Finding Key Nodes in Complex Networks via Deep Reinforcement Learning and Multi Attention Node Connectivity

Wang Kaili, Wu Muqing, Zhao Min
Beijing Laboratory of Advanced Information Networks
Beijing Key Laboratory of Network System Architecture and Convergence
Beijing University of Posts and Telecommunications
Beijing 100876, China
{wangkl, wumuqing, zhaomin}@bupt.edu.cn

***Abstract*—Identifying and safeguarding key nodes is crucial for maintaining the reliability and stability of business functions. Currently, most key node identification algorithms are based on manual or deductive models, which are complex and suboptimal. While deep reinforcement learning algorithms have demonstrated promising practical results, they often exhibit excessive randomness in network feature extraction. To address this challenge, this paper proposes a deep reinforcement learning framework based on multi attributes attention mechanism. This framework extends node attributes to encompass general business scenarios and employs graph convolutional networks combined with an attention mechanism to learn adaptive weights for different attributes of various nodes. Subsequently, reinforcement learning is utilized to determine the sequence of key nodes in the network. This algorithm is compared with six algorithms across six types of networks, consistently achieving optimal results, thereby validating the effectiveness of the proposed approach.**

***Index Terms*—Key Nodes, Multi Attributes Attention, Graph Neural Network, Deep Reinforcement Learning**

I. INTRODUCTION

Key nodes serve as central hubs within a network, significantly influencing on network functionality. The applications of key nodes is of utmost importance. However, in complex and large-scale networks, it remains challenging to comprehensively analyze and identify key nodes by integrating structural, attribute characteristics. Current mainstream research methods predominantly fall into five categories:

1) Node centrality models: Examples encompass Degree Centrality (DC), Betweenness Centrality (BC), Closeness Centrality (CC), the PageRank algorithm, and the KG Core Decomposition algorithm [1]. The Collective Influence (CI) algorithm [2] evaluates node influence by delineating a specific range of influence. While these algorithms are relatively simple and easy to implement, the set of nodes (or edges) identified as individually important may not necessarily form the most critical set collectively.

2) Optimal decycling methods: Designed for addressing network disintegration problems. The Minsum [3] proposes a message-passing algorithm. The Belief Propagation-guided Decimation (BPD) [4] assesses the removal probability of each node within the current network. The CoreHd [5] identifies the most critical nodes as core nodes. These algorithms generally surpass centrality models and exhibit greater flexibility, though they are often not optimal in most scenarios.

3) Graph partition methods: The Generalized Network Dismantling (GND) introduces a spectral partitioning approximation algorithm [6]. The RatioCut [7] method aims to minimize the number of edges removed to partition a graph into several components of similar sizes. However, these methods often become trapped in local optima.

4) Heuristic algorithms: The Index of Influence on Epidemic (IIE) algorithm [8] assesses the influence of nodes on the spread of epidemics. The Gravity Model (GM) algorithm [9] is inspired by the gravitational model from physics. The H-index [10] represents a class of algorithms based on the local properties of nodes. These algorithms can achieve superior decompositions, but they has complex inference logic.

5) Deep Reinforcement Learning (DRL): FINDER [11] combines GNN and DQN. PIANO [12] integrates influence maximization with DRL. Xu et al. [13] present a graph reinforcement learning for SDN routing path selection. Chen et al. [14] propose a method that uses link equations based on DRL. Chen et al. [15] discuss a DRL framework utilizing an attention mechanism. Although effective, DRL lacks general applicability when integrated with specific business scenarios.

To tackle this challenge, we introduce a novel unsupervised framework named ***KRA***: *Finding **K**ey Nodes in Complex Networks via Deep **R**einforcement Learning and Multi Attributes **A**ttention*. This framework employs GNN and Multi Attributes Attention (**MAA**) to extract node features which serve as representations of the network state. DRL is then utilized to compute the Q-value for all nodes, facilitating optimal action selection and state transitions by incorporating techniques such as experience replay and N-step DQN, an offline model is pre-trained, enabling direct application in real-world network scenarios. **The core contributions of this study:**

- **Present a method** of node multi attributes attention: MAA.
- **Propose a novel architecture** MAA+DRL.
- **Achieve state-of-the-art performance**: Data will be available upon request.

