KNIGCN: Key Node Identification in UAV Swarm Networks Using a Graph Convolutional Network

Qixun Ai, Changbo Hou, Zhichao Zhou, *Graduate Student Member, IEEE*, Xiangyu Wu, and Zhen Song

***Abstract*—In the field of Internet of Things (IoT), uncrewed aerial vehicle (UAV) swarms are widely used to assist IoT communication due to their simple deployment, high mobility and high cost effectiveness to help improve IoT network coverage and topological flexibility. However, due to the openness of UAV swarm network, UAV nodes are vulnerable to malicious attacks and fail, especially some key nodes, which will seriously degrade the network performance. To solve this problem, a key node identification model based on graph convolutional network (GCN), named KNIGCN, is proposed. Specifically, the topology model of UAV cluster network is first built based on flying ad hoc network (FANET) communication architecture and Dijkstra's algorithm, then node criticality labels in the network are made by means of traffic statistics, and finally KNIGCN model is used to predict the criticality scores of each node and sort them so as to screen out a certain number of key nodes. The experimental results show that the model has good performance of key node identification. Compared with traditional methods, the proposed method makes a more reasonable judgment of node criticality, and the key nodes identified have a more important impact on network performance. Moreover, the proposed scheme can effectively improve the identification speed of key nodes in large-scale, complex or dynamically changing topology of UAV cluster networks. It can provide an effective scheme for maintaining the security and stability of UAV swarm network communication.**

***Index Terms*—Dijkstra's algorithm, flying ad hoc network (FANET), graph convolutional network (GCN), key node identification, uncrewed aerial vehicle (UAV) swarm network.**

I. Introduction

With the rapid advancement of Internet of Things (IoT) technology, uncrewed aerial vehicle (UAV) swarms have emerged as a novel IoT access solution due to their strong mobility, high flexibility, and low cost. They are widely used across various domains, including traffic management, agricultural remote sensing, and battlefield reconnaissance [1], [2], [3], [4]. However, the open nature of UAV swarms makes their nodes vulnerable to attacks, especially those key nodes that serve as communication hubs and carry a large volume of information. When these key nodes are compromised, the regular operation of the UAV network can be severely disrupted, potentially leading to a total network breakdown [5], [6], [7]. Therefore, research on identifying key nodes within UAV swarm networks, followed by enhanced protection and management of these nodes, is of great significance and value for improving the security and stability of the entire network [8], [9], [10], [11].

Traditional methods for identifying key nodes in networks can primarily be categorized into two types: 1) centrality-based methods [12], [13], [14] and 2) network dismantling and partitioning methods [15], [16], [17]. Centrality-based approaches evaluate node criticality by measuring certain topological features, such as degree centrality (DC) [12], betweenness centrality (BC) [13], and closeness centrality (CC) [14]. However, these methods rely heavily on network topology and often overlook the actual attributes of nodes, making it challenging to accurately assess node criticality within the network. In contrast, network dismantling and partitioning methods go beyond topological features, assessing key nodes by analyzing network integrity and comparing the network state before and after node removal. These approaches, such as node contraction (NC) [15], node deletion (ND) [16], and collective influence [17] methods, generally offer better robustness and accuracy. Nonetheless, they often suffer from high computational costs, time complexity, and strong parameter dependencies, making them difficult to deploy and apply in practice.

With the rapid development of artificial intelligence and numerous breakthroughs across various fields, many researchers have begun using machine learning models and deep neural networks to fit node centrality metrics in networks, aiming to identify key nodes based on the fitting results [18], [19]. Wen et al. [18] proposed a key node identification algorithm based on least-squares support vector machines, employing the analytic hierarchy process (AHP) to obtain criticality scores for nodes and thus identify key nodes. However, the AHP method is inherently subjective and uncertain, which may affect the accuracy of node criticality evaluations. Zhao et al. [19] designed a key node identification algorithm using k-nearest neighbors (KNNs). In their approach, a feature vector combining nine different centrality metrics and infection rates was selected for each node, with the true spreading capability of each node used as the KNN label for training and prediction.

Additionally, with the rise of graph theory concepts and their widespread application in various domains, many researchers

Received 19 July 2024; revised 13 November 2024 and 17 December 2024; accepted 15 January 2025. Date of publication 20 January 2025; date of current version 23 May 2025. This work was supported by the National Natural Science Foundation of China under Grant U23A20271. *(Corresponding author: Changbo Hou.)*

Qixun Ai, Zhichao Zhou, Xiangyu Wu, and Zhen Song are with the College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China.

Changbo Hou is with the College of Information and Communication Engineering and the Key Laboratory of Advanced Marine Communication and Information Technology, Ministry of Industry and Information Technology, Harbin Engineering University, Harbin 150001, China (e-mail: houchangbo@hrbeu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2025.3531789

now view networks as graph-structured data and apply graph neural networks (GNNs) at the graph level to identify key nodes within the network [20], [21], [22]. Huang et al. [20] designed a node criticality evaluation algorithm, RGTN-NIE, which combines graph convolutional network (GCN) with graph representation learning techniques to more accurately assess node criticality using both structural and attribute features of nodes. However, the hyperparameters in this method significantly impact model performance and require considerable computational resources, which limits its practicality. Zhang et al. [21] proposed a new algorithm CGNN based on the convolutional neural network (CNN) and the GNN, which achieved a more accurate identiffcation of key nodes after the application of CGNN in the experiment. Zhao et al. [22] suggested that node criticality should be jointly determined by both network structure and node features. To this end, they proposed a new deep learning model, InfGCN, which can fully represent and learn the adjacency graph and its structural features. Extensive experiments showed that this method accurately identifies influential nodes, with performance significantly surpassing traditional approaches.

The key node identification method based on GNN enables comprehensive extraction and automatic learning of node attribute features and topological structures within network graphs, improving adaptability and generalization over traditional methods. Furthermore, the trained GNN model can directly output criticality scores for all nodes across the network, eliminating the need to assess each node individually and significantly reducing computational time. Based on this approach, a novel framework for identifying key nodes in UAV swarm network is proposed, comprising three main stages: 1) network topology mapping; 2) criticality score labeling; and 3) regression-based model prediction. The first stage utilizes the flying ad hoc network (FANET) architecture and Dijkstra's algorithm to map the UAV swarm network into a topological graph. The second stage generates criticality score labels for each node using traffic statistics. Finally, the designed KNIGCN model extracts features and performs regression to predict node criticality scores, identifying key nodes. Experimental results demonstrate that this approach achieves high accuracy in key node identification, providing an effective solution for enhancing the security and stability of UAV swarm communications. Specifically, when the critical node ratio is set at 0.1, the proposed approach achieves an identification accuracy exceeding 96% across six types of test UAV swarm networks. Moreover, for large-scale, complex, or dynamically changing UAV network topologies, it effectively reduces execution time, thereby accelerating identification speed. In comparison to traditional baseline methods, the key nodes identified by this approach have a more significant impact on network performance. Specifically, when these key nodes fail, the overall network packet loss rate rises sharply, while packet delivery rate and throughput notably decline. The primary contributions of this article can be summarized as follows.

1) A method for generating node criticality labels based on traffic statistics has been proposed. This approach defines the criticality of nodes by simulating communication conditions within a UAV swarm network and recording traffic data for each node. By leveraging this traffic information, the method enables a more accurate assessment of node criticality specifically suited to UAV swarm networks. A higher criticality label indicates that the node has a more significant impact on the communication performance of the UAV swarm network.
2) The KNIGCN model is constructed for identifying key nodes within UAV swarm networks. After the training of KNIGCN, the generative model can output the criticality scores of all nodes in the UAV swarm network at one time, and select a certain number of key nodes by sorting these scores. The proposed model can significantly improve the recognition speed while ensuring the identification accuracy of key nodes, so as to enhance the adaptability to the identification efficiency requirements of large-scale complex and dynamic UAV swarm network in actual scenarios.

The remainder of this article is organized as follows. Section II details the topology model of UAV swarm network. Section III specifically introduces the proposed key node identification model and process. In Section IV, the experimental procedure is presented, and the simulation results are studied and discussed. Section V concludes this article.

II. Topology Model of UAV Swarm Network

A. Communication Architecture

In this article, we adopt the FANET as the communication architecture for the UAV swarm network [23], [24], [25]. As depicted in Fig. 1, the FANET structure consists of a ground control station (GCS) and a set of UAV nodes. While only some UAV nodes can directly communicate with the GCS, each UAV node is independently configured with a communication module, allowing the network to be self-organizing. Nodes receive and relay information via wireless communication to establish the network structure. The UAV swarm network topological graph based on the FANET architecture can be represented as follows:

$$G =< V, E > \tag{1}$$

where $V = \{v_1, v_2, \ldots, v_n, \ldots, v_N\}$ represents the set of nodes in G, v_n represents the nth node in V, and N represents the total number of nodes in G. $E = \{E_{ij} | 0 \leq i \leq N, 0 \leq j \leq N\}$ represents the set of edges in G, E_{ij} represents the connected edges of nodes v_i and v_j, which corresponds to the communication links between nodes in the UAV swarm network. The existence of E_{ij} is determined by

$$d(v_i, v_j) \leq d_m \tag{2}$$

where $d(v_i, v_j)$ represents the position distance between v_i and v_j in the UAV swarm network, d_m represents the maximum communication distance of nodes. The adjacency matrix

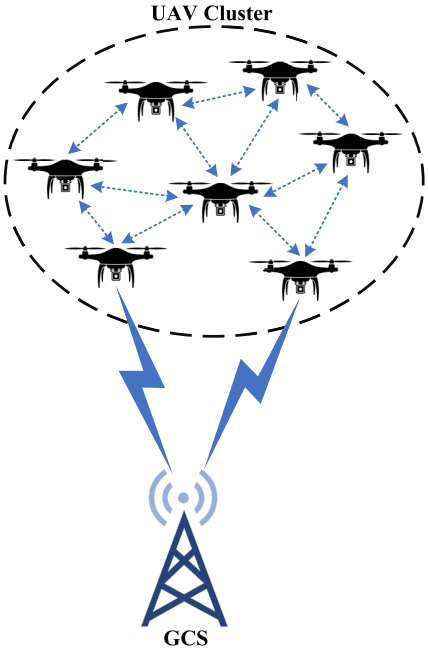


Fig. 1. Structure of FANET.

$A(G) = (a_{ij})_{N\times N}$ of G is an N-by-N matrix. The element a_{ij} at the ith row and jth column of $A(G)$ is defined as follows:

$$a_{ij} = \begin{cases} 1, \text{ if } (v_i, v_j) \in E \\ 0, \text{ otherwise} \end{cases} \tag{3}$$

where $a_{ij} = 1$ represents a direct connection between nodes v_i and v_j, while $a_{ij} = 0$ represents that there is no direct link between these nodes.

For a single UAV swarm network topological graph constructed using the above method, the topology is fixed. However, in real-world scenarios, UAV node positions may vary over time, leading to dynamic changes in the UAV swarm network topology [26]. Therefore, we assume that the UAV swarm network topology remains static within a small time interval, with each topological graph representing the UAV swarm network topology structure during each small interval. The dynamic changes in the UAV swarm network topology can then be represented by a sequence of topological graphs as follows:

$$\boldsymbol{G} = \{G_1, G_2, \ldots, G_t, \ldots, G_T\} \tag{4}$$

where T represents the total observation time,and G_t represents UAV swarm network topological graph corresponding to the tth small time interval.

