	0.5
1st hidden layer size	400
2nd hidden layer size	rand (3, 5)
Learning rate (α)	1×10^{-3}
Mini batch size	100
Experience replay size (χ_j)	50000
Discount factor	0.9

Total episode ($episode^{max}$)	2000
Length of each slot	1 s

5.2. Convergence Analysis

To show the convergence of the DRLCO algorithm, we compared the DRLCO scheme with the DQN algorithm as shown in Figure 3. We observed that both the DRLCO and DQN techniques converge after a certain time, and the reward is stable. This means that agents can obtain an optimal offloading policy. Furthermore, we observed that DRLCO reaches a steady state earlier than DQN. Initially, because the M-UAV has no knowledge of the system environment, the two techniques fluctuate at a very low value because of the agent's random action selection. Gradually, the M-UAV acts on the environment, begins to collect samples, and trains the network when the experience replay memory has sufficient samples. As mentioned earlier, because the DRLCO scheme solves the overestimation problem of Q-values by separating the estimation into two networks, it can learn valuable states without observing the impact of each action at every state. Thus, we observed that the reward of the DRLCO scheme is significantly higher than that of the DQN scheme. In contrast, the DQN algorithm consists of a single neural network, that is, a value function network that utilizes a random policy for training. Thus, we observed that the reward gained in DRLCO scheme is significantly higher than that of the DQN scheme and it requires almost 1000 episodes to converge for the DQN scheme. However, our DRLCO scheme requires approximately only 250 episodes to reach a stable state and it provides a higher reward compared with the DQN scheme because of the introduction of the target network in DRLCO, which aids in converging faster. Thus, the proposed method can save time and obtain an optimal offloading decision.

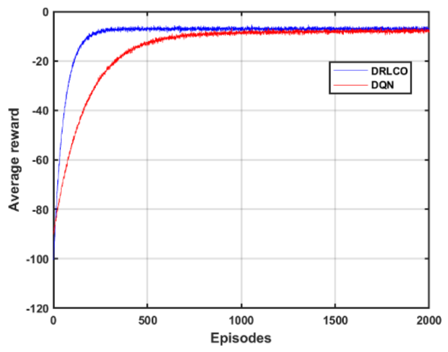


Figure 3. Convergence of DRLCO and DQN

5.2. Performance Metrics

After completing the training process of the proposed DRLCO, we verified our proposed scheme to validate our model by performing experiments on various performance metrics. Intuitively, the main objective of the proposed multi-agent computation offloading algorithm is to successfully execute the task with the lowest total cost. Because the M-UAV can execute the task locally or through offloading into the H-UAV or the ground edge server, the energy consumption as well as delay involved in the transmission and computation of the task were considered in our performance analysis along with the total offloading cost. This is because both energy consumption and delay are crucial offloading metrics, and determining the optimal offloading policy, which has less delay and consumes less energy, indicates the optimal offloading policy. Before quantitatively verifying the performance of the proposed DRLCO with other benchmarks, we briefly discuss the performance metrics below.

- Average offloading cost: We discuss the average offloading cost in terms of offloading task data size, computation capacity of the H-UAV, and number of M-UAVs. This is the average cost of all the agents (M-UAVs) for performing the task.

- Task execution delay: Delay is another crucial metric for determining the network performance and improving the quality of service of the system. Reducing the delay involved in the offloading and execution of a task can significantly reduce the total system overhead. Thus, to demonstrate the effectiveness of this study, we formulated task execution delay as the sum of both transmission and processing delays and compared it with conventional schemes.

- Energy consumption: In a UAV-enabled edge computing system, the total energy consumption for offloading a task is the most important metric because it has low computing energy to improve the offloading performance. Thus, inefficient task allocation may result in high overhead and tasks being dropped. The total energy includes both the energy required for the transmission and the execution of the task.

To validate the effectiveness of the proposed DRLCO scheme, we considered the following three conventional methods and compared the performance of our study with these methods.

- Local: In this setup, tasks were executed locally by each M-UAV in each time slot to the maximum computation capacity.

- Edge: All tasks were executed on a ground edge server, which had sufficient computing capacity.

- DQN: We applied the DQN algorithm in our proposed scenario as a benchmark technique because the action space is also discrete in the DQN.