978-3-903176-66-9

II. PROBLEM FORMULATION

This section outlines the problem formulation in two parts: **1) Node Attributes and Node Embedding:** Introduce the multi attributes of nodes and the process of node embedding; **2) Network Metrics:** Design a node removal strategy aimed at minimizing the Accumulated Normalized Connectivity(ANC).

A. Node Attributes and Node Embedding

This paper proposes a method to compute node importance based on multiple types of node attributes (*blue*) $x_i = \{T_1, T_2, B_1, B_2, B_3\}$, as Fig.1 shows: topological attributes (T_1, T_2) and business attributes (B_1, B_2, B_3). Through GNN (*grey*) and Multi Attributes Attention (MAA, *yellow*), various attributes are weighted differently in node embedding (*green*).

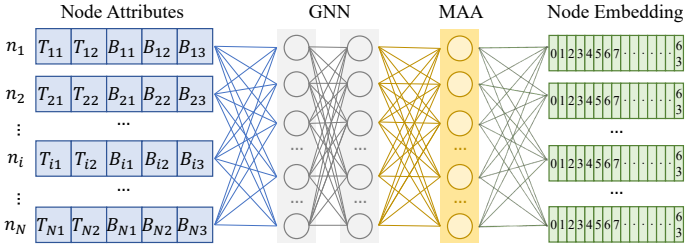

Fig. 1: **The Node Embedding Based on MAA**

1) Node Attributes: The method contains both business and topological dimensions, as Fig.1 *blue* shows.

a) Topological Attributes: The model utilizes degree and betweenness centrality as topological attributes for training.

- *Degree:* The most basic and important attributes.
- *Betweenness:* A global static feature.

b) Business Attributes: Different types of networks have various business characteristics, allowing readers to customize according to their specific requirements.

- *Management Business:* Business of managing the network.
- *Security Business:* Business of stability and resilience.
- *Basic Business:* Basic functions and services of the network.

2) GNN and MAA: Map the structural and attribute characteristics to a low-dimensional space. We employ the GraphSAGE and MAA(*yellow*) to aggregate messages on the graph $G = (V, E)$ (*grey*), combining topological and business attributes.

3) Node Embedding: As Fig.1 (*green*), 64 dimensional vector.

B. Network Metrics

In this paper, our learning objective is to design a node removal strategy that minimizes the cost of network disintegration, formulated as Eq.(1), called Accumulated Normalized Connectivity (ANC). $R(v_1, v_2, \dots v_N)$ represents the overall connectivity after removing corresponding nodes.

$$R(v_1, v_2, \dots v_N) = \frac{1}{N} \sum_{k=1}^{n} \frac{\sigma(g/v_1, v_2, \dots v_n)}{\sigma(g)}. \quad (1)$$

The Giant Connected Component (GCC) serves as the indicator of network connectivity. $\sigma(g)$ selects the GCC with Eq.(2). It will be utilized in Sec.III-C *Decoding*.

$$\sigma(g) = max\{\sigma_m : C_m \in g\}. \quad (2)$$

III. THE PROPOSED MODEL

This section outlines the architecture of KRA. As shown in Fig.2, KRA is combined with MAA+DRL:

- **Training Dataset:** Synthetic BA networks.
- **Encoding:** Node embedding (*color bars*) by Fig.2 *Encoder*.
- **Decoding:** Q-values (*green bars*) by Fig.2 *Decoder*.
- **Loss Function& Traning Process**

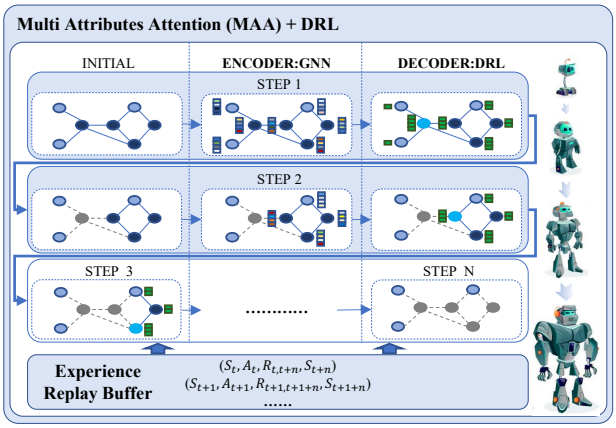

Fig. 2: **The Framework of KRA:** *Encoder* extracts node embeddings (*color bars*) and state as inputs for DRL. *Decoder* computes the Q-values (*green bars*) for each node to select the next *Action*. Calculate the *Reward* with ANC Eq.1. Then transfer to the next *State* and repeats N episodes until the network becomes isolated nodes. Select new $(S_t, A_t, R_{t,t+n}, S_{t+n})$ in *Experience Replay Buffer* for subsequent training sessions.