B. Routing Protocol

As a core technology in UAV swarm network, the routing protocol selects optimal paths for data transmission between nodes, ensuring that the UAV swarm can maintain continuous, efficient, and reliable communication. For the UAV swarm network model in this article, we select the commonly used shortest path routing protocol, which determines communication paths between nodes using Dijkstra's algorithm [27], [28], [29]. Dijkstra's algorithm is designed to solve

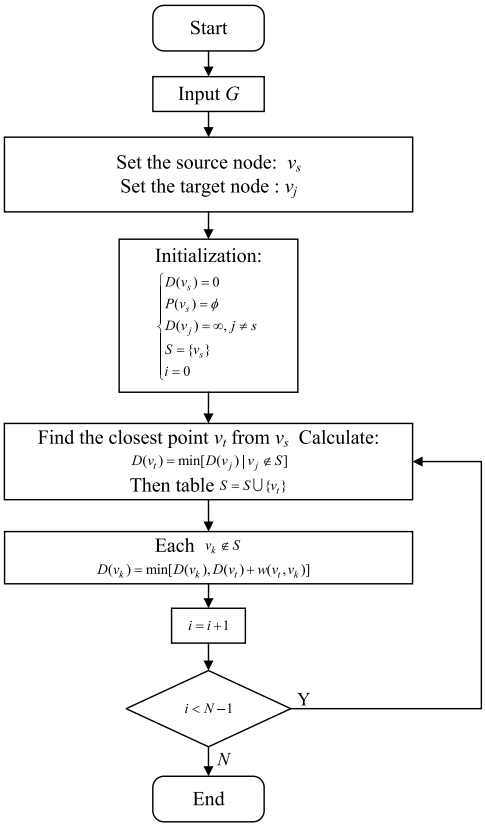


Fig. 2. Flowchart of Dijkstra's algorithm.

the single-source shortest path problem, and its implementation process is shown in Fig. 2. The specific steps of Dijkstra's algorithm are as follows.

1) The initialization of each node in the input the UAV swarm network topological graph G is carried out as follows:

$$\begin{cases} D(v_s) = 0 \\ P(v_s) = \phi \\ D(v_j) = \infty, j \neq s \\ S = \{v_s\} \end{cases} \tag{5}$$

where v_s and v_j, respectively, represent the source node and any node other than the source node. Based on the FANET architecture, the source node is the GCS node, while the destination node is any UAV node. $D(v_j)$ represents the length of the shortest path from v_s to v_j, with $D(v_j) = \infty$ indicating that, during initialization, the source node and all destination nodes are not directly connected. $P(v_j)$ represents the number of preceding nodes in the shortest path to v_j, which is used for path backtracking once the algorithm completes. S refers to the set of marked nodes, initially containing only v_s.

2) All destination nodes are traversed and the node v_t closest to v_s is selected. v_t satisfies

$$D(v_t) = \min\left[D(v_j) | v_j \notin S\right]. \tag{6}$$

v_t can also be considered as an adjacent node of v_s. Let $S = S \bigcup \{v_t\}$, which indicates that the node has already been visited, and there is no need to repeat the selection.

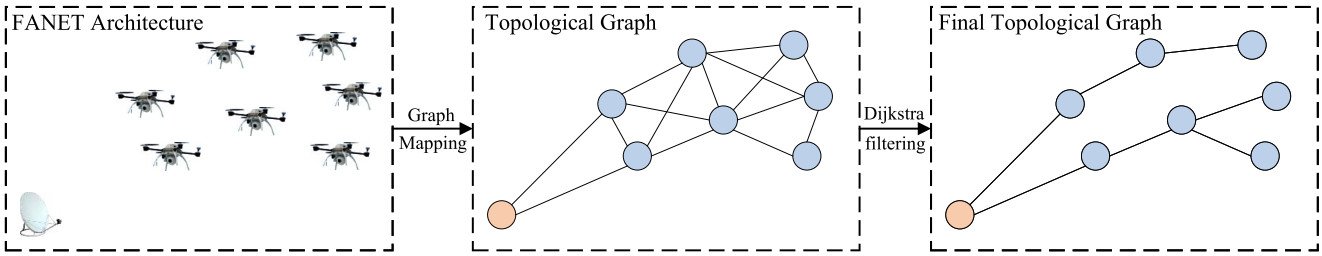


Fig. 3. Topology modeling process of UAV swarm network.

3) Modify the length of the shortest path $D(v_k)$ from v_t to any unlabeled node v_k other than S as follows:

$$D(v_k) = \min[D(v_k), D(v_t) + w(v_t, v_k)] \tag{7}$$

where $w(v_t, v_k)$ is the path length from v_t to v_k.

4) Traverse all nodes, except for v_s, as destination nodes and repeat steps 1–3, obtaining the set E' of all optimal paths from v_s to all destination nodes.

According to the optimal path set E' from the GCS node to all UAV nodes obtained by the above steps, we further filter the edges in the UAV swarm network topological graph constructed in the previous section, and only retain the edges corresponding to the optimal path in E'. The filtering process can be expressed as

$$a'_{ij} = \begin{cases} 1, \text{ if } a_{ij} = 1 \text{ and } (v_i, v_j) \in E' \\ 0, \text{ others} \end{cases} \tag{8}$$

where a'_{ij} represents the elements of the adjacency matrix corresponding to the final UAV swarm network topological graph after filtering. $a'_{ij} = 1$ represents the presence of an edge between the node v_i and the node v_j in the final topological graph, while $a'_{ij} = 0$ indicates the absence of such an edge. In summary, the process of modeling the final UAV swarm network topology is shown in Fig. 3.

Algorithm 1 Making Method of Node Criticality Labels

Input: The topological graph of the UAV swarm network: G.
Output: The dictionary of node critical labels: *dict*.
Initialize the dictionary $dict = \{i : 0 | i \in \{1, 2, \ldots, N\}\}$.
- N: The number of nodes in G.
- g: GCS node index.

Calculate the dictionary of data transmission path from g node to other nodes by Dijkstra's algorithm: *paths*.
for all $n = 1, 2, \ldots, N$ **do**
  if $i \neq g$ **then**
    Find the list of the path from g node to n node: $path = paths[n]$.
    Calculate the length of *path*: L.
    for all $l = 1, 2, \ldots, L$ **do**
      $dict[path[l]] = dict[path[l]] + 1$.
    end for
  end if
end for
Find the minimum value in *dict*: $\min(dict.value)$.
Find the maximum value in *dict*: $\max(dict.value)$.
for all $k = 1, 2, \ldots, N$ **do**
  $dict[k] = \frac{dict[k] - \min(dict.value)}{\max(dict.value) - \min(dict.value)}$.
end for
return *dict*.

III. Key Node Classification Scheme

A. Node Critical Labels

After generating the topological graph of the UAV swarm network, we assigned criticality labels to each node based on a traffic statistical method. This approach simulates the communication among nodes and, upon completion of the simulation, computes the traffic values for each node to derive their criticality scores. The specific implementation process is as follows.

1) For the input topological graph G of the UAV swarm network, the g node representing the GCS node in it sends a single data packet to each of the other nodes.
2) The total number of data packets sent, forwarded and received by each node in G is counted as the communication traffic of the node, and the corresponding traffic value of each node is stored in the dictionary *dict*.
3) Min-max normalization is performed on each traffic value in *dict*, and the resulting normalized values serve as the criticality labels for the respective nodes.

The algorithm for making node criticality labels is shown in Algorithm 1. This criticality label reflects the role of each node in the communication process within the UAV swarm network. A higher criticality label indicates that the node is situated on more communication paths between other nodes and the GCS node, playing a central role in network communication and having a greater impact on the network's performance. In addition, when the proposed node criticality labeling algorithm is applied to a practical dynamic UAV swarm network, the dynamic network topology can be represented as a sequence of static UAV swarm topologies at discrete time intervals. As a result, a critical node sequence is obtained, which consists of the critical nodes corresponding to each static topology. This sequence highlights the nodes that have a significant impact on the communication performance of the UAV swarm network at each time step and reflects the dynamic changes in critical nodes under real-world conditions.

B. KNIGCN Model

Then, the KNIGCN model is constructed to fit the criticality labels of the nodes in the UAV swarm network topological graph G. The specific structure of the KNIGCN model is

shown in Fig. 4. For the input topological graph G, the feature vectors of the UAV nodes are initialized as 1-D unit vectors [1.], indicating that all UAV nodes can only receive signals originating from the GCS address. The feature vector for the GCS node is set to $[(N-1).]$, representing that the GCS node can send signals to all UAV nodes. The final feature matrix for all nodes in the network topology is then as follows:

$$\begin{aligned} \boldsymbol{X} &= \left[\boldsymbol{x}_{u_0}, \boldsymbol{x}_{u_1}, \ldots, \boldsymbol{x}_g, \ldots, \boldsymbol{x}_{u_{N-1}}\right] \\ &= [[1.], [1.], \ldots, [(N-1).], \ldots, [1.]] \end{aligned} \tag{9}$$

where $\boldsymbol{x}_{u_0}$ and $\boldsymbol{x}_g$ represent the feature vectors of UAV nodes and GCS nodes, respectively. Then, the G containing node features is input to the GCN layer for feature extraction. GCN is a semi-supervised algorithm, which uses the structure of the graph and the node feature vector to learn the node representation vector. The calculation process of GCN can be expressed as follows [30]:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right) \tag{10}$$

$$\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I} \tag{11}$$

$$\tilde{\boldsymbol{D}} = \sum_j \tilde{\boldsymbol{A}}_{ij} \tag{12}$$

where $\boldsymbol{I}$ represents the identity matrix, $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{D}}$ represent the adjacency matrix and degree matrix of G after symmetric normalization, respectively, $\tilde{\boldsymbol{D}}^{-[1/2]}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-[1/2]}$ form a sparse matrix, $\boldsymbol{H}^{(l)}$ is the feature matrix of the input of the l layer of the network, $\boldsymbol{H}^{(l+1)}$ is the feature matrix of the output of the l layer of the network and the input of the $l+1$ layer of the network, and $\boldsymbol{W}^{(l)}$ is the learnable weight parameter of the l layer of the network. σ is the activation function, and the activation function is set as ReLU in our scheme [31]. After node feature representation through GCN, the fully connected (FC) layer is used for feature aggregation. The specific operation is as follows:

$$\boldsymbol{H}^{(l+1)} = \text{Linear}\left(\boldsymbol{H}^{(l)}\boldsymbol{W}^T + b\right) \tag{13}$$

where $\boldsymbol{W}$ represents the weight parameter to be learned in the linear layer, $(\cdot)^T$ stands for transpose, b stands for bias. The output feature size of FC layer is set to 1, that is, the single feature value after the feature aggregation of each node is output as the predicted value of node criticality, and the mean-square error (MSE) loss function is used to measure the difference between the predicted result and the actual node criticality label, so as to continuously optimize the model in the training process. The MSE loss function is defined as

$$\text{MSELoss} = \frac{1}{N}\sum_{i\in N}\left(\text{output}_i - \text{label}_i\right)^2 \tag{14}$$

where N represents the total number of nodes, output_i and label_i, respectively, represent the output prediction value and criticality score label corresponding to node v_i.

