

# Dismantling Complex Networks by a Neural Model Trained from Tiny Networks

Jiazheng Zhang

Huazhong University of Science and Technology (HUST)  
Wuhan, China  
jiazhengzhang@hust.edu.cn

## ABSTRACT

Can we employ one neural model to efficiently dismantle many complex yet unique networks? This article provides an affirmative answer. Diverse real-world systems can be abstracted as complex networks each consisting of many functional nodes and edges. Percolation theory has indicated that removing only a few vital nodes can cause the collapse of whole network. However, finding the least number of such vital nodes is a rather challenging task for large networks due to its NP-hardness. Previous studies have proposed many centrality measures and heuristic algorithms to tackle this network dismantling (ND) problem. Different from theirs, this article tries to approach the ND task by designing a neural model which can be trained from tiny synthetic networks but will be applied for various real-world networks. It seems a discouraging mission at first sight, as network sizes and topologies are quite different across distinct real-world networks. Nonetheless, this article initiates insightful efforts of designing and training a *neural influence ranking model* (NIRM). Experiments on fifteen real-world networks validate its effectiveness for its mostly requiring fewer vital nodes to dismantle a network, compared with the state-of-the-art competitors. The key to its success lies in that our NIRM can efficiently encode both local structural and global topological signals for ranking nodes, in addition to our innovative labelling method in training dataset construction.

## CCS CONCEPTS

- Mathematics of computing → Graph algorithms; • Computing methodologies → Learning to rank.

## KEYWORDS

Network Dismantling, Neural Node Ranking Method, Graph Neural Networks, Complex Networks.

## ACM Reference Format:

Jiazheng Zhang and Bang Wang. 2022. Dismantling Complex Networks by a Neural Model Trained from Tiny Networks. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3511808.3557290>

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

*CIKM '22, October 17–21, 2022, Atlanta, GA, USA*

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9236-5/22/10...\$15.00

<https://doi.org/10.1145/3511808.3557290>

Bang Wang

Huazhong University of Science and Technology (HUST)  
Wuhan, China  
wangbang@hust.edu.cn

## 1 INTRODUCTION

Most real world systems, like airport transportation [15, 50], power grid [1, 18], Internet infrastructure [54] and etc., can be abstracted as complex networks each consisting of many functional nodes and edges, denoted by  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  with  $N$  nodes in  $\mathcal{V}$  and  $M$  edges in  $\mathcal{E}$ . In practical applications, it has been often reported that the failure of a few nodes and/or edges could significantly degrade the functionality and stability of a network, even leading to the system collapse. A typical example was the Italian nationwide blackout on September 28 2003 due to cascading failures of power stations starting from a single powerline [17]. Another recent example was the outage of the Internet service of the American operator Century Link [29] on August 30, 2020 arising from a single router malfunction.

Many efforts have been devoted on studying the dynamics and properties of network stability under different failure scenarios [3, 44]. Some insights have been gained from studying network failure processes, including cascading failure [12], network percolation and phase transition [13, 30, 36]. One of important insights is that a single node can disable its attached edge(s), which may further cause the failure of its neighboring nodes. Furthermore, if a few nodes are disabled at the same time, their removals could not only cause local structural damages but also lead to global topological collapse. This observation has recently motivated lots of studies on how to select a few *vital nodes* to attack, such that their removals can dismantle a network, i.e., the network dismantling problem.

The *network dismantling* (ND) problem is to determine a *target attack node set* (TAS), denoted by  $\mathcal{V}_t$  ( $\mathcal{V}_t \subseteq \mathcal{V}$ ), for its removal leading to the disintegration of a network into many disconnected components, among which the *giant connected component* (GCC), denoted as  $\mathcal{G} \setminus \mathcal{V}_t$ , is smaller than a predefined threshold. The objective of network dismantling is to find such a TAS  $\mathcal{V}^* \subseteq \mathcal{V}$  with the minimum size, that is,

$$\mathcal{V}^* \equiv \min \{ \mathcal{V}_t \subseteq \mathcal{V} : |\mathcal{G} \setminus \mathcal{V}_t| / N \leq \Theta \} \quad (1)$$

where  $\Theta$  is the predefined dismantling threshold. The ND problem has been proven as NP-hard [10], and many solutions have been proposed to find a suboptimal TAS for large networks [22, 25, 37, 57].

Recently, some researchers have considered on applying deep learning techniques for problems of combinatorial optimization on graph [14, 19, 46]. For example, many *Graph Neural Networks* (GNNs) [26, 53, 58] have been designed and applied to find feasible solutions for a large class of graph optimization problems, like the maximum cut, minimum vertex cover, maximum independent set problem. Also some GNNs have been designed to approximate some global centralities of a node [16, 21, 33], which normally with

high computation complexities due to graph-wide operations. For example, the betweenness centrality requires to first find the shortest path for any pair of nodes. With the powerful representation capability of a GNN, such graph centralities or metrics can be approximated with high accuracy and low computational complexity.

For network dismantling, it is also very intriguing to ask the question about whether we can design and train a neural model to output the target attack node set for any input network? However, this question has not yet been well studied in the literature to our best knowledge. This may be due to the fact that not only network sizes but also topological characteristics are quite different across different real-world networks, which seems to discourage applying one neural model to dismantle different networks. Nonetheless, this article presents an affirmative answer to the question through our initiative efforts of designing an effective neural model for network dismantling.

In this article, we are interested in the following question: *Can we find a smallest TAS to dismantle any real-world network via a neural model?* That is, given an input graph  $\mathcal{G}$  with its adjacency matrix  $\mathbf{A}$ , a neural model can output a *dismantling score*  $s_i^{dis} \in \mathbb{R}$  for each node  $v_i$  to facilitate the selection of target attack nodes. In this article, we design and train a neural model, called *neural influence ranking model* (NIRM), for the network dismantling problem. Some challenges on designing and training a neural model have been addressed in this article:

- Design a neural model: The influence of a node to the stability of a whole network should be not only evaluated from its own structural characteristics, but also compared with other farther apart nodes.
- Train a neural model: The *training dataset* as well as the so-called *ground truth* labels for training a neural model are not available, not even mention to their appropriateness and trustfulness.

In NIRM, we learn both local structural and global topological characteristics for each node, so as to compute and fuse its local and global influence scores for outputting each node a dismantling score. Our NIRM model is trained by some tiny synthetic networks of no more than thirty nodes, where we can find the optimal TAS(s) via exhaustive search. We design a labelling rule for selected target nodes and propose a training score propagation algorithm to obtain labels for other nodes. We conduct extensive experiments on various real-world network, and results validate the effectiveness of our neural model: Compared with the classic approaches and state-of-the-art competitors, the proposed NIRM performs the best for most of real-world networks.

The rest of the paper is organized as follows: Section 2 reviews the related work. The design and training details of our NIRM are introduced in Section 3 and Section 4, respectively. Section 5 presents the experimental results. Finally, Section 6 concludes the paper.

## 2 RELATED WORK

### 2.1 Deep Learning on Node Ranking

Recently, the problem of ranking nodes on a graph has been revisited from deep learning viewpoint and some neural models have been designed.

Centrality-oriented approaches focus on fast approximating the relative rank of nodes in terms of their centralities to reduce computation complexity [21, 24, 33–35, 40]. Grando et al. [24] propose to estimate eight graph centralities by a neural network with degree and eigenvector as input features. Fan et al. [21] propose a GNN-based encoder-decoder framework to select the top-K highest betweenness centralities. Sunil et al. [34] propose a neural model to predict both betweenness and closeness centrality.