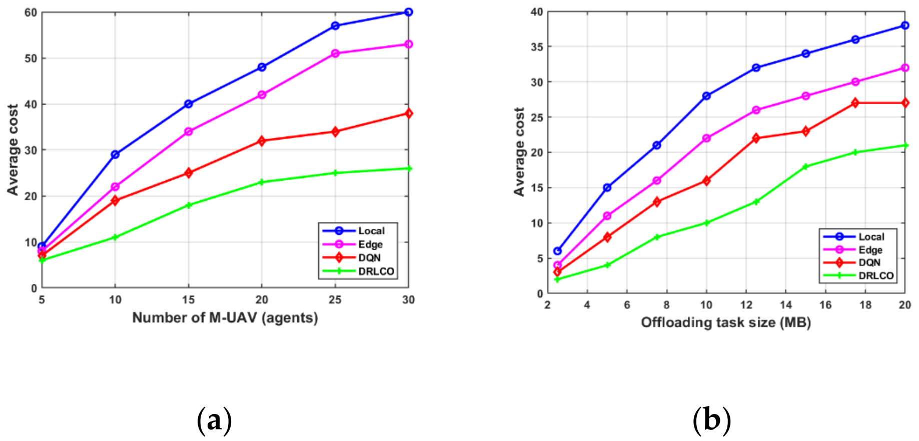
5.2. Simulation Results and Discussion

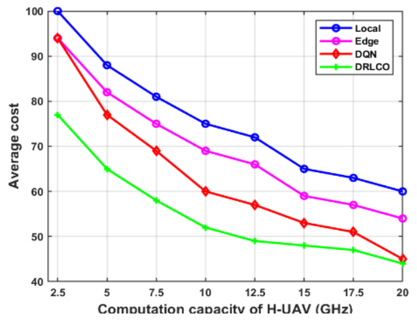
In this section, we present the simulation results of the proposed DRLCO algorithm and compare the with benchmark techniques. First, we analyze the average cost with respect to the number of M-UAVs, computation capacity of the H-UAV, and offloading task size.

Figure 4(a) shows the impact of increasing the number of M-UAVs on minimizing the average cost. We observed that with an increase in the number of agents (M-UAV), the average cost also increased gradually. This was because increasing the number of agents increased the number of sensing tasks. Thus, to manage a large number of tasks, the transmission and execution delays also increased. The DRLCO algorithm reduced the average cost by 55.13%, 48.08%, and 29.21% compared with the local, edge, and DQN, respectively. Thus, we can conclude that, with an increasing number of agents, the DRLCO scheme outperforms the three benchmarks in terms of the number of agents.

Figure 4(b) shows the impact of the computation capacity on the average cost. The figure shows that increasing the computational capacity of the H-UAV reduced the average cost by a significant margin. This was because, with an increase in the computation capacity, the H-UAV obtained adequate computing resources. Because of this, the agents considered offloading the computation-intensive tasks to the H-UAV as the capacity increased instead of offloading to the ground edge node to minimize the total cost. This reduced the transmission and processing delays of the task. The proposed DRLCO algorithm outperformed the three benchmark techniques and reduced the cost by 27.15%, 20.86%, and 15.70% compared with the local execution, edge execution, and DQN, respectively.

Figure 4(c) shows the impact of the offloading task size on the average cost. As the task size offloaded from the M-UAV increased, the total cost also increased. This was because the CPU cycles required to execute tasks with large sizes also increased for the H-UAV.





(c)

Figure 4. Average offloading cost in terms of (a) number of M-UAV, (b) offloading task size, and (c) computation capacity of H-UAV.

Hence, the computational time increased significantly. However, compared with other benchmarks, our proposed DRLCO scheme reduced the total cost by 54.28%, 43%, and 31% compared with local computing, edge computing, and the DQN approach, respectively. Therefore, the proposed DRLCO scheme can obtain a higher reward to minimize the total cost.

