


Transformer with Sparse Adaptive Mask for Network Dismantling

Yu Liu[†], Fanghao Hu[†], Haojun Huang, and Bang Wang 


School of Electronic Information and Communications, Huazhong University of
Science and Technology (HUST), Wuhan 430074, China.
{yuliu_, hfh, hjhuang, wangbang}@hust.edu.cn

Abstract. The task of network dismantling aims to attack the least number of critical nodes to decompose a network into many small sub-networks. Recent approaches design task-oriented neural models to encode nodes’ structural features for predicting their importance. Instead of crafting small models, an interesting question is about whether and how large models like Transformers can be exploited for this classic yet NP-hard task in the network science domain. This paper provides an affirmative answer. The key lies in how to enable a Transformer to encode nodes’ representations based on comparisons over their importance to network integrity. In this paper, we propose to encode node egonet characteristics as well as internode spatial dependences from a global view. Furthermore, for each node encoding, we propose to include peer attention to enable networkwide importance comparison. A new fusion module with a sparse adaptive mask is designed into the Transformer architecture for encoding node comparative importance to network integrity. Experiments on real-world networks and synthetic networks validate the effectiveness of our design over the state-of-the-art schemes. The source code and datasets are available at: <https://github.com/valyentine/TSAM>.

Keywords: Network Dismantling · Transformer with Sparse Attention Mask · Network Science · Representation Learning and Ranking.

1 Introduction

Numerous real-world physical systems can be abstracted as complex networks, where each network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with the node set \mathcal{V} and edge set \mathcal{E} defining its topology. Percolation theory [28] demonstrates that the failure of a small fraction of nodes can cause a large-scale network to disintegrate into numerous disconnected small subnetworks, leading to network instability or even collapse. A notable example is the Century Link outage in the United States on August 30, 2020, triggered by the malfunction of its critical backbone router AS3356 [19]. This incident highlights the significance of the *network dismantling* (ND) problem.

[†] The first two authors contribute equally to this work.  Bang Wang is the corresponding author.

Following the literature [5,34,29], the ND problem can be formally defined as to find the smallest subset of nodes, termed the *target attack node set* (TAS), whose removal disintegrates a network into disconnected components such that the size of the *giant connected component* (GCC) falls below a predefined threshold θ . Formally, the ND problem seeks to find the optimal TAS $\mathcal{V}^* \subseteq \mathcal{V}$:

$$\mathcal{V}^* = \arg \min_{\mathcal{V}_t \subseteq \mathcal{V}} \{|\mathcal{V}_t| : |\mathcal{G}_t|/|\mathcal{G}| \leq \theta\}, \quad (1)$$

where $|\mathcal{V}_t|$ is the cardinality of the TAS, $|\mathcal{G}_t|$ denotes the number of nodes in the GCC after removing \mathcal{V}_t , $|\mathcal{G}|$ is the original network size. The ND problem has been proven as non-deterministic polynomial hard (NP-hard) [5].

For tackling the ND problem, early efforts have proposed many centrality-based heuristic solutions, i.e., the degree centrality [1], collective influence [25], PageRank [26] and etc., to rank a node importance for constructing the TAS. However, they tend to be susceptible to interference from particular structures, while neglecting to compare the global competence among nodes. In the last few years, a few studies have employed *graph neural networks* (GNN) to learn nodes' representations from their local structures and/or global topologies for their comparison and ranking to the importance of network dismantling [37,38,22,23]. Compared to the centrality-based methods, these ND-oriented neural network approaches exhibit superior performance, yet they predominantly employ relatively small models with few parameters.

Recently, some researches have employed Transformers to address graph-related tasks, such as the node classification, link prediction and etc.[7,39,36]. The global attention mechanism of Transformer enables to encode nodes' representations with attentions to other nodes', thus can help discovering global relations in an encoding process for downstream tasks. However, due to the $\mathcal{O}(n^2)$ computation complexity of Transformer gradient backpropagation updates, it is difficult to extend them to large graphs. Some researches propose to apply a sparse attention mask in Transformers for reducing computation complexity[4,27], that is, the attention only takes care of an extended local structure of a node with its few hops away neighbors. Nevertheless, such approaches risk compromising latent dependencies in between far apart nodes.

To the best of our knowledge, we note that the application of large models such as Transformers to the ND problem remains unexplored. For the ND problem, the removal of a node and its associated edges directly impact on its local structure, however only local structure cannot necessarily reflect its global competence to the network integrity. That is, even two nodes are with the same local structure, they may play different role in dismantling a large network. For example, two such nodes are close to each other, so only one of them is necessary to be included in the target set. If two such nodes are far away, it is also necessary to compare which one, if with other target nodes, can contribute more to the network integrity.

Motivated from the aforementioned considerations, we are interested to explore potentials on exploiting a large model like Transformer to address the classic ND problem. The basic idea is to use a Transformer to encode nodes'

representations for their ranking of importance to dismantle a network. For attribute-less nodes, we propose to iteratively encode the local structure of a node and its global relations to other nodes into its representation. For the dismantling task, we propose to compare the importance to network integrity for all nodes. A fusion model is designed to take care of both topological information and task importance for representation learning.

This paper proposes a *Transformer with Sparse Adaptive Mask* (TSAM) for network dismantling, where nodes’ representations are iteratively updated with attention bias and mask for ranking their importance. In particular, the TSAM takes learnable degree ranking encodings and last round nodes’ representations as input, and the output is the nodes’ representations that are then converted to ranking scores via a linear layer. Top-ranked nodes are selected as the target attack ones. In the Transformer architecture, we propose to include a new fusion module, which fuses topological biases with a sparse adaptive mask to update the original Transformer attention matrix. The topological biases consists of local structural bias via degree ranking encodings and global relational bias via shortest-path encodings. The sparse adaptive mask consists of neighbor mask and peer mask. The peer mask is constructed by comparing nodes of similar representations, as they would score similar dismantling importance even if they are far apart in the input network. Experiments are conducted on eleven typical real-world networks and four synthetic network types. Results validate the superiority of our TSAM over the state-of-the-art schemes in terms of using fewer attack nodes in most cases.

