Full length article

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

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

\textsuperscript{a} Institute of Research and Development, Duy Tan University, Da Nang, Viet Nam
\textsuperscript{b} School of Medicine and Pharmacy, Duy Tan University, Da Nang, Viet Nam
\textsuperscript{c} Department of Computer Science and Engineering, Sejong University, Seoul 05006, Republic of Korea
\textsuperscript{d} Future Technology Research Center, National Yunlin University of Science and Technology, Yunlin, Taiwan
\textsuperscript{e} Department of Computer Engineering, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia
\textsuperscript{f} Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
\textsuperscript{g} Department of Information Technology, University of Human Development, Sulaymaniyah, Iraq
\textsuperscript{h} Pattern Recognition and Machine Learning Lab, Gachon University, 1342 Seongnam-daero, Sujeong-gu, Seongnam 13120, Republic of Korea
\textsuperscript{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...
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; Rivara-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; Abdulhafe 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 to 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 MANETs. 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,
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. \( Q(S, A) \)) to different states and actions. \( Q(S, A) \) estimates how desirable it is to do the action \( A \) in the state \( S \). 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, to go to another state. This process continues until the agent reaches the goal. In this situation, an episode is completed. In QL, \( Q(S, A) \) is estimated based on Eq. (1) to calculate Q-values when interacting with the environment (Ganesha and Xu, 2022; Sutton and Barto, 2018).

\[
Q(S, A) = Q(S, A) + \alpha \left[ R_{t+1} + \gamma \max_{A} Q(S_{t+1}, A) - Q(S, A) \right]
\]  

\( S \) and \( S_{t+1} \) show the current and next states of the agent, respectively. \( A \) indicates the best 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. \( \max Q(S_{t+1}, A) \) indicates the maximum Q-value under the next state. In the following, some learning parameters related to QL are described briefly:

- **Discount factor** (\( \gamma \)): \( \gamma \) 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** (\( \alpha \)): \( \alpha \) shows the knowledge acquired from the environment for updating \( Q(S, A) \).
- **\( \epsilon \)-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 \( \epsilon \).

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 \( r_c \) while it has created an indirect or multi-hop communication with other flying nodes that are out of \( r_c \). In QSCR, \( U_i \) has access to a repositioning 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 \( (C_j) \) so that \( k = 1, 2, \ldots, m \). \( C_j \) includes a cluster leader \( (LU_j) \) and a number of cluster member UAVs \( (MU_j^k) \) so that \( j = 1, 2, \ldots, N_j \). Each \( C_j \), some cluster members close to the cluster border also play the role of inter-cluster gateways \( (GU_j^k) \) 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...
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.

• \(MU_{ij}\): In QSCR, each \(MU_{ij}\) 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.

• \(GU_{ig}\): In QSCR, each \(GU_{ig}\) must perform the tasks of a cluster member node, namely sensing the environment and sending information to \(LU_i\). 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
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, spatial coordinates \( \text{Loc}_i = (x_i', y_i', z_i') \), speed and movement angle \( \text{Mob}_i = (V_i', \theta_i', \phi_i') \), identifier of cluster leader \( (I D_{LU}) \), and the propagation period of \( H_i \) \( (H_T) \). 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 \( I D_{LU} \). Otherwise, it inserts the identifier of its cluster leader in this field.

\[
H_i = \{ ID \| \text{Loc}_i \| \text{Mob}_i \| I D_{LU} \| H_T \}
\]  

