Research on Self-Learning Method for Key Nodes Identification in Heterogeneous Networks

Luyao Wang
Zhiwei Yang
Kewei Yang
College of Systems Engineering
National University of Defense Technology
Changsha, Hunan, P. R. China

Libin Chen
College of Intelligence Science and Technology
National University of Defense Technology
Changsha, Hunan, P. R. China

Identifying key nodes in heterogeneous networks is both theoretically important and practically valuable. Traditional methods require precise parameters and constraints, limiting adaptability and autonomy. To address this, we propose the deep reinforcement learning–based heterogeneous network key nodes identification (DRLKHN) method, a self-learning method for identifying key nodes. DRLKHN autonomously learns strategies for identifying key nodes, utilizing a graph convolution network (GCN) for feature extraction and designing action and state space vectors. Experimental results show that DRLKHN outperforms traditional methods like high degree adaptive (HDA), high eigenvector adaptive (HEA), high closeness adaptive (HCA) and high PageRank adaptive (HPA) in simulated networks. In the real-world force, intelligence, networking and C2 (FINC) network, DRLKHN improves performance by 28.6%, 32.2%, 12.7% and 36.3% over HDA, HEA, HCA and HPA, respectively. Despite its relatively high time complexity, DRLKHN effectively integrates the GCN and reinforcement learning to manage complex relationships in graph data, providing intelligent decision support for identifying key nodes in real networks.

Keywords: heterogeneous network; key nodes identification; graph representation; deep reinforcement learning

1. Introduction

A network, or graph, is a general data structure for describing complex interactive systems, such as the internet, social media, transportation networks, power grids, food webs, bioanalytics networks and others. In a complex network, each node plays a specific role, but

https://doi.org/10.25088/ComplexSystems.34.4.425

the relative importance of nodes varies greatly. Some nodes are in a hub position or contribute significantly to the overall network performance, so these nodes are more important than others and are called key nodes. Finding an optimal set of key nodes in complex networks has been a longstanding problem in network science with many real-world applications, including terrorist networks [1], transportation [2], protein networks [3], epidemic spreading network [4] and rumor diffusion network [5].

The method of identifying key nodes in networks has been widely considered by the academic community, and new methods are being constantly developed. Identifying key nodes can be mainly divided into neighborhood-based centrality, path-based centrality and random walk–based methods. However, there are some shortcomings in these methods. Traditional modeling and solving methods require precise system parameters and constraints, and the quality of the solution depends on the optimization model. As the constraints change, it is necessary to constantly update and optimize the model, resulting in poor adaptability and lack of decision-making autonomy.

Reinforcement learning (RL) is a rapidly developing artificial intelligence (AI) technology that adapts to changes in dynamic environments. RL has been successfully applied in areas like manufacturing systems [6], transportation route selection [7] and portfolio optimization [8]. However, RL struggles with high-dimensional continuous state spaces due to limited and discrete state spaces in these problems. Deep reinforcement learning (DRL), including methods like deep Q-network (DQN) [9, 10], DRL with double Q-learning [11] and dueling DQN (DDQN) [12], addresses these issues, performing better in dynamic and continuous state problems.

The focus of the previous review on key nodes identification methods is on utilizing network topology and certain attribute information, which mainly applies to general scenarios but is not suitable for complex heterogeneous networks. In heterogeneous networks, both network topology and node heterogeneity must be considered. Thus, this paper proposes the deep reinforcement learning–based heterogeneous network key nodes identification (DRLKHN) method. Unlike traditional methods, DRLKHN does not rely on precise system parameters or complex constraints, making it more adaptable and versatile. The main contributions of this paper follow.

- To address the representation of heterogeneous networks, we propose a method based on the graph convolutional network (GCN) to map graph data into low-dimensional vectors, enabling the capture of node and edge characteristics and their relationships for further analysis.
- To identify key nodes in heterogeneous networks, we leverage the perceptual and decision-making capabilities of DRL, introducing the DRLKHN method for autonomous key nodes identification.

- To solve the key nodes identification problem, we represent each potential action (node, a) and the remaining state (graph, s) as feature vectors, which are used as inputs to the DQN, defining the state and action spaces of the model.

This paper is organized as follows. Section 2 reviews related work on key nodes identification. Section 3 defines the key nodes identification problem in heterogeneous networks. Section 4 presents the framework for key nodes identification using deep reinforcement learning. Section 5 introduces the key nodes identification algorithm based on deep reinforcement learning. Section 6 demonstrates the effectiveness of the proposed method with examples and analyzes the time complexity of DRLKHN. Finally, Section 7 concludes the study.

2. Related Work

Over the past few decades, the identification of key nodes in a network has been a topic of extensive academic research, with continuous innovations in identification methods. These methods can mainly be divided into neighborhood-based centrality methods, path-based centrality methods, random walk–based methods and machine learning–based methods. The latter category not only represents an innovation based on the former but also a conceptual revolution.

2.1 Neighborhood-Based Centrality Methods

Intuitively, the importance of a node is related not only to its own information but also to its neighbors. Neighborhood-based centrality measures mainly evaluate a node by the number of its direct or indirect neighbors. The more neighbors a node has, the more important it is considered. Degree centrality [13] is the most straightforward and simple method for measuring node centrality. The degree of a node is represented as the number of first-order neighbors, and its normalized value is the degree centrality. Since degree centrality only calculates node centrality based on first-order neighbors, Chen et al. [14] proposed an improvement: semi-local centrality, which evaluates the importance of a node using both first- and second-order neighbors. The clustering coefficient is a measure of the closeness of links between a node's neighbors. The degree and clustering coefficient-based centrality (DCC) method [15] combines node degree and clustering coefficient to calculate node centrality. Opsahl et al. [16] combined degree centrality with weighted networks, proposing the generalized degree centrality algorithm, which evaluates node centrality based on both node degree and edge weights. Zhao et al. [17] evaluate the node centrality in directed weighted networks using three

indicators: node degree, edge weight and edge direction. Since degree centrality does not consider the attribute vectors of nodes, eigenvector centrality [18, 19] fully considers the neighborhood and the attribute vectors of neighbors. The degree centrality of a node does not necessarily mean it has a high eigenvector centrality; if the first-order neighbors of a node have low eigenvector centralities, then its eigenvector centrality will also be low. Neighborhood-based centrality measures are simple and intuitive with low computational complexity, but their main disadvantage is that they only assess the importance of nodes from a local perspective, ignoring the impact of nodes on the entire network.

2.2 Path-Based Centrality Methods

Path-based centrality measures overcome some of the limitations of neighborhood-based centrality by considering the importance of a node's position in the network. Major methods include Katz centrality [20], closeness centrality [13] and betweenness centrality [21]. Betweenness centrality, first introduced by L. C. Freeman [21], is defined as the normalized value of the number of shortest paths passing through a node. This measure is suitable for identifying key nodes in networks with traffic flow, such as power networks and infrastructure attack-defense [22] or congestion node expansion in transportation networks [23]. Closeness centrality [13] is the inverse of the average distance to other nodes, used to evaluate how close a node is to others. Compared to betweenness centrality, closeness centrality is closer to the geometric center of the network. Katz centrality [20] assigns different weights to paths of different lengths, calculating node centrality based on this weighted sum, with shorter paths being considered more important. Hage and Harary [24] defined eccentricity centrality as the maximum distance from a node to all other nodes, using extreme values to calculate node centrality. This method has a relatively limited application scenario. Subgraph centrality [25] measures node centrality by the weighted sum of closed path lengths, similar to closeness centrality, where shorter path lengths are considered more important. While path-based centrality measures can assess node importance from a global perspective, they are limited to connected graphs and are not suitable for calculating centrality in sparse or disconnected graphs.

2.3 Random Walk–Based Methods

Random walk methods are a network analysis approach based on random processes. The basic idea is to randomly select a node as the starting point, then walk through the network following certain rules, recording the visit count for each node. By analyzing the visit counts,

key nodes in the network can be identified. Random walk methods are an improvement upon path-based centrality methods, implementing a random sampling process on graph information based on paths. Page [26] first applied the random walk process to internet page ranking in 1999, modeling the internet as a directed graph, where the more hyperlinks a webpage has, the more important it is considered, or if a webpage is linked by more influential networks, it is also deemed more important. The hypertext-induced topic search (HITS) algorithm [27] evaluates webpage importance using content authority (Authority) and link authority (Hub). Content authority corresponds to the number of inbound links, with more citations indicating higher authority. Link authority corresponds to the number of outbound links, with higher link authority associated with a greater number and quality of outbound links. The HITS algorithm has been widely used in search engines, online advertising, e-commerce and other fields. The LeaderRank [28] algorithm is an improvement on the PageRank algorithm, similar in that it is a network-based node ranking algorithm. However, LeaderRank improves connectivity by adding a background node that is connected to all nodes in the network, better reflecting the relationship between webpages. The random walk with restart (RWR) algorithm [29, 30] is an enhanced random walk method that introduces a restart probability during the random walk process. It combines transition probability and restart probability, considering both local and global information. Compared to ordinary random walk algorithms, RWR can better capture the relationships between nodes.