A. Training Dataset

BA networks are randomly generated using Networkx, comprising 30-50 nodes. 500 batches of data were updated every 5000 iterations, totaling approximately 5 million iterations.

B. Encoding

In Fig.2, *Encoder* presents Encoding. Node embeddings are as *color bars* shown, and node's various business and topological attributes are assigned different weights through Multi Attributes Attention (MAA). The details are as follows:

1) GraphSAGE Aggregate: GraphSAGE updates node representations by sampling neighboring nodes and aggregating their features with eq.(3). h_v^k is the representation of node v at layer k, $N(v)$ is the set of neighbors of v, AGG is the aggregation function, and σ is an activation function.

$$h_v^k = \sigma\left(W \bullet AGG^{k-1}\left(h_v^{k-1} \cup \{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)\right). \quad (3)$$

2) Attention Coefficients Calculation: The attribute vector of node v is x_v. The attention mechanism is used to calculate the weight of each attribute as Eq.(4), where a, W are learnable weight vectors.The attention coefficients α_{vi} is normalized using the softmax function in Eq.(4).

$$\begin{aligned} e_{vi} &= LeakyReLU(a^T[Wx_{vi}]). \\ \alpha_{vi} &= \frac{\exp(e_{vi})}{\sum_{j=1}^{n} \exp(e_{vj})}. \end{aligned} \quad (4)$$

3) Weighted Aggregation: The normalized attention coefficients are used to compute a weighted sum of the transformed features of the neighbors with Eq.(5).

$$h_v = \sum_{i=1}^{n} \alpha_{vi} x_{vi}; h_v = \frac{h_v}{\|h_v\|_2}; h'_v = [h_v^k \| h_v]. \tag{5}$$

The MAA leads KRA to learn the importance of each neighbor's attributes in neighborhood, providing a powerful tool for capturing complex patterns and relationships in graph-structured data, which can improve the quality of node embeddings. **Algorithm.1** outlines the encoding process employed in KRA. W_1,W_2,W_3 are weight parameters, $N(v)$ is the neighbor set of node v. To represent the graph state, we use a virtual node s connected to all nodes but no one connected to it.

Algorithm 1: Encoding Process in KRA

Input: $g = (V, \varepsilon)$;$X_v, v \in V$;$N(v), v \in V$;W_1, W_2, W_3
Output: Embedding Vector $z_v, v \in V$

Create a virtual node s to denote the state of graph ;
Initialize $h_v^0 \leftarrow \frac{ReLU(X_0 \cdot W_1)}{\|ReLU(X_0 \cdot W_1)\|_2}$;
for $l = 0$ *to* K **do**
  for $v \in V \cup \{s\}$ **do**
    $h_{N(v)}^{l-1} \leftarrow \sum_{j \in N(v)} h_u^{l-1}$;
    $h_v^l \leftarrow ReLU(W_2 \cdot h_{N(v)}^{l-1} \| W_3 \cdot h_v^{l-1})$;
  end
  $h_v^l \leftarrow \frac{h_v^l}{\|h_v^l\|_2}$;
end
Compute α_{vi} with Eq.4, Compute h'_v with Eq.5;
$z_v \leftarrow h'_v, \forall v \in V \cup \{s\}$;

C. Decoding

In Fig.2, *Decoder* presents Decoding. The node embeddings encoded in Sec.III-B *Encoding*, serve as inputs into the DRL to compute the Q-values of the nodes (Fig.2 *green bars*). The decoder is a basic Multi-Layer Perceptron (MLP), as Eq.(6). Z_a and Z_s are embeddings obtained from Algorithm.1, learned within the DRL. W_4,W_5 are weight parameters.