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>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_i$</td>
<td>Flying node $i$ in the network</td>
</tr>
<tr>
<td>$n$</td>
<td>Total number of flying nodes in the network</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Communication range of $U_i$</td>
</tr>
<tr>
<td>$N_r$</td>
<td>Number of neighbors of $U_i$ in Table $N_{\text{Neigh}}$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>$i$th cluster</td>
</tr>
<tr>
<td>$m$</td>
<td>Total number of clusters in the network</td>
</tr>
<tr>
<td>$LU_i$</td>
<td>Cluster leader in $C_i$</td>
</tr>
<tr>
<td>$MU_j$</td>
<td>$j$th cluster member in $C_i$</td>
</tr>
<tr>
<td>$N_j$</td>
<td>Total number of cluster members in $C_i$</td>
</tr>
<tr>
<td>$GU_j$</td>
<td>$j$th inter-cluster gateways in $C_i$</td>
</tr>
<tr>
<td>$H_i$</td>
<td>Hello packet related to $U_i$</td>
</tr>
<tr>
<td>$HT_i$</td>
<td>Propagation time interval corresponding to $H_i$</td>
</tr>
<tr>
<td>$ID_i$</td>
<td>Identifier of $U_i$</td>
</tr>
<tr>
<td>$Loc_i = {x_i, y_i, z_i}$</td>
<td>Spatial coordinates of $U_i$</td>
</tr>
<tr>
<td>$Mob_i = {V'_i, \theta'_i, \phi'_i}$</td>
<td>Speed and movement angle of $U_i$</td>
</tr>
<tr>
<td>$T_{\text{table}}^{N_{\text{Neigh}}}$</td>
<td>Neighbor table stored in $U_i$</td>
</tr>
<tr>
<td>$V_{\text{max}}$</td>
<td>Upper boundary of UAVs' speed in the network</td>
</tr>
<tr>
<td>$V_{\text{min}}$</td>
<td>Lower boundary of UAVs' speed in the network</td>
</tr>
<tr>
<td>$m_f$</td>
<td>Merit of $U_i$</td>
</tr>
<tr>
<td>$E_r$</td>
<td>Residual energy of $U_i$</td>
</tr>
<tr>
<td>$E_{\text{initial}}$</td>
<td>Initial energy capacity of UAVs in the network</td>
</tr>
<tr>
<td>$CNR$</td>
<td>Centrality of $U_i$</td>
</tr>
<tr>
<td>$D_{\text{max}}$</td>
<td>Maximum distance between UAVs in the network</td>
</tr>
<tr>
<td>$D_{\text{deg}}$</td>
<td>Neighbor degree of $U_i$</td>
</tr>
<tr>
<td>$\mathbf{SC}_i$</td>
<td>Average similarity between the velocity vector of $U_i$ and those of its neighbors</td>
</tr>
<tr>
<td>$LT_{ij}$</td>
<td>Validity time of link between $U_i$ and $U_j$</td>
</tr>
<tr>
<td>$M_{\text{merg}}$</td>
<td>Merit of $U_i$</td>
</tr>
<tr>
<td>$M_{\text{init}}$</td>
<td>Leader message sent by $LU_k$</td>
</tr>
<tr>
<td>$M_{\text{update}}$</td>
<td>Connection message from $MU_j$ to $LU_k$</td>
</tr>
<tr>
<td>$M_{\text{NCR}}$</td>
<td>Accept message send by $LU_i$</td>
</tr>
<tr>
<td>$M_{\text{initNCR}}$</td>
<td>Gateway message from $GU_j$ to $LU_k$</td>
</tr>
<tr>
<td>$R_{\text{f}}$</td>
<td>Reward function</td>
</tr>
<tr>
<td>$a$</td>
<td>Learning rate</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Discount factor</td>
</tr>
<tr>
<td>$C_{\text{change}}$</td>
<td>Stability value of $C_i$</td>
</tr>
<tr>
<td>$N_{\text{change}}$</td>
<td>Number of role changes in $C_i$</td>
</tr>
</tbody>
</table>

**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

\[
SC_{ij} = \frac{1}{N_j} \sum_{j=1}^{N_j} SC_{ij}^j \tag{7}
\]

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

So that $N_j$ indicates the number of cluster members in $C_k$.

Then, $LU_k$ must obtain $HT_i$ from Eq. (8).

\[
HT_i^j = \begin{cases} 
HT_i^{j,0}, & t = 0 \\
HT_i^{j,1-1} - \frac{\sqrt{V_{LU_k} - V_{\text{min}}}}{V_{\text{max}} - V_{\text{min}}}, & \text{else}
\end{cases} \tag{8}
\]

So that,

\[
HT_i^j = HT_i^{0,0}e^{\gamma t} = \frac{\sqrt{V_{LU_k} - V_{\text{min}}}}{V_{\text{max}} - V_{\text{min}}} \tag{9}
\]

Here, $HT_i^{0,0}$ represents a default value, for example, one second, for broadcasting hello messages in the network.

$V_{LU_k}$ is the velocity length of $LU_k$.

**Algorithm 1 Dynamic neighbor discovery**

**Input:** $U_i$: Flying node $i$ so that $i = 1, 2, \ldots, n$  
$N_j$: Number of neighbors of $U_i$  
$H_i$: Hello message related to $U_i$  
$I_D$: Identifier of $U_i$  
$L_{\text{loc}} = \{x_i, y_i, z_i\}$: Location information of $U_i$  
$Mob_i = \{V'_i, \theta'_i, \phi'_i\}$: Mobility information of $U_i$  
$I_D_{\text{cluster}}$: Identifier of the leader corresponding to $U_i$  
$N_j$: Number of members of the cluster $k$  
$HT_i$: Hello dissemination period