Some other neural ranking approaches do not estimate nodes' centralities, but directly output ranking scores for some downstream tasks [38, 48, 51, 56]. For example, Song et al. [48] introduce a variant of Recurrent Neural Network for node ranking in heterogeneous temporal graphs to dynamically estimate temporal and structural nodes' influences over time. Yu et al. [56] propose a CNN-based model to identify critical nodes in temporal networks. Park et al. [38] presented an attentive GNN for predicate-aware score aggregation to estimate entity importance in Knowledge Graphs.

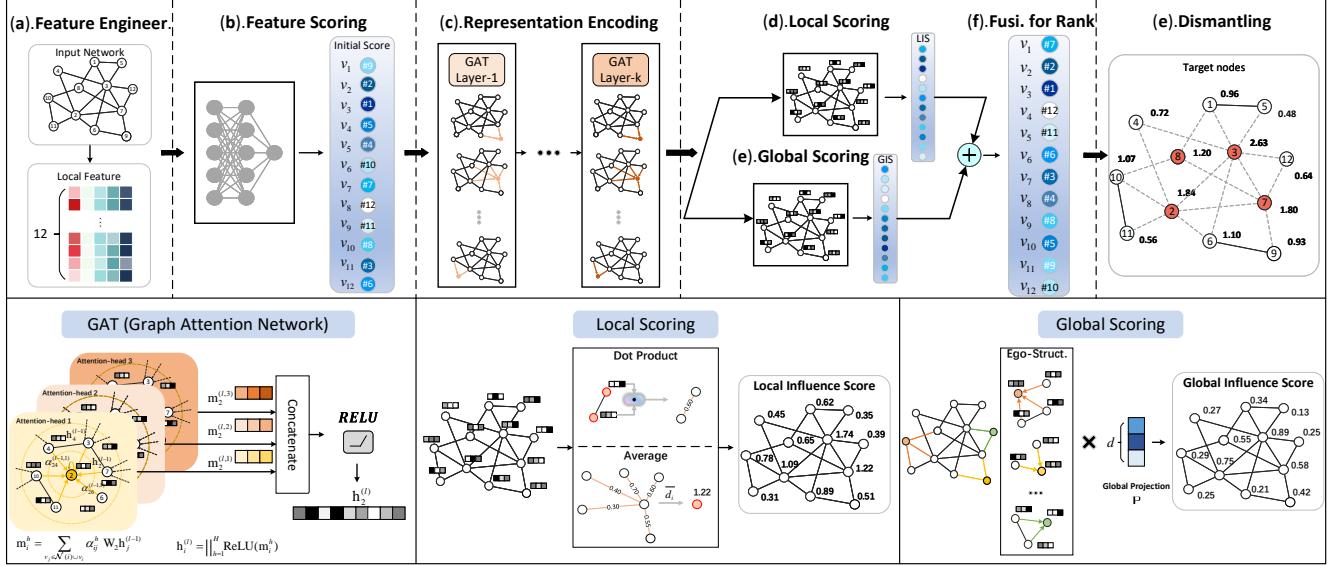
### 2.2 Network dismantling

Network dismantling is a typical discrete combinatorial optimization problem. Attacking different TAS will lead to inestimable combinatorial effects on network connectivity. Most existing solutions to the ND problem can be generally divided into two categories: centrality metric-based and network decycling-based. For the former, the basic idea is to first compute some centrality metric for each node, and then rank nodes and select the top- $K$  important nodes to form TAS. Some commonly used centralities include degree centrality (DC) [2], closeness centrality (CC) [7], betweenness centrality (BC) [23], eigenvector centrality (EC) [9] and etc. Recently, Collective Influence (CI) [36] was proposed as an improved version of DC, which quantifies the importance of a node by considering not only its one-hop neighbors but also its high-order neighbors.

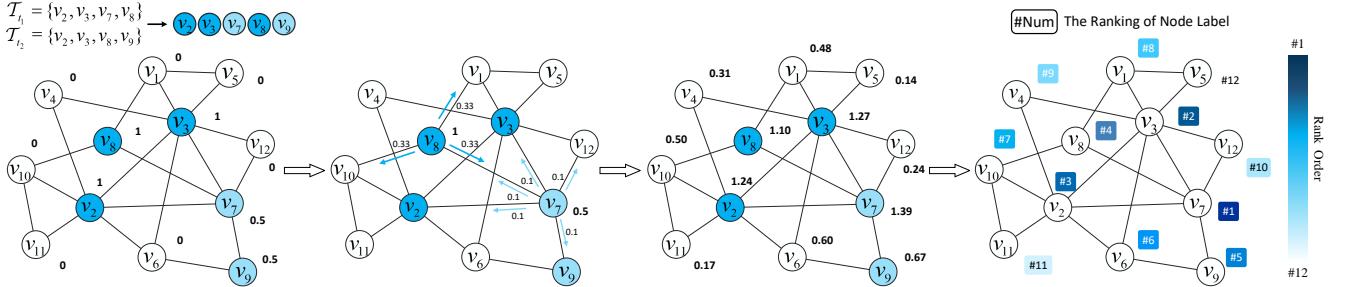
The network decycling-based approaches, including the Minsum [10], BPD [37], CoreHD [57] and etc., tried to first finding those nodes whose removals can cause an acyclic network, that is, a network does not contain loops. Next, a kind of greedy tree-breaking algorithms are used to break the residual forest into small disconnected components. In addition, after determining the TAS, some nodes can be reinserted back into the original network, if such reinsertion would not cause the increase of the residual GCC. For example, the BPD [37] applies the spin glass theory to solve the feedback vertex set problem when searching nodes for breaking all network loops. Besides, the GND [41] proposes a spectral approach unifying the equal graph partitioning and vertex cover problem, iteratively partitioning the network into two components.

## 3 NIRM: A NEURAL INFLUENCE RANKING MODEL FOR NETWORK DISMANTLING

The proposed neural influence ranking model (NIRM) takes the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$  of a network as input, and outputs a vector of *dismantling scores*  $\mathbf{s} \in \mathbb{R}^{N \times 1}$  for all nodes. Although different input networks are with different sizes (viz., different  $N$ ), we design several neural modules not only for converting network dimensions but also for learning nodes' representations. Note that all learnable parameters in neural modules are trained from tiny synthetic networks. We compute local and global influence scores



**Figure 1: The NIRM framework:** The upper left part is feature scoring that generates initial scores by converting local features. The lower left part illustrates GAT, which encodes initial scores between neighbors as well as neighborhood structure for learning each node representation. Local scoring and global scoring respectively evaluates a node's influence to the stability of node-centric local structure and network-wide global topology. The final dismantling score is a fusion of local score and global score. The upper right part marks the selected attack nodes in red, which are the same as one of optimal solutions by exhaustive search.



**Figure 2: Illustration of training score initialization, propagation and labeling.** For a tiny synthetic network, exhaustively search the optimal dismantling sets ( $\{v_2, v_3, v_7, v_8\}$  and  $\{v_2, v_3, v_8, v_9\}$  in the given example), and initialize the training score for a node as its normalized appearances in the optimal dismantling sets. For each vital node (colored blue), its initial score is equally propagated to its one-hop neighbors. After score propagation, a node training score is the sum of its own and its received scores (if any), of which the normalized rank is ground truth influence label. The number next to each node in the third figure is its training score. The forth figure presents the ranking results according to the training scores.

based on learned nodes' representations and fuse them to output dismantling score.