Next, we studied the two most crucial performance metrics in the network: energy consumption and task execution delay. Using the proposed DRLCO scheme, agents can dynamically select a suitable computational node to minimize the overall cost. To demonstrate the performance of our proposed DRLCO algorithm, Figure 5 shows the impact of the number of agents, computation capacity, and offloading task size on the energy consumption of the agent. As shown in Figure 5(a), the overall energy consumption increased when the number of agents (M-UAV) increased. This was because an increased number of M-UAVs generated more computational tasks to be executed. Thus, the computation node consumed additional power because the agents showed interest in offloading tasks on a suitable computation node to minimize the total cost. However, the local, edge, and DQN algorithms exhibited higher energy consumption than our proposed scheme. Because the proposed scheme dynamically allocated the computation task in terms of task characteristics, the task execution delay reduced by 55%, 48.08%, and 29% for the local, edge, and DQN approaches, respectively. In Figure 5(b) and 5(c), we analyze the impact of the computation capacity and task size on the energy consumption. With an increase in the computing capacity of the H-UAV, the task execution time decreased, and the energy consumption also increased shown in Figure 5(b). This meant that the increased computation capacity of the H-UAV enabled the M-UAV to offload energy-sensitive tasks more often than previously, which increased the node’s power consumption. The proposed DRLCO scheme reduced the energy consumption by 4.45%, 12.50%, and 21.81% compared with the DQN, edge, and local execution,