**Output:** $Table^{N_{\text{Neigh}}}$: Neighboring table stored in $U_i$

**Begin**

1: repeat
2: if $N_{\text{change}} \mod HT_i = 0$ then
3: \hspace{1em} $U_i$: Obtain $L_{\text{loc}}$, and $Mob$ from a positioning system (GPS);  
4: \hspace{1em} if $U_i$ plays the role of $LU_k$ then
5: \hspace{2.5em} for $j = 1$ to $N_j$ do
6: \hspace{4em} $U_i$: Obtain the cosine similarity ($SC_{ij}^j$) between the velocity vectors of $U_i$ and $MU_j^k$ from Equation (3);  
7: \hspace{4em} end for
8: \hspace{1em} $U_i$: Calculate the average cosine similarity ($SC_{ij}$) based on Equation (7);  
9: \hspace{1em} $U_i$: Refresh $HT_i$ based on Equation (8);  
10: \hspace{1em} $U_i$: Send $HT_i$ to all $MU_j^k$ in its cluster;  
11: else
12: \hspace{1em} $U_i$: Send a request message to $LU_k$ to obtain updated $HT_i$;  
13: end if
14: \hspace{1em} $U_i$: Insert $I_D$, $L_{\text{loc}}$, $Mob$, $I_D_{\text{cluster}}$, and $HT_i$ into $H_i$ based on Equation (2);  
15: \hspace{1em} $U_i$: Broadcast $H_i$ to the neighboring nodes;  
16: end if
17: for $j = 1$ to $N_j$ do
18: \hspace{1em} if $U_i$ has information about $U_j$ in $Table^{N_{\text{Neigh}}}$ and $U_j$ sends a new $H_i$ to $U_i$ then
19: \hspace{2.5em} $U_i$: Refresh the entry related to $U_j$ based on this $H_i$;  
20: else if $U_j$ has information about $U_i$ in $Table^{N_{\text{Neigh}}}$ and $U_j$ does not send any new $H_i$ to $U_i$ then
21: \hspace{2.5em} $U_i$: Remove the entry related to $U_j$ from $Table^{N_{\text{Neigh}}}$;  
22: else if $U_j$ has not any information about $U_j$ in $Table^{N_{\text{Neigh}}}$ and $U_j$ sends a new $H_i$ to $U_i$ then
23: \hspace{2.5em} $U_i$: Add a new entry related to $U_j$ to $Table^{N_{\text{Neigh}}}$;  
24: \hspace{2.5em} $U_i$: Insert the information of $U_j$ into $Table^{N_{\text{Neigh}}}$ based on this new $H_i$;  
25: end if
26: end for
27: until Simulation time is finished

**End**
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 ($GU_i^k$) 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 ($m_i^c$) to obtain the role of the cluster leader $LU_i$ 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 ($E_i^r$):** 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 $LU_i$ 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, $LU_i$ 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).

$$E_i = \frac{E_{i}^{Residual}}{E_{i}^{Initial}}$$

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

- **Centrality ($CNR_i$):** It is very important to consider the centrality parameter when determining the merit value of $U_i$ for playing the role of $LU_i$. 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 $CNR_i$ that indicates the average distance between $U_i$ and its neighbors, such as $U_{ij}$, in $Table_i^{Neighbor}$.

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

where $Loc_i = (x_i, y_i, z_i)$ and $Loc_j = (x_j, y_j, z_j)$ 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 $CNR_i$ to the interval $[0, 1]$. This causes the same effect of merit parameters with different units on $m_i^c$.

$$CNR_i^{norm} = \frac{CNR_i}{D_{max}}$$

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.

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

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

- **Neighbor degree ($Deg_i$):** This parameter is very important for determining the merit value of $U_i$ and accepting the role of $LU_i$ 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 $[0, 1]$.

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

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 ($SC_i^S$):** 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 $LU_i$ 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 $SC_i^S$. The only point that is important here is that $SC_i^S$ 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 $[0, 1]$.
\[
\frac{SC_{i}}{C_i} = \frac{SC_i + 1}{2} \quad (15)
\]

**Link validity time** ($L_{Ti}$): 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 $C_i$. However, if there are weak connections between $LU_k$ and its cluster members, these connections will be broken rapidly, and consequently, $C_i$ 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).

\[
\left( D_{ij}^{LT} \right)^2 = \left( \frac{x_i - x_j}{\alpha_i} \right) + \left( \frac{y_i - y_j}{\beta_i} \right) + \left( \frac{z_i - z_j}{\rho_i} \right) + \left( \frac{V'_i - V'_j}{\omega_i} \right) LT_{ij}
\]

\[
\left( D_{ij}^{LT} \right)^2 = \sum_{q=1}^{3} \left( \frac{(a_i)^2}{\alpha_i} \right) + \sum_{q=1}^{3} \frac{(\beta_i)^2}{\omega_i} \sum_{q=1}^{3} \left( \frac{(b_i)^2}{\beta_i} \right) \quad (16)
\]

So that ($V'_i, V''_i, V'''_i$) is the velocity vector of $U_i$ obtained through Eqs. (17), (18), and (19). Also, ($V'_i, V''_i, V'''_i$) shows the velocity vector of $U_j$ and is calculated in a similar manner.

