Energy-Efficient and Fresh Data Collection in IoT Networks by Machine Learning

A Thesis Submitted
to the College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in the Department of Electrical and Computer Engineering
University of Saskatchewan

by

Botao Zhu

Saskatoon, Saskatchewan, Canada

Permission to Use

Acknowledgments

I am very grateful for an opportunity of meeting many people during my studies who had helped me to make this dissertation possible. A few words mentioned here cannot adequately express my appreciation.

First, I would like to express my deepest gratitude to my supervisors, Prof. Ebrahim B. Mohamed and Prof. Ha H. Nguyen. Throughout my research program at the university, it is a great honor to receive invaluable support and extremely helpful guidance of Prof. Mohamed and Prof. Nguyen, without which, this dissertation would not have been completed.

Second, I would like to also thank the other members of the committee, Prof. Francis Bui, Prof. Xiaodong Liang, Prof. Brian Berscheid, and Prof. FangXiang Wu from the University of Saskatchewan and Prof. Lin Cai from the University of Victoria for reviewing and evaluating this thesis. Their insightful comments and suggestions have significantly improved the quality of this thesis.

Last but not least, I would like to thank my family for the support that they have provided me throughout my study. Without their love and encouragement, I would not have finished this thesis. My special thanks are extended to all my friends: Peter, Long, Ali, Khai, Nghia, Shania, Aiman, Mohammad, Mahendra, Om Jee, Alireza, Son, and Atefeh in Communications Theories Research Group (CTRG) for sharing their knowledge and invaluable assistance.

Abstract

The Internet-of-Things (IoT) is rapidly changing our lives in almost every field, such as smart agriculture, environmental monitoring, intelligent manufacturing system, etc. How to improve the efficiency of data collection in IoT networks has attracted increasing attention. Clustering-based algorithms are the most common methods used to improve the efficiency of data collection. They group devices into distinct clusters, where each device belongs to one cluster only. All member devices sense their surrounding environment and transmit the results to the cluster heads (CHs). The CHs then send the received data to a control center via single-hop or multi-hops transmission. Using unmanned aerial vehicles (UAVs) to collect data in IoT networks is another effective method for improving the efficiency of data collection. This is because UAVs can be flexibly deployed to communicate with ground devices via reliable air-to-ground communication links. Given that energy-efficient data collection and freshness of the collected data are two important factors in IoT networks, this thesis is concerned with designing algorithms to improve the energy efficiency of data collection and guarantee the freshness of the collected data.

Our first contribution is an improved soft-k-means (IS-k-means) clustering algorithm that balances the energy consumption of nodes in wireless sensor networks (WSNs). The techniques of "clustering by fast search and find of density peaks" (CFSFDP) and kernel density estimation (KDE) are used to improve the selection of the initial cluster centers of the soft k-means clustering algorithm. Then, we utilize the flexibility of the soft-k-means and reassign member nodes by considering their membership probabilities at the boundary of clusters to balance the number of nodes per cluster. Furthermore, we use multi-CHs to balance the energy consumption within clusters. Extensive simulation results show that, on average, the proposed algorithm can postpone the first node death, the half of nodes death, and the last node death when compared to various clustering algorithms from the literature.

The second contribution tackles the problem of minimizing the total energy consumption of the UAV-IoT network. Specifically, we formulate and solve the optimization problem that jointly finds the UAV's trajectory and selects CHs in the IoT network. The formulated problem is a constrained combinatorial optimization and we develop a novel deep reinforcement learning (DRL) with a sequential model strategy to solve it. The proposed method can effectively learn the policy represented by a sequence-to-sequence neural network for designing the UAV's trajectory in an unsupervised manner. Extensive simulation results show that the proposed DRL method can find the UAV's trajectory with much less energy consumption when compared to other baseline algorithms and achieves close-to-optimal performance. In addition, simulation results show that the model trained by our proposed DRL algorithm has an excellent generalization ability, i.e., it can be used for larger-size problems without the need to retrain the model.

The third contribution is also concerned with minimizing the total energy consumption of the UAV-aided IoT networks. A novel DRL technique, namely the pointer network-A* (Ptr-A*), is proposed, which can efficiently learn the UAV trajectory policy for minimizing the energy consumption. The UAV's start point and the ground network with a set of pre-determined clusters are fed to the Ptr-A*, and the Ptr-A* outputs a group of CHs and the visiting order of CHs, i.e., the UAV's trajectory. The parameters of the Ptr-A* are trained on problem instances having small-scale clusters by using the actor-critic algorithm in an unsupervised manner. Simulation results show that the models trained based on 20-clusters and 40-clusters have a good generalization ability to solve the UAV's trajectory planning problem with different numbers of clusters, without the need to retrain the models. Furthermore, the results show that our proposed DRL algorithm outperforms two baseline techniques.

In the last contribution, the new concept, age-of-information (AoI), is used to quantify the freshness of collected data in IoT networks. An optimization problem is formulated to minimize the total AoI of the collected data by the UAV from the ground IoT network. Since the total AoI of the IoT network depends on the flight time of the UAV and the data collection time at hovering points, we jointly optimize the selection of the hovering points and the visiting order to these points. We exploit the state-of-the-art transformer and the weighted A* to design a machine learning algorithm to solve the formulated problem. The whole UAV-IoT system, including all ground clusters and potential hovering points of the UAV, is fed to the encoder network of the proposed algorithm, and the algorithm's decoder network outputs the visiting order to ground clusters. Then, the weighted A* is used to find the hovering point for each cluster in the ground IoT network. Simulation results show that the model trained by the proposed algorithm has a good generalization ability to generate solutions for IoT networks with different numbers of ground clusters, without the need to retrain the model. Furthermore, results show that our proposed algorithm can find better UAV trajectories with the minimum total AoI when compared to other algorithms.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant Colony Optimization
AoI	Age of Information
BS	Base Station
CHs	Cluster Heads
CFSFDP	Clustering by Fast Search and Finding of Density Peaks
CNNs	Convolutional Neural Networks
CVRP	Capacitated Vehicle Routing Problem
DQN	Deep Q-network
DRL	Deep Reinforcement Learning
DNNs	Deep Neural Networks
DDPG	Deep Deterministic Policy Gradient
EV	Energy Variance
FPT	Fixed Parameter Tractable
FND	First Node Death
GTSP	Generalized TSP
HND	Half of Nodes Death
IoT	Internet of Things
IS-k-means	Improved Soft-k-means
KDE	Kernel Density Estimation
LoS	Line-of-Sight
LBCP	Load Balanced Clustering Problem

LEACH	Low Energy Adaptive Clustering Hierarchy
LSTM	Long Short-Term Memory
LND	Last Node Death
NLoS	Non Line-of-Sight
PDF	Probability Density Function
Ptr-A*	Pointer Network-A*
RL	Reinforcement Learning
RNNs	Recurrent Neural Networks
Seq2Seq	Sequence-to-Sequence
SNR	Signal-to-Noise Ratio
TDMA	Time Division Multiple Access
TDM	Time Division Multiplexing
TSP	Traveling Salesman Problem
TSPN	TSP with Neighborhoods
TWA*	Transformer-weighted-A*
3D	Three-dimensional
UAVs	Unmanned Aerial Vehicles
VRP	Vehicle Routing Problem
WSNs	Wireless Sensor Networks

1. Introduction

1.1 Motivation

Cisco forecasts that by 2030 approximately 500 billion Internet-of-Things (IoT) devices will be deployed worldwide [1]. These devices are linked via IoT systems, and the data generated by these devices can be accumulated, processed, and distributed by IoT services and applications for different purposes. Wireless Sensor Networks (WSNs) are among the basic components of IoT, which integrate the physical world with the information world to expand the functions of existing networks and the ability of human to understand the world. IoT applications span a wide range of areas in our daily life. For example, monitoring in different fields is one of the most common applications. In agricultural monitoring, IoT devices are deployed in given areas to monitor and collect physical conditions of the surrounding environment such as temperature, humidity, and pressure [2]. Devices also can be deployed in cities to monitor the concentration of dangerous gases [3]. Furthermore, in high-speed train monitoring, IoT devices can provide information on the health of bridges, tracks, and various train components, which can help improve safety and reduce maintenance costs. In addition to monitoring, IoT has gradually appeared in various new forms, such as industrial IoT, Internet of vehicles, smart home, intelligent manufacturing systems, to name a few [4].

In IoT networks, most of the IoT devices are designed to be small, inexpensive, and wireless. Since IoT devices also need energy resources to acquire, process, and transfer data, they are commonly powered by on-devices power supply. However, battery technology is significantly limited in large-scale IoT deployments because the amount of energy that can be stored in batteries is limited. Batteries need to be recharged or replaced periodically, which not only leads to inconvenience and high cost, but can also be impossible in some deployments. Hence, reducing the energy consumption of IoT networks is very important to prolong their lifetime.

Clustering in energy-limited IoT networks has been widely investigated to reduce the energy consumption of the overall IoT networks [5]. Clustering is the task of dividing the "data" into clusters of similar objects. Each cluster includes the data points that are most similar to other data points in the same group, and less similar to the data points in other groups. In IoT networks, clustering-based algorithms group devices that are close to each other into the same cluster based on distance, and each device belongs to one cluster only. All member devices sense their surrounding environment and send the results to the cluster heads (CHs). Then, CHs collect and transmit the sensed data to the base station (BS) via single-hop or multi-hops transmission. Each device consumes a certain amount of energy when it collects, processes, and transmits data, and a device is defined to be dead when it runs out of energy.

Clustering algorithms can significantly reduce energy consumption of devices in IoT networks when compared to non-clustering-based data collection methods, because they can avoid long-distance communication for devices. Although clustering methods are considered to be efficient ways to save energy for devices in IoT networks, the structure of clusters has a considerable impact on networks' lifetime. This is because a poor clustering structure of IoT networks often leads to inefficient energy consumption. Clustering algorithms have been extensively studied in IoT networks. However, there are several design challenges that, if not properly addressed, may lead to poor clustering structures. The first of these challenges is how to choose the initial cluster centers, which will affect the subsequent cluster formation. Second, unbalanced clusters (large or small) can lead to unbalanced energy consumption of devices, which can significantly affect the lifetime of networks. Third, choosing unsuitable CHs can quickly exhaust their energy because CHs need to consume more energy to receive and forward data. Fourth, in some clustering algorithms, the devices in IoT networks are organized in clusters for a fixed number of communication rounds. This may result in unbalanced energy consumption for devices. Hence, the first objective of this research is to balance energy consumption of devices in wireless networks in order to extend their lifetime.

The use of unmanned aerial vehicles (UAVs) to collect data in IoT networks has received increasing attention due to their numerous advantages, such as cost-effectiveness, high probability of line-of-sight (LoS) links with the ground devices, mobility, and reliable network access [6]. UAVs are considered as mobile sinks for receiving data from CHs, and then, they can carry/transmit the collected data to terrestrial BSs for further processing. Using UAVs as mobile sinks can reduce energy consumption of ground devices in IoT networks when compared to the traditional multi-hop networks which transmit data from each device to a BS over a long distance or several hops. Despite these advantages, the integration of UAVs and IoT networks still grapples with many challenges. First of all, due to the limited energy source carried by UAVs, the service range of UAVs is constrained by the reality that they cannot travel very long distances or fly for a long period of time. Second, the battery life of ground devices in IoT networks is typically limited. As a result, frequent communication with UAVs can cause devices to exhaust their energy rapidly. Hence, it is important to study the energy consumption minimization problem in UAV-enabled IoT networks.

Prior works on energy consumption minimization for UAV-enabled wireless networks can be classified into three categories depending on the objectives. The first category only considers minimizing the UAV's energy consumption. In contrast, the second category considers only minimizing the energy consumption of ground devices in the UAV-aided wireless networks. In the third category, the energies of both the UAV and ground devices are taken into account when minimizing energy consumption of the UAV-enabled networks. However, most of prior works assume that the UAV directly communicates with each device of the ground wireless network. Such a scenario leads to high energy consumption of both the UAV and ground devices, especially when the network size increases. As a result, the UAV may run out of its energy in flight or may need to recharge its battery frequently, and ground devices may quickly exhaust their energy. Hence, the second objective of this research is to minimize the overall energy consumption of the UAV-enabled IoT networks.

Apart from energy consumption, the freshness of the collected information is another key performance metric in time-sensitive IoT applications, such as environmental monitoring and safety protection. In these applications, the generated data needs to be sent to the

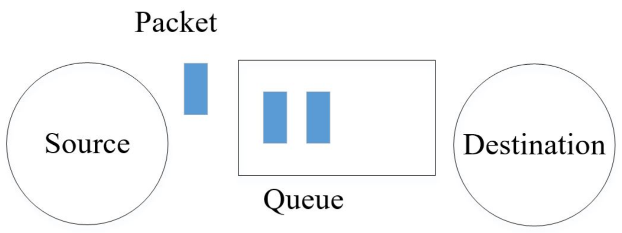


Figure 1.1 A simple information update system.

destination as soon as possible. Outdated information can lead to incorrect control and even cause major disasters. Therefore, it is essential to ensure the freshness of data that arrives at the destination. To measure the freshness of information, a new performance metric called the age of information (AoI) was proposed in [7]. From the perspective of the receiver, AoI describes the amount of time elapsed since the generation of the most recent data update. According to the definition of AoI, the AoI of a packet at time ζ is defined as [8]

$$A(\zeta) = (\zeta - u(\zeta))^{+} \tag{1.1}$$

where $u(\zeta)$ is the instant at which the packet is generated, and $(x)^{+} = \max\{0, x\}$.

AoI-based data collection can achieve information freshness in IoT networks, which is quite different from the traditional delay-based metric. To better explain the concept of AoI, consider a simple information update system with a source node and a destination node as illustrated in Figure 1.1. Based on the enqueue-and-forward model [9], the source node enqueues updates and sends them to a destination via a channel. The low data update frequency of the source node will cause short queuing delays because the queue is almost empty. But the destination may eventually have stale data due to low update frequency. However, high data update frequency will not only increase queuing delay, but also increase the AoI of data. This is because data updates require a long waiting time in the queue. Therefore, the delay increases as the update frequency increases, but the AoI first decreases and then increases with respect to the update frequency [8].

Energy-constrained IoT networks generally experience higher likelihood of packet loss, which can affect the timely delivery of sensed data. This is a more serious problem in large-scale deployment IoT networks, where devices may have very poor direct links to their destinations. New IoT network architectures need to be designed to efficiently utilize the limited energy of IoT devices and ensure timely delivery of their data and status updates. As UAVs are becoming key components of wireless network architectures, utilizing them to maintain the freshness of data has attracted increasing attention and will also be the third objective of this research.

1.2 Literature Review

This subsection gives a detailed review of existing studies that are directly related to our research objectives. Note that devices, nodes, and sensors are synonymous terminologies and used interchangeably in this thesis. Likewise, BS, gateway, destination, and sink have the same meaning and are used interchangeably.

1.2.1 Balancing WSNs' Energy Consumption

To balance the energy consumption of devices and extend their lifetime, different clustering techniques have been used in WSNs. By clustering, devices in WSNs are divided into small clusters where each cluster elects one device as the CH to collect data from member devices in the same cluster and forward the collected data to the BS. Figure 1.2 illustrates a cluster-based WSN architecture. Clustering algorithms can effectively avoid long-distance communications of devices, and hence, reduce their energy consumption. There is rich literature concerning the problem of clustering in WSNs.

The low energy adaptive clustering hierarchy (LEACH) [10] is one of the well-known energy-efficient clustering algorithms. In every round of LEACH, each node is assigned a probability between 0 and 1, which is used to determine whether or not it could be a CH. If the assigned probability of a node is less than a predetermined threshold, then this node becomes a CH. After CHs are decided, other member nodes join the cluster of the nearest CH depending on the strength of the advertisement message from CHs. However, LEACH may result in a nonuniform distribution of CHs.

Low energy adaptive clustering hierarchy centralized (LEACH-C) [11] is a modified version of LEACH, where each node sends its current location and residual energy to the BS. The

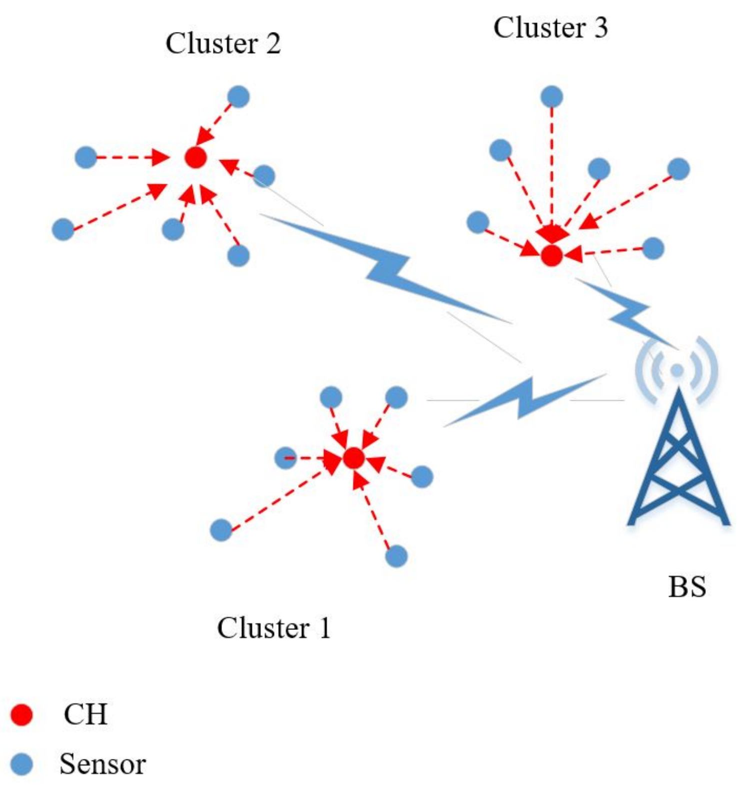


Figure 1.2 A cluster-based WSN architecture.

BS determines the required number of clusters by using the simulated annealing algorithm and ensures balancing the load among different clusters. The average energy consumption of LEACH-C is lower than that of LEACH. However, this centralized approach can increase the communication overhead required for CHs selection. The authors in [12] proposed an efficient CHs election method where CHs are alternately selected among the nodes with higher energy in different communication rounds. In particular, the method considers the initial energy, residual energy, and an optimal number of CHs in the phase of CHs selection. Then, member nodes join different CHs according to the distances between them and CHs to form clusters.

A joint clustering and routing (JCR) algorithm is proposed in [13] to improve the energy efficiency of large-scale WSNs. JCR employs a back-off timer and gradient routing to simultaneously execute the CH selection and multi-hop routing under the constraint on the maximum transmission range. The authors in [14] presented a node-density-based clustering and mobile elements algorithm (NDCMC) to combine the hierarchical routing and mobile elements data collection in WSNs. In NDCMC, the nodes surrounded by a large number of deployed nodes are selected as CHs in order to improve the efficiency of intra-cluster routing. To solve the load balanced clustering problem (LBCP) in WSNs, the authors in [15] used a fixed parameter tractable (FPT) approximation algorithm with an approximation factor of 1.2 based on the parameterized complexity theory to develop a new energy-efficient and energy-balanced routing algorithm. The FPT-approximation algorithm determines which gateway each sensor node must be assigned to, which can better balance load and energy consumption among the gateways. In addition, the authors proposed a routing tree for the inter-cluster communication to distribute the overhead of the routing among almost all of the nodes. In [16], the authors proposed an FPT-approximation algorithm with an approximation factor of 1.1, which is more precise than previous approximation factors reported for LBCP. The FPT-approximation algorithm is used to assign sensor nodes to gateways such that the maximum load of the gateways is minimized. Then, an energy-aware routing algorithm is employed to find the optimal routing tree between gateways and the sink with the aim of balancing the energy consumption of the nodes. The same authors also considered another FPT-approximation algorithm with an approximation factor of 1.1 in [17]. In order to make the FPT-approximation algorithm to be practical in large-scale WSNs, a virtual grid infrastructure with several equal-size cells is used where the FPT-approximation algorithm runs in each cell independently. These FPT-approximation algorithm only chooses one CH in each cluster, which may cause all nodes to be re-clustered frequently because CHs may quickly exhaust their energy. In addition, they use a fixed number of communication rounds during each steady-phase, which may lead to CHs to die earlier.

The k-means algorithm and its variants are the most common machine learning algorithms used in WSNs to balance the devices' energy consumption and prolong their lifetime. The authors in [18] proposed a modified k-means clustering algorithm that considers two factors, namely, (i) distances among CHs and their member nodes, and (ii) the remaining energy of nodes, to reduce the overall energy consumption and extend the network lifetime. In [19], the authors proposed a hybrid clustering algorithm based on the k-means clustering algorithm and LEACH, where balanced clusters are generated by k-means and CHs are selected by LEACH. This hybrid algorithm outperforms LEACH in terms of the energy consumption. However, due to the frequent re-clustering, the energy consumption of the nodes may increase in the phase of cluster formation and CHs selection.

An energy efficient clustering protocol based on k-means (EECPK-means) is proposed in [20] with the aim of balancing the load of CHs in WSNs. The midpoint method is used to improve the initial selection of centroids in the k-means algorithm in order to generate balanced clusters. In [21], the authors proposed a method based on fuzzy c-means clustering and particle swarm optimization (FCM-PSO) to reduce the total energy consumption of the network and reduce the number of network disconnects. The FCM-PSO algorithm considers the energy consumption and constraints of communication in the calculations of the CHs and nodes' membership probability. The energy-efficient k-means LEACH (KM-LEACH) algorithm is proposed in [22] to create symmetric clusters and reduce the average intra-cluster communication distance, which can save nodes' energy and improve the network lifetime.

To address the problem of how to control the failure of a CH in each cluster, the k-medoids clustering algorithm and vice CH scheme (VLEACH) are used together with LEACH in [23]. A vice CH will become a new CH in case the CH of a given cluster dies, which helps to prolong the lifetime of WSNs by balancing the nodes' energy consumption. The authors in [24] used k-means and Gaussian elimination algorithms to reduce energy consumption of WSNs and extend their lifetime. An innovative classification algorithm based on "clustering by fast search and finding of density peaks" (CFSFDP) for balancing energy is proposed in [25]. The authors extend the original CFSFDP algorithm to take into account residual energy (in addition to local density and distance) to select CHs, and accordingly cluster nodes based on the selected CHs.

With respect to prolonging the network lifetime, there are several outstanding-questions that need to be answered. (1) How to determine the number of clusters in WSNs? (2) How to choose the initial centers of clusters for machine learning-based clustering algorithms? (3) How to form balanced clusters to avoid unbalanced energy consumption in different clusters? (4) How to choose CHs to prolong the WSNs' lifetime? (5) How to balance the

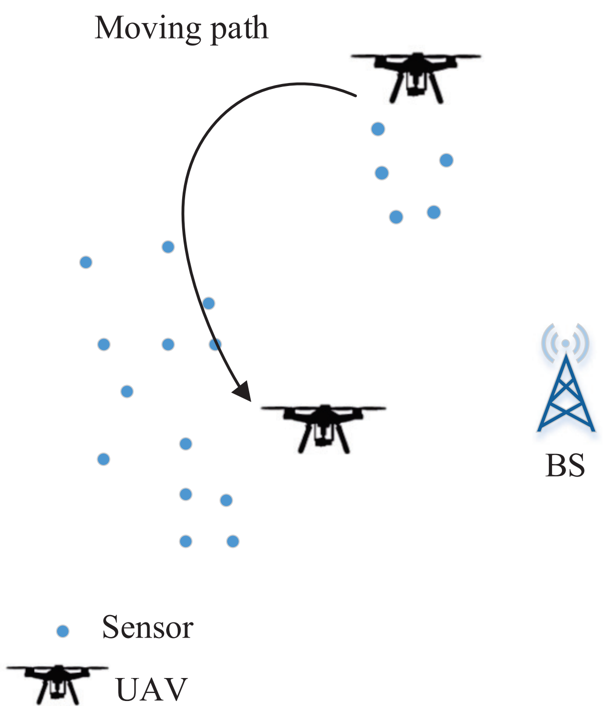


Figure 1.3 An UAV-aided IoT network for data collection.

energy consumption among CHs? The research works discussed previously only consider some of these five problems. In this research, we propose an energy-efficient algorithm with the aim of balancing the energy consumption of all devices in WSNs and extending their lifetime by considering all these five problems.

The reader is referred to Section 1 of Chapter 2 and [5] for more detailed discussion on related works.

1.2.2 Saving Energy in UAV-Aided Networks

Employing UAVs has attracted growing interests and emerged as a state-of-the-art technology for data collection in wireless networks due to their high flexibility and high maneuverability. Figure 1.3 illustrates an example of UAV-aided IoT network where a UAV is dispatched to collect data from ground sensors and forwards the collected data to the BS. Because of the limited energy resources carried by UAVs and devices of wireless networks, it is important to study the energy saving problem in UAV-aided networks.

Comparison of UAVs

Depending on different use cases, there are two main types of UAVs that are used for data collection in IoT networks: fixed wing and rotary wing [26]. Fixed-wing UAVs use fixed static wings like aeroplanes and use energy to provide forward motion only in order to generate lift. Rotary-wing UAVs usually use multiple rotors to create lift via diverting the air downwards. These two types of UAVs have their own strengths and weaknesses. Fixed-wing UAVs are able to cover longer distances, fly for longer times, and can carry heavy payload when compared to rotary-wing UAVs that have limited payload and endurance. Rotary-wing UAVs, on the other hand, have smaller size and are easier to control than fixed-wing UAVs. One downside of fixed-wing UAVs is that they are unable to hover in one spot, which limits their usage for data collection in some IoT applications. In contrast, rotary-wing UAVs not only have the ability to stay stationary to collect data from the ground IoT network but also can move in any directions. Another downside of fixed-wing UAVs is that they tend to be significantly more expensive than rotary-wing UAVs. In general, the choice of UAVs depends on the specific requirements of data collection.

Energy Consumption of UAV-Aided Networks

By jointly considering the UAV's trajectory and devices' transmission schedule, the authors in [27] use an efficient differential evolution-based method to minimize the maximum energy consumption of all devices in an IoT network. In [28], the authors aim to minimize the transmission energy consumption of the sensor nodes within a given data collection time by jointly optimizing the UAV's trajectory and the transmission policy of nodes. In [29], the authors consider maximizing the minimum residual energy of sensor nodes after data transmission in order to prolong the network lifetime. The authors in [30] jointly optimize the sensor nodes' wake-up schedule and the UAV trajectory to reduce the maximum energy consumption of all sensor nodes. In [31], the authors minimize the maximum energy consumption of all IoT devices subject to the UAV's energy budget.

The aforementioned works only focus on minimizing the energy consumption of ground devices in an UAV-aided wireless network. In contrast, other works consider UAV-related energy consumption minimization when UAVs are deployed in wireless networks. In [32], the authors study the problem of minimizing the completion time and the energy consumption of an UAV flying over a large area and propose a fly-and-communicate protocol. The authors in [33] aim to minimize the total energy consumption of the UAV, including both the propulsion and communication energy, in a UAV-enabled system serving multiple ground nodes. The authors in [34] study methods to control a group of UAVs for effectively covering a large geographical region while minimizing their energy consumption. In [35], the authors minimize the total UAV's energy consumption for a given path by optimizing its velocity.

The above two categories of research focus on minimizing the energy consumption of UAVs and ground devices separately; while research in the third category aim to jointly minimize the energy consumption of both UAVs and ground devices. An energy-efficient solution is proposed in [36] to minimize UAVs' and sensors' energy consumption when collecting data from the spatially distributed WSN. By considering the communication power consumption of ground devices and the propulsion power consumption of the UAV, the energy consumption trade-off between the UAV and ground devices in the UAV-enabled data collection system is studied in [37]. The authors in [38] aim to minimize the weighted sum of the energy consumption of all UAVs and the energy consumption of all sensors in a multi-UAVs enabled WSN.

Most of the above studies assume that the UAV directly communicates with each device of the ground wireless network. In this case, if the UAV flies over all devices in a large-scale WSN, it would lead to a long flight trajectory for the UAV which increases its energy consumption. As a result, the UAV may run out of its energy in flight or may need to recharge its battery frequently. Against the above literature, in this research, we consider to minimize the total energy consumption of the ground network and the UAV by designing an energy-efficient UAV trajectory in a cluster-based IoT network. An important difference between our work and existing studies is that we take into account how the UAV interacts with the ground network. Specifically, as illustrated in Figure 1.4, we consider a clustered IoT network and that the UAV only communicates with the CH of each cluster in order to reduce the energy consumption. As such, it is clearly important and relevant to study the

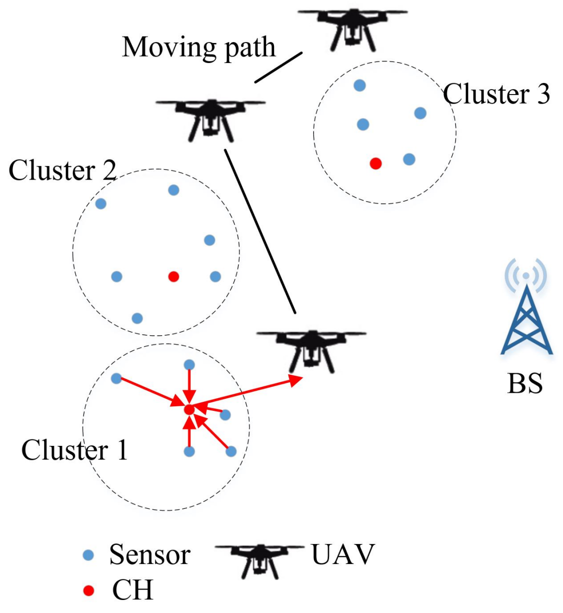


Figure 1.4 An example of UAV-aided cluster-based IoT network.

energy-efficient UAV's trajectory planning in clustered IoT networks in order to minimize the overall energy consumption of the UAV and the ground network.

Energy-Efficient UAV Trajectory Planning

Energy-efficient trajectory planning for UAVs has recently attracted significant research interest, and multiple solutions have been proposed for UAV-enabled wireless networks. Most of UAVs trajectory design problems in wireless networks are $\mathcal{NP}$-hard, and hence, can not be solved by polynomial-time complexity algorithms.[1]

In general, existing solutions for energy-efficient UAV trajectory planning can be classified into two categories: traditional methods and machine learning based techniques. In the first category, researchers mostly use mathematical programming or heuristic algorithms to solve the trajectory optimization problem. In order to find the most suitable trajectory for the UAV to collect data, the authors in [6] firstly find the global optimal solution for a small-scale IoT network by using the high-complexity branch, reduce, and bound algorithm, and then

[1]NP-hardness (non-deterministic polynomial-time hardness) is the defining property of a class of problems that are informally "at least as hard as the hardest problems in NP".

they use the successive convex approximation to develop a sub-optimal algorithm to generate the solution for the large-scale IoT network. The authors in [39] jointly optimize the route planning and task assignment for a number of UAVs from an energy-efficient perspective by using the dynamic programming and the Gale-Shapley algorithms. The authors of [40] develop a hierarchical directional dynamic programming algorithm to solve the UAV's path planning. In [29], the authors jointly optimize the path and the UAV's velocities to minimize the total energy consumption. The optimization problem is solved by dynamic programming, which achieves a good performance at a high computation cost. However, the computation time of mathematical programming algorithms may increase exponentially as the problem size increases. Although some heuristic algorithms are applied to design the energy-efficient path in the UAV-enabled wireless networks, such as ant colony optimization [41], differential evolution [27], and cuckoo search [42], they usually cannot fully adapt to the increasing complexity of scalable wireless networks and also not guaranteed to provide close-to-optimal performance.

Regarding the machine learning-based category, deep reinforcement learning (DRL) and reinforcement learning (RL) are the most common techniques in solving the UAV's trajectory planning problems. In [34], the authors propose a DRL-based method which is composed of two deep neural networks (DNNs) and deep deterministic policy gradient (DDPG) to maximize the energy efficiency for a group of UAVs by jointly considering communications coverage, energy consumption, and connectivity. In order to minimize the UAV's transmission and hovering energy, the authors in [43] formulate the energy-efficient optimization problem as a Markov decision process. Then, they use two DNNs and the actor-critic-based RL algorithm to develop an online DRL algorithm that shows a good performance in terms of energy savings. In [44], the authors jointly optimize the UAV's 3D trajectory and the frequency band allocation of ground users by considering the UAV's energy consumption and the fairness of the ground users. A DDPG-based DRL algorithm is developed to generate the energy-efficient trajectory with fair communication service to ground users. In [45], with the aim of designing an energy-efficient UAV's route for long-distance sensing tasks, the authors propose a DRL-based framework where convolutional neural networks (CNNs) are used for extracting features and the deep Q-network (DQN) is utilized to make decisions. Towards realizing green UAV-enabled IoT, the authors in [46] formulate the UAV's path planning problem as a dynamic decision optimization problem, which is solved by dueling DQN. In [47], the authors use the DQN with experience replay memory to solve the formulated energy-efficient trajectory optimization problem, while maintaining data freshness [2]. With the objective of saving energy, the authors in [48] propose a deep stochastic online scheduling algorithm based on two DNNs and the actor-critic to overcome the traditional DRL's limitations in addressing the UAV trajectory optimization problem.

The aforementioned machine learning-based methods show strong ability to handle complex wireless environments and effectively learn the UAV's trajectory policy. Motivated by this, we also make use of machine learning to solve the trajectory planning problems in our research. Specifically, in this thesis, we formulate the UAV's trajectory problem as a combinatorial optimization problem which can be modeled as a sequential decision-making problem. The UAV works as the agent to interact with the environment to perform a set of actions to construct a solution. In sequential decision problems, the decision maker makes successive observations of the process where each observation has a cost [49]. The goal of sequential decision-making is to form a rule, called a policy, that maximizes a measure of the total amount of accumulated cost. The above mentioned CNN, DQN, and DDPG are not suitable for dealing with sequential problems as the output of the current step depends on the output of previous steps in sequential problems, and CNN, DQN, and DDPG do not have the ability to store information of previous steps for a very long time. Sequence-to-sequence (Seq2Seq) models composed of recurrent neural networks (RNNs) and an attention mechanism are emerging as promising machine learning techniques to handle sequential problems. This is largely because the RNNs' hidden units can store historical information [50]. In this research, we exploit the appealing concept of Seq2Seq to design the DRL algorithm to solve the UAV's path planning problem in clustered IoT networks.

[2]The concept of data freshness is explained in Section 1.1.

Sequence-to-Sequence Model

Seq2Seq learning was first introduced for machine translation by Google [51]. It concerns training models to map input sequences from one domain (e.g. English language) to output sequences in another domain (e.g. French language) that are based on deep learning. Seq2Seq models have two main components: *encoder* and *decoder*, as illustrated in Figure 1.5. For the illustration in Figure 1.5 with an example of the input sequence containing three elements, the encoder of a seq2seq network is a RNN that reads the input sequence $\boldsymbol{\mathcal{G}} = \{g_1, g_2, g_3\}$ and generates a vector which is a semantic summary of the input sequence. Then, the semantic summary vector is given as an input to the decoder, which is another RNN, to output a target sequence $\boldsymbol{\mathcal{T}} = \{\pi_1, \pi_2, \pi_3\}$. The number of total decoding steps is equals to the length of the input sequence. The probability of the target sequence $\boldsymbol{\mathcal{T}}$ can be factorized by a product of conditional probabilities according to the chain rule:

$$P_\theta(\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{G}}) = \prod_{t=1}^{K} P(\pi_t|\pi_1, \dots, \pi_{t-1}, \boldsymbol{\mathcal{G}}), \tag{1.2}$$

where t is the time step, K is the length of the input sequence, $P_\theta(\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{G}})$ parameterized by θ is a *stochastic policy*. The conditional probability $P(\pi_t|\cdot)$ models the probability of any element being decoded at the t-th time step according to the given $\boldsymbol{\mathcal{G}}$ and the elements already decoded in previous time steps. A trained θ can assign high probabilities to good results and low probabilities to bad results. The reinforcement learning can be applied to train the optimal model policy θ^* for producing the optimal output sequence $\boldsymbol{\mathcal{T}}^*$ with the highest probability.

Sequence-to-Sequence Learning for Combinatorial Optimization

In recent years, Seq2Seq models are used for a variety of applications. For example, some combinatorial optimization problems, such as traveling salesman problem (TSP), vehicle routing problem (VRP), etc., are often solved as Seq2Seq prediction problems in machine learning [52]. Similarly, in our considered model, since the UAV needs to visit all clusters sequentially to collect data, we also view the combinatorial optimization problem of interest as a sequential problem and aim to design an algorithm to have good ability to

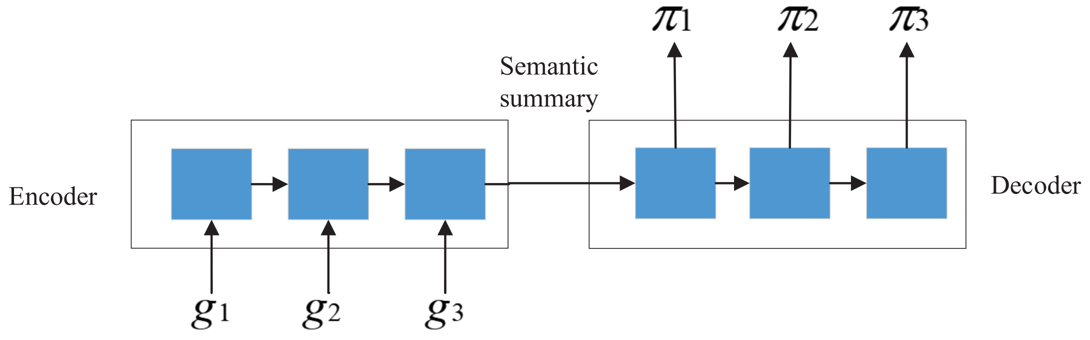


Figure 1.5 A general structure of a Seq2Seq model.

learn the policy on sequential data. Machine learning algorithms, such as CNN, DQN, and DDPG, are inefficient to handle sequential problems in which the current element of the sequence depends on historical information from previous elements of the sequence. The reason is that these algorithms cannot store information of past elements for very long time [50]. RNNs with long short-term memory (LSTM) are frequently used to deal with sequential problems because their hidden units can store historical information for long time steps. In addition, RNNs are the state-of-the-art neural networks to tackle variable-length sequences, e.g., variable-size data in our problem, by re-using the neural network blocks and parameters at every step of the sequence [53]. Attention mechanism is another technique to process a variable-length sequence by sharing its parameters. Hence, RNNs and the attention mechanism-based Seq2Seq models are emerging as attractive techniques to tackle variable-size sequential problems and they show promising results in various domains.

For example, in [54], the authors proposed a new neural network structure, called *pointer network*, which is composed of RNNs and an attention mechanism. The pointer network is trained in a supervised fashion to learn solutions to three different combinatorial optimization problems: convex hull, delaunay triangulation, and TSP. The work in [55] focuses on TSP and uses a RL-based unsupervised method to train the parameters of the pointer network by giving a set of city coordinates and considering negative tour length as the reward signal. This work obtains better results when compared to the supervised learning in [54]. In [56], the authors transformed the online routing problem to a vehicle tour generation problem and proposed a structural graph embedded pointer network to develop online vehicular routes in intelligent transportation systems. The proposed algorithm outperforms conventional strategies with limited computation time for both static and dynamic logistic systems.

The authors in [57] viewed keyword recommendations as a combinatorial optimization problem and proposed a modified pointer network to solve it in sponsored search advertising system. The model is trained by an actor-critic framework, and the equal size k-means method is proposed to accelerate the training. In [58], a simplified pointer network is introduced to solve VRP in dynamic traffic environments. The parameters of the model are trained by a policy gradient algorithm. The trained model can solve similar-sized instances without retraining for every new instance. The authors in [59] propose a DRL algorithm with Seq2Seq and the actor-critic to control traffic in WSNs for energy efficiency. The artificial agent learns from the state of a wireless sensor network, and outputs the optimal route path. The authors in [60] model each sub-problem of the multi-objective TSP optimization as a pointer network. Then, they collaboratively optimize the model parameters of all the sub-problems using a neighborhood-based parameter-transfer strategy and the DRL training algorithm. The trained model can scale to any new problems without the need to retrain the model. The network function virtualization forward graph embedding problem is formulated as a constrained combinatorial optimization problem in [61], which is solved by the proposed Seq2Seq-based pointer network.

Given that we formulate the UAV trajectory planning problem in an UAV-aided cluster-based IoT network as a combinatorial optimization problem, Seq2Seq models with RNNs and an attention mechanism are the right ingredients for developing an efficient DRL algorithm to solve this challenging problem. The reader is referred to Section 1 of Chapter 3, Section 1 of Chapter 4, and [62] for more detailed discussion on related works.

1.2.3 Freshness Data Collection in UAV-Aided Networks

AoI-Oriented Data Collection

Extensive studies have been done on AoI-oriented data collection in UAV-assisted wireless networks due to the importance of AoI. In [63], the authors aimed to minimize the average

AoI of the system by optimizing the trajectory of the UAV in a UAV-aided data collection system. In [64], the authors optimized the trajectory of the UAV to minimize the maximal AoI and the average AoI of sensor nodes. In [65], the authors assumed the UAV supports three modes to collect data and jointly optimize the trajectory and data collection modes of the UAV to minimize the average AoI of all ground nodes. In [66], the UAV trajectory, energy, and service time allocation were jointly optimized by an iterative algorithm in order to minimize the overall peak AoI of the system. The authors in [47] developed an energy-efficient navigation policy for the UAV to improve data freshness of the IoT network. In order to minimize the weighted sum of AoI, the authors in [67] jointly optimized the flight trajectory of the UAV and the transmission scheduling of sensors.

Against the above literature, in this research, we also consider to optimize the UAV trajectory to minimize the total AoI of the collected data in a UAV-aided clustered IoT network. The optimization problem is formulated as a TSP with neighborhoods (TSPN) and further converted to a generalized TSP (GTSP), which is a combinatorial optimization problem. By considering performance, computational complexity, and generalization ability when designing the proposed algorithm, we use the state-of-the-art transformer to solve the formulated GTSP.

Transformer

Transformer is a deep learning model proposed by Google for machine translation [68]. It is an architecture for transforming one sequence into another one by learning relationships in sequential data. For a vanilla transformer, its encoding component is a stack of encoders, and the decoding component is a stack of decoders, as shown in Figure 1.6. The encoders are all identical in structure that includes a self-attention layer and a feed-forward neural network layer, as shown in Figure 1.7. The self-attention layer of each encoder receives input encodings from the previous encoder and measures their relevance to each other to generate output encodings. Then, the output encodings are fed to the feed-forward neural network in the same encoder for further processing. Afterward, the output encodings of each encoder are passed to the next encoder that performs the same operations as in the previous

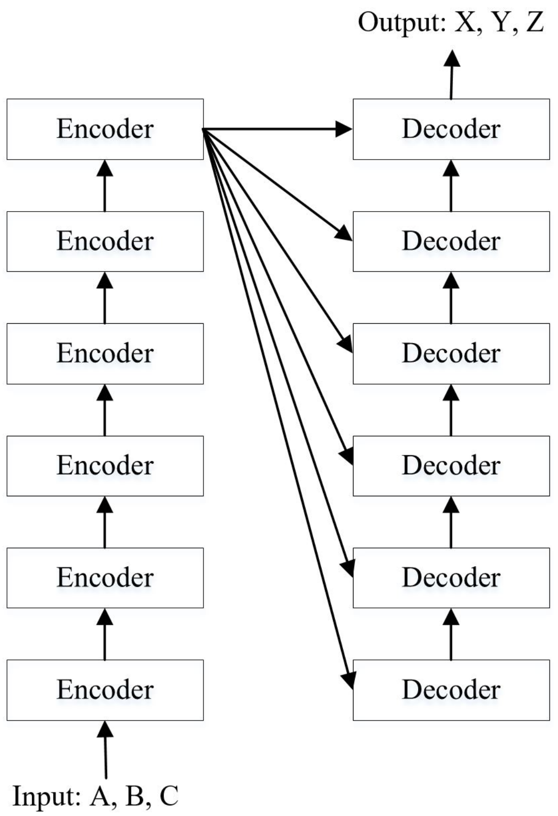


Figure 1.6 Structure of a vanilla transformer.

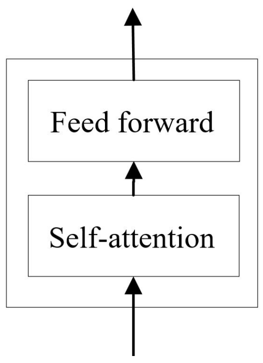


Figure 1.7 Encoder.

encoder. The function of each encoder is to produce encodings which have information regarding which parts of the inputs are relevant to each other. The decoders also have the same structure. Each decoder has a self-attention layer, a feed-forward neural network layer, and an additional attention layer over the encodings, as shown in Figure 1.8. The function of each decoder is to process the encoder component's output iteratively one layer after another, and finally uses the incorporated contextual information to generate an output

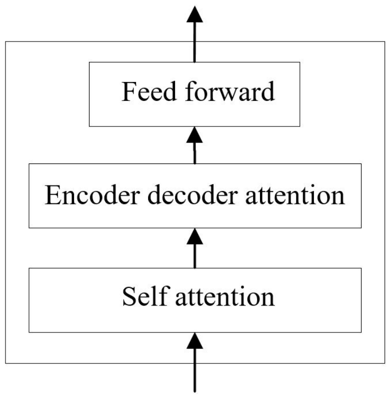


Figure 1.8 Decoder.

sequence. The structure of a transformer highlights the fact that attention mechanisms alone can match the performance of some sequential models that are composed of RNNs and an attention mechanism.

Transformer for Combinatorial Optimization

Transformer has achieved great success in many areas of artificial intelligence in the past four years, such as computer vision, audio processing, document summarization, document generation, and sequence prediction. Some researchers also attempt to use transformer and its variants to tackle combinatorial problems, such as TSP. In [69], the cities in TSP were encoded by a transformer and decoded sequentially through a query consisting of the last three cities in the partial tour. The employed transformer was trained by reinforcement learning. In [70], the authors also used the transformer architecture as the encoder network and the decoder network outputs the result sequentially based on the embeddings from the encoder and the outputs generated at previous steps. The encoder and the decoder networks were trained using a RL algorithm with a deterministic greedy baseline. The authors in [71] proposed a transformer-based framework to automatically learn improved heuristics on two representative routing problems: TSP and capacitated vehicle routing problem (CVRP). In [72], the authors used the standard transformer architecture to tackle the TSP and achieve an improved performance over recent learned heuristics.

Inspired by the successes of using transformer in solving many problems of route planning, we also adopt it in this research for solving our formulated GTSP combinatorial optimization problem. The reader is referred to Section 1 of Chapter 5 and [73], [74] for more detailed discussion on related works.

1.3 Research Objectives

There are three objectives of this thesis and they are summarized as follows.

1. **Balancing WSNs' energy consumption:** To address the challenges discussed before in applying clustering algorithms to WSNs, we develop an energy-efficient clustering algorithm to balance the energy consumption of all devices in WSNs [3] and extend the network lifetime. Different from existing clustering algorithms that select the initial cluster centers randomly, we investigate the use of a soft-k-means clustering algorithm in our considered problem. We first optimize the initial centroids of the soft k means. Then, the number of devices per cluster is optimized via a reassigning mechanism to balance clusters. Since the clustering process needs to be repeated continually, the communication cost during the clustering phase is increased. The energy consumption in this phase need to be further optimized by reducing the frequency of re-clustering. Instead of using a fixed number of communication rounds, a mechanism with a variable number of communication rounds is considered to avoid having CHs dying early, where the number of communication rounds is determined by the residual energy of CHs.

2. **Saving energy in UAV-aided IoT networks:** We investigate the problem of minimizing the total energy consumption of the UAV and ground devices in a clustered IoT network, which has not been well researched in the literature. We assume that devices on the ground have been clustered according to some specific criterion, e.g., based on their geographical locations. In each pre-determined cluster, one of the ground devices is selected as the CH, which is responsible for collecting data from non-CH devices in the same cluster. Hence, the UAV only needs to visit a set of CHs for gathering data along a planned trajectory that is determined by locations of the ground CHs.

[3]In this thesis, we assume all devices are stationary after deployed.

The selection of CHs affects both the energy consumption of ground devices and that of the UAV. As such, the problem of overall energy consumption minimization of an UAV-aided IoT network can be viewed as the UAV trajectory optimization problem by jointly selecting CHs and planning the UAV's visiting order to these CHs. Such a problem can be formulated as a combinatorial optimization problem. DRL techniques are investigated to solve the formulated problem.

3. **Freshness data collection in UAV-aided IoT networks:** Most existing works discussed before on optimizing the AoI-oriented transmission policies for the update of data in wireless networks are based on the enqueue-and-forward model. However, this model is not suitable for data update in UAV-aided IoT networks. This is because the UAV needs to collect data from different IoT devices, and it cannot verify whether packets are generated from certain IoT devices in a timely manner. Hence, existing AoI-based transmission policies for wireless networks in prior works cannot be applied to UAV-assisted IoT networks.

   In this research, we investigate an AoI-oriented data collection model in cluster-based IoT networks. The problem of interest is to jointly optimize the UAV's hovering points and trajectory to achieve the minimal AoI data collection. The optimization problem is formulated as a TSPN, which is extremely challenging because it includes a continuous problem (optimization of hovering points) and a combinatorial problem (optimization of visiting order). To reduce the computational complexity for obtaining solutions, we convert the formulated continuous optimization TSPN into a GTSP by using the sampling-based idea. By viewing the converted GTSP as a "machine translation", we use the state-of-the-art transformer to solve such a problem.

1.4 Organization of the Thesis

The thesis is written in a manuscript-based style. In each chapter, a brief introduction precedes each manuscript in order to connect the manuscript to the main context of the thesis.

Chapter 1 gives the motivation of the research and an in-depth review of related literature, and outlines the three research objectives.

Chapter 2 studies the problem of balancing energy consumption of WSNs and extending their lifetime. An improved soft-k-means clustering algorithm is proposed. In order to show advantages of the proposed algorithm in balancing the energy consumption, comparisons with other algorithms from literature are also discussed.

In Chapter 3, the problem of minimizing the energy consumption of UAV-IoT networks is investigated, which is formulated as jointly designing the UAV's trajectory and selecting cluster heads in IoT networks. A sequential model is used to model and solve the formulated problem. Chapter 4 further investigates the energy consumption minimization problem in UAV-assisted networks by using the sequence-based pointer network and A* search. The generalization ability of the proposed algorithm is studied on both small-scale-clusters networks and large-scale-clusters networks.

Chapter 5 investigates the problem of freshness data collection in UAV-aided IoT networks. The optimization problem is formulated as an AoI-oriented UAV trajectory planning problem where the hovering points of the UAV and the visiting order to these points are jointly optimized. The state-of-the-art transformer and the weighted A* are used to design a machine learning algorithm to solve the formulated problem.

Finally, Chapter 6 summarizes the contributions of this thesis and suggests potential research problems for future work.

References

[1] Cisco, "Visual networking index: forecast and trends, 2017–2022," tech. rep., Jul. 2019.

[2] O. Friha, M. A. Ferrag, L. Shu, L. Maglaras, and X. Wang, "Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, pp. 718–752, Apr. 2021.

[3] T. Anagnostopoulos, A. Zaslavsky, K. Kolomvatsos, A. Medvedev, P. Amirian, J. Morley, and S. Hadjieftymiades, "Challenges and opportunities of waste management in IoT-enabled smart cities: A survey," *IEEE Transactions on Sustainable Computing*, vol. 2, pp. 275–289, Jul. 2017.

[4] F. Javed, M. Afzal, M. Sharif, and B. Kim, "Internet of things (IoT) operating systems support, networking technologies, applications, and challenges: A comparative review," *IEEE Communications Surveys and Tutorials*, vol. 20, pp. 2062–2100, Mar. 2018.

[5] L. Xu, R. Collier, and G. M. P. O'Hare, "A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios," *IEEE Internet of Things Journal*, vol. 4, pp. 1229–1249, Oct. 2017.

[6] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 34–46, Jan. 2020.

[7] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update?," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pp. 2731–2735, Mar. 2012.

[8] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Transactions on Information Theory*, vol. 63, pp. 7492–7508, Aug. 2017.

[9] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Transactions on Information Theory*, vol. 62, pp. 1897–1910, Apr. 2016.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. the 33rd Annual Hawaii International Conference on System Sciences (HICSS)*, pp. 1–10, Jan. 2000.

[11] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, Oct. 2002.

[12] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "Residual energy-based cluster-head selection in WSNs for IoT application," *IEEE Internet of Things Journal*, vol. 6, pp. 5132–5139, Jun. 2019.

[13] Z. Xu, L. Chen, C. Chen, and X. Guan, "Joint clustering and routing design for reliable and efficient data collection in large-scale wireless sensor networks," *IEEE Internet of Things Journal*, vol. 3, pp. 520–532, Aug. 2016.

[14] R. Zhang, J. Pan, D. Xie, and F. Wang, "NDCMC: A hybrid data collection approach for large-scale WSNs using mobile element and hierarchical clustering," *IEEE Internet of Things Journal*, vol. 3, pp. 533–543, Aug. 2016.

[15] R. Yarinezhad and S. N. Hashemi, "A routing algorithm for wireless sensor networks based on clustering and an FPT-approximation algorithm," *Journal of Systems and Software*, vol. 155, pp. 145–161, Sep. 2019.

[16] R. Yarinezhad and S. N. Hashemi, "Increasing the lifetime of sensor networks by a data dissemination model based on a new approximation algorithm," *Ad Hoc Networks*, vol. 100, p. 102084, Apr. 2020.

[17] R. Yarinezhad and S. N. Hashemi, "Solving the load balanced clustering and routing problems in WSNs with an FPT-approximation algorithm and a grid structure," *Pervasive and Mobile Computing*, vol. 58, p. 101033, Jun. 2019.

[18] S. Randhawa and S. Jain, "Performance analysis of LEACH with machine learning algorithms in wireless sensor networks," *International Journal of Computer Applications*, vol. 147, pp. 7–12, Aug. 2016.

[19] A. Mahboub *et al.*, "Energy-efficient hybrid k-means algorithm for clustered wireless sensor networks," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 2054–2060, Aug. 2017.