2.4 Machine Learning–Based Methods

Machine learning–based key nodes identification methods are represented by the graph neural network (GNN), which accurately represents nodes in a network to better handle downstream tasks, such as node classification and graph classification. For key nodes identification, the node deletion method is often combined with GNN to judge node importance, using the change in network performance before and after a node is deleted as the basis for training the GNN. Fan et al. [31] proposed the FINDER method, which uses GNN and DQN to identify key nodes in the network. Experiments have shown that this method can disrupt the largest connected component in the network at the fastest speed. Zeng et al. [32] proposed the SHATTER model, using the node deletion method to identify key nodes in a heterogeneous combat network (HCN). The MINER framework [33] converts multiplex network degradation strategies into an encoding-decoding process using deep network representation learning, with attention mechanisms and RL evaluating network actions. While

https://doi.org/10.25088/ComplexSystems.34.4.425

research typically targets high-value node attack sequences, Chen [34] highlighted the vulnerability of edges, proposing the use of DRL to identify high-value edge attack sequences.

In recent years, GNN have received widespread attention. Some representative models follow. Kipf et al. [35], inspired by the convolutional neural network (CNN), proposed the graph convolutional network (GCN), allowing graph data to undergo convolution operations in the spatial domain, similar to the CNN. The GCN has achieved good results in node classification tasks but faces challenges when dealing with large-scale graphs. GraphSAGE [36] adopts a random sampling method to embed nodes, which is more efficient than the full-graph training approach of the GCN, making GraphSAGE more suitable for handling large-scale graph data and improving scalability. The graph attention network (GAT) [37] introduces attention mechanisms into node representation models, where different weights are assigned to the aggregation of node attributes, allowing the learning of updated weights to more accurately capture important information. Additionally, models such as GIN [38], graph u-net [39], graph-transformer [40], GCNII [41] and ST-GCN [42] have been proposed.

The classical method of key nodes identification is summarized in Table 1. The field of key nodes identification in complex networks is relatively mature, but there are still some shortcomings: research on

Category	Description	Classical Method
Neighborhood-based centrality methods	It is evaluated by the number of direct or indirect neighbors, and the more neighbors it has, the more important the node is.	Degree centrality
Path-based centrality methods	The importance of the node's position in the network is used as the centrality index of the node.	Katz centrality; closeness centrality; betweenness centrality
Closeness centrality	A node is randomly selected as the starting point, and key nodes are identified by analyzing the visit data as it moves through the network.	PageRank
Betweenness centrality	The node deletion method is often combined with GNN to judge node importance, using the change in network performance before and after a node is deleted as the basis for training GNN	FINDER; SHATTER; MINER

Table 1. The classical method of key nodes identification.

key nodes identification in heterogeneous networks is lacking. Heterogeneous networks are more complex than homogeneous networks. Due to the different types of nodes and links in heterogeneous networks, it is difficult to accurately determine the importance of a node by relying solely on topology and vague attribute inputs.

3. Problem Description

Before we begin our problem description, we need to clarify some concepts.

Key nodes identification. Finding an optimal set of nodes whose activation (or removal) would maximally enhance (or degrade) certain network functionality. (A key node can also be called the influential node, vital node, critical node, etc.)

Network disintegration. The process of disrupting the structure of a network, weakening its performance and preventing activities on the network by removing nodes or edges.

There are differences and connections between key nodes identification and network disintegration. As shown in Figure 1, our paper involves the intersection of the two. That is, finding an optimal set of nodes whose removal would maximally degrade certain network functionality.

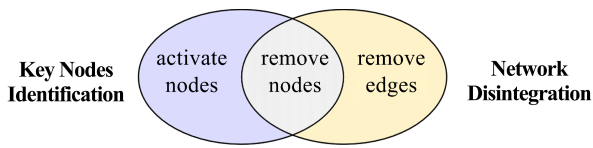

Figure 1. Venn diagram between key nodes identification and network disintegration.

Heterogeneous network. A heterogeneous network can be represented as $G = (V, E)$, where V represents the set of nodes and E represents the set of edges between nodes. The node type mapping function is defined as $\varphi : V \to V_{\text{type}}$, and the edge type mapping function is defined as $\varphi : E \to E_{\text{type}}$, in which V_{type} and E_{type} are sets of labels identifying unique types or roles for nodes and edges, respectively. The network is called a heterogeneous network if it contains multiple node types $|V_{\text{type}}| > 1$ or edge types $|E_{\text{type}}| > 1$.

Given the heterogeneous network $G = (V, E)$, where V is the set of nodes and E is the set of edges, the objective of key nodes identification is to minimize the network robustness based on a node removal strategy. In current research, scholars extensively study how to measure network robustness, including metrics such as network diameter,

https://doi.org/10.25088/ComplexSystems.34.4.425

network connectivity, network efficiency, network fragmentation and network structural entropy. Among these, network connectivity is the most widely used indicator. Network connectivity metrics typically include the number of connected components, pairwise connectivity and the largest connected component (LCC) size [43]. Specifically, the LCC size is a commonly studied metric, representing the number of nodes in the largest subgraph in the current network. A predefined metric for the size of the LCC is denoted as κ. The objective of key nodes identification is to design a node removal strategy, that is, an order of node removal that minimizes accumulated normalized connectivity (ANC) based on LCC size:

$$\mathrm{ANC}(v_1, v_2, \ldots, v_N) = \frac{1}{N}\sum_{k=1}^{N}\frac{\kappa(G\backslash\{v_1, v_2, \ldots, v_k\})}{\kappa(G)} \tag{1}$$

where N is the number of nodes in G. $\kappa(G)$ represents the LCC size of the initial graph G. $\kappa(G\backslash\{v_1, v_2, \ldots, v_k\})$ represents the LCC size after removing nodes $\{v_1, v_2, \ldots, v_k\}$ from G in order. As shown in Figure 2, the ANC can be seen as an estimate of the area under the normalized connectivity versus attack intensity curve. The horizontal axis represents the attack intensity, denoted by i/N, and the vertical axis represents the normalized connectivity,

$$\frac{\kappa(G\backslash\{v_1, v_2, \ldots, v_i\})}{\kappa(G)}.$$

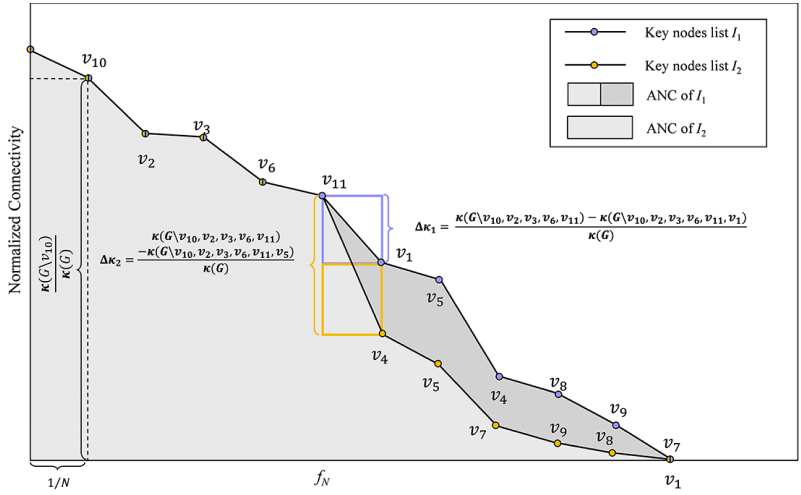

Figure 2. Effect of lists of potential key nodes I_1 and I_2 on the ANC of G.