\[
V'_i = V'_i \sin \varphi_i \cos \theta_i
\]

\[
V''_i = V'_i \sin \varphi_i \sin \theta_i
\]

\[
V'''_i = V'_i \cos \varphi_i
\]

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

\[
\left( D_{ij}^{LT} \right)^2 = \sum_{q=1}^{3} \left( \frac{(a_i)^2}{\alpha_i} \right) + \sum_{q=1}^{3} \frac{(\beta_i)^2}{\omega_i} \sum_{q=1}^{3} \left( \frac{(b_i)^2}{\beta_i} \right) \quad (20)
\]

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

\[
\left( \sum_{q=1}^{3} \left( \frac{(a_i)^2}{\alpha_i} \right) - (r_i)^2 \right)^2 + \sum_{q=1}^{3} \frac{(\beta_i)^2}{\omega_i} \sum_{q=1}^{3} \left( \frac{(b_i)^2}{\beta_i} \right) = 0 \quad (21)
\]

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

\[
LT_{ij} = \frac{-\sum_{q=1}^{3} 2a_i\beta_i + \sqrt{(\sum_{q=1}^{3} 4a_i\beta_i)^2 - 4(\sum_{q=1}^{3} a_i\beta_i)^2}}{2\sum_{q=1}^{3} \left( \frac{(b_i)^2}{\beta_i} \right)^2}
\]

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

\[
\overline{LT_{ij}} = \frac{1}{N_i} \sum_{j=1}^{N_i} LT_{ij}
\]

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

\[
\overline{LT_{ij}}^{\text{norm}} = \frac{\overline{LT_{ij}}}{\overline{LT_{ij}}_{\text{max}}} \quad (24)
\]

Finally, $U_i$ uses Eq. (25) to calculate $m'_i$.

\[
m'_i = \omega_1 E_i + \omega_2 D_{ij} + \omega_3 SC_i^{\text{norm}} + \omega_4 LT_{ij}^{\text{norm}}
\]

where $\omega_i$ ($i = 1, 2, 3, 4, 5$) represents the weight coefficient that determines the effect of the relevant parameter on $m'_i$. 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 5.3.5.

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 $C_i$, $U_i$ calculates its merit value $m'_i$ in accordance with Eq. (25) in Section 5.2.1 and then broadcasts this information through the merit message $M'_i^{\text{merit}}$ to surrounding nodes in the network. See Fig. 4. $M'_i^{\text{merit}}$ includes $ID_i$, $m'_i$, and a timestamp $T_{\text{merit}}$. Its structure is stated in Eq. (26).

\[
M'_i^{\text{merit}} = \left( ID_i \mid m'_i \mid T_{\text{merit}} \right)
\]

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 $m'_i$ 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 $M'_{U_k}$ to its adjacent nodes. The structure of $M'_{U_k}$ is stated in Eq. (27) and contains $ID_k$, $m'_i$, the role of $U_i$ (i.e. $LU_k$), and a timestamp $T_{LU_k}$.

\[
M'_{LU_k} = \left( ID_k \mid m'_i \mid T_{LU_k} \right)
\]

- **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 $MU_{ij}$ and transfers a connection message $M'_{MU_{ij}}$ to $LU_k$. This message contains $ID_k$, $MU_{ij}$ (role of $U_i$), $ID_{LU_k}$, and a timestamp $T_{MU_{ij}}$, and its structure is presented in Eq. (28).

\[
M'_{MU_{ij}} = \left( ID_k \mid ID_{LU_k} \mid T_{MU_{ij}} \right)
\]

When this message reaches $LU_k$, it sends an accept message $M'_{LU_k}^\text{Accept}$ to $U_i$ and accepts it as $MU_{ij}$ in the cluster $C_i$.
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 (\( ID_{LU} \)) through hello messages to other adjacent nodes, and this information is recorded in \( T_{Neighbor}^{\text{Adjacent}} \). To determine gateway nodes, such as \( GU^k \), suppose that \( MU^k \) belongs to the cluster \( C_j \), and its cluster leader is \( LU_j \). In this case, \( MU^k \) examines its neighbor table, if it finds a neighboring node that belongs to another cluster such as \( C_j \) and its leader cluster is \( LU_j \), then \( MU^k \) is known as a gateway node to connect the cluster \( C_j \) and transmits a gateway message \( M_{Gateway}^{GU^k} \) to \( LU_j \). This message contains the gateway ID (\( ID_{LU} \)), \( ID_{LU_j} \), and timestamp \( T_{Gateway} \). The format of this message is stated in Eq. (30).

\[
M_{Gateway}^{GU^k} = \left( ID_{GU^k} \right) \left( ID_{LU} \right) \left( ID_{LU_j} \right) \left| T_{Gateway} \right)
\]  