[20] A. Ray and D. De, "Energy efficient clustering protocol based on k-means (EECPK-means)-midpoint algorithm for enhanced network lifetime in wireless sensor network," *IET Wireless Sensor Systems*, vol. 6, pp. 181–191, Dec. 2016.

[21] N. T. Tam, D. T. Hai, L. H. Son, and L. T. Vinh, "Improving lifetime and network connections of 3D wireless sensor networks based on fuzzy clustering and particle swarm optimization," *Wireless Networks*, vol. 24, pp. 1477–1490, Jul. 2018.

[22] M. Bidaki, R. Ghaemi, S. Reza, and K. Tabbakh, "Towards energy efficient k-means based clustering scheme for wireless sensor networks," *International Journal of Grid and Distributed Computing*, vol. 9, pp. 265–276, Jul. 2016.

[23] A. Sasikala *et al.*, "Improving the energy efficiency of LEACH protocol using VCH in wireless sensor network," *International Journal of Engineering Development and Research*, vol. 3, pp. 918–924, May 2015.

[24] E. Rabiaa, B. Noura, and C. Adnene, "Improvements in LEACH based on k-means and gauss algorithms," *Procedia Computer Science*, vol. 73, pp. 460–467, Dec. 2015.

[25] Y. Zhang, M. Liu, and Q. Liu, "An energy-balanced clustering protocol based on an improved CFSFDP algorithm for wireless sensor networks," *Sensors*, vol. 18, pp. 1–18, Mar. 2018.

[26] A. A. Khuwaja, Y. Chen, N. Zhao, M. S. Alouini, and P. Dobbins, "A survey of channel modeling for UAV communications," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2804–2821, 2018.

[27] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, “Energy-efficient data collection and device positioning in UAV-assisted IoT,” *IEEE Internet of Things Journal*, vol. 7, pp. 1122–1139, Feb. 2020.

[28] B. Liu and H. Zhu, “Energy-effective data gathering for UAV-aided wireless sensor networks,” *Sensors*, vol. 19, pp. 1–12, May 2019.

[29] J. Baek, S. I. Han, and Y. Han, “Energy-efficient UAV routing for wireless sensor networks,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 1741–1750, Feb. 2020.

[30] C. Zhan, Y. Zeng, and R. Zhang, “Energy-efficient data collection in UAV enabled wireless sensor network,” *IEEE Wireless Communications Letters*, vol. 7, pp. 328–331, Jun. 2018.

[31] C. Zhan and H. Lai, “Energy minimization in internet-of-things system based on rotary-wing UAV,” *IEEE Wireless Communications Letters*, vol. 8, pp. 1341–1344, Oct. 2019.

[32] Q. Song, S. Jin, and F. Zheng, “Completion time and energy consumption minimization for UAV-enabled multicasting,” *IEEE Wireless Communications Letters*, vol. 8, pp. 821–824, Jun. 2019.

[33] Y. Zeng, J. Xu, and R. Zhang, “Energy minimization for wireless communication with rotary-wing UAV,” *IEEE Transactions on Wireless Communications*, vol. 18, pp. 2329–2345, Apr. 2019.

[34] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, “Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach,” *IEEE Journal on Selected Areas in Communications*, Sep. 2018.

[35] D. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, “Coarse trajectory design for energy minimization in UAV-enabled,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 9483–9496, Sep. 2020.

[36] M. B. Ghorbel, D. Rodríguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, "Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2165–2175, Mar. 2019.

[37] D. Yang, Q. Wu, Y. Zeng, and R. Zhang, "Energy tradeoff in ground-to-UAV communication via trajectory design," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 6721–6726, Jul. 2018.

[38] C. Zhan and Y. Zeng, "Aerial–ground cost tradeoff for multi-UAV-enabled data collection in wireless sensor networks," *IEEE Transactions on Communications*, vol. 68, pp. 1937–1950, Mar. 2020.

[39] Z. Zhou *et al.*, "When mobile crowd sensing meets UAV: Energy-efficient task assignment and route planning," *IEEE Transactions on Communications*, vol. 66, pp. 5526–5538, Nov. 2018.

[40] Z. Zixuan, W. Qinhao, Z. Bo, Y. Xiaodong, and T. Yuhua, "UAV flight strategy algorithm based on dynamic programming," *Journal of Systems Engineering and Electronics*, vol. 29, pp. 1293–1299, Dec. 2018.

[41] A. A. Al-Habob, O. A. Dobre, S. Muhaidat, and H. Vincent Poor, "Energy-efficient data dissemination using a UAV: An ant colony approach," *IEEE Wireless Communications Letters*, vol. 10, pp. 16–20, Jan. 2021.

[42] K. Zhu, X. Xu, and S. Han, "Energy-efficient UAV trajectory planning for data collection and computation in mMTC networks," in *Proc. IEEE Globecom Workshops*, pp. 1–6, Dec. 2018.

[43] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Actor-critic deep reinforcement learning for energy minimization in UAV-aided networks," in *Proc. European Conference on Networks and Communications (EuCNC)*, pp. 348–352, Jun. 2020.

[44] R. Ding, F. Gao, and X. S. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning

approach," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 7796–7809, Dec. 2020.

[45] B. Zhang, C. H. Liu, J. Tang, Z. Xu, J. Ma, and W. Wang, "Learning-based energy-efficient data collection by unmanned vehicles in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1666–1676, Apr. 2018.

[46] W. Liu, P. Si, E. Sun, M. Li, C. Fang, and Y. Zhang, "Green mobility management in UAV-assisted IoT based on dueling DQN," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2019.

[47] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 5994–6006, Dec. 2021.

[48] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten, "Energy minimization in UAV-aided networks: Actor-critic learning for constrained scheduling optimization," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 5028–5042, Apr. 2021.

[49] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1226–1252, 2021.

[50] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, pp. 3826–3839, Sep. 2020.

[51] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. International Conference on Neural Information Processing Systems (NIPS)*, pp. 3104–3112, Dec. 2014.

[52] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for

combinatorial optimization: A survey," *Computers Operations Research*, p. 105400, Dec. 2021.

[53] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, pp. 405–421, Apr. 2021.

[54] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 28, pp. 1–9, Dec. 2015.

[55] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR) Workshop*, pp. 1–5, Apr. 2017.

[56] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 3806–3817, Oct. 2019.

[57] Z. Li, J. Wu, L. Sun, and T. Rong, "Combinatorial keyword recommendations for sponsored search with deep reinforcement learning," *arXiv preprint arXiv:1907.08686*, 2019.

[58] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. International Conference on Neural Information Processing Systems (NIPS)*, pp. 9861–9871, Dec. 2018.

[59] J. Lu, L. Feng, J. Yang, M. M. Hassan, A. Alelaiwi, and I. Humar, "Artificial agent: The fusion of artificial intelligence and a mobile agent for energy-efficient traffic control in wireless sensor networks," *Future Generation Computer Systems*, vol. 95, pp. 45–51, Jun. 2019.

[60] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 51, pp. 3103–3114, Jun. 2020.

[61] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 292–303, Feb. 2020.

[62] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys and Tutorials*, vol. 21, pp. 2334–2360, Mar. 2019.

[63] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for UAV-assisted data collection," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 553–558, Apr. 2018.

[64] J. Liu, P. Tong, X. Wang, B. Bai, and H. Dai, "UAV-aided data collection for information freshness in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 2368–2382, Apr. 2021.

[65] Z. Jia, X. Qin, Z. Wang, and B. Liu, "Age-based path planning and data acquisition in uav-assisted iot networks," in *Proc. IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, May 2019.

[66] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in UAV-assisted IoT networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2003–2008, Feb. 2019.

[67] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 716–721, Jul. 2020.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 6000–6010, Dec. 2017.

[69] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in *Proc. Integration of Constraint Program-*

ming, Artificial Intelligence, and Operations Research (CPAIOR), pp. 170–181, Jun. 2018.

[70] W. Kool, H. van Hoof, and M. Welling, “Attention, learn to solve routing problems!,” in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–25, May 2019.

[71] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, “Learning improvement heuristics for solving routing problems,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, Apr. 2021.

[72] X. Bresson and T. Laurent, “The transformer network for the traveling salesman problem,” *arXiv preprint arXiv:2103.03012*, 2021.

[73] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, “Age of information: An introduction and survey,” *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 1183–1210, May 2021.

[74] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, “On the role of age of information in the internet of things,” *IEEE Communications Magazine*, vol. 57, pp. 72–77, Dec. 2019.

2. Improved Soft-k-Means Clustering Algorithm for Balancing Energy Consumption in Wireless Sensor Networks

Published as:

B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton and J. Henry, "Improved soft-k-means clustering algorithm for balancing energy consumption in wireless sensor networks," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4868–4881, Mar. 2021.

The manuscript included in this chapter studies the problem of balancing the energy consumption of WSNs in order to extend the networks' lifetime. To achieve this goal, an improved machine learning-based clustering algorithm is proposed. The proposed algorithm improves the selection of initial cluster centers and reassigns nodes at the edge of different clusters to a low-density cluster according to the nodes' membership probabilities to balance the number of nodes per cluster. Furthermore, multi-CHs scheme was used in the selection of final CHs to balance the traffic load of CHs.

Improved Soft-k-Means Clustering Algorithm for Balancing Energy Consumption in Wireless Sensor Networks

Botao Zhu, Ebrahim Bedeer, Ha Nguyen,
Robert Barton, and Jerome Henry

Abstract

Energy load balancing is an essential issue in designing wireless sensor networks (WSNs). Clustering techniques are utilized as energy-efficient methods to balance the network energy and prolong its lifetime. In this paper, we propose an improved soft-k-means (IS-k-means) clustering algorithm to balance the energy consumption of nodes in WSNs. First, we use the idea of "clustering by fast search and find of density peaks" (CFSFDP) and kernel density estimation (KDE) to improve the selection of the initial cluster centers of the soft k-means clustering algorithm. Then, we utilize the flexibility of the soft-k-means and reassign member nodes considering their membership probabilities at the boundary of clusters to balance the number of nodes per cluster. Furthermore, the concept of multi-cluster heads is employed to balance the energy consumption within clusters. Extensive simulation results under different network scenarios demonstrate that for small-scale WSNs with single-hop transmission, the proposed algorithm can postpone the first node death, the half of nodes death, and the last node death on average when compared to various clustering algorithms from the literature.

Index Terms

Clustering by fast search and find of density peaks (CFSFDP), energy load balancing, kernel density estimation (KDE), multi-cluster heads, soft k-means, wireless sensor networks (WSNs).

2.1 Introduction

The general concept of Internet-of-Things (IoT) is to facilitate the network connection of billions of devices to collect and exchange information to provide various services [1, 2]. Wireless sensor networks (WSNs) are among important parts of an IoT system because they can be used to gather and send data [3]. WSNs are like the eyes and ears of the IoT and they build the bridge between the real and the digital worlds. WSNs typically consist of a large number of low-cost sensor nodes with restricted battery supplies. Sensor nodes are deployed in various application scenarios to monitor and collect physical conditions of the surrounding environment such as temperature, humidity, pressure, position, vibration, and sound, to name a few [4]. The collected data is then sent to the base station (BS) for further analysis and processing.

Reducing energy consumption is a key challenge in WSNs, as sensor nodes can be placed in hard-to-reach areas and/or their batteries may not be rechargeable [5]. Clustering in energy-limited WSNs has been widely investigated to reduce the energy consumption [6]. Clustering-based algorithms group sensor nodes into distinct clusters, where each sensor node belongs to one cluster only. All member nodes sense their surrounding environment and send the results to the cluster heads (CHs). Then, CHs collect and process the data and send information to the BS [7]. Each node consumes a certain amount of energy when it collects, processes, and sends data, and a node is defined to be dead when it runs out of energy [8]. Hence, it is crucial to develop efficient clustering algorithms to balance the energy consumption among sensor nodes in WSNs.

Different clustering techniques have been proposed to design energy-efficient WSNs and increase their lifetime. The authors in [9] proposed a CH election method, which rotates the CH positions among the nodes with higher energy in different communication rounds. In particular, the method considers the initial energy, residual energy, and an optimal number of CHs to decide the next group of CHs among the nodes in the network. Then, member nodes join different CHs according to the distances between them and CHs to form clusters. A joint clustering and routing algorithm is proposed in [10] to improve the energy efficiency of large-scale WSNs. This algorithm employs a back-off timer and gradient routing to execute the CH selection and multi-hop routing simultaneously. The authors in [11] presented a node-density-based clustering and mobile elements algorithm (NDCMC) for collecting data in WSNs. In NDCMC, the nodes surrounded by more deployed nodes are selected as CHs in order to improve the efficiency of intra-cluster routing. The authors presented a fixed parameter tractable (FPT) approximation algorithm with an approximation factor of 1.2 based on the parameterized complexity theory in [12] in order to solve load balanced clustering problem (LBCP) in WSNs. The FPT-approximation algorithm determines which gateway each sensor node must be assigned to, which can lead to more balanced load and energy consumption among the gateways. On the other hand, a routing tree for the inter-cluster communication is proposed, which can distribute the overhead of the routing among almost of all of the nodes. In [13], the authors further proposed an FPT-approximation algorithm with an approximation factor of 1.1, which is more precise than previous approximation factors reported for LBCP. The FPT-approximation algorithm is used to assign sensor nodes to gateways such that the maximum load of the gateways is minimized. Then, an energy-aware routing algorithm is employed to find the optimal routing tree between gateways and the sink with the aim of balancing the energy consumption of the nodes. The same authors also considered another FPT-approximation algorithm with an approximation factor of 1.1 in [14]. In order to make the FPT-approximation algorithm to be practical in large-scale WSNs, a virtual grid infrastructure with several equal-size cells is used where the FPT-approximation algorithm runs in each cell independently. In [15], a distributed multi-objective based clustering algorithm is presented to assign sensor nodes to appropriate CHs. Then, an energy-efficient routing algorithm is proposed to balance the relay load among the CHs. In [16], the authors implemented a distributed clustering algorithm by considering a trade-off between the energy efficiency and coverage requirement. This algorithm can form unequal-size clusters to balance the load of the CHs. The same authors in [17] proposed a distributed fuzzy logic-based unequal clustering approach and routing algorithm (DFCR) to solve the hot spot problem, which is caused by the fact that some CHs deplete their energy much faster as compared to other CHs. The DFCR algorithm designs an unequal clustering mechanism by reducing the cluster size nearest to the BS.

The authors in [18] proposed a modified k-means clustering algorithm that considers two factors, namely, (i) distances among CHs and their member nodes, and (ii) the remaining energy of nodes, to reduce the overall energy consumption and extend the network lifespan. In [19], the authors proposed a hybrid clustering algorithm based on the k-means clustering algorithm and LEACH [20], where balanced clusters are generated by k-means and CHs are selected by LEACH. This hybrid algorithm outperforms LEACH in terms of the energy consumption. However, due to the frequent re-clustering, the energy consumption of the nodes may increase in the phase of cluster formation and CH selection. An energy efficient clustering protocol based on k-means (EECPK-means) is proposed in [21] with the aim of balancing the load of CHs in WSNs. The midpoint method is used to improve the initial selection of centroids in the k-means algorithm in order to generate balanced clusters. In [22], the authors proposed a method based on fuzzy c-means clustering and particle swarm optimization (FCM-PSO) to reduce the total energy consumption of the network and reduce the number of network disconnects. The FCM-PSO algorithm considers the energy consumption and constraints of communication in the calculations of the CHs and nodes' membership probability. The energy-efficient k-means LEACH (KM-LEACH) algorithm is proposed in [23] to create symmetric clusters and reduce the average intra-cluster communication distance, which can save nodes' energy and improve the network lifetime. To address the problem of how to control the failure of a CH in each cluster, the k-medoids clustering algorithm and vice CH scheme (VLEACH) are used together with LEACH in [24]. Vice CH will become a new CH in case the CH of a given cluster dies, which helps to prolong the lifetime of WSNs by balancing the nodes' energy consumption. The authors in [25] used the k-means and Gaussian elimination algorithms to reduce energy consumption of WSNs and extend their lifetime. An innovative classification algorithm based on "clustering by fast search and finding of density peaks" (CFSFDP) [26] algorithm for balancing energy is proposed in [27]. The authors extend the original CFSFDP algorithm to take into account residual energy (in addition to local density and distance) to select CHs, and accordingly cluster nodes based on the selected CHs.

Against the above background, in this paper, an improved soft-k-means (IS-k-means) clustering algorithm is proposed with the aim of balancing the energy consumption of all nodes in WSNs and extending the network lifetime. The proposed IS-k-means can be widely used in industrial control, smart home, smart agriculture, environment perception, health monitoring, etc., because it can extend the life of sensor nodes in these application scenarios. The novelty of the proposed algorithm can be summarized as follows.

1) Compared with existing clustering algorithms that select the initial cluster centers randomly, we choose the initial centroids of the IS-k-means clustering algorithm by using the idea of density from CFSFDP and kernel density estimation (KDE) [28] to achieve a better clustering result. The nodes with high local density and relative large node distances are chosen as the initial centroids.

2) After the proposed algorithm converges, we reassign member nodes that are located at the boundary of two or more clusters to balance the number of nodes per cluster according to the flexibility of the soft-k-means.

3) Since the clustering process needs to be repeated continually, the communication cost during the clustering phase is increased. We use multi-cluster heads (multi-CHs) scheme to balance traffic load of CHs of different clusters and reduce the frequency of clustering.

The rest of this paper is organized as follows. The necessary background for our research is discussed in Section 2.2. Section 2.3 describes the proposed IS-k-means algorithm. In Section 2.4, we compare the performance of the proposed IS-k-means with other algorithms. Finally, Section 2.5 concludes the paper.

2.2 Preliminaries

2.2.1 Soft k-Means

The soft k-means [29] is a kind of fuzzy clustering algorithm where clusters are represented by their respective centers. Since traditional k-means clustering techniques are hard clustering algorithms, which may fail to separate overlapping clusters or properly cluster noisy data [30], the soft k-means algorithm can be applied to address these cases. With the soft k-means algorithm, each node may belong to one or more clusters with different degrees of membership [31]. Nodes located at the boundaries of clusters are not forced to fully belong to a given cluster, but rather they can be members of many clusters with membership degrees or probabilities between 0 and 1 [32]. Nodes at the edge of a cluster may have lower membership probabilities than nodes close to the center of a cluster. This flexibility of the soft k-means clustering is in sharp contrast with the k-means clustering, where a node belongs to only a single cluster.

For a set of nodes' locations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ in WSNs, the goal of the soft k-means is to partition the n nodes into k sets $\boldsymbol{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_k\}$ with small intra-cluster distances and large inter-cluster distances. Thus, we define the following cost function:

$$J(\boldsymbol{X}; \boldsymbol{Z}, \boldsymbol{M}) = \sum_{v=1}^{k} \sum_{j=1}^{n} z_{vj} ||\boldsymbol{x}_j - \boldsymbol{\mu}_v||^2, \tag{2.1}$$

where $\boldsymbol{M}(\boldsymbol{\mu}_v; v = 1, \ldots, k)$ is the matrix of cluster centers, and $\boldsymbol{Z}(z_{vj}; v = 1, \ldots, k; j = 1, \ldots, n)$ is the membership probability matrix of $\boldsymbol{X}$. z_{vj} is the membership value of the jth node to the vth cluster and is defined as [29]

$$z_{vj} = \frac{e^{-\beta ||\boldsymbol{x}_j - \boldsymbol{\mu}_v||^2}}{\sum_{l=1}^{k} e^{-\beta ||\boldsymbol{x}_j - \boldsymbol{\mu}_l||^2}}, \tag{2.2}$$

where β is the stiffness parameter that impacts the membership probability of each node. The best clustering solution is obtained by minimizing J, which differs from the conventional k-means since weighted squared errors are used in the cost function instead of squared errors [29]. The result of the soft k-means algorithm will depend on the choice of β. We will discuss the choice of β when presenting simulation results.

In order to minimize the objective function in (2.1), z_{vj} must satisfy the following three constraints [29].

1. Each node is assigned a membership probability between 0 and 1 for belonging to a cluster:

$$z_{vj} \in [0, 1], \; v = 1, \ldots, k, \; j = 1, \ldots, n. \tag{2.3}$$

2. The sum of the membership probabilities for one node over all clusters is equal to 1:

$$\sum_{v=1}^{k} z_{vj} = 1,\ \ j = 1, \ldots, n. \tag{2.4}$$

3. There will be at least one node with some non-zero membership probability for belonging to each cluster

$$\sum_{j=1}^{n} z_{vj} > 0,\ \ v = 1, \ldots, k. \tag{2.5}$$

By minimizing the objective function, we can calculate the cluster centers as [29]

$$\boldsymbol{\mu}_v = \frac{\sum_{j=1}^{n} z_{vj}\boldsymbol{x}_j}{\sum_{j=1}^{n} z_{vj}}. \tag{2.6}$$

The operations of the soft k-means algorithm can be summarized as follows: the algorithm calculates the membership probabilities and the cluster centers according to (2.2) and (2.6) in each round, respectively. If the changes of the membership probabilities $\boldsymbol{Z}$ or the cluster centers $\boldsymbol{M}$ are below given thresholds, the clustering process ends. Otherwise, the algorithm recalculates the new membership probabilities $\boldsymbol{Z}$ and the new cluster centers $\boldsymbol{M}$. If the algorithm does not converge after a given number of iterations, it will re-initiate by choosing new initial cluster centers. Fig. 2.1 shows an example of the clustering result of 100 nodes by the soft k-means algorithm.

2.2.2 Kernel Density Estimation

Non-parametric estimators are flexible for modeling probability density function (PDF) of data points. They have no fixed functional form and depend on data points to reach an estimate when compared to parametric estimators [33]. Non-parametric estimators can be classified into histogram-based and kernel-based estimation. A histogram-based estimator needs large data sets to guarantee convergence, and it cannot produce smooth continuous estimation curve [34]. KDE finds the distribution characteristics from data points without attaching any assumptions to data. It can ensure a smooth PDF approximation for given data points [28]. In KDE, the kernel function is centered at each data point, and it has the

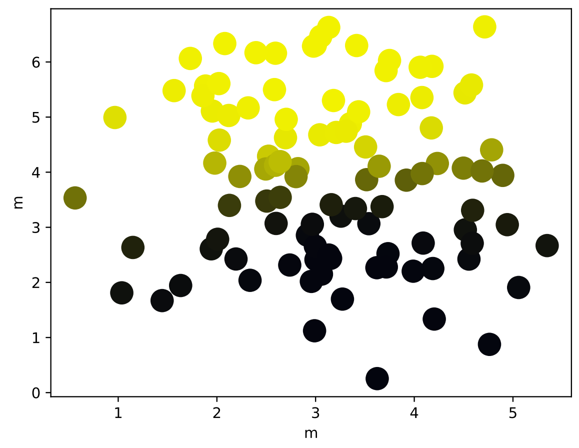


Figure 2.1 Example of soft k-means clustering.

peak value at the data point location while decreasing in intensity with the distance from this location [28].

Using KDE, the PDF of the nodes' locations $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \in \mathbb{R}^d$ is represented by a weighted sum of the kernel functions [35]

$$\hat{f}_h(\boldsymbol{x}_i) = \frac{1}{nh^d} \sum_{t=1}^{n} \mathcal{K}\left(\frac{\boldsymbol{x}_t - \boldsymbol{x}_i}{h}\right), \tag{2.7}$$

where h is the smoothing parameter called the bandwidth and it controls the size of the neighborhood around $\boldsymbol{x}_i, i \in 1, \dots, n$. $\mathcal{K}(\cdot)$ is called the kernel function, which is defined in a d-dimensional space. The kernel function controls the weight given to $\boldsymbol{X}$ at each point $\boldsymbol{x}_i$ based on their proximity. To yield meaningful estimates, a kernel function should satisfy the following conditions [28].

1. Normalization:

$$\int_{\mathbb{R}^d} \mathcal{K}(\boldsymbol{u}) d\boldsymbol{u} = 1. \tag{2.8}$$

2. Symmetry:

$$\mathcal{K}(-\boldsymbol{u}) = \mathcal{K}(\boldsymbol{u}). \tag{2.9}$$

3. Non-negative and real-valued integrable:

$$\mathcal{K}(\boldsymbol{u}) > 0. \tag{2.10}$$

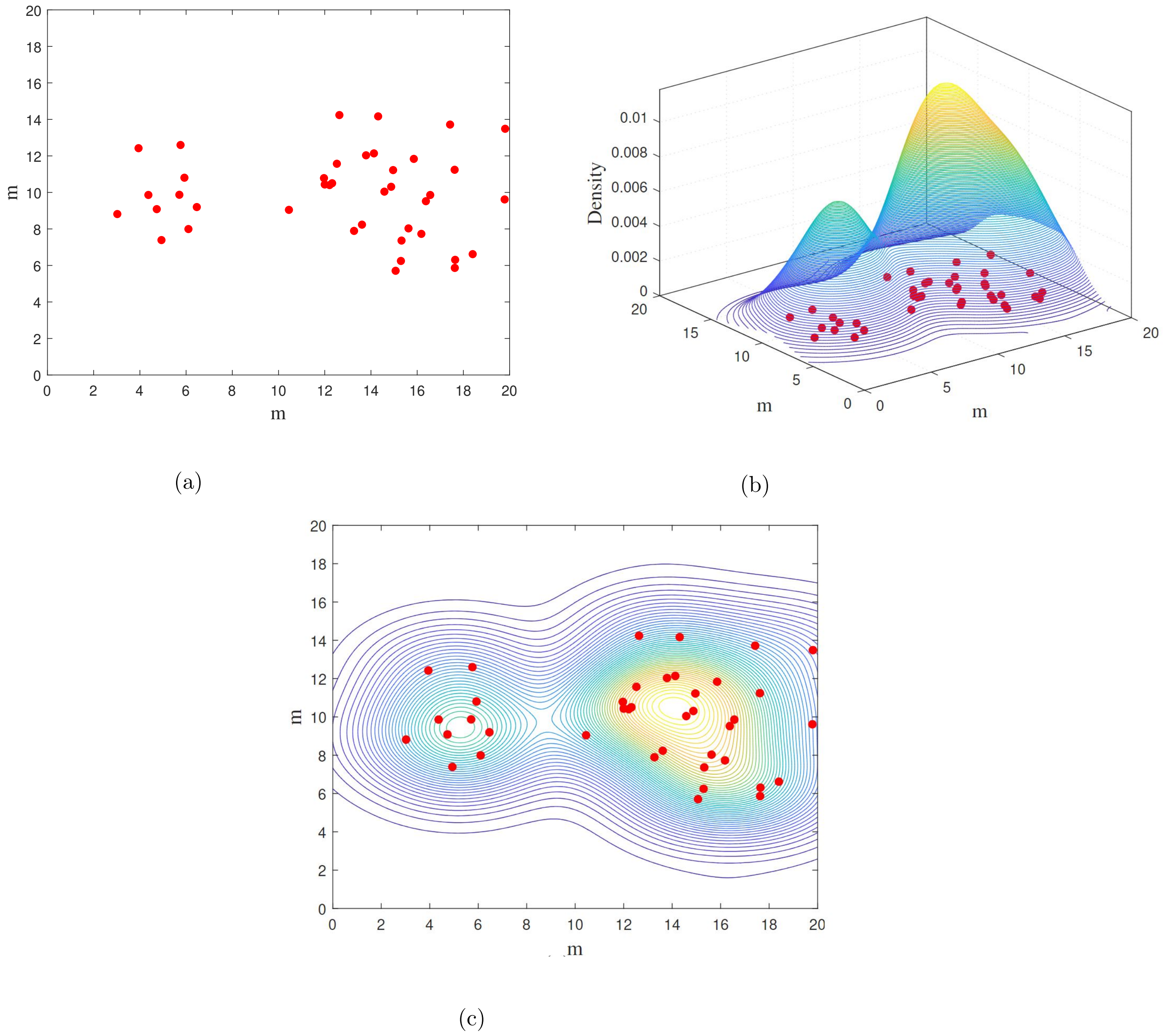


Figure 2.2 An example of KDE. (a) Nodes distribution. (b) 3-dimensional density contour of nodes in (a). (c) 2-dimensional density contour of nodes in (a).

A multivariate kernel function can be seen as a product of symmetric univariate kernel functions [36]

$$\mathcal{K}(\boldsymbol{u}) = \prod_{j=1}^{d} \phi(u_j), \tag{2.11}$$

where u_j is the jth component of the d-dimensional vector $\boldsymbol{u}$, and $\phi(\cdot)$ is a univariate kernel function. In our proposed algorithm, we use the Gaussian kernel function due to its well-known properties [37], which is defined as follows:

$$\phi(u_j) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u_j^2}{2}\right). \tag{2.12}$$

Fig. 2.2 is an example of KDE for a set of data. The set of discrete points is transformed into a smooth density map, as shown in Fig. 2.2 (b), which displays its spatial distribution. The higher the PDF value in a location is, the higher the density is.

2.2.3 "Clustering by Fast Search and Find of Density Peaks" Algorithm

CFSFDP is a new clustering algorithm proposed by Rodriguez and Laio [26]. It is based on the assumptions that cluster centers are surrounded by lower local density neighbors and they are at a relatively large distance from any nodes with a higher local density. This method needs to calculate two quantities for each node i: local density ρ_i and distance δ_i. The cluster centers are the nodes with higher local density and larger distance. For a set of nodes' locations $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, and nodes' label set $\boldsymbol{I} = \{1, \ldots, n\}$, the local density of a node $\boldsymbol{x}_i$ is defined as

$$\rho_i = \sum_{i \neq j} \chi(d_{ij} - d_{\mathrm{c}}), \tag{2.13}$$

where

$$\chi(\alpha) = \begin{cases} 1, & \alpha < 0, \\ 0, & \alpha \geq 0, \end{cases} \tag{2.14}$$

d_{ij} is the distance between nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, and d_{c} is the cutoff distance. The choice of d_{c} should yield an average number of neighbors around 1 to 2% of the total number of nodes. In essence, ρ_i can be seen as the number of nodes that are neighbor to node $\boldsymbol{x}_i$ in the range of d_{c}.

Two cases need to be considered in calculating a node's distance. If node i has the highest density, then its distance δ_i is defined as the distance d_{ij} between node i and node j which is furthest from node i in $\boldsymbol{I}$. Otherwise, the distance of node i is defined as the distance d_{ij} between node i and its nearest neighbor node j having a higher density [38]. Specifically, the distance δ_i is expressed as

$$\delta_i = \begin{cases} \max(d_{ij})_{j \in \boldsymbol{I}}, & \text{if } \rho_i \text{ is maximum,} \\ \min(d_{ij})_{j \in \boldsymbol{I}^{(i)}}, & \text{otherwise,} \end{cases} \tag{2.15}$$

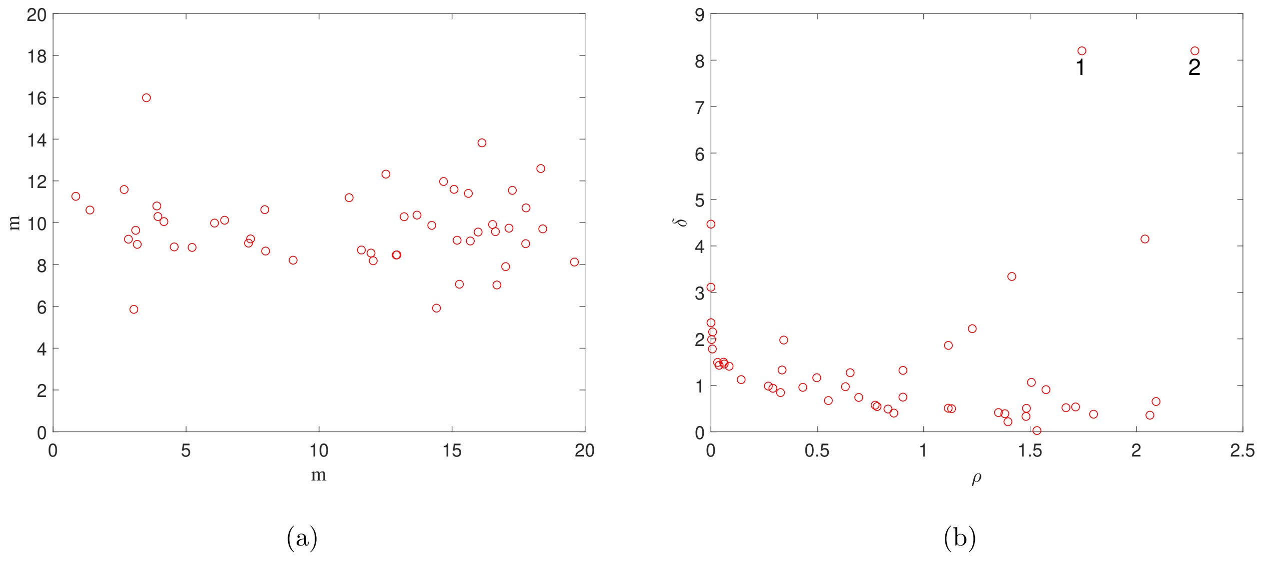
$$\boldsymbol{I}^{(i)} = \{t \in \boldsymbol{I} : \rho_t > \rho_i\}, \tag{2.16}$$

where $\boldsymbol{I}^{(i)}$ is the nodes' label set with node densities greater than ρ_i. After these two quantities are calculated, the cluster centers are selected from nodes with high values of both ρ_i and δ_i. Then, the CFSFDP algorithm assigns other remaining points to the nearest cluster center to form clusters. Specifically, if ρ_i is large and δ_i is small for node i, it means node i is close to the cluster center but not the center. On the other hand, if node i has small ρ_i and large δ_i, it implies that the node is away from the cluster center [39].

Fig. 2.3 (b) shows the plot of δ_i as a function of ρ_i for each node in Fig. 2.3 (a). This representation is called the decision graph. According to the decision graph, we can have two nodes with higher values of both density ρ and distance δ. Hence, they can be chosen as cluster centers, as shown in Fig. 2.3 (c).

2.3 Proposed IS-k-Means Algorithm

The proposed IS-k-means algorithm involves two phases: (i) set-up phase, and (ii) steady phases. During the set-up phase, each node broadcasts a HELLO message including its ID and location within the range of its coverage so that each node can acquire information of its neighbor nodes. Next, each node sends its information to the BS by the geographic multi-hop routing algorithm [11] because it already knows the positions of its neighbor nodes. The BS runs the proposed IS-k-means algorithm according to the information received from all nodes. The proposed algorithm uses CFSFDP and KDE algorithms to optimize the selection of initial cluster centers of the soft k-means clustering method. Then, the soft k-means is used to form clusters and node reassigning scheme is employed to balance the numbers of nodes in different clusters. In order to balance the energy overhead of CHs, the multi-CHs scheme is utilized. After formulation of clusters and selection of CHs are completed, the BS broadcasts the results to all nodes by the restricted flooding method [11]. Thus, each node



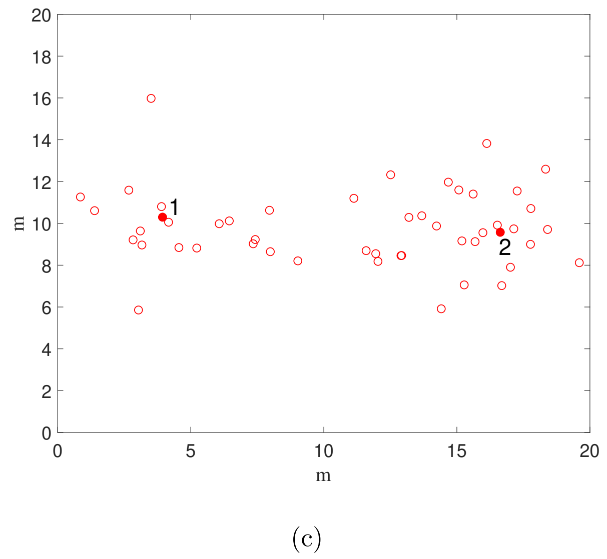


Figure 2.3 CFSFDP in two dimensions. (a) Nodes distribution. (b) Decision graph for nodes in (a): X-coordinate is local density ρ, and Y-coordinate is δ. (c) Two center nodes are determined.

can identify its role, e.g., CH or member node, and choose to join a corresponding CH if it is a member node. The steady phase is composed of many communication rounds. In each round r, member nodes collect and transmit data to CHs in their allotted time slots, and CHs aggregate the data and send it to the BS. When the energy of a CH is less then a threshold, it will broadcast a SWITCH message to activate the next candidate CH in the same cluster as the new CH and inform member nodes to send data to this new CH. If all CHs in a certain cluster are enabled sequentially, the last working CH will send a RESTART message to the BS to trigger re-clustering. The flowchart of the proposed IS-k-means algorithm is shown in Fig. 2.4.

2.3.1 Energy Model

The first-order radio model [20] is used to calculate the energy consumption of the network. The transmitter's energy consumption involves the transmitter circuitry and the power amplifier, while the energy consumption of the receiver accounts for the receiver circuitry. The free space and the multipath fading models are used in the transmitter power amplifier. If the distance between the transmitter and the receiver is less than a threshold, the power amplifier uses the free space model; otherwise, the multipath model is used [40]. The energy consumption of the transmitter and the receiver for transmitting an l-bit message can be calculated as follows [20]

$$E_{\mathrm{T}} = \begin{cases} lE_{\mathrm{elec}} + l\varepsilon_{\mathrm{fs}}d^2, & d \leq d_0, \\ lE_{\mathrm{elec}} + l\varepsilon_{\mathrm{mp}}d^4, & d > d_0, \end{cases} \tag{2.17}$$

$$E_{\mathrm{R}} = lE_{\mathrm{elec}}, \tag{2.18}$$

$$d_0 = \sqrt{\frac{\varepsilon_{\mathrm{fs}}}{\varepsilon_{\mathrm{mp}}}}, \tag{2.19}$$

where E_{T} is the dissipated energy in the transmitter and E_{R} is the dissipated energy in the receiver. E_{elec} is the dissipated energy per bit in both the transmitter circuitry and the receiver circuitry. d is the transmission distance between the transmitter and the receiver. d_0 is the distance threshold. $\varepsilon_{\mathrm{fs}}$ and $\varepsilon_{\mathrm{mp}}$ represent the radio amplifier energy parameter of the free space and multipath fading models [11], respectively.

Because there are many rounds within the steady phase, the energy consumption of a CH in round r can be calculated as

$$E_{\mathrm{CH}}(r) = gcE_{\mathrm{T}} + g(clE_{\mathrm{DA}} + E_{\mathrm{R}}), \tag{2.20}$$

where E_{DA} represents the dissipated energy of data aggregation and c is the data aggregation ratio. The first term of the right hand side of (2.20) is the energy consumption of a CH

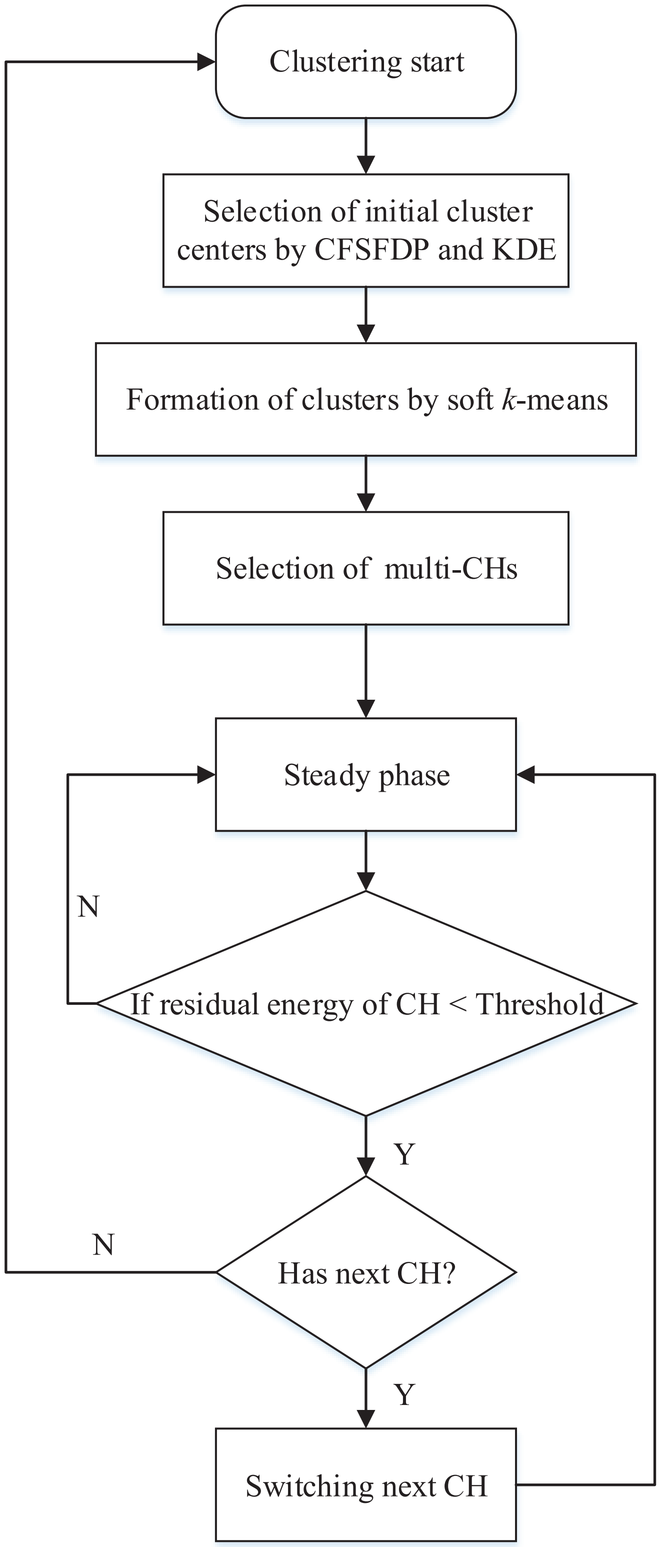

Clustering start
Selection of initial cluster centers by CFSFDP and KDE
Formation of clusters by soft k-means
Selection of multi-CHs
Steady phase
N
If residual energy of CH < Threshold
Y
N
Has next CH?
Y
Switching next CH

Figure 2.4 Flowchart of the proposed algorithm.

for sending aggregated data to the BS and the second term is the energy consumption of receiving and aggregating data of g member nodes. The energy consumption of a member node sending data to its CH in round r is

$$E_{\mathrm{nonCH}}(r) = E_{\mathrm{T}}. \tag{2.21}$$

Hence, the residual energy of node i in round r can be computed by

$$E_i\left(r\right) = \begin{cases} E_i\left(r-1\right) - E_{\mathrm{CH}}(r), & i \in \mathrm{CHs}, \\ \\ E_i\left(r-1\right) - E_{\mathrm{nonCH}}(r), & i \notin \mathrm{CHs}, \end{cases} \tag{2.22}$$

where $E_i\left(r-1\right)$ is the residual energy of node i in the $r-1$ round.

2.3.2 Selection of Initial Cluster Centers

We use CFSFDP and KDE algorithms to determine the initial cluster centers as the input to the soft k-means clustering algorithm to produce a better clustering result. Because cluster centers are surrounded by neighbors with lower local density and they are at a relatively large distance from any points with a higher local density, they are selected by the maximum distance δ and relatively high local density ρ, which is illustrated in Fig. 2.3. First, we calculate the density of each node and find the nodes' set $\boldsymbol{X'}$ with relatively high density $\boldsymbol{\rho'}$. Then, the distances δ among nodes in $\boldsymbol{X'}$ are computed. In order to choose cluster centers, we only choose nodes with relatively high density, and then we multiply their density ρ_i and distance δ_i together as

$$\gamma_i = \rho_i \times \delta_i,\ i \in \{1, \ldots, m\}, \tag{2.23}$$

where m is the number of nodes with relatively high density. Since each initial cluster center node should have a high γ value, we choose nodes with relatively large γ value as the initial cluster centers. In addition, the value of k is equal to the number of the initial cluster centers. Algorithm 1 describes the detailed steps.

2.3.3 Cluster Formation

Some k-means-based algorithms form clusters according to the distances between normal nodes and CHs, such as distributed k-means clustering algorithm [41] and improved k-means

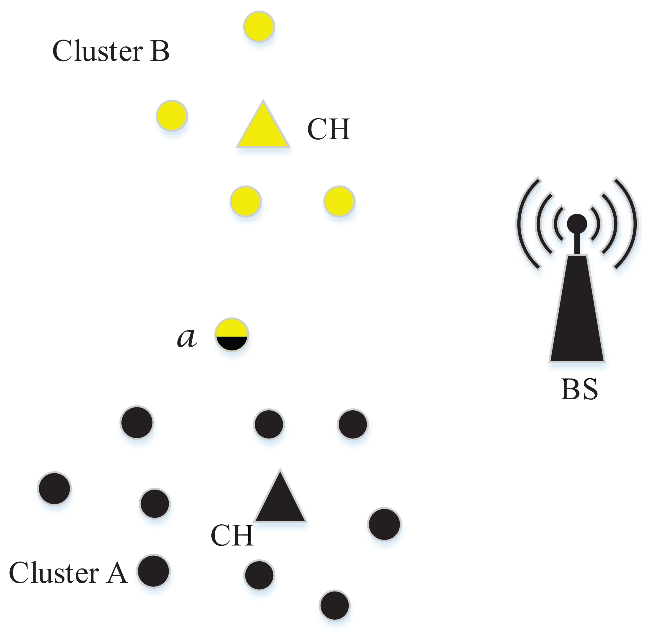


Figure 2.5 A node at the boundary of two clusters.

Algorithm 1 Selection of initial cluster centers

```
Input: $\boldsymbol{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$
Output: Initial cluster centers: $\boldsymbol{M}$
 1: for i = 1 : $n$ do
 2:     calculate $\rho_i$
 3: end for
 4: $\boldsymbol{\rho} = \{\rho_1, \dots, \rho_n\}$
 5: choose nodes with local maximum density $\boldsymbol{X}' = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ and get their density set
    $\boldsymbol{\rho}' = \{\rho_1, \dots, \rho_m\}, m < n$
 6: for i = 1 : $m$ do
 7:     calculate $\delta_i$
 8: end for
 9: $\boldsymbol{\Delta} = \{\delta_1, \dots, \delta_m\}$
10: calculate $\gamma_i$ by (2.23) to determine the initial cluster centers
11: return $\boldsymbol{M} = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_k\}$
```

cluster-based routing [42]. These k-means-based algorithms can easily lead to a large gap in the number of nodes in different clusters in WSNs and may cause unbalanced energy

Algorithm 2 Cluster formation

Input: $\boldsymbol{M} = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, the maximum number of iterations $r_{\max}$

Output: k clusters

```
 1: for $r = 1 : r_{\max}$ do
 2:     for $v = 1 : k$ do
 3:         for $j = 1 : n$ do
 4:             $z' = 0$
 5:             for $l = 1 : k$ do
 6:                 $z' = z' + e^{-\beta||\boldsymbol{x}_j - \boldsymbol{\mu}_l||^2}$
 7:             end for
 8:             $z_{vj} = \frac{e^{-\beta||\boldsymbol{x}_j - \boldsymbol{\mu}_v||^2}}{z'}$
 9:         end for
10:     end for
11:     $\boldsymbol{Z}_r = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ z_{k1} & \cdots & \cdots & z_{kn} \end{bmatrix}$
12:     for $v = 1 : k$ do
13:         $\boldsymbol{\mu}_v = \frac{\sum_{j=1}^{n} z_{vj}\boldsymbol{x}_j}{\sum_{j=1}^{n} z_{vj}}$
14:     end for
15: end for
16: final membership probabilities $\boldsymbol{Z}_{r_{\max}}$
17: for j $= 1 : n$ do
18:     assign node $j$ to cluster with the highest probability according to $\boldsymbol{Z}_{r_{\max}}$
19: end for
20: $k$ clusters $\boldsymbol{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_k\}$
21: for j $= 1 : n$ do
22:     reassign node $j$ located on the border to different cluster
23: end for
24: return $k$ new clusters $\boldsymbol{C}' = \{\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k\}$
```

consumption of CHs. Hence, compared with these k-means-based clustering algorithms, our proposed IS-k-means algorithm uses the soft k-means clustering algorithm to address this problem. Each node can be a member of more than one clusters at the same time according to membership probabilities in the soft k-means. However, member nodes need to join only one cluster with the highest membership probability at a time. Some boundary nodes may have similar probabilities to join multiple clusters. After the convergence of our proposed IS-k-means algorithm, we may reassign nodes to different clusters to balance the number of nodes per cluster. For example, node a is at the edge of two clusters and it has a higher probability to join cluster A, as illustrated in Fig. 2.5. Before reassigning node a, cluster A already has 10 member nodes and cluster B has 5 member nodes. Since all member nodes send messages to their CH, CH of cluster A will deal with more information from its member nodes. In order to balance the energy consumption of CHs, it is better to reassign node a to cluster B. Reassigning node a from cluster A to cluster B may increase slightly the energy consumption of transmitting messages between node a and its CH because the transmission distance d is increased. However, this slight increase of the transmission energy consumption is negligible as compared to the total energy consumption in CH. If the difference of the probabilities of a node belonging to two clusters is less than a certain threshold, it will join the cluster with low density. If a node is at the boundary of three or more clusters, the proposed algorithm only choose the first two maximum probabilities and follow the same rule. Algorithm 2 outlines the cluster formation algorithm.

2.3.4 Selection of Multi-CHs

Normally, the numbers of nodes in different clusters are different in WSNs. If only one CH is selected in each cluster, CH will consume too much energy to deal with the information from its member nodes in a high density cluster, which will cause its death too early. Hence, our proposed IS-k-means algorithm designs a scheme of multi-CHs. The number of CHs is not fixed in each cluster, and it is determined by the number of nodes per cluster. The larger the number of nodes in a cluster is, the higher the number of CHs will be. The remaining energy of nodes and distances between nodes and their cluster centers are considered in choosing CHs. Nodes close to their cluster center and having higher residual energy than the average

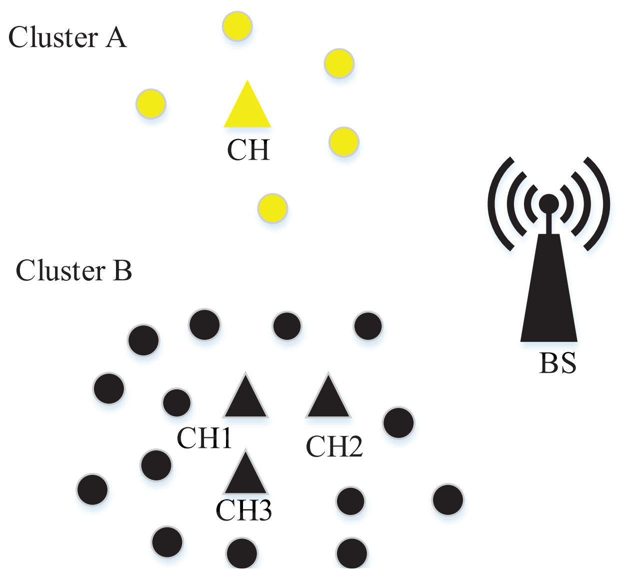


Figure 2.6 Multi-CHs scheme.

energy of the cluster can become CHs. We define a matrix $\mathbf{CHs} = \{\mathbf{CH}_1, \ldots, \mathbf{CH}_k\}$, which is composed of all CHs of k clusters, and $\mathbf{CH}_v, 1 \leq v \leq k$, represents the set of CHs of cluster v. The total remaining energy of cluster $v \in \{1, \ldots, k\}$ can be computed as

$$E_v = \sum_{i=1}^{S_v} E_i\left(r\right), \tag{2.24}$$

where S_v is the size of cluster v, $E_i\left(r\right)$ is the residual energy of node i in current round r, which can be obtained from (2.22).

The average energy of cluster v is calculated as

$$E_{\mathrm{ave}_v} = \frac{E_v}{S_v}. \tag{2.25}$$

As an example, since the number of nodes in cluster B is around 3 times that of cluster A in Fig. 2.6, cluster B will have three CHs if only one CH is selected in cluster A. After the number of CHs is determined, the nodes which have larger remaining energy and close to the cluster center are selected as CHs. This multi-CHs scheme can balance the energy consumption of CHs per cluster in WSNs, and is summarized in Algorithm 3.

After the set of CHs and clusters are determined, the first node in each $\mathbf{CH}_v$ is selected as the current CH in that cluster and the BS notifies all member nodes to join the cluster to

Algorithm 3 Selection of multi-CHs

```
Input: $\boldsymbol{C}' = \{\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_k\}$
Output: CHs
1: for $v = 1 : k$ do
2:     calculate the size of cluster $\boldsymbol{c}'_v$: $S_v$
3:     calculate average energy of cluster $E_{\text{ave}_v}$
4:     $p = \frac{S_v}{\text{constant}}$, the number of CHs of cluster $v$
5:     for i = 1 : $S_v$ do
6:         Iterate $\boldsymbol{x}_i$ from near the center of cluster
7:         if $E_i > E_{\text{ave}_v}$ and $p > 0$ then
8:             $p = p - 1$
9:             $\mathbf{CH}_v\,(p) = \boldsymbol{x}_i$
10:        end if
11:    end for
12: end for
13: return CHs = $\{\mathbf{CH}_1, \ldots, \mathbf{CH}_k\}$, CHs of $k$ clusters
```

which they belong. CHs broadcast time division multiple access schedules to their member nodes for transmitting data in different time slots to avoid data collision. Then, the network enters the steady phase and begins to exchange data between normal nodes and their CHs.

2.3.5 Switching to a Next CH

For balancing the energy consumption of CHs, if the energy consumption ratio of a current CH of any cluster is below a threshold value, the next candidate CH in that cluster is enabled. Until all CHs in a given cluster are executed, the algorithm starts re-clustering. The specific steps are described in Algorithm 4.