Two lists of potential key nodes I_1 and I_2 are shown in Figure 2:

$$I_1 = (v_{10}, v_2, v_3, v_6, v_{11}, v_1, v_5, v_4, v_8, v_9, v_7) \tag{2}$$

$$I_2 = (v_{10}, v_2, v_3, v_6, v_{11}, v_4, v_5, v_7, v_9, v_8, v_1) \tag{3}$$

It can be observed that removing the node list I_2 results in a smaller ANC value than removing the node list I_1. The decrease in ANC, represented by the area shaded in dark gray in the graph, indicates that I_2 is the more critical list of nodes. The difference between the two node lists occurs after removing node v_{11}. For node list I_1, the network connectivity decreases as a result of removing node v_1, represented as:

$$\Delta\kappa_1 = \frac{1}{\kappa(G)}\kappa(G\backslash\{v_{10}, v_2, v_3, v_6, v_{11}\}) - \kappa(G\backslash\{v_{10}, v_2, v_3, v_6, v_{11}, v_1\}). \tag{4}$$

For node list I_2, the network connectivity decreases as a result of removing node v_4, represented as:

$$\Delta\kappa_2 = \frac{1}{\kappa(G)}\kappa(G\backslash\{v_{10}, v_2, v_3, v_6, v_{11}\}) - \kappa(G\backslash\{v_{10}, v_2, v_3, v_6, v_{11}, v_4\}). \tag{5}$$

Since $\Delta\kappa_1 < \Delta\kappa_2$, removing node v_4 has a greater impact on the network connectivity. The ANC is sensitive to the sequence of node removals. For example, removing nodes v_1 and v_4 in different orders (as in I_1 and I_2) results in distinct impacts on network connectivity ($\Delta\kappa_1 \neq \Delta\kappa_2$), highlighting the necessity of modeling this problem as a sequential decision-making process.

Based on our description, the problem of identifying key nodes in a heterogeneous network can be regarded as a combinational optimization problem. The model for identifying key nodes in a heterogeneous network can be represented by:

$$\min \text{ANC}(v_1, v_2, \ldots, v_N)$$
$$\text{subject to}: \begin{cases} \tilde{V} = (v_1, v_2, \ldots, v_N) \\ |\tilde{V}| = f_N|V| \end{cases} \tag{6}$$

where $\min \text{ANC}(v_1, v_2, \ldots, v_N)$ represents the optimization objective. The constraint is the attack intensity f_N, which is the proportion of attacked nodes to the total number of nodes. $\tilde{V} = (v_1, v_2, \ldots, v_N)$ denotes a sequence of key nodes to be identified.

https://doi.org/10.25088/ComplexSystems.34.4.425

4. Design of a Key Nodes Identification Framework Based on Deep Reinforcement Learning

This paper reframes the problem of identifying key nodes in heterogeneous networks as a learning problem, where the node selection process is treated as an agent's decision-making task. DRL is an efficient method to solve the sequential decision problem [9, 44]. Therefore, we use DRL to realize the self-learning identification of key nodes in a heterogeneous network.

The process of self-learning to identify key nodes in a heterogeneous network can be viewed as a Markov decision process: generating a series of states, actions and rewards through interaction with the environment. Therefore, we design a DRL-based heterogeneous network key nodes identification (DRLKHN) framework. DRLKHN utilizes the perception and decision-making abilities of DRL to self-learn key nodes identification strategies for heterogeneous networks. DRLKHN is based on a combined framework of the GCN and DQN algorithms; see Figure 3.

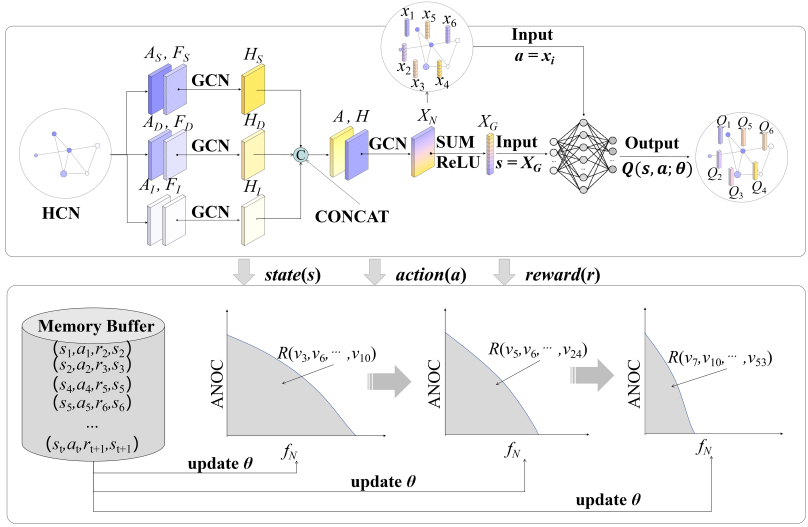

Figure 3. DRLKHN framework. Assuming the heterogeneous network consists of three types of nodes.

The GCN [45] is a specialized type of neural network designed to extract features from graph data, which consists of nodes and edges representing unstructured data. Traditional neural networks, such as the convolutional neural network (CNN) and the recurrent neural network (RNN), are effective at capturing features from structured data like images and videos. However, they are not well suited for extracting features from unstructured graph data. The GCN, on the

other hand, leverages both node features and the graph structure to capture feature representations, reflecting the unique properties of individual nodes as well as the interactions between nodes. The features extracted by a GCN can improve performance in subsequent graph analysis tasks. In this paper, we use graph data to record information in a heterogeneous network, where the GCN effectively characterizes both node attributes and internode interactions.

Thus, within the DRLKHN framework, a GCN serves as the feature extractor for heterogeneous networks, mapping the input graph data into low-dimensional vectors. Each possible action (node, a) and state (remaining graph, s) is represented as a set of feature vectors that capture the characteristics related to these actions and states. These feature vectors are then utilized in the DRL process to estimate the future expected return $Q(s, a)$ of actions with respect to states.

The DQN algorithm is employed to learn the DRL component of the network parameters. DQN is a DRL algorithm that builds upon Q-learning by incorporating the approximation of the action-value function $Q(s, a; \theta)$ into the Q-learning framework using deep neural networks. This approach replaces the traditional method of storing $Q(s, a)$ in a Q-table, enabling DQN to handle more complex environments with larger state spaces and perform well in solving problems with discrete action spaces. To improve training efficiency and stability, DQN utilizes experience replay and fixed target networks, which help the algorithm converge more reliably toward the optimal Q value.

Based on the DRLKHN framework, this paper designs the state space, action space, reward function and loss function involved in the framework, making it suitable for self-learning to identify key nodes in a heterogeneous network.

4.1 Action Space Setting

The action space contains all the actions that the agent may take in self-learning tasks. In this paper, actions refer to the nodes selected by the agent for attack. In the DRLKHN framework, a GCN is introduced to extract features from the graph data of the heterogeneous network. Therefore, the action space can be designed as:

$$\mathbb{A} = \{a \mid a = x_i\} \tag{7}$$

where x_i represents the representation vector of node i. The calculation process of x_i follows.

First, the heterogeneous network is divided into different categories based on node type j, and adjacency matrices A_j are computed for each category. Then, a node feature matrix F_j is constructed based on node features. Next, a GCN is applied to extract features from the

https://doi.org/10.25088/ComplexSystems.34.4.425

categories of nodes, resulting in H_j:

$$H_j = \mathrm{GCN}(F_j, A_j) = \sigma\left(\hat{D}_j^{-1/2}\hat{A}_j\hat{D}_j^{-1/2}F_jW_1\right) \tag{8}$$

where $\sigma(\cdot)$ represents the activation function $\hat{A}_j = A_j + I_j$, I_j is the identity matrix with the same structure as A_j, $\hat{D}_j$ is the degree matrix of $\hat{A}_j$, and W_1 is the weight parameter of the GCN network.

Finally, to extract information about the connections between different types of nodes, the concatenated matrix H, obtained by concatenating H_j (as shown in equation (9)), is combined with the adjacency matrix A of the entire graph and further processed through the GCN for feature extraction, resulting in the node representation matrix X_N. The i^{th} row of x_i of X_N represents the representation vector of node i, as shown in equation (10):

$$H = [H_1 \,\|\, H_2 \,\|\, \cdots \,\|\, H_j] \tag{9}$$

$$X_N = \mathrm{GCN}(H, A) = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}HW_1\right) \tag{10}$$

where $[\cdot \,\|\, \cdot]$ represents the concat operation $\hat{A} = A + I$, I is the identity matrix with the same structure as A, $\hat{D}$ is the degree matrix of $\hat{A}$, and W_1 is the weight parameter of the GCN network.

4.2 State Space Setting

The state space serves as the basis for action selection. We define a state as the remaining network after the agent attacks a node. Building upon the extraction of node features using a GCN, the state space can be designed as:

$$\mathbb{S} = \{s \mid s = X_G\} \tag{11}$$

where X_G is the representation vector of the heterogeneous network graph data. The computation process of X_G is as follows: the representation vectors of all nodes are summed up one by one to obtain the representation vector of the entire heterogeneous network:

$$X_G = \sigma\left(W_3 \cdot W_2 \cdot \sum_i^N x_i\right) \tag{12}$$

where the parameters W_2 and W_3 are learnable matrices, following the standard formulation of weight matrices in a GNN [35] and DQN framework [10]. Algorithm 1 shows the action and state space setting algorithm.

Algorithm 1. Action and state space setting algorithm.