(30)
Algorithm 3 Supporting the clustering process

**Input:** Uᵢ: Flying node i so that i = 1, 2, ..., n

<table>
<thead>
<tr>
<th>Table 3: Neighbor table stored in Uᵢ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nᵢ: Number of neighbors of Uᵢ</td>
</tr>
<tr>
<td>Cᵢ: Cluster k so that k = 1, 2, ..., n</td>
</tr>
<tr>
<td>LUYᵢ: Leader of Cᵢ</td>
</tr>
<tr>
<td>MUYᵢ: Cluster member UAV so that j = 1, 2, ..., Nᵢ</td>
</tr>
<tr>
<td>GUᵢ: Gateway UAV in Cᵢ</td>
</tr>
</tbody>
</table>

| Timer: Tᵢ for measuring the network time. |

| RTYᵢ: Hello dissemination time |

| Hᵢ: Hello message related to LUYᵢ |

**Output:** Updating clusters C₁, ..., Cₜ, ..., Cₙ in the network

1. **Begin**
   2. **repeat**
      3. **for** k = 1 to n do
         4. if k mod HTYᵢ = 0 then
            5. LUYᵢ: Adjust a message Hᵢ, based on Equation (29);
            6. LUYᵢ: Broadcast Hᵢ to its cluster member nodes;
         7. if MUᵢ does not receive any Hᵢ from LUYᵢ then
            8. MUᵢ: Adjust a message Mᵢ*, based on Equation (28);
            9. MUᵢ: Send Mᵢ* to the nearest LUYᵢ;
            10. if this new LUYᵢ receives Mᵢ* from MUᵢ then
                11. LUYᵢ: Set a message Mᵢ*, based on Equation (29);
                12. LUYᵢ: Send Mᵢ* to the relevant MUᵢ;
            13. end if
            14. end if
            15. MUᵢ: Adjust a message Hᵢ, based on Equation (2);
            16. MUᵢ: Broadcast Hᵢ to its LUYᵢ;
            17. if LUYᵢ does not receive any Hᵢ from MUᵢ then
                18. LUYᵢ: Remove Mᵢ* from its member list;
            19. end if
            20. end for
            21. end if
            22. end for
            23. if a new Uᵢ is connected to the network then
                24. Uᵢ: Play the role of MUᵢ and adjust a message Mᵢ*, based on Equation (28);
                25. Uᵢ: Send Mᵢ* to the nearest LUYᵢ;
                26. if LUYᵢ receives Mᵢ* from this new node then
                    27. LUYᵢ: Set a message Mᵢ*, based on Equation (29);
                    28. LUYᵢ: Send Mᵢ*, to the relevant MUᵢ;
                29. end if
                30. end if
            31. 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:** LUYᵢ periodically examines its connections with its cluster members through the exchange of hello messages. If LUYᵢ does not receive any hello message from MUᵢ*, then it has gone out of the communication area of LUYᵢ and should be removed from the list of cluster members.

- **Connecting to the cluster leader:** MUᵢ* periodically controls its connection with LUYᵢ through the exchange of hello messages. If MUᵢ* does not receive any hello message from LUYᵢ*, it is clear that LUYᵢ has gone out of the communication area of MUᵢ*. As a result, MUᵢ* must send a connection request to the nearest cluster leader to join a new cluster.

- **Connecting to gateways:** GUᵢ* 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, GUᵢ* must report this issue to its cluster leader.

Table 3

<table>
<thead>
<tr>
<th>Component related to the learning model in QSCR.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Learning component</strong></td>
</tr>
<tr>
<td>Learning issue</td>
</tr>
<tr>
<td>Learning environment</td>
</tr>
<tr>
<td>Learning agent</td>
</tr>
<tr>
<td>State space</td>
</tr>
<tr>
<td>Action space</td>
</tr>
</tbody>
</table>

- **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 (ωᵢ) 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. Uᵢ makes a state message, which includes its five merit parameters, namely remaining energy (Eᵢ), centrality (CNᵢ), neighbor degree (Dᵢ), speed similarity (SCᵢ), and link validity time (LTᵢ), 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 ωᵢ 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 = {ω₁| 1 ≤ i ≤ 5, 0 ≤ ωᵢ ≤ 1 and ∑ωᵢ = 1}. For each action aᵢ, the state space (STᵢ) represents a set of merit values of UAVs so that STᵢ = [m₁, m₂, ..., mᵢ]. Then, GCS examines the effect of action aᵢ 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** (f₁): It evaluates the effect of the selected action (aᵢ = [ω₁, ω₂, ω₃, ω₄, ω₅]) 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, f₁ 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. Additionally, f₁ 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
cluster members in the network, and cluster leaders are selected from high-energy nodes.