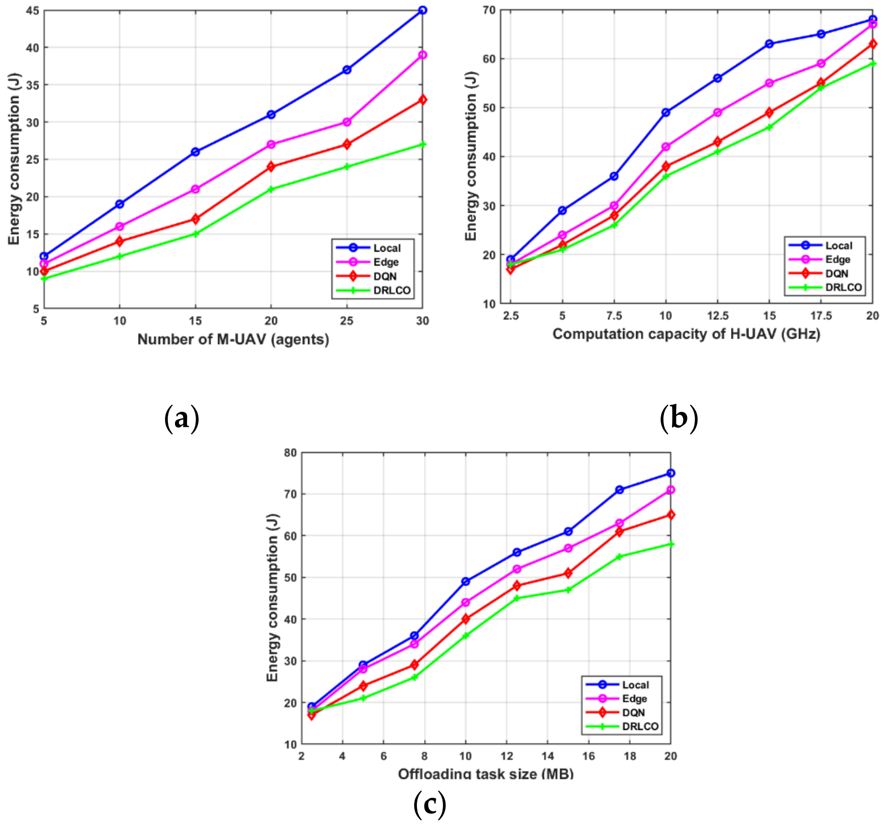


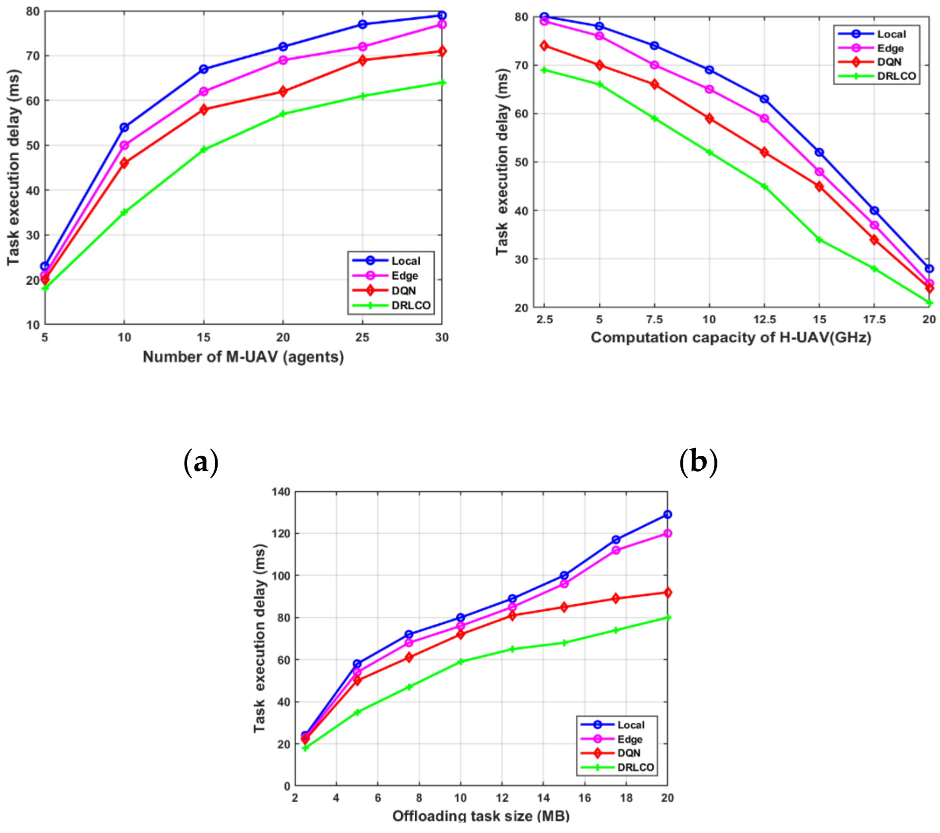
Figure 5. Energy consumption in terms of (a) number of M-UAV, (b) offloading task size, and (c) computation capacity of H-UAV.

respectively. Increasing the task size also increased energy consumption (Figure 5(c)). This indicated that a larger task requires more CPU cycles to complete, thereby increasing the power consumption of the computation node. Hence, the energy consumption also increased. However, the proposed algorithm significantly minimized the energy consumption by 11%, 22.27%, and 33.08% compared with DQN, edge, and cloud execution, respectively, owing to the dynamic allocation of the task according to the task characteristics.

Next, we focus on the delay performance of our proposed DRLCO scheme in terms of the number of agents (MUAV), task size, and varying computational capacity with the three other benchmark techniques. Figure 6(a) shows that increasing the number of agents increased the task execution delay. Because more M-UAVs generated tasks together, the computation node (H-UAV or edge) required more CPU cycles and computation time to complete the task. In addition, transmitting a task to the ground edge server incurred a transmission delay. However, the proposed DRLCO obtained a comparatively smaller delay than the other benchmarks. The proposed method reduced the delay by 23%, 19%, and 7% compared with the local, edge, and DQN approaches, respectively. Thus, the proposed DRLCO approach significantly outperformed the three benchmarks.

Figure 6(b) and Figure 6(c) depict the delay performance of the proposed DRLCO in terms of the computation capacity of the H-UAV and varying task size. As shown in Figure 6(b), we observed that increasing the computational capacity of the H-UAV reduced the task execution delay. Initially, when the computation capacity of the H-UAV was 2.5 GHz, the task execution delay was higher because the computation time was longer owing to the limited capacity. However, with an increase in the computation power, the computation time decreased further, and thus agents tended to offload more energy-sensitive tasks to the H-UAV, which reduced both the transmission and execution delays. The proposed DRLCO scheme reduced the task execution delay by 22%, 18.51%, and 11.79% compared with the local execution, edge execution, and DQN schemes, respectively.

Figure 6(c) depicts the impact of task size on task execution delay. We observed that increasing the task size also increased task execution delay.



(a) (b)

(c)

Figure 6. Task execution delay in terms of (a) number of M-UAV, (b) offloading task size, and (c) computation capacity of H-UAV.

Because a large task required more CPU cycles to execute, the computation time was longer; thus, the delay increased. In addition, when the computationally intensive task was offloaded from the M-UAV to the ground edge server, transmission and processing delays were incurred. However, the proposed DRLCO scheme can reduce the delay by allocating the task dynamically to either the H-UAV or ground edge server based on the task characteristics. The proposed DRLCO scheme reduced the task execution delay by 22.72%, 18%, and 11.79% compared with the local execution, edge execution, and DQN approaches, respectively. Therefore, the proposed

DRLCO is applicable to a multi-UAV-aided network system to reduce the total offloading cost by reducing energy consumption and task execution delay.