Fig. 1 presents the NIRM architecture, which consists of the following modules: (1) feature engineering, (2) feature scoring, (3) representation encoding, (4) local scoring, (5) global scoring, and (6) fusion and rank.

### 3.1 Feature engineering

This module is to construct a feature vector for each node based on the adjacency matrix of input network. Although many attributes and measures, either local node-centric or global network-wise, can be used as features, we prefer to focus on those local ones, as they do not incur too much computation burden. Specifically, our NIRM uses the following five local node-centric attributes to construct a *local feature matrix*  $X \in \mathbb{R}^{N \times 5}$ , where each row is the feature vector  $x_i$  for node  $v_i$ , consisting of (1) the number of its one-hop

neighbors, viz., its degree; (2) the number of its two-hop neighbors; (3) average degree of its one-hop neighbors; (4) its local clustering coefficient; (5) a constant for regulation. These features contains the basic neighborhood information of a node, revealing its local connectivity and structure characteristics to some extent.

### 3.2 Feature scoring

This module is to make an initial evaluation of the importance to network stability for each node based on its local feature. On the one hand, we want to measure a node's connectivity and structure characteristics; On the other hand, we also would like to make a network-wide importance estimation from local feature across all nodes. To this end, we use a fully connected neural network with a network-wide shared kernel to convert local features into scores:

$$s_i^{init} = \text{ReLU}(\mathbf{W}_1 \mathbf{x}_i^\top + \mathbf{b}_1), \quad (2)$$

where  $s_i^{init}$  is the initial score of node  $v_i$ ,  $\mathbf{W}_1$  and  $\mathbf{b}_1$  are the shared kernel with learnable parameters. We note that other neural networks can also be used for initial scoring.

### 3.3 Representation encoding

This module is to encode the initial score of each node together with its neighbors' scores, yet in a discriminate way, into a representation vector in a latent space. After feature engineering and scoring, the initial score could reflect a node's influence to network stability to some extent. However, the initial score obtained from network-wide conversion only focuses on statistical metrics in the node-centric ego-net, which may not well exploit neighbors' properties as well as multi-relational interactions between neighbors. We would like to further mine potential neighborhood interactions for evaluating the influence of a node specific to its structure. To this end, we employ the graph neural network technique for encoding nodes' representations.

We introduce Graph Attention Network (GAT) [49] consisting of  $L$  neighborhood aggregation (NA) layers to learn nodes' representations from their initial scores and high-order structure information. The input is the initial score vector  $\mathbf{s}^{init}$ , and output is the representation matrix  $\mathbf{H}$ . Each NA layer applies  $H$  attention heads to encode node-centric structural properties from different views. We take a node  $v_i$  for example to introduce the core operations in  $l$ -th NA layer:

- Compute the coefficients  $\alpha_{ij}^h$  between  $v_i$  and its one-hop neighbors  $v_j \in \mathcal{N}_i$  (including itself) by the  $h$ -th attention head  $\mathbf{a}_h$  ( $\mathbf{a}_h^\top$  as its transposition):

$$\alpha_{ij}^h = \frac{\exp[\text{LeakyReLU}(\mathbf{a}_h^\top \mathbf{h}_{ij})]}{\sum_{v_k \in \mathcal{N}_i \cup v_i} \exp[\text{LeakyReLU}(\mathbf{a}_h^\top \mathbf{h}_{ik})]}, \quad (3)$$

where  $\mathbf{h}_{ij} = \mathbf{h}_i^{(l-1)} || \mathbf{h}_j^{(l-1)}$  stands for the concatenation of the hidden embeddings from the  $(l-1)$ -th NA layer.

- Aggregate the converted neighborhood hidden embeddings in a weighted way to obtain an aggregation embedding  $\mathbf{m}_i^h$  for the  $h$ -th attention head:

$$\mathbf{m}_i^h = \sum_{v_j \in \mathcal{N}(i) \cup v_i} \alpha_{ij}^h \mathbf{W}_2 \mathbf{h}_j^{(l-1)}. \quad (4)$$

- Concatenate the converted aggregation embeddings for  $H$  attention heads to output the hidden embedding  $\mathbf{h}_i^l$  for the  $l$ -th NA layer.

$$\mathbf{h}_i^{(l)} = \|\mathbf{h}_i^l\|_H^H \text{ReLU}(\mathbf{m}_i^h). \quad (5)$$

Neighboring nodes influence each other through their own initial scores as well as local structural properties. For one NA layer, low-order structural information are attentively encoded for node representation learning. Through multiple NA layers,  $L$ -hop apart neighbors are also included in representation learning, which can help to capture high-order neighbors' influences as well as some high-order topological information.

### 3.4 Local scoring

This module is to evaluate the local influence of each node, for its removal, to the destruction of the local network structure centered to the node. From a node-centric view, one could expect that if a node is with more similar neighbors, then its removal may not only influence the connectivity of its local structure, but also cause further instability cascade through its similar neighbors. For such considerations, we propose to compute a local influence score  $s_i^{local}$  for each node  $v_i$  as follows:

$$s_i^{local} = \frac{1}{|\mathcal{N}_i|} \sum_{v_j \in \mathcal{N}_i} \langle \mathbf{h}_i, \mathbf{h}_j \rangle + \bar{d}_i, \quad (6)$$

where  $\langle \cdot, \cdot \rangle$  stands for the dot product of two vectors, and  $\bar{d}_i = d_i/d_{max}$  is the normalized node degree and  $d_{max}$  the largest degree of input network.

### 3.5 Global scoring

This module is to compare the global influence of one node, for its removal, to the possible damage of the global topology with that of other nodes. Consider that two nodes are with similar local structure but distant from each other. As the dismantling is for the whole network, it is necessary to further distinguish the importance of the two nodes for their global ranking. For such considerations, we design a global projection operator  $\mathbf{P}$  to learn a global influence score for each node:

$$s_i^{global} = \langle \mathbf{P}^\top, \hat{\mathbf{h}}_i \rangle, \text{ where } \hat{\mathbf{h}}_i = \sum_{v_j \in \mathcal{N}_i \cup v_i} \frac{1}{\sqrt{|\mathcal{N}_i| + 1}} \cdot \mathbf{h}_j, \quad (7)$$

where  $\mathbf{P}$  denotes the learnable projection vortor and  $\hat{\mathbf{h}}_i$  is the neighborhood representation of node  $v_i$ . We note that using  $\hat{\mathbf{h}}_i$  instead of  $\mathbf{h}_i$  is again to enjoy node-centric local structural information such that global comparison is conducted for a kind of *ego-structure* though represented by a single node.

### 3.6 Fusion and rank

This module is to fuse the local score  $s_i^{local}$  and global score  $s_i^{global}$  as the *dismantling score*  $s_i^{dis}$  for each nodes. As we have already designed sophisticated mechanisms for computing  $s_i^{local}$  and  $s_i^{global}$ , we simply add up the two scores for  $s_i^{dis}$ , that is,

$$s_i^{dis} = s_i^{global} + s_i^{local}. \quad (8)$$

Finally, nodes are ranked according to their dismantling scores.