After training, the generative model is able to predict the criticality score of each node in the input UAV swarm network topological graph. The nodes are then ranked by their criticality scores in descending order, allowing the selection of a specified number of top-scoring nodes as the identified key nodes. This selection process can be expressed as follows:

$$\text{idx} = \text{top-k}(\text{output}, [kN]) \tag{15}$$

where top-k$(\cdot)$ is a function that returns the index corresponding to the maximum kN value in the output vector of the criticality prediction value of nodes in the KNIGCN model. $[\cdot]$ represents the floor function. k represents the proportion of the number of key nodes in the total number of nodes, and a certain number of key node sets can be obtained by setting k.

C. Time Complexity Analysis

Finally, we conduct a comparative analysis of the time complexity between the original node criticality label generation algorithm and the node criticality score prediction algorithm based on KNIGCN to verify the efficiency of our approach in identifying key nodes. In the subsequent analysis, we define algorithm time complexity as the relationship between the execution time growth rate and the number of nodes N in the input UAV swarm network topological graph G, represented as a function $f(N)$ and expressed using O notation as follows:

$$T(G) = O(f(N)) \tag{16}$$

where $T(G)$ represents the execution time of the algorithm when the input is G.

For the algorithm of node criticality label generation, as illustrated in Algorithm 1, the time complexity for an input topological graph G with N nodes can be described as

$$T_{\text{Label}}(G) = O\left(N^2 + N - 1\right) + O\left(N \times \overline{L}\right) + O(N) = O\left(N^2\right) \tag{17}$$

where $O(N^2 + N - 1)$ represents the time complexity of Dijkstra's algorithm. $O(N)$ represents the time complexity of the min-max normalization process. $O(N \times \overline{L})$ represents the time complexity for calculating node traffic, $\overline{L}$ represents the average path length from the GCS node to all other nodes, which can be expressed as

$$\overline{L} = \frac{\sum_{i=1}^{N-1} L_i}{N-1} \tag{18}$$

where L_i represents the path length from GCS node to other i node. It is evident that $T_{\text{Label}}(G)$ is primarily determined by the number of nodes N and the average path length $\overline{L}$ in G.

For the time complexity of the node criticality score prediction algorithm based on KNIGCN, the prediction process can be regarded as a forward propagation process. When the same topological graph G with N nodes is input, the time complexity of the graph convolutional layer in KNIGCN can be expressed as follows:

$$T_{\text{gcl}}(G) = O\left(C_{\text{gcl}} \times F_{\text{gcl}} \times (N-1)\right) = O(N) \tag{19}$$

where C_{gcl} represents the dimensionality of the input node feature vector for the graph convolutional layer. F_{gcl} represents the dimensionality of the output feature vector for the graph

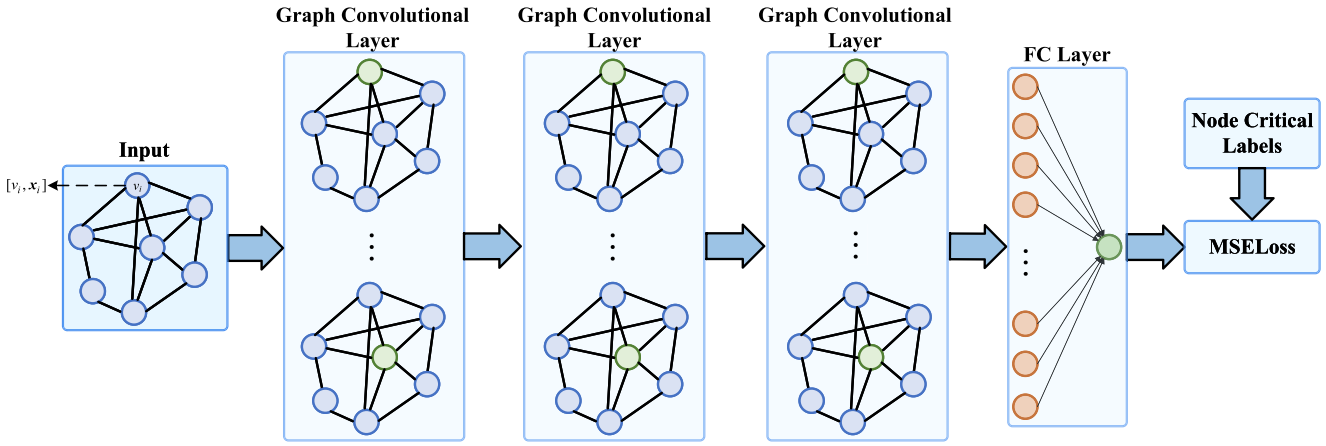


Fig. 4. Structure of KNIGCN.

convolutional layer. The time complexity of the FC layer in KNIGCN can be expressed as

$$T_{fcl}(G) = O\big(C_{fcl} \times F_{fcl} \times N\big) \ = O(N) \tag{20}$$

where C_{fcl} represents the dimensionality of the input node feature vector for the FC layer. F_{fcl} represents the dimensionality of the output feature vector for the FC layer. Fig. 4 shows that the KNIGCN model consists of three graph convolutional layers and one FC layer, where the initial input node feature vector dimensionality is 1, and the final output node feature vector dimensionality from the FC layer is also 1, representing the predicted node criticality score. Therefore, the time complexity of KNIGCN is as follows:

$$\begin{aligned} T_{\text{KNIGCN}}(G) = O(F_1 \times N) \ + O(C_1 \times F_2 \times N) \\ +O(C_2 \times F_3 \times N) \ + O(C_3 \times N) = O(N) \end{aligned} \tag{21}$$

where C_1, C_2 and C_3 represent the dimension size of the input node feature vectors for the first, second, and third layers, respectively, while F_1, F_2 and F_3 represent the dimension size of the output node feature vectors for these layers. These values satisfy the relationships $C_1 = F_1$, $C_2 = F_2$ and $C_3 = F_3$. Additionally, during the prediction process, KNIGCN can accelerate prediction speed by splitting the input topological graph G into multiple subgraphs for parallel processing on a CPU or GPU. This results in the final time complexity of KNIGCN being expressed as

$$\begin{aligned} T_{\text{KNIGCN}}(G) &= \frac{1}{P}(O(F_1 \times N) \ + O(C_1 \times F_2 \times N) \\ &\quad +O(C_2 \times F_3 \times N) \ + O(C_3 \times N)) \\ &= \frac{O(N)}{P} = O(N) \end{aligned} \tag{22}$$

where P represents the number of parallel threads that can be executed by the CPU or GPU. It is evident that T_{KNIGCN} is primarily determined by the number of nodes N, the number of neurons in each layer of the network, and the number of parallel threads P that the CPU or GPU can execute.

By comparing $T_{\text{Label}}(G)$ with $T_{\text{KNIGCN}}(G)$, it can be observed that $T_{\text{Label}}(G) = O(N^2) > O(N) = T_{\text{KNIGCN}}(G)$, which indicates that the number of nodes N in G has a greater impact on $T_{\text{Label}}(G)$. Additionally, $T_{\text{Label}}(G)$ is affected by the average path length $\overline{L}$, whereas T_{KNIGCN} can be optimized by adjusting network parameters and further reduced through parallel processing on CPU or GPU. Therefore, for the large-scale and complex UAV swarm network topological graph with a large number of nodes and a large density of edges, the KNIGCN method of the proposed scheme can effectively reduce the execution time of critical node identification.

Furthermore, we analyze the time complexity of our proposed approach in the context of dynamic changes in UAV swarm network topology. Given a sequence of the UAV swarm network topologies $\{G_1, G_2, \ldots, G_t, \ldots, G_T\}$ obtained over a period of observation, where T represents the number of topologies, the time complexity of the node criticality labeling algorithm can be expressed as follows:

$$\begin{aligned} T_{\text{Label}}(\{G_1, G_2, \ldots, G_t, \ldots, G_T\}) &= T_{\text{Label}}(G_1) + T_{\text{Label}}(G_2) \\ &\quad + \cdots + T_{\text{Label}}(G_t) \\ &\quad + \cdots + T_{\text{Label}}(G_T) \\ &= O\Big(N^2\Big) \end{aligned} \tag{23}$$

where $T_{\text{Label}}(G_t)$ represents the time complexity of the algorithm for the UAV swarm network topological graph G_t at time t, with the calculation method consistent with (17). N represents the number of nodes in the topological graph, which remains the same across all topological graphs in the dynamic sequence. For the node criticality score prediction algorithm based on KNIGCN, when inputting $\{G_1, G_2, \ldots, G_t, \ldots, G_T\}$, the sequence can be merged into a single large graph using batch processing. During the prediction process, the large graph can be split into multiple subgraphs for parallel processing via CPU or GPU, thus accelerating the prediction speed. The time complexity for processing the dynamic topology sequence with KNIGCN can be expressed as follows:

$$\begin{aligned} &T_{\text{KNIGCN}}(\{G_1, G_2, \ldots, G_t, \ldots, G_T\}) \\ &= \frac{T_{\text{KNIGCN}}(\text{batch}\{G_1, G_2, \ldots, G_t, \ldots, G_T\})}{P} \\ &= \frac{O(T \times N)}{P} = O(N) \end{aligned} \tag{24}$$

where batch represents the batch processing method used to merge $\{G_1, G_2, \ldots, G_t, \ldots, G_T\}$ into a single large graph. In

TABLE I
SIMULATION PARAMETERS OF UAV SWARM COMMUNICATION NETWORK

Parameters	Value
Scene	1000m×1000m
Number of nodes	51
MAC	IEEE 802.11b
Channel	WI-FI Channel
Routing protocol	SR
Simulation time	30s
Mobility model	Constant Position
Communication distance	200m
The packet size of data	512byte
The transfer rate of data	2 packets/s

summary, compared to the time complexity of identifying key nodes in a single topology, both the node criticality labeling method and the KNIGCN method for dynamic graph sequences have a time complexity that is, on average, T-times greater. However, during the prediction process, KNIGCN further optimizes its time complexity through batch processing and parallel execution on the CPU or GPU. Therefore, even in the case of dynamic changes in UAV swarm network topology, the KNIGCN method can effectively reduce the time complexity of the UAV swarm network topology sequence, resulting in faster prediction speeds.

IV. SIMULATIONS AND RESULT ANALYSIS

A. Datasets

We used NS3 to simulate UAV swarm network. The network model includes a GCS node, positioned at (500 m, 500 m), with remaining UAV nodes randomly distributed across a 1000 m $\times$ 1000 m square area. The layout ensures network connectivity. Each node is set up with signal transmission and reception capabilities to establish an FANET. Table I provides the specific parameters used in the UAV swarm network model. The specific parameters of the UAV swarm network model are detailed in Table I. The resulting network topologies are extracted to create the UAV swarm network topology datasets.

In addition, considering the training and evaluation of KNIGCN, we set the output feature dimension of each graph convolutional layer in KNIGCN to 4. A total of 1000 UAV swarm network topological graphs, each containing 51 nodes, were generated. These topological graphs were then split into training and testing sets in a 9:1 ratio.

B. Environment

All simulation experiments are configured in a unified environment, and KNIGCN and all baseline model simulations are based on the Pytorch deep learning framework and run on NVDIA GeForce RTX 3070 Ti. In addition, for the training parameters, the learning rate was set to 0.001, the training batch size was set to 100, the test batch size was set to 50, and the training rounds were set to 200. Adam optimizer was used to complete the training of the KNIGCN model.

