Journal of King Saud University - Computer and Information Sciences 36 (2024) 101894

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University - Computer and Information Sciences

journal homepage: www.sciencedirect.com

Full length article

A Q-learning-based smart clustering routing method in flying Ad Hoc networks

Mehdi Hosseinzadeh [a,b,1], Jawad Tanveer [c,1], Amir Masoud Rahmani [d], Khursheed Aurangzeb [e], Efat Yousefpoor [f], Mohammad Sadegh Yousefpoor [f], Aso Darwesh [g], Sang-Woong Lee [h,*], Mahmood Fazlali [i,*]

[a] Institute of Research and Development, Duy Tan University, Da Nang, Viet Nam
[b] School of Medicine and Pharmacy, Duy Tan University, Da Nang, Viet Nam
[c] Department of Computer Science and Engineering, Sejong University, Seoul 05006, Republic of Korea
[d] Future Technology Research Center, National Yunlin University of Science and Technology, Yunlin, Taiwan
[e] Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia
[f] Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
[g] Department of Information Technology, University of Human Development, Sulaymaniyah, Iraq
[h] Pattern Recognition and Machine Learning Lab, Gachon University, 1342 Seongnamdaero, Sujeonggu, Seongnam 13120, Republic of Korea
[i] Cybersecurity and Computing Systems Research Group, School of Physics, Engineering and Computer Science, University of Hertfordshire, AL10 9AB Hertfordshire, UK

ARTICLE INFO

Keywords:
Flying ad hoc networks (FANETs)
Clustering
Unmanned aerial vehicles (UAVs)
Reinforcement learning (RL)
Machine learning (ML)

ABSTRACT

Flying ad hoc networks (FANETs) have particular importance in various military and civilian applications due to their specific features, including frequent topological changes, the movement of drones in a three-dimensional space, and their restricted energy. These features have created challenges for designing cluster-based routing protocols. In this paper, a Q-learning-based smart clustering routing method (QSCR) is suggested in FANETs. In QSCR, each node discovers its neighbors through the periodic exchange of hello messages. The hello time interval is different in each cluster, and cluster leaders determine this interval based on the average speed similarity. Next, an adaptive clustering process is presented for categorizing drones in the clusters. In this step, the cluster leader is selected based on a new parameter called merit value, which includes residual energy, centrality, neighbor degree, speed similarity, and link validity time. Then, a centralized Q-learning model is presented to tune weight coefficients related to merit parameters dynamically. In the last step, the routing process is done using a greedy forwarding technique. Finally, QSCR is run on NS2, and the simulation results of QSCR are compared with those of ICRA, WCA, and DCA. These results show that QSCR carries out the clustering process rapidly but has less cluster stability than ICRA. QSCR gets energy efficiency and improves network lifetime. In the routing process, QSCR has a high packet delivery rate compared to other schemes. Also, the number of isolated clusters created in QSCR is less than other clustering methods. However, the proposed scheme has a higher end-to-end delay than ICRA. Also, this scheme experiences more communication overhead than ICRA slightly.

1. Introduction

Recently, unmanned aerial vehicles (UAVs) have been successfully used in military and civilian areas for different applications, including aerial photos, 5G technology, wildfire monitoring, road traffic monitoring, intelligent UAV-assisted agriculture, UAV-assisted connections, and search and rescue (Alam and Moh, 2022; Alam et al., 2022). In these

* Corresponding authors.
E-mail addresses: slee@gachon.ac.kr (S.-W. Lee), m.fazlali@herts.ac.uk (M. Fazlali).
[1] These authors contributed equally to this work.
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.101894
Received 1 November 2023; Received in revised form 14 December 2023; Accepted 15 December 2023
Available online 8 January 2024

applications, reliable and fast wireless communication between UAVs must be formed using a network. This is a motivation to carry out drone missions in the form of a flying ad hoc network (FANET) (Hosseinzadeh et al., 2023c; Yousefpoor et al., 2021). In this network, drones cooperate with each other, and consequently, FANET is more effective and efficient than a single-UAV system in terms of cost development, scalability, survivability, and efficiency (Messaoudi et al., 2023; Hosseinzadeh et al., 2023d). In addition, FANET includes distinct features that make it different from other ad hoc networks such as mobile ad hoc networks (MANETs) or vehicular ad hoc networks (VANETs). Some of these features are highly moving UAVs, very dynamic topology, sparse network, and limited energy capacity (Darabkh et al., 2022; Rahmani et al., 2022b). It is essential to pay attention to these specific features when creating routing protocols in these networks. For this reason, many routing protocols proposed for MANET and VANET cannot be used directly in FANET (Lee et al., 2021; Lansky et al., 2023). Generally, routing protocols presented in FANET can be categorized into two groups: single-path routing and multi-path routing (Almeida et al., 2021; Rovira-Sugranes et al., 2021). Single-path routing protocols consider a static routing table, which contains routing paths calculated before network operation begins. These routes are unchanged at all times. Multi-path routing protocols guide data packets step-by-step toward the destination (Zhang et al., 2022b; Yang et al., 2022). A key subject in the route discovery process of these routing protocols is how to single out the next-hop node. In addition, routing protocols can be grouped into two categories: topology-based routing and position-based routing. The first group includes three classes: proactive, reactive, and hybrid (Liu et al., 2023; Jin et al., 2023).

To transfer data packets between UAVs in a high-dense FANET, researchers have proposed cluster-based routing protocols. In these methods, FANET includes a hierarchical structure in which UAVs act as cluster heads (CHs) or cluster members (CMs) (Dhall and Dhongdi, 2023; Hosseinzadeh et al., 2023e). The selection of cluster heads (also called cluster leaders) is a major challenge in the clustering process. This role is determined based on different factors such as residual energy, centrality, neighbor degree, and so on (Arafat and Moh, 2019; Abdulhae et al., 2022). Note that some clustering techniques are single-criterion methods, meaning that, cluster leaders are chosen only based on one criterion such as energy, while other clustering techniques are multi-criteria methods, meaning that, they consider different criteria to choose cluster leaders (Hosseinzadeh et al., 2023a; Mansoor et al., 2023). Multi-criteria clustering methods allocate a specific weight to each clustering criterion, and the weighted sum of these clustering criteria is considered as the final score for determining cluster leaders (Gharib et al., 2022; Khedr et al., 2023). In the clustering process, an important challenge is how to choose a suitable weight coefficient for each clustering criterion. Most studies only emphasize that the sum of these weight coefficients should be equal to one (Khan et al., 2019, 2020). Most research studies consider the same value for these weight coefficients, while some protocols also determine different weights for each clustering criterion. In a clustered network, CMs are responsible for transmitting data collected from the environment to its cluster leader (Jawhar et al., 2017; Shahzadi et al., 2021). In addition, cluster leaders are responsible for managing intra-cluster communication and intra-cluster data aggregation. They act as relay nodes and form inter-cluster communication. Cluster leaders are also responsible for communicating with the ground control station (GCS) (Alzahrani et al., 2020; Hadi et al., 2023). Due to the dynamic nature of UAVs in FANET, intra or inter-cluster communication links have a very short lifetime, and the stability of clusters is very weak. These affect the reliability of the data transmission process and, consequently, cause routing inefficiency and weaken the quality of services (QoS) in FANET (Fanian and Rafsanjani, 2019; Cheng et al., 2023).

Today, reinforcement learning (RL) is widely applied for improving wireless communication paths in ad hoc networks. In RL, agents execute various actions in the dynamic environment such as FANET, and utilize previous experiences to make more smart decisions and increase reward (Khan et al., 2022; Lansky et al., 2022a; Hosseinzadeh et al., 2023b). RL is employed for many applied scenarios such as forecasting network topology, estimating communication channels, optimizing flight trajectory, and designing routing. A well-known reinforcement learning technique is Q-learning (QL) whose features are off-policy, model-free, and value-based (Rahmani et al., 2022c; Lansky et al., 2022b). It can be used for multi-objective optimization goals in FANET. QL examines the expected cumulative reward and adopts an optimal policy according to the obtained experiences. In addition, QL is a successful solution to choose an optimal path in the data transfer process to the destination (Rahmani et al., 2022a; Wang et al., 2022).

In this paper, a Q-learning-based smart clustering routing method (QSCR) is suggested for flying ad hoc networks. QSCR includes four phases: dynamic neighbor discovery, adaptive clustering, dynamic and smart adjustment of clustering parameters, and greedy routing. In QSCR, the clustering phase is responsible for constructing network topology, and the greedy routing process is responsible for creating paths. In the clustering process, cluster leaders are chosen based on the weighted sum of five merit parameters, including residual energy, centrality, neighbor degree, speed similarity, and link validity time. The weight coefficient related to each merit parameter is tuned up based on a centralized QL-based learning model so that these weight coefficients change according to the conditions of UAVs in the network, and consequently, the importance of each parameter will be different accordingly. QSCR adjusts Q-learning parameters to be more consistent with the dynamic topology of FANET. In general, the main innovations in this paper are as follows:

- In QSCR, the neighbor discovery process focuses on the content of each hello packet and its propagation time interval. Each cluster leader is responsible for specifying the hello propagation time interval in its cluster based on the average similarity between its velocity vector and cluster members' velocity vector.
- In QSCR, an adaptive clustering process is proposed for FANET to distribute energy consumption in the network uniformly and increase network longevity. At this phase, cluster leaders are chosen based on a new parameter called merit, which includes five parameters, namely residual energy, centrality, neighbor degree, speed similarity, and link validity time.
- In QSCR, a QL-based learning model is presented to tune weight coefficients related to the merit parameters and determine their effect on the merit value in a dynamic manner and according to FANET conditions. In this learning process, the reward function is also calculated based on three metrics, namely the balance of energy consumption, the number of isolated clusters, and the distribution of cluster leaders in the network.
- In QSCR, learning parameters are chosen adaptively. QSCR calculates the learning rate based on the stability of clusters. In addition, the discount coefficient is calculated based on the similarity between the velocity vectors of cluster members and the cluster leader.
- In QSCR, the inter-cluster routing process follows a greedy routing technique so that the cluster leader examines the positions of its adjacent clusters and selects the closest cluster leader to the destination as its next-hop node.

The structure of this paper includes the following sections: Section 2 introduces some related works in flying ad hoc networks. Section 3 provides the concepts of reinforcement learning, especially Q-learning, because of its use in QSCR. Section 4 expresses the assumptions of the network model. Section 5 describes the details of the proposed method. In Section 6, the simulation and evaluation results of the proposed method are compared to other schemes. Finally, this paper is concluded in Section 7.

2. Related works

In Ergenç et al. (2019), a dependability-based clustering algorithm (DCA) is offered in MANETs. It can properly deal with the challenge that originates from the lack of a central administrator and centralized infrastructure in MANETs. This scheme makes a clustered topology, which has high scalability and reliability. DCA pays attention to the dependency of clusters along with individual nodes to stabilize the clustered network structure. In addition, DCA analyzes various criteria and introduces a complete optimization structure to choose CHs and calculate the dependency of clusters in a weighted clustering algorithm. In DCA, the weight coefficient related to each clustering metric is computed by the moment-independent Delta analysis technique. DCA makes a flexible clustering structure, which is stable, energy-efficient, and provides quality of services (QoS) in the network. Each node employs the dependency related to each cluster for joining a suitable cluster. Simulation results indicate the superiority of DCA compared to other approaches for fixed and dynamic scenarios.

In Chatterjee et al. (2002), an on-demand decentralized clustering scheme called WCA is suggested in MANET. In these dynamic networks, nodes are mobile in nature. Consequently, frequent connections and disconnections of nodes to/from the clusters threaten network stability, so the system needs to be configured again. In WCA, CHs create a dominant set to guarantee network scalability and stability. WCA takes into consideration several metrics, including ideal degree, movement of nodes, residual energy, and transmission power, to decide on cluster heads. It benefits from a weighted approach to choose CHs based on their connection and energy levels. In WCA, the number of nodes around a cluster must be more than a threshold value to improve the performance of the media access control (MAC) protocol. WCA employs a non-periodic CH selection process to lower computational and communication costs. CHs are responsible for making inter-cluster communications in the network. The experimental results show that WCA works better than other approaches.

In Guo et al. (2022), an intelligent clustering routing approach (ICRA) is presented for FANETs. ICRA constitutes three processes, namely clustering procedure, clustering adjustment procedure, and routing procedure. The first procedure calculates the fitness value of each node to determine its state, namely CH or CM. Furthermore, to extend topology stability and network lifespan in diverse states, reinforcement learning assists the clustering adjustment procedure to learn constantly the network environment and obtain the best strategy by taking different actions to calculate the fitness of UAVs in each network state. In accordance with this knowledge, the clustering adjustment procedure aids the clustering technique to be compatible with the current network state. In addition, in the last procedure, ICRA employs a few gateways in each cluster to facilitate the inter-cluster routing procedure. This shortens end-to-end delay and increases the packet delivery ratio. The simulation results emphasize that ICRA is better than other clustering methods.

In Lansky et al. (2022c), a Q-learning-based routing approach called QFAN to control weather conditions using flying ad hoc networks. In QFAN, there are two phases, namely the discovery and the maintenance of paths created in the network. The first phase constitutes a Q-learning-aided routing model. In this learning model, the state space is restricted by calculating a filtering parameter, which removes some UAVs from this space. The second phase repairs the paths, which may fail in the near future. Evaluation results confirm that QFAN has a superior performance than other routing approaches. However, overhead in QFAN is slightly high.

In Zhang et al. (2022a), a three-dimensional Q-learning-based routing (3DQ) approach is offered to provide QoS requirements, especially PDR, in flying ad hoc networks. 3DQ includes a learning model, which benefits from Q-learning to make routing decisions on the networks. Two main parts of 3DQ are link-state prediction and path construction. The first part, link-state prediction, enables UAVs to forecast the status of links connected to their neighbors with regard to their 3D movement and transmit data packets to the destination. Each UAV makes its routing decisions in accordance with the path construction part in 3DQ. The evaluation results show that 3DQ is suitable for FANET and is more successful than other routing schemes.

In Alam and Moh (2023), a Q-learning routing scheme based on adaptive flocking control (QRIFC) in FANETs. This scheme provides an adaptive flocking control system to manage the optimized movement of UAVs with regard to the distance traveled by each UAV. Hence, this system can handle the density of UAVs in FANET. Furthermore, it relies on the information of two-hop neighbors to calculate the upper and lower boundaries of distance between UAVs in the network. This causes a balance between area coverage and network connectivity, increases the stability time of communication links between neighboring nodes, and decreases control overhead in the network. A multi-objective Q-learning model is designed to carry out the exploration and exploitation operations for finding the best path in terms of delay, stability, and energy efficiency. The evaluation results express the superiority of QRIFC compared to other methods.

In Cui et al. (2022), a topology-aware and flexible Q-learning routing algorithm (TARRAQ) is presented in FANETs. It tracks topology changes, controls overhead, and chooses routing paths in an independent and distributed manner. TARRAQ checks the behavior of UAVs using queue theory and gets two scales, namely closed-form solutions of neighbor change rate (NCR) and neighbors' change inter-arrival time (NCIT). According to NCR and NCIT, a flexible evaluation time interval is calculated in accordance with a new metric called the expected perception delay of events that occur in the network. In addition, a Q-learning-based routing model is suggested to find different paths between UAVs in a distributed, independent, and adaptive manner. The expected perception delay obtained in this scheme can lower communication costs when updating the action set. The evaluations performed on this scheme confirm the high accuracy and successful performance of TARRAQ in FANET.

In Arafat and Moh (2021), a Q-learning-based topology-aware routing (QTAR) protocol for FANET. It uses the information of two-hop neighbors to get local network topology and create reliable routes between UAVs in the network. To find the most suitable paths in the network, QTAR employs the information of two-hop neighbors and considers several metrics such as residual energy, location, speed, and delay. QTAR extends its knowledge about local network topology due to the use of two-hop neighbor information and, consequently, makes optimal routing decisions in the network. In addition, these routing decisions are made based on a Q-learning system to manage topological changes adaptively. The evaluation results confirm the efficiency of QTAR compared to other routing schemes.

Table 1 expresses the most important strengths and weaknesses of scheme mentioned in this section.

3. Basic concept

Reinforcement learning (RL) belongs to the family of machine learning (ML) techniques in artificial intelligence (AI). This science can be explained using a simple example. Assume that a person did not test a particular food previously. Now, he (she) wants to test this food for the first time. As a result, this person detects whether this food has a good taste or not. Accordingly, he (she) acquires knowledge about various foods, and this acquired knowledge can be applied to decide whether this person will again eat this food or not in the future (da Costa et al., 2021; Adams et al., 2022). In computer science, RL is used in algorithms, which need knowledge about their task. These algorithms utilize RL to make the best decision for doing this task. Furthermore, RL can be used in FANETs to improve and enhance applications, for example, resource management, channel modeling, localization, network security aspects, and routing. RL is made up of various elements, namely agent, learning environment, action, state,

Table 1
Comparison of related works.