6. Conclusion and Future Works

In this paper, we investigate the decision-making problem of computation offloading in a UAV swarm-enabled edge-computing system. To support the UAV in successfully executing all tasks, the ground edge server provides assistance by enabling the UAV to offload computation-intensive tasks. Specifically, we formulate the offloading problem as a weighted sum cost minimization problem by jointly considering energy consumption and task execution delay. We then propose a multi-agent reinforcement learning framework called the DRLCO scheme to reduce the total system cost. Each M-UAV acts as an agent to determine the optimal offloading policy and performs offloading decisions based on the DRLCO scheme. Finally, simulation experiments were performed to validate the performance of the proposed DRLCO scheme. From the simulation results, we observed that the proposed technique can learn the optimal offloading policy and can significantly minimize the total cost, energy consumption, and task execution delay. Compared with the local execution, edge execution, and DQN techniques, the proposed method can reduce the total cost by 54.28%, 43%, and 31%, respectively, in terms of offloading task size.

In our future research, we will consider incorporating blockchain technology to ensure the privacy of the data and dependent task offloading in an edge computing scenario.

Bibliography

[1] X. Diao, W. Yang, L. Yang, and Y. Cai, “UAV-Relaying-Assisted Multi-access Edge Computing with Multi-antenna Base Station: Offloading and Scheduling Optimization,” *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–1, 2021, doi: 10.1109/tvt.2021.3101298.

[2] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, “Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks,” *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, 2014, doi: 10.1109/MSP.2014.2334709.

[3] I. Yaqoob, E. Ahmed, A. Gani, S. Mokhtar, M. Imran, and S. Guizani, “Mobile ad hoc cloud: A survey,” *Wirel. Commun. Mob. Comput.*, vol. 16, no. 16, pp. 2572–2589, 2016, doi: 10.1002/wcm.2709.

[4] X. Zheng, M. Li, M. Tahir, Y. Chen, and M. Alam, “Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing,” *IEEE Access*, vol. 7, no. 2, pp. 72247–72256, 2019, doi: 10.1109/ACCESS.2019.2919651.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” *MCC’12 - Proc. 1st ACM Mob. Cloud Comput. Work.*, pp. 13–15, 2012, doi: 10.1145/2342509.2342513.

[6] M. Y. Arafat and S. Moh, “A Survey on Cluster-Based Routing Protocols for Unmanned Aerial Vehicle Networks,” *IEEE Access*, vol. 7, pp. 498–516, 2019, doi: 10.1109/ACCESS.2018.2885539.

[7] A. Habib, Y. Arafat, and S. Moh, “Routing Protocols based on Reinforcement Learning for Wireless Sensor Networks: A Comparative Study,” *J. Adv. Res. Dyn. Control Syst.*, no. 14, pp. 427–435, 2018.

[8] M. Y. Arafat and S. Moh, “Routing protocols for unmanned aerial vehicle networks: A survey,” *IEEE Access*, vol. 7, pp. 99694–99720, 2019, doi: 10.1109/ACCESS.2019.2930813.

[9] M. Y. Arafat and S. Moh, “Localization and Clustering Based on Swarm Intelligence in UAV Networks for Emergency Communications,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8958–8976, 2019, doi: 10.1109/JIOT.2019.2925567.

[10] M. Y. Arafat, M. A. Habib, and S. Moh, “Routing protocols for UAV-aided wireless sensor networks,” *Appl. Sci.*, vol. 10, no. 12, pp. 1–23,

2020, doi: 10.3390/APP10124077.

[11] M. Y. Arafat, S. Poudel, and S. Moh, “Medium Access Control Protocols for Flying Ad Hoc Networks: A Review,” *IEEE Sens. J.*, vol. 21, no. 4, pp. 4097–4121, 2021, doi: 10.1109/JSEN.2020.3034600.

[12] M. Y. Arafat and S. Moh, “Bio-inspired approaches for energy-efficient localization and clustering in uav networks for monitoring wildfires in remote areas,” *IEEE Access*, vol. 9, pp. 18649–18669, 2021, doi: 10.1109/ACCESS.2021.3053605.