\[ f_1 = \ell \left( \frac{1}{\sum_{i=1}^{m} E_{\text{Residual}}(t-1) - E_{\text{Residual}}(t)} \right) + (1 - \ell) \sum_{i=1}^{m} \frac{E_{\text{Residual}}(t)}{\left( \sum_{j \in C_k} E_{\text{Residual}}(t) \right)} \]  

(31)

So that \( n \) indicates the total number of flying nodes, \( m \) is the total number of clusters, \( E_{\text{Residual}}(t-1) \) and \( E_{\text{Residual}}(t) \) show the energy levels of \( U_j \) in two moments \( t-1 \) and \( t \), respectively. \( \ell \) 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 \( a' = \{ \omega'_1, \omega'_2, \omega'_3, \omega'_4, \omega'_5 \} \) 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.

\[ f_2 = 1 - \left( \frac{m_{\text{isolated cluster}}}{m} \right) \]  

(32)

where \( m \) and \( m_{\text{isolated cluster}} \) 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 \( a' = \{ \omega'_1, \omega'_2, \omega'_3, \omega'_4, \omega'_5 \} \) 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 (\( V \in MU_k \in C_k \)) 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.

\[ f_3 = \beta \frac{1}{\sum_{i=1}^{m} \left( \frac{d(LU_k, LU_i)}{N_k} \right)} + (1 - \beta) \sum_{j \neq LU_k} \frac{d(LU_k, LU_j)}{D_{\text{max}}} \]  

(33)

So that \( d(LU_k, LU_i) \) is the distance between \( LU_k \) and \( LU_i \), and \( d(MU_i, LU_k) \) indicates the distance between \( LU_k \) and \( MU_i \). \( N_k \) is the number of cluster members in \( C_k \), and \( m \) is the total number of clusters in the network. \( \beta \) is also a weight coefficient in [0, 1]. \( D_{\text{max}} \) indicates the maximum distance between UAVs in the network and is determined using Eq. (34).

\[ D_{\text{max}} = \sqrt{X^2 + Y^2 + Z^2} \]  

(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.

\[ R_c(F) = \begin{cases} R_{\text{max}}, & F = 1 \\ R_{\text{min}}, & F = -1 \\ F, & \text{Otherwise} \end{cases} \]  

(35)

So that \( F = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3 \). In addition, \( \lambda_1, \lambda_2, \) and \( \lambda_3 \) express weight coefficients and \( \sum_{i=1}^{3} \lambda_i = 1 \). According to the reward function, if \( F = 1 \), the selected action \( a' = \{ \omega'_1, \omega'_2, \omega'_3, \omega'_4, \omega'_5 \} \) obtains the maximum reward value \( R_{\text{max}} \). On the other hand, if \( F = -1 \), then, \( a' = \{ \omega'_1, \omega'_2, \omega'_3, \omega'_4, \omega'_5 \} \) gets the least reward value \( R_{\text{min}} \). 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. \( \alpha \) and \( \gamma \), in an adaptive manner and based on network conditions to improve its performance in the dynamic environment of FANET. Learning rate \( (\alpha) \) 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).
\[
C_{k}^{\text{change}} = \begin{cases} 
0, & N_{k}^{\text{change}} \leq \varphi \\
-1, & N_{k}^{\text{change}} > \varphi 
\end{cases} 
\tag{36}
\]
where \(N_{k}^{\text{change}}\) represents the number of changes in \(C_{k}\) in a specified time interval. If \(N_{k}^{\text{change}}\) is less than a threshold value \(\varphi\), \(C_{k}\) is a stable cluster. In this case, the learning rate \(\alpha\) 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).
\[
\overline{C_{k}^{\text{change}}} = \frac{1}{m} \sum_{k=1}^{m} C_{k}^{\text{change}} 
\tag{37}
\]
So that \(m\) the number of clusters in the network. Finally, \(\alpha\) is calculated based on Eq. (38).
\[
\alpha = \begin{cases} 
1 - e^{\overline{C_{k}^{\text{change}}}}, & \overline{C_{k}^{\text{change}}} \neq 0 \\
0.1, & \overline{C_{k}^{\text{change}}} = 0 
\end{cases} 
\tag{38}
\]

**Algorithm 4 Dynamic and smart adjustment of weight coefficients**

**Input:** GCS (Ground control station): Agent
\(U_{j}\): Flying node \(i\) so that \(i = 1, 2, ..., n\)
Table, \(\text{Neighbor}\): Neighboring table stored in \(U_{i}\)
\(N_{i}\): Number of neighbors of \(U_{i}\)
\(\alpha, \gamma\): Learning parameters
\(ST = \{m_{i1}^j, m_{i2}^j, ..., m_{id}^j\}\): State space set
Action = \{ \(\omega_{i1} \leq \omega_{i2} \leq ..., \omega_{im} \) \}: Action set