Input:
$G = (V, E)$;
adjacency matrix A_j
feature matrix F_j
training parameters W_1, W_2, W_3

Output:
node representation vector x_i
node representation matrix X_N
graph representation matrix X_G

1. **for each** node type $j \in V_{\text{type}}$ **do**
2. add self-connections: $\hat{A}_j \leftarrow A_j + I_j$
3. compute degree matrix: $\hat{D}_j \leftarrow \text{diag}\left(\sum_k \hat{A}_{j,ik}\right)$
4. for each node type j, apply a GCN layer to its adjacency matrix A_j and feature matrix F_j: $H_j \leftarrow \sigma\left(\hat{D}_j^{-1/2}\hat{A}_j\hat{D}_j^{-1/2}F_jW_1\right)$
   σ: LeakyReLU activation; $W_1 \in \mathrm{R}^{d\times d}$
5. **end for**
6. concatenate type-specific representations: $H \leftarrow \left[H_1 \,\|\, H_2 \,\|\, \cdots \,\|H_{|V_{\text{type}}|}\right]$
7. global GCN convolution
   $\hat{A} \leftarrow A + I$
   $\hat{D} \leftarrow \text{diag}\left(\sum_k \hat{A}_{ik}\right)$
   $X_N \leftarrow \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}HW_1\right)$
8. graph representation aggregation
   $X_G \leftarrow \sigma\left(W_3 \cdot W_2 \cdot \sum_{i=1}^{N} x_i\right)$
   x_i: i^{th} row of X_N; W_2, $W_3 \in \mathrm{R}^{d\times d}$
9. **return** $\{x_i\}$, X_N, X_G

4.3 Reward Function Setting

The goal of DRL is to maximize cumulative rewards. Therefore, the design of the reward function is crucial. The reward function for the task of identifying key nodes in a heterogeneous network is designed as:

$$r = \frac{1}{N} \cdot \sum_{i}^{N} \frac{\kappa(G\backslash v_i)}{\kappa(G)} \tag{13}$$

where $\kappa(G\backslash\{v_i\})$ represents the connectivity of the heterogeneous network when action $a = x_i$ (removing v_i from the network) is taken in state G ($s = X_G$). $1/N$ is the reward normalization factor. $\kappa(G)$ represents the connectivity of the heterogeneous network when none of its nodes are under attack. It is important to note that the reward designed in this paper is as small as possible because the cumulative reward is the ANC. According to the problem description in

Section 3, the objective of identifying key nodes is to identify a node removal strategy, that is, an order of node removal $\tilde{V} = (v_1, v_2, \ldots, v_N)$ that minimizes the ANC.

4.4 Loss Function Setting

The use of a multilayer neural network such as the Q-network allows for learning the mapping from state-action pairs (s, a) to real-valued $Q(s, a)$. $Q(s, a)$ predicts the maximum cumulative reward that can be obtained by executing action a in state s. In this paper, a two-layer perceptron with ReLU activation functions is employed to parameterize the Q-function:

$$Q(s, a) = W_5 \cdot \mathrm{ReLU}(x_i \cdot X_G \cdot W_4) \tag{14}$$

where x_i and X_G are the representation vectors of the action space and the action space, respectively, and W_4 and W_5 are the learnable parameter matrix.

A neural network with the same structure and asynchronously updated parameters is used as the evaluation network and target network of the DQN, and the loss function is designed as follows:

$$\mathcal{L}(\theta) = E\left[r + \gamma \min_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right]^2 \tag{15}$$

where:

- r is the immediate reward as defined in equation (13).
- E is the expectation and represents the average over batches of experiences sampled from the replay buffer. Each experience is a tuple (s, a, r, s').
- γ is the discount factor, a hyperparameter ($0 \le \gamma \le 1$) that determines the weight of future rewards.
- θ is the evaluation network parameters and includes all learnable weights in the GCN and DQN, that is, $\theta = (W_1, W_2, W_3, W_4, W_5)$.
- θ^- is the target network parameters, which are periodically copied from θ (e.g., every 100 steps) to stabilize training.
- $Q(s, a; \theta)$ and $Q(s', a'; \theta^-)$ represent the Q values predicted by the evaluation network and the target network, respectively, the former based on the current parameter θ and the latter based on the parameter θ^-, which is updated with delay.

5. Design of Key Nodes Identification Algorithm Based on Deep Reinforcement Learning

Based on the given framework design, the algorithm for identifying key nodes of a heterogeneous network based on the DQN is designed as follows (as shown in Figure 4):

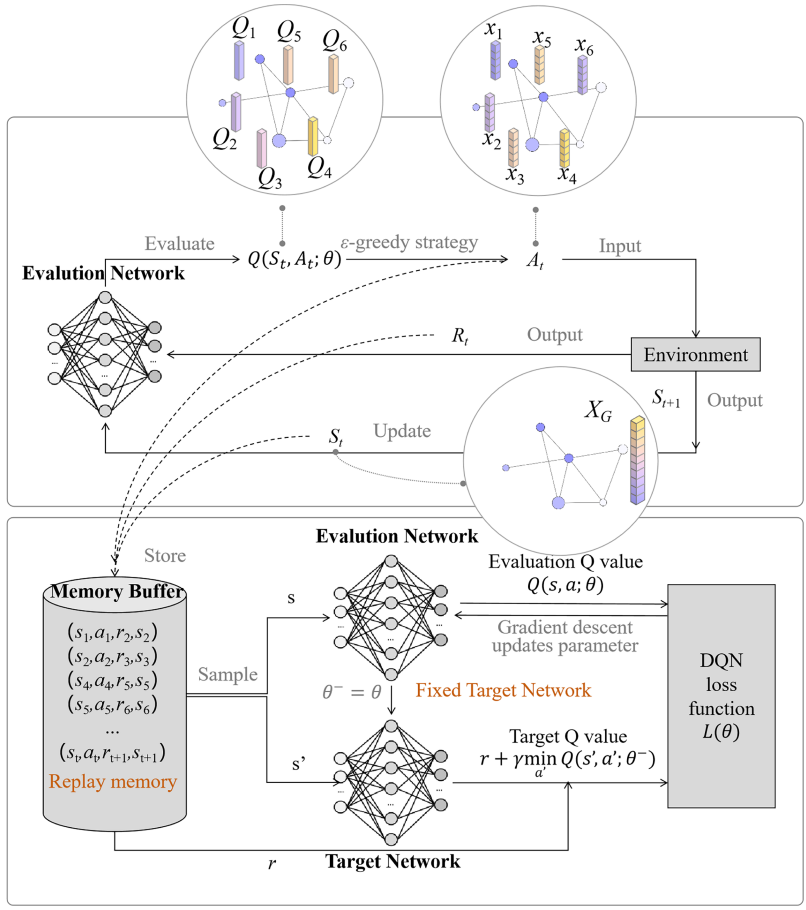

Figure 4. The algorithm for identifying key nodes in a heterogeneous network based on DQN.

Step 1. Build two neural networks with the same structure, namely the evaluation network and the target network. Initialize the parameters of the evaluation network as θ and set $\theta^- = \theta$. Initialize the parameters of the target network as θ^-.

Step 2. Input the current observation state S_t into the evaluation network. Use the neural network to predict the value of each action $Q(S_t, A_t; \theta)$ under the state S_t.

Step 3. According to the ε-greedy strategy, select action A_t. This means that with a probability of $1 - \varepsilon$, we select the action with the highest Q value, and with a probability of ε, we randomly select an action.

https://doi.org/10.25088/ComplexSystems.34.4.425

Step 4. The action A_t acts on the environment, producing a new state S_{t+1}.

Step 5. Environment output reward R_t.

Step 6. Store $[S_t, A_t, R_t, S_{t+1}]$ in the memory buffer.

Step 7. Update the current state to S_{t+1} as input to the neural network and go to step 2.

Steps 1 through 7 are the process of interaction between the agent and the environment. After a certain number of interactions, the agent trains the neural network based on the trajectory data stored in the memory buffer, as shown in the following steps.

Step 8. Use the experience replay strategy to randomly sample a certain amount of recorded data from the memory buffer.

Step 9. The sampled data is separately inputted into the evaluation network and the target network.

Step 10. The evaluation Q value, denoted as $Q(s, a; \theta)$, is calculated using the evaluation network for the sampled action a in state s.

Step 11. The target Q value, denoted as $\min_{a'} Q(s', a'; \theta^-)$, $r + \gamma \min_{a'} Q(s', a'; \theta^-)$, is calculated using the target network with the minimum Q value in state s'. Where r represents the immediate reward obtained by taking action a, $\gamma \in [0, 1]$ is the discount factor that balances the importance of future rewards.