**Table 1: Statistical properties of real-world networks**

Network	Category	Nodes	Edges	Density	AVD	Diameter
PPI [11]	Protein	2,224	6,609	0.0027	5.94	11
HI-II-14 [43]	Protein	4,165	13,087	0.0016	6.28	11
Ca-GrQc [32]	Collab.	4,158	13,422	0.0016	6.46	17
NetScience [31]	Collab.	1,461	2,742	0.0026	3.75	17
DNCEmails [31]	Comm.	1,866	4,384	0.0025	4.70	8
Innovation [31]	Comm.	241	923	0.0319	7.66	5
Infectious [31]	Contact	410	2,765	0.0330	13.49	9
Genefusion [27]	Gene	291	279	0.0066	1.92	9
P-H [31]	Social	2,000	16,098	0.0081	16.10	10
HM [31]	Social	1,858	12,534	0.0073	13.49	14
UsPower [52]	Grid	4,941	6,594	0.0005	2.67	46
Crime [31]	Malicious	829	1,473	0.0043	3.55	10
Corruption [42]	Malicious	309	3,281	0.0689	21.24	7
Roget [6]	Lexicon	1,010	3,646	0.0072	7.22	10
Bible [31]	Lexicon	1,773	9,131	0.0058	10.30	8

### 3.7 Complexity analysis

The NIRM model requires node features matrix  $\mathbf{X}$  and the sparse adjacency matrix  $\mathbf{A}$  as input, which can be applied to any given network to infer dismantling scores. The time complexity of NIRM consists of three parts, the first part explicitly learns the initial score from local feature, which takes  $O(|V|)$  and  $V$  is the number of nodes. The second part GAT applies the self-attention mechanism to encode high-order structure and neighbors information into topology representation, the time complexity of the embedding process is  $O(L(|V| + |E|))$ , where  $L$  is the number of propagation layers(e.g., 3),  $E$  is the number of edges. After GAT, then combine the local influence scores and global influence scores to sort nodes in the entire graph, where the time complexity of calculating the local and global influence scores is  $O(|E|)$ ,  $O(|V|)$  respectively and the node ranking operation takes  $O(|V| \log |V|)$ . Therefore, the overall complexity of the NIRM model is given by  $O(|V| + |E| + |V| \log |V|)$ .

## 4 MODEL TRAINING

### 4.1 Training datasets

We first introduce how to construct training datasets from synthetic model networks. In network science, some generative models have been widely accepted for producing synthetic networks, including Erdős-Rényi (ER,  $p = 0.1$ ) [20], Watt-Strogatz (WS,  $k = 4, p = 0.1$ ) [52], Barabási-Albert (BA,  $m = 3$ ) [5] and Powerlaw-Cluster (PLC,  $m = 3, p = 0.05$ ) [28]. Such synthetic model networks can well reflect one or more kinds of topological characteristics of many real-world networks, such as uniform or power-law node degree distribution. We use these generative models to produce a large number of tiny synthetic networks to compose a training dataset. Each synthetic network contains only a few of nodes (randomly selected from 20 to 30).

We use *exhaustive search* to first find the optimal TAS(s) for each tiny synthetic network. Note that the computational cost

**Table 2: NIRM model configurations and hyperparameters.**

Hyper-parameter	Value
Adam optimizer learning rate	$1 \times 10^{-3}$
regularization term (L2 penalty)	$1 \times 10^{-4}$
learning decay	0.4
mini-batch size	6
number of self-attention heads	8, 4, 2
maximum training epoches	50
neighbor-aggregation layers	3
dimensions of node embedding	32, 16, 8
patience period	8
probability of dropout	0.1, 0.2
negative slope of Leaky ReLU	0.2
number of neurons per layer	5, 8

of exhaustive search increases exponentially with the increase of network size. This is also why we only use small-scale synthetic networks for training dataset. Further notice that for a synthetic training network  $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ , there could exist more than one optimal TAS. For  $\mathcal{G}_t$ , let  $\mathcal{T}_t$  denote the set of its optimal TASs. For each node  $v_i \in \mathcal{T}_t$ , we count its frequency of appearing in different TASs, denote by  $N_i$ . Furthermore, let  $N_{max}$  denote the maximum of  $N_i$ . Then for each  $v_i \in \mathcal{T}_t$ , we set its initial training score  $c_i^0 = N_i / N_{max}$ .

Although the aforementioned approach can set a so-called ground-truth score for each node  $v_i \in \mathcal{T}_t$ , there could exist some node  $v_j \in \mathcal{V}_t \setminus \mathcal{T}_t$  with no such a score, i.e.,  $c_j^0 = 0$ . On the one hand, such a node  $v_j$  may still play a role to network stability, though only with a smaller influence. On the other hand, the objective of our neural model is to output dismantling scores for final ranking. To take care of all nodes and avoid the unbalance of score, we propose the following *training score propagation* algorithm to propagate the initial training score of a node  $v_i \in \mathcal{T}_t$  to its one-hop neighbors:

$$c_i = \sum_{v_j \in \mathcal{N}_i} \frac{c_j^0}{|\mathcal{N}_j|} + c_i^0, \quad \forall c_i \in \mathcal{V}_t, \quad (9)$$

where  $\mathcal{N}_j$  is the set of  $v_j$ 's neighbors and  $c_j^0$  denotes initial training score of  $v_j$ , which is equally allocated to  $v_j$ 's neighbors (divided by the number of its neighbors). After score propagation, training score  $c_i$  of node  $v_i$  is the sum of its own and its received scores.

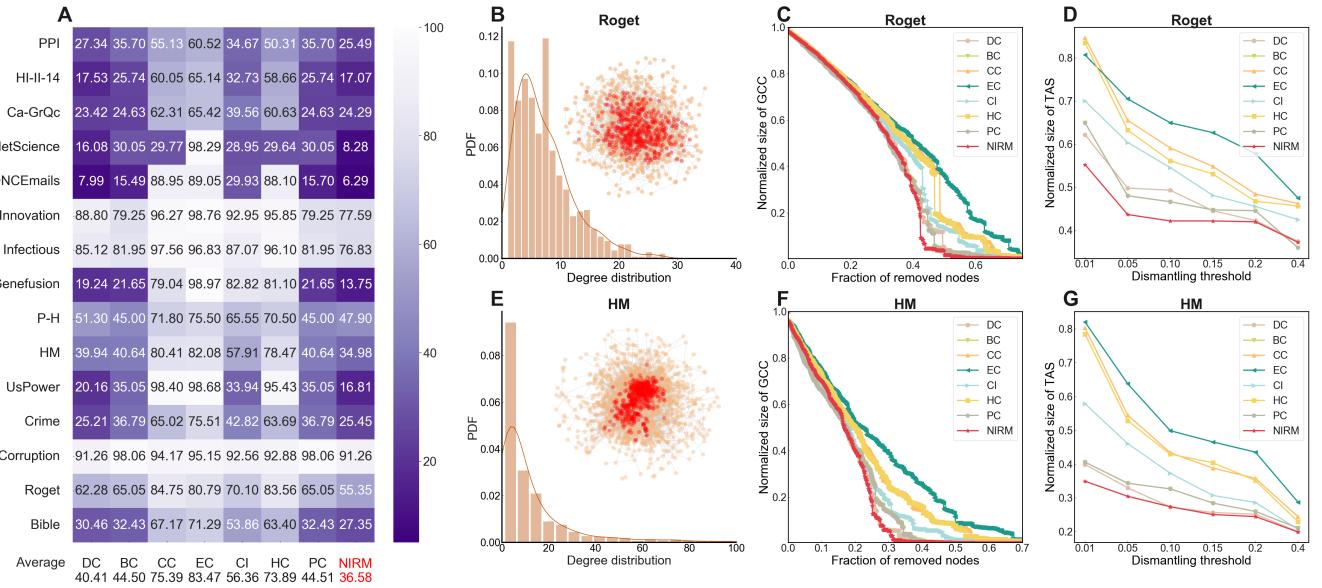
Fig. 2 illustrates the training score initialization, propagation and labeling by an example network.