**Output:** Updating Q-table for determining the best weight coefficients in Equation (25)

1: Begin
2: for \(i = 1\) to \(n\) do
3: \(U_{i}\): Send its merit parameters including \(E_{i}, CNR_{i}, D_{it}, SC_{i},\) and \(ST_{i}\) to GCS;
4: end for
5: GCS: Set \(e\) in the interval \([0, 1]\) randomly;
6: GCS: Adjust Q-values in Q-table on zero;
7: repeat
8: GCS: Select the next-state as \(ST_{i} = \{m_{i1}^j, m_{i2}^j, ..., m_{id}^j\}\) randomly;
9: GCS: Extract \(a\) and \(\gamma\) from Equations (38) and (40), respectively;
10: \(i = i + 1;\)
11: while \(i \leq N\) do
12: if \(\omega_{i} \leq e\) then
13: GCS: Obtain the action \(a' = \{a_{i1}', a_{i2}', ..., a_{im}'\}\) from Action = \{ \(\omega_{i1}' \leq \omega_{i2}' \leq ..., \omega_{im}' \) \};
14: else if \(\omega_{i} > e\) then
15: GCS: Extract the action with maximum Q-value from Q-table;
16: end if
17: GCS: Calculate \(F = \alpha_{1}f_{1} + \alpha_{2}f_{2} + \alpha_{3}f_{3}\) based on Equations (31), (32), and (33);
18: if \(F = 1\) then
19: \(K_{i} = K_{\text{max}};\)
20: else if \(F = 0\) then
21: \(K_{i} = K_{\text{min}};\)
22: end if
23: \(K_{i} = F;\)
24: GCS: Get the next-state \(ST' = \{m_{i1}', m_{i2}', ..., m_{id}'\}\) from the environment;
25: GCS: Refresh Q-value in Q-table based on \(K_{i};\)
26: \(m = m + 1;\)
27: end if
28: end while
29: \(episode = episode + 1;\)
30: until \(episode \leq M\)
31: End

In addition, QSCR seeks to adjust the discount coefficient \(\gamma\) 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, \(\gamma\) 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. \(\overline{SC_{k}^{\text{score}}}\) is the average speed similarity between \(LU_{i}\) 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 \([0, 1]\).
\[
\overline{SC_{k}^{\text{score}}} = \frac{\overline{SC_{k}} + 1}{2} 
\tag{39}
\]
As a result, \(\gamma\) is calculated based on Eq. (40).
\[
\gamma = 1 - e^{-\left(\frac{\sum_{k=1}^{n} \overline{SC_{k}^{\text{score}}}}{n}\right)} 
\tag{40}
\]
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. \(LU_{j}\) in the cluster \(C_{j}\)) as the next-hop node. If \(LU_{i}\) and \(LU_{j}\) are in each other’s communication range, \(LU_{k}\) sends its data to \(LU_{j}\) directly. Otherwise, \(LU_{k}\) sends this data to the gateway node that is connected to \(C_{j}\).

- **Second mode:** If \(U_{i}\) acts as a cluster member \(MU_{k}\), 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 \(GU_{k}\) in \(C_{i}\), then it examines the destination address inserted in the data packet. If it is the address of the adjacent cluster that is connected to \(GU_{k}\), then, \(GU_{k}\) sends this packet to the nearest gateway node in the adjacent cluster. Otherwise, \(GU_{k}\) transmits its data to its cluster leader \(LU_{k}\).

**Algorithm 5 Greedy routing**

**Input:** \(U_{i}\): Flying node \(i\) so that \(i = 1, 2, ..., n\)
\(U_{j}\): Destination node

**Output:** Forming a route between \(U_{i}\) and \(U_{d}\)

1: Begin
2: if \(U_{i}\) seeks to send its data packets to \(U_{d}\) then
3: if \(U_{i}\) and \(U_{d}\) are neighbor then
4: \(U_{i}\): Send its data packet to \(U_{d}\) directly;
5: else
6: if \(U_{i}\) plays the role of a \(LU_{j}\) in the network then
7: \(LU_{j}\): Select the nearest cluster toward \(U_{d}\) for example \(LU_{j}\), as the next-hop;
8: if \(LU_{i}\) and \(LU_{j}\) communicate each other directly then
9: \(LU_{j}\): Send its data packets to \(LU_{j}\);
10: else
11: \(LU_{j}\): Send its data packets to the \(GU_{k}\) in \(C_{k}\) connected to \(LU_{j}\);
12: end if
13: if \(U_{i}\) plays the role of a \(MU_{k}\) in \(C_{i}\) then
14: \(MU_{k}\): Send its data packets to its \(LU_{k}\);
15: if \(U_{i}\) plays the role of a \(GU_{k}\) in \(C_{i}\) then
16: \(GU_{k}\): Checks the next-hop address inserted in to the data packet;
17: if the next-hop address is the address of the adjacent cluster then
18: \(GU_{k}\): Send its data packets to the nearest gateway node to \(U_{d}\) in this adjacent cluster;
19: else
20: \(GU_{k}\): Send its data packets to its \(LU_{k}\);
21: end if
22: end if
23: end if
24: end if
25: end if
26: end if
27: end if
28: end if
29: end if
30: end if
31: end if
32: end if
33: end if
34: end if
35: end if
36: end if
37: end while
38: \(episode = episode + 1;\)
39: until \(episode \leq M\)
40: 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 km². 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 time-frame 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.