2.3.6 Complexity Analysis

The run-time complexity of the proposed IS-k-means algorithm mainly involves three phases. In the phase of selecting initial cluster centers, IS-k-means needs $O(n^2)$ operations

Algorithm 4 Switching to a next CH

```
Input: CHs of $k$ clusters, $\mathbf{CHs} = \{\mathbf{CH}_1, \dots, \mathbf{CH}_k\}$
Output: Next CH
 1: Current round
 2: for $v = 1 : k$ do
 3:     $T = \frac{\text{residual energy of } \mathbf{CH}_v(p) \text{ in current round}}{\text{residual energy of } \mathbf{CH}_v(p) \text{ in last round}}$
 4:     if $T <$ Threshold then
 5:         if $\mathbf{CH}_v$ has $\mathbf{CH}_v(p+1)$ then
 6:             switch to $\mathbf{CH}_v(p+1)$
 7:         else
 8:             re-clustering
 9:         end if
10:     end if
11: end for
```

[43] to execute CFSFDP and KDE to calculate the nodes' densities and distances where n is the number of nodes. Then, the algorithm requires $O(nk^2r_{\max})$ operations [44] to execute the soft k-means and $O(2n)$ operations to assign nodes to form final clusters. Because the selection of initial cluster centers has been optimized, the algorithm converges quickly and the value of $r_{\max}$ is very small. In the third phase, the algorithm needs $O(n)$ operations to select CHs. Thus, the overall time complexity of the proposed IS-k-means algorithm is $O(n^2 + nk^2r_{\max} + 3n)$ operations. Obviously, the time complexity of the IS-k-means depends mainly on the execution time of the first phase. The time complexity of the soft k-means algorithm is $O(nk^2r_{\max})$ operations [44], which is lower than that of our proposed IS-k-means algorithm. However, the higher complexity of the proposed IS-k-means algorithm can be well justified by its ability to better balance the energy consumption of nodes. As for the memory requirement, the proposed algorithm needs $O(n)$ memory units to store nodes first. Then, it costs $O(n)$ memory units [45] to store ρ and δ in the phase of selecting initial cluster centers. Then, $O(nk)$ memory units are required to store membership probabilities in the phase of cluster formation. Hence, the total storage requirement of the proposed algorithm

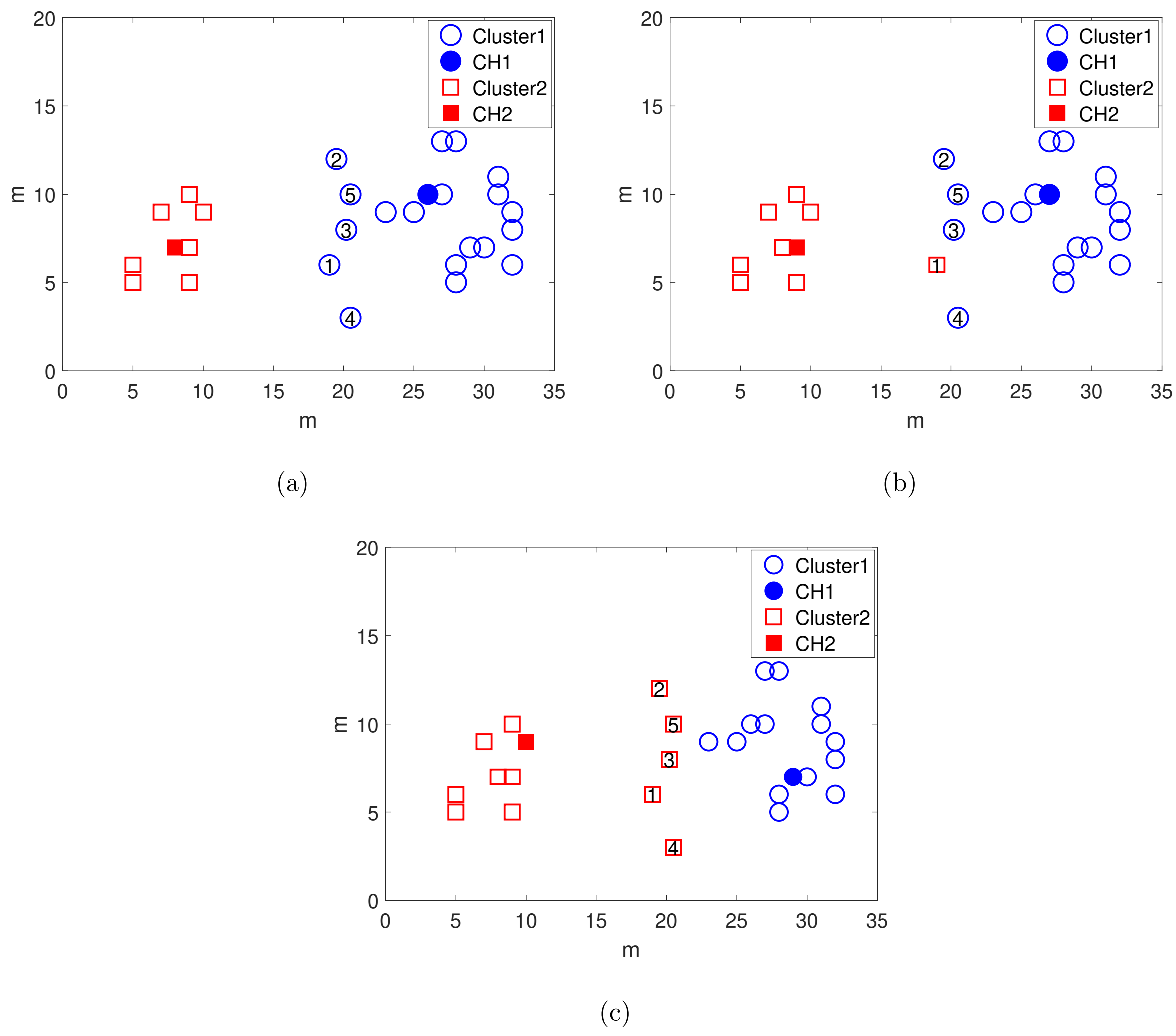


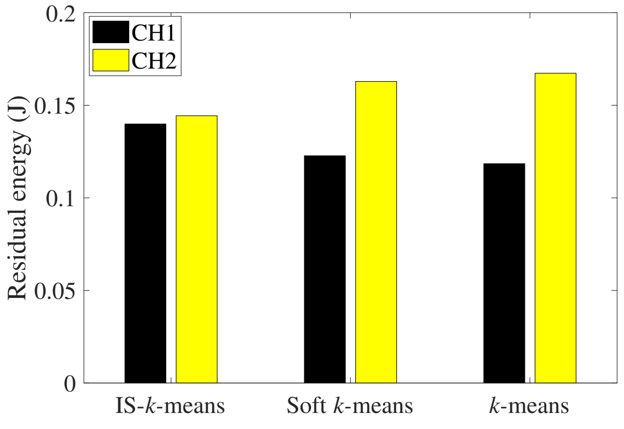
Figure 2.7 Comparison of different clustering results, $\beta = 0.2$. (a) k-means clustering result. (b) Soft k-means clustering result. (c) IS-k-means clustering result.

is $O(2n + nk)$ memory units.

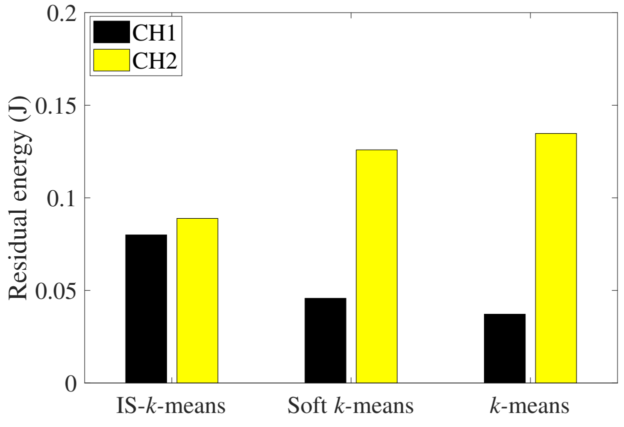
2.4 Experiment Results and Analysis

2.4.1 Simulation Settings

To evaluate the performance of the proposed algorithm, we consider two different scenarios. In Scenario 1, the network size is $100\,\text{m} \times 100\,\text{m}$ and the BS is located at $(50\,\text{m}, 150\,\text{m})$. Scenario 2 has the size of $200\,\text{m} \times 200\,\text{m}$ with the BS at location $(100\,\text{m}, 200\,\text{m})$. The main



(a)



(b)

Figure 2.8 Comparison of residual energy of CHs. (a) Residual energy of CHs after 5 rounds. (b) Residual energy of CHs after 10 rounds.

simulation parameters are selected as in [7] and listed in Table 2.1. The experiments are implemented using MATLAB R2017b.

2.4.2 Nodes Reassigning of Improved Soft k-Means Analysis

In this subsection, we will show the advantage of the node reassigning scheme incorporated in the proposed IS-k-means algorithm to balance the energy consumption of CHs. A total of 28 sensor nodes are randomly distributed in scenario 1. First, we use the k-means clustering method to classify these nodes and obtain two clusters, as shown in Fig. 2.7 (a). It is found that cluster 1 contains 20 nodes, which is quite larger than the number of nodes

Table 2.1 Simulation parameters

Parameter	**Value**
Area	$100\,\mathrm{m} \times 100\,\mathrm{m}$, $200\,\mathrm{m} \times 200\,\mathrm{m}$
BS coordinates	$(50\,\mathrm{m},\ 150\ \ \mathrm{m})$, $(100\,\mathrm{m},\ 200\ \ \mathrm{m})$
Initial energy	$0.2\,\mathrm{J}$, $1\,\mathrm{J}$
Packet length	4000 bits
Control length	100 bits
E_{T}	$50\,\mathrm{nJ/bit}$
E_{R}	$50\,\mathrm{nJ/bit}$
$\varepsilon_{\mathrm{fs}}$	$10\,\mathrm{pJ/bit/m^2}$
$\varepsilon_{\mathrm{mp}}$	$0.0013\,\mathrm{pJ/bit/m^4}$
E_{DA}	$5\,\mathrm{nJ/bit}$
d_0	$88\,\mathrm{m}$
Number of sensor nodes	28, 100
Maximum communication range	$250\,\mathrm{m}$ [46]

in cluster 2. As a result, CH of cluster 1 will be exhausted much earlier than that of cluster 2. Fig. 2.7 (b) shows the clustering result of the soft k-means algorithm. In Section III, we define β as the stiffness parameter, which represents the tightness of a node belonging to a cluster. Setting $\beta = 0.2$, we can find that the nodes at the edge of two clusters having similar membership probabilities belonging to these two clusters, such as node 1, node 2, node 3, node 4, and node 5, as shown in Table 2.2. Furthermore, if the value of β changes, the probabilities also will change. When $\beta = 1$, all five nodes belong to the clusters with higher probabilities when compared to the case where $\beta = 0.2$. In our proposed algorithm, we set $\beta = 0.2$ in the following simulations. According to the rule of node reassigning, node 2, node 3, node 4, and node 5 are reassigned to cluster 2 from cluster 1 as shown in Fig. 2.7

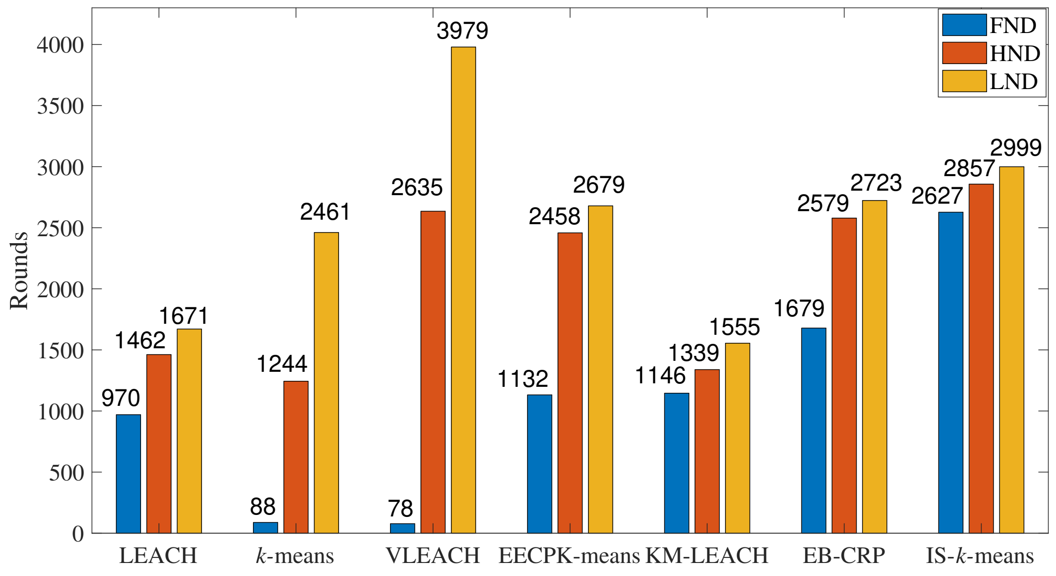
Table 2.2 Probabilities comparison

Probability		Node 1	Node 2	Node 3	Node 4	Node 5
$\beta = 0.2$	Cluster 1	0.4852	0.5537	0.5684	0.6125	0.6120
	Cluster 2	0.5148	0.4463	0.4316	0.3875	0.3880
$\beta = 1$	Cluster 1	0.0438	0.9787	0.9860	0.9919	0.9992
	Cluster 2	0.9562	0.0213	0.014	0.0081	0.0008

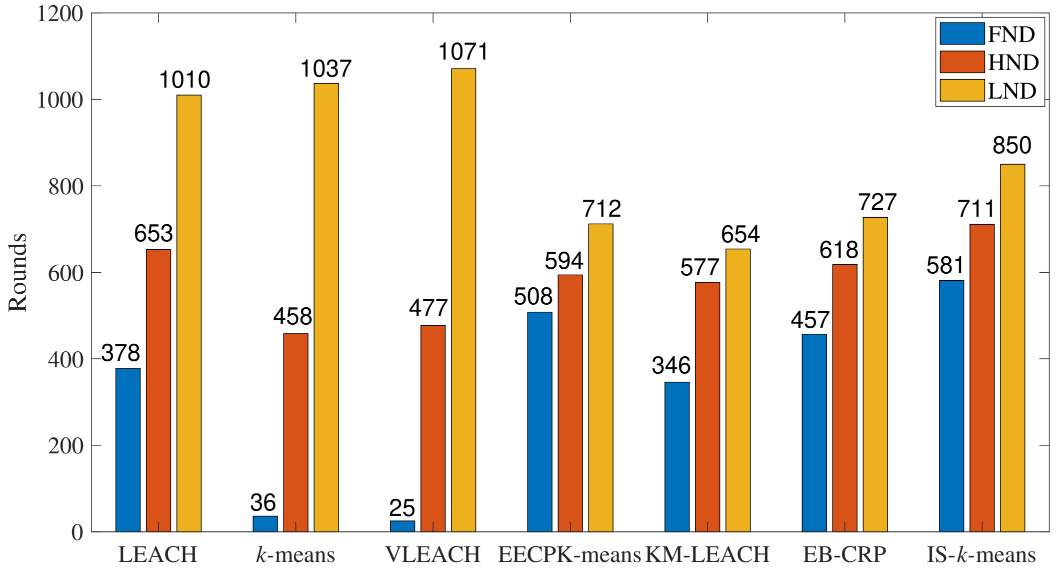
(c), which balances the energy overhead of CHs in these two clusters. The residual energy of CHs, computed by (2.22), in each round could be used to check the advantage of this scheme. Fig. 2.8 (a) and Fig. 2.8 (b) compare the residual energy of CHs among k-means, soft k-means, and IS-k-means after 5 rounds and 10 rounds, respectively. The IS-k-means algorithm achieves an equilibrium of energy consumption in both CHs when compared to the k-means and the soft k-means algorithms.

2.4.3 Network Lifetime

To test the performance of the proposed IS-k-means algorithm, we compare it with KM-LEACH [23], VLEACH [24], LEACH [20], k-means [47], EECPK-means [21], and EB-CRP [13] with the same parameters shown in Table 2.1. Here, we state two things about the implementation of the EB-CRP algorithm in our experiment. First, the original EB-CRP algorithm does not need to select the CHs because the authors consider a certain number of gateways with enough energy to act as CHs in WSN. However, our implemented EB-CRP algorithm needs to select CHs randomly from all sensor nodes because the network considered in our simulation contains only sensor nodes with the same initial energy and functionality. In order to have a fair comparison, we set the number of CHs in EB-CRP to be the same as that in our proposed algorithm. Thus, the location of CHs may be different in each steady-state phase because all nodes have the same chance to be CHs. Second, the steady-state phase of the original EB-CRP algorithm is composed of pre-specified 75 rounds. This is quite reasonable because the authors set the initial energy of CHs to be 10 J, which can maintain a high number of communication rounds. However, considering the



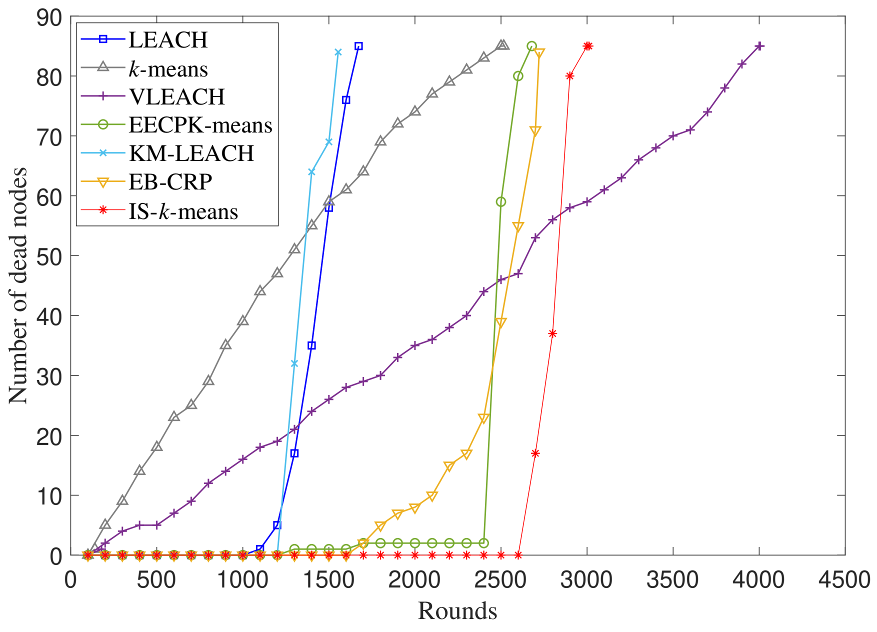
(a)



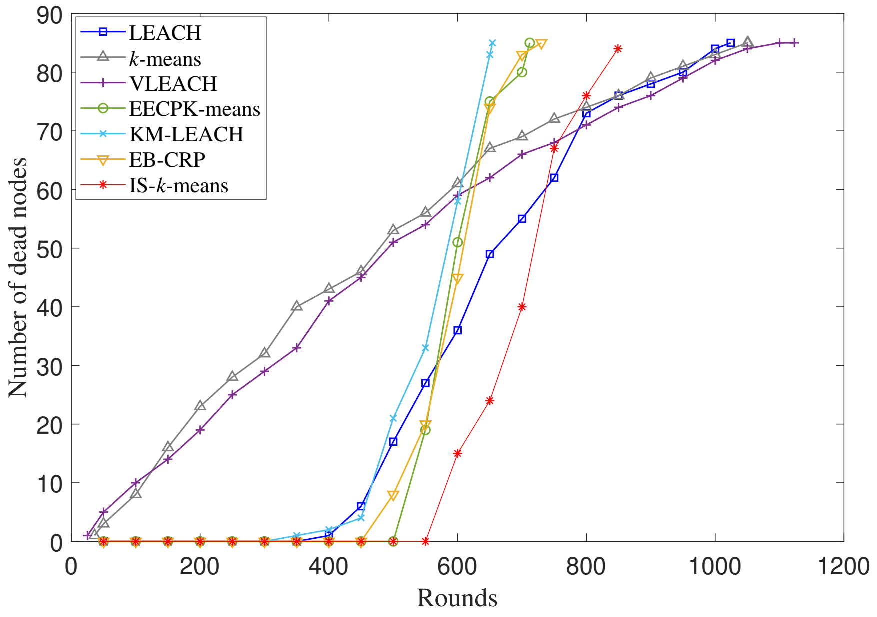
(b)

Figure 2.9 Comparison of FND, HND, and LND. (a) Scenario 1. (b) Scenario 2.

limited energy of CHs in our simulation, each steady-state phase is composed of 20 rounds in our implemented EB-CRP algorithm, which can achieve the best results for the EB-CRP algorithm.



(a)



(b)

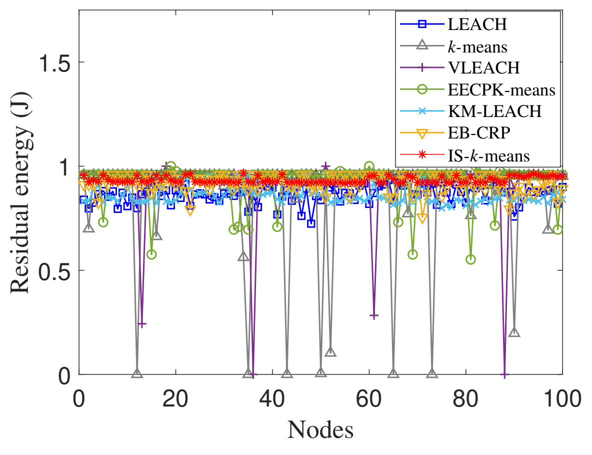
Figure 2.10 Comparison of network lifetime of LEACH, *k*-means, VLEACH, EECPK-means, KM-LEACH, EB-CRP, and IS-*k*-means. (a) Scenario 1. (b) Scenario 2.

We assume there are 100 sensor nodes that are randomly distributed in both scenario 1 and scenario 2. The obtained results are the averages of 20 independent experiments. The authors in [20] found the optimum number of clusters to be between 3 and 5 for 100-node network in LEACH. Thus, in scenario 1, we set 4 as the initial number of clusters in LEACH. For the other six algorithms, we use CFSFDP and KDE to determine the number of clusters in order to ensure the same number of clusters for each algorithm. The initial number of clusters found from the CFSFDP and KDE algorithms is 4 in scenario 1. In scenario 2, all algorithms are set with the same number of clusters 6 that is determined by CFSFDP and KDE. We assume that the death of 85% nodes means all nodes are dead.

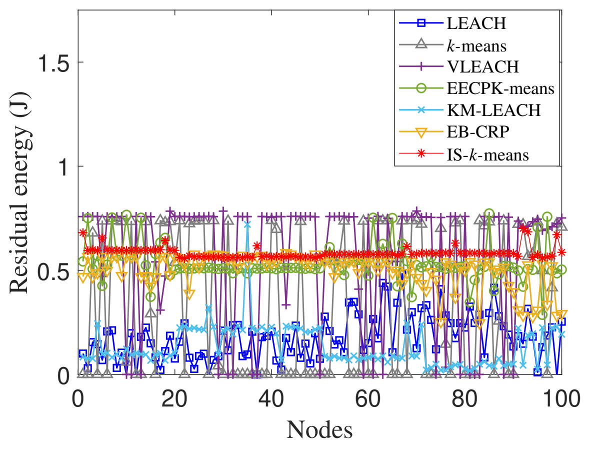
Fig. 2.9 shows the first node death (FND), half of nodes death (HND), and the last node death (LND) for these seven algorithms when the number of nodes is 100. If an algorithm can balance energy well, the first node death will be very late. In Fig. 2.9 (a), the average number of rounds of FND in k-means is 88, which is much earlier than 970 in LEACH and 2627 in IS-k-means, and the average LND happens later when compared to LEACH and the IS-k-means algorithms. Thus, it is obvious that the energy consumption of k-means is unbalanced. Although VLEACH uses the vice CH scheme in each cluster to extend the network lifetime, it exhibits a poor performance in balancing energy consumption because its FND is 78 and LND is 3979, as shown in Fig. 2.9 (a). EECPK-means improves the selection of initial cluster centers of the k-means algorithm by using the midpoint algorithm. It outperforms LEACH and KM-LEACH in both balancing energy consumption and extending network lifetime. For the EB-CRP algorithm, its FND is about 1.7 times that of LEACH, 19 times that of k-means, 21 times that of VLEACH, 1.5 times that of EECPK-means, and 1.5 times that of KM-LEACH, which demonstrates that the EB-CRP algorithm can postpone the death of the first node when compared with the other five algorithms. In addition, the HND of EB-CRP is 2579, which is larger than 1462 in LEACH, 1244 in k-means, 2458 in EECPK-means, and 1339 in KM-LEACH. This result means that the EB-CRP algorithm can delay the death of the first 50% of nodes as compared to LEACH, k-means, EECPK-means, KM-LEACH. Thus, the EB-CRP shows a good performance in balancing the energy consumption of the nodes and increasing the network lifetime. In view of Fig. 2.9 (a), our proposed IS-k-means algorithm can effectively postpone the FND, HND and LND. The average FND of IS-k-means is 2627, which is around 2.7 times that of LEACH, 30 times that of k-means, 34 times that of VLEACH, 2.3 times that of KM-LEACH, 2.4 times that of EECPK-means, and 1.5 times that of EB-CRP. Instead of using a fixed number of communication rounds during each steady-phase, like in the EB-CRP algorithm, the communication rounds in our proposed IS-k-means algorithm are determined by the residual energy of CHs. If the residual energy of any CH is below the threshold, the algorithm will stop the current steady-phase and trigger re-clustering, which can avoid CHs to die earlier than EB-CRP. Thus, the IS-k-means algorithm can keep all nodes in the network alive in most rounds. The average HND of the IS-k-means is also around 2 times among LEACH, k-means, and KM-LEACH.

In Fig. 2.9 (b), the average FND, HND, and LND of all algorithms are decreased. This is because extending the network size will increase the communication distance of the nodes which leads to an increase in the energy consumption. The VLEACH and k-means algorithms still show a very poor outcome in balancing the energy consumption, a consequence of having small FND and large LND. However, EECPK-means and EB-CRP have relatively large values of the FND and HND, which means most nodes in these two algorithms live longer when compared with k-means, VLEACH, and KM-LEACH. In addition, it is evident that our proposed IS-k-means algorithm has the best results in postponing the FND, HND, and LND as compared with the other six algorithms.

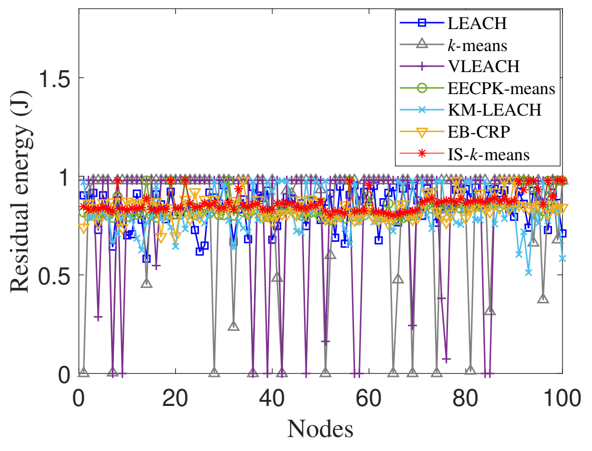
Fig. 2.10 shows the network lifetime comparison of our proposed IS-k-means algorithm and the other six algorithms. As can be seen from Fig. 2.10 (a), the network lifetime curves of KM-LEACH, LEACH, EECPK-means, and the proposed IS-k-means algorithms are approximately vertical. This means that, in these algorithms, the majority of nodes die approximately after the same number of rounds. Furthermore, one can see that the proposed IS-k-means algorithm outperforms KM-LEACH, LEACH, and EECPK-means algorithms in terms of the energy consumption equilibrium. The results in Fig. 2.10 (a) also show that VLEACH has a longer network lifetime than our proposed IS-k-means algorithm. This is reasonable since the objective of VLEACH is to extend the network lifetime, whereas our proposed IS-k-means algorithm aims to balance the energy consumption in the network. As



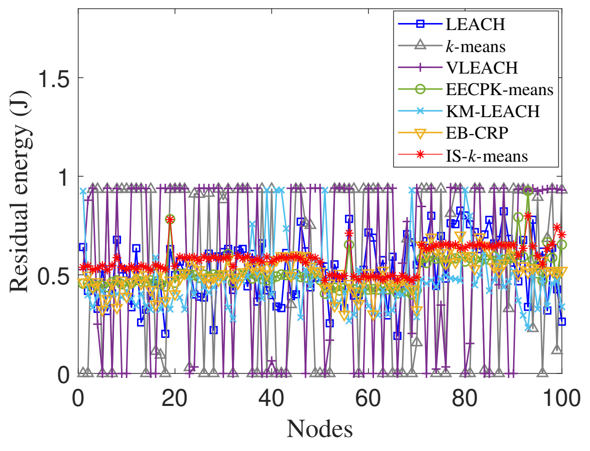
(a)



(b)



(c)



(d)

Figure 2.11 Comparison of residual energy curve. (a) Residual energy after 400 rounds in scenario 1. (b) Residual energy after 1000 rounds in scenario 1. (c) Residual energy after 100 rounds in scenario 2. (d) Residual energy after 300 rounds in scenario 2.

a result, some nodes die very early and others die very late in VLEACH, which likely results in the inability to collect sensing data from certain areas where some nodes are dead. In Fig. 2.10 (b), although none of the algorithms shows a nearly vertical curve, like in Fig. 2.10 (a), our proposed algorithm still outperforms the other six algorithms in balancing the energy consumption and prolonging the network lifetime.

Table 2.3 Comparison of energy variance of different rounds

		LEACH	k-means	VLEACH	EECPK-means	KM-LEACH	EB-CRP	IS-k-means
Scenario 1	200 rounds	0.0018	0.0374	0.0271	0.0127	0.0013	0.0019	0.0002
	400 rounds	0.0039	0.0768	0.0496	0.0264	0.0025	0.0024	0.0004
	600 rounds	0.0074	0.1161	0.0775	0.0396	0.0085	0.0026	0.0004
	800 rounds	0.0108	0.1163	0.0756	0.0432	0.0098	0.0038	0.0005
	1000 rounds	0.0168	0.1148	0.0882	0.0498	0.0094	0.0044	0.0008
	1200 rounds	0.0172	0.1016	0.0904	0.0515	0.0055	0.0067	0.0007
	1400 rounds	0.0095	0.0988	0.0901	0.0496	0.0024	0.0072	0.0009
Scenario 2	100 rounds	0.0081	0.0983	0.1070	0.0031	0.0112	0.0028	0.0022
	200 rounds	0.0131	0.1642	0.1645	0.0058	0.0186	0.0052	0.0045
	300 rounds	0.0248	0.1921	0.1899	0.0073	0.0258	0.0073	0.0046
	400 rounds	0.0375	0.1806	0.1963	0.0105	0.0392	0.0094	0.0080
	500 rounds	0.0388	0.1713	0.1821	0.0135	0.0312	0.0122	0.0110
	600 rounds	0.0357	0.1491	0.1647	0.0167	0.0181	0.0168	0.0141

2.4.4 Energy Variance

Fig. 2.11 compares the average residual energy of all 100 nodes in WSNs among the seven algorithms after different rounds in two scenarios. It is found that the residual energy curve of all nodes in the IS-k-means algorithm is smoother than that of the other six algorithms. This result demonstrates that the IS-k-means algorithm is good at balancing the energy consumption of all nodes in WSNs. For the purpose of estimating performance of the proposed algorithm, we introduce a new parameter called energy variance (EV), which is expressed as

$$\mathrm{EV} = \frac{\sum_{i=1}^{n} \left(E_i\left(r\right) - \overline{E}\right)^2}{n}, \tag{2.26}$$

where $\overline{E}$ is the average energy of all nodes. Table 2.3 clearly reveals that EB-CRP has relatively smaller variances than LEACH, k-means, VLEACH, KM-LEACH, and EECPK-means in different rounds. In addition, our proposed IS-k-means algorithm achieves the smallest variances among seven algorithms, which demonstrates that the IS-k-means can keep the residual energy of 100 nodes to be the most uniform in WSNs.

It is worthy to mention that the EB-CRP algorithm shows better performance in extending the network lifetime for WSNs with large network sizes [13], while the proposed algorithm has good performance in balancing the energy consumption and extending the network lifetime for smaller network sizes. We briefly summarize the reasons why the proposed algorithm performs better than the other six algorithms for WSNs of smaller sizes. First, optimizing the initial cluster centers of the soft k-means algorithm and reassigning nodes can better balance the number of nodes in different clusters to form good clustering results. Second, our algorithm selects nodes with more residual energy as the CHs, which can prevent the CHs from dying too early and support a high number of communication rounds. Third, the multi-CHs scheme of the proposed IS-k-means can reduce the communication energy consumption in the set-up phase caused by re-clustering because it reduces the number of re-clustering. Thus, all sensors can save energy to maintain more communication rounds in the steady phase, which extends the network lifetime. However, for the EB-CRP algorithm, it only chooses one CH in each cluster, which may cause all nodes to re-cluster frequently because CHs may quickly exhaust their energy. Fourth, instead of using a fixed number of communication rounds during each steady-phase, like in EB-CRP, the communication rounds in our proposed IS-k-means algorithm are determined by the residual energy of CHs. If the residual energy of any CH is below the threshold, the algorithm will stop the current steady-phase and trigger re-clustering, which can avoid CHs to die earlier than the EB-CRP algorithm.

2.5 Conclusions

In this paper, we proposed an energy balanced IS-k-means algorithm based on the soft k-means for WSNs. The proposed algorithm improves the selection of initial cluster centers by using CFSFDP and KDE algorithms. In order to balance the number of nodes per cluster, the proposed algorithm reassigns nodes at the edge of different clusters to a low-density cluster according to the nodes' membership probabilities. Furthermore, multi-CHs scheme was used in the selection of final CHs, which can effectively balance the traffic load of CHs, reduces the number of re-clustering and saves communication cost in the set-up phase. In order to show the advantages of the IS-k-means in balancing energy consumption, we compared it with LEACH, k-means, VLEACH, EECPK-means, KM-LEACH, and EB-CRP. In scenario 1, simulation results demonstrated that the proposed IS-k-means algorithm postponed the FND by 2.7 times, 34 times, 2.3 times, 2.4 times, 30 times, and 1.5 times when compared to LEACH, VLEACH, KM-LEACH, EECPK-means, k-means, and EB-CRP on average, respectively. The HND of the IS-k-means algorithm also was delayed by 2 times when compared to LEACH, k-means, and KM-LEACH. In addition, the IS-k-means algorithm achieved an excellent result in postponing the FND and HND in scenario 2 as compared with other mentioned algorithms. The IS-k-means algorithm also extended network lifetime in both scenarios as compared to KM-LEACH, EECPK-means, and EB-CRP. Furthermore, the proposed algorithm also yields smoother average remaining energy curves of all nodes in different rounds and smaller average energy variances. Hence, the proposed IS-k-means algorithm is promising in balancing energy consumption in WSNs. In a future work, we plan to design an energy-efficient multi-hop routing algorithm to extend the IS-k-means algorithm to large-scale WSNs.

References

[1] L. Chettri and R. Bera, "A comprehensive survey on internet of things (IoT) toward 5G wireless systems," *IEEE Internet of Things Journal*, vol. 7, pp. 16–32, Jan. 2020.

[2] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis, and E. K. Markakis, "A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues," *IEEE Communications Surveys and Tutorials*, vol. 22, pp. 1191–1221, Jan. 2020.

[3] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow, and M. N. Hindia, "An overview of internet of things (IoT) and data analytics in agriculture: Benefits and challenges," *IEEE Internet of Things Journal*, vol. 5, pp. 3758–3773, Oct. 2018.

[4] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "I-SEP: An improved routing protocol for heterogeneous WSN for IoT-based environmental monitoring," *IEEE Internet of Things Journal*, vol. 7, pp. 710–717, Jan. 2020.

[5] F. Deng, X. Yue, X. Fan, S. Guan, Y. Xu, and J. Chen, "Multisource energy harvesting system for a wireless sensor network node in the field environment," *IEEE Internet of Things Journal*, vol. 6, pp. 918–927, Feb. 2019.

[6] L. Xu, R. Collier, and G. M. P. O'Hare, "A survey of clustering techniques in WSNs and consideration of the challenges of applying such to 5G IoT scenarios," *IEEE Internet of Things Journal*, vol. 4, pp. 1229–1249, Oct. 2017.

[7] H. Xie, Z. Yan, Z. Yao, and M. Atiquzzaman, "Data collection for security measurement in wireless sensor networks: A survey," *IEEE Internet of Things Journal*, vol. 6, pp. 2205–2224, Apr. 2019.

[8] J. S. Lee and T. Y. Kao, "An improved three-layer low-energy adaptive clustering hierar-

chy for wireless sensor networks," *IEEE Internet of Things Journal*, vol. 3, pp. 951–958, Dec. 2016.

[9] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "Residual energy-based cluster-head selection in WSNs for IoT application," *IEEE Internet of Things Journal*, vol. 6, pp. 5132–5139, Jun. 2019.

[10] Z. Xu, L. Chen, C. Chen, and X. Guan, "Joint clustering and routing design for reliable and efficient data collection in large-scale wireless sensor networks," *IEEE Internet of Things Journal*, vol. 3, pp. 520–532, Aug. 2016.

[11] R. Zhang, J. Pan, D. Xie, and F. Wang, "NDCMC: A hybrid data collection approach for large-scale WSNs using mobile element and hierarchical clustering," *IEEE Internet of Things Journal*, vol. 3, pp. 533–543, Aug. 2016.

[12] R. Yarinezhad and S. N. Hashemi, "A routing algorithm for wireless sensor networks based on clustering and an fpt-approximation algorithm," *Journal of Systems and Software*, vol. 155, pp. 145–161, Sep. 2019.

[13] R. Yarinezhad and S. N. Hashemi, "Increasing the lifetime of sensor networks by a data dissemination model based on a new approximation algorithm," *Ad Hoc Networks*, vol. 100, p. 102084, Apr. 2020.

[14] R. Yarinezhad and S. N. Hashemi, "Solving the load balanced clustering and routing problems in WSNs with an fpt-approximation algorithm and a grid structure," *Pervasive and Mobile Computing*, vol. 58, p. 101033, Aug. 2019.

[15] N. Mazumdar and H. Om, "DUCR: Distributed unequal cluster-based routing algorithm for heterogeneous wireless sensor networks," *International Journal of Communication Systems*, vol. 30, p. e3374, Jul. 2017.

[16] N. Mazumdar and H. Om, "Distributed fuzzy logic based energy-aware and coverage preserving unequal clustering algorithm for wireless sensor networks," *International Journal of Communication Systems*, vol. 30, p. e3283, Sep. 2017.

[17] N. Mazumdar and H. Om, "Distributed fuzzy approach to unequal clustering and routing algorithm for wireless sensor networks," *International Journal of Communication systems*, vol. 31, p. e3709, May 2018.

[18] S. Randhawa and S. Jain, "Performance analysis of LEACH with machine learning algorithms in wireless sensor networks," *International Journal of Computer Applications*, vol. 147, pp. 7–12, Aug. 2016.

[19] A. Mahboub *et al.*, "Energy-efficient hybrid k-means algorithm for clustered wireless sensor networks," *International Journal of Electrical and Computer Engineering*, vol. 7, pp. 2054–2060, Aug. 2017.

[20] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, Oct. 2002.

[21] A. Ray and D. De, "Energy efficient clustering protocol based on k-means (EECPK-means)-midpoint algorithm for enhanced network lifetime in wireless sensor network," *IET Wireless Sensor Systems*, vol. 6, pp. 181–191, Dec. 2016.

[22] N. T. Tam, D. T. Hai, L. H. Son, and L. T. Vinh, "Improving lifetime and network connections of 3D wireless sensor networks based on fuzzy clustering and particle swarm optimization," *Wireless Networks*, vol. 24, pp. 1477–1490, Nov. 2018.

[23] M. Bidaki, R. Ghaemi, and S. R. K. Tabbakh, "Towards energy efficient k-means based clustering scheme for wireless sensor networks," *International Journal of Grid and Distributed Computing*, vol. 9, pp. 265–276, Jul., 2016.

[24] A. Sasikala *et al.*, "Improving the energy efficiency of LEACH protocol using VCH in wireless sensor network," *International Journal of Engineering Development and Research*, vol. 3, pp. 918–924, May 2015.

[25] E. Rabiaa, B. Noura, and C. Adnene, "Improvements in LEACH based on k-means and gauss algorithms," *Procedia Computer Science*, vol. 73, pp. 460–467, Dec. 2015.

[26] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, pp. 1492–1496, Jun. 2014.

[27] Y. Zhang, M. Liu, and Q. Liu, "An energy-balanced clustering protocol based on an improved CFSFDP algorithm for wireless sensor networks," *Sensors*, vol. 18, p. 881, Mar. 2018.

[28] A. Ihsani and T. H. Farncombe, "A kernel density estimator-based maximum a posteriori image reconstruction method for dynamic emission tomography imaging," *IEEE Transactions on Image Processing*, vol. 25, pp. 2233–2248, May 2016.

[29] C. Bauckhage, "Lecture notes on data science: Soft *k*-means clustering," tech. rep., Technical Report, Univ. Bonn, DOI: 10.13140/RG. 2.1. 3582.6643, Oct. 2015.

[30] P. Shen and C. Li, "Distributed information theoretic clustering," *IEEE Transactions on Signal Processing*, vol. 62, pp. 3442–3453, Jul. 2014.

[31] R. Sharma, V. Vashisht, and U. Singh, "EEFCM-DE: energy-efficient clustering based on fuzzy *c* means and differential evolution algorithm in WSNs," *IET Communications*, vol. 13, pp. 996–1007, May 2019.

[32] H. Yang, Q. Yao, A. Yu, Y. Lee, and J. Zhang, "Resource assignment based on dynamic fuzzy clustering in elastic optical networks with multi-core fibers," *IEEE Transactions on Communications*, vol. 67, pp. 3457–3469, May 2019.

[33] A. Majdara and S. Nooshabadi, "Nonparametric density estimation using copula transform, bayesian sequential partitioning, and diffusion-based kernel estimator," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, pp. 821–826, Apr. 2019.

[34] Y. Li, Y. Zhang, M. Yu, and X. Li, "Drawing and studying on histogram," *Cluster Computing*, vol. 22, pp. 3999–4006, Mar. 2019.

[35] A. Qahtan, S. Wang, and X. Zhang, "KDE-track: An efficient dynamic density estimator for data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 642–655, Mar. 2016.

[36] V. Savic, E. G. Larsson, J. Ferrer-Coll, and P. Stenumgaard, “Kernel methods for accurate UWB-based ranging with reduced complexity,” *IEEE Transactions on Wireless Communications*, vol. 15, pp. 1783–1793, Mar. 2016.

[37] B. Qin and F. Xiao, “A non-parametric method to determine basic probability assignment based on kernel density estimation,” *IEEE Access*, vol. 6, pp. 73509–73519, Nov. 2018.

[38] R. Mehmood, G. Zhang, R. Bie, H. Dawood, and H. Ahmad, “Clustering by fast search and find of density peaks via heat diffusion,” *Neurocomputing*, vol. 208, pp. 210–217, Oct. 2016.

[39] J. Zhong, W. T. Peter, and Y. Wei, “An intelligent and improved density and distance-based clustering approach for industrial survey data classification,” *Expert Systems with Applications*, vol. 68, pp. 21–28, Feb. 2017.

[40] W. B. Heinzelman, *Application-specific protocol architectures for wireless networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[41] J. Qin, W. Fu, H. Gao, and W. X. Zheng, “Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory,” *IEEE Transactions on Cybernetics*, vol. 47, pp. 772–783, Mar. 2017.

[42] M. Lehsaini and M. B. Benmahdi, “An improved k-means cluster-based routing scheme for wireless sensor networks,” in *Proc. IEEE International Symposium on Programming and Systems (ISPS)*, pp. 1–6, 2018.

[43] W. Zhang and J. Li, “Extended fast search clustering algorithm: widely density clusters, no density peaks,” *arXiv preprint arXiv:1505.05610*, 2015.

[44] T. C. Havens, J. C. Bezdek, C. Leckie, L. O. Hall, and M. Palaniswami, “Fuzzy c-means algorithms for very large data,” *IEEE Transactions on Fuzzy Systems*, vol. 20, pp. 1130–1146, Dec. 2012.

[45] M. Wang, F. Min, Z. Zhang, and Y. Wu, “Active learning through density clustering,” *Expert systems with applications*, vol. 85, pp. 305–317, Nov. 2017.

[46] L. Chhaya, P. Sharma, G. Bhagwatikar, and A. Kumar, “Wireless sensor network based smart grid communications: Cyber attacks, intrusion detection system and topology control,” *Electronics*, vol. 6, pp. 1–22, Jan. 2017.

[47] P. Sasikumar and S. Khara, “k-means clustering in wireless sensor networks,” in *Proc. IEEE Fourth International Conference on Computational Intelligence and Communication Networks*, pp. 140–144, Nov. 2012.

3. Joint Cluster Head Selection and Trajectory Planning in UAV-Aided IoT Networks by Reinforcement Learning with Sequential Model

Published as:

B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton and J. Henry, "Joint cluster head selection and trajectory planning in UAV-aided IoT networks by reinforcement learning with sequential model," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12071–12084, Jul. 2022.

In the previous chapter, an efficient clustering algorithm is proposed for balancing the energy consumption among devices in WSNs. Given the growing interest in using UAVs to collect data in IoT networks, in this chapter we study the energy saving problem in UAV-aided networks.

The manuscript included in this chapter investigates the problem of jointly designing the UAV's trajectory and selecting CHs for an IoT network to minimize the total energy consumption in the UAV-IoT system. DRL with a Seq2Seq neural network is used to learn the policy of the UAV trajectory to meet the objective of minimizing the total energy consumption of the whole UAV-IoT system. The proposed algorithm offers an appealing balance between performance and complexity.

Joint Cluster Head Selection and Trajectory Planning in UAV-Aided IoT Networks by Reinforcement Learning with Sequential Model

Botao Zhu, Ebrahim Bedeer, Ha Nguyen,

Robert Barton, and Jerome Henry

Abstract

Employing unmanned aerial vehicles (UAVs) has attracted growing interests and emerged as the state-of-the-art technology for data collection in Internet-of-Things (IoT) networks. In this paper, with the objective of minimizing the total energy consumption of the UAV-IoT system, we formulate the problem of jointly designing the UAV's trajectory and selecting cluster heads in the IoT network as a constrained combinatorial optimization problem which is classified as $\mathcal{NP}$-hard, and challenging to solve. We propose a novel deep reinforcement learning (DRL) with a sequential model strategy that can effectively learn the policy represented by a sequence-to-sequence neural network for the UAV's trajectory design in an unsupervised manner. Through extensive simulations, the obtained results show that the proposed DRL method can find the UAV's trajectory that requires much less energy consumption when compared to other baseline algorithms and achieves close-to-optimal performance. In addition, simulation results show that the trained model by our proposed DRL algorithm has an excellent generalization ability to larger problem sizes without the need to retrain the model.

Index Terms

Deep reinforcement learning, Internet-of-Things, UAV, cluster head selection, trajectory planning.

3.1 Introduction

The Internet-of-Things (IoT) is a system that connects a massive number of devices to the Internet, which is rapidly changing the way we live in almost every field [1]. Wireless Sensor Networks (WSNs) are viewed as the basic component of IoT. WSN can integrate the physical world with the information world to expand the functions of existing networks and the ability of humans to understand the world. A typical WSN is composed of a large number of devices deployed over a geographical area for monitoring physical events. These devices form a multi-hop self-organizing network to monitor, sense, and collect the information of the target area, and transfer the collected data to users for processing [2]. However, WSNs have gradually merged with different applications and appeared in various new forms, such as industrial IoT [3], Internet of vehicles (e.g., 5G long distance WSN) [4], smart home (e.g., short distance WSN) [5], environmental monitoring (e.g., autonomous underwater vehicles network) [6], intelligent manufacturing system, etc.

Traditional long-distance multi-hop communication requires high energy consumption of end devices, and because of the limited energy resources of end devices in IoT networks, it leads to shortening the network lifetime. In [7], an ant optimization based routing algorithm is proposed to dramatically reduce the energy consumption of networks. In [8], the authors consider economic theory and compressive sensing theory to propose an energy-saving routing algorithm to extend the lifetime of WSNs. These energy saving techniques only focus on designing the routing algorithms to reduce the energy consumption of networks. Recently, the use of unmanned aerial vehicles (UAVs) has received increasing attention due to their high flexibility and high maneuverability. Compared with the conventional IoT networks that use static multi-hop data collection methods, UAV-enabled IoT networks dispatch a UAV to collect data from ground IoT devices based on the planned UAV's trajectory [9], which can effectively reduce the energy consumption of devices. However, any increase in the flight time or distance of the UAV for a given data collection mission will increase its energy consumption. Hence, there is a need to carefully design the UAV trajectory to minimize the overall energy consumption of the wireless network.

There is rich literature concerning the problem of energy consumption in UAV-aided wireless networks. By jointly considering the UAV's trajectory and devices' transmission schedule, the authors in [10] use an efficient differential evolution-based method to minimize the maximum energy consumption of all devices in an IoT network. In [11], the authors aim to minimize the transmission energy consumption of the sensor nodes within a given data collection time by jointly optimizing the UAV's trajectory and the transmission policy of nodes. In [12], the authors consider maximizing the minimum residual energy of sensors after data transmission in order to prolong the network lifetime. The authors in [13] jointly optimize the sensor nodes' wake-up schedule and the UAV trajectory to reduce the maximum energy consumption of all sensor nodes.

All the above mentioned works only focus on minimizing the energy consumption of ground devices in an UAV-aided wireless network. In contrast, other works consider UAV-related energy consumption minimization when UAVs are deployed in wireless networks. In [14], the authors study the problem of minimizing the completion time and the energy consumption of an UAV flying over a large area and propose a fly-and-communicate protocol. The authors in [15] aim to minimize the total energy consumption of the UAV, including both the propulsion and communication energy, in a UAV-enabled system serving multiple ground nodes. The authors in [16] study methods to control a group of UAVs for effectively covering a large geographical region while minimizing their energy consumption. In [17], the authors minimize the total UAV's energy consumption for a given path by optimizing its velocity.

Against the above literature, we consider to minimize the total energy consumption of the ground network and the UAV by designing an energy efficient UAV trajectory in a cluster-based IoT network. An important difference between our work and existing studies is how the UAV interacts with the ground network. Specifically, existing studies consider the scenario that the UAV directly communicates with each device of the ground network. Such a scenario leads to high energy consumption of both the UAV and ground devices, especially when the network size increases. In contrast, we consider a clustered IoT network and that the UAV only communicates with the cluster head (CH) of each cluster in order to reduce the energy consumption. As such, the UAV trajectory optimization problem is formulated to jointly select CHs and plan the UAV's visiting order to these CHs to minimize the overall energy consumption of the UAV-aided IoT network. The formulated optimization problem turns out to be one canonical example of combinatorial optimization problems, i.e., the generalized traveling salesman problem (GTSP).

In general, existing solutions for energy-efficient UAV trajectory planning can be classified into two categories: *traditional methods* (including exact algorithms, heuristic or meta-heuristic algorithms, etc.), and *machine learning based techniques*. Although exact algorithms can provide the optimal solutions (through systematic enumeration, mathematical programming, etc), as the size of the optimization problems increases, their computational costs grow and may prohibit practical implementations [18]. As for heuristic or meta-heuristic algorithms, there is no guarantee that the obtained solutions are close-to-optimal solutions [19]. On the other hand, deep reinforcement learning (DRL) techniques have gained remarkable attention in solving the energy efficient UAV trajectory planning problems. For instance, the authors in [20] propose a DRL algorithm to design the UAV cruise route in a smart city environment, where convolutional neural networks (CNNs) are used for feature extraction and the deep Q-network (DQN) is utilized to make decisions. In [21], the authors use the DQN with experience replay memory to solve the formulated energy-efficient trajectory optimization problem, while maintaining data freshness. To provide energy-efficient and fair communication service, the authors in [22] design a DRL algorithm based on deep neural networks (DNNs) and deep deterministic policy gradient (DDPG) to plan the UAV trajectory in a 3D coverage scenario. With the objective of saving energy, the authors in [23] propose a deep stochastic online scheduling algorithm based on two DNNs and the actor-critic to overcome the traditional DRL's limitations in addressing UAV trajectory optimization problem.

In designing a machine learning-based algorithm to solve our formulated combinatorial optimization problem, it is useful to require the algorithm to have the following capabilities: scalability, generalization, and automation on variable-length data structures. This stems from the fact that the number of clusters or nodes in the IoT network may not be the same in different data collection tasks over a given region. The scalability means that the algorithm is not only able to handle IoT networks with small-scale clusters but also scale to IoT networks with large-scale clusters. The generalization means that the machine learning algorithm should also perform well on unseen problems. The automation of the machine learning algorithm is to automatically execute operations of generalization and scalability on new problem instances, without retraining the model or manually modifying its parameters. In other words, once the model is trained by the designed machine learning algorithm in our work, it can automatically produce good solutions to new IoT networks with different number of clusters and different locations of nodes.

Since combinatorial optimization problems, such as TSP, vehicle routing problem (VRP), etc., are often solved as sequence-to-sequence (Seq2Seq) prediction problems in machine learning [24], we require the designed machine learning algorithm to have an excellent ability to learn policy on sequential data as in our formulated combinatorial optimization problem that also can be seen as a sequential problem. Machine learning algorithms, such as CNN, DQN, and DDPG, are inefficient to handle sequential problems where the current element of the sequence depends on historical information from the previous elements of the sequence. This makes it hard for these algorithms to store information of past elements for very long time [25]. Recurrent neural networks (RNNs) with long short-term memory (LSTM) are frequently used to process sequential problems because their hidden units can store historical information for long time steps [24]. In addition, RNNs are the state-of-the-art neural networks to tackle variable-length sequences, e.g., variable-size data in our problem, by re-using the neural network blocks and parameters at every step of the sequence [26]. Attention mechanism is another technique to process a variable-length sequence by sharing its parameters [26]. Hence, RNNs and the attention mechanism-based Seq2Seq models that are commonly composed of the encoder component and the decoder component are emerging as attractive techniques to tackle variable-size sequential problems and they show promising results in various domains, see e.g. [27–32]. Given that we formulate the UAV trajectory planning problem in an UAV-aided cluster-based IoT network as a GTSP, Seq2Seq model with RNNs and the attention mechanism are the right ingredients for developing an efficient DRL algorithm to solve this challenging problem. The main contributions of this paper are summarized as follows:

1. We formulate the energy consumption minimization problem in the UAV-IoT system by jointly selecting CHs from a cluster-based IoT network and planning the UAV's trajectory to the selected CHs.