[13] M. Y. Arafat and S. Moh, “A Q-Learning-Based Topology-Aware Routing Protocol for Flying Ad Hoc Networks,” *IEEE Internet Things J.*, vol. 4662, no. c, pp. 1–1, 2021, doi: 10.1109/jiot.2021.3089759.

[14] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.

[15] M. Y. Arafat and S. Moh, “Location-Aided Delay Tolerant Routing Protocol in UAV Networks for Post-Disaster Operation,” *IEEE Access*, vol. 6, pp. 59891–59906, 2018, doi: 10.1109/ACCESS.2018.2875739.

[16] S. Poudel and S. Moh, “Medium Access Control Protocols for Unmanned Aerial Vehicle-Aided Wireless Sensor Networks: A Survey,” *IEEE Access*, vol. 7, no. July, pp. 65728–65744, 2019, doi: 10.1109/ACCESS.2019.2917948.

[17] X. Hu, K. K. Wong, K. Yang, and Z. Zheng, “UAV-Assisted Relaying and Edge Computing: Scheduling and Trajectory Optimization,” *IEEE Trans. Wirel. Commun.*, vol. 18, no. 10, pp. 4738–4752, 2019, doi: 10.1109/TWC.2019.2928539.

[18] M. Y. Arafat and S. Moh, “A Q-Learning-Based Topology-Aware Routing Protocol for Flying Ad Hoc Networks,” *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1985–2000, 2022, doi: 10.1109/JIOT.2021.3089759.

[19] Y. Chen, Y. Chen, H. Zhang, and Y. Hu, “Optimal power and bandwidth allocation for multiuser video streaming in UAV relay networks,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 6644–6655, 2020, doi: 10.1109/TVT.2020.2985061.

[20] Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. S. Shu, "Path Planning for UAV-Mounted Mobile Edge Computing with Deep Reinforcement Learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, 2020, doi: 10.1109/TVT.2020.2982508.

[21] W. You, C. Dong, X. Cheng, X. Zhu, Q. Wu, and G. Chen, "Joint Optimization of Area Coverage and Mobile-Edge Computing with Clustering for FANETs," *IEEE Internet Things J.*, vol. 8, no. 2, pp. 695–707, 2021, doi: 10.1109/JIOT.2020.3006891.

[22] M. M. Alam and S. Moh, "Joint topology control and routing in a UAV swarm for crowd surveillance," *J. Netw. Comput. Appl.*, p. 138954, 2022, doi: https://doi.org/10.1016/j.jnca.2022.103427.

[23] S. M. A. Huda and S. Moh, "Survey on computation offloading in UAV-Enabled mobile edge computing," *J. Netw. Comput. Appl.*, vol. 201, no. May 2022, p. 103341, 2022, doi: 10.1016/j.jnca.2022.103341.

[24] D. Callegaro and M. Levorato, "Optimal Edge Computing for Infrastructure-Assisted UAV Systems," *IEEE Trans. Veh. Technol.*, vol. 70, no. 2, pp. 1782–1792, 2021, doi: 10.1109/TVT.2021.3051378.

[25] B. Dai, J. Niu, T. Ren, Z. Hu, and M. Atiquzzaman, "Towards Energy-Efficient Scheduling of UAV and Base Station Hybrid Enabled Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–16, 2021, doi: 10.1109/TVT.2021.3129214.

[26] X. Diao, J. Zheng, Y. Cai, Y. Wu, and A. Anpalagan, "Fair Data Allocation and Trajectory Optimization for UAV-Assisted Mobile Edge Computing," *IEEE Commun. Lett.*, vol. 23, no. 12, pp. 2357–2361, 2019, doi: 10.1109/LCOMM.2019.2943461.

[27] Y. Liu *et al.*, "Joint Communication and Computation Resource Scheduling of a UAV-Assisted Mobile Edge Computing System for Platooning Vehicles," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–16, 2021, doi: 10.1109/TITS.2021.3082539.

[28] 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 Trans. Mob. Comput.*, vol. 1233, no. c, pp. 1–15, 2021, doi: 10.1109/TMC.2021.3059691.

[29] Y. Liu, J. Yan, and X. Zhao, "Deep Reinforcement Learning based Latency Minimization for Mobile Edge Computing with Virtualization

in Maritime UAV Communication Network," *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–1, 2022, doi: 10.1109/tvt.2022.3141799.