$$Q(s, a) = W_5^T ReLU(Z_a^T \bullet Z_s \bullet W_4). \tag{6}$$

KRA adopts N-step DQN, with detailed design considerations for states, actions, rewards, and loss functions as follows:
1) State: The embedding of virtual nodes.
2) Action: Selecting an action to execute, delete a node.
3) Reward: After executing each action, the value of the objective function on the graph $g = (V, \varepsilon)$ is calculated. The reward is derived from a tuple $(S_t, A_t, R_{t,t+n}, S_{t+n})$ and used in training the N-step DQN model. The N-step reward $r_{t,t+n}$ is defined in Eq.(1). The calculation method of $\sigma(g)$ refers to Eq.(2). The reward is based on minimizing the Eq.(1)-ANC.
4) Policy: Based on the Q-values of each node (Fig.2 *green bars*), we use the ϵ-greedy strategy to select the action with the maximum Q-value $argmax_a Q(s, a)$ with a probability of 1-ϵ. Otherwise, a node is selected uniformly at random. Execute the Action, and the system transitions to the next state.
5) Experience Replay: To address issues of sample correlation and smoothness, we employ Fig.2 *Experience Replay Buffer*, from which mini-batches are randomly sampled for training.
6) Double DQN: Stabilize the training process.

D. Loss Function

The target network is defined as $y_t = r_{t,t+n} + \gamma max_{a'} \hat{Q}(s_{t+n}, a'; \hat{\theta}_Q)$, where $r_{t,t+n}$ is the reward, γ is the discount factor, s_{t+n} is the state after n steps, and $\hat{Q}$ represents the target Q-values with parameters $\hat{\theta}_Q$. The prediction value of DQN is $Q(s_t, a_t; \theta_Q)$. The loss is defined as Eq.(7).

$$Loss(\theta_Q) = \alpha \sum_{i,j=1}^{N} s_{i,j} \|y_i - y_j\|^2 + E_{(S_t, A_t, R_{t,t+n}, S_{t+n})} \left[\left(r_{t,t+n} + \gamma max_{a'} \hat{Q}(s_{t+n}, a'; \hat{\theta}_Q) - Q(s_t, a_t; \theta_Q) \right)^2 \right]. \tag{7}$$

E. Training Process

The inputs to the model include embedding vectors z_v, z_a, episode N, and time T. $\theta_Q = \{W_1, W_2, W_3\}$ are parameters. The tuple $(S_t, A_t, R_{t,t+n}, S_{t+n})$ is randomly selected from the Fig.2 *Experience Replay Buffer* to train the model. The detail is presented in **Algorithm.2**.

Algorithm 2: Training process of **KRA**.

Input : embedding vectors z_v, z_a, episode N, time T
Output: Target Q network with parameters θ_Q

Initialize Buffer B with size M Q-net with θ_Q;
for $epoch = 1$ *to* N **do**
  Generate BA randomly; Initialize state $s_1 = ()$;
  for $t = 1$ *to* T **do**
    $a_t = \begin{cases} \text{argmax}_a Q(s_t, a; \theta_Q) & 1 - \epsilon \\ \text{node } v \in S_t & \epsilon \end{cases}$;
    Compute $I(v_i)$, $r_t(s_t, a_t)$, $s_{t+1} = s_t - \{a_t\}$;
    if $t \geq n$ **then**
      Store $(s_{t-n}, a_{t-n}, r_{t-n,t}, s_t)$ in B;
      Sample $(s_j, a_j, r_{j,j+n}, s_{j+n})$ from B;
      $y_j =$
      $\begin{cases} r_{j,j+n}, \text{terminal state} \\ r_{j,j+n} + \gamma \max_{a'} Q(s_{j+n}, a'; \hat{\theta}_Q), \text{ else} \end{cases}$
      Update θ_Q with Eq.(7)
    end
  end
end

This section introduces the training process of KRA. Firstly, nodes are encoded using Algorithm.1, then put node embeddings into the DRL to identify the key nodes, as Algorithm.2. The pre-trained KRA will be tested in the next section.

IV. EXPERIMENTS

In this section, we evaluate six real-world networks on the proposed algorithm and six benchmarks, to assess the performance and applicability of KRA across various domains.
Real-world networks: covers various fields and multi-scales.
1) Crime: A 911 crime network;
2) HI-II-14: A biological protein network;
3) Digg: A communication network;
4) Gnutella31: A peer-to-peer Gnutella file sharing network;
5) Facebook: A well-known social network;
6) Youtube: Another prominent social network.

The parameters of real-world networks shows in Table.I: number of nodes(N), number of edges(E), average degree(D), average shortest path length(S), clustering coefficient(C).