2. By viewing the formulated UAV trajectory planning problem in the clustered IoT network as a combinatorial optimization problem, we propose a Seq2Seq neural network to model and solve the trajectory planning problem. The inputs to the Seq2Seq neural network are all clusters and the UAV's start/end point; while the output of the Seq2Seq is the UAV's trajectory including the set of selected CHs. Reinforcement learning (RL) is used to train the parameters of the Seq2Seq in an unsupervised way to produce a close-to-optimal trajectory that ensures the minimum energy consumption in the UAV-IoT system.

3. Extensive simulations demonstrate that the proposed DRL-based method can find the optimal or close-to-optimal UAV's trajectory and outperforms baseline techniques when evaluating both the energy consumption in UAV-IoT system and the algorithms' computation time. In addition, the trained model by our proposed DRL algorithm shows good abilities of scalability, generalization, and automation to deal with IoT networks with different numbers of clusters without the need to retrain the model.

The rest of this paper is organized as follows. Section 3.2 describes the system model and problem formulation. Section 3.3 explains how deep reinforcement learning can be used to address the UAV's trajectory planning problem considered in our work. Section 3.4 presents simulation results. Finally, Section 3.5 concludes the paper.

3.2 System Model and Problem Formulation

We assume that one rotary-wing UAV is dispatched to collect data from K ground clusters. Each cluster $G_k, k = 1, \ldots, K$, is composed of N nodes, and only one node is selected as the CH, denoted by $b_k, b_k \in G_k$. The selection of CHs will be determined by the proposed algorithm. In each cluster, member nodes are responsible for sensing and collecting the environmental data and then send the collected data to the CH. The UAV is assumed to have the flying-hovering model without considering the acceleration-deceleration pattern. It takes off from the start hovering point c_0, corresponding to the ground BS b_0, visits each target hovering point c_k in a certain order, which is vertically above each CH b_k, and returns to c_0 after completing the data collection mission. The location of b_k in the cluster G_k is represented by a 3D Cartesian coordinate $(x_k^{\text{CH}}, y_k^{\text{CH}}, 0)$, and the position of n-th member node in this cluster is $(x_k^{(n)}, y_k^{(n)}, 0)$. Similarly, the position of each hovering point c_k can be represented as $(x_k^{\text{CH}}, y_k^{\text{CH}}, H)$ where H is the fixed flight height of the UAV. The problem of UAV trajectory planning can be regarded as the determination of hovering points $\{c_k\}_{k=1}^{K}$ and a permutation of $\{c_k\}_{k=1}^{K}$ and c_0. As an illustrative example in Fig. 3.1, if we choose the center node of each cluster as the CH, the energy consumption in the ground network will be minimum because the Euclidean distances between member nodes and their CHs are small in each cluster [33]. However, this increases the length of the UAV trajectory, and thus increases the energy consumption of the UAV (see the golden dashed line). On the other hand, if a boundary node in each cluster is chosen as the CH, the energy consumption of the UAV will be lower because it has a short trajectory (see the black dashed line). However, in this case the energy consumption for communication in the ground network will increase. Therefore, studying the problem of jointly selecting CHs and planning the UAV's trajectory to minimize the energy consumption of the UAV-IoT network is relevant and very important. This problem will be described in more detail in the following subsections.

3.2.1 Channel Model

In this work, we consider the air-to-ground channel model as in [34] where line-of-sight (LoS) links and nonline-of-sight (NLoS) links are used between the UAV and the ground devices. The probability of a LoS link typically is given by

$$P_{\text{LoS}} = \frac{1}{1 + \eta \exp\left(-\beta[\tau - \eta]\right)} \tag{3.1}$$

where η and β are environment constants, and $\tau = \arcsin\left(H/d_k\right)180/\pi$, where $d_k = ||c_k - b_k||$ is the distance between the UAV and the ground CH b_k when the UAV hovers at c_k. The

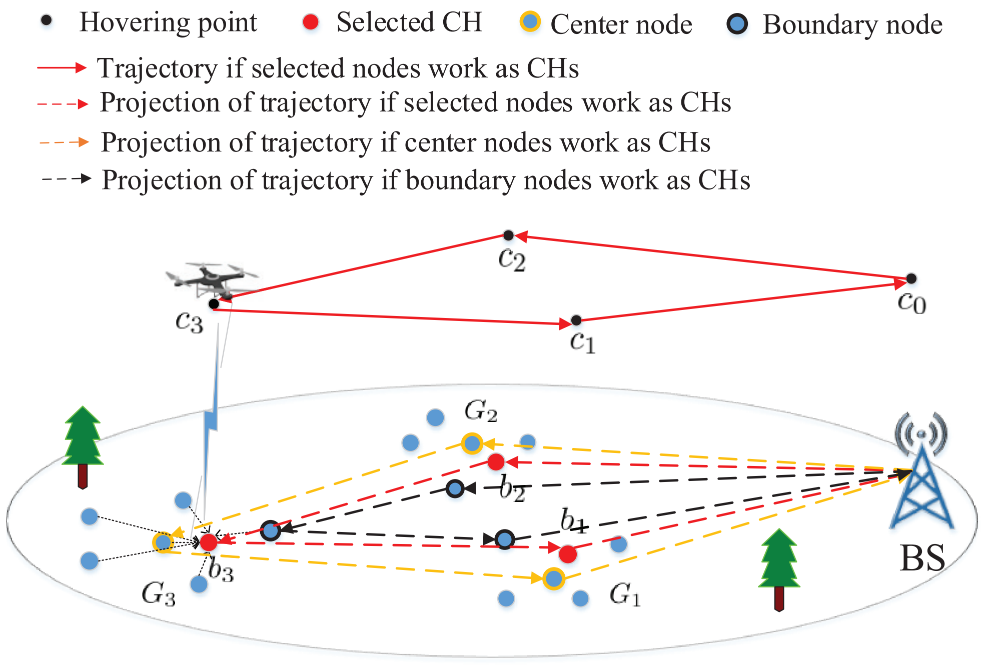

Hovering point
Selected CH
Center node
Boundary node
Trajectory if selected nodes work as CHs
Projection of trajectory if selected nodes work as CHs
Projection of trajectory if center nodes work as CHs
Projection of trajectory if boundary nodes work as CHs
c_0
c_1
c_2
c_3
G_1
G_2
G_3
b_1
b_2
b_3
BS

Figure 3.1 System model of an UAV-aided cluster-based IoT network.

probability of a NLoS link is given by $P_{\text{NLoS}} = 1 - P_{\text{LoS}}$. The average path loss between b_k and the UAV can be expressed as [34]

$$\begin{aligned}\overline{P}_{\text{loss}} = P_{\text{LoS}}\left(10\alpha\log_{10}\left(\frac{4\pi f_c H}{c}\right) + \mu_{\text{LoS}}\right)\\ + P_{\text{NLoS}}\left(10\alpha\log_{10}\left(\frac{4\pi f_c H}{c}\right) + \mu_{\text{NLoS}}\right)\end{aligned} \tag{3.2}$$

where μ_{LoS} and μ_{NLoS} are the average additional losses in LoS and NLoS links, respectively, α is the path loss exponent, c is the speed of light, and f_c is the carrier frequency. Assuming that all CHs have the same transmit power P_{CH}, the average data rate for the communication between each CH and the UAV is defined by [34]

$$r_{\text{data}} = B_{\text{width}}\log_2\left(1 + \frac{P_{\text{CH}}}{\overline{P}_{\text{loss}}N_0}\right) \tag{3.3}$$

where B_{width} is the communication bandwidth and N_0 is the noise power.

3.2.2 UAV's Energy Model

Without loss of generality, we assume that the UAV flies with a fixed speed v_{UAV} from one hovering point to another. The propulsion power consumption of the UAV for movement is given by [34, 35]

$$P_{\text{move}} = \sqrt{\frac{(m_{\text{tot}} g)^3}{2\pi r_p^2 n_p \rho}} + \frac{P_{\text{full}} - P_{\text{s}}}{v_{\text{full}}} v_{\text{UAV}} + P_{\text{s}} \tag{3.4}$$

where g, m_{tot}, r_p, n_p, and ρ are the earth gravity, UAV's mass, propeller radius, number of propellers, and air density, respectively. P_{full} and P_{s} are the hardware power levels when the UAV is moving at full speed v_{full} and when the UAV hovers, respectively. When the UAV hovers at the hovering point c_k to collect data from the ground CH b_k, its power consumption P_{hover} for hovering status is obtained by substituting $v_{\text{UAV}} = 0$ in (5.9). We assume that the hovering time of the UAV is equal to the data transmission time. Hence, its energy consumption is given by

$$E_{c_k} = \frac{D_k}{r_{\text{data}}} (P_{\text{hover}} + P_{\text{com}}) \tag{3.5}$$

where D_k is the amount of data needed to be collected, and P_{com} is the UAV's communication power. The energy consumption of the UAV for moving from point c_k to another point c_j is given by

$$E_{c_k, c_j} = \frac{||c_k - c_j||}{v_{\text{UAV}}} P_{\text{move}}. \tag{3.6}$$

Hence, by substituting (5.9) into (3.6), the total energy consumption of the UAV in flight can be written as

$$\begin{aligned} E_{\text{flight}} &= \sum_{k=0}^{K} \sum_{\substack{j=0 \\ j \neq k}}^{K} E_{c_k, c_j} L_{c_k} \\ &= \sum_{k=0}^{K} \sum_{\substack{j=0 \\ j \neq k}}^{K} \frac{L_{c_k, c_j} ||c_k - c_j|| \left(P_{\text{full}} - P_{\text{s}}\right)}{v_{\text{full}}} \\ &+ \sum_{k=0}^{K} \sum_{\substack{j=0 \\ j \neq k}}^{K} \frac{L_{c_k, c_j} ||c_k - c_j||}{v_{\text{UAV}}} \left(\sqrt{\frac{(m_{\text{tot}} g)^3}{2\pi r_p^2 n_p \rho}} + P_{\text{s}} \right), \\ & \forall c_k, c_j \in \mathcal{C} \end{aligned} \tag{3.7}$$

where $\mathcal{C} = \{c_0, c_1, \ldots, c_K\}$, c_k is determined by b_k, $b_k \in G_k$, and L_{c_k,c_j} indicates whether the UAV travels from c_k to c_j. Specifically, it is defined as

$$L_{c_k,c_j} = \begin{cases} 1, & \text{the path goes from } c_k \text{ to } c_j \\ 0, & \text{otherwise.} \end{cases} \tag{3.8}$$

As can be seen from (3.7), E_{flight} is inversely proportional to the speed v_{UAV}. Furthermore, one can show that the choices of v_{UAV} and hovering points in $\mathcal{C}$ have independent effects on E_{flight}. This means that the UAV speed v_{UAV} and the hovering points in $\mathcal{C}$ can be optimized separately. It is simple to see that, according to our system model, in order to minimize E_{flight}, and hence, the overall energy consumption, v_{UAV} should be set to the maximum flight speed v_{full}. It should be pointed out, however, that for other power models of UAVs (see [15] for example), the optimal value of v_{UAV} can be any value lower than or equal to v_{full}.

The following constraints need to be considered for the UAV's trajectory:

$$\sum_{\substack{k=0 \\ k \neq j}}^{K} L_{c_k,c_j} = 1, \quad \forall c_k, c_j \in \mathcal{C} \tag{3.9}$$

$$\sum_{\substack{j=0 \\ j \neq k}}^{K} L_{c_k,c_j} = 1, \quad \forall c_k, c_j \in \mathcal{C} \tag{3.10}$$

$$\sum_{c_k \in \mathcal{F}} \sum_{c_j \in \mathcal{F}} L_{c_k,c_j} \leq |\mathcal{F}| - 1, \quad \forall \mathcal{F} \subset \mathcal{C}; |\mathcal{F}| \geq 2. \tag{3.11}$$

The constraints (3.9) and (3.10) guarantee that the UAV should visit each point in $\mathcal{C}$ exactly once. Constraint (3.11) enforces that there is only one single trajectory without partial loop exists, where $\mathcal{F}$ is the subset of $\mathcal{C}$ [36].

3.2.3 IoT Network's Energy Models

The total energy consumption in the ground network includes energy consumption for intra-cluster communication and energy consumed for data transmission from CHs to the UAV. We use the first-order radio model [37] to calculate the intra-cluster energy consumption. In order to transmit an l-bit message to its CH b_k, the energy consumed by a member node n is given by [37]

$$E_n^{b_k} = lE_{\text{elec}} + l\left(\chi\varepsilon_{\text{fs}} d_{n,b_k}^2 + (1-\chi)\,\varepsilon_{\text{mp}} d_{n,b_k}^4\right) \tag{3.12}$$

where

$$\chi = \begin{cases} 1, & d_{n,b_k} \le d_0 \\ 0, & d_{n,b_k} > d_0 \end{cases} \tag{3.13}$$

and

$$d_0 = \sqrt{\frac{\varepsilon_{\text{fs}}}{\varepsilon_{\text{mp}}}}; \tag{3.14}$$

E_{elec} is the dissipated energy per bit in the circuitry, d_{n,b_k} is the distance between CH b_k and member nodes n, d_0 is the distance threshold, ε_{fs} and ε_{mp} are the radio amplifier's energy parameters corresponding to the free space and multi-path fading models, respectively [38]. On the other hand, the energy consumption of CH b_k to receive an l-bit message from member node n is calculated as [37]

$$E_{b_k}^{(n)} = lE_{\text{elec}}. \tag{3.15}$$

Furthermore, the energy consumed by CH b_k to complete data transmission to the UAV is

$$E_{b_k} = P_{\text{CH}} \frac{D_k}{r_{\text{data}}}, \tag{3.16}$$

where $D_k = (N-1)l$.

3.2.4 Problem Formulation for UAV's Trajectory

Based on the discussed energy models, after the UAV completes a round of data collection task, the total weighted energy consumption of the UAV-IoT system is formulated as

$$\begin{aligned} E\left(b_0, b_1, \ldots, b_K\right) = \omega &\left(\sum_{k=1}^{K}\sum_{n=1}^{N-1}\left(E_n^{b_k} + E_{b_k}^{(n)}\right) + \sum_{k=1}^{K} E_{b_k}\right) \\ &+ (1-\omega)\left(E_{\text{flight}} + \sum_{k=1}^{K} E_{c_k}\right), \; 0 \le \omega \le 1 \end{aligned} \tag{3.17}$$

where ω is a weighting coefficient that adjusts the energy consumption trade-off between the UAV and the ground networks. Note that the first term is the total energy consumption of the ground network, which only depends on the positions of CHs, and the second term is the total energy consumption of the UAV. E_{c_k} depends on CHs, and E_{flight} is related to CHs and the visiting order to CHs. With the aim of minimizing the overall weighted energy consumption of the UAV-IoT system, we formulate the optimization problem as jointly selecting CHs and designing the UAV's trajectory, which can be written as

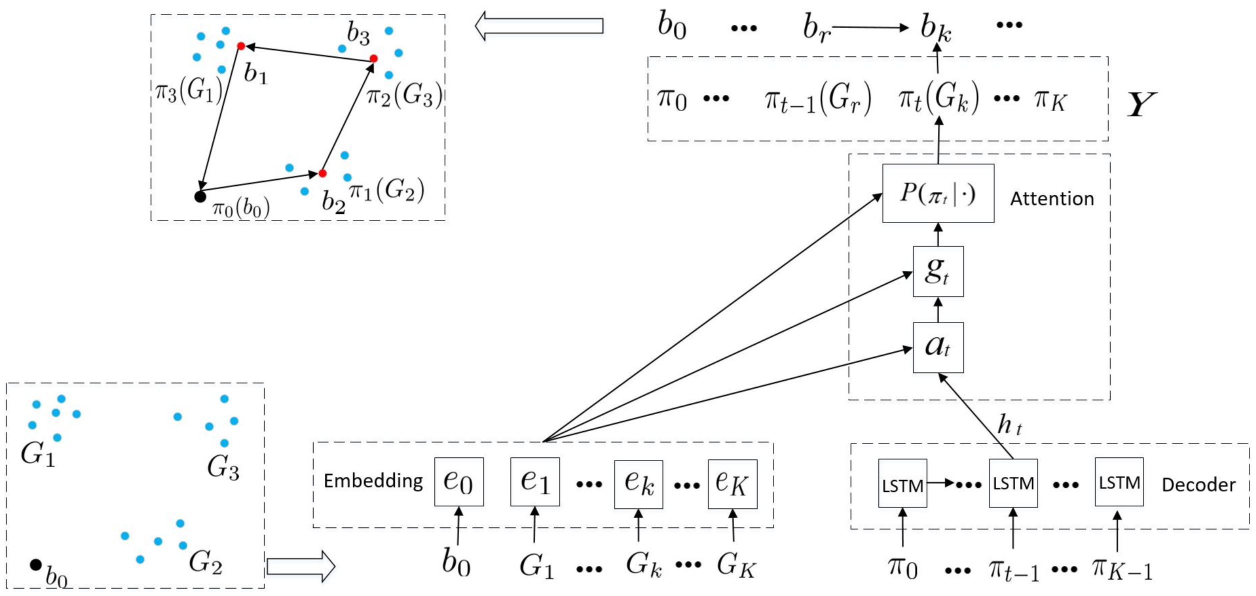
$$\begin{aligned} &\min_{\substack{\{b_0,b_1,\ldots,b_k,\ldots,b_K\} \\ b_k \in G_k}} \quad E\left(b_0, b_1, \ldots, b_K\right) \\ &\text{s.t.}\,(3.8)-(3.11). \end{aligned} \tag{3.18}$$

Clearly, the above formulated problem is GTSP, where the UAV is required to find a tour with the minimal energy consumption of the UAV-IoT system that includes exactly one node from each cluster. Due to the $\mathcal{NP}$-hardness of the formulated problem, it is difficult to solve with conventional methods such as heuristic algorithms. Recent major breakthroughs in DRL have shown that DRL has the ability to successfully solve some combinatorial optimization problems [39]. Hence, we propose a sequential model-based DRL method to tackle the problem of jointly selecting CHs and planning the UAV's trajectory.

3.3 Deep Reinforcement Learning for UAV Trajectory

3.3.1 UAV's Trajectory as Sequence Prediction

Because the UAV needs to visit all clusters sequentially to collect data, the trajectory planning problem can be viewed as the visiting decision problem by a policy. This policy can be modeled as a Seq2Seq neural network where one network encodes the input clusters, and then another network is used to convert the encoded information to a visiting order of clusters as its output. Given the start position b_0 and K clusters, the input of the Seq2Seq model is $\boldsymbol{C} = \{b_0, G_1, \ldots, G_K\}$ and the output is the UAV's visiting order to these elements in $\boldsymbol{C}$, denoted as $\boldsymbol{Y} = \{\pi_0, \pi_1, \ldots, \pi_K\}$. Because the UAV takes off from b_0, b_0 should be in the first position of $\boldsymbol{Y}$. For keeping consistency of symbols, we use π_0 to represent b_0 in $\boldsymbol{Y}$. Hence, the probability of $\boldsymbol{Y}$, i.e., the probability that the UAV follows the corresponding



b_0 ··· b_r → b_k ···
π_0 ··· $\pi_{t-1}(G_r)$ $\pi_t(G_k)$ ··· π_K
$\boldsymbol{Y}$
$P(\pi_t|\cdot)$
Attention
g_t
a_t
h_t
Embedding
e_0 e_1 ··· e_k ··· e_K
b_0 G_1 ··· G_k ··· G_K
LSTM LSTM LSTM
Decoder
π_0 ··· π_{t-1} ··· π_{K-1}
b_1 b_2 b_3
$\pi_3(G_1)$ $\pi_2(G_3)$ $\pi_1(G_2)$ $\pi_0(b_0)$
G_1 G_2 G_3 b_0

Figure 3.2 Seq2Seq model with encoder-decoder framework.

trajectory, can be decomposed using the chain rule as follows

$$P_\theta(\boldsymbol{Y}|\boldsymbol{C}) = \prod_{t=0}^{K} P(\pi_t|\pi_0, \ldots, \pi_{t-1}, \boldsymbol{C}) \tag{3.19}$$

where t is the time step, $P(\pi_t|\cdot)$ models the probability of any cluster being visited at the t-th time step based on the clusters that have been visited at previous time steps and $\boldsymbol{C}$ [40]. Note that the *stochastic policy* $P_\theta(\boldsymbol{Y}|\boldsymbol{C})$ is parameterized by θ. In the following subsections, we will use the Seq2Seq neural network architecture in [41] to calculate the probability $P(\pi_t|\cdot)$.

3.3.2 Encoder-Decoder Framework for UAV's Trajectory

A typical Seq2Seq neural network includes an encoder and a decoder, where the encoder reads and arranges the input sequence into a vector, and the decoder outputs a target sequence by decoding this vector [42].

Encoder

Since the inputs are coordinates of the nodes of all clusters, which do not convey sequential information, we use a set of embeddings corresponding to different elements of the input as the encoder in our model instead of using RNNs. Specifically, the embeddings are to map the low-dimensional inputs to a high D-dimensional vector space. By doing so, the computational complexity of the embedding layer is reduced without reducing its efficiency. The mapping from the input $\boldsymbol{C}$ to the embedded output $\overline{\boldsymbol{C}}$ is shown as

$$\overline{\boldsymbol{C}} = W_b \boldsymbol{C} \tag{3.20}$$

where W_b is the embedding matrix, $\overline{\boldsymbol{C}} = \{e_k\}_{k=0}^{K}$, $e_k \in \mathbb{R}^D$. For example, in Fig. 3.2, there are three clusters and one start position, hence $\boldsymbol{C} = \{b_0, G_1, G_2, G_3\}$. After embedding, $\boldsymbol{C}$ is converted into $\overline{\boldsymbol{C}} = \{e_0, e_1, e_2, e_3\}$.

Decoder

Since the hidden units of the RNN can be used for learning historical information, it is very common in the literature to use the RNN as the decoder in the encoder-decoder framework. However, the traditional RNN shows poor performance in dealing with the problem of the long-term dependencies, which makes it difficult to be trained in practice [43]. Hence, we use the LSTM which is capable of learning long-term dependencies to construct a RNN as the decoder. The number of decoding steps is equal to the length of $\overline{\boldsymbol{C}}$. At each decoding step t, the hidden state $h_t \in \mathbb{R}^D$ of the LSTM, which stores information of previous steps, and the embedded $\overline{C}$ are used to generate the conditional probability $P(\pi_t|\pi_0, \ldots, \pi_{t-1}, \boldsymbol{C})$ for deciding the output in this step. Calculating the conditional probabilities is performed by the attention mechanism, whose details are described next.

Attention Mechanism

Attention mechanism is used to improve the encoder-decoder model, which allows the model to give different weights to different elements of the input [44]. For planning of the UAV's trajectory, attention mechanism tells us the relationship between each cluster in $\boldsymbol{C}$ at current step t and the output π_{t-1} of the last decoding step. The most relevant cluster with the maximum probability is chosen at decoding step t. Specifically, h_t is the hidden state of the LSTM at decoding step t. The quantity a_t^k represents how relevant each element e_k in

$\overline{\boldsymbol{C}}$ is at decoding step t. It is calculated using the softmax function[1] as

$$a_t^k = \text{softmax}\left(u_t^k\right) \tag{3.21}$$

where

$$u_t^k = \varphi_a \tanh\left(W_1 e_k + W_2 h_t\right), \tag{3.22}$$

with $\varphi_a \in \mathbb{R}^{1\times D}$, $W_1 \in \mathbb{R}^{D\times D}$, $W_2 \in \mathbb{R}^{D\times D}$. The context vector $g_t \in \mathbb{R}^D$ is computed as

$$g_t = \sum_{k=0}^{K} a_t^k e_k. \tag{3.23}$$

Then, we combine g_t with the embedded inputs

$$\widetilde{u}_t^k = \varphi_g \tanh\left(W_3 e_k + W_4 g_t\right) \tag{3.24}$$

with $\varphi_g \in \mathbb{R}^{1\times D}$, $W_3 \in \mathbb{R}^{D\times D}$, $W_4 \in \mathbb{R}^{D\times D}$. The vector $\widetilde{u}_t = \{\widetilde{u}_t^0, \widetilde{u}_t^1, \ldots, \widetilde{u}_t^k, \ldots, \widetilde{u}_t^K\} \in \mathbb{R}^{(K+1)}$ is called the logits. To encourage exploration, we use a logit clipping function to control the distribution of the logits

$$\bar{u}_t = C_{\mathrm{L}} \tanh\left(\widetilde{u}_t\right) \tag{3.25}$$

where C_{L} is a hyper-parameter that limits the range of the logits to $[-C_{\mathrm{L}}, C_{\mathrm{L}}]$, and hence, the entropy associated with $P(\cdot)$. The value of C_{L} is set to 10 by following [39]. To avoid clusters being visited more than once, we apply the mask vector to $\bar{u}_t$ to mark clusters that have been visited before:

$$\widehat{u}_t = \{\widehat{u}_t^0, \ldots, \widehat{u}_t^k, \ldots, \widehat{u}_t^K\} = \bar{u}_t + \mathcal{M}_t \tag{3.26}$$

where $\mathcal{M}_t \in \mathbb{R}^{(K+1)}$ is the mask vector, which is initialized to a vector of all zeros and its values are updated at each decoding step. If a cluster is selected for access, we update the value in the corresponding position of $\mathcal{M}_t$ to $-\infty$. Then, the element in the corresponding position of $\widehat{u}_t$ also becomes $-\infty$. Finally, we compute the probability of each element in $\widehat{u}_t$ as

$$P(\pi_t|\pi_0, \pi_1, \ldots, \pi_{t-1}, \boldsymbol{C}) = \text{softmax}\left(\widehat{u}_t\right) \tag{3.27}$$

[1] The softmax function is defined as: $a_t^k = \frac{\exp\{u_t^k\}}{\sum_{j=0}^{K}\exp\{u_t^j\}}$.

where the negative infinities in $\widehat{u}_t$ get zeroed out after using the softmax function. We choose the cluster pointed by the highest probability as the output at decoding step t and update the value of the same position in $\mathcal{M}_t$ to $-\infty$. Thus, each $P(\cdot)$ distribution is represented by the softmax function over all elements in the input sequence. The learnable variables are $\varphi_a, \varphi_g, W_1, W_2, W_3$, and W_4, which make up the policy parameter θ.

We further give an example to explain how the masking mechanism works. As shown in Fig. 3.2, there are three clusters and one start position; hence, $\boldsymbol{C} = \{b_0, G_1, G_2, G_3\}$. After embedding, $\boldsymbol{C}$ is converted into $\overline{\boldsymbol{C}} = \{e_0, e_1, e_2, e_3\}$. We assume that the outputs of the decoder network at decoding step 0 and 1 are b_0 and G_2, respectively. In order to get the output of decoding step 2, $\widetilde{u}_2 = \{\widetilde{u}_2^0, \widetilde{u}_2^1, \widetilde{u}_2^2, \widetilde{u}_2^3\}$ is obtained by equations (3.21)–(3.24), and the mask vector is $\mathcal{M}_2 = \{-\infty, 0, -\infty, 0\}$. By summing the elements of $\bar{u}_2$ and $\mathcal{M}_2$ at the same position, we can obtain $\widehat{u}_2 = \{\widehat{u}_2^0, \widehat{u}_2^1, \widehat{u}_2^2, \widehat{u}_2^3\} = \{-\infty, \bar{u}_2^1, -\infty, \bar{u}_2^3\}$. Applying equation (3.27) to $\widehat{u}_2$, we assume the final probability distribution over $\widehat{u}_2$ is calculated as $\{0, 0.2, 0, 0.8\}$. Hence, the cluster G_3 is selected at this step because the highest probability value points to it. Then, the mask vector is updated as $\mathcal{M}_2 = \{-\infty, 0, -\infty, -\infty\}$. As we can see, the masked clusters cannot be visited again. Hence, the masking scheme in our proposed algorithm can effectively prevent the clusters from being visited multiple times.

Selection of CHs

We assume that the output π_t of the model at step t is the cluster G_k, and its CH b_k is chosen by

$$b_k = \min\{E_{(b_r,n)}\}_{n=1}^{N} \tag{3.28}$$

where b_r is the CH of cluster G_r that is visited at step $(t-1)$, $E_{(b_r,n)}$ is the energy consumption of the UAV and the ground IoT network when the UAV flies from the CH b_r to a node n in the next cluster that will be visited. The node n in G_k that can guarantee the minimum energy consumption of the UAV-IoT from b_r to n is selected as the CH of the cluster G_k. In the example of Fig. 3.2, the start point b_0 is visited at the 0-th decoding step. Then, the output π_1 of the decoding step 1 is the cluster G_2 because it has the highest probability $P(\pi_1|\pi_0, \boldsymbol{C})$. We calculate the overall energy consumption of the UAV-IoT from b_0 to each node in G_2 and choose the CH by (3.28). Finally, we obtain a set of sorted CHs and output the UAV's trajectory, which is shown as

$$b_0(\pi_0) \rightarrow b_2(\pi_1) \rightarrow b_3(\pi_2) \rightarrow b_1(\pi_3).$$

The above trajectory may not be the best. Hence, we need to train the policy parameter θ from samples by RL to produce the optimal or close-to-optimal trajectory.

3.3.3 Training Method

In RL, an agent optimizes its behavior by interacting with the environment. The goal of the agent is to search for an optimal policy that can solve the constrained optimization problem through iterative training. All ground clusters, our objective function, and all constraints are considered as the environment. Note that for the agent, the environment is actually treated as a black box. The goal of the agent is to maximize the accumulated rewards by learning an optimal policy which is a mapping of states and actions.

State

The state of the problem at time step t is composed of the coordinates of all clusters, the location of the UAV, and the energy consumption.

Action

The action for the UAV at current step t is the selection of the next cluster and its CH to be visited. Hence, we define the output of the attention mechanism and the CH selection as the action at each step.

Reward

The reward function is defined as the negative of the total energy consumption in the formulated problem (3.18). The reward of one full episode generated under the policy is denote as $R = -E$.

REINFORCE [45], the well-known policy gradient, is employed in this paper, and the UAV works as the agent. Unlike value-based methods such as DQN that finds the optimal policy through Q-values, a policy gradient method directly optimizes the policy by changing its parameters. The REINFORCE algorithm uses an estimate of the gradient of the expected reward to update the policy parameter θ. The agent observes a full sequence that includes all states, actions, and rewards from start to finish generated under the policy. We compute the sum reward from this sequence by setting the discount factor to one, which is actually based on the real observed return. To train the proposed Seq2Seq model, the REINFORCE algorithm includes the actor network (policy network) and the critic network (value network). The Seq2Seq model works as the actor network that generates a set of ordered CHs for a given input problem instance. In the critic network, the output probabilities of the actor network are used to compute a weighted sum of the embedding inputs. Then, the weighted sum vector is fed into two-fully connected layers with one ReLU activation and one linear layer with a single output. The critic network, denoted by ψ, provides an approximated baseline of the solution for any problem instance to reduce the variance of gradients [46]. Given a problem instance $\boldsymbol{C}$, the training objective is the expected reward, which is defined as

$$J\left(\theta|\boldsymbol{C}\right) = \mathbb{E}_{\boldsymbol{Y}\sim p_{\theta(.|\boldsymbol{C})}}[R]. \tag{3.29}$$

One can use policy gradient and stochastic gradient descent to optimize θ. The gradient of (3.29) is formulated using REINFORCE algorithm, which can provide an unbiased gradient, as follows

$$\nabla_\theta J\left(\theta|\boldsymbol{C}\right) = \mathbb{E}_{\boldsymbol{Y}\sim p_{\theta(.|\boldsymbol{C})}}\left[\left(R - V_\psi\left(\boldsymbol{C}\right)\right)\nabla_\theta \log p_\theta\left(\boldsymbol{Y}|\boldsymbol{C}\right)\right] \tag{3.30}$$

where $V_\psi\left(\boldsymbol{C}\right)$ is a parametric baseline implemented by the critic network to reduce the variance of the gradient. We use batches to speed up the training process. Assuming there are B problem instances in each batch, the gradient in (3.30) is approximated with Monte Carlo sampling as

$$\nabla_\theta J\left(\theta\right) \approx \frac{1}{B}\sum_{i=1}^{B}\left(R_i - V_\psi\left(\boldsymbol{C}_i\right)\right)\nabla_\theta \log p_\theta\left(\boldsymbol{Y}_i|\boldsymbol{C}_i\right). \tag{3.31}$$

Algorithm 5 REINFORCE with the baseline algorithm

```
Input: Batch size $B$, training step $S$, training data set $\boldsymbol{\mathcal{Q}} = \{\boldsymbol{C}_1, \dots, \boldsymbol{C}_{S\times B}\}$
 1: Initialize the actor network parameter $\theta$ and the critic network parameter $\psi$
 2: for $s = 0$ to $S - 1$ do
 3:     Obtain train data $\boldsymbol{\mathcal{C}}_s = \{\boldsymbol{C}_{1+(s\times B)}, \dots, \boldsymbol{C}_{(s\times B)+B}\}$ from $\boldsymbol{\mathcal{Q}}$ for the current training step
 4:     Find CHs and calculate $R_i$ for each $\boldsymbol{C}_i$ in $\boldsymbol{\mathcal{C}}_s$ with
        the actor network
 5:     Calculate $V_\psi(\boldsymbol{C}_i)$ with the critic network
 6:     $d\theta \leftarrow \frac{1}{B} \sum_{i=1+(s\times B)}^{(s\times B)+B} (R_i - V_\psi(\boldsymbol{C}_i)) \nabla_\theta \log p_\theta(\boldsymbol{Y}_i|\boldsymbol{C}_i)$
 7:     $\mathcal{L}(\psi) \leftarrow \frac{1}{B} \sum_{i=1+(s\times B)}^{(s\times B)+B} (V_\psi(\boldsymbol{C}_i) - R_i)^2$
 8:     $\theta \leftarrow \mathrm{Adam}(\theta, d\theta)$
 9:     $\psi \leftarrow \mathrm{Adam}(\psi, \nabla_\psi \mathcal{L}\psi)$
10: end for
11: return $\theta$
```

The critic network is trained by using stochastic gradient descent on a mean squared error objective between $V_\psi(\boldsymbol{C}_i)$ and the actual energy consumption, which is given by

$$\mathcal{L}(\psi) = \frac{1}{B} \sum_{i=1}^{B} (V_\psi(\boldsymbol{C}_i) - R_i)^2 . \tag{3.32}$$

The training procedure of the actor network and the critic network is shown in Algorithm 5. The parameters of the actor and critic networks are updated iteratively by using the Adam algorithm [47].

3.4 Numerical Results

We compare the proposed approach with the following three common baseline methods:

1. Greedy: The greedy algorithm follows the problem-solving heuristic of making the locally optimal choice at each stage [48]. When looking for a solution, it always takes the best immediate or local decision, which may lead to poor solutions for some problems.

The greedy algorithm is usually faster than exact methods because it does not consider the details of possible alternatives.

2. Gurobi: The Gurobi optimizer is the most powerful mathematical optimization solver for linear programming, quadratic programming, mixed integer linear programming, mixed-integer quadratic programming, mixed-integer quadratically constrained programming, etc [49]. It is an exact algorithm solver that enables users to build mathematical models for their problems and produces the optimal solutions globally. For the optimization problem considered in this paper, the presented optimal solutions are obtained using Gurobi.

3. Ant colony optimization (ACO): ACO is a meta-heuristic method inspired by the observation of real ant colonies, which can be used to solve various combinatorial optimization problems. In ACO, multi-ants leave their nest and walk randomly until they find food. Each ant deposits a substance called pheromone along its trail so that the other ants can follow. An ant tends to choose the path with the highest pheromone concentration because its length is the shortest. Since our formulated problem is GTSP, we use the extended ACO method proposed in [50] to compare with our proposed DRL technique. In the following simulations, the parameters of ACO are set as follows. The number of ants is 30, the number of iterations is 200, the pheromone evaporation coefficient is set as 0.1, and the importance of pheromone and the relative importance of visibility are 1 and 5, respectively.

To thoroughly evaluate the performance of the proposed DRL algorithm, we compare the trajectories of the UAV and the energy consumptions obtained by the proposed algorithm with that obtained by three baseline methods. It should be noted that the energy consumption mentioned in all comparisons refers to the energy consumption in one communication round. In each round, member nodes send data to their CHs, and the UAV visits these CHs to collect data.

Table 3.1 Simulation parameters

Parameter	Value	Parameter	Value
P_{CH}	21 dBm/Hz [35]	r_p	20 cm [51]
B_{width}	1 MHz	n_p	4 [51]
N_0	-174 dBm/Hz [35]	ρ	1.225 kg/m^3 [51]
f_c	2 GHz [35]	m_{tot}	500 g [51]
α	3 [35]	P_{full}	5 W [51]
H	50 m	P_{s}	0 W [51]
μ_{LoS}, μ_{NLoS}	1 dB, 20 dB [52]	$v_{\text{UAV}} = v_{\text{full}}$	15 m/s [35]
β	0.03 [35]	P_{com}	0.0126 W [35]
η	10 [35]	N	20
ε_{fs}	10 pJ/bit/m^2 [37]	ε_{mp}	0.0013 pJ/bit/m^2 [37]
l	1 MB		

3.4.1 Complexity Comparison

In terms of computational complexity, the greedy algorithm has $O(KN)$ time complexity, which performs K steps to visit all clusters and consumes $O(N)$ operations to select a CH at each step. The computational complexity of ACO is estimated by $O(I_{\max}K^2M_{\text{ant}}N)$ where $I_{\max}$ is the number of iterations and M_{ant} is the number of ants [50]. At inference, the computational complexity of the attention mechanism in our proposed algorithm is $O(K+1)$ at each decoding step, and the computational complexity of selecting a CH is $O(N)$. We have to perform $K+1$ steps to output the final result, and hence, the total computational complexity of our proposed algorithm can be further simplified as $O((K+1)^2 + KN)) \approx O(K(K+N))$, which is lower than the complexity of the ACO. According to Gurobi's website [49], they use the branch-and-bound method to solve optimization problems. Thus,

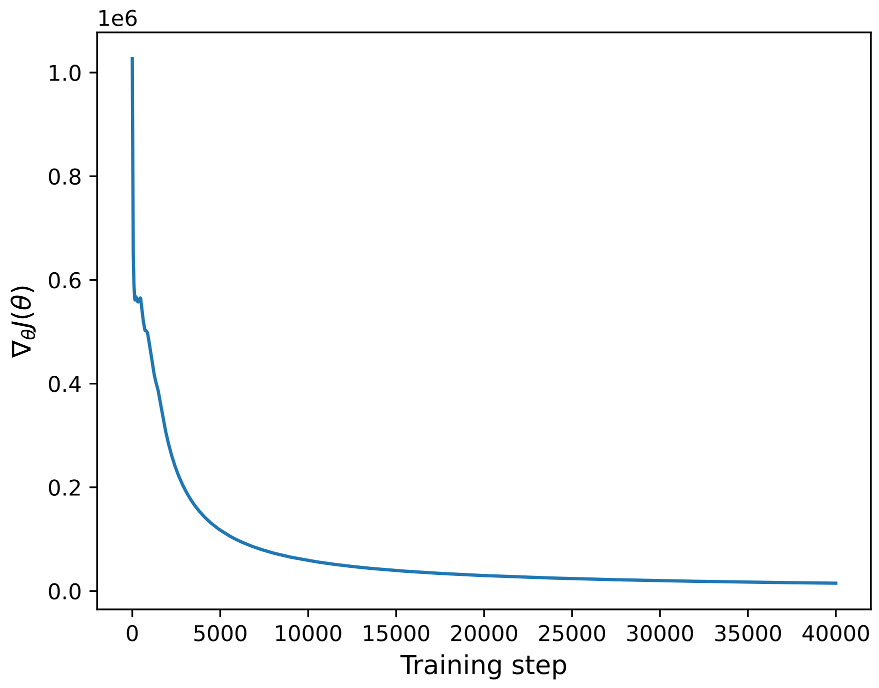


Figure 3.3 Training curve of the actor network.

the computational complexity of Gurobi is ultimately exponential, which is worse than our proposed algorithm.

3.4.2 Environment and Parameters Settings

We consider a target area of $1\,\mathrm{km} \times 1\,\mathrm{km}$ where the BS is located with coordinate $b_0 = (500\,\mathrm{m}, 0\,\mathrm{m})$. Simulation parameters are presented in Table 3.1. We employ Pytorch 1.4 and Python 3.7 on a computer with 1 NVIDIA TESLA P100 GPU to implement the proposed DRL algorithm. Each problem instance $\boldsymbol{C}$ is composed of the initial location b_0 and K clusters. The center (x_k, y_k) of each cluster G_k is firstly sampled from the distribution $torch.rand(2, 1) * 1000$. Then, the nodes in each cluster are sampled from the uniform distribution $G_k = np.random.uniform([x_k - \zeta, y_k - \zeta], [x_k + \zeta, y_k + \zeta], [N, 2])$, where ζ is a constant, $[x_k - \zeta, y_k - \zeta]$ represents the left and lower boundaries of cluster G_k in the 2-dimensional space, $[x_k + \zeta, y_k + \zeta]$ is the right and upper boundaries, $0 < x_k - \zeta < x_k + \zeta < 1000$, $0 < y_k - \zeta < y_k + \zeta < 1000$, and all clusters do not overlap with each other, i.e., $G_1 \cap \cdots \cap G_k \cap \cdots \cap G_K = \emptyset$. The final training data set $\boldsymbol{\mathcal{Q}}$ is composed of all sampled problem instances. Setting $K = 4$, we implement 40,000 training steps to train the model where the batch size B is equal to 256 at each training step. Each element in any problem instance $\boldsymbol{C}$ is embedded into a vector of size 128 by the encoder network. Accordingly, we use LSTM cells with 128 hidden units in the decoder network. The initial learning rate of the actor network and the critic network is set at 0.0001.

Fig. 3.3 shows the training curve of the actor network. One can see that the value of $\nabla_\theta J(\theta)$ decreases sharply in early steps, which is due to the rough approximation at initialization that causes a large loss. When the number of iterations increases, $\nabla_\theta J(\theta)$ stabilizes and the proposed algorithm converges.

3.4.3 Trajectory and Energy Consumption Comparison

To show the effectiveness of the proposed algorithm, we compare its performance with the performances of the greedy algorithm, ACO, and Gurobi in this section. We generate a problem instance with four clusters $\{G_1, G_2, G_3, G_4\}$ in the same way as generating the train data. Then, the test data is fed into the trained model to evaluate the proposed DRL algorithm.

In Fig. 3.4 (a), the value of ω in (4.21) is set to 0, which means that the goal is to minimize the energy consumption of UAV only, which is proportional to UAV's flight distance in our system model. Since the greedy algorithm only yields locally optimal solutions by visiting the nearest next CH as shown in Fig. 3.4 (a), it will not achieve the shortest UAV trajectory. However, the trajectory generated by our proposed DRL algorithm completely coincides with the optimal one obtained from Gurobi, which ensures the energy consumption of the UAV is minimum. In addition, there is a visible gap between the trajectory generated by ACO and the optimal trajectory.

For the results in Fig. 3.4 (b), we set $\omega = 0.3$, which means that the energy consumptions of ground nodes and UAV account for 30% and 70% of the total energy consumption, respectively. In order to minimize the total energy consumption in this case, the CH of each cluster should be between the center and the edge of the cluster and close to the edge. Obviously, all four algorithms can select CHs in the right position as well as plan trajectories

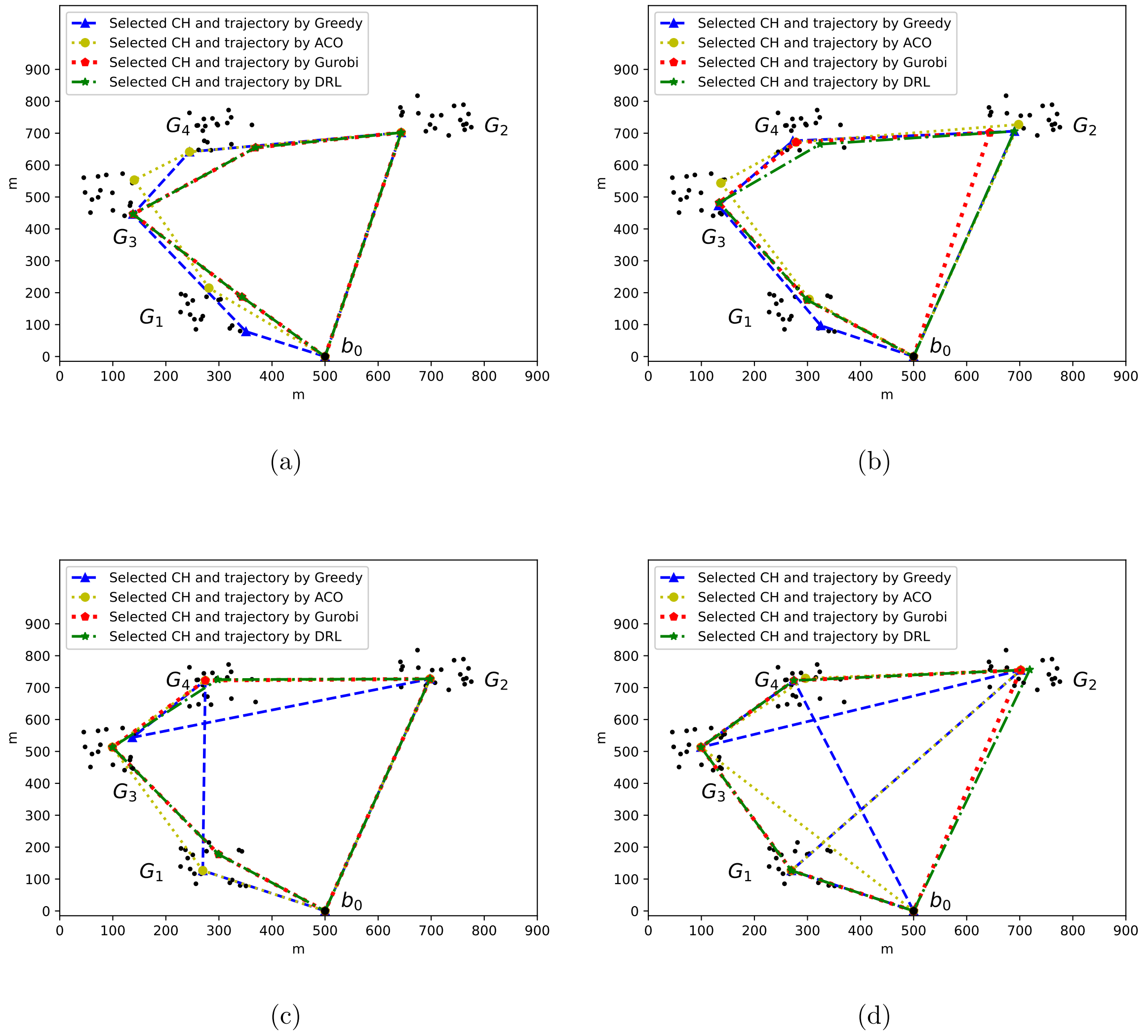


Figure 3.4 Trajectory comparison of DRL, greedy algorithm, ACO, and Gurobi for different values of ω. (a) $\omega = 0$. (b) $\omega = 0.3$. (c) $\omega = 0.6$. (d) $\omega = 0.9$.

to access these CHs. However, the trajectory obtained by our proposed DRL algorithm and the one by Gurobi are almost identical, which exhibits our proposed algorithm can produce the close-to-optimal solution.

The results in Fig. 3.4 (c) are obtained by setting $\omega = 0.6$, which means that the energy consumption of the ground nodes accounts for a larger proportion of the total energy consumption. As a consequence, it is expected that CHs should be closer to the center of

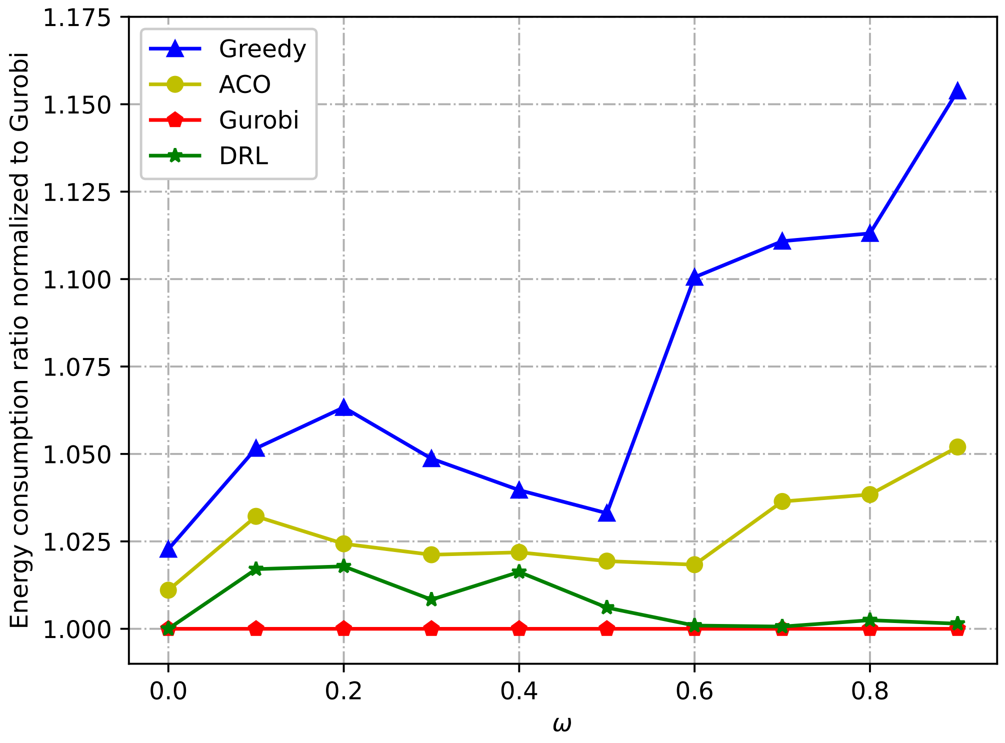


Figure 3.5 Energy consumption comparison for 4 clusters.

the cluster. The visiting path produced by the greedy algorithm is

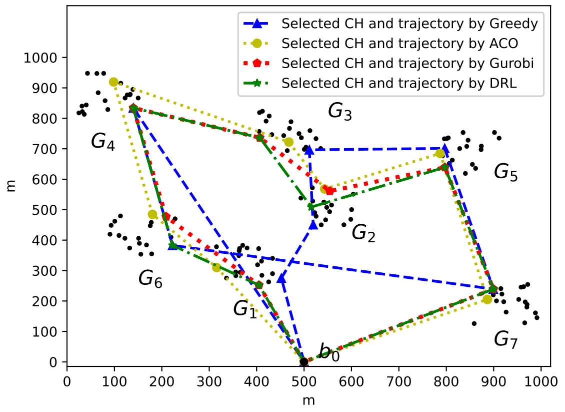
$$b_0 \to G_1 \to G_4 \to G_3 \to G_2 \to b_0$$

which travels suitable CHs, but does not present the optimal path to access these CHs. The trajectory produced by ACO is much better than the one of the greedy algorithm, which is given by

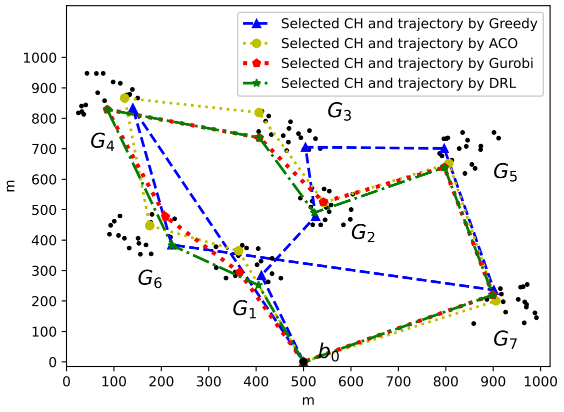
$$b_0 \to G_1 \to G_3 \to G_4 \to G_2 \to b_0.$$

However, our algorithm not only can find the appropriate CHs but also plan the optimal access path to these CHs. The trajectories found by the proposed DRL algorithm and Gurobi almost coincide again, as can be seen in Fig. 3.4 (c).

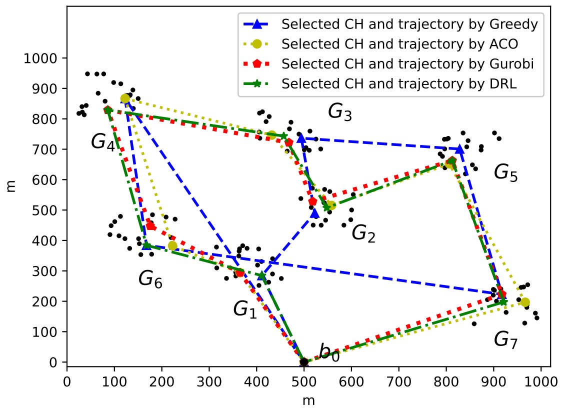
For the results in Fig. 3.4 (d), the value of ω is set to 0.9. The greedy algorithm determines CHs and the trajectory according to its local "greedy" strategy, and it finally



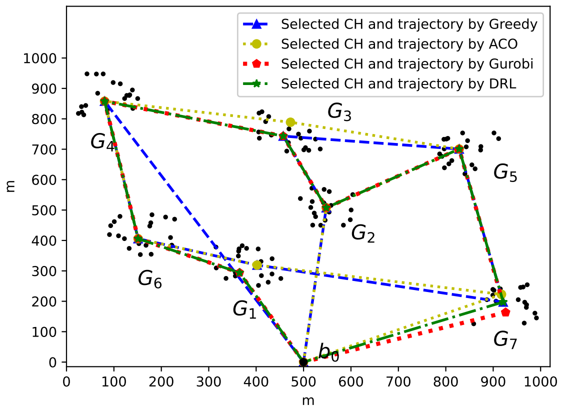
(a)



(b)



(c)



(d)

Figure 3.6 Trajectory comparison of DRL, greedy algorithm, ACO, and Gurobi for different values of ω. (a) $\omega = 0.1$. (b) $\omega = 0.3$. (c) $\omega = 0.5$. (d) $\omega = 0.8$.

produces the below access order to the four clusters

$$b_0 \to G_4 \to G_3 \to G_2 \to G_1 \to b_0.$$

The access order to clusters obtained by ACO is

$$b_0 \to G_1 \to G_2 \to G_4 \to G_3 \to b_0$$

which is better than the order obtained with the greedy algorithm, and inferior to the one found by our proposed DRL algorithm. As for the proposed DRL algorithm, the access order to four clusters is found to be

$$b_0 \rightarrow G_1 \rightarrow G_3 \rightarrow G_4 \rightarrow G_2 \rightarrow b_0$$

which also nearly coincides with the trajectory obtained by Gurobi. Through the above four cases, it is clear that our DRL algorithm can find optimal or nearly optimal trajectories when compared with the trajectories found by Gurobi, and it also performs much better than the greedy and the ACO algorithms.