Step 12. To train the neural network θ, we use the mean squared error (MSE) to evaluate the error between the evaluation Q value and the target Q value. The MSE is calculated as:

$$\mathcal{L}(\theta) = E\left[r + \gamma \min_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right]^2. \tag{16}$$

The MSE is also the loss function of the deep neural network. The gradient of the loss function with respect to the parameter θ is:

$$\nabla_\theta \mathcal{L}(\theta) = E\left[r + \gamma \min_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right]^2 \nabla_\theta Q(s, a; \theta). \tag{17}$$

According to this gradient, perform one step of gradient descent on the parameter θ:

$$\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \mathcal{L}(\theta) \tag{18}$$

where α is the learning rate of the deep neural network.

Step 13. When using gradient descent to update parameters, the parameters θ of the Q network are updated once per step, while the parameters θ^- of the target Q network are updated a fixed number of times per iteration.

Step 14. Training terminates when the ANC reduction rate falls below a threshold (0.1% per 1000 iterations).

6. Experimental Results and Analysis

6.1 Experimental Settings

Table 2 presents the main parameter settings for this experiment. All experiments were conducted on a system with 16GB of memory and an RTX 3070 GPU, using the Python programming language. The model was implemented using PyTorch 1.10.1 and trained with the Adam optimizer.

Parameter Name	Value	Description
Memory capacity	500	DQN memory buffer size
Batch size	32	The number of experiences used in the memory buffer each time
Training episode	16 000	Number of agent operation training episodes
Learning rate α	0.00001	Adam optimizer learning rate
Exploration probability ε	0.9	The probability of agent selecting an action based on the model
Discount factor γ	0.9	The weight of future rewards
Activation function σ	Leaky_relu	Neural network activation function

Table 2. Model-related parameter settings.

6.2 Experimental Data

First, the algorithm model is trained on the simulated network in this paper. The simulated network is a heterogeneous network with 30 nodes generated, including three types of nodes. There are 2000 training set networks. The test set includes 200 networks with a node scale of 30. Simulated heterogeneous networks for validation are based on random networks, Barabási–Albert networks and Watts–Strogatz networks, with node sizes of 30, 60 and 120, respectively. One hundred graphs are randomly generated for each node size. Such a setting is to verify the mobility of the DRLKHN framework under different network sizes, and then the network is collapsed and the average performance is calculated using the evaluation model. In testing on the real FINC [46] network, comprised of 89 nodes and 155 edges, there are three types of nodes: S, D and I, with 51, 25 and 13 nodes, respectively.

6.3 Baseline

Baseline methods include centrality-based methods, where centrality methods primarily use node structural centrality as selection criteria in sequence, incorporating local or global structural metrics such as degree, eigenvector, closeness and PageRank. An adaptive version is employed, meaning after each node removal, all centrality metrics are recalculated for all nodes, and the next node is selected based on updated values. Accordingly, the methods are referred to as high

degree adaptive (HDA), high eigenvector adaptive (HEA), high closeness adaptive (HCA) and high PageRank adaptive (HPA).

6.4 Time Complexity Analysis

To analyze the time complexity of the DRLKHN, we need to focus on the key components and steps of the algorithm that influence the overall computational complexity.

Action Space Analysis

In the action space, each action corresponds to selecting a node for attack. The node features are extracted using a GCN, which requires matrix operations involving the adjacency matrix and feature matrix. The time complexity for a single GCN layer with adjacency matrix A, feature matrix F and weight matrix W is $O(N^2 + Nd)$, where N is the number of nodes and d is the number of features per node. For k layers of a GCN, the complexity increases linearly with the number of layers: $O(k(N^2 + Nd))$, but it increases quadratically with the number of nodes N.

The overall complexity for action space calculation is $O(k(N^2 + Nd))$.

State Space Analysis

The state space is computed by summarizing the node feature vectors to form a representation for the entire graph. To compute X_G, the representation vectors of all nodes are summed up, and matrix multiplications involving the parameters W_2 and W_3 are performed. The summation step involves adding up all node feature vectors, which has complexity $O(Nd)$. The matrix multiplication has a complexity of $O(d^2)$, considering the learnable matrix W_2 and W_3.

The overall complexity for state space calculation is $O(Nd + d^2)$.

Reward Function Analysis

The reward calculation involves computing the connectivity of the graph after node removal. The complexity of this step depends on the underlying graph structure. If connectivity is checked by a graph traversal, the time complexity will be $O(N + E)$, where E is the number of edges in the graph.

The overall complexity for reward calculation is $O(N + E)$.

Loss Function and Q-Function Analysis

The Q-function is estimated using a neural network that involves matrix multiplications for the state-action pair (state S_t and action A_t) and the representation matrix X_G. We assume a two-layer neural network with ReLU activations. The time complexity for each

forward pass is $O(Nd + d^2)$, where d is the dimension of the feature space.

The overall complexity for the Q-function is $O(Nd + d^2)$ per forward pass.

Experience Replay and Gradient Descent Analysis

During training, the experience replay buffer stores the transitions. When training the model, a batch of size B is sampled from the memory buffer, and the Q-values are updated using the loss function. The complexity of experience replay and backpropagation depends on the batch size B and the size of the neural network. For a neural network with L layers, each with d units, the complexity per update is $O(BLd^2)$.

The overall complexity per training step is $O(BLd^2)$.

Overall Complexity per Iteration

For each iteration, we calculate the state, action, reward and Q-values, which involves multiple steps: feature extraction, reward computation, forward pass through the neural network and loss/backpropagation. Assuming there are N nodes, d features and B samples in a batch, the time complexity per iteration is determined by:

$$O(k(N^2 + Nd) + N + E + Nd + d^2 + BLd^2) \tag{19}$$

where:

- N is the number of nodes;
- d is the feature dimension;
- E is the number of edges in the graph;
- B is the batch size for experience replay;
- L is the number of layers in the neural network;
- k is the number of GCN layers.

In Table 3, the time complexity of DRLKHN is compared with other baseline algorithms. N is the number of network nodes.

Algorithm	Time Complexity
HDA	$O(N^2)$
HEA	$O(N^4)$
HCA	$O(N^3)$
HPA	$O(N^3)$
DRLKHN	$O(k(N^2 + Nd) + N + E + Nd + d^2 + BLd^2)$

Table 3. Time complexity analysis.

https://doi.org/10.25088/ComplexSystems.34.4.425

If the graph size is large (i.e., N and E are large), the complexity of DRLKHN can become very high, because the computations in the GCN layers and the neural network depend on the number of nodes and feature dimensions. The complexity of the training step also increases with the batch size B and the number of layers L in the neural network. Therefore, for large-scale graphs or complex network structures, higher computational resources may be required.

6.5 Experimental Results and Analysis

6.5.1 Offline Training Results and Analysis

The model parameters were periodically saved every 250 iterations during the training process. As the training progressed and the loss function began to stabilize, the parameters saved at these intervals were subsequently loaded into DRLKHN for further evaluation. The performance of the model was assessed using the test set, where the ANC value was calculated for each network, as shown in Figure 5. As training advanced, the ANC values of DRLKHN on the test set consistently decreased, gradually approaching convergence. This trend suggests that the DRLKHN model was effectively trained and continues to exhibit reliable performance on the test set, confirming its stability and robustness over time.

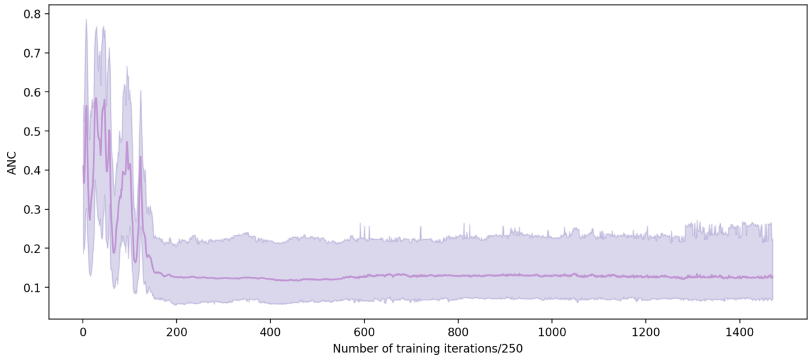

Figure 5. Convergence curve of DRLKHN on the test set.

6.5.2 Online Application Results and Analysis

Figures 6 through 8 illustrate a comparative performance analysis of DRLKHN against the HDA, HEA, HCA and HPA methods. This comparison is conducted across different network types, including simulated random heterogeneous, scale-free heterogeneous and small-world heterogeneous networks, with varying node scales. The objective of this experiment is to evaluate how well DRLKHN adapts and performs across different network structures and sizes. The ANC

metric is employed to quantify the performance of each method, providing a clear measure of their effectiveness under varying conditions.