C. Baseline Model

We chose six different baseline key node identification methods to compare with the KNIGCN model, namely DC [12], BC [13], CC [14], k-shell (KS) decomposition method [32], NC [15], and ND [16], the following are simple definitions of these methods.

1) *DC:* DC judges the criticality of a node according to the number of connected edges. The larger the DC of a node, the greater the number of connected edges.
2) *BC:* BC evaluates the criticality of a node according to the number of shortest paths through the node, which can reflect the role of the node as a bridge in information flow.
3) *CC:* CC uses the average shortest path length between nodes in the network and other nodes to judge the criticality of a node. The larger the CC of a node, the faster it can reach other nodes in the network.
4) *KS:* KS is a coarse-grained node criticality division method, which divides each node in the network into layers according to the node degree, and all nodes in each layer have the same KS value. The larger KS value indicates that the node is located in the core position of the network and has higher influence and criticality.
5) *NC:* NC iteratively shrinks and aggregates the nodes and their adjacent nodes in the network into a node and computes the cohesion degree of the network to obtain the criticality of nodes. The greater the cohesion of the network after NC, the more important the node is.
6) *ND:* ND judges the criticality of a node by counting the changes in the number of spanning trees before and after the deletion of a node. The more the number of spanning trees is reduced, the greater the influence of the node on the whole network is, and the more key the node is.

D. Result and Discussions

In the following, we analyze the feasibility of KNIGCN model for identifying key nodes. Fig. 5 shows a comparison of the KNIGCN model's predictions after training with the actual criticality labels of 200 randomly selected test nodes. Fig. 5 indicates that while minor discrepancies exist between predicted and actual criticality scores, the absolute error remains low, with a maximum error around 0.1. Additionally, the overall trend between predicted and actual criticality is closely aligned, allowing for accurate identification of key nodes through ranking of the predicted results.

Then, we calculate the regression performance metrics of the KNIGCN model on the test set, including MSE, root mean squared error (RMSE), mean absolute error (MAE), mean bias error (MBE), and the Coefficient R2 of Determination, as shown in Table II. From Table II, it can be observed that MSE, RMSE, and MAE values are relatively low, indicating a small difference between the model's predictions and the actual criticality scores. The negative MBE indicates that the predicted values are, on average, slightly lower than the actual criticality scores. The R2 value, close to 1, suggests a good model fit and high prediction accuracy.

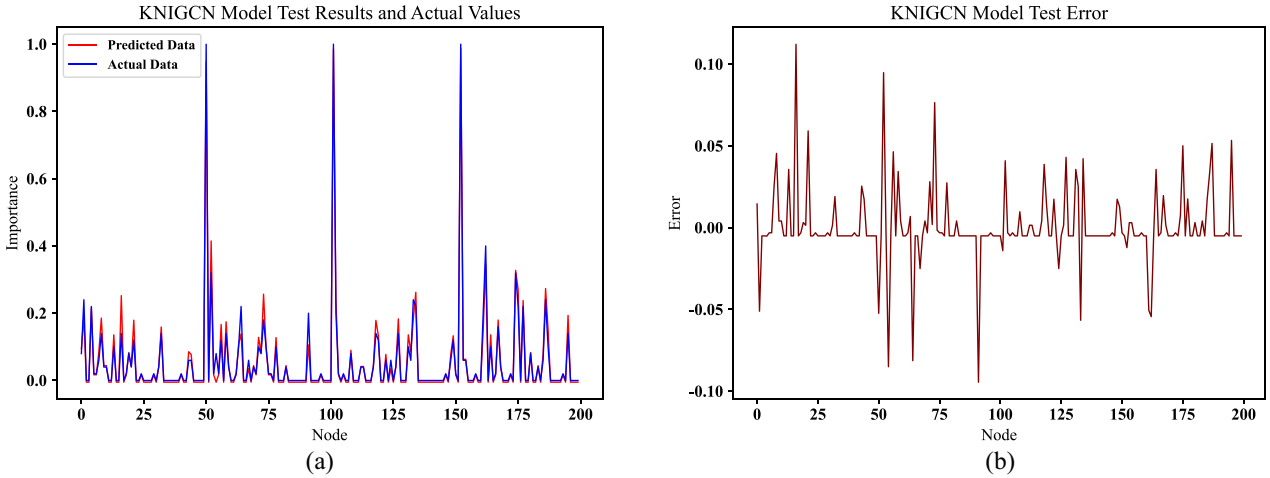


Fig. 5. Comparison of KNIGCN test results with actual criticality scores. (a) KNIGCN model test results and actual values. (b) KNIGCN model test error.

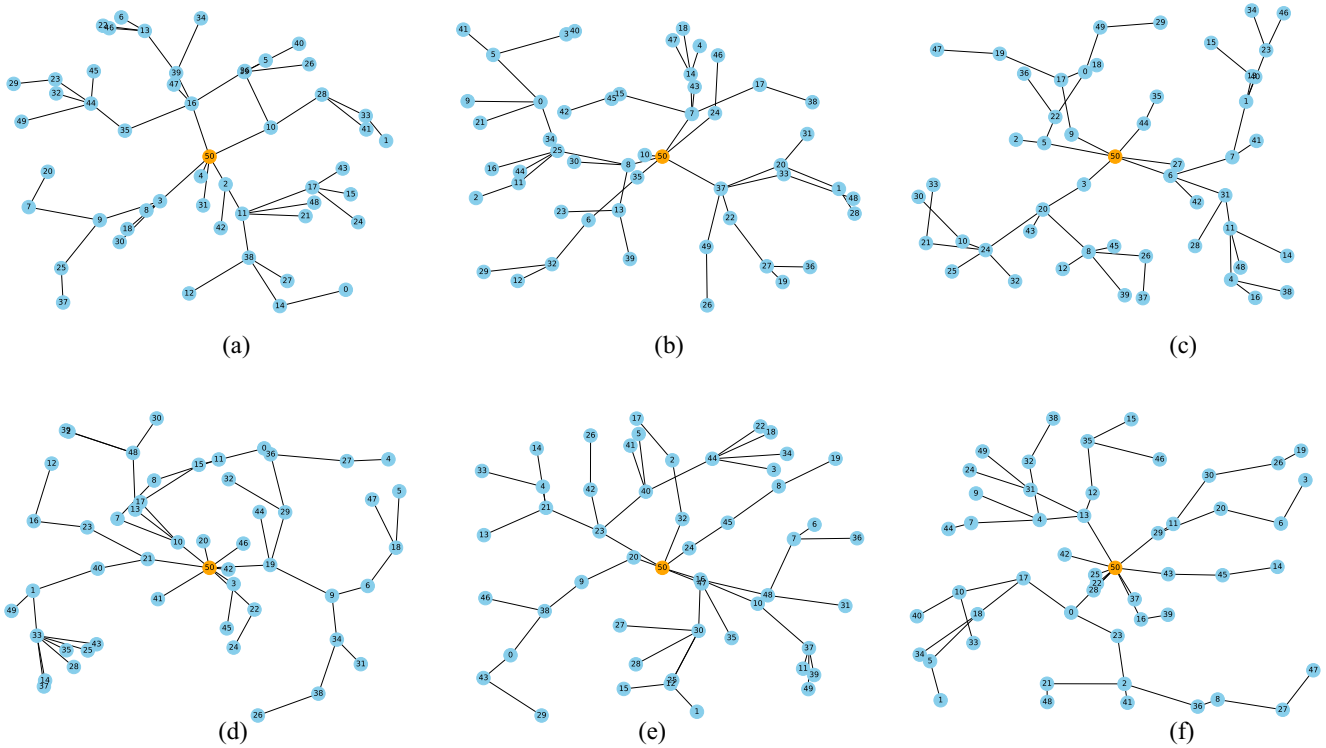


Fig. 6. Structure of six test network topologies. (a) Topology (a). (b) Topology (b). (c) Topology (c). (d) Topology (d). (e) Topology (e). (f) Topology (f).

TABLE II
PREDICTIVE REGRESSION PERFORMANCE INDICATORS

Performance metrics	Value
MSE	0.0018
RMSE	0.0426
MAE	0.0168
MBE	-0.0031
R2	0.9278

Additionally, we randomly generated six UAV swarm network topologies to further evaluate the KNIGCN model. The structures of these six topologies are shown in Fig. 6, with each topology consisting of one GCS node and 50 UAV nodes. Fig. 7 presents a visualization of the KNIGCN model's predictions on these six test topologies. Nodes are shaded with varying color intensities, representing differences in their predicted criticality scores. Darker colors indicate higher criticality scores. Notably, in each topology shown in Fig. 7, Node 50, representing the GCS node, is the darkest, indicating the highest predicted criticality. However, in subsequent analyses for key node selection, the GCS node is excluded, focusing only on key UAV nodes.

To select a subset of nodes with the highest predicted criticality scores as key nodes, we investigated how the proportion k of key nodes to the total node count affects the accuracy of key node identification. For values of k ranging from 0.05 to 0.3 with a step size of 0.01, Fig. 8 displays the

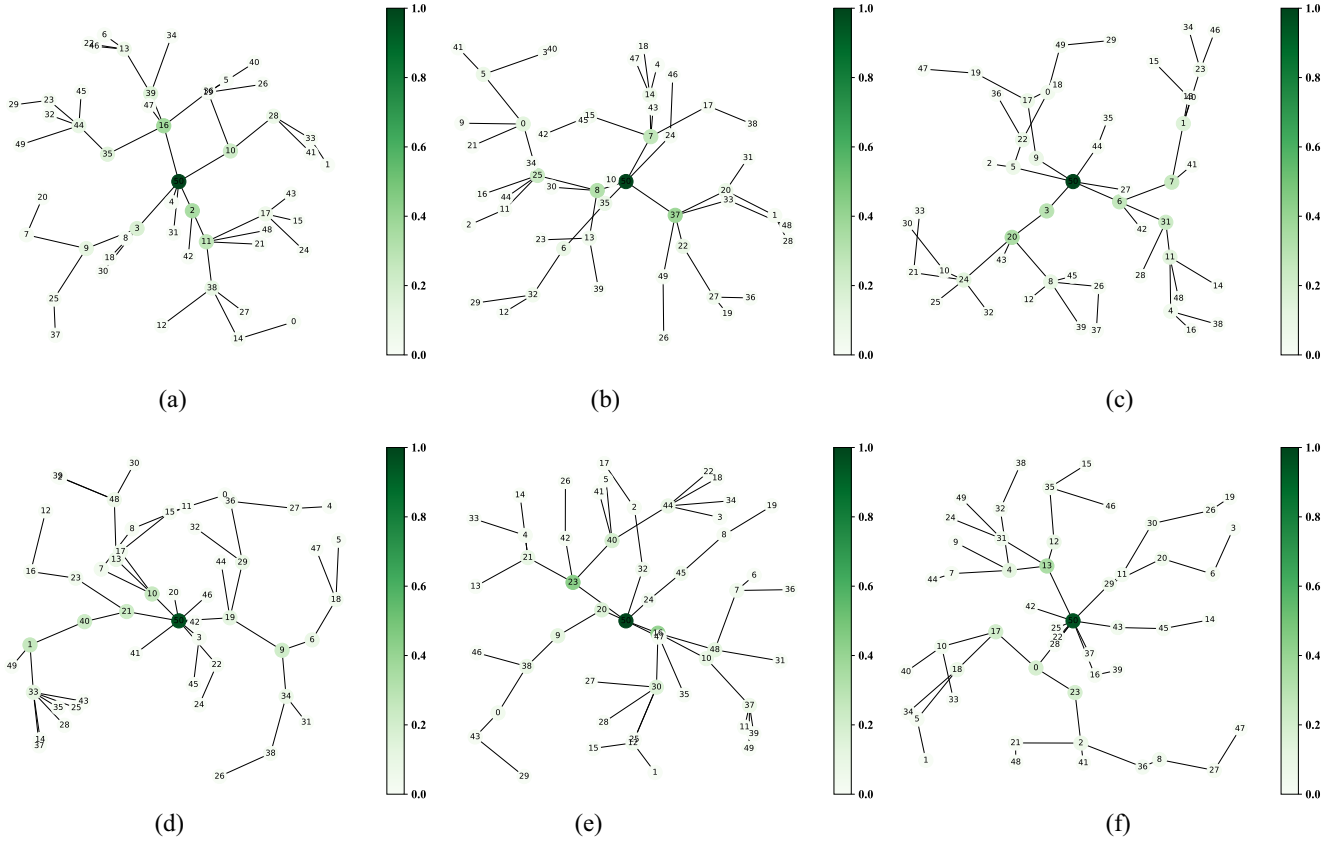


Fig. 7. Visualization results of KNIGCN model prediction. (a) Topology (a). (b) Topology (b). (c) Topology (c). (d) Topology (d). (e) Topology (e). (f) Topology (f).