[30] F. Tang, H. Hofner, N. Kato, K. Kaneko, Y. Yamashita, and M. Hangai, "A Deep Reinforcement Learning-Based Dynamic Traffic Offloading in Space-Air-Ground Integrated Networks (SAGIN)," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 1, pp. 276–289, 2021, doi: 10.1109/jsac.2021.3126073.

[31] S. S. Yilmaz and B. Ozbek, "Multi-Helper NOMA for Cooperative Mobile Edge Computing," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–10, 2021, doi: 10.1109/TITS.2021.3116421.

[32] N. Nouri, F. Fazel, J. Abouei, and K. Plataniotis, "Multi-UAV Placement and User Association in Uplink MIMO Ultra-Dense Wireless Networks," *IEEE Trans. Mob. Comput.*, vol. 1233, no. AUGUST, pp. 1–18, 2021, doi: 10.1109/TMC.2021.3108960.

[33] N. C. Luong *et al.*, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019, doi: 10.1109/COMST.2019.2916583.

[34] M. G. R. Alam, M. M. Hassan, M. Zi. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications," *Futur. Gener. Comput. Syst.*, vol. 90, pp. 149–157, 2019, doi: 10.1016/j.future.2018.07.050.

[35] A. Asheralieva and D. Niyato, "Hierarchical Game-Theoretic and Reinforcement Learning Framework for Computational Offloading in UAV-Enabled Mobile Edge Computing Networks with Multiple Service Providers," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8753–8769, 2019, doi: 10.1109/JIOT.2019.2923702.

[36] Y. Liu, S. Xie, and Y. Zhang, "Cooperative Offloading and Resource Management for UAV-Enabled Mobile Edge Computing in Power IoT System," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12229–12239, 2020, doi: 10.1109/TVT.2020.3016840.

[37] X. Cheng *et al.*, "Space/Aerial-Assisted Computing Offloading for IoT Applications: A Learning-Based Approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019, doi:

10.1109/JSAC.2019.2906789.

[38] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, “Multi-UAV-Enabled Load-Balance Mobile-Edge Computing for IoT Networks,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, 2020, doi: 10.1109/JIOT.2020.2971645.

[39] G. Faraci, C. Grasso, and G. Schembra, “Design of a 5G Network Slice Extension with MEC UAVs Managed with Reinforcement Learning,” *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2356–2371, 2020, doi: 10.1109/JSAC.2020.3000416.

[40] H. Wang, H. Ke, and W. Sun, “Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning,” *IEEE Access*, vol. 8, pp. 180784–180798, 2020, doi: 10.1109/ACCESS.2020.3028553.

[41] H. Van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-Learning,” *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2094–2100, 2016.

[42] S. Zhu, L. Gui, D. Zhao, N. Cheng, Q. Zhang, and X. Lang, “Learning-Based Computation Offloading Approaches in UAVs-Assisted Edge Computing,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 1, pp. 928–944, 2021, doi: 10.1109/TVT.2020.3048938.

[43] L. Chen and X. Kuang, “Intelligent Mobile Edge Computing Networks for Internet of Things,” *IEEE Access*, vol. 9, pp. 95665–95674, 2021, doi: 10.1109/ACCESS.2021.3093886.

[44] B. Yang, X. Cao, C. Yuen, and L. Qian, “Offloading Optimization in Edge Computing for Deep-Learning-Enabled Target Tracking by Internet of UAVs,” *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9878–9893, 2021, doi: 10.1109/JIOT.2020.3016694.

[45] R. Chen *et al.*, “Joint Computation Offloading, Channel Access and Scheduling Optimization in UAV Swarms: A Game-theoretic Learning Approach,” *IEEE Open J. Comput. Soc.*, vol. PP, pp. 1–1, 2021, doi: 10.1109/ojcs.2021.3100870.

[46] E. I. S. Group, “GS MEC 003 - V2.2.1 - Multi-access Edge Computing (MEC); Framework and Reference Architecture,” vol. 1, pp. 1–21, 2020.

[47] G. Brown, “Computation offloading game for an UAV network in mobile edge computing,” *Juniper White Pap.*, no. July, 2016, [Online]. Available: https://www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf.

[48] J. Chen *et al.*, “A Multi-leader Multi-follower Stackelberg Game for Coalition-based UAV MEC Networks,” *IEEE Wirel. Commun. Lett.*, vol. 2337, no. c, pp. 1–1, 2021, doi: 10.1109/lwc.2021.3100113.