In the random heterogeneous network (Figure 6), the ANC means for HDA, HEA, HCA, HPA and DRLKHN at a node scale of 30 are 0.095, 0.091, 0.089, 0.105 and 0.094, respectively. For a node scale of 60, the ANC means are 0.162, 0.160, 0.151, 0.173 and 0.094. At node scales of 120 and 240, the ANC means for these methods are 0.305, 0.306, 0.304, 0.308 and 0.094; and 0.407, 0.410, 0.409, 0.408 and 0.094, respectively. As the node scale increases, HDA, HEA, HCA and HPA show a significant decline in performance, with their ANC mean increasing notably from the 30-node scale. In contrast, DRLKHN maintains stable performance and outperforms the other methods in large-scale networks. For example, the ANC means for HDA at node scales of 60, 120 and 240 increase by 6.6%, 20.9% and 31.1%, respectively, compared to the 30-node scale. This is because HDA, HEA, HCA and HPA rely on centrality-based node selection, which works well in small networks but proves too simplistic for large-scale heterogeneous networks. In such networks, an optimal attack strategy must consider factors like node types and internode interactions, which centrality metrics alone cannot capture.

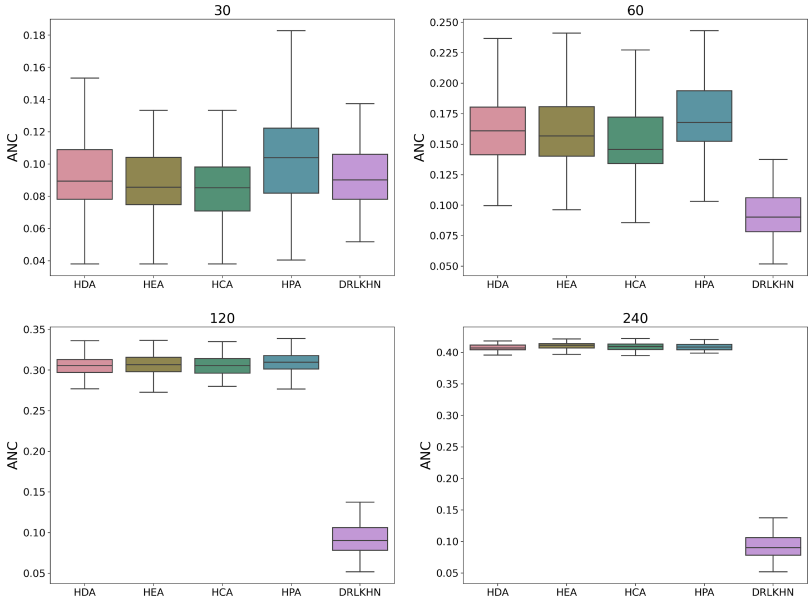

Figure 6. Comparison of ANC obtained by different attack algorithms on simulated random heterogeneous networks.

In the Barabási–Albert heterogeneous network (Figure 7), the ANC means for HDA, HEA, HCA, HPA and DRLKHN at a node scale of 30 are 0.158, 0.156, 0.155, 0.161 and 0.094, respectively. At a node

scale of 60, the means are 0.135, 0.131, 0.130, 0.141 and 0.094. For node scales of 120 and 240, the ANC means are 0.118, 0.113, 0.113, 0.125 and 0.094; and 0.108, 0.104, 0.102, 0.116 and 0.094, respectively. As the node scale increases, the performance of HDA, HEA, HCA and HPA gradually improves. For instance, the ANC means for HDA at node scales 60, 120 and 240 decrease by 2.3%, 4.1% and 5.0%, respectively, compared to the 30-node scale. However, DRLKHN consistently outperforms all other methods. This is because in Barabási–Albert heterogeneous networks, centrality metrics like degree, eigenvector, closeness and PageRank still capture node importance, although their precision decreases as the network grows. Despite this, these centrality-based methods cannot outperform DRLKHN, as they do not fully consider factors such as node types and network heterogeneity.

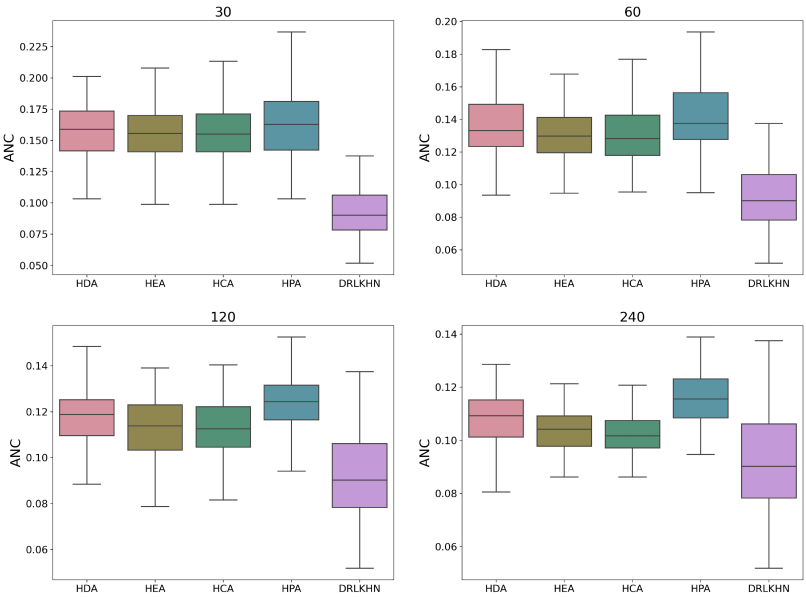

Figure 7. Comparison of ANC obtained by different attack algorithms on simulated Barabási–Albert heterogeneous networks.

In the Watts–Strogatz heterogeneous network (Figure 8), the ANC means for HDA, HEA, HCA, HPA and DRLKHN at a node scale of 30 are 0.357, 0.365, 0.327, 0.355 and 0.094, respectively. At a node scale of 60, the means are 0.345, 0.355, 0.313, 0.346 and 0.094. For node scales of 120 and 240, the ANC means are 0.337, 0.350, 0.300, 0.340 and 0.094; and 0.333, 0.345, 0.296, 0.334 and 0.094, respectively. As the node scale increases, the performance of HDA, HEA, HCA, HPA and DRLKHN remains relatively stable, showing no

significant changes. DRLKHN consistently outperforms the other methods. This stability is due to the balanced connections and high clustering in Watts–Strogatz networks, where centrality metrics like degree, eigenvector, closeness and PageRank effectively capture node importance across different scales. However, these centrality-based methods fall short of DRLKHN, as they do not account for node types and the network's inherent heterogeneity.

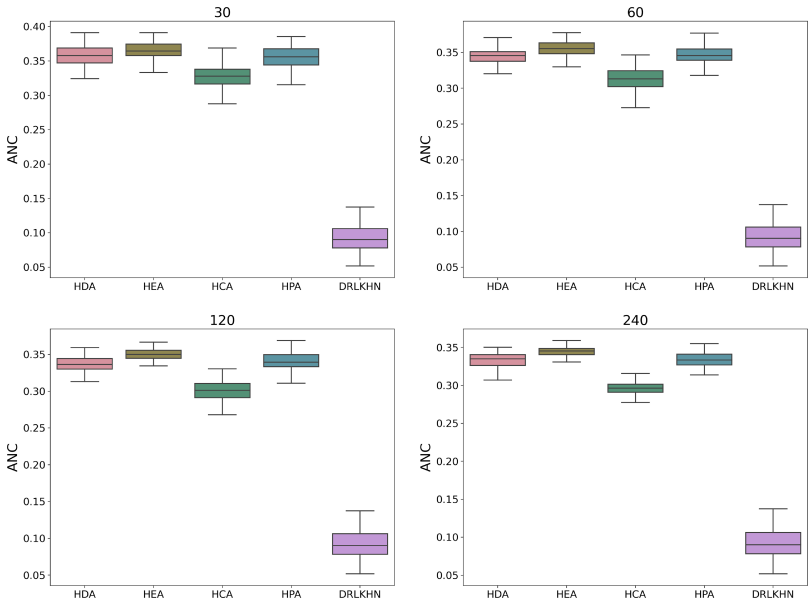

Figure 8. Comparison of ANC obtained by different attack algorithms on simulated Watts–Strogatz heterogeneous networks.

In summary, DRLKHN demonstrates consistent and superior performance compared to HDA, HEA, HCA and HPA, with little variation across different network types and node scales.

We conducted experiments to validate the performance of DRLKHN on a real dataset FINC (as shown in Figure 9). The normalized connectivity variation curves of HDA, HEA, HCA, HPA and DRLKHN on FINC are shown in Figure 10. The ANC values of HDA, HEA, HCA, HPA and DRLKHN on FINC are shown in Table 4. It can be observed in Figure 10 that the key nodes identified by DRLKHN can rapidly decrease the normalized connectivity of FINC. Table 4 shows that DRLKHN achieves robust performance on the real-world FINC network, outperforming baseline methods by 28.6%, 32.2%, 12.7% and 36.3% over HDA, HEA, HCA and HPA, respectively.