key node identification accuracy achieved by KNIGCN across the six test network topologies. The key node identification accuracy is defined as follows:

$$\mathrm{Acc} = \frac{|\mathrm{idx}_{\mathrm{KNIGCN}} \cap \mathrm{idx}_{\mathrm{Label}}| + |\overline{\mathrm{idx}_{\mathrm{KNIGCN}}} \cap \overline{\mathrm{idx}_{\mathrm{Label}}}|}{N} \quad (25)$$

where $\mathrm{idx}_{\mathrm{KNIGCN}}$ represents the set of node indices corresponding to the top k key nodes identified by the KNIGCN model, while $\mathrm{idx}_{\mathrm{Label}}$ represents the set of node indices corresponding to the k nodes with the highest criticality labels. The sets $\overline{\mathrm{idx}_{\mathrm{KNIGCN}}}$ and $\overline{\mathrm{idx}_{\mathrm{Label}}}$ are complementary to each other. $|\cdot|$ represents the size of the set. N represents the total number of nodes, where $N = 51$. As shown in Fig. 8, the choice of k affects the accuracy of key node identification in the KNIGCN model. Furthermore, the model's performance varies depending on the network topology. However, regardless of the value of k, the KNIGCN model achieves an accuracy of over 88% in identifying key nodes across all UAV swarm network topologies. Specifically, the recognition accuracy for the UAV swarm network topologies (a), (b), (c), and (e) exceeds 96%. To optimize the recognition performance of the KNIGCN model, we set $k = 0.1$ for all subsequent analyses, ensuring that the accuracy exceeds 96% across all topologies.

Based on $k = 0.1$, we further evaluated the performance of the KNIGCN model in identifying key nodes across six topological graphs. The key performance metrics, including precision, recall, and F1 score are presented in Table III. These

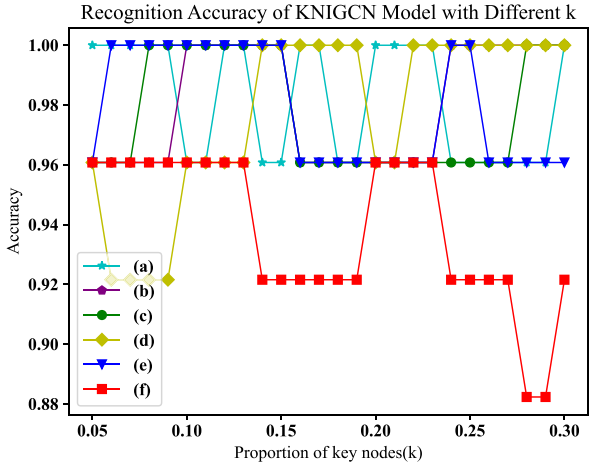


Fig. 8. Identification accuracy of KNIGCN model with different k.

metrics are defined as follows:

$$\mathrm{Pre} = \frac{|\mathrm{idx}_{\mathrm{KNIGCN}} \cap \mathrm{idx}_{\mathrm{Label}}|}{[kN]} \quad (26)$$

$$\mathrm{Rec} = \frac{|\mathrm{idx}_{\mathrm{KNIGCN}} \cap \mathrm{idx}_{\mathrm{Label}}|}{[kN]} \quad (27)$$

$$\mathrm{F1} = \frac{2 \times \mathrm{Pre} \times \mathrm{Rec}}{\mathrm{Pre} + \mathrm{Rec}}. \quad (28)$$

Table III demonstrates that the KNIGCN model achieves precision, recall, and F1 scores of 80% or higher across all six

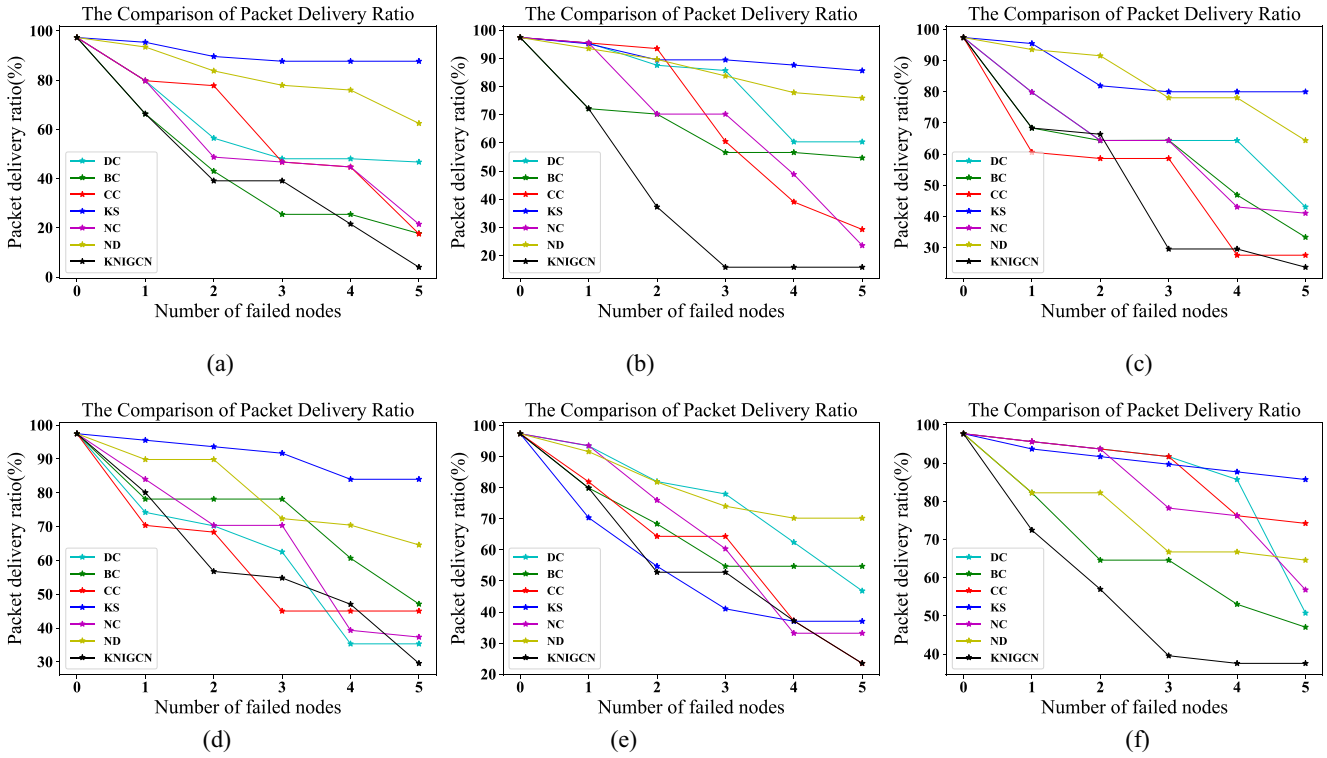


Fig. 9. Comparison of packet delivery ratio. (a) Topology (a). (b) Topology (b). (c) Topology (c). (d) Topology (d). (e) Topology (e). (f) Topology (f).

TABLE III
IDENTIFICATION PERFORMANCE OF KNIGCN WHEN $k = 0.1$

Index	Acc(%)	Pre(%)	Rec(%)	F1(%)
(a)	96.08	80.00	80.00	80.00
(b)	100.00	100.00	100.00	100.00
(c)	100.00	100.00	100.00	100.00
(d)	96.08	80.00	80.00	80.00
(e)	100.00	100.00	100.00	100.00
(f)	96.08	80.00	80.00	80.00

topological graphs. This suggests that when selecting 5 key nodes from 50 UAV nodes at a $k = 0.1$ ratio, the model made at most one incorrect key node prediction. These results further confirm the robustness of the KNIGCN model in identifying key nodes.

In the following, we will further verify the innovative aspects of this approach. First, we examine the validity and applicability of the algorithm used for generating criticality labels for nodes. The criticality of nodes within six types of the UAV swarm network topological graphs is assessed using our node-criticality labeling algorithm and several baseline methods. For each topological graph, we rank nodes based on their criticality scores from each method in descending order. The results for the critical node indices and their ranks when $k = 0.1$ are shown in Table IV. In this table, higher index values indicate greater node criticality, suggesting these nodes hold higher importance. As shown in Table IV, the critical node indices identified by each method for a given topological graph share some overlap, indicating certain methodological correlations. However, the ranking orders of key nodes vary between methods, reflecting different priorities in their assessments of node criticality.

We then used the NS3 software to simulate network communication for six UAV swarm network topologies. In the simulation environment detailed in Table I, we tested the overall network performance after the identified key nodes from each method failed. Specifically, the packet delivery rate, packet loss rate, and throughput are shown in Figs. 9–11, respectively. As illustrated, when $k = 0.1$, network performance declines most noticeably following the failure of five key nodes identified by our node-criticality labeling algorithm, compared to other baseline methods. This includes the highest increase in packet loss rate and the largest decrease in packet delivery rate and throughput. This suggests that the key nodes identified by our method handle more data during communication, and their failure leads to the breakdown of more communication links, causing network disruptions. These results support the validity and applicability of our node-criticality labeling method for determining criticality in UAV swarm network. It is noteworthy, however, that for a certain number of key nodes, some baseline methods have a greater impact on network performance degradation than ours. Interestingly, nodes that cause a sharp decline in network performance are often among the five identified by our method, further confirming the accuracy and reliability of our making method of node criticality labels for UAV swarm network.