Based on the above simulation results, we present a more detailed analysis of our proposed algorithm in Fig. 3.5. Specifically, this figure plots the ratios of the energy consumptions of our proposed DRL, the greedy and the ACO algorithms to the energy consumptions of Gurobi which are all normalized to one at different values of ω. The results are averaged over 30 test instances. It can be clearly seen that the energy consumption of our proposed DRL algorithm is very close to the optimal value obtained by Gurobi and less than that of the ACO and the greedy algorithms for all different values of ω. As expected, the ACO algorithm outperforms the greedy algorithm in reducing the energy consumption.

3.4.4 Trajectory and Energy Consumption Comparison on the 7-Clusters IoT Network

In this subsection, we generate a 7-clusters problem instance using the same way as generating the training data, and observe the obtained trajectories on the previously trained 4-clusters model and the three baselines. As shown in Fig. 3.6, our proposed DRL algorithm can find the appropriate CH from each cluster even when the value of ω changes. In addition, it can plan the access trajectory to CHs under different ω, as follows

$$b_0 \rightarrow G_1 \rightarrow G_6 \rightarrow G_4 \rightarrow G_3 \rightarrow G_2 \rightarrow G_5 \rightarrow G_7 \rightarrow b_0.$$

We can see that the trajectories obtained with the proposed DRL algorithm are very close or fully coincide with the optimal trajectories generated by Gurobi in four cases of Fig. 3.6. However, the greedy algorithm shows the worst trajectory planning ability. In Fig. 3.6 (a), (b), (c), the UAV's access order to clusters by the greedy algorithm is given by

$$b_0 \rightarrow G_1 \rightarrow G_2 \rightarrow G_3 \rightarrow G_5 \rightarrow G_7 \rightarrow G_6 \rightarrow G_4 \rightarrow b_0$$

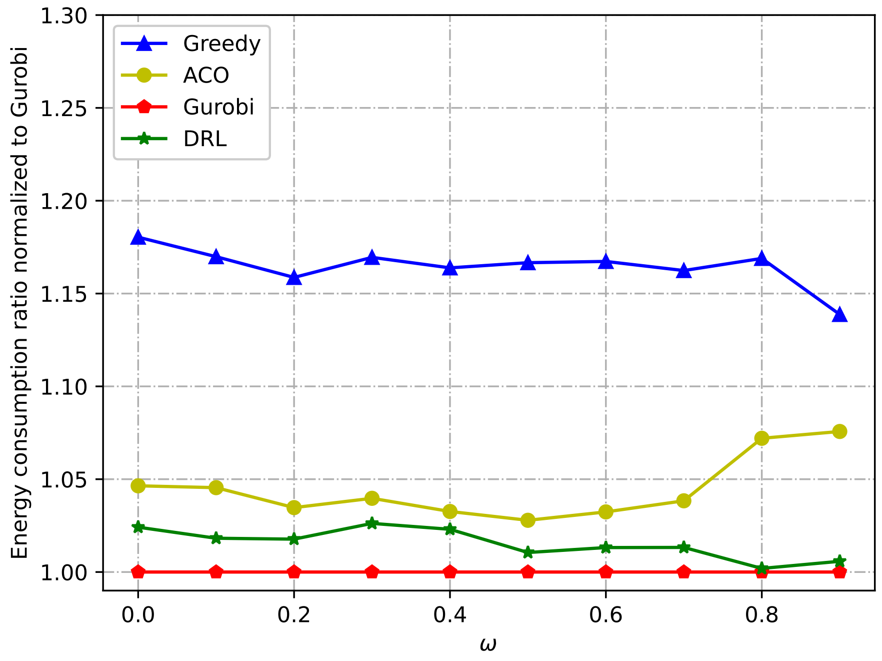


Figure 3.7 Energy consumption comparison for 7 clusters.

and in Fig. 3.6 (d), the access order is

$$b_0 \rightarrow G_2 \rightarrow G_3 \rightarrow G_5 \rightarrow G_7 \rightarrow G_1 \rightarrow G_6 \rightarrow G_4 \rightarrow b_0.$$

As for the ACO algorithm, when compared to the greedy and our proposed DRL algorithms, it can plan a reasonable trajectory to 7 clusters as shown in Fig. 3.6 (a), (b), (c)

$$b_0 \rightarrow G_1 \rightarrow G_6 \rightarrow G_4 \rightarrow G_3 \rightarrow G_2 \rightarrow G_5 \rightarrow G_7 \rightarrow b_0$$

but in Fig. 3.6 (d), its trajectory turns worse, which is given by

$$b_0 \rightarrow G_2 \rightarrow G_5 \rightarrow G_3 \rightarrow G_4 \rightarrow G_6 \rightarrow G_1 \rightarrow G_7 \rightarrow b_0.$$

The trajectory comparison results show that the model trained by our proposed DRL algorithm has good scalability and generalization abilities to plan the trajectories for new problem instance without retraining the model.

In Fig. 3.7, energy consumption comparison over the averaged results of 30 test instances shows that our proposed DRL algorithm can achieve close-to-minimum energy consumption

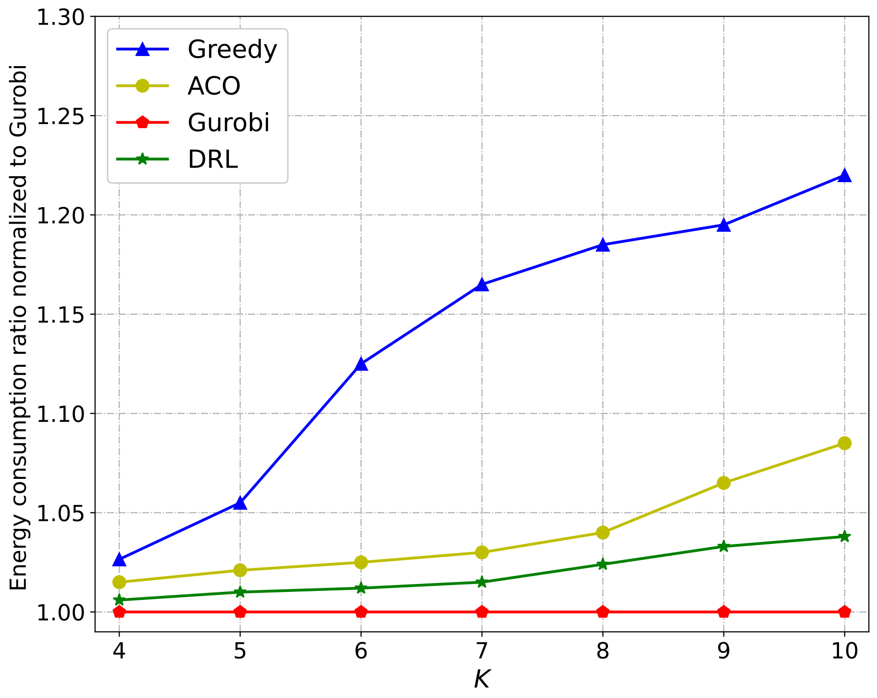


Figure 3.8 Energy consumption comparison when K varies.

obtained by Gurobi, and performs better than the greedy and the ACO algorithms. From the above analysis, one can see that the trained model by a large amount of 4-clusters problem instances can plan a near-optimal trajectory on 7-clusters problem instance, without retraining the model for the test data. This is consistent with the fact that RNNs used in our model have been shown to have very good scalability and generalization [53].

3.4.5 Further Investigation for the Generalization Ability

The scalability and generalization abilities of the trained model is further studied when K varies, and the results are shown in Fig. 3.8 for $\omega = 0.5$. Since Gurobi is the exact solver, it always obtains the optimal solutions at different values of K. It is clear that the performance gap between the other three algorithms and Gurobi gradually increases as the value of K increases. However, our proposed DRL algorithm clearly exhibits a superior performance than the greedy and the ACO algorithms in terms of saving the energy consumption. Table 3.2 compares the running times of different algorithms. As the number of clusters increases,

Table 3.2 Running time comparison.

		Time			
		Greedy	ACO	Gurobi	DRL
K	4	0.18 s	6.92 s	1800 s	0.36 s
	5	0.19 s	10.15 s	2711 s	0.37 s
	6	0.22 s	14.33 s	3722 s	0.39 s
	7	0.25 s	19.23 s	5405 s	0.39 s
	8	0.29 s	25.82 s	6908 s	0.41 s
	9	0.32 s	31.21 s	9701 s	0.41 s
	10	0.33 s	38.14 s	12500 s	0.42 s

the computation times of all four algorithms increase. Although Gurobi obtains the best performance in reducing the energy consumption according to the previous simulation results, it takes the most computational time to deliver the optimal results. The computation time of our DRL algorithm is slightly higher than that of the greedy algorithm, but significantly less than that of the ACO algorithm and Gurobi.

3.5 Conclusion

In this paper, we have investigated the problem of jointly designing the UAV's trajectory and selecting CHs for an IoT network to minimize the total energy consumption in the UAV-IoT system. Inspired by the promising development of DRL, we propose a novel DRL-based method to solve this problem. In our proposed method, DRL with a Seq2Seq neural network is used to learn the policy of the trajectory planning with the aim of minimizing the total weighted energy consumption of the UAV-IoT system. Extensive simulation results demonstrated that our proposed method outperforms the ACO and greedy algorithms in planning the UAV's trajectory and achieves nearly optimal results when compared to the results obtained by the Gurobi optimizer. In addition, our proposed DRL algorithm has excellent abilities of generalization, scalability, and automation to solve different problem instances with different numbers of clusters, without retraining the model for new problems. Considering computation times and the energy consumption results, our proposed method offers an appealing balance between performance and complexity.

Acknowledgement

This work was supported by an NSERC/Cisco Industrial Research Chair in Low-Power Wireless Access for Sensor Networks.

References

[1] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of IoT: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things Journal*, vol. 1, pp. 349–359, Aug. 2014.

[2] H. Xie, Z. Yan, Z. Yao, and M. Atiquzzaman, "Data collection for security measurement in wireless sensor networks: A survey," *IEEE Internet of Things Journal*, vol. 6, pp. 2205–2224, Apr. 2019.

[3] K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 3567–3569, Aug. 2018.

[4] Q. Zhang, M. Liu, X. Lin, Q. Liu, J. Wu, and P. Xia, "Optimal resonant beam charging for electronic vehicles in internet of intelligent vehicles," *IEEE Internet of things journal*, vol. 6, pp. 6–14, Feb. 2018.

[5] C. Tseng, C. Cheng, Y. Hsu, and B. Yang, "An IoT-based home automation system using Wi-Fi wireless sensor networks," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2430–2435, Oct. 2018.

[6] D. Wei, L. Yan, C. Huang, J. Wang, J. Chen, M. Pan, and Y. Fang, "Dynamic magnetic induction wireless communications for autonomous-underwater-vehicle-assisted underwater IoT," *IEEE Internet of Things Journal*, vol. 7, pp. 9834–9845, Aug. 2020.

[7] Y. Kim, E. Lee, and H. Park, "Ant colony optimization based energy saving routing for energy-efficient networks," *IEEE Communications Letters*, vol. 15, pp. 779–781, Jul. 2011.

[8] D. Lin, W. Min, and J. Xu, "An energy-saving routing integrated economic theory with compressive sensing to extend the lifespan of WSNs," *IEEE Internet of Things Journal*, vol. 7, pp. 7636–7647, Aug. 2020.

[9] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 34–46, Jan. 2019.

[10] Z. Wang, R. Liu, Q. Liu, J. S. Thompson, and M. Kadoch, "Energy-efficient data collection and device positioning in UAV-assisted IoT," *IEEE Internet of Things Journal*, vol. 7, pp. 1122–1139, Feb. 2019.

[11] B. Liu and H. Zhu, "Energy-effective data gathering for UAV-aided wireless sensor networks," *Sensors*, vol. 19, p. 2506, May 2019.

[12] J. Baek, S. I. Han, and Y. Han, "Energy-efficient UAV routing for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 1741–1750, Feb. 2019.

[13] C. Zhan, Y. Zeng, and R. Zhang, "Energy-efficient data collection in UAV enabled wireless sensor network," *IEEE Wireless Communications Letters*, vol. 7, pp. 328–331, Jun. 2018.

[14] Q. Song, S. Jin, and F. Zheng, "Completion time and energy consumption minimization for UAV-enabled multicasting," *IEEE Wireless Communications Letters*, vol. 8, pp. 821–824, Jun. 2019.

[15] Y. Zeng, J. Xu, and R. Zhang, "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, pp. 2329–2345, Apr. 2019.

[16] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 2059–2070, Sep. 2018.

[17] D. H. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, "Coarse trajectory design for energy minimization in UAV-enabled," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 9483–9496, Sep. 2020.

[18] D. Rojas Viloria, E. L. Solano-Charris, A. Muñoz-Villamizar, and J. R. Montoya-Torres, "Unmanned aerial vehicles/drones in vehicle routing problems: A literature review," *International Transactions in Operational Research*, vol. 28, pp. 1626–1657, Mar. 2020.

[19] K. Zhu, X. Xu, and S. Han, "Energy-efficient UAV trajectory planning for data collection and computation in mMTC networks," in *Proc. IEEE Globecom Workshops*, pp. 1–6, Dec. 2018.

[20] B. Zhang, C. H. Liu, J. Tang, Z. Xu, J. Ma, and W. Wang, "Learning-based energy-efficient data collection by unmanned vehicles in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1666–1676, Apr. 2018.

[21] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 5994–6006, Sep. 2021.

[22] R. Ding, F. Gao, and X. S. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 7796–7809, Dec. 2020.

[23] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten, "Energy minimization in UAV-aided networks: Actor-critic learning for constrained scheduling optimization," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 5028–5042, May 2021.

[24] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers Operations Research*, vol. 134, p. 105400, Oct. 2021.

[25] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multi-agent systems: A review of challenges, solutions, and applications," *IEEE Transactions on Cybernetics*, vol. 50, pp. 3826–3839, Sep. 2020.

[26] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, pp. 405–421, Apr. 2021.

[27] H. Hu, X. Zhang, X. Yan, L. Wang, and Y. Xu, "Solving a new 3D bin packing problem with deep reinforcement learning method," *arXiv preprint arXiv:1708.05930*, 2017.

[28] J. Lu, L. Feng, J. Yang, M. M. Hassan, A. Alelaiwi, and I. Humar, "Artificial agent: The fusion of artificial intelligence and a mobile agent for energy-efficient traffic control in wireless sensor networks," *Future Generation Computer Systems*, vol. 95, pp. 45–51, Jun. 2019.

[29] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 3806–3817, Oct. 2019.

[30] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 51, pp. 3103–3114, Jun. 2021.

[31] R. Solozabal, J. Ceberio, A. Sanchoyerto, L. Zabala, B. Blanco, and F. Liberal, "Virtual network function placement optimization with deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 38, pp. 292–303, Feb. 2020.

[32] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 9540–9554, Sep. 2021.

[33] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "Improved soft-k-means clustering algorithm for balancing energy consumption in wireless sensor networks," *IEEE Internet of Things Journal*, vol. 8, pp. 4868–4881, Mar. 2021.

[34] J. Yao and N. Ansari, "Qos-aware power control in internet of drones for data collection service," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 6649–6656, Jul. 2019.

[35] M. B. Ghorbel, D. Rodríguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, "Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2165–2175, Mar. 2019.

[36] R. Roberti and P. Toth, "Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison," *EURO Journal on Transportation and Logistics*, vol. 1, pp. 113–133, Jun. 2012.

[37] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, Mar. 2002.

[38] W. B. Heinzelman, *Application-specific protocol architectures for wireless networks.* PhD thesis, Massachusetts Institute of Technology, 2000.

[39] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR) Workshop*, pp. 1–5, Apr. 2017.

[40] I. Bello, S. Kulkarni, S. Jain, C. Boutilier, E. Chi, E. Eban, X. Luo, A. Mackey, and O. Meshi, "Seq2slate: Re-ranking and slate optimization with RNNs," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–12, May 2019.

[41] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Advances Neural Information Processing Systems (NIPS)*, pp. 9861–9871, Dec. 2018.

[42] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 28, pp. 2692–2700, Dec. 2015.

[43] Y. Zhu, X. Dong, and T. Lu, "An adaptive and parameter-free recurrent neural structure for wireless channel prediction," *IEEE Transactions on Communications*, vol. 67, pp. 8086–8096, Nov. 2019.

[44] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, Jun. 2015.

[45] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[46] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Advances Neural Information Processing Systems (NIPS)*, pp. 1008–1014, Dec. 1999.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–13, May 2015.

[48] F. B. Mismar, B. L. Evans, and A. Alkhateeb, "Deep reinforcement learning for 5G networks: Joint beamforming, power control, and interference coordination," *IEEE Transactions on Communications*, vol. 68, pp. 1581–1592, Mar. 2020.

[49] Gurobi, "Math programming modeling basics," 2021. [Online]. Available: https://www.gurobi.com/resource/mip-basics/.

[50] J. Yang, X. Shi, M. Marchese, and Y. Liang, "An ant colony optimization method for generalized TSP problem," *Progress in Natural Science*, vol. 18, pp. 1417–1422, Nov. 2008.

[51] H. Ghazzai, M. Ben Ghorbel, A. Kadri, M. J. Hossain, and H. Menouar, "Energy-efficient management of unmanned aerial vehicles for underlay cognitive radio systems," *IEEE Transactions on Green Communications and Networking*, vol. 1, pp. 434–443, Dec. 2017.

[52] A. Al-Hourani, S. Kandeepan, and S. Lardner, "Optimal lap altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, pp. 569–572, Dec. 2014.

[53] Z. Tu, F. He, and D. Tao, "Understanding generalization in recurrent neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–16, Apr. 2020.

4. UAV Trajectory Planning in Wireless Sensor Networks for Energy Consumption Minimization by Deep Reinforcement Learning

Published as:

B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9540–9554, Sep. 2021.

In Chapter 3, the total energy consumption minimization problem in a UAV-IoT system is formulated to jointly design the UAV's trajectory and select CHs for the IoT network. The formulated problem is viewed as a Seq2Seq decision problem and solved by the DRL algorithm with a sequential model. The proposed DRL algorithm can achieve close-to-optimal performance on small-scale-clusters networks when compared to other baseline algorithms.

In this chapter, we continue to investigate the total energy consumption minimization problem in an UAV-WSN network. Because the visiting order of the UAV to ground clusters can be regarded as a sequential decision problem, we design a novel DRL-based method, called pointer network-A* (Ptr-A*), to find the trajectory of the UAV. The UAV's start point and all clusters, as the input, are fed into the Ptr-A* model, and its output is a set of CHs and the visiting order to these CHs, i.e., the UAV's trajectory. Specifically, the pointer network of the proposed Ptr-A* model is used to determine the visiting order to clusters. Then, a search graph for all clusters is built according to the visiting order. The

A* search algorithm is utilized to quickly find the CH for each cluster from the search graph with the aim of minimizing the energy consumption of the UAV-WSN system. In order to obtain optimal parameters of the Ptr-A*, we employ the model-free RL method to train the proposed Ptr-A* model in an unsupervised manner. The proposed Ptr-A* model shows good performance not only on small-scale-clusters networks, but also on large-scale-clusters networks.

UAV Trajectory Planning in Wireless Sensor Networks for Energy Consumption Minimization by Deep Reinforcement Learning

Botao Zhu, Ebrahim Bedeer, Ha Nguyen,

Robert Barton, and Jerome Henry

Abstract

Unmanned aerial vehicles (UAVs) have emerged as a promising candidate solution for data collection of large-scale wireless sensor networks (WSNs). In this paper, we investigate a UAV-aided WSN, where cluster heads (CHs) receive data from their member nodes, and a UAV is dispatched to collect data from CHs. We aim to minimize the total energy consumption of the UAV-WSN system in a complete round of data collection. Toward this end, we formulate the energy consumption minimization problem as a constrained combinatorial optimization problem by jointly selecting CHs from clusters and planning the UAV's visiting order to the selected CHs. The formulated energy consumption minimization problem is $\mathcal{NP}$-hard, and hence, hard to solve optimally. To tackle this challenge, we propose a novel deep reinforcement learning (DRL) technique, pointer network-A* (Ptr-A*), which can efficiently learn the UAV trajectory policy for minimizing the energy consumption. The UAV's start point and the WSN with a set of pre-determined clusters are fed into the Ptr-A*, and the Ptr-A* outputs a group of CHs and the visiting order of CHs, i.e., the UAV's trajectory. The parameters of the Ptr-A* are trained on small-scale clusters problem instances for faster training by using the actor-critic algorithm in an unsupervised manner. Simulation results show that the trained models based on 20-clusters and 40-clusters have a good generalization ability to solve the UAV's trajectory planning problem in WSNs with different numbers of clusters, without retraining the models. Furthermore, the results show that our proposed DRL algorithm outperforms two baseline techniques.

Index Terms

Combinatorial optimization, deep reinforcement learning, trajectory planning, UAV, WSN.

4.1 Introduction

The use of unmanned aerial vehicles (UAVs) has recently attracted a lot of attention from both the research community and industry. UAVs have been used for a variety of purposes [1], such as environmental monitoring, mobile cloud computing, disaster management, security operations, and wireless power transfer, to name a few. The popularity and widespread applications of UAVs are due to their many advantages, such as cost-effectiveness, having line-of-sight (LoS) links with the ground devices, mobility, and reliable network access [2].

In order to overcome the limitations of the tradition wireless sensor networks (WSNs), extensive research has been done on the integration of UAVs and WSNs for long-distance mission communications where UAVs are considered as mobile sinks for receiving data from cluster heads (CHs), and then, they can transmit the collected data to terrestrial BSs for further processing. Using UAVs as mobile sinks can reduce the energy consumption of ground nodes in WSNs when compared to the traditional multi-hop WSNs which transmit data from each node to the sink node over a long distance or several hops [3]. Despite the advantages, the integration of UAVs and WSNs still grapples with many challenges. First of all, due to the limited energy source carried by UAVs, the service range of UAVs is constrained by the reality that they cannot travel very long distances or fly for long periods of time. Second, battery life of sensor nodes in WSNs is typically limited, and in many cases it is hard to regularly replace their batteries. As a result, frequent communication with UAVs can cause sensor nodes to exhaust their energy rapidly. Hence, it is important to study the energy saving problem in UAV-enabled WSNs.

4.1.1 Motivation

Prior works on the energy consumption minimization for UAV-enabled WSNs can be classified into three categories depending on the objectives. The first category only considers minimizing the UAV's energy consumption, e.g., [4, 5]. In contrast, the second category considers only minimizing the energy consumption of the ground devices in the UAV-aided wireless networks, e.g., [6, 7]. In the third category, the energy of both the UAV and the ground devices are taken into account when minimizing the energy consumption of the UAV-enabled system, e.g., [8, 9]. However, most of the aforementioned studies assume the UAV directly communicates with each device of the ground wireless network. In this case, if the UAV flies over all devices in a large-scale WSN, it would lead to a long flight trajectory for the UAV which increase its energy consumption. As a result, the UAV may run out of its energy in flight or may need to recharge its battery frequently.

Motivated by the aforementioned works, in this paper we investigate the problem of minimizing the total energy consumption of the UAV and the ground devices in a clustered WSN, which has not been well researched in prior works. We assume that devices on the ground have been clustered according to some specific criterion, e.g., based on their geographical locations; hence, clustering techniques for WSNs will not be discussed in this paper. In each pre-determined cluster in our system model, one of the ground devices will be selected as the CH, which is responsible for collecting data from the non-CH devices in the same cluster. Hence, the UAV only needs to visit a set of CHs for gathering data along the planned trajectory that is determined by locations of the ground CHs. The selection of CHs affects both the energy consumption of the ground devices and the UAV, as shown in the following illustrative example.

Consider the case where nodes of three clusters are deployed in a given area as shown in Fig. 4.1. There are two candidate solutions of potential CHs. One possible solution, i.e., trajectory 1, selects the "center" node of each cluster as the CH, such as cc_1, cc_2, cc_3, and the other possible solution, i.e., trajectory 2, selects non-center nodes as the CHs, e.g. c_1, c_2, c_3. The start/end location of the UAV is (0, 0). If the UAV chooses to follow trajectory 1, the energy consumption between CHs and their member nodes will be minimal because the Euclidean distances between member nodes and their CHs are minimal (on average) per cluster [10]. However, this will lead to an increase in energy consumption of the UAV as its flight trajectory may be longer, and hence, may not be optimal from the start point to the

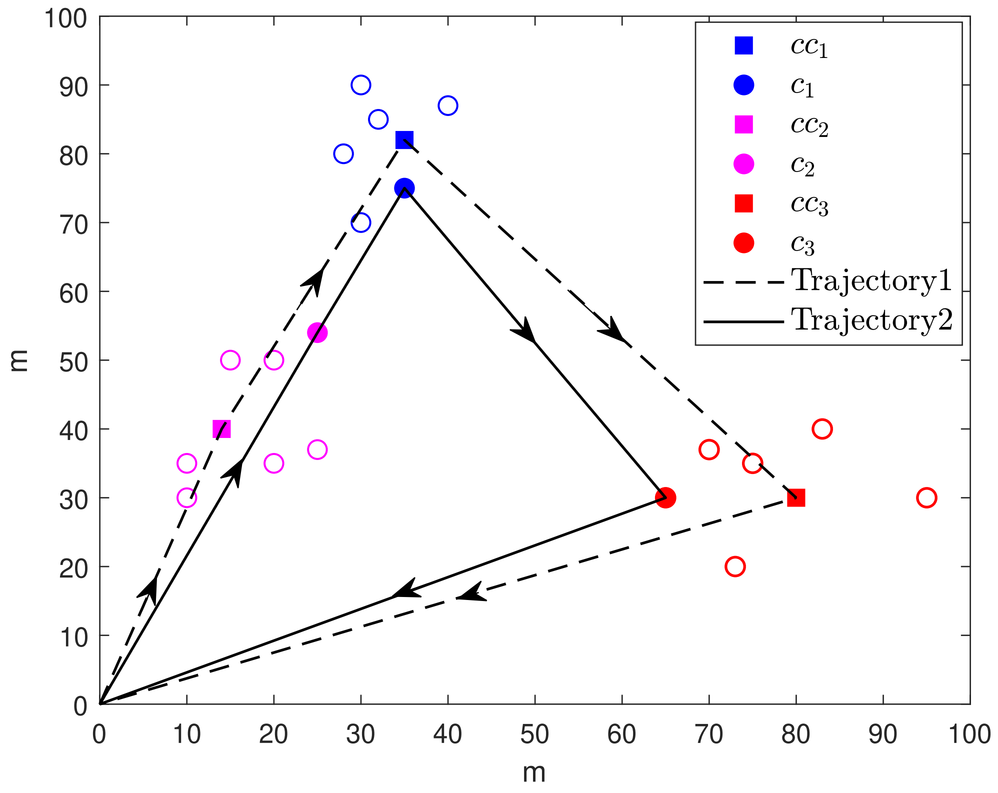


Figure 4.1 Comparison of different UAV's trajectories.

end point, which can be seen from trajectory 1 in Fig. 1. On the contrary, if the UAV goes through trajectory 2, it will consume less energy in flight because trajectory 2 is shorter than trajectory 1. But, the communication energy consumption between CHs and their member nodes will be higher in this case. From the above simple example and discussion, it is clearly important and relevant to study the energy-efficient UAV's trajectory planning in clustered WSNs in order to minimize the overall energy consumption of the UAV and the ground network.

4.1.2 Related Works

UAV Trajectory Planning

Energy-efficient trajectory planning for UAVs has recently attracted significant research interest, and multiple solutions have been proposed for UAV-enabled wireless networks. In general, existing solutions for energy-efficient UAV trajectory planning can be loosely classified into two categories: non-machine learning-based methods and machine learning-based methods. In the first category, researchers mostly use mathematical programming or heuristic algorithms to solve the trajectory optimization problem. However, the computation time of mathematical programming algorithms may increase exponentially as the problem size increases, e.g., [11, 12]. Although some heuristic algorithms are applied to design the energy-efficient path in the UAV-enabled wireless networks, such as ant colony optimization [13] and cuckoo search [14], they usually cannot fully adapt to the increasing complexity of scalable wireless networks.

Regarding the machine learning-based category, deep reinforcement learning (DRL) and reinforcement learning (RL) are the most common techniques in solving the UAV's trajectory planning. In [15], the authors propose a DRL-based method which is composed of two deep neural networks (DNNs) and deep deterministic policy gradient (DDPG) to maximize the energy efficiency for a group of UAVs by jointly considering communications coverage, energy consumption, and connectivity. In order to minimize the UAV's transmission and hovering energy, the authors in [16] formulate the energy-efficient optimization problem as a Markov decision process. Then, they use two DNNs and the actor-critic-based RL algorithm to develop an online DRL algorithm that shows a good performance in terms of energy savings. In [17], the authors jointly optimize the UAV's 3D trajectory and the frequency band allocation of ground users by considering the UAV's energy consumption and the fairness of the ground users. A DDPG-based DRL algorithm is developed to generate the energy-efficient trajectory with fair communication service to ground users. In [18], with the aim of designing an energy-efficient UAV's route for long-distance sensing tasks, the authors propose a DRL-based framework where convolutional neural networks (CNNs) are used for extracting features and the deep Q-network (DQN) is utilized to make decisions. Towards realizing green UAV-enabled Internet of Things (IoT), the authors in [19] formulate the UAV's path planning problem as a dynamic decision optimization problem, which is solved by dueling DQN. The aforementioned machine learning-based methods show strong ability to handle complex wireless environments and effectively learn the UAV's trajectory policy from experiences; however, they are implemented by using some common neural network models, such as DNNs, DQN, and CNNs. Instead of common neural network models, in this paper we exploit the appealing concept of sequence-to-sequence learning that originally emerged in *Natural Language Processing* to design the DRL algorithm to solve the UAV's path planning problem in clustered WSNs.

Sequence-to-Sequence Learning

Sequence-to-sequence learning has shown great success in machine translation where sentences are mapped to correct translations [20]. Over the last several years, many neural networks based on sequence-to-sequence models have been proposed in different applications. For example, pointer network is one of the extensively studied models because of its excellent ability in solving sequence decision problems. In [21], pointer network is trained in a supervised fashion to solve the traveling salesman problem (TSP). The work of [22] uses a RL-based unsupervised method to train the pointer network and obtains better results when compared to the supervised learning in [21]. In [23], the authors propose a structural graph embedded pointer network to develop online vehicular routes in intelligent transportation systems. The authors in [24] propose a modified pointer network to solve the keyword recommendation problem in sponsored search advertising system. In [25], a simplified pointer network is introduced to solve Vehicle Routing Problem (VRP) in dynamic traffic environments. Different from the above-discussed research works, we extend the state-of-the-art pointer network-based DRL to solve the UAV's trajectory planning problem. Our contributions are elaborated in the next subsection.

4.1.3 Contributions

We aim to minimize the overall UAV-WSN's energy consumption by designing an efficient UAV's trajectory in a clustered WSN. Since the UAV's visiting order to the ground CHs can be seen as a sequence decision problem, we propose a sequence-to-sequence learning-based DRL strategy with pointer network to deal with the challenging trajectory planning problem. The pointer network can capture the relation between a problem instance and its solution by using a sequence-to-sequence neural network. It has been demonstrated to be an effective method to solve some $\mathcal{NP}$-hard problems, such as TSP [22] and VRP [25]. Hence, it is expected that the pointer network-based DRL algorithm is also promising for solving the problem of the UAV trajectory planning for the UAV-WSN system. The main contributions of this paper are summarized as follows:

1. We consider a UAV-enabled energy-efficient data collection framework for clustered WSNs. We formulate an optimization problem to minimize the energy consumption of the entire UAV-WSN system by jointly designing the UAV's trajectory and selecting CHs in pre-determined clusters of the ground WSN.

2. We show that the UAV's trajectory planning problem in the clustered WSN can be seen as a sequence of decisions. Hence, a sequence-to-sequence pointer network-A* (Ptr-A*) model is proposed to solve the formulated problem. Particularly, the pointer network is utilized to model the visiting order of all ground clusters, and A* algorithm [26] is used to efficiently select CHs from clusters' ground nodes. The UAV's start point and all clusters, as the input, are fed into the Ptr-A* model, and its output is a set of CHs and the visiting order to these CHs, i.e., the UAV's trajectory.

3. We use a self-driven learning mechanism that only needs the reward calculation to train the parameters of Ptr-A* network on problem instances with small-scale clusters for faster training.

4. Our proposed DRL method has an excellent generalization capability with respect to the number of clusters used for training. In other words, given a new problem instance with any number of clusters, the trained model can automatically generate a trajectory for the UAV to visit clusters, without retraining the new model.

5. We perform extensive simulations to demonstrate that the proposed DRL method outperforms other baseline techniques when considering both the computation times and energy consumption results.

The rest of this paper is organized as follows. Section 4.2 presents the system model and the problem formulation. Section 4.3 describes the proposed DRL algorithm. Section 4.4 provides simulation results. Finally, Section 4.5 concludes the paper.

4.2 System Model and Problem Formulation

As mentioned earlier, we assume that devices on the ground have been clustered according to some specific criterion, e.g., based on their geographical locations; hence, clustering techniques for the ground WSN are not discussed in this work. In particular, we consider K clusters of sensor nodes $\{\boldsymbol{G}_1, \ldots, \boldsymbol{G}_K\}$ located in the sensing (service) area for data collection. Each cluster contains N nodes, one of which is the CH, represented by $b_k \in \boldsymbol{G}_k$, that will be selected by our proposed algorithm. We assume that only one rotary-wing UAV is dispatched to visit CHs to collect data from the ground network. The UAV takes off from the start position b_0 and then back to b_0 after finishing the data collection task. The trajectory of the UAV should contain the start/end hovering position c_0 corresponding to b_0, and the K target hovering positions $\{c_1, \ldots, c_K\}$, which are vertically above the ground CHs. Hence, the trajectory planning problem of the UAV can be seen as a permutation of $(K+1)$ hovering positions. It is obvious that the locations of CHs determine the flight trajectory, and hence, the energy consumption of UAV and the ground nodes. We consider a three-dimensional (3D) Cartesian coordinates system to define the positions of ground nodes and the UAV. The position of the CH of the k-th cluster is $b_k = (x_k, y_k, 0)$. Correspondingly, the coordinate of the UAV's hovering position c_k can be represented by (x_k, y_k, H), where H is the fixed flight height of the UAV. Similarity, the coordinate of the n-th member node of the k-th cluster $b_k^{(n)}$ is denoted as $\left(x_k^{(n)}, y_k^{(n)}, 0\right), n = 1, \ldots, N-1,\ k = 1, \ldots, K$, and $b_k^{(n)} \neq b_k$.

4.2.1 Channel Model

There is a number of channel models that have been developed for UAV communications, e.g., [27,28]. In this work, we consider a simple air-to-ground channel model that is described as follows. For ground-to-air communication, there is a certain probability that each CH b_k has a LoS view towards the UAV when it hovers at the hovering position c_k. This probability typically depends on the environment and elevation angle, and is given by [8]

$$P_{\text{LoS}} = \frac{1}{1 + \eta \exp\left(-\beta[\tau - \eta]\right)}, \tag{4.1}$$

where η and β are constants determined by environment, and $\tau = \frac{180}{\pi} \times \sin^{-1}\left(\frac{H}{d_k}\right)$, where d_k is the distance between b_k and c_k. Since it is assumed that each hovering position is directly above the corresponding CH, one has $d_k = H$. Obviously, the non-line-of-sight (NLoS) probability is given by $P_{\text{NLoS}} = 1 - P_{\text{LoS}}$. The average path loss between each CH and the UAV can be expressed as [8]

$$\overline{P}_{\text{loss}} = P_{\text{LoS}}\left(K_0 + \mu_{\text{LoS}}\right) + P_{\text{NLoS}}\left(K_0 + \mu_{\text{NLoS}}\right) \tag{4.2}$$

where μ_{LoS} and μ_{NLoS} are the mean values of the excessive path losses in LoS and NLoS links, respectively, $K_0 = 10\alpha \log_{10}\left(\frac{4\pi f_c H}{c}\right)$, α is the path loss exponent, c is the speed of light, and f_c is the carrier frequency. Thus, the average data rate from each CH to the UAV can be computed as [8]

$$r_{\text{data}} = B_{\text{width}} \log_2\left(1 + \frac{P_{\text{CH}}}{\overline{P}_{\text{loss}} N_0}\right) \tag{4.3}$$

where B_{width} is the available bandwidth, N_0 is the noise power spectral density, and P_{CH} is the transmit power of each CH.

4.2.2 UAV's Energy and Trajectory Model

We assume that the UAV supports a flying-hovering mode without considering acceleration-deceleration patterns. After the UAV flies to the hovering position c_k with a fixed speed v_{UAV}, it hovers there and transfers a beacon frame to wake up the corresponding CH b_k from sleep mode to active model. Then, b_k starts to collect data from its member nodes by time-division multiple access (TDMA) and forwards the collected data to the UAV. At each hovering position, the energy consumption of the UAV includes two parts: communication-related energy and hover-related energy. The hovering power is given by [8], [29]

$$P_{\text{hover}} = \sqrt{\frac{\left(m_{\text{tot}} g\right)^3}{2\pi r_p^2 n_p \rho}} \tag{4.4}$$

where g is the earth gravity, ρ is the air density, n_p is the number of propellers, r_p is the propeller radius, and finally m_{tot} is the mass of the UAV. Thus, the energy consumed by the

UAV at each hovering position c_k is given by

$$\begin{aligned} E_{c_k} &= T_k(P_{\text{hover}} + P_{\text{com}}) \\ &= \frac{D_k}{r_{\text{data}}}(P_{\text{hover}} + P_{\text{com}}) \end{aligned} \tag{4.5}$$

where T_k is the total hovering time of the UAV at c_k, D_k is the amount of data that needs to be transferred from CH b_k to the UAV, and P_{com} is the communication power of the UAV. In order to simplify the analysis, we assume that the hovering time is equal to the data transmission time from b_k to the UAV.

The horizontal movement power is assumed as a linear function of the UAV's flight speed v_{UAV}, which is expressed as [8], [29]

$$P_{\text{move}} = \frac{P_{\text{max}} - P_{\text{idle}}}{v_{\text{max}}} v_{\text{UAV}} + P_{\text{idle}} \tag{4.6}$$

where v_{max} is the maximum speed of the UAV, P_{max} and P_{idle} are the hardware power levels when the UAV is moving at full speed and when the UAV is in idle state, respectively. Because the UAV needs to start from the start hovering location c_0, goes through all target hovering positions $c_1, \ldots, c_K$, and then back to c_0, the total energy consumption of the UAV in flight is given by [8], [29]

$$E_{\text{flight}} = T_{\text{flight}} \left(P_{\text{hover}} + P_{\text{move}}\right) \tag{4.7}$$

where T_{flight} is the total flight time, which can be expressed as

$$T_{\text{flight}} = \frac{1}{v_{\text{UAV}}} \sum_{i=0}^{K} \sum_{\substack{j=0 \\ j \neq i}}^{K} d_{c_i,c_j} L_{c_i,c_j}, \quad \forall c_i, c_j \in \boldsymbol{C} \tag{4.8}$$

where $\boldsymbol{C} = \{c_0, c_1, \ldots, c_K\}$, c_k is determined by b_k, $b_k \in \boldsymbol{G}_k$, and L_{c_i,c_j} specifies whether the UAV travels from stop position c_i to c_j, which is defined as

$$L_{c_i,c_j} = \begin{cases} 1, & \text{if the path goes from } c_i \text{ to } c_j \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

The quantity d_{c_i,c_j} is the Euclidean distance between c_i and c_j, which is given by

$$d_{c_i,c_j} = ||c_i - c_j|| = ||b_i - b_j||. \tag{4.10}$$

In order to meet the requirements of the UAV's trajectory, we need to consider the following constraints:

$$\sum_{\substack{i=0 \\ i\neq j}}^{K} L_{c_i,c_j} = 1, \quad \forall c_i, c_j \in \boldsymbol{C} \tag{4.11}$$

$$\sum_{\substack{j=0 \\ j\neq i}}^{K} L_{c_i,c_j} = 1, \quad \forall c_i, c_j \in \boldsymbol{C} \tag{4.12}$$

$$\sum_{c_i \in \boldsymbol{F}} \sum_{c_j \in \boldsymbol{F}} L_{c_i,c_j} \leq |\boldsymbol{F}| - 1, \quad \forall \boldsymbol{F} \subset \boldsymbol{C}; |\boldsymbol{F}| \geq 2. \tag{4.13}$$

The constraints (4.11) and (4.12) guarantee that there is only one UAV path entering and leaving a given node, which means that the UAV should visit each point in $\boldsymbol{C}$ exactly once. Constraint (4.13) is the sub-trajectories elimination constraint and enforces that no partial loop exists where $\boldsymbol{F}$ is the subset of $\boldsymbol{C}$ [30], which means there is only one single trajectory covering all CHs.

According to the above analysis, the total energy consumption of the UAV is composed of the flying-related and the hovering-related energy consumption, which can be written as

$$E_{\text{UAV}} = E_{\text{flight}} + \sum_{k=1}^{K} E_{c_k}. \tag{4.14}$$

4.2.3 Ground Network and Energy Model

We assume that all nodes have the same computation and transmission capabilities. In other words, all nodes are capable of acting as a CH. Nodes are static after being deployed. All member nodes transmit their sensing information to CHs periodically and CHs forward the collected data to the UAV. We also assume that the transmission energy of each node is sufficient to send messages to its CH. In addition, the UAV can simultaneously connect to at most one CH. Hence, there is no interference among neighboring CHs.

The energy consumption in the ground network includes two components. The first component is the communication energy consumption between CHs and their member nodes. The first-order radio model [31] is used to calculate the energy consumption of the ground network. The transmission energy is consumed by the transmitter's circuitry and power amplifier. If the distance between a member node and its CH is less than a given threshold, the power amplifier uses the free space model; otherwise, the multi-path model is used [32]. The energy consumed to transmit an l-bit message from a member node to its CH b_k is given by [31]

$$E_n^{b_k} = lE_{\text{elec}} + l\left(\chi\varepsilon_{\text{fs}}d_{n,b_k}^2 + (1-\chi)\,\varepsilon_{\text{mp}}d_{n,b_k}^4\right) \tag{4.15}$$

where

$$\chi = \begin{cases} 1, & d_{n,b_k} \le d_0 \\ 0, & d_{n,b_k} > d_0 \end{cases} \tag{4.16}$$

and

$$d_0 = \sqrt{\frac{\varepsilon_{\text{fs}}}{\varepsilon_{\text{mp}}}}. \tag{4.17}$$

In (4.15), E_{elec} is the dissipated energy per bit in the circuitry, d_{n,b_k} is the distance between the CH b_k and one of its member nodes n, $n = 1, \ldots, N-1$, d_0 is the distance threshold, ε_{fs} and ε_{mp} represent the radio amplifier's energy parameter of the free space and multi-path fading models, respectively. Moreover, the energy consumed to receive an l-bit message from member node n by b_k is given by [31]

$$E_{b_k}^{(n)} = lE_{\text{elec}}. \tag{4.18}$$

In addition, the second component of the energy consumption of the ground network is the energy consumed by each CH b_k to complete its data transmission to the UAV. This component can be written as

$$E_{b_k} = P_{\text{CH}}T_k = P_{\text{CH}}\frac{(N-1)l}{r_{\text{data}}} \tag{4.19}$$

where $(N-1)l$ is the amount of data transferred by b_k to the UAV. Hence, the total energy consumption of all nodes in the ground network in a complete data collection task, where member nodes transmit the sensing data to their CHs and CHs forward data to the UAV, is

$$E_{\text{ground}} = \sum_{k=1}^{K}\sum_{n=1}^{N-1}\left(E_n^{b_k} + E_{b_k}^{(n)}\right) + \sum_{k=1}^{K} E_{b_k}. \tag{4.20}$$

4.2.4 Problem Formulation for UAV's Trajectory

Based on (4.14) and (4.20), the total weighted energy consumption in the UAV-WSN system can be formulated as

$$\begin{aligned} E = \omega &\left(\sum_{k=1}^{K} \sum_{n=1}^{N-1} \left(E_n^{b_k} + E_{b_k}^{(n)} \right) + \sum_{k=1}^{K} E_{b_k} \right) \\ &+ (1-\omega) \left(E_{\text{flight}} + \sum_{k=1}^{K} E_{c_k} \right), \; 0 \le \omega \le 1 \end{aligned} \tag{4.21}$$

where the first term corresponds to the total energy consumption of the ground network, while the second term is the energy consumption of the UAV, and ω is the weighting coefficient that can be adjusted to achieve the trade-off between the two terms. With the aim of minimizing the total energy consumption of the ground network and the UAV, we jointly find a set of CHs from the ground cluster-based WSN and design the UAV's visiting order to these CHs. The optimization problem of interest is formulated as

$$\begin{aligned} &\min_{\substack{\{b_0, b_1, \ldots, b_k, \ldots, b_K\} \\ b_k \in \boldsymbol{G}_k}} \quad E \\ &\text{s.t.}\, (4.9), (4.11) - (4.13). \end{aligned} \tag{4.22}$$

Obviously, the problem at hand is a constrained combinatorial optimization problem, which is $\mathcal{NP}$-hard. Some promising approaches have been put forward to solve such combinatorial optimization problems, and their advantages and disadvantages are discussed below.

1. *Exact methods*: Exact methods often search for the optimal solution of the problem through systematic enumeration, integer programming, and constraint programming, etc. [33]. At least in theory, they can provide the optimal solution for the optimization problem. However, such algorithms cannot be applied to combinatorial optimization problems with large data scale because their computation complexity becomes prohibitive.

2. *Heuristics*: Heuristics are higher-level problem-independent algorithmic frameworks

that provide a set of guidelines to develop optimization algorithms [34]. However, they generally cannot guarantee to find globally optimal solutions.

3. *RL*: Q-learning, one of the RL techniques, is demonstrated to be promising in solving $\mathcal{NP}$-hard problems [35]. Specifically, it can deal with the path decision problem when provided with sufficient state space variables. However, if the number of ground nodes is high, Q-learning will need more storage space for action and state space variables [18].

As discussed before, DRL has recently shown to have important advantages in solving combinatorial optimization problems. A typical neural combinatorial framework is proposed in [22] that uses RL to optimize a policy modeled by the pointer network. In [25], the authors view a combinatorial optimization problem as a sequence of decisions, and they use a sequence-to-sequence neural network model and the RL approach to obtain a near-optimal solution for the optimization problem. Inspired by these promising developments, we extend the application of sequence-to-sequence model to solve the UAV's trajectory planning problem described earlier.

4.3 Deep Reinforcement Learning for UAV Trajectory Planning

Because all clusters $\{\boldsymbol{G}_1, \ldots, \boldsymbol{G}_K\}$ must be visited by the UAV sequentially, we convert the visiting decisions problem into a sequence-to-sequence prediction problem. The problem can be simply formalized as follows. Given start position and all clusters, denoted by $\boldsymbol{\mathcal{G}} = \{b_0, \boldsymbol{G}_1, \ldots, \boldsymbol{G}_K\}$, we want to output a permutation of the items in $\boldsymbol{\mathcal{G}}$ that maximizes some measure of interest. The output sequence is denoted as $\boldsymbol{\mathcal{T}} = \{\pi_0, \pi_1, \ldots, \pi_K\}$, where each π_t is the index of any element in $\boldsymbol{\mathcal{G}}$ being placed at the t-th position of $\boldsymbol{\mathcal{T}}$. In fact, $\boldsymbol{\mathcal{T}}$ is the UAV's visiting order to clusters in our problem. Thus, for a given input sequence $\boldsymbol{\mathcal{G}}$, the probability of the output sequence $\boldsymbol{\mathcal{T}}$ can be factorized by a product of conditional probabilities according to the chain rule

$$P_\theta(\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{G}}) = \prod_{t=0}^{K} P(\pi_t|\pi_0, \ldots, \pi_{t-1}, \boldsymbol{\mathcal{G}}) \tag{4.23}$$

where t is the time step, $P_\theta(\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{G}})$ parameterized by θ is a *stochastic policy* for deciding the visiting order. The conditional probability $P(\pi_t|\cdot)$ models the probability of any cluster

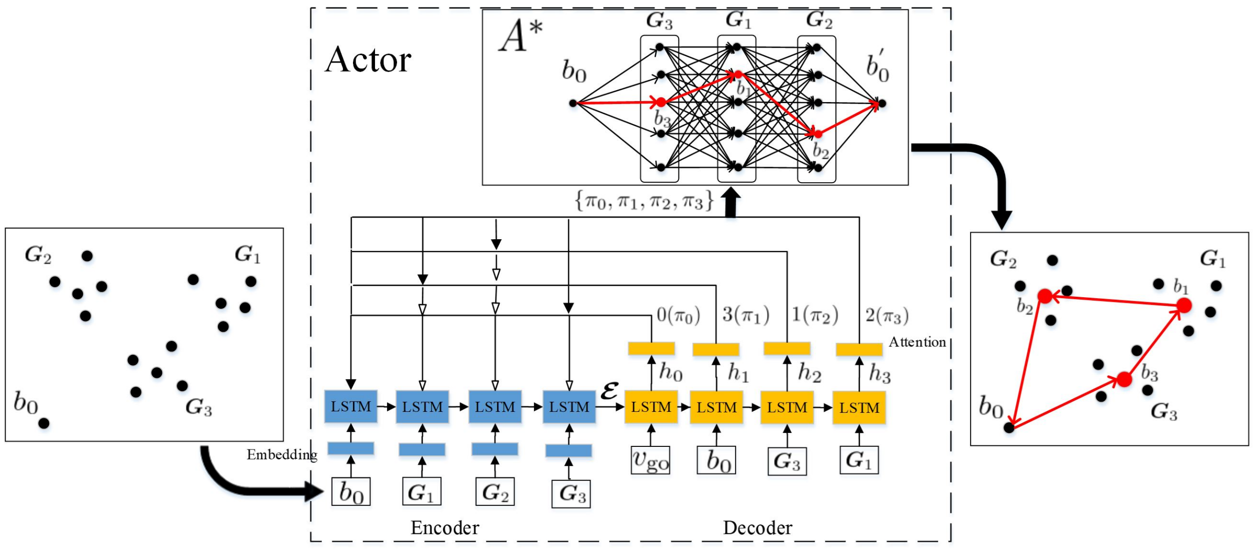


Figure 4.2 Example of Ptr-A* architecture for a 3-clusters network.

being visited at the t-th time step according to the given $\boldsymbol{\mathcal{G}}$ and clusters already visited at previous time steps [36]. A trained θ can assign high probabilities to good results and low probabilities to bad results. The reinforcement learning can be applied to train the optimal model policy θ^* for producing the optimal visiting order $\boldsymbol{\mathcal{T}}^*$ with the highest probability.

4.3.1 Pointer Network-A* Architecture for UAV's Trajectory Planning

With the rapid development of neural network techniques, the neural network-based frameworks have been applied to sequence-to-sequence learning [20]. The general sequence-to-sequence neural network [25] encodes the input sequence into a vector that includes information of the input by a recurrent neural network (RNN), called encoder, and decodes the vector to the target sequence by another RNN, called decoder [21]. In this work, we employ the pointer network to model the conditional probability $P(\pi_t|\cdot)$, which has been proved to be effective to solve the combinatorial optimization problems. The architecture of the pointer network is similar to sequence-to-sequence network model, but it uses attention mechanism as a pointer to choose items of its input sequence as the output. The proposed Ptr-A* model in this work is elaborated as follows.

Encoder

The traditional RNN shows poor performance in dealing with the problem of long-term dependencies, which makes it difficult to be trained in practice [37]. Hence, we use Long Short-Term Memory (LSTM) cells which are capable of learning long-term dependencies to construct a RNN as the encoder. Each item (a start position or a cluster) in $\boldsymbol{\mathcal{G}}$ is converted into a high D-dimensional vector space, which enables the policy to extract useful features much more efficiently in the transformed space [38]. Then, the embedding vectors are fed into LSTM cells. At each encoding step, the LSTM cell reads one embedded item and outputs a latent memory state. Finally, the input sequence $\boldsymbol{\mathcal{G}}$ is transformed into a sequence of latent memory states $\boldsymbol{\mathcal{E}} = \{e_0, \ldots, e_K\}$, each $e_k \in \mathbb{R}^D$. In fact, the motive of the encoder network is to acquire the representation for each element in $\boldsymbol{\mathcal{G}}$.

Decoder

We also adopt LSTM cells to construct the RNN of the decoder network. The output $\{e_0, \ldots, e_K\}$ of the encoder are given to the decoder network. At each decoding step t, the LSTM cell outputs the hidden state $h_t \in \mathbb{R}^D$ that includes the knowledge of previous steps. And then, the decoder employs the attention mechanism to output the visiting decision π_t based on h_t and $\{e_0, \ldots, e_t\}$. Attention mechanism can help the model to give different weights to different elements of the input and extract more critical information [39]. It tells us the relationship between each element in the input at current step t and the output π_{t-1} of the last decoding step. The most relevant element with the maximum conditional probability is selected as the access element at decoding step t. Thus, the calculation is given by

$$u_j^t = \begin{cases} \varphi \tanh\left(W_1 e_j + W_2 h_t\right), & \text{if } j \notin \{\pi_0, \ldots, \pi_{t-1}\} \\ -\infty, & \text{otherwise} \end{cases} \tag{4.24}$$

where W_1, $W_2 \in \mathbb{R}^{D\times D}$ are attention matrices, $\varphi \in \mathbb{R}^{1\times D}$ is the attention vector. W_1, W_2, and φ are denoted collectively by θ, which is the learnable parameter in our pointer network. In essence, u_j^t is the score associated with item j (e_j) in position t.