Approach	Strengths	Weaknesses
DCA (Ergenç et al., 2019)	Making reliable communication links between nodes, improving energy efficiency, getting high scalability, prioritizing nodes in the CH selection process, reducing communication overhead	High time complexity, need to high time for creating clusters due to high computational complexity, lack of adaptability to the dynamic environment of FANET
WCA (Chatterjee et al., 2002)	Increasing energy efficiency, scalability, high fault tolerance	High communication overhead due to the exchange of control messages between CHs, lack of adaptation to the very dynamic topology of FANET because of its high topology changes
ICRA (Guo et al., 2022)	Improvement of energy consumed by nodes in the network, enhancing the quality of services, growing throughput in the clustering procedure, stabilizing network topology, shortening end-to-end delay, rising packet delivery rate, lowering routing overhead	Use of fixed learning parameters in the proposed Q-learning model, not calculating a propagation time interval for disseminating hello messages in the network
QFAN (Lansky et al., 2022c)	Filtering the search space, converging to the optimal solution rapidly, shortening delay, increasing the data transfer rate, improving network scalability	Considering the fixed values for Q-learning parameters, namely learning rate and discount factor, increasing communication overhead
3DQ (Zhang et al., 2022a)	Suitable throughput, improving QoS requirements, including delay and PDR in the network, avoiding congestion in the routing process, preventing routing holes	Considering two fixed values for learning rate and discount factor in Q-learning, ignoring the energy consumed by UAVs, low scalability
QRIFC (Alam and Moh, 2023)	Decreasing delay in the network, creating stable routes, getting energy efficiency, attention to learning parameters and updating them constantly, improving throughput, PDR, and reliability	High time complexity, the possibility of falling the adaptive flocking control algorithm in a local optimum
TARRAQ (Cui et al., 2022)	Improving data transmission rate, minimizing energy consumption, increasing reliability and throughput, adaptability to topological changes in FANET, calculating the hello propagation time interval adaptively	Low scalability, converging to optimal solution slowly because of having a large Q-table, high time and computational complexities
QTAR (Arafat and Moh, 2021)	Adaptability to the dynamic environment of FANET, high scalability, decreasing delay in the routing process, increasing throughput and PDR in the network, tuning learning parameters (α and γ) adaptively	High time complexity, having a large Q-table, and converging to the optimal solution slowly

and reward. Continuous interaction with the environment causes the agent to learn its best behavior in a learning environment. During the learning process, the agents acquire diverse experiences by experiencing different scenarios on the environment. Each scenario is called a state. However, in each state, the agent can single out an action from a set of permissible actions. The selected action affects the environment and produces a result. Accordingly, the agent gets a reward or penalty from the environment. Over time, the agent tries to increase the rewards gotten from the environment, and consequently, it will learn its best behavior in each state (Prudencio et al., 2023; Elallid et al., 2022).

Q-learning is a subset of RL, which uses a new concept called Q-value to optimize the agent's behavior using an iterative manner. This algorithm allocates Q-values (i.e. $Q(S_t, A_t)$) to different states and actions. $Q(S_t, A_t)$ estimates how desirable it is to do the action A_t in the state S_t. According to Q-learning, the agent is in an initial state. Then, it is transferred from the current state to the next state. Each transition occurs because of an action performed by the agent when interacting with the environment. Therefore, at each step, the agent applies its selected action to the environment and gets a reward from it. Then, it goes to another state. This process continues until the agent reaches the goal. In this situation, an episode is completed. In QL, $Q(S_t, A_t)$ is estimated based on Eq. (1) to calculate Q-values when interacting the agent with the environment (Ganesh and Xu, 2022; Sutton and Barto, 2018).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_A Q(S_{t+1}, A) - Q(S_t, A_t) \right] \quad (1)$$

S_t and S_{t+1} show the current and next states of the agent, respectively. A_t indicates the present action selected based on a particular policy. A' is the best action selected based on the current Q-value. R_{t+1} expresses the reward obtained from the environment when performing the current action. $\max_A Q(S_{t+1}, A)$ indicates the maximum Q-value under the next state. In the following, some learning parameters related to QL are described briefly:

- **Discount factor** (γ): γ is a coefficient limited to the interval $[0, 1]$. Q-learning algorithms often assume that the value of future rewards is less than that of current rewards. As a result, Q-learning function discounts them.
- **Learning rate** (α): α shows the knowledge acquired from the environment for updating $Q(S_t, A_t)$.
- ϵ-**Greedy policy:** It is a simple solution to choose actions by considering the estimations of Q-values. Accordingly, the next action is chosen based on the highest Q-value by considering the probability $(1 - \epsilon)$ or it is selected randomly based on the probability ϵ.

4. Network model

In QSCR, the flying ad hoc network includes a number of flying nodes and a ground station control (GCS). In this scheme, each flying node is displayed as U_i where $i = 1, 2, \ldots, n$. U_i has a direct communication with flying nodes available in its communication range (r_i) while it has created an indirect or multi-hop communication with other flying nodes that are out of r_i. In QSCR, U_i has access to a positing system and can obtain its position and speed at any moment. In this network, UAVs are connected to each other through UAV-to-UAV communication (U2U) in the air. However, UAVs and GCS are connected to each other through UAV-to-GCS communication (U2G) or GCS-to-UAV communication (G2U). These UAVs are categorized into m groups using a dynamic and intelligent clustering method. Each group is called cluster (C_k) so that $k = 1, 2, \ldots, m$. C_k includes a cluster leader (LU_k) and a number of cluster member UAVs (MU_j^k so that $j = 1, 2, \ldots, N_k$). In each C_k, some cluster members close to the cluster border also play the role of inter-cluster gateways (GU_g^k) and are used to connect with adjacent clusters. This network model is shown in Fig. 1. In the following, the task of each node is explained in the network.

- **GCS:** It benefits from an unlimited and stable energy source. GCS is responsible for monitoring the network, controlling flight routes, determining the mission of flying nodes, and transmitting

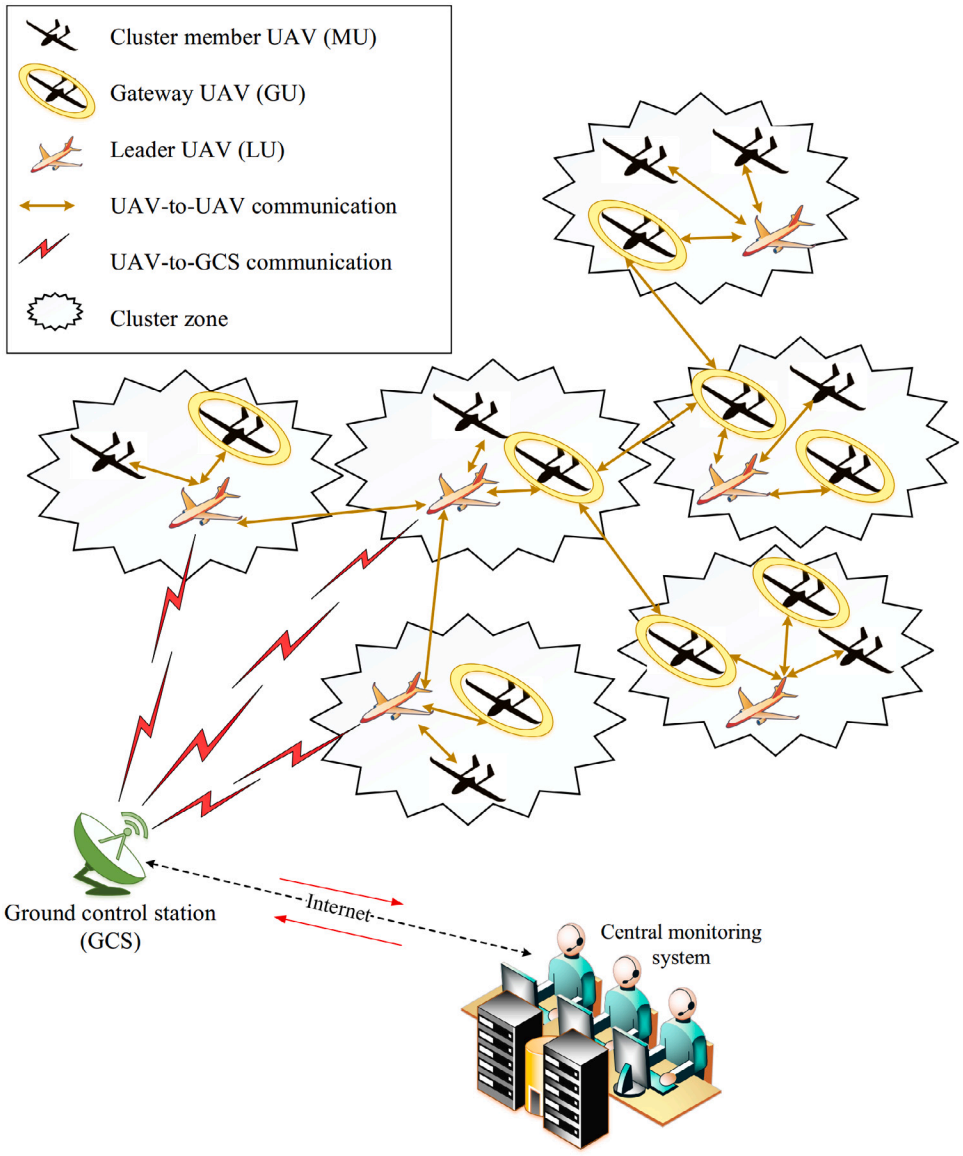

Fig. 1. Network model in QSCR.

various commands to UAVs. In QSCR, the most important task of GCS is to tune clustering parameters using a Q-learning algorithm intelligently and dynamically.

- **LU_k:** In QSCR, the role of cluster leader periodically rotates between flying nodes so that energy consumption is uniformly distributed between UAVs. This increases network lifespan. Each LU_k is responsible for managing and monitoring its clusters. It creates two types of communication, namely intra-cluster and inter-cluster communications on the network.
- **MU_j^k:** In QSCR, each MU_j^k is responsible for sensing the surrounding environment and transmitting this information to it. These nodes support only one type of communication, namely, intra-cluster communication.
- **GU_g^k:** In QSCR, each GU_g^k must perform the tasks of a cluster member node, namely sensing the environment and sending information to LU_k. Also, it participates in inter-cluster routing process and sends the required information to adjacent clusters. These nodes support two types of communication, namely inter-cluster and intra-cluster communications.

5. Proposed scheme

Here, a Q-learning-based smart clustering routing method (QSCR) is described for flying ad hoc networks. QSCR includes four phases:

- Dynamic neighbor discovery
- Adaptive clustering
- Smart and dynamic adjustment of clustering parameters
- Greedy routing

See the schematic design of QSCR in Fig. 2. In addition, Table 2 presents the symbols used in the proposed scheme.

5.1. Dynamic neighbor discovery

In QSCR, the neighbor discovery process gives U_i this opportunity to get a local network topology and use this local information in clustering and routing processes. In this process, hello packets play an important role in finding neighboring nodes and building a neighbor table. The hello propagation operation between neighboring UAVs is shown in Fig. 3. In QSCR, the hello packet related to U_i is marked as H_i. The content of H_i and its propagation period are two important

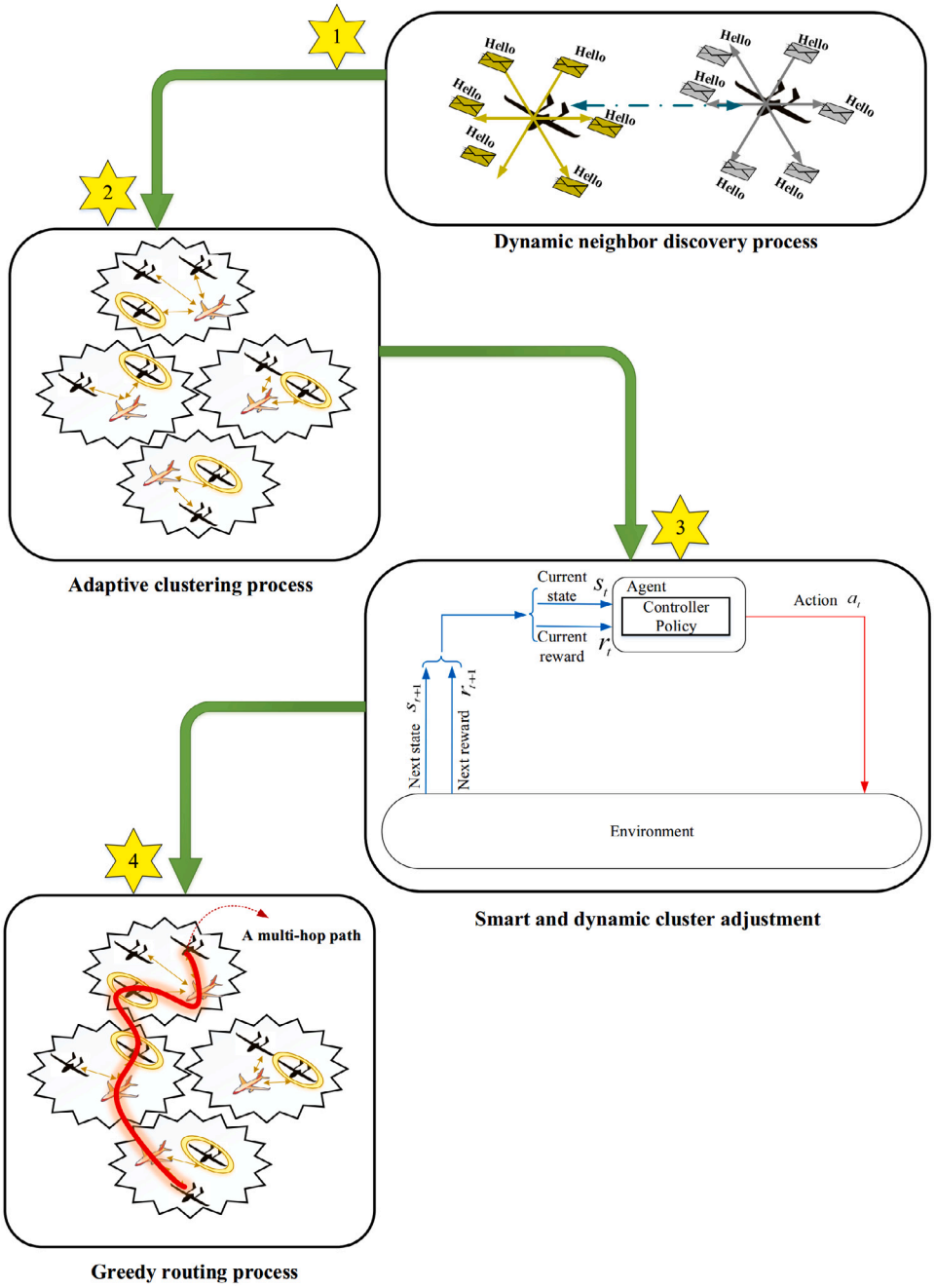

Fig. 2. Schematic design of QSCR.

points in the neighbor discovery process. The content of H_i specifies that U_i wants to access what information from its neighboring nodes, and the propagation period determines what time interval is suitable for broadcasting these packets on the network regularly. Algorithm 1 provides the pseudo-code related to this process.

- **Content:** H_i includes ID_i, spatial coordinates $Loc_i = (x_i^t, y_i^t, z_i^t)$, speed and movement angle $Mob_i = (V_i^t, \theta_i^t, \varphi_i^t)$, identifier of cluster leader (ID_{LU_k}), and the propagation period of H_i (HT_i). The structure of H_i is stated in Eq. (2). Note that if U_i is a cluster leader, it inserts its ID into the field ID_{LU_k}. Otherwise, it inserts the identifier of its cluster leader in this field.

$$H_i = \left\langle ID_i \middle\| Loc_i \middle\| Mob_i \middle\| ID_{LU_k} \middle\| HT_i \right\rangle \tag{2}$$

When U_i gets H_j from one of the adjacent UAVs, such as U_j, then U_i examines the following conditions for storing the content of H_j in its neighbor table ($Table_i^{Neighbor}$).

- If U_i has information about U_j in $Table_i^{Neighbor}$, then U_i examines the new content of H_j and updates the entry related to U_j in $Table_i^{Neighbor}$.
- If U_i has no information about U_j in $Table_i^{Neighbor}$, then U_i extracts the content of H_j and stores it in a new entry added to $Table_i^{Neighbor}$.
- If U_i has information about U_j in $Table_i^{Neighbor}$ but it does not get any new H_j from U_j, then U_i deletes U_j from its neighbor list by removing its information from $Table_i^{Neighbor}$.