Finally, we analyzed the execution time of the KNIGCN model in identifying key nodes to verify its advantages in recognition speed. We set the range of node counts to [51, 510] with increments of 51, while keeping other parameters constant. For each node count, we randomly generated 1000 UAV swarm network topological as inputs and obtained the average critical node identification accuracy of the KNIGCN model across different node quantities. We also measured the average

TABLE IV
INDEX INFORMATION OF CRITICAL NODES IDENTIFIED BY KNIGCN WHEN $k = 0.1$

Index	Serial number	DC	BC	CC	KS	NC	ND	**Our method**
(a)	1	V3	V16	V3	V15	V3	V25	V16
	2	V11	V11	V4	V17	V16	V38	V2
	3	V19	V3	V16	V21	V4	V9	V11
	4	V43	V38	V31	V24	V31	V45	V10
	5	V4	V28	V2	V43	V11	V11	V3
(b)	1	V30	V37	V30	V4	V30	V3	V37
	2	V14	V30	V10	V14	V25	V5	V8
	3	V43	V0	V8	V18	V34	V27	V7
	4	V25	V49	V7	V43	V7	V32	V25
	5	V34	V39	V35	V46	V37	V30	V0
(c)	1	V9	V20	V6	V28	V9	V19	V20
	2	V31	V3	V27	V31	V31	V28	V3
	3	V0	V28	V31	V42	V28	V1	V6
	4	V4	V9	V3	V14	V6	V23	V7
	5	V6	V1	V42	V4	V27	V9	V31
(d)	1	V10	V40	V21	V14	V33	V34	V1
	2	V3	V1	V20	V25	V21	V38	V10
	3	V19	V33	V10	V28	V35	V1	V40
	4	V21	V9	V17	V33	V19	V40	V21
	5	V15	V19	V7	V35	V20	V48	V9
(e)	1	V8	V23	V20	V16	V47	V0	V23
	2	V30	V30	V23	V20	V23	V21	V16
	3	V47	V9	V47	V24	V20	V37	V40
	4	V16	V40	V16	V47	V16	V38	V20
	5	V20	V38	V24	V32	V40	V43	V24
(f)	1	V22	V17	V22	V16	V22	V2	V13
	2	V25	V23	V28	V22	V28	V8	V17
	3	V28	V2	V25	V25	V17	V17	V23
	4	V43	V4	V17	V28	V25	V21	V0
	5	V0	V43	V42	V37	V0	V23	V4

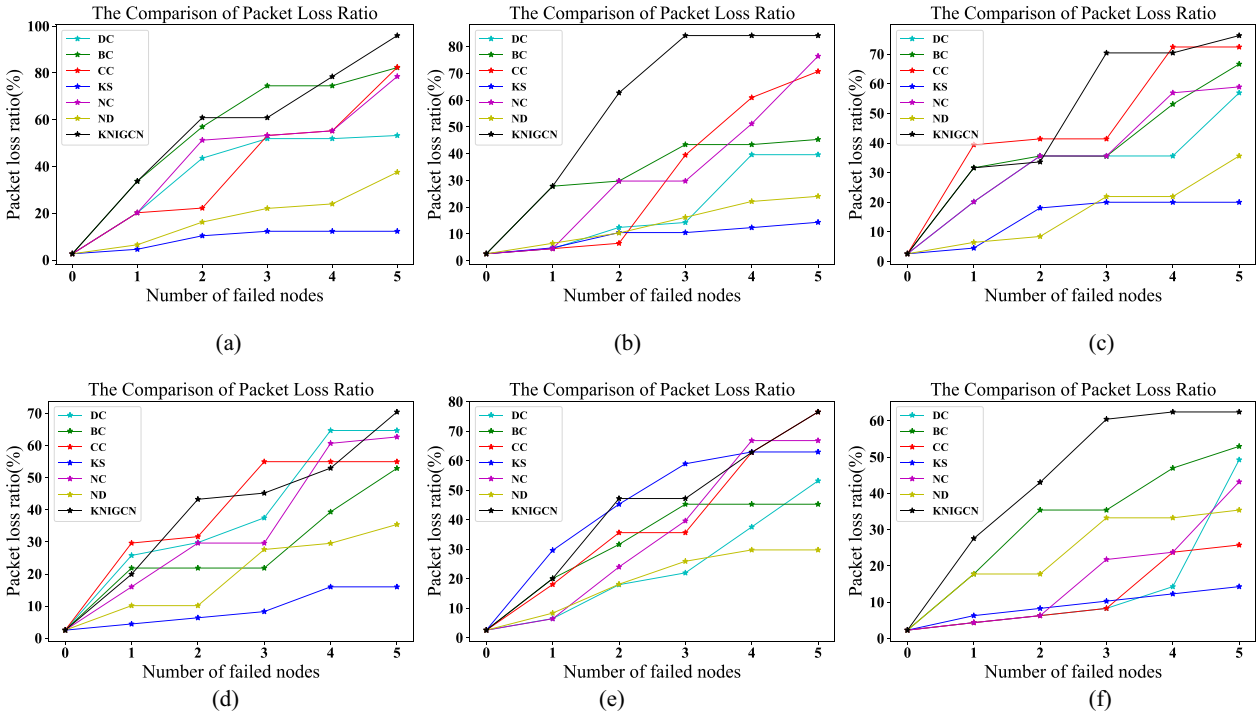


Fig. 10. Comparison of packet loss ratio. (a) Topology (a). (b) Topology (b). (c) Topology (c). (d) Topology (d). (e) Topology (e). (f) Topology (f).

time required by both the node-criticality labeling method and the KNIGCN model for assessing and predicting node criticality as node count varied. The results are shown in Fig. 12. As illustrated, KNIGCN achieves an average critical node identification accuracy above 96% across all tested node counts. Additionally, when the node count exceeds 102, the

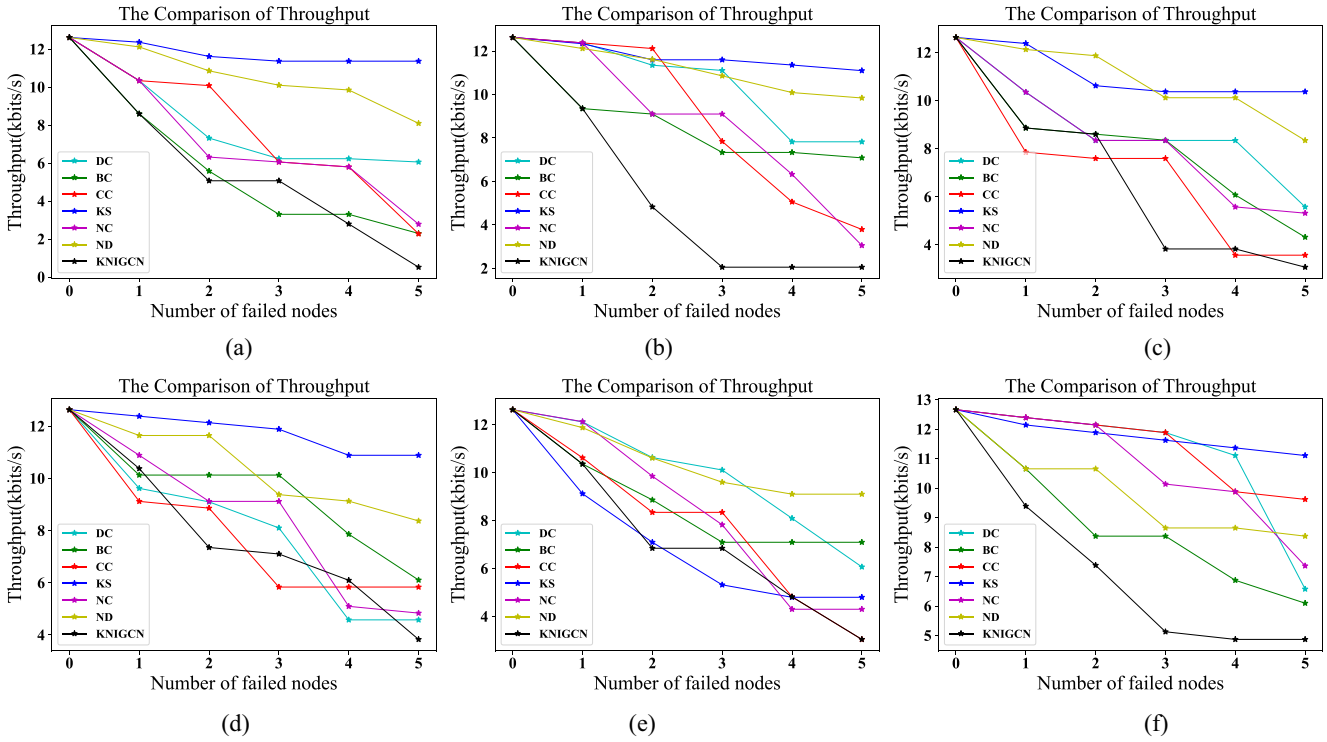


Fig. 11. Comparison of throughput. (a) Topology (a). (b) Topology (b). (c) Topology (c). (d) Topology (d). (e) Topology (e). (f) Topology (f).

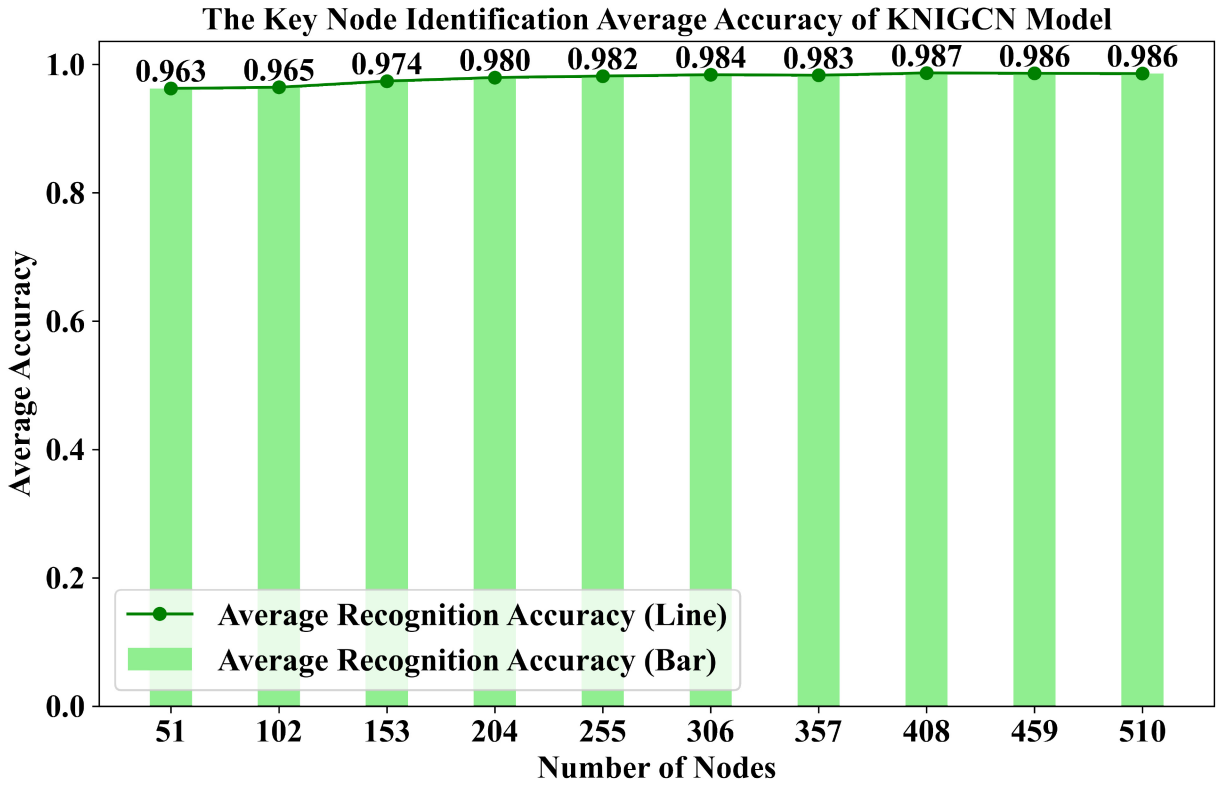
KNIGCN model's average execution time is shorter than that of the node-criticality labeling method, and this time advantage increases with higher node counts. At a node count of 510, the average execution time of the labeling method is approximately 19 times that of the KNIGCN model. These results demonstrate that, for the large and complex UAV swarm network, the KNIGCN model significantly improves the speed and efficiency of critical node identification without compromising accuracy.