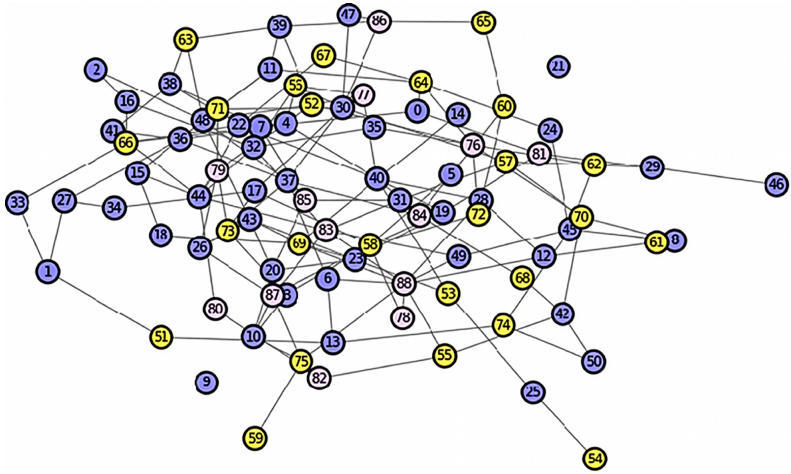

Figure 9. Schematic diagram of FINC networks.

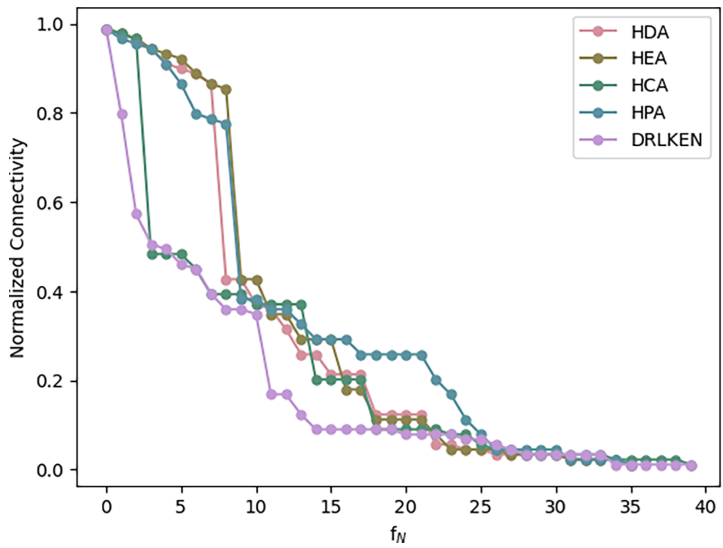

Figure 10. Comparison of normalized connectivity obtained by different attack algorithms on FINC networks.

	HDA	HEA	HCA	HPA	DRLKHN
ANC	0.1288	0.1356	0.1054	0.1444	0.092

Table 4. Comparison of ANC obtained by different attack algorithms on FINC networks.

The outstanding performance of DRLKHN on both simulated and real networks stems from its use of a combination of the GNN and DRL. This approach enables DRLKHN to better learn and understand the complex structures of heterogeneous networks. Compared to traditional attack methods based on single centrality metrics, DRLKHN leverages deep learning models to capture more network features and intricate relationships between nodes. This end-to-end learning capability allows DRLKHN to flexibly adjust strategies when confronted with networks of varying scales and structures, thereby maintaining stable performance across different scenarios.

7. Conclusion

This paper proposes a deep reinforcement learning–based heterogeneous network key nodes identification (DRLKHN) method based on the graph convolution network (GCN) and the deep Q-network (DQN). We first describe the key nodes identification problem, formulate the objective function and constraints and then design a deep reinforcement learning (DRL) framework and algorithm for key nodes identification. The model is then trained offline and applied online, validating the superiority of the proposed method through simulations and real network applications. The results show that, compared to the existing methods high degree adaptive (HDA), high eigenvector adaptive (HEA), high closeness adaptive (HCA) and high PageRank adaptive (HPA), DRLKHN delivers stable and optimal performance in simulated networks. In the real network, FINC, DRLKHN outperforms the other four methods in terms of preventing network breakdown. Specifically, DRLKHN improves performance by 28.6%, 32.2%, 12.7% and 36.3% over HDA, HEA, HCA and HPA, respectively.

DRLKHN is a fully data-driven method that requires no prior knowledge, self-learning the global impact of the nodes on the network, thereby enhancing the accuracy and reliability of identified key nodes. However, DRLKHN has a relatively high time complexity, but it combines GCN and reinforcement learning methods, enabling it to handle the complex relationships between graph structures and node features. This makes it well suited for applications that require attack and defense tasks on graphs. Its flexibility and multilayered structure give it an advantage in complex tasks, but it also requires more computational resources to support both training and inference.

DRLKHN is primarily validated on static, medium-scale heterogeneous networks with $N \leq 240$. For super-large networks of $N > 10^3$, the quadratic complexity of GCN operations and DQN training poses

significant computational challenges, which could be mitigated via techniques like graph sampling or federated learning. Furthermore, dynamic networks (e.g., time-evolving topologies), multilayer networks (e.g., interdependent interactions) and weighted heterogeneous networks (e.g., edges with varying interaction strengths) remain unexplored. Addressing these scenarios would require model extensions.

Acknowledgments

Throughout the writing of this dissertation, I have received a great deal of support and assistance. This paper is derived and expanded from our conference paper [47] under the approval of editors. In addition, we thank the support of National Natural Science Foundation of China (72374209) and Natural Science Foundation of Hunan Province (2023JJ30641).

References

[1] A. Arulselvan, C. W. Commander, L. Elefteriadou and P. M. Pardalos, "Detecting Critical Nodes in Sparse Graphs," *Computers & Operations Research*, **36**(7), 2009 pp. 2193–2200. doi:10.1016/j.cor.2008.08.016.

[2] B. Vitoriano, M. T. Ortuño, G. Tirado and J. Montero, "A Multi-criteria Optimization Model for Humanitarian Aid Distribution," *Journal of Global Optimization*, **51**(2), 2011 pp. 189–208. doi:10.1007/s10898-010-9603-z.

[3] I. D. Kuntz, "Structure-Based Strategies for Drug Design and Discovery," *Science*, **257**(5073), 1992 pp. 1078–1082. doi:10.1126/science.257.5073.1078.

[4] R. Pastor-Satorras and A. Vespignani, "Epidemic Spreading in Scale-Free Networks," *Physical Review Letters*, **86**(14), 2001 pp. 3200–3203. doi:10.1103/PhysRevLett.86.3200.

[5] D. Kempe, J. Kleinberg and É. Tardos, "Maximizing the Spread of Influence through a Social Network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, Washington, D.C., New York: Association for Computing Machinery, 2003 pp. 137–146. doi:10.1145/956750.956769.

[6] Y. Wang and H. Yan, "Adaptive Dynamic Scheduling Strategy in Knowledgeable Manufacturing Based on Improved Q-Learning," *Control and Decision*, **30**(11), 2015 pp. 1930–1936. jglobal.jst.go.jp/en/detail?JGLOBAL_ID=201602210159580319.

[7] J. Wu, D. Wang, J. Li and F. Yang, "Game Analysis of Hazardous Chemicals Transport Route Selection Based on Reinforcement Learning-Model," *Systems Engineering—Theory & Practice*, **35**(2), 2015 pp. 388–393. sysengi.cjoe.ac.cn/EN/10.12011/1000-6788(2015)2-388.

[8] K. Zhu, R. Liu and M. Wang, "Reinforcement Learning State and Value Function Selection for Portfolio Optimization," *Journal of FZU (Natural Science)*, **48**(2), 2020 pp. 146–151. jglobal.jst.go.jp/en/detail?JGLOBAL_ID=202002258439215442.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," arxiv.org/abs/1312.5602.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves et al., "Human-Level Control through Deep Reinforcement Learning," *Nature*, **518**(7540), 2015 pp. 529–533. doi:10.1038/nature14236.

[11] H. van Hasselt, A. Guez and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)* Phoenix, AZ, Palo Alto, CA: AAAI Press, 2016 pp. 2094–2100. doi:10.1609/aaai.v30i1.10295.

[12] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot and N. Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *Proceedings of the 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), New York: PMLR, 2016 pp. 1995–2003. proceedings.mlr.press/v48/wangf16.html.

[13] L. C. Freeman, "Centrality in Social Networks Conceptual Clarification," *Social Networks*, **1**(3), 1978, pp. 215–239. doi:10.1016/0378-8733(78)90021-7.