Table 2
Symbols used in QSCR.

Symbol	Description
U_i	Flying node i in the network
n	Total number of flying nodes in the network
r_i	Communication range of U_i
N_i	Number of neighbors of U_i in $Table_i^{Neighbor}$
C_k	Cluster k
m	Total number of clusters in the network
LU_k	Cluster leader in C_k
MU_j^k	jth cluster member in C_k
N_k	Total number of cluster members in C_k
GU_g^k	gth inter-cluster gateways in C_k
H_i	Hello packet related to U_i
HT_i	Propagation time interval corresponding to H_i
ID_i	Identifier of U_i
$Loc_i = (x_i^t, y_i^t, z_i^t)$	Spatial coordinates of U_i
$Mob_i = (V_i^t, \theta_i^t, \varphi_i^t)$	Speed and movement angle of U_i
$Table_i^{Neighbor}$	Neighbor table stored in U_i
$V_{\max}$	Upper boundary of UAVs' speed in the network
$V_{\min}$	Lower boundary of UAVs' speed in the network
m_i^t	Merit of U_i
E_i	Residual energy of U_i
$E_{initial}$	Initial energy capacity of UAVs in the network
$\overline{CNR_i}$	Centrality of U_i
$D_{\max}$	Maximum distance between UAVs in the network
Deg_i	Neighbor degree of U_i
$\overline{SC_i^t}$	Average similarity between the velocity vector of U_i and those of its neighbors
LT_{ij}	Validity time of link between U_i and U_j
M_{merit}^i	Merit message of U_i
$M_{LU_k}^i$	Leader message sent by LU_k
$M_{MU_j^k}^i$	Connection message from MU_j^k to LU_k
$M_{LU_k}^{Accept}$	Accept message send by LU_k
$M_{GU_s^k}^{Gateway}$	Gateway message from GU_g^k to LU_k
$R_i(F)$	Reward function
α	Learning rate
γ	Discount factor
C_k^{change}	Stability value of C_k
$N_{C_k^{change}}$	Number of role changes in C_k

- **Propagation time interval (HT_k)**: In QSCR, the cluster leader LU_k is responsible for specifying the hello propagation time interval (HT_k) in its cluster (C_k) and updating it at each step. Then, LU_k must notify HT_k to the cluster member UAVs (MU_j^k so that $j = 1, 2, \ldots, N_k$), and MU_j^k must adhere to this propagation time interval. To determine HT_k in C_k, LU_k calculates the cosine similarity between two velocity vectors related to LU_k and MU_j^k based on Eq. (3).

$$SC_{kj}^t = \frac{V_{LU_k}^x V_{MU_j^k}^x + V_{LU_k}^y V_{MU_j^k}^y + V_{LU_k}^z V_{MU_j^k}^z}{\sqrt{\left(V_{LU_k}^x\right)^2 + \left(V_{LU_k}^y\right)^2 + \left(V_{LU_k}^z\right)^2} \times \sqrt{\left(V_{MU_j^k}^x\right)^2 + \left(V_{MU_j^k}^y\right)^2 + \left(V_{MU_j^k}^z\right)^2}} \tag{3}$$

So that $\left(V_{LU_k}^x, V_{LU_k}^y, V_{LU_k}^z\right)$ is the velocity vector of LU_k calculated through Eqs. (4), (5), and (6). $\left(V_{MU_j^k}^x, V_{MU_j^k}^y, V_{MU_j^k}^z\right)$ is also the velocity vector of MU_j^k that is obtained using a similar technique.

$$V_{LU_k}^x = V_{LU_k}^t \sin\varphi_{LU_k}^t \cos\theta_{LU_k}^t \tag{4}$$

$$V_{LU_k}^y = V_{LU_k}^t \sin\varphi_{LU_k}^t \sin\theta_{LU_k}^t \tag{5}$$

$$V_{LU_k}^z = V_{LU_k}^t \cos\varphi_{LU_k}^t \tag{6}$$

where $V_{LU_k}^t$, $\theta_{LU_k}^t$, and $\varphi_{LU_k}^t$ indicate the speed information of LU_k, namely the velocity length, the angle between the projection of the speed vector on the plane XY and the positive axis X, and the angle between the velocity vector and the positive axis Z.

Then, LU_k calculates the average similarity of its velocity vector to cluster members' velocity based on Eq. (7).

$$\overline{SC_k^t} = \frac{1}{N_k} \sum_{j=1}^{N_k} SC_{kj}^t \tag{7}$$

So that N_k indicates the number of cluster members in C_k. Then, LU_k must obtain HT_k from Eq. (8).

$$HT_k^t = \begin{cases} HT_k^0 & , \quad t = 0 \\ HT_k^{t-1} e^{\left(\frac{\overline{SC_k^t} - \overline{SC_k^{t-1}}}{HT_k^{t-1}}\right)}, & else \end{cases} \tag{8}$$

So that,

$$HT_k^0 = HT^{Default} e^{-\left(\frac{V_{LU_k}^t - V_{\min}}{V_{\max} - V_{\min}}\right)} \tag{9}$$

Here, $HT^{Default}$ represents a default value, for example, one second, for broadcasting hello messages in the network. $V_{\max}$ and $V_{\min}$ indicate the upper and lower boundaries related to the speed of UAVs in the network. $V_{LU_k}^t$ is the velocity length of LU_k.

Algorithm 1 Dynamic neighbor discovery

Input: U_i: Flying node i so that $i = 1, 2, \ldots, n$
N_i: Number of neighbors of U_i
H_i: Hello message related to U_i
ID_i: Identifier of U_i
$Loc_i = (x_i^t, y_i^t, z_i^t)$: Location information of U_i
$Mob_i = (V_i^t, \theta_i^t, \varphi_i^t)$: Mobility information of U_i
ID_{LU_k}: Identifier of the leader cluster corresponding to U_i
N_k: Number of members of the cluster k
HT_i: Hello dissemination period
t_{Net}: A timer for measuring the network time.
Output: $Table_i^{Neighbor}$: Neighboring table stored in U_i
Begin
repeat
  if t_{Net} mod $HT_i = 0$ **then**
    U_i: Obtain Loc_i and Mob_i from a positioning system (GPS);
    if U_i plays the role of LU_k **then**
      for $j = 1$ to N_k **do**
        U_i: Obtain the cosine similarity (SC_{kj}^t) between the velocity vectors of U_i and MU_j^k from Equation (3);
      end for
      U_i: Calculate the average cosine similarity ($\overline{SC_k^t}$) based on Equation (7);
      U_i: Refresh HT_i based on Equation (8);
      U_i: Send HT_i to all MU_j^k in its cluster;
    else
      U_i: Send a request message to LU_k to obtain updated HT_i;
    end if
    U_i: Insert ID_i, Loc_i, Mob_i, ID_{LU_k}, and HT_i into H_i based on Equation (2);
    U_i: Broadcast H_i to the neighboring nodes;
  end if
  for $j = 1$ to N_i **do**
    if U_i has information about U_j in $Table_i^{Neighbor}$ **and** U_j sends a new H_j to U_i **then**
      U_i: Refresh the entry related to U_j based on this H_j;
    else if U_i has information about U_j in $Table_i^{Neighbor}$ **and** U_j does not send any new H_j to U_i **then**
      U_i: Remove the entry related to U_j from $Table_i^{Neighbor}$;
    else if U_i has not any information about U_j in $Table_i^{Neighbor}$ **and** U_j sends a new H_j to U_i **then**
      U_i: Add a new entry related to U_j to $Table_i^{Neighbor}$;
      U_i: Insert the information of U_j into $Table_i^{Neighbor}$ based on this new H_j;
    end if
  end for
until Simulation time is finished
End

5.2. Adaptive clustering process

In this section, QSCR proposes an adaptive clustering process for flying ad hoc networks to categorize n flying nodes into m groups. Each

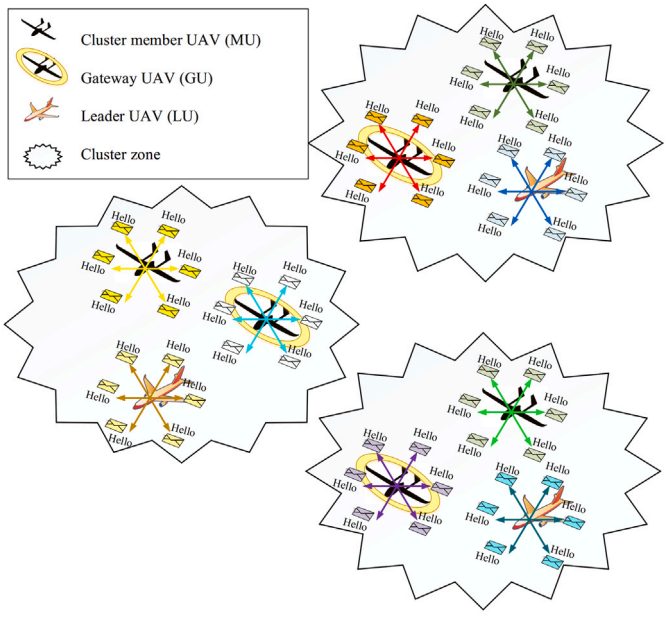

Fig. 3. Hello propagation operation between neighboring UAVs.

group is called a cluster C_k so that $k = 1, 2, \ldots, m$. C_k includes a cluster leader (LU_k) and a number of cluster members (MU_j^k) so that $j = 1, 2, \ldots, N_k$. Note that N_k indicates the total number of cluster members in C_k. In QSCR, the role of LU_k periodically rotates between UAVs so that energy consumption is uniformly distributed between all UAVs in the network. This leads to the improvement of network lifespan. LU_k is responsible for managing and monitoring its clusters. It provides two types of communication, namely intra-cluster and inter-cluster communications, on the network. Furthermore, some cluster member nodes close to the cluster border play the role of inter-cluster gateways (GU_g^k) to connect with the adjacent cluster. Algorithm 2 expresses the clustering process in QSCR. Furthermore, three following steps, namely merit value, cluster formation, and cluster support, explain how to select and support these nodes.

5.2.1. Merit value

In this step, each U_i firstly finds its neighbors and establishes $Table_i^{Neighbor}$ in accordance with the dynamic neighbor discovery process in Section 5.1. Then, U_i must prove its merit (m_i^t) to obtain the role of the cluster leader LU_k among other neighboring nodes. To get this purpose, U_i calculates its merit with regard to five merit parameters, namely remaining energy, centrality, neighbor degree, speed similarity, and link validity time.

- **Remaining energy ($\mathbf{E_i}$)**: It is essential to pay attention to energy in determining the merit value of flying nodes to accept the role of cluster leader. If U_i has little remaining energy, then it cannot play the role of LU_k well because the cluster leader has important and decisive tasks, namely cluster management, determination of the hello propagation time interval, intra-cluster communication, data aggregation, and inter-cluster communication. As a result, it deals with a lot of control and computational overhead. Therefore, LU_k needs more energy than normal nodes. In order to balance energy consumption in the network and prevent the death of low-energy UAVs in FANET, U_i should consider its remaining energy for calculating its merit. U_i knows about its energy level at any moment and normalizes it through Eq. (10). Note that QSCR carries out normalization operation to restrict energy value to $[0, 1]$. This leads to the same effect of merit parameters with different units on m_i^t.

$$E_i = \frac{E_i^{Residual}}{E_{initial}} \tag{10}$$

where $E_i^{Residual}$ and $E_{initial}$ present the current and initial energies of U_i, respectively.

- **Centrality ($\overline{\mathbf{CNR_i}}$)**: It is very important to consider the centrality parameter when determining the merit value of U_i for playing the role of LU_k. If U_i is close to the center of the cluster, the distance between it and cluster members is short. As a result, U_i requires less energy to form intra-cluster communication with MU_j^k. Eq. (11) calculates $\overline{CNR_i}$ that indicates the average distance between U_i and its neighbors, such as U_j, in $Table_i^{Neighbor}$.

$$\overline{CNR_i} = \frac{1}{N_i} \sum_{j=1}^{N_i} \sqrt{\left(x_i^t - x_j^t\right)^2 + \left(y_i^t - y_j^t\right)^2 + \left(z_i^t - z_j^t\right)^2} \tag{11}$$

where $Loc_i = \left(x_i^t, y_i^t, z_i^t\right)$ and $Loc_j = \left(x_j^t, y_j^t, z_j^t\right)$ shows the location information of U_i and U_j in the current time t, respectively. Furthermore, N_i indicates total number of adjacent UAVs around U_i in $Table_i^{Neighbor}$. As mentioned above, QSCR carries out normalization operation for limiting $\overline{CNR_i}$ to the interval $[0, 1]$. This causes the same effect of merit parameters with different units on m_i^t.

$$\overline{CNR_i^{norm}} = \frac{\overline{CNR_i}}{D_{\max}} \tag{12}$$

So that $D_{\max}$ indicates the longest distance between UAVs in the network. It is determined by Eq. (13) and is dependent on the dimensions of the network.

$$D_{\max} = \sqrt{X^2 + Y^2 + Z^2} \tag{13}$$

As, X, Y, and Z represent the length, width, and height of the desired network.

- **Neighbor degree ($\mathbf{Deg_i}$)**: This parameter is very important for determining the merit value of U_i and accepting the role of LU_k because if U_i has many neighbors, it has high communication capability. As a result, it can create stable and suitable communication with its cluster members as well as other adjacent cluster leaders. However, if U_i does not have a high neighbor degree, it may not be able to accept any node as its cluster member, and consequently, an isolated cluster is formed in the network. Deg_i is calculated based on the number of neighbors in $Table_i^{Neighbor}$. Note that QSCR performs normalization operation to limit Deg_i to $[0, 1]$.

$$Deg_i = \frac{N_i}{n - 1} \tag{14}$$

Where N_i is the number of adjacent flying nodes around U_i stored in $Table_i^{Neighbor}$ and n represents the total number of flying nodes in the network.

- **Speed similarity ($\overline{\mathbf{SC_i^t}}$)**: The importance of speed similarity to the merit value is that if the velocities of U_i and its neighbors are similar, then the cluster C_k is stable, and U_i can play the role of LU_k for a long period. However, if the difference between the speed of U_i and its neighbors is high, cluster members may speedily leave the cluster area of C_k, if U_i is selected as LU_k. As a result, the need to rebuild the cluster will increase. This imposes a lot of communication and computational overhead. In Section 5.1, Eq. (3) explains how to calculate $\overline{SC_i^t}$. The only point that is important here is that $\overline{SC_i^t}$ has a value in $[-1, 1]$. Hence, the normalization operation is performed on it using Eq. (15). In this case, the normalized value is limited to $[0, 1]$.

$$\overline{SC_i^{norm}} = \frac{\overline{SC_i^t} + 1}{2} \tag{15}$$

- **Link validity time ($\overline{LT_i^t}$)**: The importance of the link validity time in determining the merit value of U_i is that a flying node with powerful and stable communication links can play the role of LU_k better, and this increases the stability of C_k. However, if there are weak connections between LU_k and its cluster members, these connections will be broken rapidly, and consequently, C_k needs to be reconstructed. To calculate the link validity time, the distance between U_i and U_j in moment LT_{ij} is obtained from Eq. (16).

$$\left(D_{ij}^{LT_{ij}}\right)^2 = \left(\underbrace{\left(x_i^t - x_j^t\right)}_{\alpha_1} + \underbrace{\left(V_i^x - V_j^x\right)}_{\beta_1} LT_{ij}\right)^2 + \left(\underbrace{\left(y_i^t - y_j^t\right)}_{\alpha_2} + \underbrace{\left(V_i^y - V_j^y\right)}_{\beta_2} LT_{ij}\right)^2 + \left(\underbrace{\left(z_i^t - z_j^t\right)}_{\alpha_3} + \underbrace{\left(V_i^z - V_j^z\right)}_{\beta_3} LT_{ij}\right)^2 \tag{16}$$

So that $\left(V_i^x, V_i^y, V_i^z\right)$ is the velocity vector of U_i obtained through Eqs. (17), (18), and (19). Also, $\left(V_j^x, V_j^y, V_j^z\right)$ shows the velocity vector of U_j and is calculated in a similar manner.

$$V_i^x = V_i^t \sin\varphi_i^t \cos\theta_i^t \tag{17}$$

$$V_i^y = V_i^t \sin\varphi_i^t \sin\theta_i^t \tag{18}$$

$$V_i^z = V_i^t \cos\varphi_i^t \tag{19}$$

The general form of Eq. (16) is stated below.