Furthermore, we examined the execution time of the KNIGCN model for identifying key nodes under dynamically changing UAV swarm network topologies. We set the range of graph counts within the input dynamic sequence to [50, 500], with increments of 50, and maintained a constant node count of 51 per graph, while keeping other parameters constant. Fig. 13 illustrates the KNIGCN model's average critical node identification accuracy across different graph counts and compares the total time required by both making method of node criticality labels and the KNIGCN model for assessing and predicting node criticality. As shown in Fig. 13, KNIGCN consistently achieves an average identification accuracy above 96% across all tested graph counts. Additionally, for any graph count within this range, KNIGCN model's total execution time is shorter than that of making method of node criticality labels. The time difference between the two methods increases as the graph count rises, with the total execution time of making method of node criticality labels reaching approximately 30 times that of the KNIGCN model at a graph count of 500. These results demonstrate that the KNIGCN model can maintain high accuracy in critical node identification while significantly enhancing speed and efficiency, even when dealing with dynamic UAV swarm network topologies.

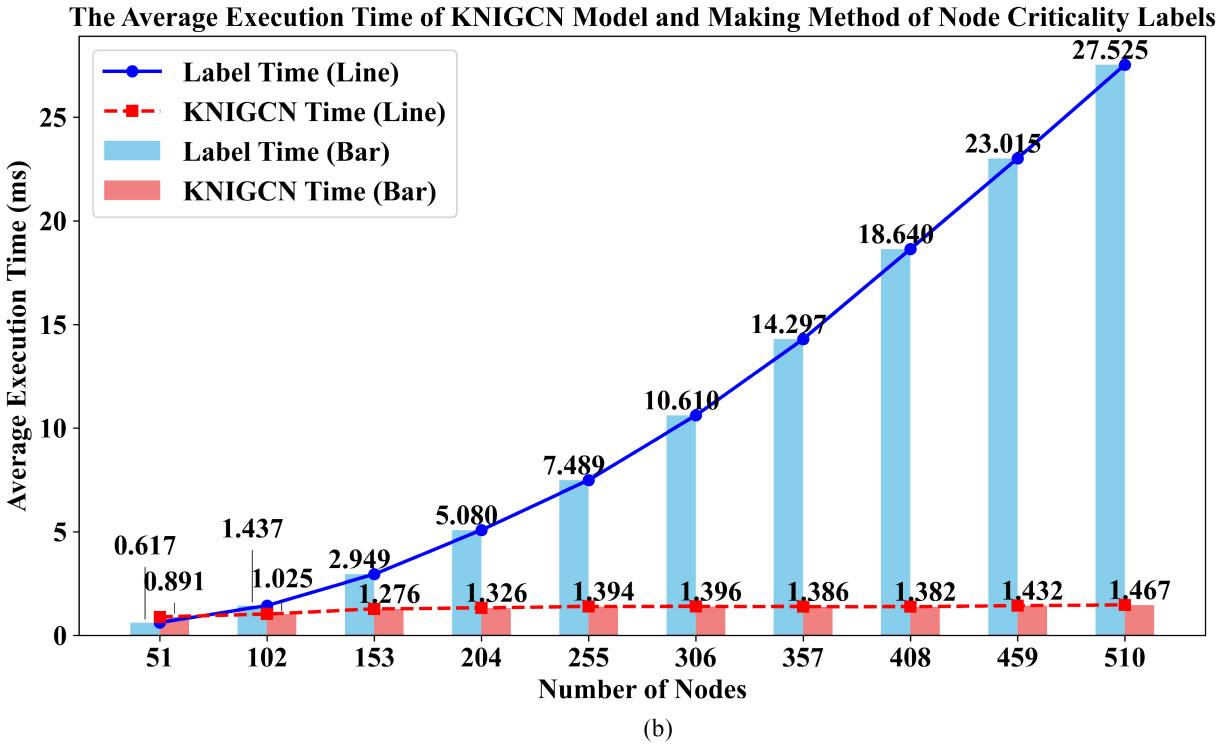
V. Conclusion

In this article, we propose KNIGCN, a GCN-based model for identifying key nodes in UAV swarm network. The approach begins by constructing a network topology based on the FANET communication architecture and Dijkstra's algorithm. A traffic statistics-based method is then used to label the criticality of each node within the UAV swarm network. KNIGCN is subsequently trained on the input topology to predict the criticality scores of nodes, ranking them to identify a specified number of key nodes. Simulation results validate the effectiveness of KNIGCN, achieving over 96% accuracy in identifying key nodes for six test UAV swarm networks when key nodes constitute 0.1 of the total nodes. Compared with baseline methods, our criticality labeling approach provides a more reasonable and applicable assessment of node importance, as the selected nodes contribute more significantly to communication performance. Furthermore, for the large-scale or dynamically evolving UAV swarm network, KNIGCN improves identification speed and efficiency considerably while ensuring high accuracy in critical node identification.

In conclusion, there are several areas for further research and development in this work. First, the proposed node criticality labeling method can be improved by incorporating additional influencing factors, such as battery status, node functions [33], and geographical location [34], to achieve a more comprehensive evaluation of node criticality in UAV swarm networks. We have already outlined plans to enhance



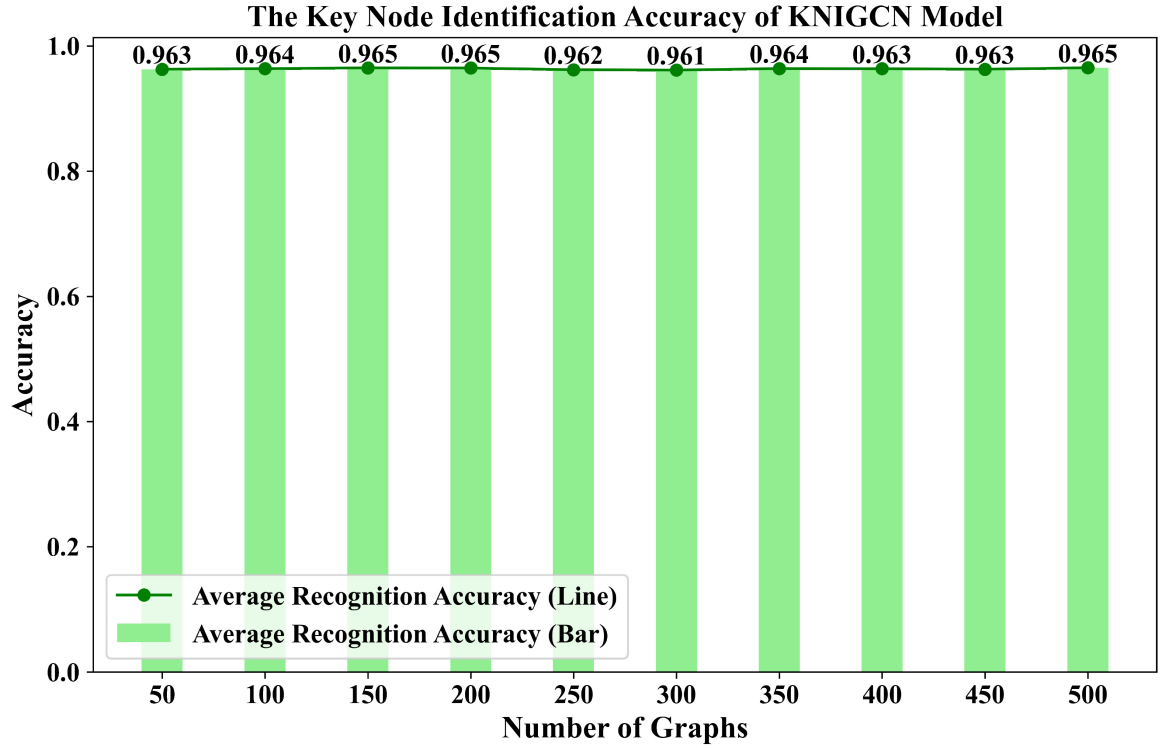
(a)



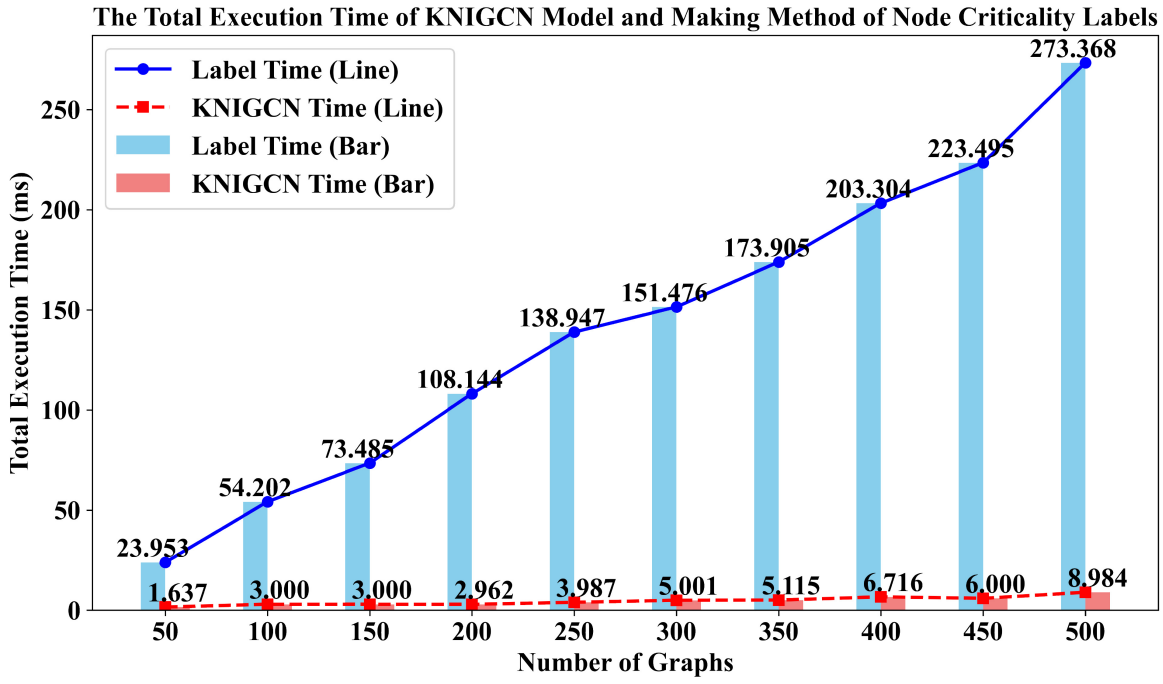
(b)