The conditional probability is calculated by a *softmax* function over the remaining items (not visited in the previous steps), as follows:

$$\begin{aligned} P(\pi_t = j | \pi_0, \ldots, \pi_{t-1}, \boldsymbol{\mathcal{G}}) &= \operatorname{softmax}\left(u_j^t\right) \\ &= \frac{\exp\left(u_j^t\right)}{\sum_{m \notin \{\pi_0, \ldots, \pi_{t-1}\}} \exp\left(u_m^t\right)}, \, j \in m. \end{aligned} \tag{4.25}$$

The probability $P\left(\pi_t = j|\cdot\right)$ represents the degree to which the model points to item j at the decoding step t [36]. As shown in the example in Fig. 4.2, the start position b_0 and clusters $\boldsymbol{G}_1, \boldsymbol{G}_2, \boldsymbol{G}_3$ are inputted into the encoder network. v_{go} is the start tag of the decoder, which is a learned vector. At each decoding step, the item of the input sequence with the highest probability is pointed by a thicker black arrow. The output of the 0-th decoding step points to $b_0\left(\pi_0\right)$, which will be visited at this step and given as the input of the next decoding step. Finally, we will obtain a visiting order sequence $\boldsymbol{\mathcal{T}} = \{\pi_0, \pi_1, \pi_2, \pi_3\}$ corresponding to the input sequence.

To help the reader familiarize with the attention mechanism, a numerical example is provided in Appendix A.

A* search

Once the output sequence of the decoder network is obtained, we can build a search graph for all clusters according to this sequence, where each layer is composed of nodes of one cluster. This is illustrated with an example in Fig. 4.2. It is worth mentioning that the first layer is the start position b_0 and the last layer is the end position $b_0^{'}$ which is the copy of b_0. Thus, the created graph has a total of (K+2) layers. We use the A* search algorithm, one of best path-finding algorithms, to find the CH from each cluster to build a path having the smallest cost (total weighted energy consumption of the UAV-WSN system) from the start position to the end position. In each iteration, the A* algorithm needs to calculate the cost of the traversed path and the estimated cost required to extend the path to the end to determine which of its partial paths to expand into one or more longer paths [26]. Any node m is chosen to be visited by the following function

$$f(m) = g(m) + h(m) \tag{4.26}$$

where $g(m)$ represents the exact energy consumption of the UAV-WSN system when the UAV moves from the start node to a candidate node m, following the path generated to get there, $h(m)$ is the estimated energy consumption of the UAV to travel from the candidate node m to the end. Then, the node with the lowest $f(m)$ value is selected from candidate nodes as the next node to be traversed.

The main implementation of the A* algorithm is to maintain two lists. The OPEN list contains those nodes that are candidates for checking. The CLOSED list contains those nodes that have been checked. The neighbor nodes of any node located in any layer are defined as all nodes in its previous and next layers. Also, each node keeps a pointer to its parent node so that we can determine how it was found, which is implemented by a map COME_FROM. The pseudocode of using the A* algorithm to find the path from the start position to the end position is described in Algorithm 1. As shown in the example in Fig. 4.2, the output of the A* algorithm is the trajectory from b_0 to $b_0^{'}$, which can ensure the minimum energy consumption E. In addition, the CH of each cluster is found on this trajectory, given by $\{b_0, b_3, b_1, b_2, b_0^{'}\}$. Finally, the Ptr-A* model outputs the trajectory and the minimum energy consumption E.

4.3.2 Parameters Optimization with Reinforcement Learning

In order to find a good trajectory for the UAV, we need to obtain the optimal model parameter θ^* that can be trained from samples. If we adopt a supervised learning to train the model parameter, high-quality labeled data is needed because it decides the performance of the model. However, it is expensive to get the high-quality labeled data in practice for the proposed UAV's trajectory problem. Instead, we choose the well-known model-free policy-based RL, known as the actor-critic algorithm [40], to train the model because it is shown to be an appropriate paradigm for training neural networks for combinatorial optimization [22]. The UAV works as the agent to make a sequential action set in a given state of the environment. In the following, we describe the state, action, reward, and training of the proposed DRL algorithm.

Algorithm 6 A* search algorithm for the trajectory planning

```
Input: $\boldsymbol{\mathcal{T}}$
Output: Trajectory, minimum energy consumption $E$
 1: Build a search graph by $\boldsymbol{\mathcal{T}}$
 2: Initialize OPEN, CLOSED, and COME_FROM
 3: $f(b_0) = 0$, OPEN.add($b_0$)
 4: while OPEN is not empty do
 5:     Find the node $q$ with the lowest $f(q)$ from OPEN
 6:     if $q = b_0^{'}$ then
 7:         Construct path from $b_0$ to $b_0^{'}$ by COME_FROM
 8:         return Trajectory, $E$
 9:     end if
10:     OPEN.remove($q$)
11:     CLOSED.add($q$)
12:     Obtain neighbor nodes of $q$
13:     for each neighbor node $m$ of $q$ do
14:         $cost = g(q)$ + the energy consumption of UAV-WSN from $q$ to $m$
15:         if $m$ in OPEN and $cost < g(m)$ then
16:             OPEN.remove($m$)
17:         end if
18:         if $m$ in CLOSED and $cost < g(m)$ then
19:             CLOSED.remove($m$)
20:         end if
21:         if $m$ not in OPEN and CLOSED then
22:             $g(m) = cost$
23:             $f(m) = g(m) + h(m)$
24:             OPEN.add($m$)
25:             COME_FROM[$m$]= $q$ //set $m$'s parent
26:         end if
27:     end for
28: end while
```

State

The state includes coordinates for all clusters, the UAV's location, and the energy consumption of UAV-WSN at current step t.

Action

The action represents the choice of the next cluster to be selected at current step t and the CH in this cluster. Thus, we define the output of the right-hand side of (5.26) and the CH selection by A* as the action at each step.

Reward

We design the reward as the negative of the total energy consumption in (4.21). This means that the DRL is set to get the maximal reward (minimal energy consumption).

Training

The actor-critic method includes the actor network and the critic network. The actor network is the proposed Ptr-A* in this work. The critic network is used to provide an approximated baseline of the reward for any problem instance to reduce the variance of gradients during the training phase and increases the speed of learning [40]. Our critic network, parameterized by ψ , has the same architecture as that of the encoder of the Ptr-A*. Then, its hidden states are decoded into a baseline prediction by two fully-connected ReLU layers [22]. Our training objective is the expected energy consumption, which is defined as

$$J\left(\theta|\boldsymbol{\mathcal{G}}\right) = \mathbb{E}_{\boldsymbol{\mathcal{T}}\sim p_{\theta(.|\boldsymbol{\mathcal{G}})}}[E]. \tag{4.27}$$

We use policy gradient method and stochastic gradient descent to optimize θ. The gradient of (4.27) is formulated by REINFORCE [41] algorithm

$$\nabla_\theta J\left(\theta|\boldsymbol{\mathcal{G}}\right) = \mathbb{E}_{\boldsymbol{\mathcal{T}}\sim p_{\theta(.|\boldsymbol{\mathcal{G}})}}\left[\left(E - V_\psi\left(\boldsymbol{\mathcal{G}}\right)\right)\nabla_\theta \log p_\theta\left(\boldsymbol{\mathcal{T}}|\boldsymbol{\mathcal{G}}\right)\right] \tag{4.28}$$

where $V_\psi\left(\boldsymbol{\mathcal{G}}\right)$ is a baseline function for reducing the variance of the gradients, which is implemented by the critic network. Assume we have B *i.i.d* train samples, the gradient in

Algorithm 7 Training Ptr-A* by Actor-Critic algorithm

```
Input: Training samples set $\boldsymbol{D} = \{\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \dots\}$, batch size $B$, training steps $S$
 1: Initialize actor network $\theta$ and critic network $\psi$ with random weights
 2: for $s = 1$ to $S$ do
 3:     Sample $\boldsymbol{\mathcal{G}}_i$ from $\boldsymbol{D}$, $\forall i \in \{1, \dots, B\}$
 4:     Calculate $E_i$ and $\boldsymbol{\mathcal{T}}_i$ with Ptr-A* network, $\forall i \in \{1, \dots, B\}$
 5:     Calculate $V_\psi(\boldsymbol{\mathcal{G}}_i)$ with Critic network, $\forall i \in \{1, \dots, B\}$
 6:     $d\theta \leftarrow \frac{1}{B}\sum_{i=1}^{B}(E_i - V_\psi(\boldsymbol{\mathcal{G}}_i))\nabla_\theta \log p_\theta(\boldsymbol{\mathcal{T}}_i|\boldsymbol{\mathcal{G}}_i)$
 7:     $L(\psi) \leftarrow \frac{1}{B}\sum_{i=1}^{B}(V_\psi(\boldsymbol{\mathcal{G}}_i) - E_i)^2$
 8:     $\theta \leftarrow \mathrm{Adam}(\theta, d\theta)$
 9:     $\psi \leftarrow \mathrm{Adam}(\psi, \nabla_\psi L(\psi))$
10: end for
11: return $\theta^* = \theta$
```

(5.35) can be approximated with Monte Carlo sampling as follows

$$\nabla_\theta J(\theta) \approx \frac{1}{B}\sum_{i=1}^{B}(E_i - V_\psi(\boldsymbol{\mathcal{G}}_i))\nabla_\theta \log p_\theta(\boldsymbol{\mathcal{T}}_i|\boldsymbol{\mathcal{G}}_i). \tag{4.29}$$

We train the parameters of the critic with stochastic gradient descent on a mean squared error objective $L(\psi)$ between its predictions $V_\psi(\boldsymbol{\mathcal{G}}_i)$ and the actual energy consumption. $L(\psi)$ is formulated as

$$L(\psi) = \frac{1}{B}\sum_{i=1}^{B}(V_\psi(\boldsymbol{\mathcal{G}}_i) - E_i)^2. \tag{4.30}$$

The training procedure is presented in Algorithm 9. Notice that Adam algorithm is used to update the parameters of the actor network and the critic network iteratively. Adam algorithm designs independent adaptive learning rates for different parameters via calculating the first and second moment estimates of the gradient instead of using a single learning rate to update all parameters by the traditional random gradient descent [44]. Given the initial learning rate, the learning rates in different steps adaptively change according to the learning results. Because of the generalization property of RNNs [45], our proposed models, including the Ptr-A* and critic, have a very good generalization ability. In the training phase, we can

Table 4.1 Simulation parameters

Parameter	**Description**	**Value**
ε_{fs}	Amplifier's energy parameter of the free space fading	10 pJ/bit/m^2 [31]
ε_{mp}	Amplifier's energy parameter of the multi-path fading	0.0013 pJ/bit/m^2 [31]
E_{elec}	Energy consumption per bit in the circuitry	50 nJ/bit [31]
P_{CH}	Transmit power of each CH	21 dBm/Hz [8]
N	Number of nodes per cluster	20
B_{width}	Bandwidth	1 MHz
N_0	Noise power	-174 dBm/Hz [8]
f_c	Carrier frequency	2 GHz [8]
α	Path loss exponent	3 [8]
H	UAV's flight height	50 m
μ_{LoS}, μ_{NLoS}	Mean values of the excessive path loss	1 dB, 20 dB [42]
β, η	Environmental parameters	0.03, 10 [8]
$v_{\text{UAV}} = v_{\text{max}}$	UAV's flight speed	15 m/s [8]
m_{tot}	UAV's mass	500 g [43]
r_p	Radius of UAV's propellers	20 cm [43]
n_p	Number of propellers	4 [43]
P_{max}	UAV's hardware power level at full speed	5 W [43]
P_{idle}	UAV's hardware power level when it hovers	0 W [43]
P_{com}	UAV's communication power	0.0126 W [8]

use small problem instances to train the model, and then the trained model can be utilized to solve large problem instances.

4.4 Numerical Results

In this section, we first introduce detailed environment settings, and then describe the decoding search strategies at inference. Furthermore, we compare the performance of the proposed DRL algorithm with several baseline algorithms.

4.4.1 Environmental Settings and Model Training

We consider the ground network size of 2 km $\times$ 2 km, and the start position of the UAV is located at $(0\,\text{m}, 0\,\text{m})$. Simulation parameters are listed in Table 4.1. We use mini-batches of size 512 and LSTM cells with 128 hidden units in the encoder and the decoder. We implement the proposed model by using Pytorch 1.4 and Python 3.7 on a VM instance of Google Cloud Platform with 1 NVIDIA TESLA P100 GPU. The parameters of both the actor and critic networks are initialized by the Xavier initialization method and trained by the Adam optimizer with an initial learning rate of 0.0001 and decayed every 5,000 steps by 0.96.

It is assumed that nodes in a given cluster $\boldsymbol{G}_k$ are distributed according to the Gaussian distribution. Each cluster's nodes are sampled from a $torch.normal(\nu, \text{std})$ where ν is the mean and std is the constant standard deviation. Each Gaussian distribution's ν is randomly sampled from a *torch.rand()* function to determine the position of each cluster in the two-dimensional space. We train the model using instances of 20 clusters and 40 clusters, respectively. The 20-clusters model is trained for 100,000 steps, and the 40-clusters model is trained for 200,000 steps. We give a simple example on how to obtain the train data. At each training step of the 20-clusters model, we sample 20 means from *torch.rand()* for 20 Gaussian distributions, respectively. Then, we use these 20 *i.i.d.* distributions to generate a set (problem instance) of 20 clusters where the number of nodes per cluster is 20. The test data sets are also generated in the same way, only the number of clusters is different.

4.4.2 Decoding Search Strategies at Inference

Given a new problem instance $\boldsymbol{\mathcal{G}}$ at inference, the decoder network of our trained Ptr-A* architecture can easily output an access sequence for all clusters. The decoding process of

Algorithm 8 Active Search

```
Input: Test input $\boldsymbol{\mathcal{G}}$, steps $S$, $\zeta$
 1: Randomly sample a solution $\boldsymbol{\mathcal{T}}$ for $\boldsymbol{\mathcal{G}}$
 2: Calculate $E$ according to $\boldsymbol{\mathcal{T}}$ by A*
 3: $O \leftarrow E$
 4: for $s = 1$ to $S$ do
 5:     $\boldsymbol{\mathcal{T}}_i \sim$ Sample solutions $P_\theta(\cdot|\boldsymbol{\mathcal{G}})$, $\forall i \in \{1, \ldots, Q\}$
 6:     $E_{(\boldsymbol{\mathcal{T}}_j|\boldsymbol{\mathcal{G}})} \leftarrow \text{argmin}\big(E_{(\boldsymbol{\mathcal{T}}_1|\boldsymbol{\mathcal{G}})}, \ldots, E_{(\boldsymbol{\mathcal{T}}_Q|\boldsymbol{\mathcal{G}})}\big)$
 7:     if $E_{(\boldsymbol{\mathcal{T}}_j|\boldsymbol{\mathcal{G}})} < E$ then
 8:         $\boldsymbol{\mathcal{T}} \leftarrow \boldsymbol{\mathcal{T}}_j$
 9:         $E \leftarrow E_{(\boldsymbol{\mathcal{T}}_j|\boldsymbol{\mathcal{G}})}$
10:     end if
11:     $d\theta \leftarrow \frac{1}{Q}\sum_{i=1}^{Q}\big(E_{(\boldsymbol{\mathcal{T}}_i|\boldsymbol{\mathcal{G}})} - O\big)\nabla_\theta \log p_\theta\left(\boldsymbol{\mathcal{T}}_i|\boldsymbol{\mathcal{G}}\right)$
12:     $\theta \leftarrow \text{Adam}\left(\theta, d\theta\right)$
13:     $O \leftarrow \zeta O + (1-\zeta)\frac{1}{Q}\sum_{i=1}^{Q} V_\psi\left(\boldsymbol{\mathcal{G}}\right)$
14: end for
15: return $\boldsymbol{\mathcal{T}}, E$
```

the decoder at inference shows how solvers search over a large set of feasible access sequences. In this work, we consider the following three decoding search strategies.

Greedy Search

Greedy search strategy always select the cluster with the largest probability at each decoding step during inference, which is labeled as DRL-greedy in the simulation results.

Sampling Search

This strategy samples M candidate solutions from the *stochastic policy* $P_\theta(\cdot|\boldsymbol{\mathcal{G}})$ by running the trained Ptr-A* on a single test input $\boldsymbol{\mathcal{G}}$ and selects the one with the minimum expected energy consumption from M candidate outputs. The more we sample, the more likely we will get the better output. In the simulation, we set $M = 51200$, and this strategy is labeled as DRL-sampling.

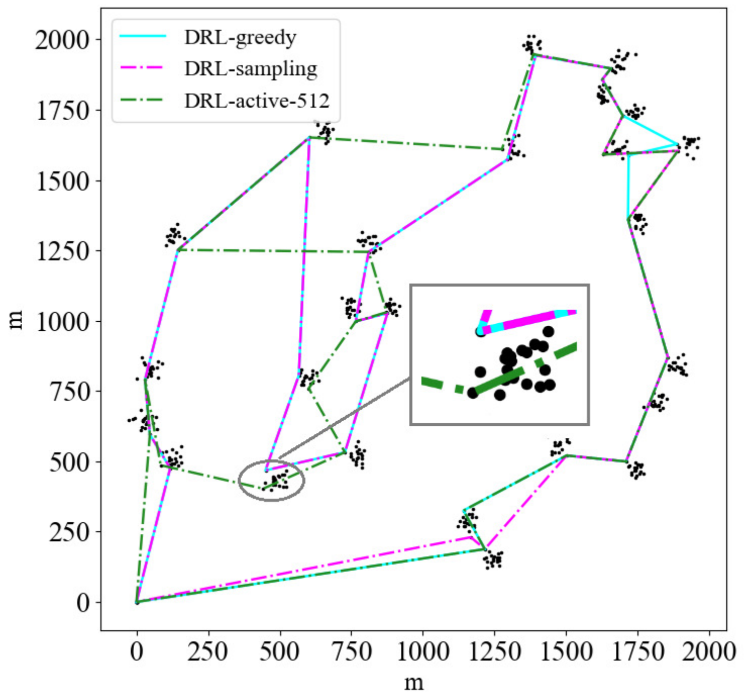
Active Search

Unlike the greedy search and the sampling search, this strategy can refine the parameter θ of the Ptr-A* during inference to minimize the expected energy consumption on a single test input $\boldsymbol{\mathcal{G}}$. Active search samples multiple solutions $\boldsymbol{\mathcal{T}}_1, \ldots, \boldsymbol{\mathcal{T}}_Q$ from $P_\theta(\cdot|\boldsymbol{\mathcal{G}})$ for a single test input $\boldsymbol{\mathcal{G}}$ and uses policy gradients to refine θ [22]. The process is presented in Algorithm 8. In the simulation, we sample three different sets of candidate solutions, $\{512, 5120, 10240\}$, which are labeled with DRL-active-512, DRL-active-5120, and DRL-active-10240, respectively.

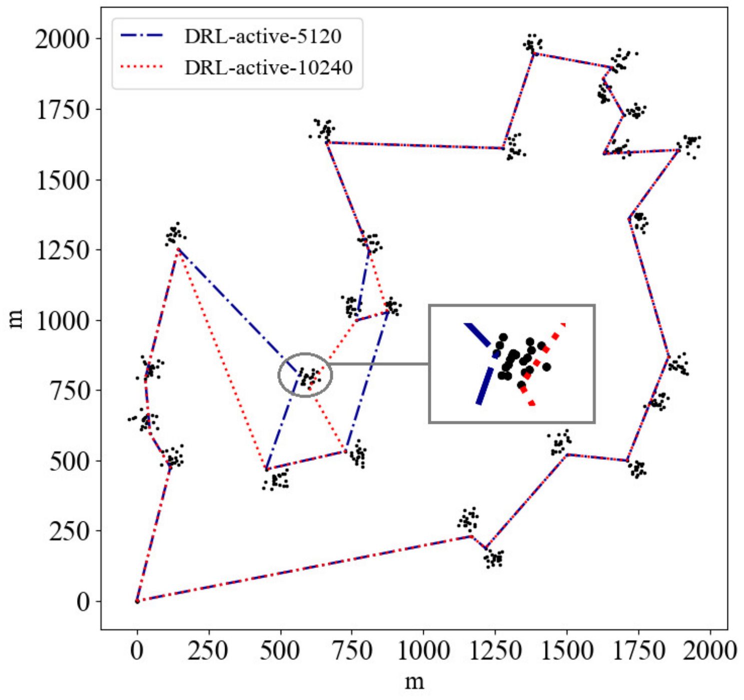
4.4.3 Small-Scale Clusters

To thoroughly evaluate the performance of the proposed DRL algorithm, we first test the trained 20-clusters model on small-scale clusters. Since ω in (4.21) is the weighting coefficient, its value does not impact the comparison results among algorithms. When $\omega = 0$, our optimization problem only considers the energy consumption of the UAV, which mainly depends on the flying distance of the UAV. Fig. 4.3 illustrates how the proposed DRL algorithm performs with different search strategies on the 25-clusters problem instance. As can be seen, the trajectory generated by DRL-greedy is the longest (9241 m), while DRL-active-10240 produces the shortest trajectory (8693 m) among strategies.

Next, we compare our proposed DRL algorithm having different decoding search strategies with the nearest neighbor (NN) heuristic [46] and the genetic algorithm [47]. We first investigate the energy consumption comparison between our proposed DRL on the trained 20-clusters model and two baselines when $\omega = 0.5$. The genetic algorithm runs for 4,000 generations, the chance of mutation is 0.5%, and the size of population is 150. In Fig. 4.4, we plot the average ratios of the energy consumption of our proposed DRL algorithm with different search strategies and two baselines to the energy consumption of DRL-active-10240 versus different numbers of clusters K. Although the model is trained on 20-clusters problem instances, it still obtains good performance on the 10-clusters, 30-clusters, 40-clusters,



(a)



(b)

Figure 4.3 Trajectories comparison on 25 clusters test instance when $\omega = 0$.

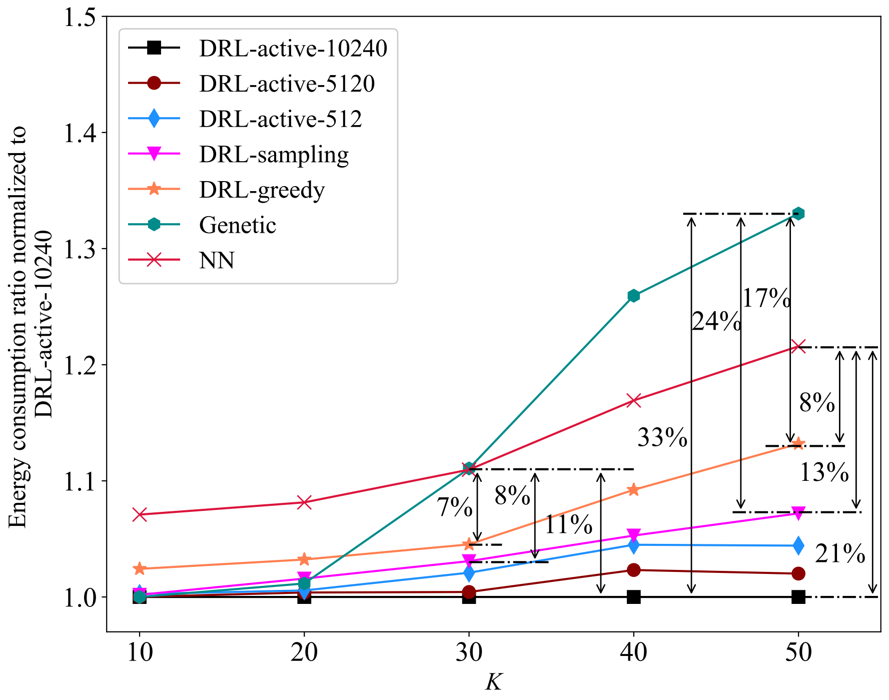
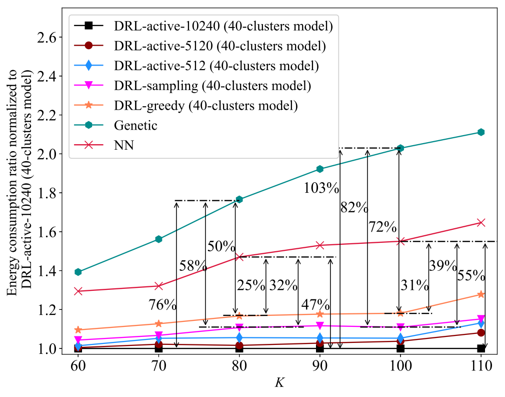
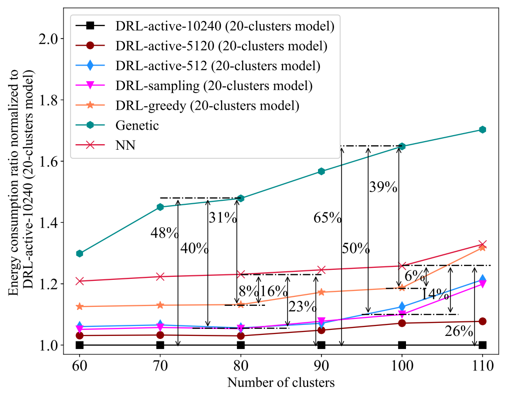


Figure 4.4 Energy consumption comparison on small-scale clusters.

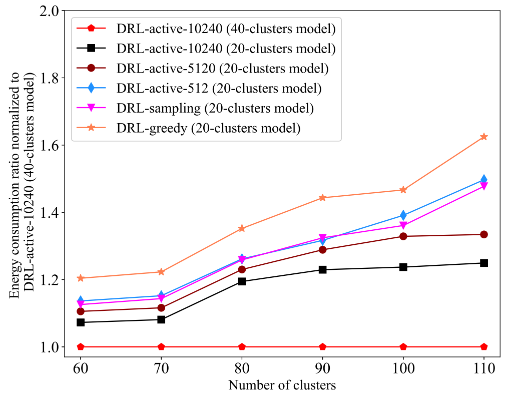
and 50-clusters networks. This shows that the proposed DRL algorithm achieves an excellent generalization ability with respect to the number of clusters used for training. When $K = 10$, genetic, DRL-active-10240, DRL-active-5120, DRL-active-512, and DRL-sampling obtain almost the same energy consumption result; however, the NN algorithm has higher energy consumption when compared with our proposed DRL with active search and sampling search strategies. As the number of clusters increases, the energy consumption savings of our proposed algorithm increase when compared to the NN and genetic algorithms. For example, when $K = 30$, the energy consumptions of UAV-WSN produced by NN and genetic algorithms are almost equal, which is about 11% more than that of DRL-active-10240, 8% more than that of DRL-sampling, and 7% more than that of DRL-greedy. When the number of clusters increases to 50, the energy consumption of NN is around 21% more than that of DRL-active-10240, 13% more than that of DRL-sampling, and 8% more than that of DRL-greedy. Likewise for $K = 50$, the energy consumption of the genetic algorithm is around 33% more than that of DRL-active-10240, 24% more than that of DRL-sampling, and



(a) On 40-clusters model.



(b) On 20-clusters model.



(c) 40-clusters model vs. 20-clusters model

Figure 4.5 Energy consumption comparison on large-scale clusters

17% more than that of DRL-greedy. It can be seen that our proposed DRL algorithm using any of three active search strategies can achieve better results than other search strategies and algorithms. This is because the active search strategy can refine the parameters of the Ptr-A* model for producing the best solution while searching for candidate solutions on a single test instance at inference. From Fig. 4.4, we can see that DRL-sampling also obtains a relatively competitive result.

Table 4.2 compares the running time at inference. As the number of clusters increases, the running time of the proposed DRL algorithm with all strategies and two baseline techniques increases. Although DRL-active-10240 obtains the best performance in reducing the energy consumption as can be seen from Fig. 4.4, it has the longest running time. This is because

Table 4.2 Running time comparison on small-scale clusters.

	Time (s)				
Algorithm \ K	10	20	30	40	50
DRL-active-10240	17.1	57.15	121.62	209.27	319.64
DRL-active-5120	8.59	29.12	60.85	105.82	160.68
DRL-active-512	0.98	3.4	7.05	12.11	18.48
DRL-sampling	14.26	29.34	43.39	59.89	81.3
DRL-greedy	0.31	0.63	1.07	1.68	2.46
Genetic	60.37	64.85	73.35	80.21	90.91
NN	1.12	1.12	1.13	1.15	1.16

it needs more iterations to update the parameters. If the number of candidate solutions is relatively small, active search strategy tends to spend less time to produce the solution, like DRL-active-512. In addition, we can see that the running time of NN is always minimal among all algorithms for all values of K. DRL-greedy's running time is comparable with that of NN. Meanwhile, the running times of DRL-sampling and DRL-active-512 are lower when compared with the genetic algorithm.

4.4.4 Large-Scale Clusters

In this subsection, we test the performance of the trained models on large-scale clusters test instances. The genetic algorithm runs for 10,000 generations. We first observe the results of the proposed DRL algorithm on the trained 40-clusters model, as shown in Fig. 4.5 (a). Clearly, our proposed DRL algorithm exhibits much better performances than the two baseline techniques in reducing the energy consumption of the UAV-WSN system. As the number of clusters increases, there is an increasing performance gap between the proposed DRL algorithm and the baseline techniques. For instance, when $K = 80$, the genetic algorithm consumes 50%, 58%, and 76% more energy than when compared to DRL-greedy, DRL-sampling, and DRL-active-10240, respectively. As the number of clusters increases to 100, the energy consumption of the UAV-WSN when using the genetic algorithm is 72% more than that of DRL-greedy, 82% more than that of DRL-sampling, and 103% more than that of DRL-active-10240. NN also shows a similar trend to that of the genetic algorithm. In particular, its energy consumption is 47% more than that of DRL-active-10240 when $K = 80$ and increases to 55% more than the energy consumption of DRL-active-10240 when $K = 100$. When compared with DRL-greedy, the extra amount of energy consumed by the UAV-WSN when using NN increases from 25% to 31% as the number of clusters increasing from 80 to 100. However, NN exhibits an obviously superior performance than the genetic algorithm. As we can see, the active search strategies outperform the greedy strategy and sampling strategy for large-scale clusters problems. DRL-sampling shows a slightly better performance than DRL-greedy, which is reasonable.

Next, we use the trained 20-clusters model to evaluate the performance of the proposed DRL algorithm on large-scale clusters test instances. In Fig. 4.5 (b), our proposed DRL algorithm with different search strategies still obtains relatively good results as compared to the two baseline techniques on large-scale problem instances when the value of K varies, while the results generated by active search strategies are still the best among all search strategies. For example, when $K = 80$, the energy consumption of the genetic algorithm is 48% more than that of DRL-active-10240, 40% more than that of DRL-sampling, and 31% more than that of DRL-greedy. Although NN exhibits a better performance than the genetic algorithm, its energy consumption is 23% more than that of DRL-active-10240, 16% more than that of DRL-sampling, and 8% more than that of DRL-greedy when $K = 80$. However, compared with the results obtained on the trained 40-clusters model, the 20-clusters model clearly shows inferior performance. When $K = 100$, the energy consumption gap between the genetic algorithm and DRL-active-10240 is 103% on the 40-clusters model, but this gap decreases to 65% on the 20-clusters model. Likewise, the gap in the energy consumption between the genetic algorithm and DRL-active-10240 decreases from 76% on the 40-clusters model to 48% on the 20-clusters model when $K = 80$.

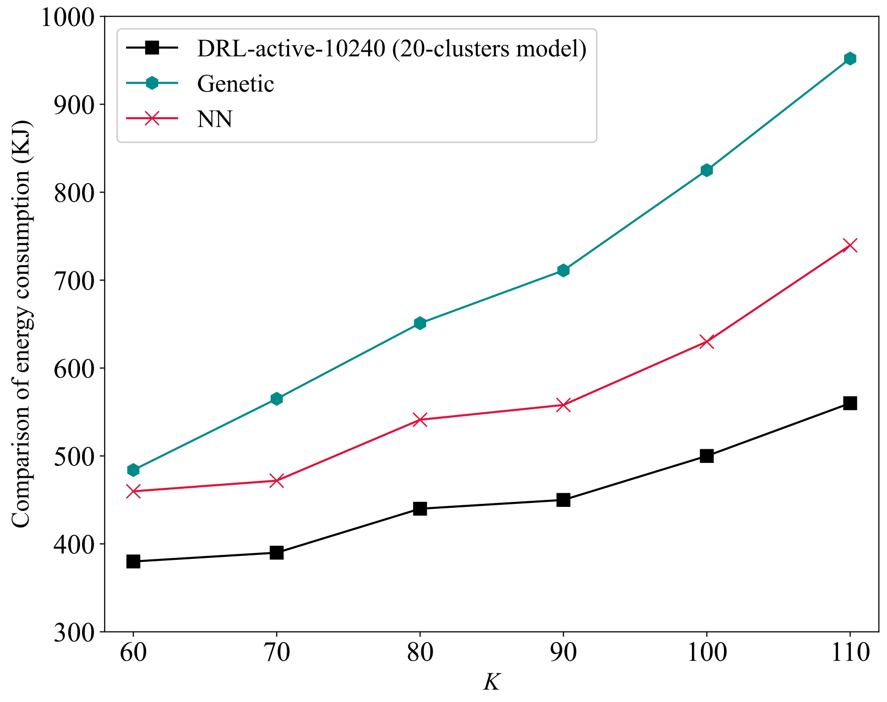


Figure 4.6 Comparison of energy consumption.

To investigate the difference between the two trained models, we compare the results obtained on the 20-clusters model with the results of DRL-active-10240 on the 40-clusters model. As shown in Fig. 4.5 (c), DRL-active-10240 (40-clusters model) clearly exhibits superior performance in reducing the energy consumption than three search strategies on the 20-clusters model. This performance merit constantly increases as the value of K increases. Thus, the trained 40-clusters model is more suitable for solving the trajectory planning problem for the UAV in cases of large-scale clusters.

Fig 4.6 shows the comparison of the total weighted energy consumption of the genetic algorithm, NN algorithm, and the DRL-active-10240. As one can see, the total weighted energy consumption E generated by our proposed algorithm is significantly lower then its counterpart of the NN and genetic algorithms. [1]

[1]This figure and the associated discussion are added based on the advisory committee request. They are not included in the published paper.

Table 4.3 Running time comparison on large-scale clusters.

Algorithm \ K		Time (s) 60	70	80	90	100	110
DRL-active-10240	20-clusters model	448.65	584.71	754.23	920.83	1149.36	1329.96
	40-clusters model	449.01	584.81	755.37	920.61	1150.05	1330.28
DRL-active-5120	20-clusters model	225.98	293.52	382.03	480.87	590.5	705.11
	40-clusters model	226.26	294.31	382.15	480.92	590.04	705.36
DRL-active-512	20-clusters model	24.99	33.08	44.08	55.54	67.21	79.31
	40-clusters model	25.75	33.57	44.1	55.78	67.54	79.88
DRL-sampling	20-clusters model	102.06	148.72	171.65	210.4	257.64	306.85
	40-clusters model	102.47	148.79	171.9	210.29	258.53	307.11
DRL-greedy	20-clusters model	3.11	4.23	5.84	7.83	9.96	12.55
	40-clusters model	3.39	4.54	5.81	7.48	10.2	12.89
Genetic		213.39	217.06	225.5	233.88	239.54	247.19
NN		1.17	1.17	1.18	1.19	1.2	1.22

The running time comparison of different strategies and algorithms at inference on large-scale problem instances is provided in Table 4.3. We can observe that for a given number of clusters, the same strategy running on different models produces solution in almost the same amount of time. For example, DRL-active-10240 takes 448.65 s on the 20-clusters model and 449.01 s on the 40-clusters model. DRL-active-10240 takes the longest time among all the algorithms because it requires more iterations to refine the parameters of the Ptr-A* to produce the best performance. The running time of DRL-sampling is acceptable in comparison with the genetic algorithm given its performance merits. Furthermore, the computation time spent by DRL-greedy is the least among all strategies, which is also significantly less than the running time of the genetic algorithm and slightly more than the time spent by NN. Although the NN algorithm spends the least amount of time, it produces worse results.

4.5 Conclusions

In this paper, we investigated the problem of designing the UAV's trajectory for a clustered WSN to minimize the total energy consumption in the UAV-WSN system. Inspired by the recent developments of DRL, we propose a novel DRL-based method to solve the UAV's trajectory planning problem. Because the visiting order of the UAV to clusters can be regarded as a sequential decision problem, we design a Ptr-A* model to produce the trajectory of the UAV. The pointer network of the proposed Ptr-A* model is used to determine the visiting order to clusters. Then, a search graph for all clusters is built according to the visiting order. The A* search algorithm is utilized to quickly find the CH for each cluster from the search graph with the aim of minimizing the energy consumption of the UAV-WSN system. In order to obtain optimal parameters of the Ptr-A*, we employ the model-free RL method to train the proposed Ptr-A* model in an unsupervised manner. Lastly, we propose three search strategies at inference.

We conduct comprehensive experiments to evaluate the performance of the proposed DRL algorithm. The simulation results show that the proposed DRL algorithm with different search strategies can produce better trajectories for the UAV when compared with the baseline techniques. In particular, DRL-active-10240 always produces the best results with different numbers of clusters of test instances. We also analyze the impact of different trained models on the results. The trained 40-clusters model is shown to be able to solve the trajectory planning problem of the UAV on large-scale clusters problems. The proposed DRL algorithm offers an appealing balance between performance and complexity. A key advantage of our proposed DRL algorithm is its generalization ability with respect to the number of clusters used for training. The model can be trained on small-scale clusters for faster training, and then can be used to solve larger-scale clusters problems. This makes it clearly more suitable for solving large-scale clusters problems as compared to the baseline techniques.

As for future work, we are interested in exploring other search strategies at inference to further improve the performance. It would also be interesting to develop a distributed DRL algorithm based on the Ptr-A* model to solve multiple UAVs' trajectory planning problem jointly. We also plan to investigate and improve the generalization capability of the proposed DRL algorithm, i.e., when the number of nodes per cluster at inference is significantly larger than that used for training.

Acknowledgement

This work was supported by an NSERC/Cisco Industrial Research Chair in Low-Power Wireless Access for Sensor Networks.

4.6 Appendix

A. Example of Attention Mechanism

Here, we give a detailed numerical example of 3-clusters network to explain how the attention mechanism works. In Fig. 4.2, the input sequence $\boldsymbol{\mathcal{G}} = \{b_0, G_1, G_2, G_3\}$ is transformed into a sequence of latent memory states $\boldsymbol{\mathcal{E}} = \{e_0, e_1, e_2, e_3\}$, which is the input of the decoder network. At decoding step 0, we calculate correlations between all elements in $\boldsymbol{\mathcal{E}}$ and the start tag v_{go} by (4.24) and (4.25), which can be expressed as

$$u_0^0 = \varphi \tanh (W_1 e_0 + W_2 h_0) \tag{A.1}$$

$$u_1^0 = \varphi \tanh (W_1 e_1 + W_2 h_0) \tag{A.2}$$

$$u_2^0 = \varphi \tanh (W_1 e_2 + W_2 h_0) \tag{A.3}$$

$$u_3^0 = \varphi \tanh (W_1 e_3 + W_2 h_0). \tag{A.4}$$

The learning parameters, namely φ, W_1, W_2, are initialized by the Xavier initialization method and trained by the Adam optimizer as explained in Section IV.A. Then, the *softmax* function is used to normalize the vector $u^0 = \{u_0^0, u_1^0, u_2^0, u_3^0\}$. For the sake of illustration, we assume that the four elements in u^0 determine the following four conditional probability values:

$$P(\pi_0 = 0|\boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_0^0)}{\exp(u_0^0) + \exp(u_1^0) + \exp(u_2^0) + \exp(u_3^0)} = 0.6 \tag{A.5}$$

$$P(\pi_0 = 1|\boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_1^0)}{\exp(u_0^0) + \exp(u_1^0) + \exp(u_2^0) + \exp(u_3^0)} = 0.1 \tag{A.6}$$

$$P(\pi_0 = 2|\boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_2^0)}{\exp(u_0^0) + \exp(u_1^0) + \exp(u_2^0) + \exp(u_3^0)} = 0.1 \tag{A.7}$$

$$P(\pi_0 = 3|\boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_3^0)}{\exp(u_0^0) + \exp(u_1^0) + \exp(u_2^0) + \exp(u_3^0)} = 0.2. \tag{A.8}$$

Since e_0 has the highest conditional probability value at decoding step 0, the output π_0 of this step points to the first element of $\boldsymbol{\mathcal{G}}$, b_0. At decoding step 1, we use the same approach to calculate the correlations between b_0 and the remaining elements in $\boldsymbol{\mathcal{E}}$ as follows

$$u_1^1 = \varphi \tanh(W_1 e_1 + W_2 h_1) \tag{A.9}$$

$$u_2^1 = \varphi \tanh(W_1 e_2 + W_2 h_1) \tag{A.10}$$

$$u_3^1 = \varphi \tanh(W_1 e_3 + W_2 h_1). \tag{A.11}$$

Similarly, by using the *softmax* function, we calculate the following conditional probabilities based on the obtained elements $\{u_1^1, u_2^1, u_3^1\}$:

$$P(\pi_1 = 1|\pi_0, \boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_1^1)}{\exp(u_2^1) + \exp(u_2^1) + \exp(u_3^1)} = 0.2 \tag{A.12}$$

$$P(\pi_1 = 2|\pi_0, \boldsymbol{\mathcal{G}}) \\ = \frac{\exp(u_2^1)}{\exp(u_2^1) + \exp(u_2^1) + \exp(u_3^1)} = 0.1 \tag{A.13}$$

$$P(\pi_1 = 3|\pi_0, \boldsymbol{\mathcal{G}})$$
$$= \frac{\exp{(u_3^1)}}{\exp{(u_2^1)} + \exp{(u_2^1)} + \exp{(u_3^1)}} = 0.7. \tag{A.14}$$

The output π_1 of this step points to G_3 because the conditional probability of e_3 is maximum. Then, the process repeats until we obtain the full output sequence of the decoder network as $\{b_0, G_3, G_1, G_2\}$, i.e., $\{\pi_0, \pi_1, \pi_2, \pi_3\}$.

References

[1] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, “A tutorial on UAVs for wireless networks: Applications, challenges, and open problems,” *IEEE Communications Surveys Tutorials*, vol. 21, pp. 2334–2360, Mar. 2019.

[2] Y. Zeng, R. Zhang, and T. J. Lim, “Wireless communications with unmanned aerial vehicles: opportunities and challenges,” *IEEE Communications Magazine*, vol. 54, pp. 36–42, May 2016.

[3] D. A. Hedges, J. P. Coon, and G. Chen, “A continuum model for route optimization in large-scale inhomogeneous multi-hop wireless networks,” *IEEE Transactions on Communications*, vol. 68, pp. 1058–1070, Feb. 2020.

[4] D. H. Tran, T. X. Vu, S. Chatzinotas, S. ShahbazPanahi, and B. Ottersten, “Coarse trajectory design for energy minimization in UAV-enabled,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 9483–9496, Sep. 2020.

[5] Y. Zeng, J. Xu, and R. Zhang, “Energy minimization for wireless communication with rotary-wing UAV,” *IEEE Transactions on Wireless Communications*, vol. 18, pp. 2329–2345, Apr. 2019.

[6] C. Zhan and H. Lai, “Energy minimization in internet-of-things system based on rotary-wing UAV,” *IEEE Wireless Communications Letters*, vol. 8, pp. 1341–1344, Oct. 2019.

[7] J. Baek, S. I. Han, and Y. Han, “Energy-efficient UAV routing for wireless sensor networks,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 1741–1750, Feb. 2019.

[8] M. B. Ghorbel, D. Rodríguez-Duarte, H. Ghazzai, M. J. Hossain, and H. Menouar, “Joint position and travel path optimization for energy efficient wireless data gathering using unmanned aerial vehicles,” *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2165–2175, Mar. 2019.

[9] D. Yang, Q. Wu, Y. Zeng, and R. Zhang, "Energy tradeoff in ground-to-UAV communication via trajectory design," *IEEE Transactions on Vehicular Technology*, vol. 67, pp. 6721–6726, Jul. 2018.

[10] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "Improved soft-k-means clustering algorithm for balancing energy consumption in wireless sensor networks," *IEEE Internet of Things Journal*, vol. 8, pp. 4868–4881, Mar. 2021.

[11] M. Samir, S. Sharafeddine, C. M. Assi, T. M. Nguyen, and A. Ghrayeb, "UAV trajectory planning for data collection from time-constrained IoT devices," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 34–46, Jan. 2019.

[12] S. Zhang, S. Shi, S. Gu, and X. Gu, "Power control and trajectory planning based interference management for UAV-assisted wireless sensor networks," *IEEE Access*, vol. 8, pp. 3453–3464, Dec. 2019.

[13] A. A. Al-Habob, O. A. Dobre, S. Muhaidat, and H. V. Poor, "Energy-efficient data dissemination using a UAV: An ant colony approach," *IEEE Wireless Communications Letters*, vol. 10, pp. 16–20, Jan. 2021.

[14] K. Zhu, X. Xu, and S. Han, "Energy-efficient UAV trajectory planning for data collection and computation in mmtc networks," in *Proc. IEEE Globecom Workshops*, pp. 1–6, Dec. 2018.

[15] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 2059–2070, Sep. 2018.

[16] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Actor-critic deep reinforcement learning for energy minimization in UAV-aided networks," in *Proc. European Conference on Networks and Communications (EuCNC)*, pp. 348–352, Jun. 2020.

[17] R. Ding, F. Gao, and X. S. Shen, "3D UAV trajectory design and frequency band allocation for energy-efficient and fair communication: A deep reinforcement learning

approach," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 7796–7809, Dec. 2020.

[18] B. Zhang, C. H. Liu, J. Tang, Z. Xu, J. Ma, and W. Wang, "Learning-based energy-efficient data collection by unmanned vehicles in smart cities," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 1666–1676, Apr. 2018.

[19] W. Liu, P. Si, E. Sun, M. Li, C. Fang, and Y. Zhang, "Green mobility management in UAV-assisted iot based on dueling DQN," in *Proc. IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2019.

[20] H. Zhang, J. Li, Y. Ji, and H. Yue, "Understanding subtitles by character-level sequence-to-sequence learning," *IEEE Transactions on Industrial Informatics*, vol. 13, pp. 616–624, Apr. 2017.

[21] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 28, pp. 2692–2700, Dec. 2015.

[22] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," in *Proc. International Conference on Learning Representations (ICLR) Workshop*, pp. 1–5, Apr. 2017.

[23] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, pp. 3806–3817, Oct. 2019.

[24] Z. Li, J. Wu, L. Sun, and T. Rong, "Combinatorial keyword recommendations for sponsored search with deep reinforcement learning," *arXiv preprint arXiv:1907.08686*, 2019.

[25] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Advances Neural Information Processing Systems (NIPS)*, pp. 9861–9871, Dec. 2018.

[26] V. del Razo and H. Jacobsen, “Smart charging schedules for highway travel with electric vehicles,” *IEEE Transactions on Transportation Electrification*, vol. 2, pp. 160–173, Jun. 2016.

[27] Z. Ma, B. Ai, R. He, G. Wang, Y. Niu, and Z. Zhong, “A wideband non-stationary air-to-air channel model for UAV communications,” *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 1214–1226, Feb. 2020.

[28] Z. Lian, L. Jiang, C. He, and D. He, “A non-stationary 3D wideband GBSM for HAP-MIMO communication systems,” *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 1128–1139, Feb. 2019.

[29] D. Hulens, T. Goedemé, and J. Verbeke, “How to choose the best embedded processing platform for on-board UAV image processing,” in *Proc. International Conference on Computer Vision Theory and Applications (VISAPP)*, pp. 377–386, 2015.

[30] R. Roberti and P. Toth, “Models and algorithms for the asymmetric traveling salesman problem: an experimental comparison,” *EURO Journal on Transportation and Logistics*, vol. 1, pp. 113–133, Jun. 2012.

[31] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Transactions on Wireless Communications*, vol. 1, pp. 660–670, Mar. 2002.

[32] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*, vol. 2. prentice hall PTR New Jersey, 1996.

[33] J. Puchinger and G. R. Raidl, “Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification,” in *Proc. International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC)*, pp. 41–53, Jun. 2005.

[34] D. Gong and Y. Yang, “Low-latency SINR-based data gathering in wireless sensor networks,” *IEEE Transactions on Wireless Communications*, vol. 13, pp. 3207–3221, Jun. 2014.

[35] L. Liu and U. Mitra, "On sampled reinforcement learning in wireless networks: Exploitation of policy structures," *IEEE Transactions on Communications*, vol. 68, pp. 2823–2837, May 2020.

[36] I. Bello, S. Kulkarni, S. Jain, C. Boutilier, E. Chi, E. Eban, X. Luo, A. Mackey, and O. Meshi, "Seq2slate: Re-ranking and slate optimization with RNNs," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–12, May 2019.

[37] Y. Zhu, X. Dong, and T. Lu, "An adaptive and parameter-free recurrent neural structure for wireless channel prediction," *IEEE Transactions on Communications*, vol. 67, pp. 8086–8096, Nov. 2019.

[38] W. Koehrsen, "Neural network embeddings explained," 2018. [Online]. Available: https://towardsdatascience. com/neural-network-embeddings-explained-4d028e6f0526.

[39] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, Jun. 2015.

[40] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Advances Neural Information Processing Systems (NIPS)*, pp. 1008–1014, Dec. 1999.

[41] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, May 1992.

[42] A. Al Hourani, S. Kandeepan, and S. Lardner, "Optimal LAP altitude for maximum coverage," *IEEE Wireless Communications Letters*, vol. 3, pp. 569–572, Dec. 2014.

[43] H. Ghazzai, M. Ben Ghorbel, A. Kadri, M. J. Hossain, and H. Menouar, "Energy-efficient management of unmanned aerial vehicles for underlay cognitive radio systems," *IEEE Transactions on Green Communications and Networking*, vol. 1, pp. 434–443, Dec. 2017.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–13, May 2015.

[45] Z. Tu, F. He, and D. Tao, "Understanding generalization in recurrent neural networks," in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–16, Apr. 2020.

[46] B. Hu and G. R. Raidl, "Effective neighborhood structures for the generalized traveling salesman problem," in *Proc. European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP)*, pp. 36–47, Mar. 2008.

[47] J. Li, Q. Sun, M. Zhou, and X. Dai, "A new multiple traveling salesman problem and its genetic algorithm-based solution," in *Proc. IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 627–632, Oct. 2013.

5. UAV Trajectory Planning for AoI-Minimal Data Collection in UAV-Aided IoT Networks by Transformer

Submitted as:

B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton and Z. Gao, “UAV Trajectory Planning for AoI-Minimal Data Collection in UAV-Aided IoT Networks by Transformer”, under major revision, *IEEE Transactions on Wireless Communications.*

In previous chapters, we studied energy-related problems concerning data collection in wireless networks. Apart from energy, the freshness of the collected data is another important metric in wireless networks, which can be measured by a new concept, AoI.

In this chapter, we are interested in investigating the AoI-minimal data collection problem in UAV-aided IoT networks. We propose an AoI-oriented data collection model in a cluster-based IoT network and formulate a total AoI-minimal UAV trajectory planning problem. The formulated problem not only considers the flying time of the UAV but also the data collection time spent at each hovering point, which is expressed as a TSPN where the hovering points of the UAV and the visiting order to these points are jointly optimized. To reduce the computational complexity for solving the formulated optimization problem, we convert the continuous optimization TSPN into a GTSP by the sampling-based technique.

The formulated GTSP is viewed as a “machine translation” problem in which the “source language” is the whole UAV-IoT network and the “target language” is the UAV trajectory with the minimal total AoI. We use the state-of-the-art *transformer* and the weighted-A* algorithm to design a novel machine learning algorithm to solve the formulated problem.

The parameters of the proposed algorithm are trained by reinforcement learning that only needs the reward calculation.

UAV Trajectory Planning for AoI-Minimal Data Collection in UAV-Aided IoT Networks by Transformer

Botao Zhu, Ebrahim Bedeer, Ha Nguyen,
Robert Barton, and Zhen Gao

Abstract

Maintaining freshness of data collection in Internet-of-Things (IoT) networks has attracted increasing attention. By taking into account age-of-information (AoI), we investigate the trajectory planning problem of an unmanned aerial vehicle (UAV) that is used to aid a cluster-based IoT network. An optimization problem is formulated to minimize the total AoI of the collected data by the UAV from the ground IoT network. Since the total AoI of the IoT network depends on the flight time of the UAV and the data collection time at hovering points, we jointly optimize the selection of hovering points and the visiting order to these points. We exploit the state-of-the-art transformer and the weighted A* to design a machine learning algorithm to solve the formulated problem. The whole UAV-IoT system is fed into the encoder network of the proposed algorithm, and the algorithm's decoder network outputs the visiting order to ground clusters. Then, the weighted A* is used to find the hovering point for each cluster in the ground IoT network. Simulation results show that the trained model by the proposed algorithm has a good generalization ability to generate solutions for IoT networks with different numbers of ground clusters, without the need to retrain the model. Furthermore, results show that our proposed algorithm can find better UAV trajectories with the minimum total AoI when compared to other algorithms.

Index Terms

AoI, IoT, transformer, trajectory optimization, UAV

5.1 Introduction

The use of unmanned aerial vehicles (UAVs) has attracted a lot of attention from academia and industry. Because of UAVs' high maneuvering capability and mobility, they can be used as wireless relays or mobile base stations to provide reliable communications and better coverage for ground devices [1]. Thanks to these advantages, UAVs can be flexibly deployed to provide fast and reliable network access in different applications, such as disasters [2], surveillance [3], monitoring [4], to name a few.

Since UAVs can fly close to the ground devices and build low-altitude air-to-ground communication links with them, UAVs can be deployed to hover the area of interest to collect data from ground Internet-of-Things (IoT) networks. By doing so, UAV-aided data collection can save the energy of devices in traditional IoT networks, thus extending their lifetime [5]. However, maintaining the freshness of the collected information is an important issue in time-sensitive IoT applications, such as environmental monitoring and safety protection. In these applications, the generated data needs to be sent to the destination as soon as possible. Outdated information can lead to incorrect control and even cause major disasters [6]. Therefore, it is essential to ensure the freshness of the data received at the destination. To measure the freshness of information, the age of information (AoI) as a new performance metric was proposed in [7]. In a nutshell, AoI describes the amount of time elapsed since the generation of the most recent data update. AoI-based data collection can guarantee information freshness in IoT networks, which is quite different from traditional delay-based and throughput-based metrics [8]. As such, it has attracted increasing attention.