$$\left(D_{ij}^{LT_{ij}}\right)^2 = \sum_{q=1}^{3}\left(\alpha_q\right)^2 + LT_{ij}\sum_{q=1}^{3}2\alpha_q\beta_q + \left(LT_{ij}\right)^2\sum_{q=1}^{3}\left(\beta_q\right)^2 \tag{20}$$

On the other hand, the distance between U_i and U_j in moment LT_{ij} equals their communication radius, i.e. $D_{ij}^{LT_{ij}} = r_i$. Hence, it can be concluded that:

$$\left(\left(\sum_{q=1}^{3}\left(\alpha_q\right)^2\right) - \left(r_i\right)^2\right) + LT_{ij}\sum_{q=1}^{3}2\alpha_q\beta_q + \left(LT_{ij}\right)^2\sum_{q=1}^{3}\left(\beta_q\right)^2 = 0 \tag{21}$$

Now, the Delta method is used to solve the above two-order equation to calculate LT_{ij} (Eq. (22)).

$$LT_{ij} = \frac{-\sum_{q=1}^{3}2\alpha_q\beta_q + \sqrt{\left(\sum_{q=1}^{3}2\alpha_q\beta_q\right)^2 - 4\left(\sum_{q=1}^{3}\left(\beta_q\right)^2\right)\left(\left(\sum_{q=1}^{3}\left(\alpha_q\right)^2\right) - \left(r_i\right)^2\right)}}{2\sum_{q=1}^{3}\left(\beta_q\right)^2} \tag{22}$$

Then, U_i obtains the average link time from Eq. (23).

$$\overline{LT_i^t} = \frac{1}{N_i}\sum_{j=1}^{N_i} LT_{ij} \tag{23}$$

so N_i represents the number of surrounding flying nodes around U_i in $Table_i^{Neighbor}$. Eq. (24) is also used to normalize $\overline{LT_i^t}$ and limit it to $[0, 1]$.

$$\overline{LT_i^{norm}} = \frac{\overline{LT_i^t}}{LT_{\max}} \tag{24}$$

Finally, U_i uses Eq. (25) to calculate m_i^t.

$$m_i^t = \omega_1 E_i + \omega_2\left(1 - \overline{CNR_i^{norm}}\right) + \omega_3 Deg_i + \omega_4\overline{SC_i^{norm}} + \omega_5\overline{LT_i^{norm}} \tag{25}$$

where ω_i $(i = 1, 2, 3, 4, 5)$ represents the weight coefficient that determines the effect of the relevant parameter on m_i^t. These weight coefficients are limited to $[0, 1]$ so that $\sum_{i=1}^{5}\omega_i = 1$. In QSCR, GCS is responsible for adjusting these weight coefficients using a Q-learning algorithm, which will be detailed in Section 5.3.

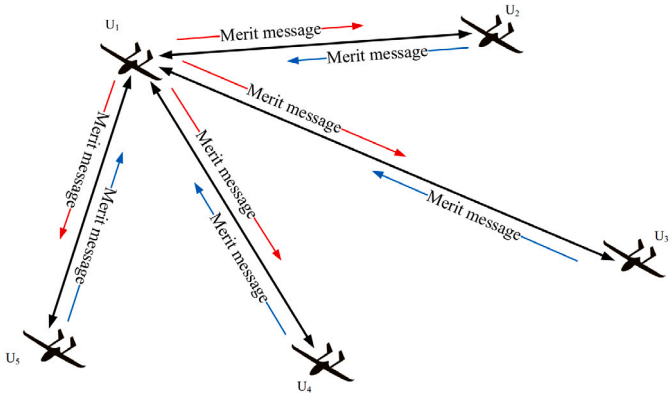

Fig. 4. Broadcasting merit messages for neighboring UAVs.

5.2.2. Cluster formation process

Here, the cluster construction steps, including cluster leader selection, connection to the cluster, and gateway selection are explained.

- ***Step (1) Cluster leader selection:*** To determine LU_k in C_k, U_i calculates its merit value m_i^t in accordance with Eq. (25) in Section 5.2.1 and then broadcasts this information through the merit message M_{merit}^i to surrounding nodes in the network. See Fig. 4. M_{merit}^i includes ID_i, m_i^t, and a timestamp T_{merit}. Its structure is stated in Eq. (26).

$$M_{merit}^i = \left\langle ID_i \middle\| m_i^t \middle\| T_{merit} \right\rangle \tag{26}$$

Now, U_i waits for a certain time interval to get merit messages from other UAVs. After receiving these messages, U_i extracts the merit values of UAVs from their merit messages and compares m_i^t with these values. Here, there are two modes.

  - **First mode:** If U_i has the greatest merit value among other neighboring nodes, it introduces itself as LU_k and transmits a leader message $M_{LU_k}^i$ to its adjacent nodes. The structure of $M_{LU_k}^i$ is stated in Eq. (27) and contains ID_i, m_i^t, the role of U_i (i.e. LU_k), and a timestamp T_{LU_k}.

$$M_{LU_k}^i = \left\langle ID_i \middle\| m_i^t \middle\| LU_k \middle\| T_{LU_k} \right\rangle \tag{27}$$

  - **Second mode:** If the merit value of U_i is less than other adjacent nodes, U_i waits to get a leader message from other flying nodes and goes to Step 2.

- ***Step (2) Connection to the cluster:*** In this step, U_i waits to get a leader message from other nodes. Now, three modes can occur. This process is shown in Fig. 5.

  - **First mode:** If U_i gets only a leader message from LU_k, then U_i acts as a cluster member MU_j^k and transfers a connection message $M_{MU_j^k}^i$ to LU_k. This message contains ID_i, MU_j^k (role of U_i), ID_{LU_k}, and timestamp $T_{MU_j^k}$, and its structure is presented in Eq. (28).

$$M_{MU_j^k}^i = \left\langle ID_i \middle\| MU_j^k \middle\| ID_{LU_k} \middle\| T_{MU_j^k} \right\rangle \tag{28}$$

When this message reaches LU_k, it sends an accept message $M_{LU_k}^{Accept}$ to U_i and accepts it as MU_j^k in the cluster C_k.

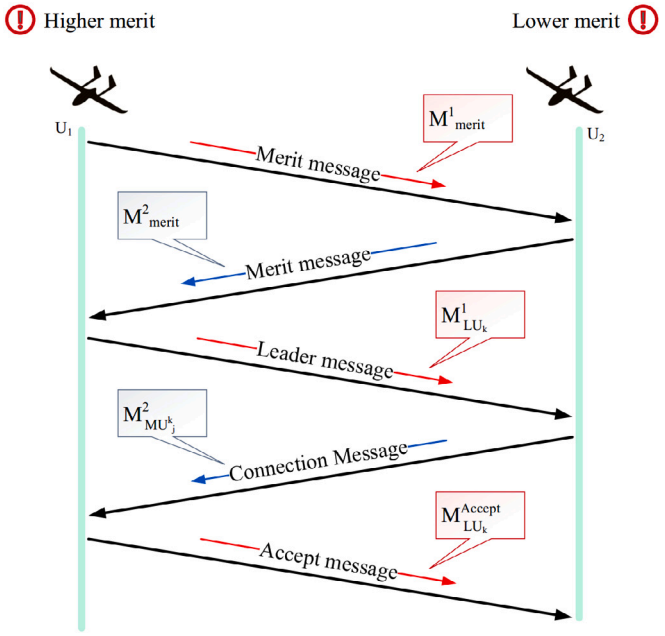

Fig. 5. Determining cluster members.

This message contains the cluster leader ID, the cluster member ID, and timestamp T_{Accept}, and its structure is stated in Eq. (29).

$$M_{LU_k}^{Accept} = \left\langle ID_{LU_k} \middle\| ID_{MU_j^k} \middle\| T_{Accept} \right\rangle \tag{29}$$

- **Second mode:** If U_i gets several leader messages from different cluster leaders, then U_i calculates the validity time of link between itself and each cluster leader. Then, it chooses LU_k with the longest connection time as its cluster leader and transmits a connection message $M_{MU_j^k}^{i}$ for it in the cluster C_k. After accepting this connection request by LU_k, U_i is accepted as MU_j^k in the cluster C_k.
- **Third mode:** If U_i does not receive any leader message from the adjacent nodes, then U_i announces itself as the cluster leader and transfers a leader message $M_{LU_k}^{i}$ to its adjacent nodes to determine its cluster members.

- ***Step (3) Gateway selection:*** At this step, gateway nodes are specified in each cluster. According to Section 5.1, U_i advertises its cluster leader ID (ID_{LU_k}) through hello messages to other adjacent nodes, and this information is recorded in $Table_i^{Neighbor}$. To determine gateway nodes, such as GU_g^k, suppose that MU_j^k belongs to the cluster C_k, and its cluster leader is LU_k. In this case, MU_j^k examines its neighbor table, if it finds a neighboring node that belongs to another cluster such as C_f and its leader cluster is LU_f, then MU_j^k is known as a gateway node to connect the cluster C_f and transmits a gateway message $M_{GU_g^k}^{Gateway}$ to LU_k. This message contains the gateway ID $ID_{GU_g^k}$, ID_{LU_k}, ID_{LU_f}, timestamp $T_{Gateway}$. The format of this message is stated in Eq. (30).

$$M_{GU_g^k}^{Gteway} = \left\langle ID_{GU_g^k} \middle\| ID_{LU_k} \middle\| ID_{LU_f} \middle\| T_{Gateway} \right\rangle \tag{30}$$

Algorithm 2 Adaptive clustering

Input: U_i: Flying node i so that $i = 1, 2, ..., n$
$Table_i^{Neighbor}$: Neighboring table stored in U_i
N_i: Number of neighbors of U_i
C_k: Cluster k so that $k = 1, 2, ..., m$
LU_k: Leader of C_k
MU_j^k: Cluster member UAV so that $j = 1, 2, ..., N_k$
GU_g^k: Gateway UAV in C_k
t_{Net}: A timer for measuring the network time.
Output: $C_1, ..., C_k, ..., C_m$: Clusters in the network
Begin
repeat
  if t_{Net} mod $Clustering\ period = 0$ **then**
    U_i: Obtain the weight coefficients (ω_i, where $i = 1, 2, 3, 4, 5$) using the learning model designed in GCS;
    U_i: Calculate m_i^t based on Equation (25);
    U_i: Insert m_i^t into the message M_{merit}^{i} based on Equation (26);
    U_i: Broadcast M_{merit}^{i} to its neighboring nodes;
    for $j = 1$ to N_i **do**
      U_i: Wait to receive M_{merit}^{j} from U_j;
      U_i: Extract m_j^t from M_{merit}^{j};
      if $m_i^t > m_j^t$ **then**
        $m_{\max} = m_i^t$;
      else
        $m_{\max} = m_j^t$;
      end if
    end for
    if $m_{\max} = m_i^t$ **then**
      U_i: Play the role of LU_k and adjust the message $M_{LU_k}^{i}$ based on Equation (27);
      U_i: Broadcast $M_{LU_k}^{i}$ to its neighboring nodes;
      if LU_k receives a message $M_{MU_j^k}^{i}$ from a MU_j^k **then**
        LU_k: Set a message $M_{LU_k}^{Accept}$ based on Equation (29);
        LU_k: Send $M_{LU_k}^{Accept}$ to the relevant MU_j^k;
      end if
    else
      U_i: Wait to receive the messages $M_{LU_k}^{i}$ from different LU_k nodes;
      if U_i receives only one $M_{LU_k}^{i}$ from a LU_k node **then**
        U_i: Play the role of MU_j^k and adjust a message $M_{MU_j^k}^{i}$ based on Equation (28);
        U_i: Send $M_{MU_j^k}^{i}$ to the relevant LU_k;
      else if U_i receives several $M_{LU_k}^{i}$ from different LU_k nodes **then**
        U_i: Calculate the link lifetime between itself and each LU_k using Equation (22);
        U_i: Select LU_k with maximum link lifetime as its leader;
        U_i: Play the role of MU_j^k and adjust a message $M_{MU_j^k}^{i}$ based on Equation (28);
        U_i: Send $M_{MU_j^k}^{i}$ to the relevant LU_k;
      else if U_i does not receive any $M_{LU_k}^{i}$ from LU_k nodes **then**
        U_i: Play the role of LU_k and adjust the message $M_{LU_k}^{i}$ based on Equation (27);
        U_i: Broadcast $M_{LU_k}^{i}$ to its neighboring nodes;
        if LU_k receives a message $M_{MU_j^k}^{i}$ from a MU_j^k **then**
          LU_k: Set a message $M_{LU_k}^{Accept}$ based on Equation (29);
          LU_k: Send $M_{LU_k}^{Accept}$ to the relevant MU_j^k;
        end if
      end if
      if MU_j^k belongs to C_k **and** MU_j^k has a neighbor, which belongs to another cluster like C_f **then**
        MU_j^k: Play the role of GU_g^k and adjust the message $M_{GU_g^k}^{Gateway}$ based on Equation (30);
        MU_j^k: Send $M_{GU_g^k}^{Gateway}$ to LU_k;
      end if
    end if
  end if
until Simulation time is finished
End

Algorithm 3 Supporting the clustering process

Input: U_i: Flying node i so that $i = 1, 2, ..., n$
$Table_i^{Neighbor}$: Neighboring table stored in U_i
N_i: Number of neighbors of U_i
C_k: Cluster k so that $k = 1, 2, ..., m$
LU_k: Leader of C_k
MU_j^k: Cluster member UAV so that $j = 1, 2, ..., N_k$
GU_g^k: Gateway UAV in C_k
t_{Net}: A timer for measuring the network time.
HT_k: Hello dissemination time
H_k: Hello message related to LU_k
Output: Updating clusters $C_1, ..., C_k, ..., C_m$ in the network
Begin
repeat
  for $k = 1$ to m **do**
    if t_{Net} mod $HT_k = 0$ **then**
      LU$_k$: Adjust a message H_k based on Equation (2);
      LU$_k$: Broadcast H_k to its cluster member nodes;
      for $j = 1$ to N_k **do**
        if MU_j^k does not receive any H_k from LU_k **then**
          MU$_j^k$: Adjust a message $M^i_{MU_j^k}$ based on Equation (28);
          MU$_j^k$: Send $M^i_{MU_j^k}$ to the nearest LU_k;
          if this new LU_k receives $M^i_{LU_k}$ from MU_j^k **then**
            LU$_k$: Set a message $M^{Accept}_{LU_k}$ based on Equation (29);
            LU$_k$: Send $M^{Accept}_{LU_k}$ to the relevant MU_j^k;
          end if
        end if
        MU$_j^k$: Adjust a message H_j based on Equation (2);
        MU$_j^k$: Broadcast H_j to its LU_k;
        if LU_k does not receive any H_j from MU_j^k **then**
          LU$_k$: Remove MU_j^k from its member list;
        end if
      end for
    end if
  end for
  if a new U_i is connected to the network **then**
    U$_i$: Play the role of MU_j^k and adjust a message $M^i_{MU_j^k}$ based on Equation (28);
    U$_i$: Send $M^i_{MU_j^k}$ to the nearest LU_k;
    if LU_k receives $M^i_{LU_k}$ from this new node **then**
      LU$_k$: Set a message $M^{Accept}_{LU_k}$ based on Equation (29);
      LU$_k$: Send $M^{Accept}_{LU_k}$ to the relevant MU_j^k;
    end if
  end if
until Simulation time is finished
End

5.2.3. Cluster support process

Here, different cluster support steps are explained.

- **Connecting a new node:** When a new node enters the network, it must propagate a connection request in the network. As soon as the connection message reaches cluster leaders in the network, they respond to this request. Finally, this new node is connected to the nearest cluster leader.
- **Connecting with cluster members:** LU_k periodically examines its connections with its cluster members through the exchange of hello messages. If LU_k does not receive any hello message from MU_j^k, then it has gone out of the communication area of LU_k and should be removed from the list of cluster members.
- **Connecting to the cluster leader:** MU_j^k periodically controls its connection with LU_k through the exchange of hello messages. If MU_j^k does not receive any hello message from LU_k, it is clear that LU_k has gone out of the communication area of MU_j^k. As a result, MU_j^k must send a connection request to the nearest cluster leader to join a new cluster.
- **Connecting to gateways:** GU_g^k also periodically examines its connection with the relevant gateway in the adjacent cluster. If this gateway node does not receive any hello message from its neighboring gateway in the adjacent cluster, then the connection between the two nodes has been cut off. Hence, GU_g^k must report this issue to its cluster leader.

Table 3
Components related to the learning model in QSCR.

Learning component	Description
Learning issue	Learning the best weight coefficients for merit parameters
Learning environment	Flying ad hoc network
Learning agent	Ground station control
State space	Merit values of UAVs
Action space	Determining the weight coefficients

- **Re-clustering:** When the clustering period is completed, UAVs again begin the clustering process in accordance with Section 5.2 to determine new clusters in the network.