Fig. 12. Average identification accuracy of KNIGCN model and the comparison of the average execution time by KNIGCN model and making method of node criticality labels with different number of nodes. (a) Average identification accuracy of KNIGCN model. (b) Average execution time of KNIGCN model and making method of node criticality label.

node criticality assessment. The first step involves incorporating battery status as a feature attribute of UAV nodes and combining it with network communication performance to evaluate node importance more holistically. The second step considers introducing geographical location factors into the UAV swarm model to resimulate the network's communication



(a)



(b)

Fig. 13. Average identification accuracy of KNIGCN model and the comparison of the total execution time by KNIGCN model and making method of node criticality labels with different number of graphs. (a) Average identification accuracy of KNIGCN model. (b) Total execution time of KNIGCN model and making method of node criticality label.

conditions. This includes modeling geographical conditions and integrating routing protocols related to geographical distributions. After the simulation, criticality label values will be computed based on the resulting communication traffic. Additionally, it is worth considering further testing, validation, and improvement of the KNIGCN critical node identification algorithm in real-world scenarios. By incorporating the designed UAV swarm network simulation parameters and selecting suitable hardware equipment, we aim to build a physical UAV swarm network. Implementing our method in this

environment will enable us to test and verify its performance and applicability outside the simulation framework. This effort will contribute to refining the KNIGCN model and ensuring its effectiveness in meeting practical critical node identification requirements. Finally, although we have demonstrated that our method achieves strong performance in identifying critical nodes within large-scale, complex, and dynamic UAV swarm networks, it may still face limitations when applied to networks with higher node density or extremely dynamic topologies. These limitations include insufficient computational resources and decreased identification efficiency, which require further optimization and resolution in future research.

REFERENCES

[1] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, and M. Pan, "IoT enabled UAV: Network architecture and routing algorithm," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3727–3742, Apr. 2019.

[2] S. Liu and Y. Bai, "Multiple UAVs collaborative traffic monitoring with intention-based communication," *Comput. Commun.*, vol. 210, pp. 116–129, Oct. 2023.

[3] P. Zambrano, F. Calderon, H. Villegas, J. Paillacho, D. Pazmiño, and M. Realpe, "UAV remote sensing applications and current trends in crop monitoring and diagnostics: A systematic literature review," in *Proc. IEEE 13th Int. Conf. Pattern Recognit. Syst. (ICPRS)*, 2023, pp. 1–9.

[4] Z. Xiaoning, "Analysis of military application of UAV swarm technology," in *Proc. 3rd Int. Conf. Unmanned Systems (ICUS)*, 2020, pp. 1200–1204.

[5] P. Basu and J. Redi, "Movement control algorithms for realization of fault-tolerant Ad Hoc robot networks," *IEEE Netw.*, vol. 18, no. 4, pp. 36–44, Jul./Aug. 2004.

[6] X. Zhou, L. Yang, L. MA, and H. He, "Towards secure and resilient unmanned aerial vehicles swarm network based on blockchain," *IET Blockchain*, vol. 4, no. 1, pp. 483–493, 2024.

[7] M. Lalou, M. A. Tahraoui, and H. Kheddouci, "The critical node detection problem in networks: A survey," *Comput. Sci. Rev.*, vol. 28, pp. 92–117, May 2018.

[8] K.-Y. Tsao, T. Girdler, and V. G. Vassilakis, "A survey of cyber security threats and solutions for UAV communications and flying Ad-Hoc networks," *Ad Hoc Netw.*, vol. 133, Aug. 2022, Art. no. 102894.

[9] Q. Wang, D. Zhuang, and H. Xie, "Identification of influential nodes for drone swarm based on graph neural networks," *Neural Process. Lett.*, vol. 53, pp. 4073–4096, Jul. 2021.

[10] D. Zhang, Y. Liu, X. Liang, and Z. Shi, "An blind identification algorithm of key nodes for UAV swarm communication network," in *Proc. IEEE 5th Int. Conf. Civil Aviat. Safety Inf. Technol. (ICCASIT)*, 2023, pp. 1186–1192.

[11] Z. Niu, T. Ma, N. Shu, and H. Wang, "Identification of critical nodes in Ad Hoc network based on the analysis of network partition," in *Proc. IEEE 19th Int. Conf. Commun. Technol. (ICCT)*, 2019, pp. 1048–1052.

[12] D. Chen, L. Lü, M.-S. Shang, Y.-C. Zhang, and T. Zhou, "Identifying influential nodes in complex networks," *Physica A, Stat. Mech. Appl.*, vol. 391, no. 4, pp. 1777–1787, 2012.

[13] Y.-M. Wang, C.-S. Pan, B. Chen, and D.-P. Zhang, "Method of key node identification in command and control networks based on level flow betweenness," *Int. J. Comput. Sci. Eng.*, vol. 19, no. 3, pp. 368–375, 2019.

[14] Y. Yu and S. Fan, "Node importance measurement based on the degree and closeness centrality," *J. Inf. Comput. Sci.*, vol. 12, no. 3, pp. 1281–1291, 2015.

[15] Y.-J. Tan, J. Wu, and H.-Z. Deng, "Evaluation method for node importance based on node contraction in complex networks," *Syst. Eng.-Theory Pract.*, vol. 26, no. 11, pp. 79–83, 2006.

[16] M. Jorgić, I. Stojmenović, M. Hauspie, and D. Simplotryl, "Localized algorithms for detection of critical nodes and links for connectivity in Ad Hoc networks," in *Proc. Mediterr. Ad Hoc Netw. Workshop*, 2011, pp. 1–12.

[17] F. Morone and H. A. Makse, "Influence maximization in complex networks through optimal percolation," *Nature*, vol. 524, no. 7563, pp. 65–68, 2015.

[18] X. Wen, C. Tu, M. Wu, and X. Jiang, "Fast ranking nodes importance in complex networks based on LS-SVM method," *Physica A, Stat. Mech. Appl.*, vol. 506, pp. 11–23, Sep. 2018.

[19] G. Zhao, P. Jia, C. Huang, A. Zhou, and Y. Fang, "A machine learning based framework for identifying influential nodes in complex networks," *IEEE Access*, vol. 8, pp. 65462–65471, 2020.

[20] H. Huang, L. Sun, B. Du, C. Liu, W. Lv, and H. Xiong, "Representation learning on knowledge graphs for node importance estimation," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discov. Data Min.*, 2021, pp. 646–655.

[21] M. Zhang, X. Wang, L. Jin, M. Song, and Z. Li, "A new approach for evaluating node importance in complex networks via deep learning methods," *Neurocomputing*, vol. 497, pp. 13–27, Aug. 2022.

[22] G. Zhao, P. Jia, A. Zhou, and B. Zhang, "InfGCN: Identifying influential nodes in complex networks with graph convolutional networks," *Neurocomputing*, vol. 414, pp. 18–26, Nov. 2020.

[23] X. Chen, J. Tang, and S. Lao, "Review of unmanned aerial vehicle swarm communication architectures and routing protocols," *Appl. Sci.*, vol. 10, no. 10, p. 3661, 2020.

[24] M. A. Khan, A. Safi, I. M. Qureshi, and I. U. Khan, "Flying Ad-Hoc networks (FANETs): A review of communication architectures, and routing protocols," in *Proc. 1st Int. Conf. Latest Trends Electr. Eng. Comput. Technol. (INTELLECT)*, 2017, pp. 1–9.

[25] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying Ad-Hoc networks (FANETs): A survey," *Ad Hoc Netw.*, vol. 11, no. 3, pp. 1254–1270, 2013.

[26] S. Yehui, D. Guoru, S. Jiachen, L. Jinghua, and X. Yitao, "Topology tracking of dynamic UAV wireless networks," *Chin. J. Aeronaut.*, vol. 35, no. 11, pp. 322–335, 2022.

[27] C. Xia, Y. Zhang, Y. Liu, K. Lin, and J. Chen, "Path planning and energy flow control of wireless power transfer for sensor nodes in wireless sensor networks," *Turk. J. Electr. Eng. Comput. Sci.*, vol. 26, no. 5, pp. 2618–2632, 2018.

[28] S. Thomas, I. Gayathri, and A. Raj, "Joint design of Dijkstra's shortest path routing and sleep-wake scheduling in wireless sensor networks," in *Proc. Int. Conf. Energy, Commun., Data Anal. Soft Comput. (ICECDS)*, 2017, pp. 981–986.

[29] M. Mathapati, P. Nandihal, P. Mishra, and V. Kotagi, "Improvisation of QoS in SDN-frame work for UAV networks using Dijkstra shortest path routing algorithm," in *Proc. Int. Conf. Ambient Intell., Knowl. Informat. Ind. Electron. (AIKIIE)*, 2023, pp. 1–7.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[31] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011, pp. 315–323.

[32] M. Kitsak et al., "Identification of influential spreaders in complex networks," *Nat. Phys.*, vol. 6, no. 11, pp. 888–893, 2010.

[33] S. Hayat, E. Yanmaz, and C. Bettstetter, "Experimental analysis of multipoint-to-point UAV communications with IEEE 802.11 n and 802.11 AC," in *Proc. IEEE 26th Annu. Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, 2015, pp. 1991–1996.

[34] L. Gupta, R. Jain, and G. Vaszkun, "Survey of important issues in UAV communication networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1123–1152, 2nd Quart., 2015.

Qixun Ai received the B.S. degree from the College of Information and Communication Engineering, Harbin Engineering University, Harbin, Heilongjiang, China, in 2023, where he is currently pursuing the master's degree in electronic communication engineering.

His research interests include deep learning, graph analysis, and signal processing.

Changbo Hou received the B.S, M.S., and Ph.D. degree from Harbin Engineering University, Harbin, Heilongjiang, China, in 2008, 2011, and 2021, respectively.

He is currently a Professor with the College of Information and Communication Engineering, Harbin Engineering University, where he is also the Deputy Director of Heilongjiang Provincial Key Laboratory of Multidisciplinary Collaborative Cognitive Artificial Intelligence Technology and Application. He had published more than 30 international peer-reviewed journal/conference papers, such as the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON RELIABILITY, and MWSCAS. His research interests include deep learning, wideband signal processing, image processing, cognitive radio, and software defined radio.

Zhichao Zhou (Graduate Student Member, IEEE) received the B.S. degree in electronic information engineering from Harbin Engineering University, Harbin, Heilongjiang, China, in 2021, where he is currently pursuing the doctor's degree in electronic communication engineering.

His research interests include deep learning, graph analysis, and signal processing.

Xiangyu Wu received the B.S. degree from Northeast Forestry University, Harbin, Heilongjiang, China, in 2021. He is currently pursuing the doctor's degree in electronic communication engineering from Harbin Engineering University, Harbin, Heilongjiang, China.

His research interests include deep learning, communication countermeasure, and signal processing.

Zhen Song received the B.S. degree from the College of Information and Communication Engineering, Harbin Engineering University, Harbin, Heilongjiang, China, in 2024, where he is currently pursuing the doctor's degree in electronic communication engineering.

His research interests include deep learning, communication countermeasure, and signal processing.