### 4.2 Loss function

Let  $\mathbf{c}$  denote the nodes' scores of an instance in the training dataset, and  $\hat{\mathbf{s}}$  is the estimated scores of NIRM. The loss of one training network is defined as the following *mean square error*:

$$\text{Loss} = \frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} (c_i - \hat{s}_i)^2. \quad (10)$$

We implemented the NIRM in the PyTorch framework and used the Adam optimizer to train the model. During the training, we decay



**Figure 3: Comparison of one-pass dismantling performance on real-world networks.** (A): The smaller the normalized size of TAS  $\rho$ , the darker the color. Among 15 real world networks, the NIRM achieves the best in 12 networks, and second best on 3 networks. The bottom row provides the averaged  $\rho$  over all 15 networks, where ours is 36.58 and second best is 40.41. (B and E): the node degree distribution and the attack nodes (red) selected by NIRM for two real world networks: Roget containing 1,010 nodes and 3,646 edges, and HM containing 1,858 nodes and 12,534 edges. (C and F) : the NGCC when removing different fractions of target attack nodes, where the area of ours is 298.79 and 296.06, the second smallest is BC of 303.28 in Roget,DC of 302.55 in HM. (D and G): the dismantling performance  $\rho$  against different dismantling thresholds  $\Theta$ .

the learning rate with an early stopping criterion based on the loss on the validation set.

In our NIRM, the learnable parameters include  $\mathbf{W}_1, \mathbf{b}_1$  in feature scoring,  $\mathbf{a}_{l,h}$  ( $l = 1, \dots, L, h = 1, \dots, H$ ),  $\mathbf{W}_2$  in representation learning, and  $\mathbf{P}$  in global scoring. There are in total 854 learnable parameters in our model. We generate 4,000 synthetic model networks with 95% as training instances and 5% as validation instances.

## 5 EXPERIMENT RESULTS AND ANALYSIS

### 5.1 Experiment settings

We evaluate NIRM on fifteen real-world networks from various domains, including infrastructure network, communication network, collaboration network, malicious network and etc. Table 1 presents the characteristics and statistics of these real-world networks, model configurations and training hyper-parameters can be found in Table 2. All the experiments were conducted on an 8-core workstation with the following configurations: Intel Xeon E5-2620 v4 CPU with 2.10GHz, 32GB of RAM and Nvidia GeForce GTX 1080Ti GPU with 11GB memory. We release our code on Github at: <https://github.com/JiazhengZhang/NIRM>.

### 5.2 Dismantling strategy

For an input network, our NIRM outputs all nodes' dismantling scores for ranking, so enabling two dismantling strategies. The first one, called *one-pass dismantling*, just selects the top- $K$  nodes according to their dismantling scores, such that the removal of only

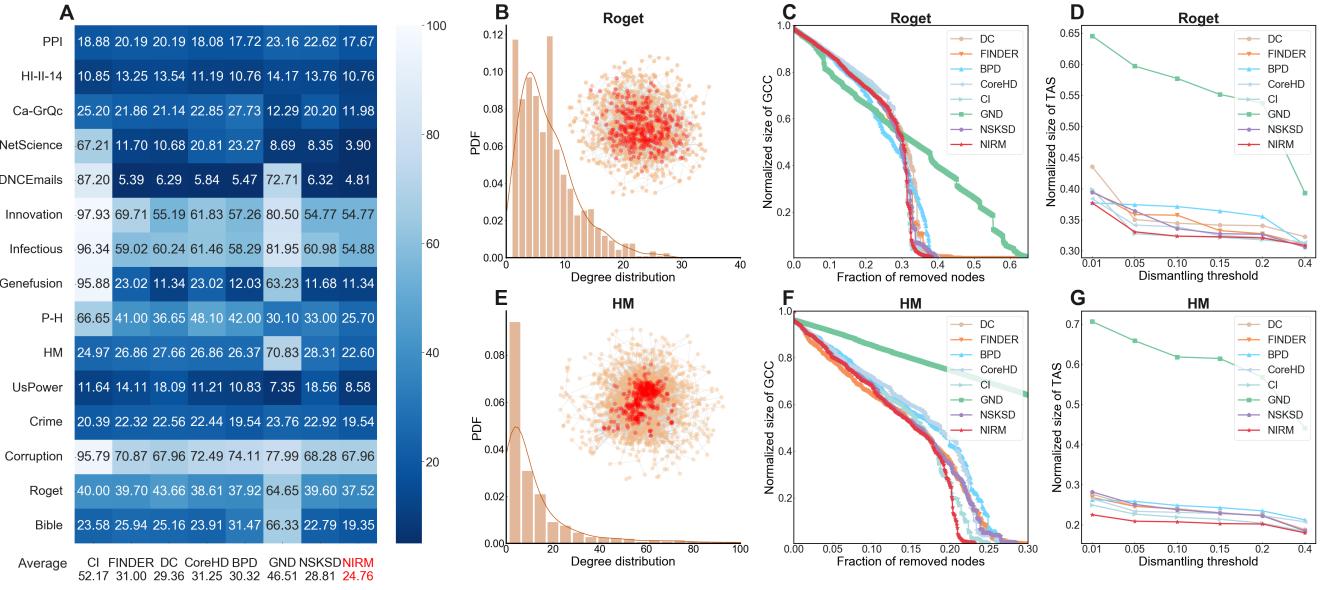
such  $K$  nodes can lead to the *normalized size of GCC* (NGCC) in the residual network not exceeding the threshold  $\Theta$ . NGCC is defined as the number of nodes in GCC divided by that of the original network. The second one, called *adaptive dismantling*, only selects the top-1 node to remove; While the residual network after removing such one selected node will be input again to output a new node for next removal; Such selection-and-removal process will be repeated until the NGCC satisfies the threshold.

### 5.3 Baseline methods

We compare NIRM with other state-of-the-art approaches which focus on estimating node importance, approaches for network dismantling can be generally classified into two categories according to the corresponding dismantling strategies.

**One-pass approaches**, common metric-based approaches mostly belong to this category, various baselines are selected and their introduction are listed below:

- DC (Degree Centrality) [2]. Sequentially remove nodes in order of degree centrality.
- CI (Collective Influence) [36]. CI is defined as the product of the node degree (minus one) and total degrees of neighboring nodes at the surface of a ball of constant radius.
- EC (Eigenvector Centrality) [9]. EC is based on the idea that the more important a neighboring node is connected, the more important that node is.



**Figure 4: Comparison of adaptive dismantling performance on the fifteen real-world networks.** (A) The smaller the normalized size of TAS  $\rho$ , the darker the color. Among the 15 networks, the NIRM achieves the best on 14 network, and the second best on 1 networks. The bottom row provides the average  $\rho$ , where ours is 24.76 and the second best is 28.81. (B and E): The selected attack nodes (red) by NIRM are fewer than those of one-pass dismantling: 37.52% vs. 55.35% in Roget, and 22.60% vs. 34.98% in HM. (C and F): the NGCC when removing different fractions of target attack nodes. (D and G): the dismantling performance  $\rho$  against different dismantling thresholds  $\Theta$ .

- BC (Betweenness Centrality) [23]. BC is a path-based metric that counts the number of shortest paths through node.
- CC (Closeness Centrality) [7]. CC measures the average distance between the node and other nodes.
- HC (Harmonic Centrality) [8]. HC is a variant of the CC algorithm that can be applied to disconnected network.
- PC (Percolation Centrality) [39]. PC quantifies relative impact of nodes based on their topological connectivity, as well as their percolation states.