Algorithm 3 expresses the pseudo-code related to the cluster support process.

5.3. Smart and dynamic adjustment of clustering parameters

Here, QSCR provides a centralized learning model to tune up weight coefficients (ω_i, so that $i = 1, 2, 3, 4, 5$) in Eq. (25). This learning model is depicted in Fig. 6. Additionally, QSCR regulates Q-learning parameters to be more consistent with FANET. Table 3 describes different components related to this learning model. GCS is the learning agent, which learns the network environment, and gets the best weight coefficients through trial and error and sends them to UAVs. Then, flying nodes apply these weight coefficients in the clustering process to choose cluster leaders. U_i makes a state message, which includes its five merit parameters, namely remaining energy (E_i), centrality ($\overline{CNR_i}$), neighbor degree (D_i), speed similarity ($\overline{SC_i^t}$), and link validity time ($\overline{LT_i^t}$), and sends it to GCS periodically. GCS stores the merit parameters related to each flying node in its database and uses Q-learning to find the best weight coefficients in Eq. (25). Therefore, the effect of each coefficient ω_i on the merit value is determined in a dynamic manner and according to network conditions. Based on the points mentioned above, the action space in the proposed learning model includes a set of weight coefficients, i.e. $Action = \left\{\omega_i | 1 \le i \le 5, 0 \le \omega_i \le 1 \ and \ \sum_{i=1}^{5} \omega_i = 1\right\}$. For each action a^t, the state space (ST) represents a set of merit values of UAVs so that $ST = \left\{m_i^t | 1 \le i \le n, \ a^t \in Action\right\}$. Algorithm 4 explains how to adjust weight coefficients dynamically and intelligently.

Assume that GCS selects an action such as $a^t = \left\{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\right\}$ from the action set. In this case, the state of UAVs is $ST^t = \left\{m_1^t, m_2^t, \ldots, m_n^t\right\}$. Then, GCS examines the effect of action a^t on the network environment by evaluating the clusters formed in FANET and the selected cluster leaders. It calculates the reward value based on Eq. (35). This reward function considers three scales, namely the balance of energy consumption, the number of isolated clusters, and the distribution of cluster leaders in the network. In the following, these scales will be precisely defined.

- **Balance of energy consumption ($\mathbf{f_1}$):** It evaluates the effect of the selected action ($a^t = \left\{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\right\}$) on the balance of energy consumption in the network. To balance energy consumption in the network, GCS must consider two points when adjusting weight coefficients in the network. Firstly, the energy consumed by UAVs must be minimized, and secondly, high-energy UAVs are selected as cluster leaders. Therefore, f_1 must evaluate the energy consumption rate of all flying nodes (cluster member node or cluster leader). If this rate is low, this means that the selected action can reduce the energy consumption of UAVs in the network. In addition, f_1 must evaluate the ratio of the sum of the residual energy of cluster leaders to the sum of the residual energy of cluster member nodes. If the selected action can increase this parameter, it means that cluster leaders have more energy than

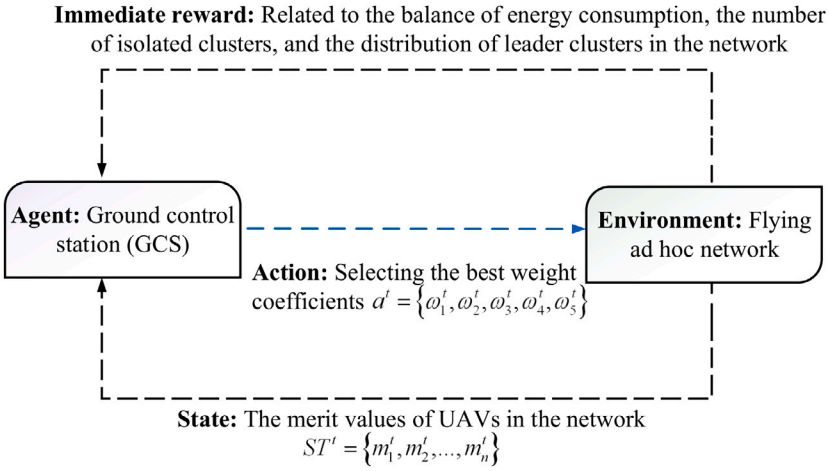

Fig. 6. Learning model in QSCR.

cluster members in the network, and cluster leaders are selected from high-energy nodes.

$$f_1 = \ell \frac{1}{\left(\frac{1}{n}\sum_{i=1}^{n} \frac{E_i^{Residual}(t-1) - E_i^{Residual}(t)}{\Delta t}\right)} + (1-\ell)\sum_{k=1}^{m}\left(\frac{E_{LU_k}^{Residual}(t)}{\left(\sum_{\forall MU_j^k \in C_k} E_{MU_j^k}^{Residual}(t)\right)}\right) \tag{31}$$

So that n indicates the total number of flying nodes, m is the total number of clusters, $E_i^{Residual}(t-1)$ and $E_i^{Residual}(t)$ show the energy levels of U_i in two moments $t-1$ and t, respectively. ℓ is a weight coefficient in $[0, 1]$.

- **Number of isolated clusters (f_2)**: This parameter in the reward function evaluates the effect of the selected action $a^t = \{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\}$ on the number of isolated clusters made in FANET. Note that isolated clusters are clusters with one or two members. If the number of isolated clusters is high in the network, the clustering process is not successful because the data transmission process between UAVs is often done through inter-cluster communications. This increases the number of hops and delay in the routing process, and boosts the energy consumption of UAVs. Therefore, GCS is looking for weight coefficients to reduce the number of isolated clusters in the network. As a result, f_2 is obtained from the ratio of the number of isolated clusters to the total number of clusters in FANET.

$$f_2 = 1 - \left(\frac{m_{isolated\ cluster}}{m}\right) \tag{32}$$

where m and $m_{isolated\ cluster}$ are the total number of clusters and the number of isolated clusters.

- **Distribution of cluster leaders (f_3)**: This parameter in the reward function expresses the effect of the selected action $a^t = \{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\}$ on the distribution of cluster leaders in FANET. Note that a desirable clustering process must consider two points. Firstly, cluster leaders are close to the cluster center. This means that the average distance between cluster members ($\forall MU_j^k \in C_k$) and its cluster leader (LU_k) is low, and secondly, the distance between cluster leaders is high. This leads to their suitable distribution throughout the network. Therefore, f_3 is looking for weight coefficients that optimize the distribution of cluster leaders on the network.

$$f_3 = \beta \frac{1}{\sum_{k=1}^{m}\left(\frac{\sum_{\forall MU_j^k \in C_k} d\left(MU_j^k, LU_k\right)}{N_k}\right)} + (1-\beta)\sum_{\forall LU_k \neq LU_f} \frac{d\left(LU_k, LU_f\right)}{D_{\max}} \tag{33}$$

So that $d\left(LU_k, LU_f\right)$ is the distance between LU_k and LU_f, and $d\left(MU_j^k, LU_k\right)$ indicates the distance between LU_k and MU_j^k. N_k is the number of cluster members in C_k, and m is the total number of clusters in the network. β is also a weight coefficient in $[0, 1]$. $D_{\max}$ indicates the maximum distance between UAVs in the network and is determined using Eq. (34).

$$D_{\max} = \sqrt{X^2 + Y^2 + Z^2} \tag{34}$$

where X, Y, and Z represent the length, width and height of the network, respectively.

Now, GCS uses Eq. (35) to calculate the reward function based on the parameters mentioned above.

$$R_t(F) = \begin{cases} R_{\max}, & F = 1 \\ R_{\min}, & F = -1 \\ F, & Otherwise \end{cases} \tag{35}$$

So that $F = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$. In addition, λ_1, λ_2, and λ_3 express weight coefficients and $\sum_{i=1}^{3} \lambda_i = 1$. According to the reward function, if $F = 1$, the selected action $a^t = \{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\}$ obtains the maximum reward value $R_{\max}$. On the other hand, if $F = -1$, then, $a^t = \{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\}$ gets the least reward value $R_{\min}$. In other modes, the reward value is dependent on F.

5.3.1. Q-learning parameters

Here, QSCR seeks to find its learning parameters, i.e. α and γ, in an adaptive manner and based on network conditions to improve its performance in the dynamic environment of FANET. Learning rate (α) is limited to $0 < \alpha \leq 1$. It determines how much the virtual agent relies on old and new information in the learning process. If $\alpha = 0$, it means that the virtual agent performs the learning process only based on old information, and if $\alpha = 1$, the agent seeks to discover and learn the environment based on new information. In QSCR, the learning rate is measured based on the stability of clusters formed in the network. The cluster stability means that flying nodes can maintain their role in clusters, so clusters need fewer changes. This parameter is obtained

based on Eq. (36).

$$C_k^{change} = \begin{cases} 0, & N_{C_k^{change}} \leq \varphi \\ -1, & N_{C_k^{change}} > \varphi \end{cases} \tag{36}$$

where $N_{C_k^{change}}$ represents the number of changes in C_k in a specified time interval. If $N_{C_k^{change}}$ is less than a threshold value φ, C_k is a stable cluster. In this case, the learning rate (α) can be reduced. However, if C_k is unstable, the learning rate must be increased. Now, the average stability of all clusters in the network is achieved through Eq. (37).

$$\bar{C}_k^{change} = \frac{1}{m}\sum_{k=1}^{m} C_k^{change} \tag{37}$$

So that m the number of clusters in the network. Finally, α is calculated based on Eq. (38).

$$\alpha = \begin{cases} 1 - e^{\bar{C}_k^{change}}, & \bar{C}_k^{change} \neq 0 \\ 0.1, & \bar{C}_k^{change} = 0 \end{cases} \tag{38}$$

Algorithm 4 Dynamic and smart adjustment of weight coefficients

Input: GCS (Ground control station): Agent
U_i: Flying node i so that $i = 1, 2, ..., n$
$Table_i^{Neighbor}$: Neighboring table stored in U_i.
N_i: Number of neighbors of U_i
ϵ, α, γ: Learning parameters
$ST = \{m_i^t | 1 \leq i \leq n, a^t \in Action\}$: State space set
$Action = \left\{ \omega_i | 1 \leq i \leq 5, 0 \leq \omega_i \leq 1 \text{ and } \sum_{i=1}^{5} \omega_i = 1 \right\}$: Action set
Output: Updating Q-table for determining the best weight coefficients in Equation (25)
Begin
for $i = 1$ to n **do**
  U_i: Send its merit parameters including E_i, $\overline{CNR_i}$, Deg_i, $\overline{SC_i^t}$, and $\overline{LT_i^t}$ to GCS;
end for
GCS: Set ϵ in the interval $[0, 1]$ randomly;
GCS: Adjust Q-values in Q-table on zero;
repeat
  GCS: Select the next-state as $ST^t = \left\{m_1^t, m_2^t, ..., m_n^t\right\}$ randomly;
  GCS: Extract α and γ from Equations (38) and (40), respectively;
  $t = 1$;
  while $t \leq N$ **do**
    GCS: Get a positive random number (pn_r) in $[0, 1]$;
    if $pn_r \leq \epsilon$ **then**
      GCS: Obtain the action $a^t = \left\{\omega_1^t, \omega_2^t, \omega_3^t, \omega_4^t, \omega_5^t\right\}$ from $Action = \left\{ \omega_i | 1 \leq i \leq 5, 0 \leq \omega_i \leq 1 \text{ and } \sum_{i=1}^{5} \omega_i = 1 \right\}$
    else if $pn_r > \epsilon$ **then**
      GCS: Extract the action with maximum Q-value from Q-table;
    end if
    GCS: Calculate $F = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$ based on Equations (31), (32), and (33);
    if $F = 1$ **then**
      $R_t = R_{\max}$;
    else if $F = -1$ **then**
      $R_t = R_{\min}$;
    else
      $R_t = F$;
      GCS: Get the next-state $ST^t = \left\{m_1^t, m_2^t, ..., m_n^t\right\}$ from the environment;
      GCS: Refresh Q-value in Q-table based on R_t;
      $t = t + 1$;
    end if
  end while
  $episode = episode + 1$;
until $episode \leq M$
End

In addition, QSCR seeks to adjust the discount coefficient γ in the network. Note that this coefficient is limited to $0 < \gamma \leq 1$. If $\gamma = 1$, it can be concluded that Q-value is stable, and the virtual agent considers past experiences to determine Q-value. However, if $\gamma = 0$, Q-value is unstable. As a result, the last reward value received from the environment should have a greater effect on the determination of Q-value. In QSCR, γ is evaluated based on the similarity between the speed vectors of cluster members and the cluster leader. If difference between these velocity vectors is high, the relevant cluster is unstable. $\overline{SC_k^t}$ is the average speed similarity between LU_k and its cluster members. It was stated in Section 5.1 and Eq. (3). Now, Eq. (39) is used to normalize this parameter and limit it to $[0, 1]$.

$$\overline{SC_k^{norm}} = \frac{\overline{SC_k^t} + 1}{2} \tag{39}$$

As a result, γ is calculated based on Eq. (40).

$$\gamma_j = 1 - e^{-\left(\frac{1}{m}\sum_{k=1}^{m} \overline{SC_k^{norm}}\right)} \tag{40}$$

where m is the number of clusters in the network.

5.4. Greedy routing

Assume that U_i intends to transmit its data packet to the destination U_d. In the following, the greedy routing process is clarified in QSCR. If U_i and U_d are directly connected to each other, U_i transfers its data packet to U_d directly. Otherwise, U_i executes a cluster-based greedy routing process and examines the following three modes. Algorithm 5 shows the pseudo-code related to this process.

- **First mode:** If U_i plays the role of LU_k in the network, then LU_k examines the position of its adjacent clusters and selects the nearest cluster leader to the destination, (i.e. LU_f in the cluster C_f) as the next-hop node. If LU_k and LU_f are in each other's communication range, LU_k sends its data to LU_f directly. Otherwise, LU_k sends this data to the gateway node that is connected to C_f.
- **Second mode:** If U_i acts as a cluster member MU_j^k, it is allowed to send its data packets to its cluster leader LU_k. Then, LU_k is responsible for sending data to the destination.
- **Third mode:** If U_i plays the role of GU_g^k in C_k, then it examines the destination address inserted in the data packet. If it is the address of the adjacent cluster that is connected to GU_g^k. Then, GU_g^k sends this packet to the nearest gateway node in the adjacent cluster. Otherwise, GU_g^k transmits its data to its cluster leader LU_k.

Algorithm 5 Greedy routing

Input: U_i: Flying node i so that $i = 1, 2, ..., n$
U_d: Destination node
Output: Forming a route between U_i and U_d
Begin
if U_i seeks to send its data packets to U_d **then**
  if U_i and U_d are neighbor **then**
    U_i: Send its data packet to U_d directly;
  else
    if U_i plays the role of a LU_k in the network **then**
      LU_k: Select the nearest cluster toward U_d, for example LU_f, as the next-hop;
      if LU_k and LU_f communicate each other directly **then**
        LU_k: Send its data packets to LU_f;
      else
        LU_k: Send its data packets to the GU_g^k connected to LU_f;
      end if
    end if
    if U_i plays the role of a MU_j^k in C_k **then**
      MU_j^k: Send its data packets to its LU_k;
      if U_i plays the role of a GU_g^k in C_k **then**
        GU_g^k: Checks the next-hop address inserted in to the data packet;
        if the next-hop address is the address of the adjacent cluster **then**
          GU_g^k: Send its data packets to the nearest gateway node to U_d in this adjacent cluster;
        else
          GU_g^k: Send its data packets to its LU_k;
        end if
      end if
    end if
  end if
end if
End

6. Simulation and evaluation of results

In this section, the simulation of QSCR is run on the network simulator 2 (NS2). In this process, nodes use the Gauss Markov (GM) mobility model to simulate the movement of UAVs in FANET. The initial energy of these nodes is 2000 joules. In simulation scenarios, the number of UAVs varies between 10 and 100 nodes, and the dimensions of the network are 10×10 km^2. UAVs have different velocities between 30 and 50 m/s, and their communication radius is 250 m. Also, the total simulation time is 1500 s. These simulation parameters are stated in Table 4. In the simulation process, the results of QSCR are compared with three methods, namely ICRA (Guo et al., 2022), WCA (Chatterjee et al., 2002), and DCA (Ergenç et al., 2019). Note that ICRA, WCA, and DCA have a fixed hello propagation time interval (i.e. 2 s), but in QSCR, the cluster leader specifies the hello propagation time interval in its cluster based on speed similarity. In the simulation process, three different scenarios are intended to examine the performance of QSCR from different aspects. These scenarios are related to initial energy, network density, and speed of UAVs in the network. In the following, these scenarios are introduced:

- **First scenario:** This scenario assumes that (1) Initial energy of UAVs in the network follows a uniform distribution. (2) UAVs are uniformly scattered in the network. (3) Speed of UAVs is the same.
- **Second scenario:** The assumptions in the second scenario are (1) Initial energy of all UAVs is the same. (2) UAVs are uniformly distributed on the network and their density is different in distinguish network areas. (3) Speed of UAVs is a fixed value.
- **Third scenario:** The assumptions of the third scenario are: (1) Initial energy of all UAVs is the same. (2) UAVs are uniformly distributed in the network. (3) Speed of UAVs is different.