TABLE I: **The parameters of the real-world networks**

	Crime	HI-II-14	Digg	Gnutella31	Facebook	Youtube
N	829	4165	29652	62561	63392	1134890
E	1473	13087	84781	147878	816831	2987624
D	3.55	6.28	5.72	4.73	25.77	5.27
S	5.04	4.16	4.68	5.96	4.31	5.55
C	0.0058	0.0444	0.0054	0.0055	0.2218	0.0808

Benchmarks: Considering that Heuristic algorithms are usually cumbersome and difficult, we will focus on classic algorithms and the state-of-the-art algorithm currently.
1)HDA(DC):A classic and effective node centrality algorithm;
2)MinSum [3]: An optimal decycling method by percolation;
3)BPD [4]: An optimal decycling method based on the spin glass model of the minimum FVS problem;
4)CoreHD [5]: An optimal decycling method based on a simple and fast heuristic algorithm;
5)GND [6]: A graph partition method, the state-of-the-art method to address the ND problem;
6)FINDER [11]: The state-of-the-art algorithm based on DRL;

The performance evaluation comprises two parts: 1) Compare the effectiveness of all algorithms, assessed by the Accumulated Normalized Connectivity (ANC) after network disintegration. 2) Analyze the running times of each algorithm.

A. *Effectiveness of Real-world Networks*

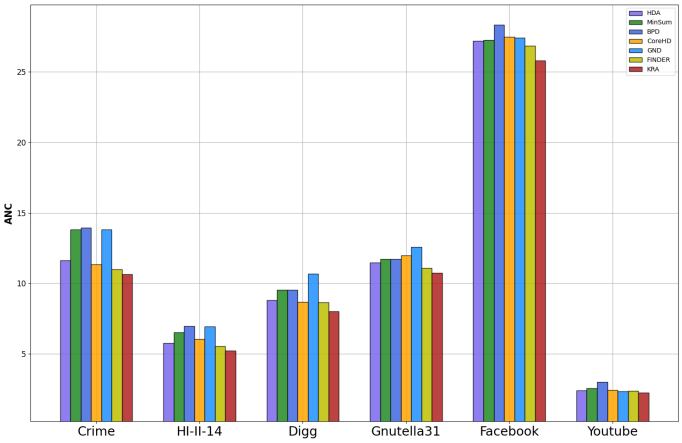

Fig. 3: **The Bars of Real-world Networks**

*1) **Bar Chart**:* As depicted in Fig.3, the horizontal axis represents the names of real-world networks, while the vertical axis shows the ANC values. The bar chart compares the performance of seven algorithms across six real-world networks, emphasizing the effectiveness of KRA. The chart clearly illustrates that KRA achieves the smallest ANC value compared to other methods.

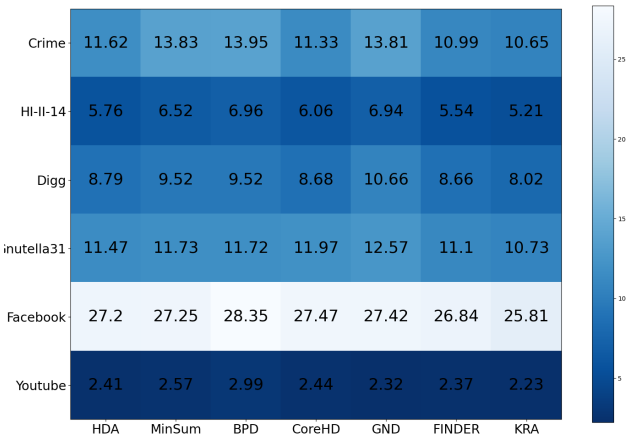

Fig. 4: **The Heat Map of Real-world Networks**

*2) **Heat Map**:* Fig.4 presents statistics on ANC values across real-world networks, using a heatmap to visualize the performance of various algorithms. The colors represent disintegration effectiveness across different networks, with varying shades indicating different levels of performance. Networks with higher average degrees and clustering coefficients tend to show lighter shades, signifying higher ANC values and greater difficulty in disintegration. Notably, the right side of the heatmap exhibits darker shades, highlighting the superior performance of KRA compared to other algorithms.