**Adaptive approaches**, we also explore the performance of adaptive approaches on network dismantling, details of these approaches are described as follows:

- DC [2]. The adaptive version of DC, which recalculates the node degree centrality in the remaining network after each removal.
- CI [36]. The adaptive version of CI, which measure the importance of nodes based on the local structure of a ball of radius around every node.
- BPD [37]. BPD considers the spin glass theory and proposes a belief propagation-guided decimation algorithm. It first remove nodes until eliminating all the loops in the residual network. After that, BPD check the size of tree components and iteratively removes the root node.
- CoreHD [57]. CoreHD is similar with BPD in that both have eliminating loops and tree breaking. First remove the nodes with the highest degree from the 2-core of the network in

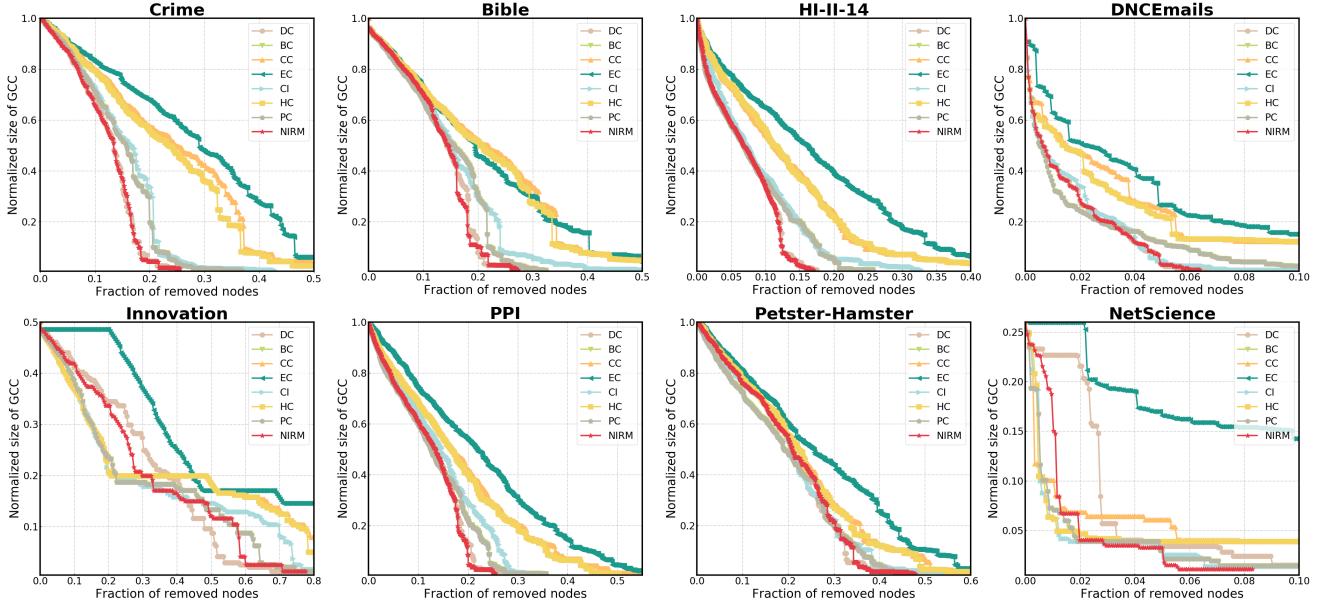
an adaptive way. After the 2-core is empty, it adopts greedy tree-breaking algorithm.

- GND [41]. GND proposes a spectral approach of node-weighted Laplacian operator. In each step, it partitions the remaining network into two components, and remove a set of nodes which covers all of the edges between two non-overlapping components at minimal cost.
- FINDER [22]. FINDER is a reinforcement learning-based method. It encodes the state (remaining network) and all possible actions (remove node) into embedding vectors, so that it take action  $a$  that represents the maximum expected rewards given state  $s$ .
- NSKSD [4]. NSKSD consider the overlapping effects between nodes, and introduced novel metrics KSD and NS. It proposed the double-turns selection mechanism to remove nodes in an adaptive way.

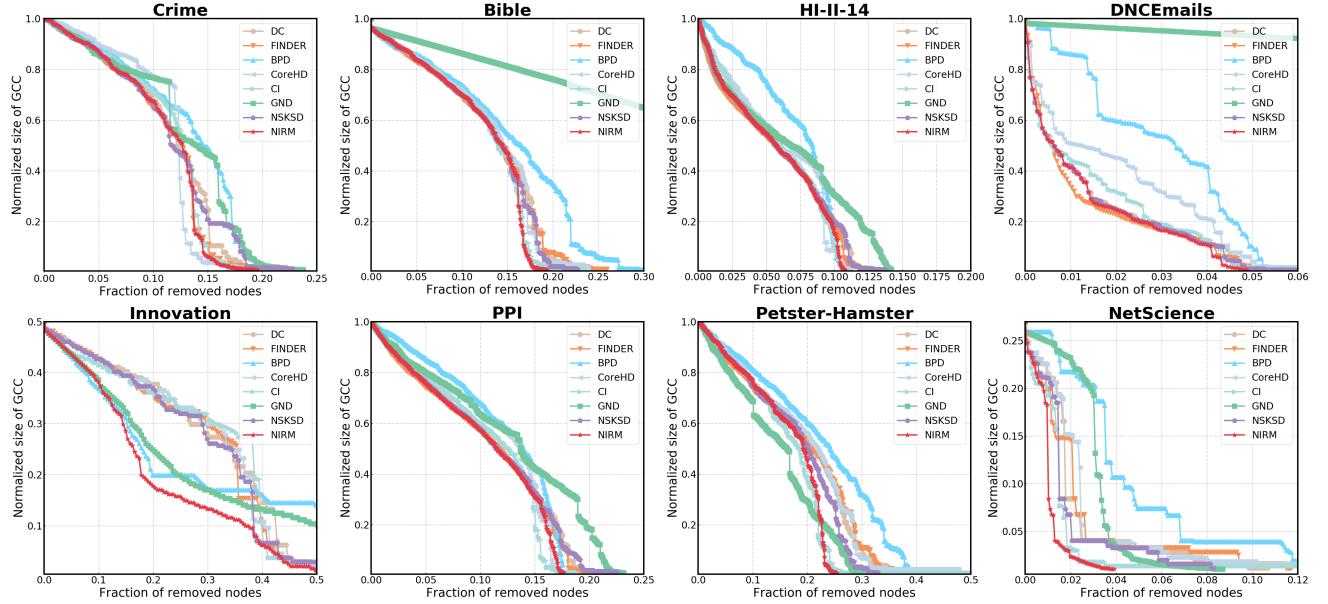
Note that some centrality-based approaches, e.g. BC, CC, EC, are not included in adaptive approaches, though they can be applied for adaptive dismantling, their adaptive versions are with great computation burdens yet not satisfactory dismantling performance. In addition, we implement NIRM as well as other state-of-the-art approaches without node reinsertion.