In the simulation process, QSCR is compared to other methods in terms of different evaluation scales, namely cluster formation time, cluster stability, network lifespan, routing overhead, and QoS requirements. These evaluation scales are introduced below:

- **Cluster creation time:** This evaluation scale represents a timeframe from when the network starts the clustering operation until it completes the first clustering process. It shows the efficiency of a clustering scheme. If a clustering method creates clusters rapidly, it is an effective scheme.
- **Cluster stability:** This evaluation scale is calculated based on the average number of changes in the role of each node in the cluster. This scale shows whether a cluster needs to rebuild its clusters or not. If a clustering method creates clusters that experience fewer changes in the role of UAVs, this method makes a stable network topology.
- **Network lifespan:** This evaluation scale shows the time that the first node dies in the network. If UAVs live for a longer time, this clustering method achieves better energy efficiency.
- **QoS requirements:** Here, three QoS scales, namely end-to-end delay, the number of isolated clusters, and packet delivery rate are considered.
- **Routing overhead:** This evaluation scale shows the number of exchanged control messages between drones to form a clustered network topology. These control messages have a high effect on the energy consumption of UAVs in FANET. When a clustering method produces a large number of control messages, its communication overhead will be high. As a result, the energy consumption of UAVs will increase in this clustering method.

Table 4
Simulation parameters.

Parameter	Value
Simulator	NS2
Network size	10×10 km^2
Number of UAVs	10–100
Initial energy of UAVs	2000 J
Communication range of UAVs	250 m
Velocity of UAVs	30–50 m/s
Size of data packet	1024 bits
Mobility model	Gauss Markov (GM)
Traffic model	Constant bit rate (CBR)
Data rate	1000 kbps
Antenna	Omni-Antenna
Simulation time	1500 s
Mac standard	IEEE 802.11
φ (Role change threshold)	2
Evaluation scales	Time of cluster creation, stability of clusters, network lifetime, routing overhead, and QoS requirements
Compared schemes	QSCR, ICRA, WCA, and DCA

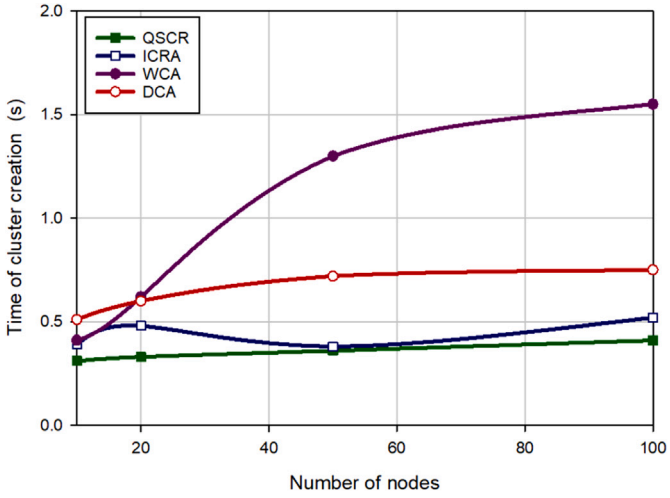

Fig. 7. Cluster creation time in different methods.

6.1. Cluster creation time

In Fig. 7, the cluster creation time is stated in different schemes for scenario one. According to this figure, QSCR has the least cluster creation time. It reduces this scale by approximately 20.34%, 63.66%, and 45.35% compared to ICRA, WCA, and DCA, respectively. As shown in this figure, ICRA also has a very good performance on this scale, but it takes a little longer time than QSCR to complete the cluster construction process. The proposed scheme has better performance than ICRA because QSCR uses an adaptive hello mechanism and has different hello propagation intervals in various clusters. This reduces the cluster formation time because this mechanism decreases communication overhead in QSCR. However, ICRA, WCA, and DCA have used a fixed time interval to broadcast hello messages in FANET. DCA considers a greater number of clustering criteria than QSCR and ICRA. Hence, it has a weaker performance than these two schemes. WCA has the worst performance because it uses a complex CH selection process. Another point in Fig. 7 is that QSCR, ICRA, and DCA are not sensitive to the number of UAVs, and change in network density does not affect the time of cluster construction, but WCA is sensitive to the network density so a high number of UAVs needs to more time for completing the clustering process in FANET.

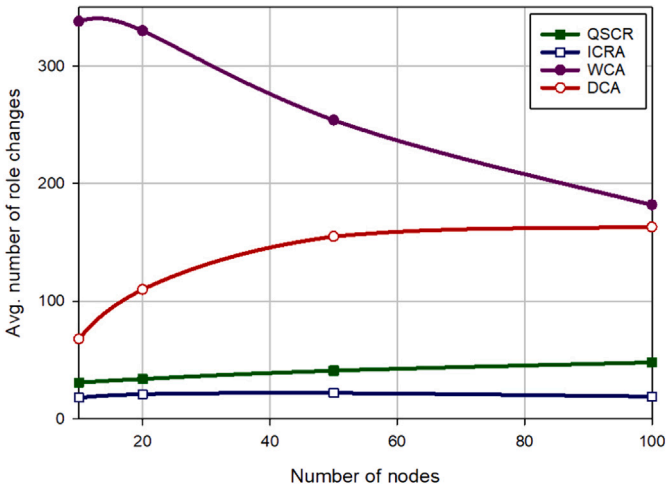

Fig. 8. Evaluation of cluster stability in the first scenario.

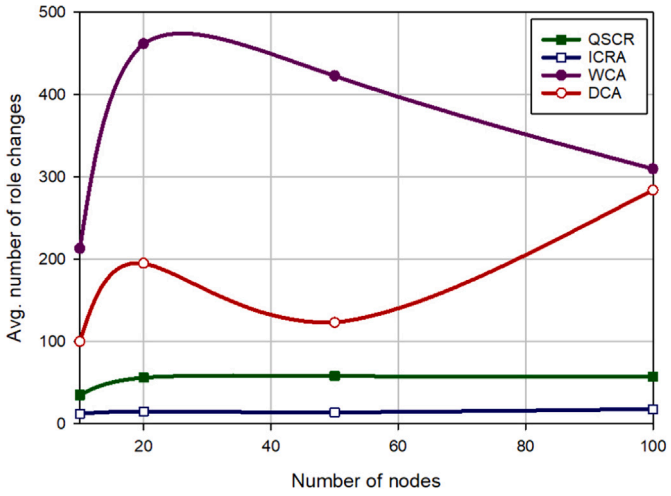

Fig. 9. Evaluation of cluster stability in the second scenario.

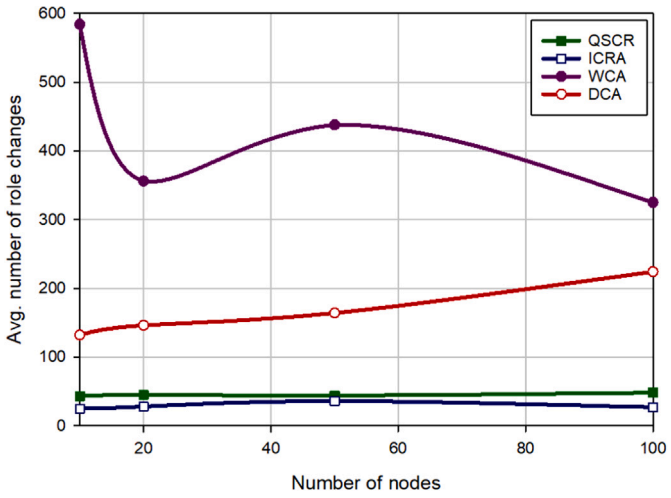

Fig. 10. Evaluation of cluster stability in the third scenario.

6.2. Cluster stability

Figs. 8, 9, and 10 show average changes in the role of UAVs in clusters for different methods in scenarios one, two, and three, respectively. In the first scenario, the average changes in the role of UAVs in clusters for QSCR, ICRA, WCA, and DCA are 38, 20, 276, and 124, respectively. In the second scenario, QSCR, ICRA, WCA, and DCA have experienced on average 51, 15, 352, and 175 role changes in UAVs, respectively. In the third scenario, the average role changes of UAVs for QSCR, ICRA, WCA, and DCA are 45, 29, 425, and 166, respectively. In general, in all scenarios, ICRA has the least changes in the role of UAVs in clusters and works better than QSCR. This means that ICRA has created a more stable topology in FANET. This is because the Q-learning-based clustering adjustment algorithm in ICRA has considered the number of changes in the role of UAVs in the reward function. As a result, ICRA adjusts weight coefficients related to clustering parameters based on cluster stability. However, in QSCR, the reward function is dependent on three scales, namely the balance of energy consumption, the number of isolated clusters, and the distribution of cluster leaders in the network. The last point in this experiment is that QSCR and ICRA are less sensitive to the number of UAVs in the network, but the performance of WCA and DCA varies based on the number of nodes. This is mainly because these methods use constant weight coefficients throughout the simulation period. However, ICRA and QSCR work well due to the use of dynamic weight coefficients.

6.3. Network lifetime

Here, network lifespan is shown for different methods in Fig. 11 (Scenario 1), Fig. 12 (Scenario 2), and Fig. 13 (Scenario 3). In all these scenarios, QSCR has the best network lifespan than other methods. In Fig. 11, i.e. scenario one, QSCR extends network lifespan by 4.06%, 81.66%, and 15.67% compared to ICRA, WCA, and DCA, respectively. Fig. 12, the second scenario, network lifetime in QSCR is 5.95%, 19.96%, and 11.27% higher than ICRA, WCA, and DCA, respectively. In addition, in Fig. 13, the third scenario, QSCR increases network lifespan by 3.5%, 25.27%, and 10.91% compared to ICRA, WCA, and DCA, respectively. This is because the proposed QL-based clustering adjustment mechanism uses a reward function based on three scales, namely energy consumption balance, number of isolated clusters, and distribution of cluster leaders. To balance energy consumption, QSCR chooses weight coefficients in such a way that minimizes the energy consumption of UAVs and selects high-energy nodes as cluster leaders. Moreover, paying attention to the number of isolated clusters in QSCR helps the successful performance of the clustering process. On the other hand, QSCR is looking for weight coefficients that improve the distribution of cluster leaders in the network. For this purpose, QSCR selects UAVs close to the cluster center as the cluster leader to reduce the average distance between cluster members and its cluster leader, and secondly, the distance between cluster leaders is high so that they are well distributed in all network areas. This also has a positive effect on network lifespan. Another point is that the residual energy is a merit parameter when selecting cluster leaders. Therefore, low-energy UAVs cannot be selected as CHs. This improves network lifetime in QSCR. Fig. 14 evaluates the remaining energy of nodes based on the number of UAVs. In this figure, QSCR improves the residual energy of nodes by 7.86%, 12.04%, and 41.13% compared to ICRA, WCA, and DCA, respectively.

6.4. QoS requirements

In Fig. 15, the end-to-end delay in different methods is evaluated based on the number of nodes in the first scenario. QSCR, ICRA, and DCA are not sensitive to the number of UAVs in FANET, but WCA is drastically sensitive to the UAV density in the network. As shown in Fig. 15, ICRA has less delay than QSCR (approximately 30.95%). However, delay in the proposed method is about 76.76% and 9.32% lower than WCA and DCA, respectively. ICRA works better than QSCR.

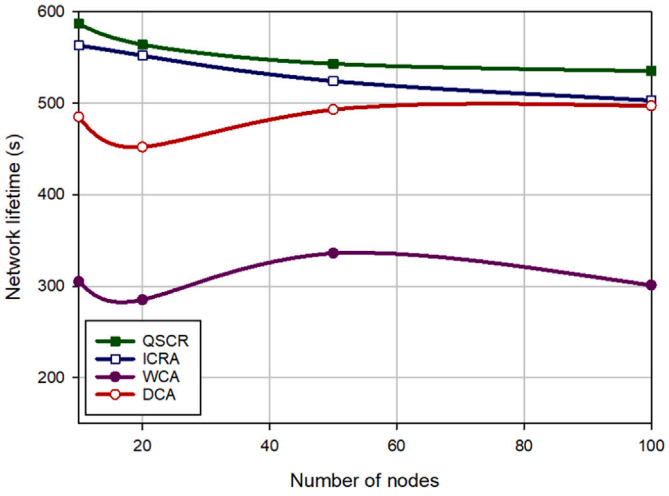

Fig. 11. Evaluation of network lifespan in the first scenario.

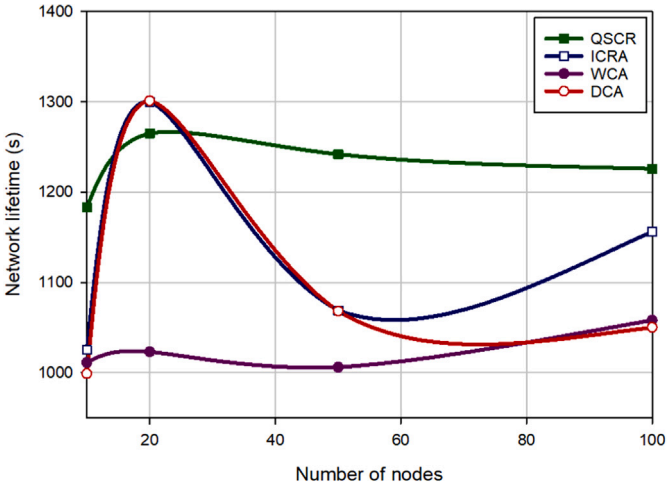

Fig. 12. Evaluation of network lifespan in the second scenario.

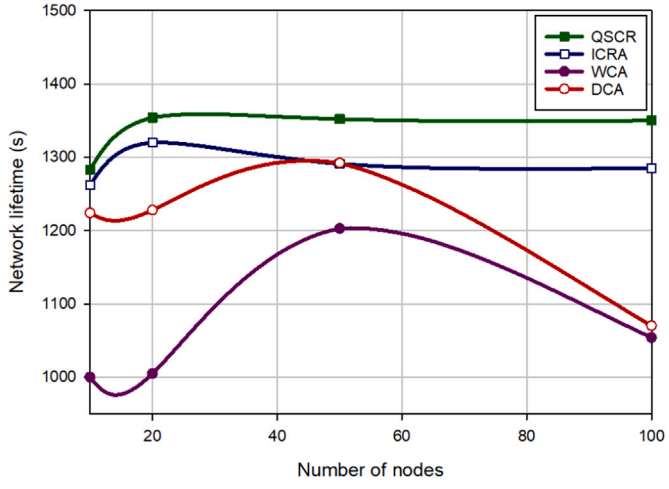

Fig. 13. Evaluation of network lifespan in the third scenario.

This has several reasons: First, the hello propagation period is calculated in QSCR dynamically to increase the accuracy of the clustering and routing in the network, but this process is time-consuming, while

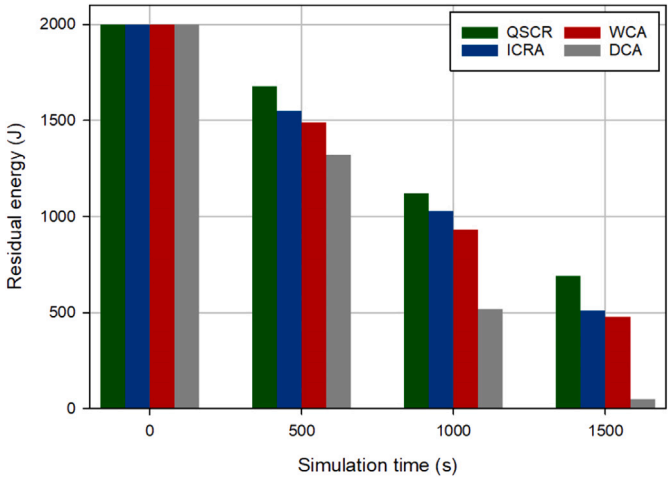

Fig. 14. Comparison of residual energy in different methods in the first scenario.