*3) **Curve Chart**:* Fig.5 illustrates the performance of seven algorithms tested across six real-world networks, highlighting the superiority of KRA over other algorithms. The horizontal axis represents the fraction of removed nodes, while the vertical axis shows the ANC values. KRA demonstrates exceptional performance, particularly in the removal of head nodes. This suggests that KRA accurately identifies the sequence of critical nodes, especially when compared to FINDER.

As shown in Fig.3,4,5, KRA exhibits significant performance improvements on real-world networks, achieving a **4.92%** enhancement over FINDER. Compared to other algorithms, KRA outperforms HDA by **7.61%**, MinSum by **14.31%**, BPD by **17.90%**, CoreHD by **8.77%**, and GND by **16.16%** across various real-world networks. This underscores the superior capability of KRA in accurately identifying key node sequences. Particularly noteworthy is the improvement of KRA over BPD by **25.42%** on the YouTube network, highlighting its enhanced effectiveness on larger-scale networks. In summary, the above experiments demonstrate that **KRA excels in four key aspects of effectiveness:**

- KRA demonstrates a notable improvement over FINDER.
- KRA surpasses other methods in identifying head nodes.
- KRA achieves extremely well on large-scale networks.

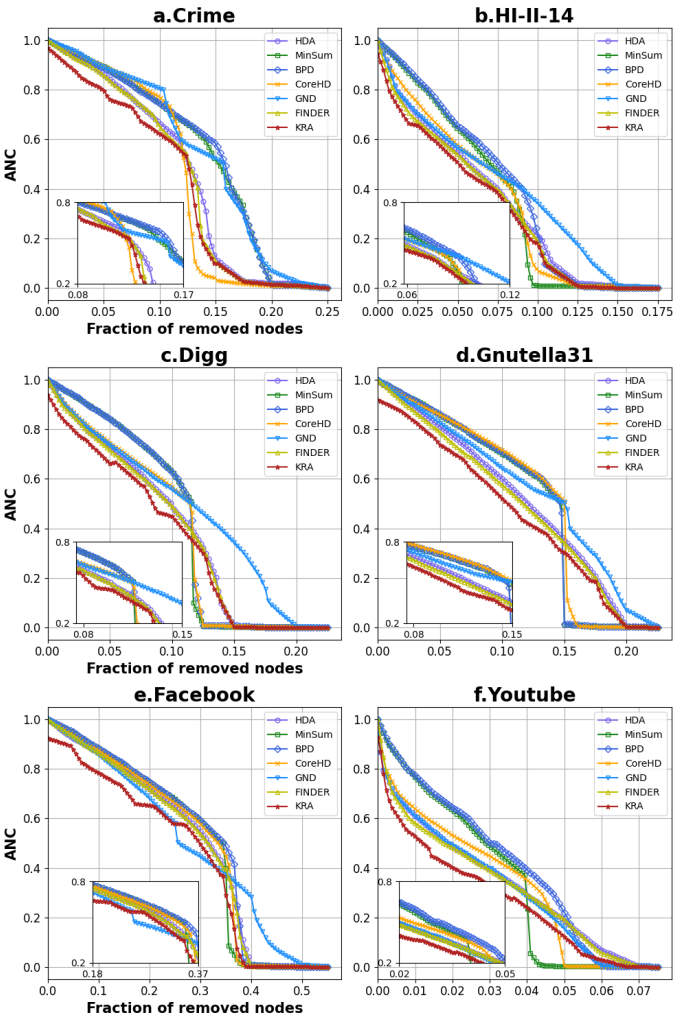

Fig. 5: **The Curves of Real-world Networks**

B. Running Times of Real-world Networks

TABLE II: **Running times of real-world networks**

Times(s)	Crime	HI-II-14	Digg	Gnutella31	Facebook	Youtube
HDA	**0.00**	**0.00**	**0.27**	**4.09**	**11.64**	1785.27
MinSum	1.82	18.18	98.36	204.91	1280.00	4980.18
BPD	0.27	8.18	35.45	37.27	762.73	420.00
CoreHD	0.09	1.82	7.73	10.82	195.27	1835.36
GND	0.09	1.18	11.55	351.18	4195.00	47496.55
FINDER	0.27	1.09	6.45	12.09	66.73	255.65
KRA	0.29	1.12	6.51	11.90	62.97	**240.44**

Table.II presents the running times of various algorithms on real-world networks used in our experiments. Entries highlighted in bold indicate optimal performance. Among the algorithms, HDA demonstrates the shortest running time. However, its effectiveness appears poor. As the network size increases, KRA consistently exhibit lower running times overall. Particularly, KRA shows superior efficiency on Youtube, indicating its effectiveness on large-scale networks.