## 5.4 Performance evaluation

We first compare the one-pass dismantling performance of our NIRM with that of seven commonly used ranking metrics. Fig. 3 A presents the dismantling performance in terms of the *normalized*



**Figure 5: The NGCC when removing different fractions of target attack nodes under one-pass dismantling.**



**Figure 6: The NGCC when removing different fractions of target attack nodes under adaptive dismantling.**

size of TAS (denoted by  $\rho$ , and  $\rho = |\mathcal{V}_{tas}| / |\mathcal{V}|$ ) for fifteen real-world networks, when setting the dismantling threshold  $\Theta = 0.01$ . Fig. 3 B and E visualize the attack nodes (red) in Roget and HM to illustrate that our model is capable of identifying critical nodes for network dismantling. For different dismantling thresholds  $\Theta$  in Roget and HM, as shown in Fig. 3 D and G, NIRM consistently outperforms competitors. Our NIRM achieves the smallest  $\rho$  in twelve real-world networks, and three smallest in three networks. On average, the

NIRM selects about 36.58% attack nodes to dismantle a network; While the second best one requires attacking 40.41% nodes.

We next focus on the adaptive dismantling strategies. In general, as each iteration of selection-and-removal is for a particular input network, it can be expected that the performance of adaptive algorithm would be better than its one-pass version. We note that although the BC, CC, HC, PC and EC algorithm can also be applied for adaptive dismantling, they are not included for comparison, as

**Table 3: Ablation study of dismantling performance  $\rho$  when using node intermediate scores and final scores to select attack nodes.**

(A) Adaptive dismantling						
Methods	UsPower	P-H	Ca-GrQc	Infectious	Bible	HM
NIRM-IS	12.37	53.95	32.23	65.61	41.06	26.43
NIRM-GS	67.86	89.35	74.43	96.83	83.42	89.24
NIRM-LS	9.05	27.55	13.32	55.85	20.47	24.17
NIRM	<b>8.58</b>	<b>25.70</b>	<b>11.98</b>	<b>54.88</b>	<b>19.35</b>	<b>22.60</b>

(B) One-pass dismantling						
Methods	UsPower	P-H	Ca-GrQc	Infectious	Bible	HM
NIRM-IS	18.94	86.80	35.95	79.02	31.75	48.44
NIRM-GS	98.91	96.45	97.55	98.78	94.47	97.42
NIRM-LS	18.32	49.15	24.63	81.71	28.26	38.64
NIRM	<b>16.81</b>	<b>47.90</b>	<b>24.29</b>	<b>76.83</b>	<b>27.35</b>	<b>34.98</b>

their adaptive versions are with great computation burdens yet with not satisfactory dismantling performance. On the other hand, we include seven state-of-the-art schemes for performance comparison, including DC [2], CI [36], BPD [37], CoreHD [57], GND [41], FINDER [22] and NSKSD [4].

Fig. 4 A compares the adaptive dismantling performance for the same fifteen real-world networks. It is not unexpected that the adaptive versions of NIRM, DC, and CI requires fewer attack nodes to meet the same dismantling threshold. Among the fifteen real-world networks, our NIRM achieves the best in fourteen ones, and second best in one. Furthermore, on average, our NIRM requires to attack only 24.76% nodes, with an improvement of 4.05% compared with the second best NSKSD requiring 28.81% attack nodes. Fig. 4 C and F plot the NGCC when removing different fractions of target attack nodes. We observe that the NIRM can often achieve the smallest NGCC against the same number of removed nodes, indicating our model better captures the response of system throughout the whole attack process. In addition, the area under such a curve can evaluate the average attack efficiency of a dismantling scheme. The areas of our NIRM are 248.27, and 250.84 in HM, and Roget, respectively, much smaller than other current popular methods FINDER of 259.81, CI of 253.56 in the corresponding network. See Fig. 5 and Fig. 6 for more dismantling results on eight real-world networks.

Table 3 reports the results of our ablation study, where six real-networks are used to examine the normalized size of TAS performance of adaptive and one-pass dismantling. Recall that in our NIRM, three intermediate scoring vectors, i.e., initial scores, local scores and global scores, can also be extracted for ranking nodes and dismantling networks. It can be seen that although local scores can already achieve competitive dismantling performance, its fusion with global scores further reduces target attack nodes. This collaborates our design objective of using a global projection to further make up network-wide comparisons of nodes' influences.

## 6 CONCLUSION

Although many heuristic algorithms have been proposed for tackling the network dismantling problem, little has been done on

employing recent deep learning techniques. The main challenge is from the fact that real world networks are with diverse sizes and distinct characteristics, which seems to discourage the use of a single neural model for different networks. In addition, training a neural model faces the difficulties of unknown ground truth labels, especially for large networks. Nonetheless, this article has provided an insightful trial of designing and training a neural influence ranking model for network dismantling. The key design philosophy is to encode local structural and global topological characteristics of each node for ranking its influence to network stability. Another factor of success is from our score propagation for only recruiting tiny synthetic networks for model training.

Experiments on fifteen real-world networks have validated our NIRM as a promising solution to the network dismantling task. Yet we would like to note again that it is this style of training from tiny but applying for real-world networks, could open a new window for further investigations. Indeed, we acknowledge that our neural model is far from perfection, and many advances can be expected. In NIRM, only a few local attributes are crafted as initial features. A possible extension is to also take care of the node-centric motifs [47] for feature encoding. Also our global project kernels are somewhat plain, while sophisticated modules like Parametric UMAP [45] and techniques like contrastive learning [55] can also be tried. Finally, we would like to expect further investigations on more delicate labeling strategies as well as training mechanisms.

## ACKNOWLEDGMENTS

This work is supported in part by National Natural Science Foundation of China (Grant No: 62172167). We also want to use our NIRM model on MindSpore<sup>1</sup>, which is a new deep learning computing framework. These problems are left for future work.

## REFERENCES

- [1] Réka Albert, István Albert, and Gary L Nakarado. 2004. Structural vulnerability of the North American power grid. *Phys. Rev. E* 69, 2 (2004), 025103.
- [2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *Nature* 406, 6794 (2000), 378–382.
- [3] Nahuel Almeira, Orlando Vito Billoni, and Juan Ignacio Perotti. 2020. Scaling of percolation transitions on Erdős-Rényi networks under centrality-based attacks. *Phys. Rev. E* 101, 1 (2020), 012306.
- [4] Yiguang Bai, Yudong Gong, Qian Li, Wenjing Song, Ahmed Almjai, and Sanyang Liu. 2021. NSKSD: Interdependent Network Dismantling via Nonlinear-Metric. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69, 3 (2021), 1722–1726.
- [5] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *Science* 286, 5439 (1999), 509–512.
- [6] Vladimir Batagelj and Andrej Mrvar. 2006. Pajek datasets. <http://vlado.fmf.uni-lj.si/pub/networks/data/>.
- [7] Alex Bavelas. 1950. Communication patterns in task-oriented groups. *The journal of the acoustical society of America* 22, 6 (1950), 725–730.
- [8] Paolo Boldi and Sebastiano Vigna. 2014. Axioms for Centrality. *Internet Mathematics* 10, 3–4 (2014), 222–262.
- [9] Phillip Bonacich. 1987. Power and centrality: A family of measures. *American journal of sociology* 92, 5 (1987), 1170–1182.
- [10] Alfredo Braunstein, Luca Dall'Asta, Guilhem Semerjian, and Lenka Zdeborová. 2016. Network dismantling. *Proceedings of the National Academy of Sciences* 113, 44 (2016), 12368–12373.
- [11] Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, and Runsheng Chen. 2003. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic Acids Research* 31, 9 (May 2003), 2443–2450.
- [12] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. 2010. Catastrophic cascade of failures in interdependent networks. *Nature* 464, 7291 (2010), 1025–1028.