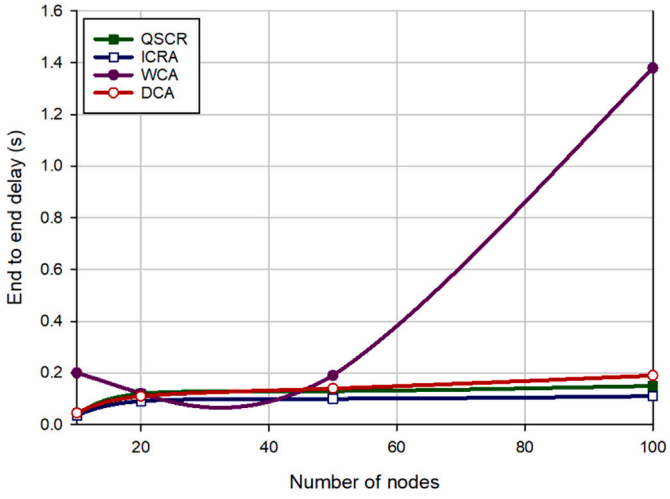

Fig. 15. Delay in different methods in the first scenario.

ICRA has a fixed broadcast time interval. Secondly, in QSCR, the learning parameters, namely discount factor and learning rate, change dynamically and according to network conditions. This increases the adaptability of QSCR to FANET, but its calculations are also time-consuming. However, ICRA considers fixed learning parameters in the learning process.

Fig. 16 shows the number of isolated clusters in different methods for the second scenario. Based on this figure, QSCR has the least number of isolated clusters in the network. It improves this parameter by 22.58%, 55.27%, and 47.83% compared to ICRA, WCA, and DCA, respectively. The main reason for this issue is that QSCR considers the number of isolated clusters when designing the reward function in the Q-learning model. According to the suggested learning model, QSCR adjusts weight coefficients in such a way that minimizes the number of isolated clusters in FANET. This has a positive effect on reducing delay and energy consumption in the suggested method.

Fig. 17 shows the performance of different methods in terms of packet delivery rate in the third scenario. According to this figure, QSCR has the best PDR and increases the evaluation scale by 17.88%, 27.19%, and 71.11% compared to ICRA, WCA, and DCA, respectively. This has various reasons: (1) adjusting the hello propagation time interval in each cluster in a dynamic and adaptive manner, (2) calculating the learning parameters dynamically, and (3) attention to the distribution of cluster leaders in the network when designing the reward function in the proposed learning model.

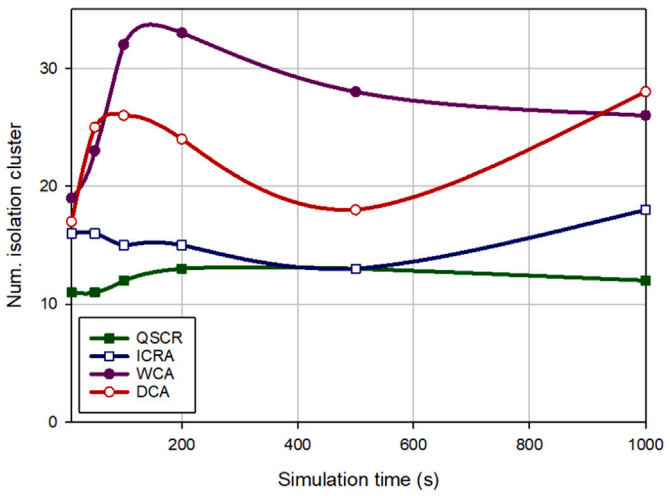

Fig. 16. The number of isolated clusters in the second scenario.

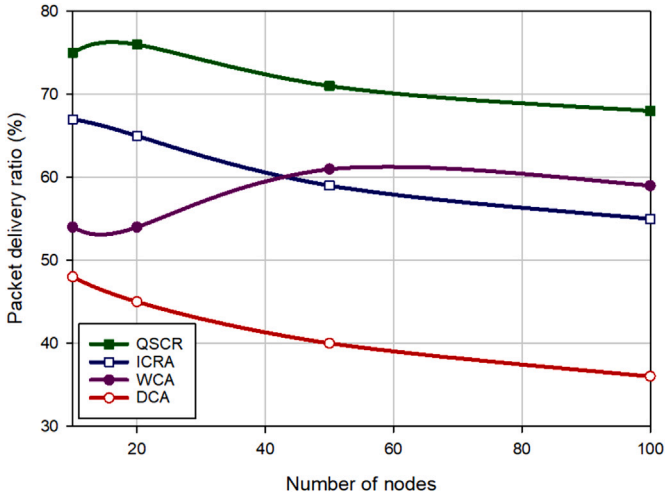

Fig. 17. Packet delivery rate in the third scenario.

6.5. Communication overhead

Here, the communication overhead of different cluster-based routing methods, namely QSCR, ICRA, DCA, and WCA is discussed. These methods use the exchange of control messages between drones to form a clustered network topology. These control messages have a high effect on the energy consumption of UAVs in FANET. To evaluate the communication overhead, the number of control messages received and sent by each drone is calculated. When a clustering method produces a large number of control messages, its communication overhead will be high. As a result, the energy consumption of drones will increase in this clustering method. For computing this parameter, the following three hypotheses are considered.

- The total number of flying nodes in the network is equal to n.
- The total number of cluster leaders in the network is equivalent to m.
- The number of neighbors of the flying node U_i is N_i.

Now, in WCA, the communication overhead is calculated according to the following steps. Then, this parameter is inserted into Table 5.

- In the neighbor discovery phase, U_i periodically broadcasts a beacon message to N_i neighboring UAVs and receives N_i beacon messages from these neighboring nodes. Therefore, the control overhead is equal to $1 + N_i$ in this phase.
- In the cluster head selection phase, U_i computes a combined weight value. Then, it inserts this information into a message and sends this message to N_i neighboring nodes. Also, in this step, U_i obtains N_i combined weight messages from its neighboring nodes. Therefore, the control overhead is $1 + N_i$ in this phase.
- If U_i has the highest combined weight value compared to other neighboring nodes, it accepts the role of cluster head. In this case, U_i sends a CH advertisement message to its neighbors. Then, in the worst case, it receives N_i membership request messages from its neighbors and sends N_i acknowledgment messages to these neighboring nodes. Therefore, the control overhead of CH is equal to $1 + 2N_i$ at this step.
- If U_i does not have the maximum combined weight value compared to other neighboring nodes, then it accepts the role of cluster member. In this case, U_i receives a CH advertisement message from the CH. Then, it sends a membership request message to its CH and receives an acknowledgment message from it. Therefore, the control overhead of CM is 3.
- In the cluster update process, each CH sends a beacon message to its cluster members and receives N_i acknowledgment messages from them. Each cluster member node also sends a beacon message to its CH and receives an acknowledgment message from it. Therefore, the communication overhead of each CH and that of each CM are $1 + N_i$ and 2 in the cluster update process, respectively.
- In the intra-cluster routing process, the CH receives N_i data packets from its cluster members at the worst case. This means that its control overhead is N_i. Additionally, the communication overhead of each cluster member is 1 at this step.
- In the inter-cluster routing process, the CH applies a route discovery process and broadcasts the route request packet (RREQ) in the network to find a suitable route to the destination. In the worst case, this node sends a RREQ message, receives N_i RREQs from its neighbors, and rebroadcasts N_i RREQs in the network. In addition, this node receives a route reply packet (RREP). Hence, the inter-cluster routing overhead of each CH is $2 + 2N_i$ at the worst case.

In DCA, the communication overhead is computed according to the following steps. It is presented in Table 5.

- In the bootstrapping phase, U_i shares its identifier with its neighboring UAVs. Then, the UAV with the lowest ID plays the role of the initial cluster head. Hence, the communication overhead is equal to $1 + N_i$ in this phase.
- In the cluster maintenance phase, U_i transmits its status information to its neighboring UAVs and receives N_i status messages from them. Therefore, its overhead is $1 + N_i$.
- In the CH selection process, each UAV obtains its score, transfers a full clustering control packet (FCP) to its neighbors and receives N_i FCP messages from them. Thus, the control overhead of each node is equivalent $1 + N_i$ at this step.
- If U_i has the highest score among its neighbors, then it accepts the role of cluster head. In this case, U_i forwards a CH advertisement message and a core clustering control message (CCP) to its cluster members and receives N_i join request messages from cluster member nodes. Then, it sends N_i cluster announcement messages (CAPs) to its cluster member nodes. Therefore, the control overhead of each CH is equal to $2 + 2N_i$ at this step.
- If U_i does not have the highest score among its neighbors, then it will accept the role of cluster member. In this case, U_i receives a CH advertisement message from the cluster head and sends a join request message to it. Finally, this cluster member receives a CAP message from its CH. Therefore, the control overhead of this cluster member node is 3.

Table 5
Control overhead in different schemes.

Scheme	Node type	Clustering overhead	Routing overhead	Total
WCA	Cluster head node	$3(1+N_i)+(1+2N_i)$	$N_i+(2+2N_i)$	$8N_i+6$
	Cluster member node	$2(1+N_i)+5$	1	$2N_i+8$
DCA	Cluster head node	$3(1+N_i)+2(2+2N_i)$	$N_i+(2+2N_i)$	$10N_i+9$
	Cluster member node	$3(1+N_i)+(2+2N_i)+3$	1	$5N_i+9$
ICRA	Cluster head node	$3(1+N_i)+(1+2N_i)+2$	N_i+2	$6N_i+8$
	Cluster member node	$2(1+N_i)+7$	1	$2N_i+10$
	Gateway node	$2(1+N_i)+7$	3	$2N_i+12$
QSCR	Cluster head node	$3(1+N_i)+(1+3N_i)+2$	N_i+2	$7N_i+8$
	Cluster member node	$2(1+N_i)+7$	1	$2N_i+10$
	Gateway node	$2(1+N_i)+10$	3	$2N_i+15$

- In the cluster update process, two FCP and CCP control packets are periodically exchanged between CHs and their cluster members to update the clusters and their role in the network. In this case, the clustering overhead of each node (CH or CM) is equivalent to $2+2N_i$.
- In the intra-cluster routing process, the CH node receives N_i data packets from its cluster members at the worst case. Hence, its control overhead is N_i at this step. In addition, the communication overhead of each cluster member is equal to 1.
- In DCA, the inter-cluster routing approach is inspired by AODV. Hence, each CH uses a route discovery process and broadcasts the RREQ message in the network to find a suitable route to the destination. In the worst case, this CH sends a RREQ message, obtain N_i RREQs from its neighbors, and rebroadcasts N_i RREQs in the network. Additionally, this node receives a RREP message. Thus, the inter-cluster routing overhead of each CH is $2+2N_i$ at the worst case.

In ICRA, the communication overhead is calculated in the following steps. This parameter is recorded in Table 5.

- In the neighbor finding phase, U_i periodically broadcasts its information to N_i neighboring UAVs through the hello message and receives N_i hello messages from these neighboring nodes. Hence, the control overhead is equal to $1+N_i$ at this step.
- In the CH selection step, U_i calculates a utility value. Then, it broadcasts this value to N_i neighbors through a utility message. In addition, U_i obtains N_i utility messages from its neighboring nodes. Therefore, the control overhead is equivalent to $1+N_i$ at this step.
- If U_i has the highest utility among other neighboring nodes, then it accepts the role of CH. In this case, U_i transmits a CH declaration message to its neighbors. Then, it receives N_i join cluster request messages from its neighbors and sends N_i join cluster response messages to these neighboring nodes. Therefore, the control overhead of each CH is equal to $1+2N_i$ at this step.
- If U_i does not have the highest utility value among the neighboring nodes, then it accepts the role of cluster member. In this case, U_i receives a CH declaration message from the CH node. Then, it sends a join cluster request message to the CH and receives a join cluster response message from it. Therefore, the control overhead of each cluster member is 3 at this step.
- In the cluster maintenance process, the CH node sends a hello message to its cluster members and receives N_i hello messages from them. Hence, the communication overhead of each CH is equal to $1+N_i$. Furthermore, each cluster member sends a hello message to its CH and receives a hello message from it. Therefore, the communication overhead of each cluster member node is 2 at this step.
- In the Q-learning-based clustering adjustment process, each node sends its status information (i.e. position, speed, etc.) to GCS and receives weight coefficients related to utility parameters from it. As a result, the control overhead of each node is 2 in this process.
- In the intra-cluster routing process, the CH node receives N_i data packets from its cluster members at the worst case. This means that its control overhead is equal to N_i at this step. Also, the communication overhead of each cluster member is 1.
- In the inter-cluster routing process, the CH node (or gateway node) receives a data packet and sends it to the closest neighboring CH node (neighboring gateway node) to the desired CH. Therefore, the inter-cluster routing overhead of each CH (or gateway node) is equivalent to 2.

In QSCR, the communication overhead is calculated according to the following steps. This parameter is presented in Table 5.

- In the neighbor discovery phase, U_i periodically broadcasts its information to N_i neighboring nodes through the hello message and receives N_i hello messages from these neighboring nodes. Here, there is an important point: QSCR defines the hello broadcast time based on the velocity of the CH node and the average velocity similarity of cluster member nodes in each cluster. Therefore, QSCR has a better adaptability to FANET. Overall, control overhead of each node is equal to $1+N_i$ in this phase.
- In the CH selection phase, U_i calculates a merit value. Then, it sends this value to its neighbors through a merit message. In addition, at this step, U_i receives N_i merit messages from its neighboring nodes. Thus, the control overhead is equal to $1+N_i$ at this step.
- If U_i has the highest merit value among other neighboring nodes, then it accepts the role of cluster head. In this case, U_i transmits a leader message to its neighbors. Then, it receives N_i connection messages from its neighbors and sends N_i accept messages to these neighboring nodes. It also receives N_i gateway messages from gateway nodes in its cluster at the worst case. Hence, the control overhead of each CH is equal to $1+3N_i$ at this step.
- If U_i does not have the maximum merit value among the neighboring nodes, then it plays the role of cluster member. In this case, U_i receives a leader message from the CH node. Then, it sends a connection message to the CH and receives an accept message from it. Therefore, the control overhead of each cluster member is 3. If this cluster member is a gateway node, it also sends a gateway message to the CH. Therefore, the control overhead of each gateway node is 4.
- In the cluster support process, the CH node sends a hello message to its cluster members and receives N_i hello messages from them. Therefore, the communication overhead of each cluster head is equal to $1+N_i$. Moreover, the cluster member node sends a hello message to its CH and receives a hello message from it. Therefore, the communication overhead of each cluster member is 2 at this step. Also, gateway nodes, in addition to controlling their connection to its CH, must check its connection with other gateway nodes. Therefore, their communication overhead is 4 in this process.
- In the smart and dynamic adjustment process of clustering parameters, each node sends its status information (position, speed,

etc.) to GCS and receives weight coefficients related to the merit parameters from it. The control overhead of each node is 2 in this process.
- In the intra-cluster routing process, the CH node receives N_i data packets from its cluster members at the worst case. This means that its control overhead is N_i in this process. Also, the communication overhead of each cluster member is 1.
- In the inter-cluster routing process, the CH node (or gateway node) receives a data packet and sends it to the closest neighboring CH node (neighboring gateway node) to the desired CH. Therefore, the inter-cluster routing overhead of each CH (or gateway node) is equivalent to 2.

7. Conclusion

In this paper, a Q-learning-based smart clustering routing method (QSCR) was proposed for flying ad hoc networks. In this scheme, the cluster leader determines the hello propagation period in the cluster based on the average similarity between its speed vector and cluster members' speed. Then, the adaptive clustering process is carried out to construct network topology. In this operation, cluster leaders are selected based on the weighted sum of the five merit parameters, namely residual energy, centrality, neighbor degree, speed similarity, and link validity time. Then, the weight coefficient of each merit parameter is adjusted according to a Q-learning model. In this process, QSCR determines the learning rate based on cluster stability. This parameter defines the ability of UAVs to maintain their role in the cluster. QSCR calculates the discount coefficient based on speed similarity. Also, this learning process designs a reward function based on three scales, namely the balance of energy consumption, the number of isolated clusters, and the distribution of cluster leaders in the network. Finally, the inter-cluster routing process was carried out using a greedy forwarding technique. In the last step, QSCR was implemented on NS2, and its results were compared with three methods, namely ICRA, WCA, and DCA in terms of different scales, namely cluster construction time, cluster stability, network lifespan, and different QoS requirements. These experiments show that QSCR has a faster cluster construction process than other methods, but its stability is less than ICRA. QSCR has more delay than ICRA. However, QSCR has good energy efficiency and greatly improves network lifespan. In the routing process, QSCR has a very good performance in terms of the packet delivery rate. Also, the number of isolated clusters in the proposed method is much less than that in other methods.