[14] D. Chen, L. Lü, M.-S. Shang, Y.-C. Zhang and T. Zhou, "Identifying Influential Nodes in Complex Networks," *Physica A: Statistical Mechanics and Its Applications*, **391**(4), 2012 pp. 1777–1787. doi:10.1016/j.physa.2011.09.017.

[15] Y. Yang, X. Wang, Y. Chen, M. Hu and C. Ruan, "A Novel Centrality of Influential Nodes Identification in Complex Networks," *IEEE Access*, **8**, 2020 pp. 58742–58751. doi:10.1109/ACCESS.2020.2983053.

[16] T. Opsahl, F. Agneessens and J. Skvoretz, "Node Centrality in Weighted Networks: Generalizing Degree and Shortest Paths," *Social Networks*, **32**(3), 2010 pp. 245–251. doi:10.1016/j.socnet.2010.03.006.

[17] G. Zhao, P. Jia and A. Zhou, "Improved Degree Centrality for Directed Weighted Network," *Journal of Computer Applications*, **40**(S1), 2020 pp. 141–145 (in Chinese).

[18] B. Ruhnau, "Eigenvector-Centrality—A Node-Centrality?," *Social Networks*, **22**(4), 2000 pp. 357–365. doi:10.1016/S0378-8733(00)00031-9.

[19] P. Bonacich, "Technique for Analyzing Overlapping Memberships," *Sociological Methodology*, **4**, 1972 pp. 176–185. doi:10.2307/270732.

[20] L. Katz, "A New Status Index Derived from Sociometric Analysis," *Psychometrika*, **18**(1), 1953 pp. 39–43. doi:10.1007/BF02289026.

[21] L. C. Freeman, "A Set of Measures of Centrality Based on Betweenness," *Sociometry*, **40**(1), 1977 pp. 35–41. doi:10.2307/3033543.

[22] L. Wang, D. Chen, J. Zhou and Y. Fu, "An Important Element Identification Based on Resilient Heterogeneous Grid," *Journal of University of Electronic Science and Technology of China*, **52**(2), 2023 pp. 280–288. doi:10.12178/1001-0548.2022077.

[23] Y. Ye and F. Yang, "Calculation of Traffic Congestion Index Based on Betweenness Centrality," *Bulletin of Surveying and Mapping*, **5**, 2021 pp. 86–90. doi:10.13474/j.cnki.11-2246.2021.0148.

[24] P. Hage and F. Harary, "Eccentricity and Centrality in Networks," *Social Networks*, **17**(1), 1995 pp. 57–63. doi:10.1016/0378-8733(94)00248-9.

[25] E. Estrada and J. A. Rodríguez-Velázquez, "Subgraph Centrality in Complex Networks," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, **71**(5), 2005 056103. doi:10.1103/PhysRevE.71.056103.

[26] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Technical Report, Stanford InfoLab, 1999. ilpubs.stanford.edu:8090/422.

[27] J. M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment," *Journal of the ACM*, **46**(5), 1999 pp. 604–632. doi:10.1145/324133.324140.

[28] L. Lü, Y.-C. Zhang, C. H. Yeung and T. Zhou, "Leaders in Social Networks, the *Delicious* Case," *PLOS One*, **6**(6), 2011 e21202. doi:10.1371/journal.pone.0021202.

[29] H. Tong, C. Faloutsos and J.-Y. Pan, "Random Walk with Restart: Fast Solutions and Applications," *Knowledge and Information Systems*, **14**(3), 2008 pp. 327–346. doi:10.1007/s10115-007-0094-2.

[30] H. Tong, C. Faloutsos and J.-Y. Pan, "Fast Random Walk with Restart and Its Applications," in *Sixth International Conference on Data Mining (ICDM'06)*, Hong Kong, China (C. W. Clifton, N. Zhong, J. Liu, B. W. Wah and X. Wu, eds.), Los Alamitos, CA: IEEE, 2006 pp. 613–622. doi:10.1109/ICDM.2006.70.

[31] C. Fan, L. Zeng, Y. Sun and Y.-Y. Liu, "Finding Key Players in Complex Networks through Deep Reinforcement Learning," *Nature Machine Intelligence*, **2**(6), 2020 pp. 317–324. doi:10.1038/s42256-020-0177-2.

[32] C. Zeng, H. Liu, L. Lu, J. Chen and Z. Zhou, "SHATTER: Searching Heterogeneous Combat Network Attack Sequences through Network Embedding and Reinforcement Learning," *IEEE Systems Journal*, **17**(3), 2023 pp. 4497–4508. doi:10.1109/JSYST.2022.3231346.

[33] C. Zeng, L. Lu, H. Liu, J. Chen and Z. Zhou, "Multiplex Network Disintegration Strategy Inference Based on Deep Network Representation Learning," *Chaos*, **32**(5), 2022 053109. doi:10.1063/5.0075575.

[34] L. Chen, L. Wang, C. Zeng, H. Liu and J. Chen, "Searching High-Value Edges Attack Sequence through Deep Reinforcement Learning," *Knowledge-Based Systems*, **272**, 2023 110562. doi:10.1016/j.knosys.2023.110562.

[35] T. N. Kipf and M. Welling, "Semi-supervised Classification with Graph Convolutional Networks," arxiv.org/abs/1609.02907.

[36] W. L. Hamilton, R. Ying and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17),* Long Beach, CA (U. von Luxburg, I. Guyon and S. Bengio, eds.), Red Hook, NY: Curran Associates Inc., 2017 pp. 1025–1035. dl.acm.org/doi/10.5555/3294771.3294869.

[37] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò and Y. Bengio, "Graph Attention Networks," arxiv.org/abs/1710.10903.

[38] K. Xu, W. Hu, J. Leskovec and S. Jegelka, "How Powerful Are Graph Neural Networks?," arxiv.org/abs/1810.00826.

[39] H. Gao and S. Ji, "Graph U-Nets," in *Proceedings of the 36th International Conference on Machine Learning (PMLR 97)*, Long Beach, CA (K. Chaudhuri and R. Salakhutdinov, eds.), PMLR, 2019 pp. 2083–2092. proceedings.mlr.press/v97/gao19a/gao19a.pdf.

[40] S. Yun, M. Jeong, R. Kim, J. Kang and H. J. Kim, "Graph Transformer Networks," *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett, eds.), Curran Associates, Inc., 2019. proceedings.neurips.cc/paper_files/paper/2019/file/9d63484abb477c97640154d40595a3bb-Paper.pdf.

[41] M. Chen, Z. Wei, Z. Huang, B. Ding and Y. Li, "Simple and Deep Graph Convolutional Networks," in *Proceedings of the 37th International Conference on Machine Learning* (H. Daumé III and A. Singh, eds.), Virtual, PMLR, 2020 pp. 1725–1735. proceedings.mlr.press/v119/chen20v/chen20v.pdf.

[42] S. Yan, Y. Xiong and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition," in *AAAI Conference on Artificial Intelligence (AAAI'18)*, New Orleans, LA, (S. A. McIlraith and K. Q. Weinberger, eds.), Palo Alto, CA: AAAI Press, 2018 pp. 7444–7452. dl.acm.org/doi/10.5555/3504035.3504947.

[43] G. Rapisardi, I. Kryven and A. Arenas, "Percolation in Networks with Local Homeostatic Plasticity," *Nature Communications*, **13**(1), 2022 122. doi:10.1038/s41467-021-27736-0.

[44] J. Chen, C. Wang, L. Xiao, J. He, L. Li and L. Deng, "Q-LDA: Uncovering Latent Patterns in Text-Based Sequential Decision Processes," *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, CA, (I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds.), Curran Associates, Inc. 2017.
papers.nips.cc/paper_files/paper/2017/file/c0e90532fb42ac6de18e25e95db73047-Paper.pdf.

[45] H. Zhang, G. Lu, M. Zhan and B. Zhang, "Semi-Supervised Classification of Graph Convolutional Networks with Laplacian Rank Constraints," *Neural Processing Letters*, **54**(4), 2022 pp. 2645–2656. doi:10.1007/s11063-020-10404-7.

[46] C. Fan, Z. Liu, X. Lu, B. Xiu and Q. Chen, "An Efficient Link Prediction Index for Complex Military Organization," *Physica A: Statistical Mechanics and Its Applications*, **469**, 2017 pp. 572–587. doi:10.1016/j.physa.2016.11.097.

[47] L. Wang, L. Chen, Z. Yang, K. Yang and Q. Yang, "DRLKHN: A Self-Learning Heterogeneous Network Key Nodes Identification Method," in *2024 World Conference on Complex Systems (WCCS)*, Mohammedia, Morocco (M. Essaaidi, M. Nemiche and S. Tayane, eds.), Piscataway, NJ: IEEE, 2024 pp. 1–8. doi:10.1109/WCCS62745.2024.10765526.