<sup>1</sup><http://www.mindspore.cn/>

- [13] Duncan S. Callaway, M. E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. 2000. Network Robustness and Fragility: Percolation on Random Graphs. *Phys. Rev. Lett.* 85 (2000), 5468–5471. Issue 25.
- [14] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. 2021. Combinatorial Optimization and Reasoning with Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. International Joint Conferences on Artificial Intelligence Organization, 4348–4355.
- [15] Vittoria Colizza, Alain Barrat, Marc Barthélémy, and Alessandro Vespignani. 2006. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences* 103, 7 (2006), 2015–2020.
- [16] Cyrus Cousins, Chloe Wohlgemuth, and Matteo Riondato. 2021. Bavarian: Betweenness Centrality Approximation with Variance-Aware Rademacher Averages. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, New York, NY, USA, 196–206.
- [17] Paolo Crucitti, Vito Latora, and Massimo Marchiori. 2004. A topological analysis of the Italian electric power grid. *Physica A: Statistical mechanics and its applications* 338, 1-2 (2004), 92–97.
- [18] Lucas Cuadra, Sancho Salcedo-Sanz, Javier Del Ser, Silvia Jiménez-Fernández, and Zong Woo Geem. 2015. A critical review of robustness in power grids using complex networks concepts. *Energy* 8, 9 (2015), 9211–9265.
- [19] Hanjun Dai, Elias B Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. 2017. Learning combinatorial optimization algorithms over graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6351–6361.
- [20] Paul Erdős, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* 5, 1 (1960), 17–60.
- [21] Changjun Fan, Li Zeng, Yuhui Ding, Muhan Chen, Yizhou Sun, and Zhong Liu. 2019. Learning to Identify High Betweenness Centrality Nodes from Scratch: A Novel Graph Neural Network Approach. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 559–568.
- [22] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. 2020. Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence* 2, 6 (2020), 317–324.
- [23] Linton C Freeman. 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry* 40, 1 (1977), 35–41.
- [24] Felipe Grando, Lisandro Z. Granville, and Luis C. Lamb. 2018. Machine Learning in Network Centrality Measures: Tutorial and Outlook. *ACM Computing Surveys (CSUR)* 51, 5 (Oct 2018), 32 pages.
- [25] Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. 2021. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nature Communications* 12, 1 (2021), 5190–5200.
- [26] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 1025–1035.
- [27] Mattias Höglund, Attila Frigyesi, and Felix Mitelman. 2006. A gene fusion network in human neoplasia. *Oncogene* 25, 18 (2006), 2674–2678.
- [28] Petter Holme and Beom Jun Kim. 2002. Growing scale-free networks with tunable clustering. *Phys. Rev. E* 65 (Jan 2002), 026107.
- [29] Goodwin Jazmin. 2020. Major internet outage: Dozens of websites and apps were down. <https://edition.cnn.com/2020/08/30/tech/internet-outage-cloudflare/index.html>.
- [30] Brian Karrer, M. E. J. Newman, and Lenka Zdeborová. 2014. Percolation on Sparse Networks. *Phys. Rev. Lett.* 113 (2014), 208702. Issue 20.
- [31] Jérôme Kunegis. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd international conference on World Wide Web*, 1343–1350.
- [32] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. <http://snap.stanford.edu/data>.
- [33] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2019. Fast Approximations of Betweenness Centrality with Graph Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 2149–2152.
- [34] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2021. Graph Neural Networks for Fast Node Ranking Approximation. *ACM Trans. Knowl. Discov. Data* 15, 5 (May 2021).
- [35] Matheus R. F. Mendonça, André M. S. Barreto, and Artur Ziviani. 2021. Approximating Network Centrality Measures Using Node Embedding and Machine Learning. *IEEE Transactions on Network Science and Engineering* 8, 1 (2021), 220–230.
- [36] Flaviano Morone and Hernán A Makse. 2015. Influence maximization in complex networks through optimal percolation. *Nature* 524, 7563 (2015), 65–68.
- [37] Salomon Mugisha and Hai-Jun Zhou. 2016. Identifying optimal targets of network attack by belief propagation. *Phys. Rev. E* 94, 1 (2016), 012305.
- [38] Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. 2019. Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 596–606.
- [39] Mahendra Piraveenan, Mikhail Prokopenko, and Liaquat Hossain. 2013. Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks. *Plos one* 8, 1 (2013), e53095.
- [40] Appan Rakaraddi and Mahardhika Pratama. 2021. Unsupervised Learning for Identifying High Eigenvector Centrality Nodes: A Graph Neural Network Approach. In *2021 IEEE International Conference on Big Data (Big Data)*, 4945–4954.
- [41] Xiao-Long Ren, Niels Gleinig, Dirk Helbing, and Nino Antulov-Fantulin. 2019. Generalized network dismantling. *Proceedings of the National Academy of Sciences* 116, 14 (2019), 6554–6559.
- [42] Haroldo V Ribeiro, Luiz G A Alves, Alvaro F Martins, Ervin K Lenzi, and Matjaž Perc. 2018. The dynamical structure of political corruption networks. *Journal of Complex Networks* 6, 6 (2018), 989–1003.
- [43] Thomas Rolland, Murat Taşan, Benoit Charlotteaux, Samuel J Pevzner, Quan Zhong, Nidhi Sahni, Song Yi, Irma Lemmens, Celia Fontanillo, Roberto Mosca, et al. 2014. A proteome-scale map of the human interactome network. *Cell* 159, 5 (2014), 1212–1226.
- [44] Abbas Ali Saberi. 2015. Recent advances in percolation theory and its applications. *Physics Reports* 578 (2015), 1–32.
- [45] Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. 2021. Parametric UMAP Embeddings for Representation and Semisupervised Learning. *Neural Computation* 33, 11 (Oct 2021), 2881–2907.
- [46] Martin JA Schuetz, J Kyle Brubaker, and Helmut G Katzgraber. 2022. Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence* 4, 4 (2022), 367–377.
- [47] Ping Shao, Yang Yang, Shengyao Xu, and Chunping Wang. 2021. Network Embedding via Motifs. *ACM Trans. Knowl. Discov. Data* 16, 3, Article 44 (2021), 20 pages.
- [48] Qi Song, Bo Zong, Yinghui Wu, Lu-An Tang, Hui Zhang, Guofei Jiang, and Haifeng Chen. 2018. TGNet: Learning to Rank Nodes in Temporal Graphs. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 97–106.
- [49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.
- [50] T. Verma, N. A. M. Araújo, and H. J. Herrmann. 2014. Revealing the structure of the world airline network. *Scientific Reports* 4, 1 (2014), 5638.
- [51] Yaojing Wang, Guosheng Pan, Yuan Yao, Hanghang Tong, Hongxia Yang, Feng Xu, and Jian Lu. 2020. Bringing Order to Network Embedding: A Relative Ranking Based Approach. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 1585–1594.
- [52] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of ‘small-world’ networks. *Nature* 393, 6684 (1998), 440–442.
- [53] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
- [54] Soon-Hyung Yook, Hawoong Jeong, and Albert-László Barabási. 2002. Modeling the Internet’s large-scale topology. *Proceedings of the National Academy of Sciences* 99, 21 (2002), 13382–13386.
- [55] Yunming You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., Red Hook, NY, USA, 5812–5823.
- [56] En-Yu Yu, Yan Fu, Xiao Chen, Mei Xie, and Duan-Bing Chen. 2020. Identifying critical nodes in temporal networks by network embedding. *Scientific Reports* 10, 1 (2020), 1–8.
- [57] Lenka Zdeborová, Pan Zhang, and Hai-Jun Zhou. 2016. Fast and simple decycling and dismantling of networks. *Scientific Reports* 6, 1 (2016), 37954.
- [58] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.