Due to the importance of AoI, a number of studies have been carried out on AoI-oriented data collection in UAV-assisted wireless networks. In [9], the authors aimed to minimize the average AoI of the system by optimizing the trajectory of the UAV in a UAV-aided data collection system. In [10], the authors optimized the trajectory of the UAV to minimize the maximal AoI and the average AoI of sensors. In [11], the authors assumed the UAV supports three modes to collect data and jointly optimize the trajectory and data collection modes of the UAV to minimize the average AoI of all ground nodes. In [12], the UAV trajectory, energy, and service time allocation were jointly optimized by an iterative algorithm in order to minimize the overall peak AoI of the system. The authors in [13] developed an energy-efficient navigation policy for the UAV to improve data freshness of the IoT network. In order to minimize the weighted sum of AoI, the authors in [14] jointly optimized the flight trajectory of the UAV and the transmission scheduling of sensors. According to the above analysis, AoI-oriented data collection problems in the UAV-assisted IoT network are usually related to UAV's trajectory design.

When collecting data in the UAV-assisted IoT network, if the UAV is dispatched to visit every ground IoT device, the energy consumption of the UAV will increase because of the increased UAV trajectory. Hence, to reduce the energy consumption of the UAV, clusters-based system model is extensively investigated in UAV-assisted wireless networks. In [15], to gather compressive data measurements, the authors divide the sensor network into multiple clusters. In each cluster, all nodes build a forwarding tree based on compressive data gathering to send data to the cluster head (CH). The UAV has then to traverse all CHs to collect the aggregated data. The authors jointly optimized the UAV trajectory, CH selection, and forward tree construction to minimize the total transmit power in the network. In [16], the authors consider a pre-clustered network where a UAV equipped with multiple-antenna communicates with multiple ground users simultaneously, in a given time slot, using space division multiple access. The authors jointly optimized the time slot allocation and the UAV hovering time to minimize the overall energy consumption. In [17], the authors considered a UAV-enabled massive machine-type communications (mMTC) data collection system where machine-type communication devices (MTCDs) are divided into several clusters. A UAV visits each hovering position which corresponds to a MTCD cluster and sequentially collects data from each MTCD in the corresponding cluster. They formulated a problem of minimizing the total energy consumption of the system. In our previous work [5], we used a UAV to collect data from a clustered IoT network, where the hovering points of the UAV are determined by the unknown CHs location. In other words, we jointly select the CHs and their visiting order to minimize the total energy consumption. Considering the above mentioned advantage, we examine the scenario that the UAV collects data from a group of clusters where the UAV only interact with the CHs of clusters. The problem of interest in this paper is to jointly optimize the UAV's hovering points and trajectory to achieve the minimal AoI data collection in a cluster-based IoT network. The optimization problem is formulated as a traveling salesman problem (TSP) with neighborhoods (TSPN), which is extremely challenging because it includes a continuous problem (optimization of hovering points) and a combinatorial problem (optimization of visiting order).

The hovering points of the UAV and the visiting order to these hovering points have a great impact on the flying time of the UAV and data collection time, which directly influence the total AoI of collected data. There have been some works on solving the TSPN efficiently. For example, a genetic algorithm was presented in [18] to solve the TSPN with the Dubins vehicle model for a set of polygonal regions. The authors in [19] proposed a memetic algorithm (an extension of the traditional genetic algorithm) to jointly optimize the selection of waypoints and the sequence of visits for the Dubins TSPN problem. These evolutionary techniques may provide acceptable solutions, but they are computationally demanding, which makes them not suitable to handle moderate or large scale problems. In order to reduce the computational complexity, some sampling-based algorithms were developed by sampling a finite set of points from a continuous state space. In [20], the Dubins TSPN was converted to a generalized TSP (GTSP) by using the sampling-based roadmap method, and then to an asymmetric TSP that can be addressed by the Lin-Kernighan heuristic algorithm. To handle the continuous optimization problem of waypoints within each circular neighborhood, the authors in [21] proposed a discretization scheme that equidistantly samples possible locations along the circular border of the interest neighborhood to determine the locations of the waypoints. In this paper, in order to reduce the computational complexity for solving the joint optimization of the UAV's hovering points and trajectory to achieve the minimal AoI data collection in a cluster-based IoT network, we transform the formulated continuous optimization TSPN into a GTSP by borrowing the sampling-based idea. The transformed

GTSP is a combinatorial optimization problem that can be solved using traditional methods, such as exact algorithms, approximate algorithms, or heuristic algorithms. However, these traditional algorithms may not achieve a good balance between optimality and computational complexity. Thus, by considering optimality, computational complexity, and generality, we plan to develop a machine learning-based algorithm to solve the transformed GTSP, i.e. the UAV's trajectory design problem.

Machine learning has been introduced as a new breakthrough technique in the UAV-assisted IoT network for solving UAV's trajectory planning problem. To minimize the weighted sum-AoI in the UAV-assisted network, the authors in [22] developed a deep reinforcement learning (DRL) to optimize the UAV's trajectory using a deep Q network (DQN) and an artificial neural network (ANN). In [23], the authors utilized Q-learning to optimize AoI-optimal UAV path by considering the deadline constraints of data in the UAV-aided sensing network. In [24], the authors jointly optimized the UAV's trajectory and the scheduling of the status update packets to minimize the normalized weighted sum of AoI in the UAV-assisted wireless network. Specifically, they used ANN, DQN, and long short-term memory (LSTM) to develop a DRL algorithm for learning the UAV trajectory in large-scale scenarios. Different from them, we employ the state-of-the-art transformer and the weighted A* search method to design a UAV trajectory planning algorithm for AoI-oriented data collection.

Transformer was originally proposed by Google as a sequence-to-sequence model to deal with machine translation problem [25]. It has achieved great success in many areas of artificial intelligence in the past four years, such as computer vision, audio processing, document summarization, and document generation. Some researchers also attempt to use transformer and its variants to tackle combinatorial problems, such as the TSP. In [26], the cities in the TSP were encoded by a transformer and decoded sequentially through a query consisting of the last three cities in the partial tour. The used transformer was trained by reinforcement learning. In [27], the authors also used the transformer architecture as the encoder network and the decoder network outputs the result sequentially based on the embeddings from the encoder and the outputs generated at previous steps. The encoder and decoder networks were trained using a reinforce algorithm with a deterministic greedy baseline. The authors in [28] proposed a transformer-based framework to automatically learn improved heuristics on two representative routing problems: the TSP and capacitated vehicle routing problem (CVRP). In [29], the authors used the standard transformer architecture to tackle TSP and achieve an improved performance over recent learned heuristics. Inspired by the success of employing transformer in solving various problems of route planning, we also exploit it in this paper for solving our formulated GTSP combinatorial optimization problem. The main contributions of this paper are summarized as follows:

1. We propose an AoI-oriented data collection model in a cluster-based IoT network and formulate a total AoI-minimal trajectory planning problem where the hovering points of the UAV and the visiting order to these points are jointly optimized.

2. We view the formulated problem as a "machine translation" problem where the "source language" is the whole UAV-IoT network and the "target language" is the UAV trajectory with the minimal total AoI. The state-of-the-art transformer-weighted-A* (TWA*) is employed to solve the formulated problem. The parameters of the proposed algorithm are trained by reinforcement learning that only needs the reward calculation.

3. The learned policy by the proposed algorithm generalizes well on different sizes of problem instances. In other words, the trained model by the proposed algorithm can automatically find a trajectory with the minimal total AoI for new problem instances, without retraining the model.

4. Extensive simulations are conducted to evaluate the performance of the proposed algorithm. Results show that the proposed algorithm achieves significant performance gain in maintaining data freshness while reducing computation time when compared with other baseline algorithms.

The rest of this paper is organized as follows. Section 5.2 introduces the system model and presents the formulated problem. Section 5.3 develops the proposed algorithm. Section 5.4 provides simulation results. Finally, Section 5.5 concludes the paper.

5.2 System Model and Problem Formulation

We consider a UAV-assisted IoT network that consists of one rotary-wing UAV, one ground base station (BS) located at b_0, and M clusters of ground sensor nodes. Specifically, each cluster m, $m = 1, \ldots, M$, has one CH, located as b_m, and N_m ordinary sensor nodes, located at as $B_m = \{b_m^{(1)}, \ldots, b_m^{(N_m)}\}$. The ground IoT network performs some sensing tasks in the surrounding area where the ordinary sensor nodes are responsible for sampling data and forwarding the collected data to their corresponding CHs. The UAV is dispatched from the start hovering point c_0 which is directly above b_0 to visit M mission hovering points $\{c_1, \ldots, c_m, \ldots, c_M\}$ by a pre-designed trajectory for data collection, and then flies back to c_0 after completing the data collection task. Each hovering point corresponds to one ground cluster and its position will be determined by the proposed algorithm. The three-dimensional (3D) Cartesian coordinates system is considered to define positions of hovering points and all CHs. The coordinate of the m-th hovering point is denoted by $c_m = (x_{c_m}, y_{c_m}, H) \in \mathbb{R}^3$, where H is the flight height of the UAV, whereas the location of the corresponding ground CH is given by $b_m = (x_{b_m}, y_{b_m}, 0) \in \mathbb{R}^3$.

We assume that the rotary-wing UAV supports a flying-hovering mode without considering acceleration-deceleration, i.e., it flies to the hovering points with a fixed speed v_{UAV} and hovers at these points with static status to collect data from ground CHs. We illustrate the UAV-assisted data collecting process in Fig. 5.1. The UAV takes off from c_0, determines the position of the hovering point c_2 that will be visited first and arrives at it. The UAV repeats this procedure until data collection of all clusters is completed, and flies back to c_0. Hence, the final trajectory of the UAV in this example is $\{c_0, c_2, c_3, c_4, c_1, c_0\}$.

5.2.1 Data Collection Model

When the UAV arrives at c_m, it sends a beacon message to wake up the corresponding CH b_m from its sleep mode. The beacon message includes the type of sensor nodes to be activated in response to the beacon, the data collection height of the UAV, a threshold to limit the number of sensor nodes in the CH (if necessary), and a trailer that has error detection capabilities. Then, b_m switches to its active mode and informs its member nodes

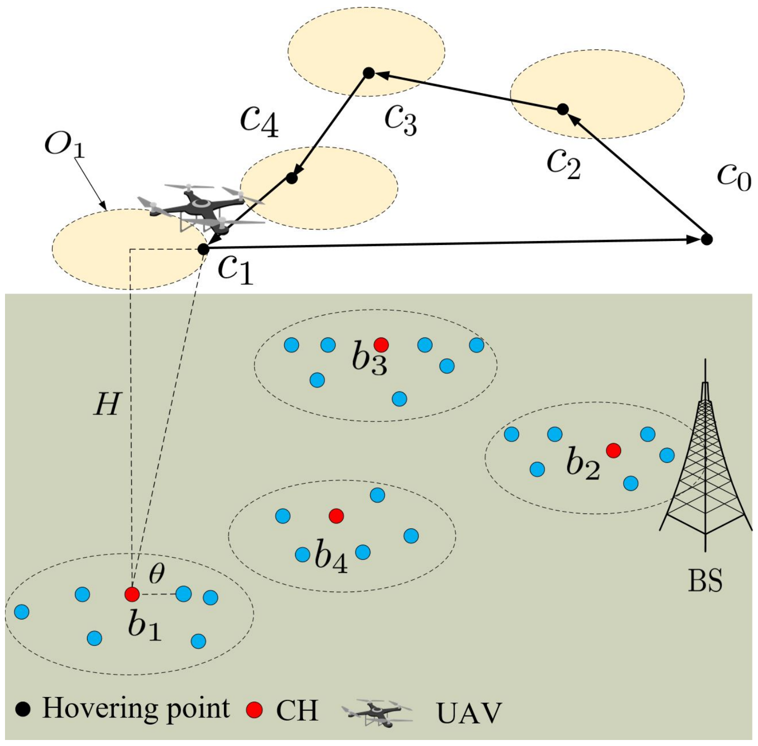


Figure 5.1 System model of a UAV-assisted IoT network.

in the same cluster to sample and send their sampled data sequentially according to the pre-allocated equal-length time slots using time-division multiplexing (TDM) protocol to avoid collision. We consider the *generate-at-will* model [30] as the data sampling model for all ordinary sensor nodes, by which nodes can generate information updates at any time. Specifically, we assume that each node can generate an update message of size L_{data} only in its allocated time slot to eliminate the waiting time. Also, each message has a time stamp, which is the start of each time slot. The length of a time slot is denoted as τ seconds. After the CH located at b_m finishes collecting data from its member nodes, it will forward the collected data to the UAV. For ease of analysis, the wake-up time of nodes, including CHs and all ordinary nodes, and the information sampling time of each node are assumed negligible as compared to the data collection time. Thus, the data collection time of the UAV at each hovering point mainly consists of two parts: the data transmission time from ordinary nodes to their CHs and the time consumed for forwarding the collected data from CHs to the UAV.

We consider both the line-of-sight (LoS) and non-line-of-sight (NLoS) links to design the ground-to-air communication when the UAV hovers at mission hovering points. The LoS link probability is related to environment, elevation angle, and transmission distance, which can be expressed as [31]

$$P_{c_m}^{(\mathrm{LoS})} = \frac{1}{1 + \beta \exp\left(-\widetilde{\beta}\left(\theta_{c_m} - \beta\right)\right)} \tag{5.1}$$

where β and $\widetilde{\beta}$ are constants determined by the environment, $\theta_{c_m} = \arctan\left(H/R_{(c_m,b_m)}\right)$ is the elevation angle between b_m and the UAV when it hovers at c_m, $R_{(c_m,b_m)} = \sqrt{\left(x_{c_m} - x_{b_m}\right)^2 + \left(y_{c_m} - y_{b_m}\right)^2}$ is the horizontal distance between the CH b_m and the hovering point. Correspondingly, the probability of NLoS is given by $P_{c_m}^{(\mathrm{NLoS})} = 1 - P_{c_m}^{(\mathrm{LoS})}$. In addition, the path loss models of LoS and NLoS between the CH b_m and the UAV follow [32]

$$\begin{aligned} L_{c_m}^{(\mathrm{LoS})} &= 20 \log_{10}\left(\frac{4\pi f_c d_{(c_m,b_m)}}{v_{\mathrm{light}}}\right) + \xi_{\mathrm{LoS}}, \\ L_{c_m}^{(\mathrm{NLoS})} &= 20 \log_{10}\left(\frac{4\pi f_c d_{(c_m,b_m)}}{v_{\mathrm{light}}}\right) + \xi_{\mathrm{NLoS}} \end{aligned} \tag{5.2}$$

where f_c is the carrier frequency, v_{light} is the speed of light, $d_{(c_m,b_m)} = \sqrt{H^2 + R_{(c_m,b_m)}^2}$ is the distance between the UAV and the CH b_m, ξ_{LoS} and ξ_{NLoS} ($\xi_{\mathrm{LoS}} < \xi_{\mathrm{NLoS}}$) are the excessive path losses in LoS and NLoS links, respectively. We consider the average path loss to describe the link from the ground CH to the UAV, which can be expressed as

$$\overline{L}_{c_m} = P_{c_m}^{(\mathrm{LoS})} L_{c_m}^{(\mathrm{LoS})} + P_{c_m}^{(\mathrm{NLoS})} L_{c_m}^{(\mathrm{NLoS})}. \tag{5.3}$$

To avoid the interference among CHs, we assume that only one CH can transmit data to the UAV at any given time. Hence, the average available transmission rate in bits per second (bps) from CH b_m to the UAV can be expressed as $r_{c_m} = B_{\mathrm{width}} \log_2\left(1 + \gamma_{c_m}\right)$, where B_{width} is the channel bandwidth in hertz (Hz), $\gamma_{c_m} = P_{\mathrm{CH}} / \left(\sigma^2 10^{\overline{L}_{c_m}/10}\right)$ is the signal-to-noise ratio (SNR) of the transmission link, σ^2 is the noise power at the UAV, and P_{CH} is the transmission power of the CH. Regarding the transmission quality, we set a SNR threshold γ_{th} and the transmission is considered successful if the SNR is greater than the threshold. Thus, the SNR constraint at the UAV receiver is given as

$$\gamma_{c_m} \geq \gamma_{\mathrm{th}}. \tag{5.4}$$

Lemma 1: Given the fixed flight height H, c_m should be located in a horizontal disk region centered at the position that directly above b_m and having the radius R^* which can guarantee that the UAV successfully receives data. When $R_{(c_m,b_m)} = R^*$, the received SNR of the UAV at c_m is equal to γ_{th}.

Proof: See Appendix A.

Based on *Lemma 1*, we formally define a hovering disk region for each mission hovering point (excluding the start point c_0) as

$$O_m = \{c_m : ||c_m - b'_m|| = R_{(c_m,b_m)} \leq R^*\} \tag{5.5}$$

where $b'_m = (x_{b_m}, y_{b_m}, H) \in \mathbb{R}^3$ is the center of the disk O_m, and R^* is the radius to maintain a pre-defined quality-of-service, which can be found numerically. As long as the UAV enters a hovering disk region, it can collect data from the corresponding ground CH. The total data collection time of the UAV at $c_m \in O_m$ (or its hovering time) can be simply written as

$$T_{c_m}^{(\text{hov})} = N_m \tau + \frac{N_m L_{\text{data}}}{r_{c_m}} \tag{5.6}$$

where the first term in the right hand side is the time consumed for transmitting data from ordinary nodes to their corresponding CH b_m, and the second term is the data transmission time from b_m to the UAV. Therefore, the energy consumption of propulsion-related and communication-related activities of the UAV while hovering at c_m is expressed as

$$E_{c_m} = P_{\text{hov}} T_{c_m}^{(\text{hov})} + P_{\text{com}} \frac{N_m L_{\text{data}}}{r_{c_m}} \tag{5.7}$$

where P_{hov} and P_{com} are the UAV's powers for hovering and communication, respectively. After finishing the data collection task, b_m switches to the sleep model for saving energy. The UAV continues to select the next hovering point and executes the same processes to collect the sensed data from the corresponding ground cluster.

5.2.2 UAV's Mobility Model

Without loss of generality, the flight trajectory of the UAV can be seen as a permutation of the visiting order to M mission hovering points, with the start point being c_0, i.e., $\boldsymbol{c} =$

$\{c_0, c_1, \ldots, c_M\}$. The set of all possible permutations is denoted as $\mathbf{\Phi}$ with the size of $M!$. We represent one of the permutations as $\boldsymbol{\pi} = \{\pi(0), \ldots, \pi(M+1)\}$ and express the ordered hovering points as $\boldsymbol{c_\pi} = \{c_{\pi(0)}, c_{\pi(1)}, \ldots, c_{\pi(M)}, c_{\pi(M+1)}\}$, where $c_{\pi(t)}, t = 0, \ldots, M+1$, is the hovering point that is visited at step t in the trajectory, and $c_{\pi(0)} = c_{\pi(M+1)} = c_0$. For ease of understanding, if the hovering point c_m is visited at step t, its corresponding cluster of ground ordinary nodes (B_m) and the number of ordinary nodes (N_m) are redefined as $B_{\pi(t)}$ and $N_{\pi(t)}$, respectively.

After finishing data collection at $c_{\pi(t)}$ with the hovering model, the UAV horizontally flies to the next hovering point $c_{\pi(t+1)}$ along the line segment connecting $c_{\pi(t)}$ and $c_{\pi(t+1)}$. The flying time of the UAV during this period is given by

$$T^{(\text{fly})}_{(c_{\pi(t)}, c_{\pi(t+1)})} = \frac{||c_{\pi(t)} - c_{\pi(t+1)}||}{v_{\text{UAV}}} \tag{5.8}$$

where $||c_{\pi(t)} - c_{\pi(t+1)}||$ is the Euclidean distance between $c_{\pi(t)}$ and $c_{\pi(t+1)}$.

Following [33], the propulsion power consumption of the UAV for horizontal movement is the function of speed v_{UAV} and given by

$$P_{\text{mov}}(v_{\text{UAV}}) = P_0\left(1 + \frac{3v_{\text{UAV}}^2}{U_{\text{tip}}^2}\right) + P_1\left(\left(1 + \frac{v_{\text{UAV}}^4}{4v_0^4}\right)^{1/2} - \frac{v_{\text{UAV}}^2}{2v_0^2}\right)^{1/2} + \frac{1}{2}d_0\rho s_0\delta v_{\text{UAV}}^3 \tag{5.9}$$

where P_0 and P_1 represent, respectively, the blade profile power and induced power in the hovering state, U_{tip} is the tip speed of the rotor blade of the UAV, v_0 is the mean rotor induced velocity in the hovering state, d_0 denotes the fuselage drag ratio, s_0 represents the rotor solidity, ρ is the density of air, and δ denotes the area of the rotor disk. According to the analysis in [33], the power consumption $P_{\text{mov}}(v_{\text{UAV}})$ firstly decreases and then increases with the increasing value of the speed v_{UAV}. The energy consumption in the UAV's flight from $c_{\pi(t)}$ to $c_{\pi(t+1)}$ is computed as

$$E_{(c_{\pi(t)}, c_{\pi(t+1)})} = P_{\text{mov}}(v_{\text{UAV}})T^{(\text{fly})}_{(c_{\pi(t)}, c_{\pi(t+1)})}. \tag{5.10}$$

In the hovering state, the power consumption of the UAV can be obtained by substituting $v_{\text{UAV}} = 0$ into (5.9), $P_{\text{hov}} = P_0 + P_1$, which is a constant value.

5.2.3 Age of Information Model in a UAV-IoT System

We use the AoI metric to measure the freshness of information. According to the definition of AoI in [34], the AoI of a packet collected from node $b_{\pi(t)}^{(n)}$ in the $\pi(t)$-th visited cluster at time ζ is defined as

$$A_{\pi(t)}^{(n)}(\zeta) = \left(\zeta - u_{\pi(t)}^{(n)}(\zeta)\right)^{+} \tag{5.11}$$

where $u_{\pi(t)}^{(n)}(\zeta)$ is the instant at which the packet is generated, and $(x)^{+} = \max\{0, x\}$. When $\zeta < u_{\pi(t)}^{(n)}(\zeta)$, we define $A_{\pi(t)}^{(n)}(\zeta) = 0$, this is because the packet of node $b_{\pi(t)}^{(n)}$ has not been sampled. It is evident that the AoI of a packet will increase with time. In the considered UAV-IoT system, the BS is seen as the observer, thus, the AoI of a data packet can be seen as the amount of time elapsed from the instant at which the packet is generated to the instant at which the UAV flies back with the collected data to the BS.

For ease of analysis, for any ordinary node $b_{\pi(t)}^{(n)}, n = 1, \ldots, N_{\pi(t)}$ in the $\pi(t)$-th visited cluster, the AoI of its packet can be simply divided into two components. The first component is the time needed for the CH of its associated cluster to collect data from $b_{\pi(t)}^{(n)}$ and other nodes whose data have not been gathered (i.e., nodes $b_{\pi(t)}^{(n+1)}, \ldots, b_{\pi(t)}^{(N_{\pi(t)})}$) and forward the collected data to the UAV. The second component is the time consumed by the UAV to carry the packet of $b_{\pi(t)}^{(n)}$ to the end point $c_{\pi(M+1)}$. Specifically, this period includes the flight time of the UAV to unvisited ground clusters and the data collection time in these clusters. For example, after completing the data collection at $c_{\pi(t)}$, the UAV will fly to the next hovering point $c_{\pi(t+1)}$ and gather information from the corresponding cluster. During this period, the AoI of the packet of $b_{\pi(t)}^{(n)}$ increases with time, which is the sum of the flight time $T_{(c_{\pi(t)}, c_{\pi(t+1)})}$ from $c_{\pi(t)}$ to $c_{\pi(t+1)}$ and data collection time $T_{c_{\pi(t+1)}}$ at hovering point $c_{\pi(t+1)}$. Then, the UAV performs the same process to unvisited clusters until it returns to the end point. The time sequence of data collection in the UAV-IoT system is illustrated in Fig. 5.2. Mathematically, the total AoI of the packet generated by $b_{\pi(t)}^{(n)}$ in the UAV-IoT system is given as

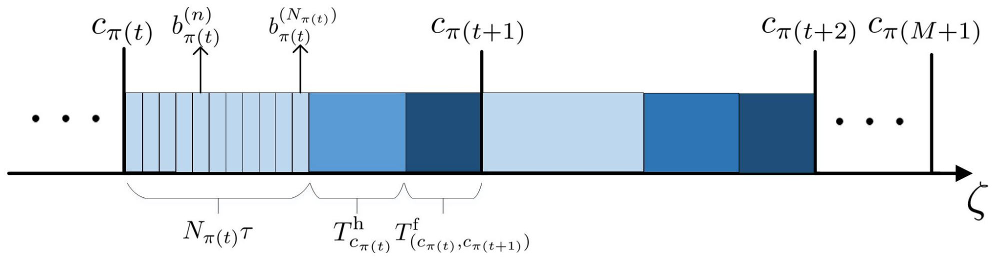

$c_{\pi(t)}$
$b_{\pi(t)}^{(n)}$
$b_{\pi(t)}^{(N_{\pi(t)})}$
$c_{\pi(t+1)}$
$c_{\pi(t+2)}$
$c_{\pi(M+1)}$
ζ
$N_{\pi(t)}\tau$
$T^{\mathrm{h}}_{c_{\pi(t)}}$
$T^{\mathrm{f}}_{(c_{\pi(t)},c_{\pi(t+1)})}$

Figure 5.2 The time sequence of data collection in the considered UAV-IoT system.

$$A_{\pi(t)}^{(n)} = \underbrace{\left(N_{\pi(t)} - (n-1)\right)\tau + \frac{N_{\pi(t)}L_{\text{data}}}{r_{c_{\pi(t)}}}}_{\text{first component}} + \underbrace{\sum_{g=t}^{M-1}\left(T^{(\text{fly})}_{(c_{\pi(g)},c_{\pi(g+1)})} + T^{(\text{hov})}_{c_{\pi(g+1)}}\right) + T^{(\text{fly})}_{(c_{\pi(M)},c_{\pi(M+1)})}}_{\text{second component}} \tag{5.12}$$

which can be further simplified as

$$A_{\pi(t)}^{(n)} = \sum_{g=t}^{M}\left(T^{(\text{hov})}_{c_{\pi(g)}} + T^{(\text{fly})}_{(c_{\pi(g)},c_{\pi(g+1)})}\right) - (n-1)\tau. \tag{5.13}$$

For packets of nodes in the same cluster, we have

$$A_{\pi(t)}^{(1)} > A_{\pi(t)}^{(2)} \cdots > A_{\pi(t)}^{(N_{\pi(t)})}. \tag{5.14}$$

On the other hand, the AoIs of packets in different clusters should satisfy

$$A_{\pi(1)}^{(n)} > A_{\pi(2)}^{(n)} > \cdots > A_{\pi(M)}^{(n)}. \tag{5.15}$$

5.2.4 Problem Formulation

The total AoI of all ordinary nodes in the network can be computed as

$$\overline{A} = \sum_{t=1}^{M}\sum_{n=1}^{N_{\pi(t)}} A_{\pi(t)}^{(n)} = \sum_{t=1}^{M}\sum_{n=1}^{N_{\pi(t)}}\sum_{g=t}^{M}\left(T^{(\text{hov})}_{c_{\pi(g)}} + T^{(\text{fly})}_{(c_{\pi(g)},c_{\pi(g+1)})}\right) - \sum_{t=1}^{M}\sum_{n=1}^{N_{\pi(t)}}(n-1)\tau. \tag{5.16}$$

According to (5.16), the total AoI is expressed as a weighted sum of the flight time of the UAV and the data collection time at each hovering point, which is determined by the locations of hovering points $\boldsymbol{c}$, the visiting order to these hovering points $\boldsymbol{\pi}$. It is evident that the hovering points of the UAV and its trajectory have a strong impact on the total AoI of data. If the position of any hovering point c_m is close to the center of the disk region O_m, a high data transmission rate can be achieved. As a result, the data transmission time from CHs to the UAV can be reduced, even though the UAV may have a longer flight trajectory, and hence the flight time. Conversely, if the UAV is located near the boundary of the disk region, the length of the UAV's trajectory might be reduced, but it will result in a lower data transmission rate, and hence increased data transmission time.

Our objective is to jointly find the hovering point from each disk and plan the visiting order to these hovering points for the UAV to minimize the total AoI of data in the considered UAV-IoT system. The optimization problem is expressed as follows:

$$\mathcal{P}_1 : \min_{\boldsymbol{c},\boldsymbol{\pi}} \quad \overline{A}\left(\boldsymbol{c},\boldsymbol{\pi}\right), \tag{5.17a}$$

$$\text{s.t.} \quad \boldsymbol{\pi} \in \boldsymbol{\Phi}, \tag{5.17b}$$

$$(5.4), (5.6), (5.8), (5.14), \text{ and } (5.15).$$

Constraint (5.17b) is the trajectory constraint. The SNR constraint is given in (5.4), and (5.6) is the data collection constraint. The flight time constraint is expressed as (5.8). AoI constraints are (5.14) and (5.15). It is evident that the formulated problem $\mathcal{P}_1$ is a TSPN [35], which combines the determination of hovering points at each disk with the problem of trajectory planning of the UAV. The traditional TSPN problem involves finding a minimum-cost tour (i.e., the total length of the tour is minimum) that travels each region exactly once for a collection of compact regions before returning to the initial departure point [36]. However, our formulated problem not only considers the traveling cost but also the cost spent at each hovering point. The problem $\mathcal{P}_1$ is extremely challenging because it is composed of a continuous problem (optimization of hovering points $\boldsymbol{c}$) and a combinatorial problem (optimization of visiting order $\boldsymbol{\pi}$). Given a set of hovering points $\boldsymbol{c}$, the optimization of $\boldsymbol{\pi}$ can be viewed as the TSP, which can be normally be solved quite effectively by some dedicated TSP solvers, such as Concorde [37], etc. However, the optimization of hovering points $\boldsymbol{c}$ consists of an infinite number of variables, which is infeasible to be solved optimally. To reduce computational time, we leverage the sampling approach that samples finite discrete sets of hovering points from a continuous state space to transform the continuous TSPN in $\mathcal{P}_1$ into the GTSP. Specifically, each disk O_m is equally partitioned into $L_{\text{sub}} \times L_{\text{sub}}$ sub-regions and the center of each sub-region is selected as the possible hovering point. For some marginal sub-regions with non-square shape, we choose the centers of their actual areas. Hence, we can obtain a cluster G_m of sampling points with the size of L_{sub}^2 from O_m. As a result, our objective is changed to jointly select hovering points from M clusters of sampled hovering points and plan the UAV's trajectory to visit selected hovering points exactly once to minimize the total AoI. Using the sampling approach, the formulated problem $\mathcal{P}_1$ is converted to

$$
\begin{aligned}
\mathcal{P}_2 : \min_{\boldsymbol{c},\boldsymbol{\pi}} \quad & \overline{A}\left(\boldsymbol{c},\boldsymbol{\pi}\right), && \text{(5.18a)}\\
\text{s.t.} \quad & c_m \in G_m, G_m \in O_m, m \in \{1,\ldots,M\}, && \text{(5.18b)}\\
& (5.4), (5.6), (5.8), (5.14), (5.15), \text{ and } (5.17b).
\end{aligned}
$$

Obviously, the formulated problem $\mathcal{P}_2$ is a combinatorial optimization problem, and hence, NP-hard. There are two traditional methods to handle combinatorial problems: exact algorithms and heuristic algorithms. Exact algorithms can find optimal solutions, but they will become intractable when the size of problems grows. Heuristic algorithms' complexity is polynomial and they commonly find sub-optimal solutions. In contrast, we cast the proposed GTSP as a sequence-to-sequence problem where the source sequence is a set of clusters of hovering points and CHs and the target sequence is a set of selected hovering points and the visiting order to these points. We adopt the transformer, the weighted A*, and reinforcement learning to efficiently solve this problem.

5.3 Transformer-Weighted A* Algorithm

Because the UAV needs to sequentially collect data from each ground cluster in the IoT network, we view the problem of the total AoI-minimal trajectory planning as a "machine translation" problem that is common in natural language processing. The whole UAV-IoT network as the "source language" is translated into the "target language", i.e., the UAV trajectory, by using our proposed TWA* algorithm. The TWA* algorithm is composed of an encoder network, a decoder network, and the weighted A* search algorithm which can effectively find the trajectory policy from hidden patterns behind a large number of training datasets.

5.3.1 Encoder

The role of the encoder network is to take the UAV-IoT network represented as an input sequence and map it into an abstract representation that is the learned information. The input sequence includes the start point of the UAV, each and every CH, number of nodes in each ground cluster, and all sampling points from each hovering disk. Specifically, we define $\boldsymbol{h}_0^{(\mathrm{in})} = c_0 \in \mathbb{R}^3$, and $\boldsymbol{h}_m^{(\mathrm{in})} = (G_m, b_m, N_m) \in \mathbb{R}^{3(L_{\mathrm{sub}}^2+1)+1}, m \in \{1, \ldots, M\}$, where the cluster G_m of sampling points is represented as a $3L_{\mathrm{sub}}^2$-dimensional vector as it includes L_{sub}^2 points with 3D Cartesian coordinates, the CH b_m is a 3-dimensional vector, and the number of nodes N_m is a constant. Hence, the input can be expressed as $\boldsymbol{H}^{(\mathrm{in})} = \left(\boldsymbol{h}_0^{(\mathrm{in})}; \boldsymbol{h}_1^{(\mathrm{in})}; \ldots; \boldsymbol{h}_M^{(\mathrm{in})}\right)$.

The encoder network used in this paper is the standard transformer encoder with one embedding layer and six identical encoder layers as in [29]. Each encoder layer is composed of one multi-head self attention sub-layer and one point-wise feed-forward network sub-layer. Each sub-layer adds a residual connection and layer normalization. The embedding layer is to map each element of input to the d_{em}-dimensional vector space by a learnable linear projection. Specifically, to enable the model to distinguish the start point of the UAV from clusters, we separately utilize different parameters to compute the embeddings of the start point and the other clusters as follows:

$$\boldsymbol{h}_m^{(0)} = \begin{cases} \boldsymbol{W}_0 \boldsymbol{h}_m^{(\mathrm{in})} + \boldsymbol{W}_{b_0}, & m = 0 \\ \boldsymbol{W}_1 \boldsymbol{h}_m^{(\mathrm{in})} + \boldsymbol{W}_b, & m = 1, \ldots, M \end{cases} \tag{5.19}$$

where $\boldsymbol{W}_0 \in \mathbb{R}^{d_{\mathrm{em}} \times 3}$, $\boldsymbol{W}_1 \in \mathbb{R}^{d_{\mathrm{em}} \times (3(L_{\mathrm{sub}}^2+1)+1)}$, $\boldsymbol{W}_{b_0} \in \mathbb{R}^{d_{\mathrm{em}}}$, and $\boldsymbol{W}_b \in \mathbb{R}^{d_{\mathrm{em}}}$ are learnable parameters. Then, the embeddings $\boldsymbol{H}^{(0)} = \left(\boldsymbol{h}_0^{(0)}; \boldsymbol{h}_1^{(0)}; \ldots; \boldsymbol{h}_M^{(0)}\right) \in \mathbb{R}^{(M+1) \times d_{\mathrm{em}}}$ are fed into the encoder layers. Note that we do not consider the positional decoding used in the original transformer in [25] because the order of the input sequence is irrelevant to the GTSP.

The attention layer in each encoder layer uses the multi-head self-attention mechanism with 8 heads to jointly attend to information from different representation subspaces at different positions. The 8 heads perform the attention calculation in parallel and their results are merged to produce an input for the next step. In the encoder layer $l, l = 1, \dots, 6$, the output of self-attention on the h-th head, $h = 1, \dots, 8$, is computed as

$$\boldsymbol{Z}_h^{(l)} = \text{Attention}(\boldsymbol{Q}_h^{(l)}, \boldsymbol{K}_h^{(l)}, \boldsymbol{V}_h^{(l)}) = \text{softmax}\left(\frac{\boldsymbol{Q}_h^{(l)} {\boldsymbol{K}_h^{(l)}}^T}{\sqrt{d_\text{v}}}\right) \boldsymbol{V}_h^{(l)} \tag{5.20}$$

where d_v is used for scaling the dot products, $\boldsymbol{Q}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$, $\boldsymbol{K}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$, and $\boldsymbol{V}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$ are matrices query, key, and value for the h-th head, respectively. They can be created by projecting the input query $\boldsymbol{Q}^{(l)}$, key $\boldsymbol{K}^{(l)}$, and value $\boldsymbol{V}^{(l)}$ of multi-head self attention with three learnable weight matrices $\boldsymbol{W}_h^{Q(l)} \in \mathbb{R}^{d_\text{em}\times d_\text{v}}$, $\boldsymbol{W}_h^{K(l)} \in \mathbb{R}^{d_\text{em}\times d_\text{v}}$, and $\boldsymbol{W}_h^{V(l)} \in \mathbb{R}^{d_\text{em}\times d_\text{v}}$, respectively, as follows

$$\boldsymbol{Q}_h^{(l)} = \boldsymbol{Q}^{(l)}\boldsymbol{W}_h^{Q(l)}, \boldsymbol{K}_h^{(l)} = \boldsymbol{K}^{(l)}\boldsymbol{W}_h^{K(l)}, \boldsymbol{V}_h^{(l)} = \boldsymbol{V}^{(l)}\boldsymbol{W}_h^{V(l)} \tag{5.21}$$

where $\boldsymbol{Q}^{(l)} = \boldsymbol{K}^{(l)} = \boldsymbol{V}^{(l)} = \boldsymbol{H}^{(l-1)}$. In this paper $\boldsymbol{H}^{(l-1)}$ is the output of the encoder layer $(l-1)$ or the output of the embedding layer before the encoder layer 1. Matrices $\boldsymbol{Q}_h^{(l)}$, $\boldsymbol{K}_h^{(l)}$, and $\boldsymbol{V}_h^{(l)}$ can be further expressed as

$$\boldsymbol{Q}_h^{(l)} = \begin{pmatrix} \boldsymbol{q}_0 \\ \vdots \\ \boldsymbol{q}_M \end{pmatrix}, \boldsymbol{K}_h^{(l)} = \begin{pmatrix} \boldsymbol{k}_0 \\ \vdots \\ \boldsymbol{k}_M \end{pmatrix}, \boldsymbol{V}_h^{(l)} = \begin{pmatrix} \boldsymbol{v}_0 \\ \vdots \\ \boldsymbol{v}_M \end{pmatrix} \tag{5.22}$$

where $\forall \boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v} \in \mathbb{R}^{d_\text{v}}$. Then, we can obtain the scaled attention scores

$$\frac{\boldsymbol{Q}_h^{(l)} \left(\boldsymbol{K}_h^{(l)}\right)^T}{\sqrt{d_\text{v}}} = \frac{1}{\sqrt{d_\text{v}}} \begin{pmatrix} (\boldsymbol{q}_0, \boldsymbol{k}_0) & \dots & (\boldsymbol{q}_0, \boldsymbol{k}_M) \\ (\boldsymbol{q}_1, \boldsymbol{k}_0) & \dots & (\boldsymbol{q}_1, \boldsymbol{k}_M) \\ \dots & (\boldsymbol{q}_i, \boldsymbol{k}_j) & \dots \\ (\boldsymbol{q}_M, \boldsymbol{k}_0) & \dots & (\boldsymbol{q}_M, \boldsymbol{k}_M) \end{pmatrix} = \begin{pmatrix} u_{00} & \dots & u_{0M} \\ u_{10} & \dots & u_{1M} \\ \dots & u_{ij} & \dots \\ u_{M0} & \dots & u_{MM} \end{pmatrix} \tag{5.23}$$

where $(\boldsymbol{q}_i, \boldsymbol{k}_j), i, j \in \{0, \dots, M\}$ is the inner product of vectors, which measures the similarity of vector $\boldsymbol{q}_i$ and vector $\boldsymbol{k}_j$. The row-wise *softmax* function is used on each element of the above scaled attention scores matrix, which is given by $\overline{u}_{ij} = e^{u_{ij}} / \sum_{j'=0}^{M} e^{u_{ij'}}$. Then, the output $\boldsymbol{Z}_h^{(l)} \in \mathbb{R}^{(M+1)\times d_\text{v}}$ of the h-th head is expressed as

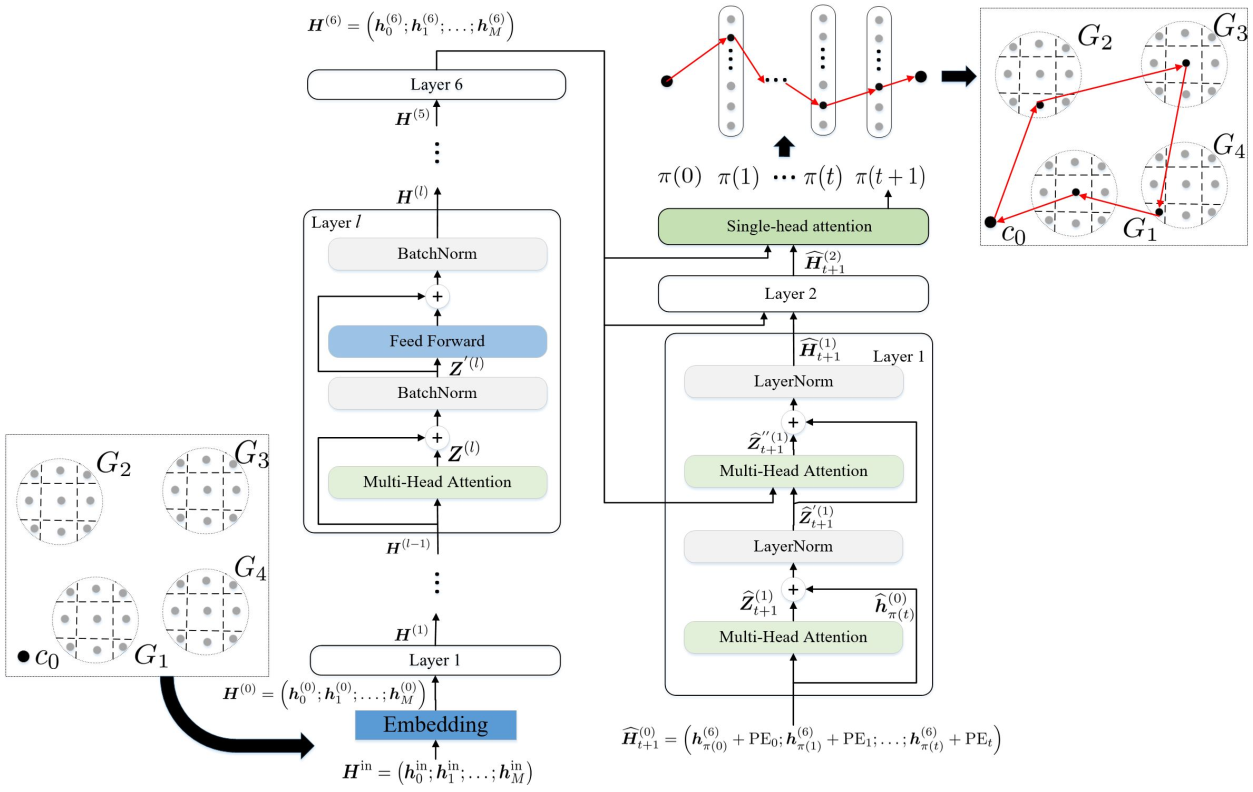

$H^{(6)} = (h_0^{(6)}; h_1^{(6)}; \dots; h_M^{(6)})$
Layer 6
$H^{(5)}$
$H^{(l)}$
Layer l
BatchNorm
+
Feed Forward
$Z'^{(l)}$
BatchNorm
+
$Z^{(l)}$
Multi-Head Attention
$H^{(l-1)}$
$H^{(1)}$
Layer 1
$H^{(0)} = (h_0^{(0)}; h_1^{(0)}; \dots; h_M^{(0)})$
Embedding
$H^{\text{in}} = (h_0^{\text{in}}; h_1^{\text{in}}; \dots; h_M^{\text{in}})$
G_1
G_2
G_3
G_4
c_0
$\pi(0)$ $\pi(1)$ $\cdots$ $\pi(t)$ $\pi(t+1)$
Single-head attention
$\widehat{H}_{t+1}^{(2)}$
Layer 2
$\widehat{H}_{t+1}^{(1)}$
Layer 1
LayerNorm
+
$\widehat{Z}''^{(1)}_{t+1}$
Multi-Head Attention
$\widehat{Z}'^{(1)}_{t+1}$
LayerNorm
+
$\widehat{Z}^{(1)}_{t+1}$
$h^{(0)}_{\pi(t)}$
Multi-Head Attention
$\widehat{H}^{(0)}_{t+1} = (h^{(6)}_{\pi(0)} + \mathrm{PE}_0; h^{(6)}_{\pi(1)} + \mathrm{PE}_1; \dots; h^{(6)}_{\pi(t)} + \mathrm{PE}_t)$

Figure 5.3 The proposed algorithm framework.

$$
\boldsymbol{Z}_h^{(l)} = \begin{pmatrix} \sum_{j=0}^{M} \overline{u}_{0j} \boldsymbol{v}_j \\ \sum_{j=0}^{M} \overline{u}_{1j} \boldsymbol{v}_j \\ \vdots \\ \sum_{j=0}^{M} \overline{u}_{Mj} \boldsymbol{v}_j \end{pmatrix} = \begin{pmatrix} \overline{\overline{\boldsymbol{u}}}_0 \\ \overline{\overline{\boldsymbol{u}}}_1 \\ \vdots \\ \overline{\overline{\boldsymbol{u}}}_M \end{pmatrix}. \tag{5.24}
$$

Hence, we end up with 8 different outputs from 8 heads where each head could learn something different. These outputs are concatenated and multiplied by an additional learnable weight matrix $\boldsymbol{W}_{\text{o}}^{(l)} \in \mathbb{R}^{8d_{\text{v}} \times d_{\text{em}}}$ to generate the final output of the multi-head attention layer, as follows:

$$
\boldsymbol{Z}^{(l)} = \left(\boldsymbol{Z}_1^{(l)}, \dots, \boldsymbol{Z}_8^{(l)}\right) \boldsymbol{W}_{\text{o}}^{(l)}, \ \boldsymbol{Z}^{(l)} \in \mathbb{R}^{(M+1) \times d_{\text{em}}}. \tag{5.25}
$$

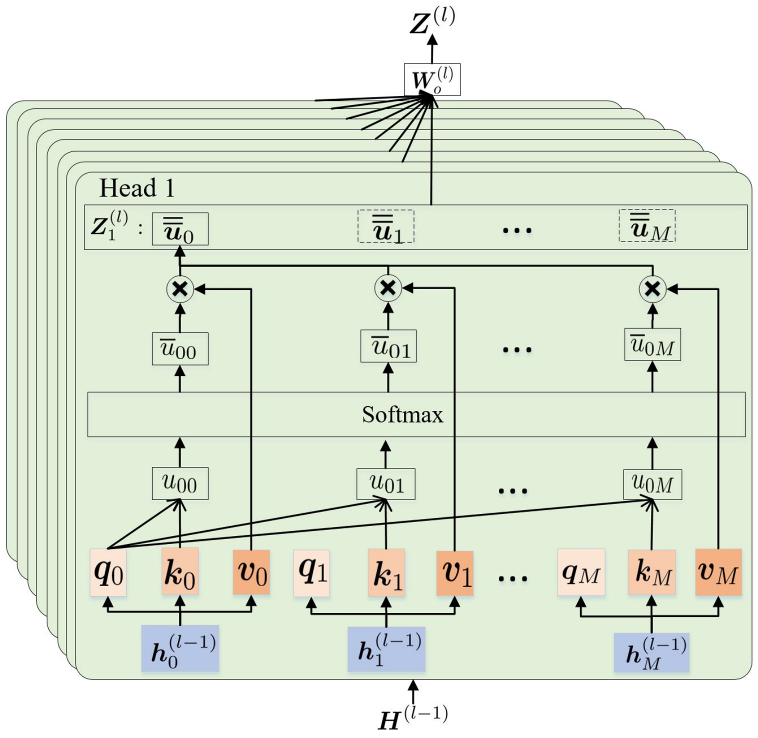


Figure 5.4 Multi-head self attention.

To facilitate the understanding of multi-head attention layer, all operations from (5.20) to (5.25) are defined as a function MHA$(\cdot)$. Thus, $\boldsymbol{Z}^{(l)} = \text{MHA}\left(\boldsymbol{Q}^{(l)}, \boldsymbol{K}^{(l)}, \boldsymbol{V}^{(l)}\right)$. Then, $\boldsymbol{Z}^{(l)}$ is added to the input of the multi-head attention in this encoder, which is a residual connection operation. Subsequently, the output of the residual connection is fed into a batch normalization, defined as a function BN$(\cdot)$, and it is written as $\boldsymbol{Z}'^{(l)} = \text{BN}\left(\boldsymbol{H}^{(l-1)} + \boldsymbol{Z}^{(l)}\right)$. The use of the residual connection is to avoid the degradation problem of the network in training, while the layer normalization can improve the training speed and the stability of the networks. The normalized residual output goes through a pointwise feed-forward network (defined as a function FFN$(\cdot)$), which is a couple of linear layers with a ReLU activation in between. Then, the output of the pointwise feed-forward network is added to its input by a residual connection and further normalized to obtain the final output $\boldsymbol{H}^{(l)} \in \mathbb{R}^{(M+1)\times d_{\text{em}}}$ of the encoder layer l, which is given by $\boldsymbol{H}^{(l)} = \text{BN}\left(\boldsymbol{Z}'^{(l)} + \text{FFN}\left(\boldsymbol{Z}'^{(l)}\right)\right)$. In each encoder layer, we perform the same computational process and finally output the final result of the encoder part at layer 6, $\boldsymbol{H}^{(6)} = \left(\boldsymbol{h}_0^{(6)}; \boldsymbol{h}_1^{(6)}; \ldots; \boldsymbol{h}_M^{(6)}\right) \in \mathbb{R}^{(M+1)\times d_{\text{em}}}$, which is the continuous representation with attention information of the input $\boldsymbol{H}^{(\text{in})}$. All of these operations will help the decoder network focus on the appropriate elements in the input during the decoding process.

5.3.2 Decoder

The decoding is autoregressive and generates the result one by one. The output of the decoder network can be represented as an ordered sequence of the input of the encoder. The decoder begins with the start point at decoding step 0 since the trajectory of the UAV should start at the start point as well as end at this point. The output of each decoding step is based on the information from the encoder and the already-generated previous output in the decoder. Hence, the decoding process can be modelled using the probability chain rule:

$$P(\boldsymbol{\pi}|\boldsymbol{H}^{(\mathrm{in})}) = \prod_{t=0}^{M+1} P(\pi(t)|\pi(0), \ldots, \pi(t-1), \boldsymbol{H}^{(\mathrm{in})}). \tag{5.26}$$

The decoding process aims at finding the optimal $\boldsymbol{\pi}$ to maximize $P(\boldsymbol{\pi}|\boldsymbol{H}^{(\mathrm{in})})$.