Based on the experiments conducted in this section, it can be concluded that **KRA achieves notable improvements in effectiveness while maintaining manageable complexity, particularly advantageous in handling large-scale scenarios within neural networks and identifying head nodes**.

V. DISCUSSION AND CONCLUSION

This paper introduces a novel deep reinforcement learning algorithm called KRA, which leverages a node attributes attention mechanism to identify key nodes in complex networks. This approach significantly enhances the accuracy of feature extraction from nodes within the network. When applying KRA in practical fields, we can adjust the network weights to represent the actual business volume. Retrain the model, then we can obtain an excellent model in the corresponding field directly. Extensive validation conducted on real-world networks underscores KRA's superiority in both effectiveness and efficiency, particularly when applied to large-scale networks. In summary, **KRA introduces three core innovations**:

- A method of node multi attributes attention: MAA.
- A novel architecture: MAA+DRL.
- Achieve the state-of-the-art results.

ACKNOWLEDGMENT

This work is supported by Ministry of Education, China-111 project (NO.B17007). This work is supported by Director Funds of Beijing Key Laboratory of Network System Architecture and Convergence (NO.2024BKL-NSAC-ZJ-01).

REFERENCES

[1] Carmi, S., Havlin, S., Kirkpatrick, S., et al. "A Model of Internet Topology Using k-shell Decomposition." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 104, no. 27, pp. 11150-11154, 2007.

[2] Morone, F., Makse, H. A. "Influence Maximization in Complex Networks Through Optimal Percolation." *Nature*, vol. 524, no. 7563, pp. 65-68, 2015.

[3] A. Braunstein, L. Dall'Asta, G. Semerjian, and L. Zdeborová, "Network dismantling," *Proceedings of the National Academy of Sciences*, vol. 113, pp. 12368-12373, 2016.

[4] Zhou, H. "Spin Glass Approach to the Feedback Vertex Set Problem." *European Physical Journal B*, vol. 86, no. 11, pp. 1-9, 2013.

[5] L. Zdeborová, P. Zhang, and H. J. Zhou, "Fast and simple decycling and dismantling of networks," *Scientific Reports*, vol. 6, p. 37954, 2016.

[6] Ren, X., Gleinig, N., Helbing, D., et al. "Generalized Network Dismantling." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 14, pp. 6554-6559, 2019.

[7] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, pp. 395-416, 2007.

[8] Zhang, L., Yu, B., Yan, T., et al. "Information Entropy Based on Propagation Feature of Node for Identifying the Influential Nodes." *Complexity*, vol. 2021, pp. 1-13, 2021.

[9] Li, Z., Ren, T., Ma, X., et al. "Identifying Influential Spreaders by Gravity Model." *Scientific Reports*, vol. 9, no. 1, pp. 1-7, 2019.

[10] Lü, L., Zhou, T., Zhang, Q. M., et al. "The H-index of a Network Node and its Relation to Degree and Coreness." *Nature Communications*, vol. 7, article no. 10168, 2016.

[11] Fan, C., Zeng, L., Sun, Y., et al. "Finding key players in complex networks through deep reinforcement learning." *Nature Machine Intelligence*, 2020, 2(6):317-324.

[12] H. Li, M. Xu, S.S. Bhowmick, J.S. Rayhan, C. Sun, J. Cui, "PIANO: Influence maximization meets deep reinforcement learning," *IEEE Transactions on Computational Social Systems*, 2022. DOI: 10.1109/TCSS.2022.3164667.

[13] J. Xu, Y. Wang, B. Zhang, et al., "A Graph reinforcement learning based SDN routing path selection for optimizing long-term revenue," *Future Generation Computer Systems*, vol. 150, pp. 412-423, 2024.

[14] P. Chen, W. Fan, "Identifying critical nodes via link equations and deep reinforcement learning," *Neurocomputing*, vol. 562, p. 126871, 2023.

[15] R. Chen, W. Li, H. Yang, "A deep reinforcement learning framework based on an attention mechanism and disjunctive graph embedding for the job-shop scheduling problem," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1322-1331, 2022.