In future research directions, QSCR must be modified to solve its weaknesses. These issues can be solved using machine learning techniques (ML) and swarm intelligence (SI). For example, some filtering techniques can be applied to reduce the state and action spaces in the learning process to accelerate the CH selection process. Also, deep reinforcement learning strategies can be used to speed up the convergence rate and reduce delay in the routing process. On the other hand, the Q-learning algorithm is quite sensitive to some key parameters, including the learning rate, the discount factor, and the initial conditions. Hence, providing a sensitivity analysis to adjust the learning rate, discount factor, and initial conditions can significantly affect the performance of Q-learning algorithms. We recommend these areas for future in-depth investigations to further refine and optimize Q-learning applications in FANETs. Also, analysis of the convergence and feasibility of the proposed algorithm based on Q-learning is a crucial factor in the practical implementation of any Q-learning-based routing protocol. We suggest a comprehensive analysis of the convergence and feasibility of the proposed algorithm in a dynamic environment such as FANET as a promising direction for future research endeavors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors extend their appreciation to the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project no. (IFKSUOR3–596–6).

References

Abdulhae, O.T., Mandeep, J.S., Islam, M., 2022. Cluster-based routing protocols for flying ad hoc networks (FANETs). IEEE Access 10, 32981–33004. http://dx.doi.org/10.1109/ACCESS.2022.3161446.

Adams, S., Cody, T., Beling, P.A., 2022. A survey of inverse reinforcement learning. Artif. Intell. Rev. 55 (6), 4307–4346. http://dx.doi.org/10.1007/s10462-021-10108-x.

Alam, M.M., Arafat, M.Y., Moh, S., Shen, J., 2022. Topology control algorithms in multi-unmanned aerial vehicle networks: An extensive survey. J. Netw. Comput. Appl. 207, 103495. http://dx.doi.org/10.1016/j.jnca.2022.103495.

Alam, M.M., Moh, S., 2022. Joint topology control and routing in a UAV swarm for crowd surveillance. J. Netw. Comput. Appl. 204, 103427. http://dx.doi.org/10.1016/j.jnca.2022.103427.

Alam, M.M., Moh, S., 2023. Q-learning-based routing inspired by adaptive flocking control for collaborative unmanned aerial vehicle swarms. Veh. Commun. 40, 100572. http://dx.doi.org/10.1016/j.vehcom.2023.100572.

Almeida, E.N., Coelho, A., Ruela, J., Campos, R., Ricardo, M., 2021. Joint traffic-aware UAV placement and predictive routing for aerial networks. Ad Hoc Netw. 118, 102525. http://dx.doi.org/10.1016/j.adhoc.2021.102525.

Alzahrani, B., Oubbati, O.S., Barnawi, A., Atiquzzaman, M., Alghazzawi, D., 2020. UAV assistance paradigm: State-of-the-art in applications and challenges. J. Netw. Comput. Appl. 166, 102706. http://dx.doi.org/10.1016/j.jnca.2020.102706.

Arafat, M.Y., Moh, S., 2019. Localization and clustering based on swarm intelligence in UAV networks for emergency communications. IEEE Internet Things J. 6 (5), 8958–8976. http://dx.doi.org/10.1109/JIOT.2019.2925567.

Arafat, M.Y., Moh, S., 2021. A Q-learning-based topology-aware routing protocol for flying ad hoc networks. IEEE Internet Things J. 9 (3), 1985–2000. http://dx.doi.org/10.1109/JIOT.2021.3089759.

Chatterjee, M., Das, S.K., Turgut, D., 2002. WCA: A weighted clustering algorithm for mobile ad hoc networks. Cluster Comput. 5, 193–204. http://dx.doi.org/10.1023/A:1013941929408.

Cheng, N., Wu, S., Wang, X., Yin, Z., Li, C., Chen, W., Chen, F., 2023. AI for UAV-assisted IoT applications: A comprehensive review. IEEE Internet Things J. http://dx.doi.org/10.1109/JIOT.2023.3268316.

Cui, Y., Zhang, Q., Feng, Z., Wei, Z., Shi, C., Yang, H., 2022. Topology-aware resilient routing protocol for FANETs: An adaptive Q-learning approach. IEEE Internet Things J. 9 (19), 18632–18649. http://dx.doi.org/10.1109/JIOT.2022.3162849.

da Costa, L.A.L., Kunst, R., de Freitas, E.P., 2021. Q-FANET: Improved Q-learning based routing protocol for FANETs. Comput. Netw. 198, 108379. http://dx.doi.org/10.1016/j.comnet.2021.108379.

Darabkh, K.A., Al-Akhras, M., Zomot, J.N., Atiquzzaman, M., 2022. RPL routing protocol over IoT: A comprehensive survey, recent advances, insights, bibliometric analysis, recommendations, and future directions. J. Netw. Comput. Appl. 207, 103476. http://dx.doi.org/10.1016/j.jnca.2022.103476.

Dhall, R., Dhongdi, S., 2023. Review of protocol stack development of flying ad-hoc networks for disaster monitoring applications. Arch. Comput. Methods Eng. 30 (1), 37–68. http://dx.doi.org/10.1007/s11831-022-09791-y.

Elallid, B.B., Benamar, N., Hafid, A.S., Rachidi, T., Mrani, N., 2022. A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving. J. King Saud Univ.-Comput. Inf. Sci. 34 (9), 7366–7390. http://dx.doi.org/10.1016/j.jksuci.2022.03.013.

Ergenç, D., Eksert, L., Onur, E., 2019. Dependability-based clustering in mobile ad-hoc networks. Ad Hoc Netw. 93, 101926. http://dx.doi.org/10.1016/j.adhoc.2019.101926.

Fanian, F., Rafsanjani, M.K., 2019. Cluster-based routing protocols in wireless sensor networks: A survey based on methodology. J. Netw. Comput. Appl. 142, 111–142. http://dx.doi.org/10.1016/j.jnca.2019.04.021.

Ganesh, A.H., Xu, B., 2022. A review of reinforcement learning based energy management systems for electrified powertrains: Progress, challenge, and potential solution. Renew. Sustain. Energy Rev. 154, 111833. http://dx.doi.org/10.1016/j.rser.2021.111833.

Gharib, M., Afghah, F., Bentley, E.S., 2022. LB-OPAR: Load balanced optimized predictive and adaptive routing for cooperative UAV networks. Ad Hoc Netw. 132, 102878. http://dx.doi.org/10.1016/j.adhoc.2022.102878.

Guo, J., Gao, H., Liu, Z., Huang, F., Zhang, J., Li, X., Ma, J., 2022. ICRA: an intelligent clustering routing approach for UAV ad hoc networks. IEEE Trans. Intell. Transp. Syst. 24 (2), 2447–2460. http://dx.doi.org/10.1109/TITS.2022.3145857.

Hadi, H.J., Cao, Y., Nisa, K.U., Jamil, A.M., Ni, Q., 2023. A comprehensive survey on security, privacy issues and emerging defence technologies for UAVs. J. Netw. Comput. Appl. 213, 103607. http://dx.doi.org/10.1016/j.jnca.2023.103607.

Hosseinzadeh, M., Ahmed, O.H., Lansky, J., Mildeova, S., Yousefpoor, M.S., Yousefpoor, E., Yoo, J., Tightiz, L., Rahmani, A.M., 2023a. A cluster-tree-based trusted routing algorithm using Grasshopper Optimization Algorithm (GOA) in Wireless Sensor Networks (WSNs). Plos one 18 (9), e0289173. http://dx.doi.org/10.1371/journal.pone.0289173.

Hosseinzadeh, M., Ali, S., Ionescu-Feleaga, L., Ionescu, B.S., Yousefpoor, M.S., Yousefpoor, E., Ahmed, O.H., Rahmani, A.M., Mehmood, A., 2023b. A novel Q-learning-based routing scheme using an intelligent filtering algorithm for flying ad hoc networks (FANETs). J. King Saud Univ.-Comput. Inf. Sci. 101817. http://dx.doi.org/10.1016/j.jksuci.2023.101817.

Hosseinzadeh, M., Mohammed, A.H., Alenizi, F.A., Malik, M.H., Yousefpoor, E., Yousefpoor, M.S., Ahmed, O.H., Rahmani, A.M., Tightiz, L., 2023c. A novel fuzzy trust-based secure routing scheme in flying ad hoc networks. Veh. Commun. 100665. http://dx.doi.org/10.1016/j.vehcom.2023.100665.

Hosseinzadeh, M., Tanveer, J., Ionescu-Feleaga, L., Ionescu, B.S., Yousefpoor, M.S., Yousefpoor, E., Ahmed, O.H., Rahmani, A.M., Mehmood, A., 2023d. A greedy perimeter stateless routing method based on a position prediction mechanism for flying ad hoc networks. J. King Saud Univ.-Comput. Inf. Sci. 35 (8), 101712. http://dx.doi.org/10.1016/j.jksuci.2023.101712.

Hosseinzadeh, M., Yoo, J., Ali, S., Lansky, J., Mildeova, S., Yousefpoor, M.S., Ahmed, O.H., Rahmani, A.M., Tightiz, L., 2023e. A cluster-based trusted routing method using fire hawk optimizer (FHO) in wireless sensor networks (WSNs). Sci. Rep. 13 (1), 13046. http://dx.doi.org/10.1038/s41598-023-40273-8.

Jawhar, I., Mohamed, N., Al-Jaroodi, J., Agrawal, D.P., Zhang, S., 2017. Communication and networking of UAV-based systems: Classification and associated architectures. J. Netw. Comput. Appl. 84, 93–108. http://dx.doi.org/10.1016/j.jnca.2017.02.008.

Jin, H., Jin, X., Zhou, Y., Guo, P., Ren, J., Yao, J., Zhang, S., 2023. A survey of energy efficient methods for UAV communication. Veh. Commun. 100594. http://dx.doi.org/10.1016/j.vehcom.2023.100594.

Khan, A., Aftab, F., Zhang, Z., 2019. Self-organization based clustering scheme for FANETs using Glowworm Swarm Optimization. Phys. Commun. 36, 100769. http://dx.doi.org/10.1016/j.phycom.2019.100769.

Khan, M.U., Hosseinzadeh, M., Mosavi, A., 2022. An intersection-based routing scheme using Q-learning in vehicular Ad Hoc networks for traffic management in the intelligent transportation system. Mathematics 10 (20), 3731. http://dx.doi.org/10.3390/math10203731.

Khan, M.F., Yau, K.L.A., Noor, R.M., Imran, M.A., 2020. Survey and taxonomy of clustering algorithms in 5G. J. Netw. Comput. Appl. 154, 102539. http://dx.doi.org/10.1016/j.jnca.2020.102539.

Khedr, A.M., Salim, A., PV, Osamy, W., 2023. MWCRSF: Mobility-based weighted cluster routing scheme for FANETs. Veh. Commun. 41, 100603. http://dx.doi.org/10.1016/j.vehcom.2023.100603.

Lansky, J., Ali, S., Rahmani, A.M., Yousefpoor, M.S., Yousefpoor, E., Khan, F., Hosseinzadeh, M., 2022a. Reinforcement learning-based routing protocols in flying ad hoc networks (FANET): A review. Mathematics 10 (16), 3017. http://dx.doi.org/10.3390/math10163017.

Lansky, J., Rahmani, A.M., Hosseinzadeh, M., 2022b. Reinforcement learning-based routing protocols in vehicular ad hoc networks for intelligent transport system (ITS): A survey. Mathematics 10 (24), 4673. http://dx.doi.org/10.3390/math10244673.

Lansky, J., Rahmani, A.M., Malik, M.H., Yousefpoor, E., Yousefpoor, M.S., Khan, M.U., Hosseinzadeh, M., 2023. An energy-aware routing method using firefly algorithm for flying ad hoc networks. Sci. Rep. 13 (1), 1323. http://dx.doi.org/10.1038/s41598-023-27567-7.

Lansky, J., Rahmani, A.M., Zandavi, S.M., Chung, V., Yousefpoor, E., Yousefpoor, M.S., Khan, F., Hosseinzadeh, M., 2022c. A Q-learning-based routing scheme for smart air quality monitoring system using flying ad hoc networks. Sci. Rep. 12 (1), 20184. http://dx.doi.org/10.1038/s41598-022-20353-x.

Lee, S.W., Ali, S., Yousefpoor, M.S., Yousefpoor, E., Lalbakhsh, P., Javaheri, D., Rahmani, A.M., Hosseinzadeh, M., 2021. An energy-aware and predictive fuzzy logic-based routing scheme in flying ad hoc networks (fanets). IEEE Access 9, 129977–130005. http://dx.doi.org/10.1109/ACCESS.2021.3111444.

Liu, Y., Xie, J., Xing, C., Xie, S., 2023. Topology construction and topology adjustment in flying Ad hoc networks for relay transmission. Comput. Netw. 228, 109753. http://dx.doi.org/10.1016/j.comnet.2023.109753.

Mansoor, N., Hossain, M.I., Rozario, A., Zareei, M., Arreola, A.R., 2023. A fresh look at routing protocols in unmanned aerial vehicular networks: A survey. IEEE Access http://dx.doi.org/10.1109/ACCESS.2023.3290871.

Messaoudi, K., Oubbati, O.S., Rachedi, A., Lakas, A., Bendouma, T., Chaib, N., 2023. A survey of UAV-based data collection: Challenges, solutions and future perspectives. J. Netw. Comput. Appl. 103670. http://dx.doi.org/10.1016/j.jnca.2023.103670.

Prudencio, R.F., Maximo, M.R., Colombini, E.L., 2023. A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Trans. Neural Netw. Learn. Syst. http://dx.doi.org/10.1109/TNNLS.2023.3250269.

Rahmani, A.M., Ali, S., Malik, M.H., Yousefpoor, E., Yousefpoor, M.S., Mousavi, A., Khan, F., Hosseinzadeh, M., 2022a. An energy-aware and Q-learning-based area coverage for oil pipeline monitoring systems using sensors and Internet of Things. Sci. Rep. 12 (1), 9638. http://dx.doi.org/10.1038/s41598-022-12181-w.

Rahmani, A.M., Ali, S., Yousefpoor, E., Yousefpoor, M.S., Javaheri, D., Lalbakhsh, P., Ahmed, O.H., Hosseinzadeh, M., Lee, S.W., 2022b. OLSR+: A new routing method based on fuzzy logic in flying ad-hoc networks (FANETs). Veh. Commun. 36, 100489. http://dx.doi.org/10.1016/j.vehcom.2022.100489.

Rahmani, A.M., Naqvi, R.A., Yousefpoor, E., Yousefpoor, M.S., Ahmed, O.H., Hosseinzadeh, M., Siddique, K., 2022c. A Q-learning and fuzzy logic-based hierarchical routing scheme in the intelligent transportation system for smart cities. Mathematics 10 (22), 4192. http://dx.doi.org/10.3390/math10224192.

Rovira-Sugranes, A., Afghah, F., Qu, J., Razi, A., 2021. Fully-echoed Q-routing with simulated annealing inference for flying adhoc networks. IEEE Trans. Netw. Sci. Eng. 8 (3), 2223–2234. http://dx.doi.org/10.1109/TNSE.2021.3085514.

Shahzadi, R., Ali, M., Khan, H.Z., Naeem, M., 2021. UAV assisted 5G and beyond wireless networks: A survey. J. Netw. Comput. Appl. 189, 103114. http://dx.doi.org/10.1016/j.jnca.2021.103114.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction. MIT Press.

Wang, F., Wang, X., Sun, S., 2022. A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization. Inform. Sci. 602, 298–312. http://dx.doi.org/10.1016/j.ins.2022.04.053.

Yang, J., Sun, K., He, H., Jiang, X., Chen, S., 2022. Dynamic virtual topology aided networking and routing for aeronautical ad-hoc networks. IEEE Trans. Commun. 70 (7), 4702–4716. http://dx.doi.org/10.1109/TCOMM.2022.3177599.

Yousefpoor, M.S., Yousefpoor, E., Barati, H., Barati, A., Movaghar, A., Hosseinzadeh, M., 2021. Secure data aggregation methods and countermeasures against various attacks in wireless sensor networks: A comprehensive review. J. Netw. Comput. Appl. 190, 103118. http://dx.doi.org/10.1016/j.jnca.2021.103118.

Zhang, M., Dong, C., Feng, S., Guan, X., Chen, H., Wu, Q., 2022a. Adaptive 3D routing protocol for flying ad hoc networks based on prediction-driven Q-learning. China Commun. 19 (5), 302–317. http://dx.doi.org/10.23919/JCC.2022.05.005.

Zhang, M., Dong, C., Yang, P., Tao, T., Wu, Q., Quek, T.Q., 2022b. Adaptive routing design for flying ad hoc networks. IEEE Commun. Lett. 26 (6), 1438–1442. http://dx.doi.org/10.1109/LCOMM.2022.3152832.