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.

### Table 4: Simulation parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulator</td>
<td>NS2</td>
</tr>
<tr>
<td>Network size</td>
<td>10 × 10 km²</td>
</tr>
<tr>
<td>Number of UAVs</td>
<td>10–100</td>
</tr>
<tr>
<td>Initial energy of UAVs</td>
<td>2000 J</td>
</tr>
<tr>
<td>Communication range of UAVs</td>
<td>250 m</td>
</tr>
<tr>
<td>Velocity of UAVs</td>
<td>30–50 m/s</td>
</tr>
<tr>
<td>Size of data packet</td>
<td>1024 bits</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Gauss Markov (GM)</td>
</tr>
<tr>
<td>Traffic model</td>
<td>Constant bit rate (CBR)</td>
</tr>
<tr>
<td>Data rate</td>
<td>1000 kbps</td>
</tr>
<tr>
<td>Antenna</td>
<td>Omni-Antenna</td>
</tr>
<tr>
<td>Simulation time</td>
<td>1500 s</td>
</tr>
<tr>
<td>Mac standard</td>
<td>IEEE 802.11</td>
</tr>
<tr>
<td>φ (Role change threshold)</td>
<td>2</td>
</tr>
<tr>
<td>Evaluation scales</td>
<td>Time of cluster creation, stability of clusters, network lifetime, routing overhead, and QoS requirements</td>
</tr>
<tr>
<td>Compared schemes</td>
<td>QSCR, ICRA, WCA, and DCA</td>
</tr>
</tbody>
</table>

![Fig. 7. Cluster creation time in different methods.](image-url)
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.
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 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. 14 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.
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 CH selection process, each CH obtains its score, transfers its status information 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 to \( 1 + N_i \) 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.
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 equal to $N_i$ 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, obtains $N_i$ 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 $1 + N_i$ 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, $U_i$ sends a CH declaration message to its neighboring nodes. 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 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 $1 + N_i$. Therefore, 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 smart and dynamic adjustment process of clustering parameters, 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 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. Therefore, 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 QSCR, the communication overhead is calculated according to the following steps. This parameter is presented in Table 5.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Node type</th>
<th>Clustering overhead</th>
<th>Routing overhead</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCA</td>
<td>Cluster head node</td>
<td>$3(1 + N_i) + (1 + 2N_i)$</td>
<td>$N_i + (2 + 2N_i)$</td>
<td>$8N_i + 6$</td>
</tr>
<tr>
<td></td>
<td>Cluster member node</td>
<td>$2(1 + N_i) + 5$</td>
<td>$1$</td>
<td>$2N_i + 8$</td>
</tr>
<tr>
<td>DCA</td>
<td>Cluster head node</td>
<td>$3(1 + N_i) + 2 + (2 + 2N_i)$</td>
<td>$N_i + (2 + 2N_i)$</td>
<td>$10N_i + 9$</td>
</tr>
<tr>
<td></td>
<td>Cluster member node</td>
<td>$3(1 + N_i) + (2 + 2N_i)$</td>
<td>$1$</td>
<td>$5N_i + 9$</td>
</tr>
<tr>
<td>ICRA</td>
<td>Cluster head node</td>
<td>$3(1 + N_i) + 1 + (2N_i) + 2$</td>
<td>$N_i + 2$</td>
<td>$6N_i + 8$</td>
</tr>
<tr>
<td></td>
<td>Gateway node</td>
<td>$2(1 + N_i) + 7$</td>
<td>$1$</td>
<td>$2N_i + 10$</td>
</tr>
<tr>
<td>QSCR</td>
<td>Cluster head node</td>
<td>$2(1 + N_i) + (1 + 3N_i) + 2$</td>
<td>$N_i + 2$</td>
<td>$7N_i + 8$</td>
</tr>
<tr>
<td></td>
<td>Cluster member node</td>
<td>$2(1 + N_i) + 7$</td>
<td>$1$</td>
<td>$2N_i + 10$</td>
</tr>
<tr>
<td></td>
<td>Gateway node</td>
<td>$2(1 + N_i) + 10$</td>
<td>$3$</td>
<td>$2N_i + 15$</td>
</tr>
</tbody>
</table>
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 Deanship for Research and Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project no. (IFKUOR3–596–6).

References


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, 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 aspect 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.