The decoder network is composed of two identical decoder layers, and a single-head attention layer. Each decoder layer contains two multi-head attention sub-layers which employ a residual connection around them followed by layer normalization. These sub-layers have the same structure as the sub-layers in the encoder network but each of them has a different job. Since the output of the decoder network is related to the order, we need to inject some information about the positions into the input sequence of the decoder network. The locations are implicitly represented by the order of the data input to the decoder network. Hence, the input of the decoder network is the output of the encoder network combined with the positional encoding. Suppose the outputs of the decoder network at previous t decoding steps are $\pi(0), \pi(1), \ldots, \pi(t)$, the decoder wants to predict the output at $t+1$ step. Then, the input to the decoder network is expressed as $\widehat{\boldsymbol{H}}_{t+1}^{(0)} = \left(\widehat{\boldsymbol{h}}_{\pi(0)}^{(0)}; \widehat{\boldsymbol{h}}_{\pi(1)}^{(0)}; \ldots; \widehat{\boldsymbol{h}}_{\pi(t)}^{(0)}\right)$. Each element in $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ can be calculated by $\widehat{\boldsymbol{h}}_{\pi(t)}^{(0)} = \boldsymbol{h}_{\pi(t)}^{(6)} + \mathrm{PE}_t$, where $\boldsymbol{h}_{\pi(t)}^{(6)} \in \mathbb{R}^{1\times d_{\mathrm{em}}}$ is one element in $\boldsymbol{H}^{(6)}$ which is decoded at the t-th step, $\mathrm{PE}_t \in \mathbb{R}^{1\times d_{\mathrm{em}}}$ is the positional encoding based on the sinusoidal function, which is given by [38]

$$\mathrm{PE}_t(d_i) = \begin{cases} \sin(\omega_{d_i} t), & \text{if } d_i \text{ is even} \\ \cos(\omega_{d_i} t), & \text{if } d_i \text{ is odd} \end{cases} \tag{5.27}$$

where d_i is the dimension, $1 \leqslant d_i \leqslant d_{\text{em}}$, ω_{d_i} is the hand-crafted frequency for each dimension. The position encoding of each position successfully provides the position information to the decoder network. The input $\widehat{\boldsymbol{H}}^{(0)}_{t+1}$ gets fed into the first multi-head attention sub-layer of the first decoder layer and pass through the residual connection and layer normalization (denoted as a function $\mathrm{LN}(\cdot)$) to prepare the query for the next multi-head attention sub-layer, as follows

$$\widehat{\boldsymbol{Z}}^{(1)}_{t+1} = \mathrm{MHA}\left(\widehat{\boldsymbol{h}}^{(0)}_{\pi(t)}, \widehat{\boldsymbol{H}}^{(0)}_{t+1}, \widehat{\boldsymbol{H}}^{(0)}_{t+1}\right), \widehat{\boldsymbol{Z}}^{(1)}_{t+1} \in \mathbb{R}^{1\times d_{\text{em}}} \tag{5.28}$$

$$\widehat{\boldsymbol{Z}}'^{(1)}_{t+1} = \mathrm{LN}\left(\widehat{\boldsymbol{h}}^{(0)}_{\pi(t)} + \widehat{\boldsymbol{Z}}^{(1)}_{t+1}\right), \widehat{\boldsymbol{Z}}'^{(1)}_{t+1} \in \mathbb{R}^{1\times d_{\text{em}}} \tag{5.29}$$

where $\widehat{\boldsymbol{h}}^{(0)}_{\pi(t)}$ is the query, $\widehat{\boldsymbol{H}}^{(0)}_{t+1}$ works as the key and the value matrices in the current multi-head attention sub-layer. The second multi-head attention sub-layer is used to match the encoder's input to the decoder's input to allow the decoder network to decide the next possible output among the non-visited elements. For this sub-layer, the encoder network's output $\boldsymbol{H}^{(6)}$ is the key and the value matrices, and $\widehat{\boldsymbol{Z}}'^{(1)}$ is the query matrix. The calculations are given by

$$\widehat{\boldsymbol{Z}}''^{(1)}_{t+1} = \mathrm{MHA}\left(\widehat{\boldsymbol{Z}}'^{(1)}_{t+1}, \boldsymbol{H}^{(6)}, \boldsymbol{H}^{(6)}\right), \widehat{\boldsymbol{Z}}''^{(1)}_{t+1} \in \mathbb{R}^{1\times d_{\text{em}}} \tag{5.30}$$

$$\widehat{\boldsymbol{H}}^{(1)}_{t+1} = \mathrm{LN}\left(\widehat{\boldsymbol{Z}}'^{(1)}_{t+1} + \widehat{\boldsymbol{Z}}''^{(1)}_{t+1}\right), \widehat{\boldsymbol{H}}^{(1)}_{t+1} \in \mathbb{R}^{1\times d_{\text{em}}}. \tag{5.31}$$

Note that we add the mask of visited elements to the scaled attention scores in this sub-layer. Then, $\widehat{\boldsymbol{H}}^{(1)}_{t+1}$ goes through the second decoder layer to get the output $\widehat{\boldsymbol{H}}^{(2)}_{t+1} \in \mathbb{R}^{1\times d_{\text{em}}}$. In order for the decoder network to compute output probabilities $P(\pi(t+1)|\pi(0),\ldots,\pi(t),\boldsymbol{H}^{(6)})$, $\widehat{\boldsymbol{H}}^{(2)}_{t+1}$ and the output $\boldsymbol{H}^{(6)}$ of the encoder network get fed into a single-head attention to get a distribution over the non-visited elements, which is given by [29]

$$\boldsymbol{P}_{t+1} = \mathrm{softmax}\left(\tanh\left(\frac{\widehat{\boldsymbol{Q}}_{t+1}\widehat{\boldsymbol{K}}^{T}_{t+1}}{\sqrt{d_{\text{em}}}} \odot \mathcal{M}_{t+1}\right)\right) \tag{5.32}$$

where $\widehat{\boldsymbol{Q}}_{t+1} = \widehat{\boldsymbol{H}}^{(2)}_{t+1}\widehat{\boldsymbol{W}}_1, \widehat{\boldsymbol{K}}_{t+1} = \boldsymbol{H}^{(6)}\widehat{\boldsymbol{W}}_2$, $\widehat{\boldsymbol{W}}_1 \in \mathbb{R}^{d_{\text{em}}\times d_{\text{em}}}$ and $\widehat{\boldsymbol{W}}_2 \in \mathbb{R}^{d_{\text{em}}\times d_{\text{em}}}$ are learnable weight matrices, $\mathcal{M}_{t+1}$ is the mask of the visited elements considered in this layer, $\odot$ is the Hadamard product, and $\boldsymbol{P}_{t+1} \in \mathbb{R}^{1\times(M+1)}$ is the distribution over the non-visited elements, which is composed of probability scores. Then, the output that will be selected is sampled from the distribution with three decoding methods:

Greedy

At each decoding step, this method greedily selects the element with the largest probability $P(\pi(t+1)|\pi(0), \ldots, \pi(t), \boldsymbol{H}^{(6)})$.

Random Sampling

This method randomly samples W_{sampling} solutions, where each solution includes fully visiting order, and selects the solution with the highest probability as the final result.

Beam Search

This method chooses the top W_{beam} possible solutions that have the highest probability at each step, where W_{beam} is the beam width. Those W_{beam} solutions will move to the next time step, and the process repeats. Then, we can obtain a tree of solutions of each step and the $\boldsymbol{\pi}$ that has the highest overall probability is picked as the final result.

We assume that the index of the highest probability score in $\boldsymbol{P}_{t+1}$ is selected with the greedy decoding as the output $\pi(t+1)$ at step $t+1$. Thus $\pi(t+1)$ points to the element at the same position of the input sequence $\boldsymbol{H}^{(\text{in})}$ of the encoder network, which is represented as $\boldsymbol{h}^{(\text{it})}_{\pi(t+1)}$. Then, the decoder network takes the encoding information of $\boldsymbol{h}^{(\text{it})}_{\pi(t+1)}$ from $\boldsymbol{H}^{(6)}$, i.e., $\boldsymbol{h}^{(6)}_{\pi(t+1)}$, and adds it with its position encoding to the list of the decoder input to continue decoding for the next step. Finally, we can obtain a set of the visiting order, $\boldsymbol{\pi}$. As shown in the example in Fig. 5.3, $\boldsymbol{H}^{(\text{in})} = (c_0; (G_1, b_1, N_1); (G_2, b_2, N_2); (G_3, b_3, N_3); (G_4, b_4, N_4))$ is the input to the encoder network and the decoder network outputs the final visiting order $\boldsymbol{\pi} = \{\pi(0), \pi(1), \pi(2), \pi(3), \pi(4)\}$ to elements in $\boldsymbol{H}^{(\text{in})}$.

5.3.3 Selection of Hovering Points

Given the visiting order $\boldsymbol{\pi}$, we know the visiting order to all hovering points clusters and construct a graph containing all of them, as illustrated in Fig. 5.3. Each layer of the graph is composed of one hovering points cluster. Then, we will calculate the path with the minimal total AoI starting from the start point (marked as $\pi(0)$ in the visiting order), going through each cluster G_m, and ending at the clone of the start point (marked as $\pi(M+1)$ in the visiting order). To guarantee that at most one hovering point is selected from each cluster, we assume that all edges between possible hovering points of consecutive clusters to be directed by $\boldsymbol{\pi}$. We use the weighted A* search algorithm [39] to quickly find the hovering point from each cluster to build the path with the minimal cost (total AoI). We assume that the UAV currently reaches the point s' and will decide the next point to be expanded by the following cost function

$$f(s) = g(s) + \omega h(s) \tag{5.33}$$

where s is any neighbor point of s', $g(s)$ is the total movement cost on the path from the start point c_0 to s, $h(s)$ is the heuristic function to estimate cost from s to the end point ${c_0}'$, and $\omega > 1$ is a constant factor. The neighbor point with a minimal $f(s)$ value is expanded. The pseudocode is described in Algorithm 9. We use COST and FRONTIER to keep track of $g(s)$ and the expanding process, respectively. Each point that has been reached keeps a pointer to its parent in CAME_FROM so that we can know where it came from. With CAME_FROM, we can construct a path having the minimal AoI from the start point to the end point, as illustrated by the solid red line with arrow in the example in Fig. 5.3.

5.3.4 Computational Complexity Analysis

In the encoder network, each encoder layer is the standard transformer encoder with the quadratic computational complexity $O((M+1)^2 d_{\text{em}})$ [25]. Since the number of layers is constant, the computational complexity of the encoder network is still $O((M+1)^2 d_{\text{em}})$. In the decoder network, although each decoder layer contains two multi-head attention sub-layers, its computational complexity is still estimated to be quadratic $O((M+1)^2 d_{\text{em}})$ [25]. Likewise, the number of decoder layers does not affect the computational complexity of the decoder network. In addition, the computational complexity of the single-head attention used in the final step of the decoder network is also quadratic $O((M+1)^2 d_{\text{em}})$ [25]. Hence, the used transformer model has the computational complexity $O((M+1)^2 d_{\text{em}})$, which is quadratic in the length of the input sequence. Different data structures used to implement the weight A* algorithm, and hence, affect its computational complexity. We use the min heap to implement

Algorithm 9 Pseudocode for weighted A* search algorithm to find hovering points

```
Input: created graph
 1: FRONTIER = PriorityQueue()
 2: FRONTIER.put($c_0$, 0)
 3: CAME_FROM = [ ]
 4: COST = [ ]
 5: CAME_FROM[$c_0$] = None
 6: COST[$c_0$] = 0
 7: while FRONTIER is not empty do
 8:     current point $s'$ = FRONTIER.get()
 9:     if $s' = {c_0}'$ then
10:         break
11:     end if
12:     for each neighbor $s$ of $s'$ do
13:         $g(s)$ = COST[$s'$]+ the total AoI from $s'$ to $s$
14:         if $s$ not in COST or $g(s) <$ COST[$s$] then
15:             COST[$s$] = g(s)
16:             $f(s) = g(s) + \omega h(s)$
17:             FRONTIER.put($s$, $f(s)$)
18:             CAME_FROM[$s$] = $s'$
19:         end if
20:     end for
21: end while
22: calculate $\overline{A}$ according to CAME_FROM
```

the weight A* algorithm. We assume that at most ML_{sub}^2 points (the total number of points in the search graph) are visited, and the min heap uses $O(\log(ML_{sub}^2))$ computational complexity to extract a point each time [40]. The weighted A* algorithm's computational complexity is estimated to be $O(ML_{sub}^2 \log(ML_{sub}^2)$. Hence, the computational complexity of the proposed algorithm is $O((M+1)^2 d_{\text{em}}) + O(ML_{sub}^2 \log(ML_{sub}^2)$.

Algorithm 10 Training TWA* by REINFORCE with rollout baseline

```
Input: Epochs $E_{\text{epochs}}$, training steps $S$, batch size $B_{\text{size}}$
 1: Initialize parameters $\vartheta$, $\vartheta^{(\text{BL})} \leftarrow \vartheta$
 2: for epoch = 1 to $E_{\text{epochs}}$ do
 3:     for step = 1 to $S$ do
 4:         $\boldsymbol{H}_i^{(\text{in})} \leftarrow$ generate instances() $\forall i \in \{1, \ldots, B_{\text{size}}\}$
 5:         $\boldsymbol{\pi}_i \leftarrow$ Sampling solution$P_{\vartheta}\left(\cdot|\boldsymbol{H}_i^{(\text{in})}\right)$
 6:         $\boldsymbol{\pi}_i^{(\text{BL})} \leftarrow$ Greedy solution$P_{\vartheta^{(\text{BL})}}\left(\cdot|\boldsymbol{H}_i^{(\text{in})}\right)$
 7:         $\overline{A}_i \leftarrow$ weighted A* $(\boldsymbol{\pi}_i)$
 8:         $\overline{A}_i^{(\text{BL})} \leftarrow$ weighted A* $\left(\boldsymbol{\pi}_i^{(\text{BL})}\right)$
 9:         $\nabla_{\vartheta} J \leftarrow \sum_{i=1}^{B_{\text{size}}} \left(\overline{A}_i - \overline{A}_i^{(\text{BL})}\right) \nabla_{\vartheta} \log P_{\vartheta}\left(\boldsymbol{\pi}_i|\boldsymbol{H}_i^{(\text{in})}\right)$
10:         $\vartheta \leftarrow \text{Adam}(\vartheta, \nabla_{\vartheta} J)$
11:     end for
12:     if $t$-test$(P_{\vartheta}(\cdot), P_{\vartheta^{(\text{BL})}}(\cdot)) < 5\%$ then
13:         $\vartheta^{(\text{BL})} \leftarrow \vartheta$
14:     end if
15: end for
```

5.3.5 Training

To enable the transformer model to produce the optimal $\boldsymbol{\pi}$, we use the well-known policy gradient approaches to train it. The transformer model is parameterized by ϑ, which includes all trainable variables in the encoder and the decoder networks. We regard the UAV as an agent to learn a good policy $\boldsymbol{\pi}$ to maximize long-term rewards by iteratively interacting with the environment to optimize parameter ϑ. At each step, the agent in a given state chooses an action by its decision policy, which actually is the mapping from states to actions.

State

The state consists of the environment encoded by the encoder network and the visited clusters before the current step in the decoder, which are $\boldsymbol{H}^{(6)}$ and $\widehat{\boldsymbol{H}}_{t+1}^{(0)}$ in the transformer, respectively.

Action

At each step, the agent makes an action $\pi(t)$ based on its state, which can be seen as the processes of the right-hand side of (5.26). Thus, we view all operations in the decoder network as the action.

Reward

The negative of the total AoI $\overline{A}$ in (5.16) is used as the reward.

Our objective for training is given by

$$J\left(\vartheta|\boldsymbol{H}^{(\text{in})}\right) = \mathbb{E}_{\boldsymbol{\pi}\sim P_{\vartheta}\left(\cdot|\boldsymbol{H}^{(\text{in})}\right)}\left(\overline{A}\right). \tag{5.34}$$

The gradient of (5.34) is calculated using the REINFORCE algorithm [41] with the greedy rollout baseline $\overline{A}^{(\text{BL})}$ [27]

$$\nabla_{\vartheta} J\left(\vartheta|\boldsymbol{H}^{(\text{in})}\right) = \mathbb{E}_{\boldsymbol{\pi}\sim P_{\vartheta}\left(\cdot|\boldsymbol{H}^{(\text{in})}\right)}\left[\left(\overline{A} - \overline{A}^{(\text{BL})}\right)\nabla_{\vartheta}\log P_{\vartheta}\left(\boldsymbol{\pi}|\boldsymbol{H}^{(\text{in})}\right)\right] \tag{5.35}$$

where $\overline{A}$ is the cost of a solution that is obtained from the current training transformer model by sampling decoding. We set the greedy policy as the baseline policy in our model, and hence, $\overline{A}^{(\text{BL})}$ is the cost of a solution of the deterministic greedy decoding, which is used to eliminate variance during training. By doing so, the transformer model is trained to improve over its (greedy) self. The training process is summarized in Algorithm 10. In each training step, new instances are generated first (line 4). Then, the transformer model uses sampling decoding and greedy decoding to produce $\boldsymbol{\pi}_i$ and $\boldsymbol{\pi}_i^{(\text{BL})}$ (lines 5 and 6), respectively. The total AoIs are further obtained from the weighted A* (lines 7 and 8). The gradient in (5.35) is approximated with Monte Carlo sampling in a batch size B_{size} (line 9). The model parameter ϑ is updated using the Adam optimizer (line 10). We compare the current policy with the greedy baseline policy and update the parameter $\vartheta^{(\text{BL})}$ only if the improvement is significant according to a paired t-test (5%) [27].

5.4 Numerical Results

We conduct extensive experiments to investigate the performance of the proposed TWA* algorithm in solving the problem of trajectory planning to minimize the total AoI for the

UAV-IoT network. The proposed model is implemented by Pytorch 1.7 and Python 3.8 and trained on a machine with 1 NVIDIA RTX 2080Ti GPU.

5.4.1 Test Settings

Decoding Strategies

As we mentioned in Section 5.3-D, the random sampling decoding and the greedy decoding are employed for training the model. At inference, we evaluate performance of all three decoding methods on test instances and they are marked as TWA*–greedy, TWA*–sampling ($W_{\text{sampling}} = 5120$), and TWA*–beam search ($W_{\text{beam}} = 100$).

Comparison Algorithms

To evaluate the effectiveness of the proposed model with different decoding methods, we compare it with the genetic algorithm [42], the simulated annealing (SA) algorithm [43], and Ptr-A* with the sampling strategy [5]. Common parameters are selected for the genetic algorithm: The population size is the number of all possible hovering points in one instance, the maximal iteration is 10000, crossover is 0.1, and mutation probability is 0.8. The parameters of SA for the initial temperature, cooling coefficient, and maximal iteration are taken as 100, 0.99, and 1000, respectively. Ptr-A* is a LSTM-based model, which processes the elements of a sequence one by one. Each element's hidden state is assumed to be dependent only on the previously hidden state. Hence, Ptr-A*'s structure makes it hard to use parallel computing to process sequences. However, TWA* is composed of a set of multi-head self-attention mechanism, which has the ability of parallel computation to process all elements of a sequence as a whole rather than one by one. Another benefit of TWA* is that it does not suffer from long dependency issues, which are very common in recurrent-based networks, such as LSTM. This is because transformer used in TWA* does not rely on the past hidden state to capture dependencies with previous elements.

Table 5.1 Simulation parameters

Parameter	Value	Parameter	Value
H	100 m	β	12.08
$\widetilde{\beta}$	0.11	P_{CH}	0.1 W
γ_{th}	20 dB (default)	ξ_{LoS}	1 dB
ξ_{NLoS}	20 dB	σ^2	-110 dBm
v_{UAV}	15 m/s	f_c	2 GHz
L_{data}	5 Mb	B_{width}	1 MHz
P_{com}	0.1 W	L_{sub}	5
P_0	99.66 W	P_1	120.16 W
U_{tip}	120 m/s	v_0	0.002 m/s
d_0	0.48	ρ	1.225 kg/m^3
τ	0.1 s	δ	0.5
s_0	0.0001		

Data Generation

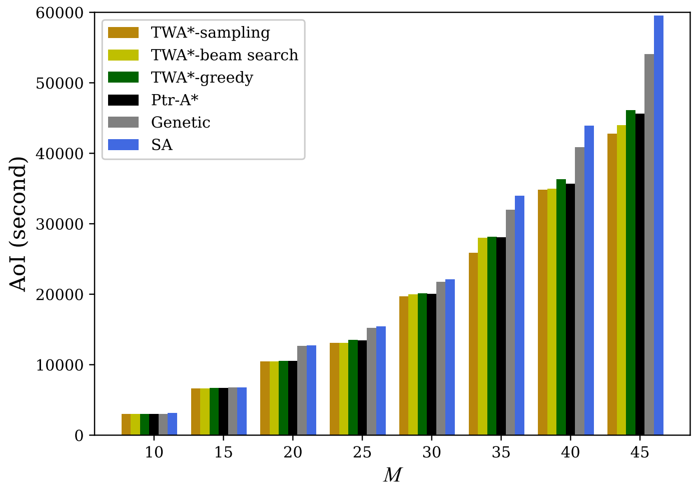
We assume there is a probability distribution over a family of problems. During training, problem instances are generated according to this distribution, and any test examples are also produced from the same distribution at inference. For any problem instances, all CHs $\{b_1, \ldots, b_M\}$ are randomly sampled from the distribution $\boldsymbol{U} = torch.FloatTensor(1,2).$ $uniform(0, 3000)$. With the SNR threshold γ_{th} and environment parameters, we can calculate the hovering disk O_m for each CH b_m, and each cluster of candidate hovering points G_m is sampled from O_m. The number of nodes N_m in each ground cluster is randomly chosen from $\{5, 10, 15, 20, 25, 30\}$. Hence, any of the problem instances is obtained as $\boldsymbol{H}^{(\text{in})} = (c_0; (G_1, b_1, N_1); \ldots; (G_m, b_m, N_m); \ldots; (G_M, b_M, N_M))$.

Environment Parameters and Hyperparameters

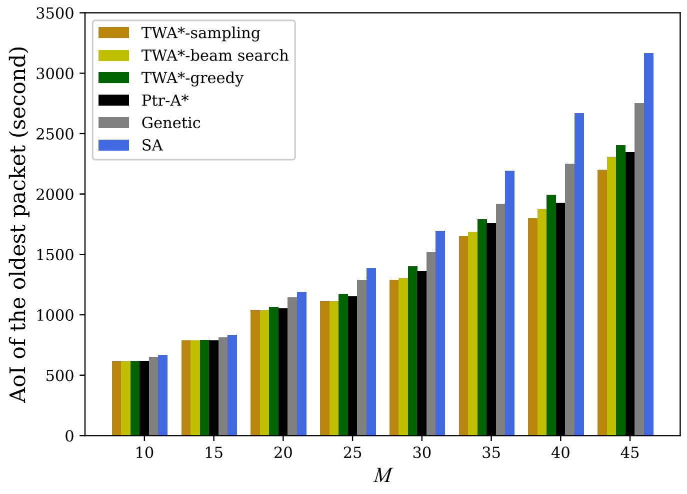
We consider a ground network with a size of 3 km $\times$ 3 km, and the start position of the UAV is located at $(0\,\mathrm{m}, 0\,\mathrm{m}, H\,\mathrm{m})$. Environment parameters are listed in Table 5.1. The embedding dimension d_{em} is equal to 512 and d_{v} is equal to 64. We train the proposed model using the Adam optimizer with a learning rate of 0.0001 on $E_{\mathrm{epochs}} = 200$ epochs, where each epoch includes $S = 1000$ training steps. At each training step, the batch size B_{size} is equal to 512, which means there are 512 instances in each batch. In each instance, we set $M = 10$.

5.4.2 Analysis of the Results

We first compare the total AoI between our proposed algorithm against the genetic and SA algorithms on the trained model when the value of M varies. Although the model is trained on 10-clusters IoT networks ($M = 10$), it still shows good performance on IoT networks with different sizes, like 20-clusters, 30-clusters, etc., as can be seen in Fig. 5.5 (a). Specifically, the TWA*-sampling algorithm always obtains the minimal total AoI when compared with other three decoding methods, as well as the two other algorithms under comparison. The TWA*-beam search and TWA*-greedy algorithms exhibit an obviously superior performance than the genetic and SA algorithms in reducing the total AoI. The above observations indicate that the proposed algorithm with the three different decoding methods achieves an excellent generalization ability with respect to the size of the IoT network used for training. When $M = 10$, TWA*-sampling, TWA*-beam search, TWA*-greedy, Ptr-A*, and the genetic algorithms obtain almost the same total AoI; however, the SA algorithm has a higher total AoI when compared to our proposed algorithm with all three decoding strategies. As the value of M increases, the performance gap increases gradually between our proposed algorithm and comparison algorithms. For instance, when $M = 25$, the total AoI values of the TWA*-sampling, TWA*-beam search, TWA*-greedy, Ptr-A*, genetic, and SA algorithms are 13134, 13134, 13546, 13431, 15205, and 15452 seconds, respectively. As M increases to 45, the total AoI values obtained by the TWA*-sampling, TWA*-beam search, and TWA*-greedy algorithms are 42803, 43971, and 46118 seconds, respectively. The total AoI value of Ptr-A is 45663 seconds. However, the total AoI values of the genetic and



(a) Comparison of the total AoI when M varies.



(b) Comparison of the oldest packet's AoI when M varies.

Figure 5.5 Comparison when M varies.

SA algorithms are 54061 and 59537 seconds, respectively, which are obviously inferior than what obtained by our proposed algorithm. In summary, our proposed algorithm using any of the three decoding methods can obtain better total AoI results than both the genetic and SA algorithms. In addition, TWA*-sampling always obtains better AoI values than Ptr-A* with the sampling strategy. This comparison result is consistent with the conclusions in [27] and [29] that transformer-based technique outperforms pointer network-based technique.

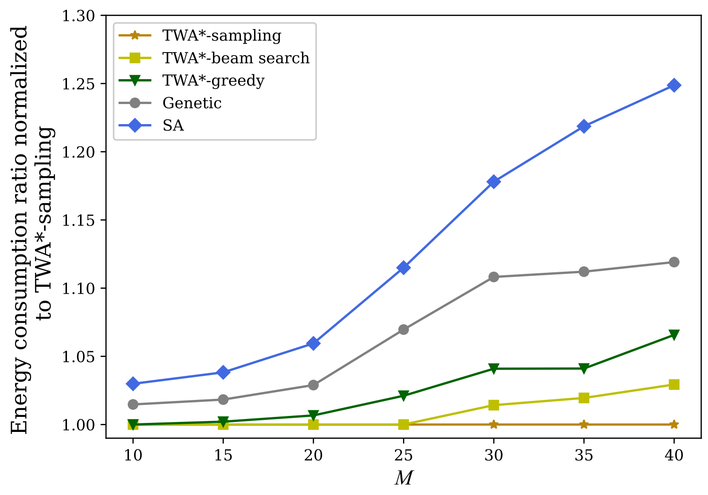


Figure 5.6 Comparison of energy consumption when M varies.

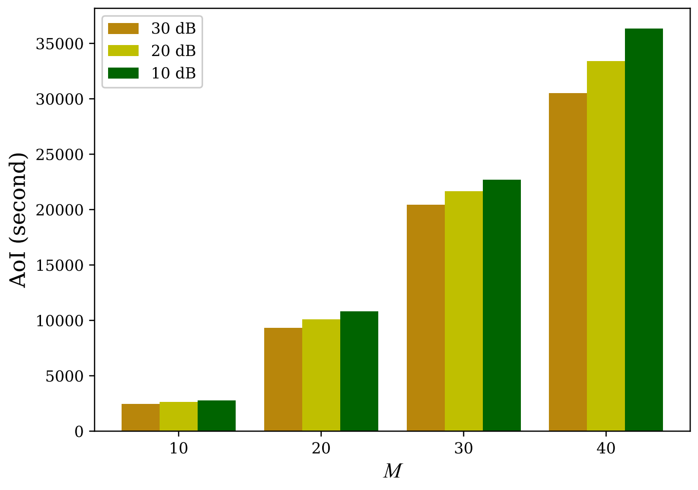
Table 5.2 Comparison of running time (second).

	Algorithm					
M	TWA*-sampling	TWA*-beam search	TWA*-greedy	Ptr-A*	Genetic	SA
10	1.9693	2.0653	1.9556	11.1556	47.57	5.5345
15	2.1412	2.2861	2.1055	19.3212	95.46	6.4262
20	2.3392	2.4900	2.3037	27.9023	163.41	6.8623
25	2.6006	2.8087	2.5778	36.9560	261.69	7.4619
30	2.8700	3.0876	2.8300	43.5498	392.97	8.3378
35	3.2018	3.4531	3.1536	52.4981	562.25	9.2576
40	3.8059	3.7506	3.5583	61.6301	749.16	9.6988
45	4.5112	4.5190	3.8995	75.8817	991.85	10.2019

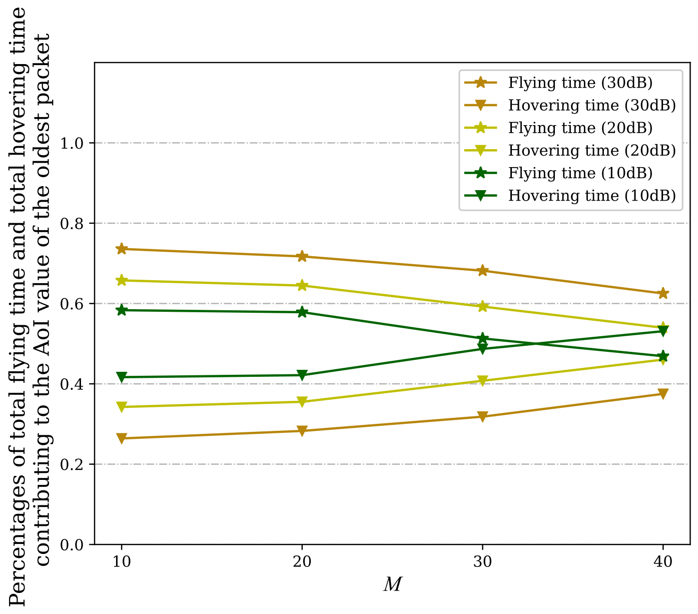
Next, we compare the AoI of the oldest packet which is from the node $b_{\pi(1)}^{(1)}$ that samples data first in the whole IoT network, among different algorithms. As can be seen in Fig. 5.5 (b), our proposed algorithm also exhibits a good performance in reducing the AoI of the oldest packet when compared with the genetic and SA algorithms. Furthermore, the TWA*-sampling algorithm obtains the best results among the three decoding methods and Ptr-A*.

Given that the proposed algorithm can find the best UAV trajectory with the minimal AoI in the UAV-IoT network among all the algorithms under comparison, it is of interest to further investigate the effective energy consumption of the UAV. It can be seen from (5.7) and (5.10) that the energy consumption of the UAV is related to its flying time and hovering time. The effective energy consumption is defined as the energy consumption of the UAV from the first visited hovering point to the end point, i.e., in completing its data collection task. Fig. 5.6 compares the effective energy consumptions for all the algorithms for different values of M. In particular, plotted in the figure are the average ratios of the effective energy consumptions by different algorithms with over that of the TWA*-sampling algorithm. As can be seen, our proposed algorithm with any of the three decoding methods has a better performance when compared to the genetic and SA algorithms, whereas the TWA*-sampling algorithm obtains the best result. The results in Fig. 5.6 are in line with expectations because with our proposed algorithm, the UAV spends less time to gather data than with the other two algorithms, which helps to reduce the effective energy consumption of the UAV.

Table 5.2 compares the running time at inference. As M increases, the running time of all the algorithms increases, which is well expected. We can see that among all the algorithms and for all the values of M, the running time of the TWA*-greedy algorithm is always shortest. Although TWA*-sampling obtains the best performance in reducing the total AoI as well as the AoI of the oldest packet (as can be seen from Fig. 5.5 (a) and Fig. 5.5 (b)), it has a longer running time than TWA*-greedy, which is reasonable. Similarly, TWA*-beam search takes slightly more time than TWA*-greedy because it needs more computational time to search for a better solution than TWA*-greedy. We can observe that the genetic algorithm takes the longest time among all the algorithms and its running time significantly increases as M increases. The running time of SA is acceptable in comparison with the genetic algorithm. Overall, the computational performance of our proposed model with all three decoding methods is significantly better than the SA and genetic algorithms. The running time of Ptr-A* is largely greater than that of TWA*-sampling, TWA*-beam search, and TWA*-greedy. This is because transformer-based techniques can process a sequence in



(a) Comparison of the total AoI for different values of γ_{th}.



(b) Percentages of the total flying time and the total hovering time for different values of γ_{th}.

Figure 5.7 Comparison for different values of γ_{th}.

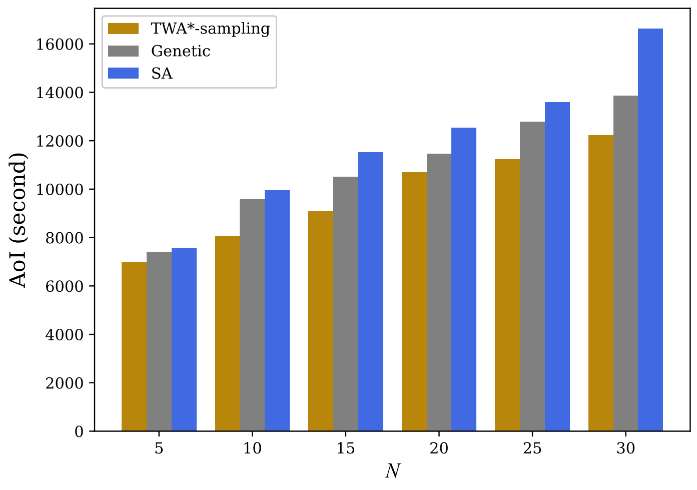
parallel. However, in Ptr-A*, the elements of a sequence must be processed one by one. Hence, our proposed TWA* with three decoding methods is faster than Ptr-A*.

In order to provide insights about the effect of γ_{th} on the total AoI, we set the same number of devices, namely $N_m = 20$, in each ground cluster and evaluate in Fig. 5.7 (a) the performance of TWA*-sampling for different values of γ_{th}. According to *Lemma 1* and (5.5), the smaller the value of γ_{th} is, the larger the area of each hovering disk O_m will be. This will affect the positions of hovering points and thus the total AoI. As we can see in Fig. 5.7 (a), for any given number of ground clusters, the total AoI gradually increases as the value of γ_{th} decreases. For example, when $M = 30$, the values of total AoI in 10 dB, 20 dB, and 30 dB are 22707, 21655, and 20457 seconds, respectively. We can also observe that the total AoI gap among three values of γ_{th} increases as γ_{th} becomes higher.

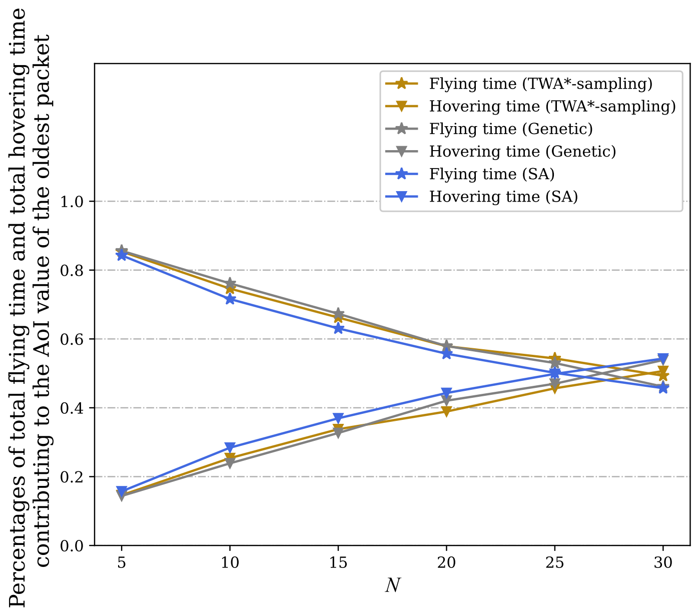
Next, we compare the total flying time and the total hovering time of the UAV that make up of the AoI value $A^{(1)}_{\pi(1)}$ of the oldest packet in networks with different number of clusters. Specifically these total flying time and total hovering time of the UAV are calculated as $\sum_{t=1}^{M} T^{(\text{fly})}_{(c_{\pi(t)},c_{\pi(t+1)})}$ and $\sum_{t=1}^{M} T^{(\text{hov})}_{c_{\pi(t)}}$, respectively. In each network, we also compare these portions of time when γ_{th} varies. Note that, the AoI values of the oldest packet are different for different γ_{th} values in a network. The percentages of the total hovering and the total flying time that contribute to the AoI value of the oldest packet are plotted in Fig. 5.7 (b). When $M = 10$, the total flying time is always higher than the total hovering time for any thresholds γ_{th}. In addition, as γ_{th} increases, the flying time portion increases. This is because the selected hovering point may be closer to the center of each hovering disk if the value of γ_{th} is large, which will cause the flight distance to be longer and thus increases the total flying time. When M increases, the UAV needs more time to collect data, and we can see that the hovering time portion slowly increases as expected.

Fig. 5.8 (a) compares the total AoI for different algorithms and for different numbers of devices in each cluster. We test the trained model on a 20-clusters instance with the same number of devices in each cluster. It can be seen that the proposed algorithm with the sampling decoding method always obtains the minimal values when compared to the genetic and SA algorithms. This clearly shows that our proposed algorithm can find a better trajectory in reducing the total AoI. As N increases, there is a large performance gap between TWA*-sampling and the two algorithms under comparison.

Fig. 5.8 (b) plots the percentages of the total flying time and the total hovering time that make up of the AoI value of the oldest packet in the 20-clusters network when N varies.



(a) Comparison of the total AoI when N varies



(b) Percentages of flying time and hovering time when N varies.

Figure 5.8 Comparison when N varies.

For all the algorithms considered in Fig. 5.8 (b), as N increases, the percentage of the total hovering time gradually increases. This trend is justified because the UAV needs more time to collect data from a larger number of ground nodes.

5.5 Conclusions

In this paper, we have investigated and solved the problem of AoI-oriented data collection in UAV-enabled cluster-based IoT networks. With the aim of minimizing the total AoI of the collected data, we formulated the trajectory optimization problem as the GTSP by jointly optimizing the selection of hovering points of the UAV and the visiting order to these hovering points. To solve the formulated problem, we designed a novel algorithm framework based on the state-of-the-art transformer. In particular, the formulated trajectory planning problem is viewed as a "translation problem". The whole UAV-IoT network serves as the "source language" to the proposed model and the "target language" of the model is the UAV's trajectory with the minimal total AoI, where the transformer is utilized to generate the visiting order and the weighted A* is used to quickly find the hovering points. The proposed model is trained by reinforcement learning to learn a trajectory planning policy. Comprehensive experiments were conducted to evaluate the performance of the proposed algorithm. The obtained simulation results showed that the learned policy by the proposed algorithm has a strong generalization ability. When compared with other algorithms, our proposed algorithm with three different decoding methods not only reduces the total AoI, but also reduces the AoI of the oldest packet and the effective energy consumption of the UAV. Moreover, our method also has lower computation time. In future, we plan to extend the system model and the proposed algorithm to the multiple UAVs-assisted IoT network.

Acknowledgement

This work was supported by an NSERC/Cisco Industrial Research Chair in Low-Power Wireless Access for Sensor Networks.

5.6 Appendix

By substituting (5.1)–(5.3) into (5.4), we can get the formulation (A.1) shown on top of the next page. For a fixed H, $20\log_{10}\left(4\pi f_c\sqrt{H^2+R^2_{(c_m,b_m)}}/c\right)$ is monotonically increasing with respect to $R_{(c_m,b_m)}$. As the analysis in [31] shows, $P^{(\text{LoS})}_{c_m}$ is monotonically increasing with respect to θ_{c_m}. Since $\theta_{c_m} = \arctan(H/R_{(c_m,b_m)})$, $P^{(\text{LoS})}_{c_m}$ is monotonically decreasing with respect to $R_{(c_m,b_m)}$ for a fixed H. Then, $(\xi_{\mathrm{LoS}} - \xi_{\mathrm{NLoS}}) \Big/ \left(1 + \beta \exp\left(-\widetilde{\beta}\,(\theta_{c_m} - \beta)\right)\right)$ is monotonically increasing with respect to $R_{(c_m,b_m)}$ because $\xi_{\mathrm{LoS}} < \xi_{\mathrm{NLoS}}$. Finally, we arrive at the conclusion that the left side of (A.1) is monotonically decreasing with respect to $R_{(c_m,b_m)}$ for a fixed H. When the SNR γ_{c_m} decreases to the threshold γ_{th}, we can obtain the maximum R^*. Hence, for any c_m, the UAV can successfully receive data from b_m if $R_{(c_m,b_m)} \leq R^*$.

$$\begin{aligned}
P_{\mathrm{CH}} \frac{1}{\sigma^2 \left(P_{c_m}^{(\mathrm{LoS})} L_{c_m}^{(\mathrm{LoS})} + \left(1 - P_{c_m}^{(\mathrm{LoS})}\right) L_{c_m}^{(\mathrm{NLoS})}\right)} &\geq \gamma_{\mathrm{th}} \\
P_{\mathrm{CH}} \frac{1}{\sigma^2 \left(20 \log_{10} \left(\frac{4\pi f_c d_{(c_m,b_m)}}{v_{\mathrm{light}}}\right) + P_{c_m}^{(\mathrm{LoS})} (\xi_{\mathrm{LoS}} - \xi_{\mathrm{NLoS}}) + \xi_{\mathrm{NLoS}}\right)} &\geq \gamma_{\mathrm{th}} \\
P_{\mathrm{CH}} \frac{1}{\sigma^2 \left(20 \log_{10} \left(\frac{4\pi f_c \sqrt{H^2 + R_{(c_m,b_m)}^2}}{v_{\mathrm{light}}}\right) + \frac{\xi_{\mathrm{LoS}} - \xi_{\mathrm{NLoS}}}{1 + \beta \exp\left(-\widetilde{\beta}(\theta_{c_m} - \beta)\right)} + \xi_{\mathrm{NLoS}}\right)} &\geq \gamma_{\mathrm{th}} \qquad (\mathrm{A.1})
\end{aligned}$$

References

[1] S. Hu, X. Chen, W. Ni, E. Hossain, and X. Wang, "Distributed machine learning for wireless communication networks: Techniques, architectures, and applications," *IEEE Communications Surveys & Tutorials*, vol. 23, pp. 1458–1493, Jun. 2021.

[2] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah, "A tutorial on UAVs for wireless networks: Applications, challenges, and open problems," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 2334–2360, Mar. 2019.

[3] S. Hu, W. Ni, X. Wang, A. Jamalipour, and D. Ta, "Joint optimization of trajectory, propulsion, and thrust powers for covert UAV-on-UAV video tracking and surveillance," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1959–1972, 2021.

[4] S. Hu, Q. Wu, and X. Wang, "Energy management and trajectory optimization for UAV-enabled legitimate monitoring systems," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 142–155, Jan. 2021.

[5] B. Zhu, E. Bedeer, H. H. Nguyen, R. Barton, and J. Henry, "UAV trajectory planning in wireless sensor networks for energy consumption minimization by deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 9540–9554, Sep. 2021.

[6] S. Zhang, H. Zhang, Z. Han, H. V. Poor, and L. Song, "Age of information in a cellular internet of UAVs: Sensing and communication trade-off design," *IEEE Transactions on Wireless Communications*, vol. 19, pp. 6578–6592, Oct. 2020.

[7] S. Kaul, R. Yates, and M. Gruteser, "Real-time status: How often should one update," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, pp. 2731–2735, Mar. 2012.

[8] H. Hu, K. Xiong, G. Qu, Q. Ni, P. Fan, and K. B. Letaief, "AoI-minimal trajectory

planning and data collection in UAV-assisted wireless powered IoT networks," *IEEE Internet of Things Journal*, vol. 8, pp. 1211–1223, Jan. 2021.

[9] J. Liu, X. Wang, B. Bai, and H. Dai, "Age-optimal trajectory planning for UAV-assisted data collection," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM)*, pp. 553–558, Apr. 2018.

[10] J. Liu, P. Tong, X. Wang, B. Bai, and H. Dai, "UAV-aided data collection for information freshness in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 20, pp. 2368–2382, Apr. 2021.

[11] Z. Jia, X. Qin, Z. Wang, and B. Liu, "Age-based path planning and data acquisition in UAV-assisted IoT networks," in *Proc. IEEE International Conference on Communications Workshops (ICC)*, pp. 1–6, May 2019.

[12] M. A. Abd-Elmagid and H. S. Dhillon, "Average peak age-of-information minimization in UAV-assisted IoT networks," *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 2003–2008, Feb. 2019.

[13] S. F. Abedin, M. S. Munir, N. H. Tran, Z. Han, and C. S. Hong, "Data freshness and energy-efficient UAV navigation optimization: A deep reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, pp. 5994–6006, Sep. 2021.

[14] M. Yi, X. Wang, J. Liu, Y. Zhang, and B. Bai, "Deep reinforcement learning for fresh data collection in UAV-assisted IoT networks," in *Proc. IEEE Conference on Computer Communications Workshops (INFOCOM)*, pp. 716–721, Jul. 2020.

[15] D. Ebrahimi, S. Sharafeddine, P. Ho, and C. Assi, "UAV-aided projection-based compressive data gathering in wireless sensor networks," *IEEE Internet of Things Journal*, vol. 6, pp. 1893–1905, Apr. 2019.

[16] Y. Yuan, L. Lei, T. X. Vu, S. Chatzinotas, S. Sun, and B. Ottersten, "Energy minimization in UAV-aided networks: Actor-critic learning for constrained scheduling op-

timization," *IEEE Transactions on Vehicular Technology*, vol. 70, pp. 5028–5042, May 2021.

[17] L. Shen, N. Wang, Z. Zhu, Y. Fan, X. Ji, and X. Mu, "UAV-enabled data collection for mMTC networks: AEM modeling and energy-efficient trajectory design," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2020.

[18] K. Obermeyer, "Path planning for a UAV performing reconnaissance of static ground targets in terrain," in *Proc. AIAA Guidance, Navigation, and Control Conference*, pp. 1–11, Mar. 2009.

[19] D. G. Macharet, A. A. Neto, V. F. da Camara Neto, and M. F. M. Campos, "Efficient target visiting path planning for multiple vehicles with bounded curvature," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3830–3836, Nov. 2013.

[20] J. T. Isaacs and J. P. Hespanha, "Dubins traveling salesman problem with neighborhoods: A graph-based approach," *Algorithms*, vol. 6, pp. 84–99, Feb. 2013.

[21] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem with neighborhoods," in *Proc. IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1555–1562, Jun. 2017.

[22] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon, and W. Saad, "Deep reinforcement learning for minimizing age-of-information in UAV-assisted networks," in *Proc. IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec. 2019.

[23] W. Li, L. Wang, and A. Fei, "Minimizing packet expiration loss with path planning in UAV-assisted data sensing," *IEEE Wireless Communications Letters*, vol. 8, pp. 1520–1523, Dec. 2019.

[24] A. Ferdowsi, M. A. Abd-Elmagid, W. Saad, and H. S. Dhillon, "Neural combinatorial deep reinforcement learning for age-optimal joint trajectory and scheduling design in UAV-assisted networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, pp. 1250–1265, May 2021.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008, Dec. 2017.

[26] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, “Learning heuristics for the TSP by policy gradient,” in *Proc. International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*, pp. 170–181, Jun. 2018.

[27] W. Kool, H. van Hoof, and M. Welling, “Attention, learn to solve routing problems!,” in *Proc. International Conference on Learning Representations (ICLR)*, pp. 1–25, May 2019.

[28] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, “Learning improvement heuristics for solving routing problems,” *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, Apr. 2021.

[29] X. Bresson and T. Laurent, “The transformer network for the traveling salesman problem,” *arXiv preprint arXiv:2103.03012*, 2021.

[30] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, “Update or wait: How to keep your data fresh,” *IEEE Transactions on Information Theory*, vol. 63, pp. 7492–7508, Nov. 2017.

[31] A. Al Hourani, S. Kandeepan, and S. Lardner, “Optimal LAP altitude for maximum coverage,” *IEEE Wireless Communications Letters*, vol. 3, pp. 569–572, Dec. 2014.

[32] J. Yao and N. Ansari, “Qos-aware power control in internet of drones for data collection service,” *IEEE Transactions on Vehicular Technology*, vol. 68, pp. 6649–6656, Jul. 2019.

[33] Y. Zeng, J. Xu, and R. Zhang, “Energy minimization for wireless communication with rotary-wing UAV,” *IEEE Transactions on Wireless Communications*, vol. 18, pp. 2329–2345, Apr. 2019.

[34] P. Tong, J. Liu, X. Wang, B. Bai, and H. Dai, "UAV-enabled age-optimal data collection in wireless sensor networks," in *Proc. IEEE International Conference on Communications Workshops (ICC)*, pp. 1–6, May 2019.

[35] A. Dumitrescu and J. S. Mitchell, "Approximation algorithms for TSP with neighborhoods in the plane," *Journal of Algorithms*, vol. 48, pp. 135–159, Aug. 2003.

[36] B. Yuan, M. Orlowska, and S. Sadiq, "On the optimal robot routing problem in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 1252–1261, Sep. 2007.

[37] Concorde. [Online]. Available: http://www.math.uwaterloo.ca/tsp/concor -de.html.

[38] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[39] R. Ebendt and R. Drechsler, "Weighted A search unifying view and application," *Artificial Intelligence*, vol. 173, pp. 1310–1342, Sept. 2009.

[40] G. Ramalingam and T. Reps, "On the computational complexity of dynamic graph problems," *Theoretical Computer Science*, vol. 158, pp. 233–277, May 1996.

[41] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[42] J. Yang, C. Wu, H. P. Lee, and Y. Liang, "Solving traveling salesman problems using generalized chromosome genetic algorithm," *Progress in Natural Science*, vol. 18, no. 7, pp. 887–892, 2008.

[43] S. Zhan, J. Lin, Z. Zhang, and Y. Zhong, "List-based simulated annealing algorithm for traveling salesman problem," *Computational Intelligence and Neuroscience*, vol. 2016, 2016.

6. Conclusions and Suggestions for Further Studies

6.1 Conclusions

This thesis proposed several algorithms to improve the efficiency of data collection in IoT networks.

In Chapter 2, we studied the problem of balancing the energy consumption of nodes in WSNs and proposed an improved soft k-means (IS-k-means) clustering algorithm. The initial centroids of the IS-k-means clustering algorithm were determined using CFSFDP and KDE. The mechanism of reassigning nodes was used to balance the number of nodes per cluster. In addition, the multi-CHs scheme was utilized to balance traffic load of CHs of different clusters and reduce the frequency of clustering. Simulation results demonstrated that the proposed IS-k-means algorithm postponed the FND by 2.7 times, 34 times, 2.3 times, 2.4 times, 30 times, and 1.5 times when compared to LEACH, VLEACH, KM-LEACH, EECPK-means, k-means, and EB-CRP respectively in a network with the size of 100 m $\times$ 100 m. The HND of the IS-k-means algorithm was delayed by 2 times when compared to LEACH, k-means, and KM-LEACH. In addition, the IS-k-means algorithm achieved an excellent result in postponing the FND and HND in the network with a size of 200 m $\times$ 200 m when compared to other algorithms. Hence, the proposed IS-k-means algorithm outperformed all the algorithms under comparison in balancing energy consumption in WSNs.

In Chapter 3, the problem of jointly designing the UAV's trajectory and selecting CHs for an IoT network was investigated to minimize the total energy consumption in the UAV-

IoT system. To solve the formulated problem, a novel Seq2Seq-based DRL method was proposed. Simulation results demonstrated that our proposed DRL method outperformed the ACO and greedy algorithms in planning the UAV's trajectory and achieved almost the same performance as the Gurobi optimizer. However, the computation time of our DRL algorithm was significantly less than that of Gurobi optimizer. Although the proposed DRL algorithm was trained on 4-clusters problems, it could plan near-optimal trajectories on 5-clusters, 6-clusters, 7-clusters, 8-clusters, 9-clusters, and 10-clusters problems. This showed that our proposed DRL algorithm had excellent abilities of generalization, scalability, and automation to solve different problem instances with different numbers of clusters, without the need to retrain the model for new problems.

In Chapter 4, we continued to investigate the total energy consumption minimization problem in an UAV-aided network. We designed a DRL model to produce the trajectory of the UAV where the pointer network was used to determine the visiting order to clusters and the A* search algorithm was utilized to find the CH for each cluster. At inference, greedy search, sampling search, and active search were used for improving the performance of the proposed model. On small-scale clusters problems, the trained 20-clusters model by the proposed DRL algorithm with three search strategies produced better trajectories for the UAV when compared with the genetic and NN algorithms. In particular, active search always obtained the best results. For example, when the number of clusters was 30, the total energy consumptions produced by NN and genetic algorithms were almost equal, which was about 11% more than that of DRL-active-10240, 8% more than that of DRL-sampling, and 7% more than that of DRL-greedy. When the number of clusters increased to 50, the energy consumption of NN was around 21% more than that of DRL-active-10240, 13% more than that of DRL-sampling, and 8% more than that of DRL-greedy. Likewise for 50 clusters, the energy consumption of the genetic algorithm was around 33% more than that of DRL-active-10240, 24% more than that of DRL-sampling, and 17% more than that of DRL-greedy. For large-scale clusters problems, the 40-clusters model trained by our proposed DRL algorithm still outperformed two algorithms under comparison. For instance, when the number of clusters was 80, the genetic algorithm consumed 50%, 58%, and 76% more energy when compared to DRL-greedy, DRL-sampling, and DRL-active-10240, respectively. As the number of clusters increased to 100, the total energy consumption of the UAV-WSN when using the genetic algorithm was 72% more than that of DRL-greedy, 82% more than that of DRL-sampling, and 103% more than that of DRL-active-10240. NN also showed a similar trend to that of the genetic algorithm. In particular, its energy consumption was 47% more than that of DRL-active-10240 in the 80-clusters problem and increased to 55% more than the energy consumption of DRL-active-10240 in the 100-clusters problem. In addition, the trained 20-clusters model and 40-clusters model showed an excellent generalization ability on small-scale clusters problems and large-scale clusters problems, respectively.

In Chapter 5, the problem of minimizing the total AoI of the collected data in UAV-enabled cluster-based IoT networks was investigated by jointly selecting hovering points of the UAV and designing the visiting order to these hovering points. The formulated optimization planning problem was viewed as a "translation problem", and a novel solution based on the state-of-the-art transformer and the weighted A* was designed, called TWA*. The whole UAV-IoT network served as the "source language" to the proposed algorithm and the "target language" of the alorithm was the UAV's trajectory with the minimal total AoI. Greedy, random sampling, and beam search decoding methods were used in the proposed algorithm. Simulation results demonstrated that the TWA*-sampling algorithm always obtained the minimal total AoI when compared with other two decoding methods, as well as the genetic and SA algorithms. The TWA*-beam search and TWA*-greedy algorithms exhibited an obviously superior performance than the genetic and SA algorithms in reducing the total AoI. For instance, when the number of clusters was 20, the total AoI values of the TWA*-sampling, TWA*-beam search, TWA*-greedy, genetic, and SA algorithms were 10503, 10503, 10543, 12666, and 15433 seconds, respectively. As the number of clusters increased to 40, the total AoI values obtained by the TWA*-sampling, TWA*-beam search, and TWA*-greedy algorithms were 35003, 35163, and 36318 seconds, respectively, which were very close to each other. However, the total AoI values of the genetic and SA algorithms were 40861 and 43937 seconds, respectively, which were clearly inferior than what obtained by our proposed algorithm. Furthermore, although the model was trained on

10-clusters IoT networks, it still showed good performance on IoT networks with different sizes such as 20-clusters and 30-clusters.

6.2 Suggestions for Further Studies

The main goal of this thesis was to develop novel techniques to achieve energy-efficient and fresh data collection in IoT networks. While conducting our research, several issues came up that would be worthwhile for further studies. These issues are elaborated below.

In Chapter 2, the improved IS-k-means clustering algorithm is proposed to balance the energy consumption of nodes in WSNs. The CH of each cluster directly communicates with BS to transmit data, which may cause longer communication distance between the CH and BS, and thus, increase the transmission energy consumption. Hence, designing an energy-efficient multi-hop routing algorithm to extend the applicability of the IS-k-means algorithm to large-scale WSNs is necessary.

In Chapter 3 and Chapter 4, we investigated the problem of minimizing energy consumption in the UAV-IoT network where one UAV is used to collect data. Machine learning techniques were developed to solve the formulated energy problems. Multi-UAVs-IoT network is worthwhile to study for improving the efficiency of data collection in more complicated UAVs-IoT network. It would also be interesting to develop distributed machine learning techniques to solve multiple UAVs' trajectory planning problem.

In Chapter 5, an AoI-oriented data collection model in a clusters-based IoT network is proposed. Then, we formulated a total AoI-minimal trajectory planning problem where the hovering points of the UAV and the visiting order to these points are jointly optimized. The state-of-the-art transformer was employed to solve the formulated problem. It will be an interesting study to jointly design the UAV's trajectory and cluster ground devices to minimize the total AoI in a general (unclustered) IoT network.