[49] T. Yang and X. S. Shen, “Multi-vessel computation offloading in maritime mobile edge computing network,” *SpringerBriefs Comput. Sci.*, vol. 6, no. 3, pp. 37–53, 2020, doi: 10.1007/978-981-15-4412-5_4.

[50] M. D. Nguyen, T. M. Ho, L. B. Le, and A. Girard, “UAV Trajectory and Sub-channel Assignment for UAV Based Wireless Networks,” *IEEE Wirel. Commun. Netw. Conf. WCNC*, vol. 2020-May, 2020, doi: 10.1109/WCNC45663.2020.9120814.

[51] Z. Yu, Y. Gong, S. Gong, and Y. Guo, “Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing,” *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3147–3159, Apr. 2020, doi: 10.1109/JIOT.2020.2965898.

[52] J. Zong *et al.*, “Flight time minimization via UAV’s trajectory design for ground sensor data collection,” *Proc. Int. Symp. Wirel. Commun. Syst.*, vol. 2019-Augus, pp. 255–259, 2019, doi: 10.1109/ISWCS.2019.8877250.

[53] Z. Zhou, Z. Chang, and H. Liao, “Dynamic Computation Offloading Scheme for Fog Computing System with Energy Harvesting Devices,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 143–161, 2021, doi: 10.1109/JSAC.2016.2611964.

[54] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, “Two-Tier Matching Game in Small Cell Networks for Mobile Edge Computing,” *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 254–265, 2022, doi: 10.1109/TSC.2019.2937777.

[55] T. Fang, J. Chen, and Y. Zhang, “Content-Aware Multi-Subtask Offloading: A Coalition Formation Game-Theoretic Approach,” *IEEE Trans. Serv. Comput.*, vol. 25, no. 8, pp. 2664–2668, 2021.

[56] D. Huang, P. Wang, and D. Niyato, “A dynamic offloading algorithm

for mobile computing," *IEEE Trans. Wirel. Commun.*, vol. 11, no. 6, pp. 1991–1995, 2012, doi: 10.1109/TWC.2012.041912.110912.

[57] A. Rudenko, P. Reiher, G. J. Popek, and G. H. Kuenning, "Saving portable computer battery power through remote process execution," *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 2, no. 1, pp. 19–26, 1998, doi: 10.1145/584007.584008.

[58] M. Ayoub Messous, H. Sedjelmaci, N. Houari, and S.-M. Senouci, "Computation offloading game for an UAV network in mobile edge computing," *2017 IEEE Int. Conf. Commun.*, 2014, [Online]. Available: https://doi.org/10.1109/ICC.2017.7996483.

[59] W. Li, F. Zhou, K. R. Chowdhury, and W. M. Meleis, "QTCP: Adaptive Congestion Control with Reinforcement Learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 6, no. 3, pp. 445–458, 2018, doi: 10.1109/TNSE.2018.2835758.

[60] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3448–3459, 2021, doi: 10.1109/TNSM.2021.3087258.

[61] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *30th AAAI Conf. Artif. Intell. AAAI 2016*, no. 2, pp. 2094–2100, 2016.

Acknowledgement

First and foremost, I would like to express my profound and sincere gratitude to my advisor, Prof. Sangman Moh. His invaluable support, encouragement, supervision, and useful suggestions throughout the master's studies have provided strong base for the completion of my thesis. I would always be indebted to him for his quintessential professionalism, persistent guidelines, and continuous encouragement through his impeccable mentorship.

Secondly, I want to express my warm and sincere thanks to the thesis committee members, Prof. Seokjoo Shin and Prof. Moonsoo Kang for their constructive comments and invigorating suggestions. All their insights regarding my research works have helped me in improving and extending it in different ways.

My sincere acknowledgement to the Department of Computer Engineering and Mobile Computing Lab for providing me such a wonderful opportunity and an atmosphere to grow me academically and otherwise. I must be thankful to all my lab members for their moral as well as academic support. I would like to heartily thank all my seniors and friends from society of Bangladesh in Chosun University for their affection and cooperation that made my life easier and cheerful in Korea.

At last but not the least, I cannot stop myself from thanking my parents and family members. Without their encouragement and love, it would have been impossible for me achieve anything. I would like to dedicate my work to them.