2 Related Work and Preliminary

2.1 Network Dismantling

Most existing network dismantling methods evaluate node importance based on certain centrality measures and select the top- K most critical nodes as the TAS. Commonly used approaches include degree centrality [1], betweenness centrality [12], closeness centrality [3], collective influence [25] and PageRank [26].

Recently, a few works have proposed neural network models to encode nodes’ structural features for scoring their importance to network integrity [11,22,38]. For example, the FINDER[11] employs a reinforcement learning framework that incorporates a node reinsertion mechanism, i.e., strategically excluding some previously selected attack nodes from the target set and reinserting them back to the input network. In contrast, our work addresses the one-path dismantling approach excluding node reinsertion, where the simultaneous removal of all nodes in the TAS is executed in one iteration. The NIRM [37] encodes local structures and global topological signals via a neural model trained over small synthetic networks. The NEES[22] designs a graph neural network to extract the core structure of the network and transforms it into a smaller-scale structure with fewer nodes and edges. The DCRS[38] constructs a role graph based on the original graph, then encodes and integrates the nodes’ propagation competitiveness and role importance to evaluate the nodes.

2.2 Transformer

We briefly introduce how one Transformer layer works, and multiple such layers can be stacked for complex tasks. Let the input to a layer be $\mathbf{H} \in \mathbb{R}^{n \times d}$, where n is the sequence length and d is the hidden dimension. Through learnable projection matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d \times d}$, the query \mathbf{Q} , key \mathbf{K} , and value \mathbf{V} are generated:

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{H}\mathbf{W}_k, \quad \mathbf{V} = \mathbf{H}\mathbf{W}_v.$$

The attention score matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is computed via scaled dot-product of \mathbf{Q} and \mathbf{K}^\top , which is then normalized by softmax. The layer output \mathbf{H}' is obtained by the product of normalized \mathbf{A} and \mathbf{V} :

$$\mathbf{A} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}, \quad \mathbf{H}' = \text{softmax}(\mathbf{A})\mathbf{V}.$$

Note that the above single-head self-attention module can be generalized into a multi-head attention via the concatenation operation.

In recent years, a number of Transformer-inspired models have been developed for some graph tasks, like the node classification, link prediction, molecular property prediction and etc.[7,39,36]. For example, the GraphTransformer[7] uses positional encodings based on the Laplacian eigenvectors to learn nodes' representations for the node classification task. Gophormer[39] combines hierarchical pre-trained language models with GNNs through a hybrid architecture, enabling joint modeling of local neighborhood aggregation and global attention for knowledge graph completion tasks. Graphormer[36] integrates three structural encodings into Transformer layers to enhance structural awareness, achieving competitive performance in molecular property prediction. However, to the best of our knowledge, there are currently no Transformer-based methods for the ND problem.

3 Transformer with Sparse Adaptive Mask

3.1 The TSAM Framework

Consider an input network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes and M edges. A Transformer can be used to iteratively encode nodes' representations $\mathbf{H} \in \mathbb{R}^{N \times d_h}$, where d_h is the representation dimension. After training, the output of the Transformer \mathbf{H} can be converted into dismantling scores via a linear layer for their ranking to construct the target set. During the Transformer encoding process, the attention mechanism necessitates computing pairwise attention scores in between all nodes. This imposes significant challenges for both parameter updates during backpropagation and memory consumption in large-scale networks, leading to $\mathcal{O}(n^2)$ computational complexity. Moreover, the fully-connected attention mechanism may degrade nodes' representational capacity by unnecessarily aggregating information from those nodes with few contribution to network dismantling.

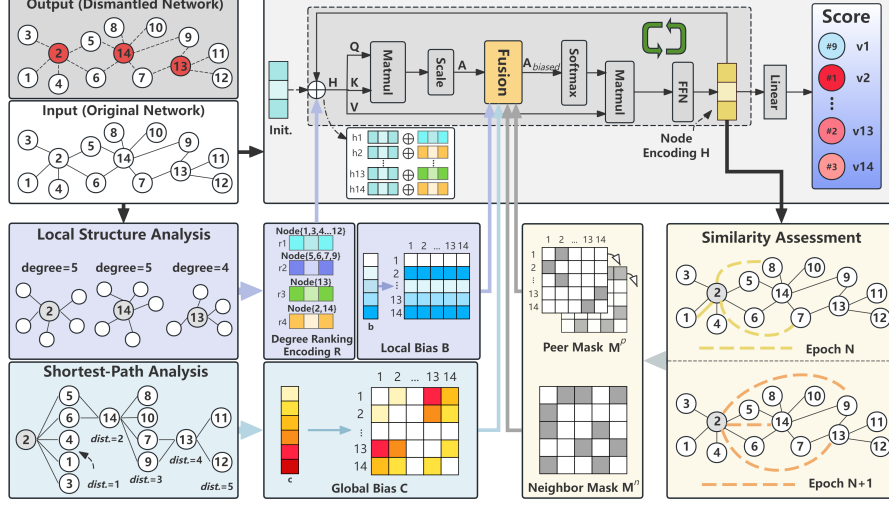


Fig. 1: The framework of Transformer with sparse adaptive mask (TSAM). The input is a connected network, and the output is nodes’ dismantling scores which are used to compose the target attack node set (TAS). The TSAM includes a new fusion module in the Transformer architecture, which fuses the Transformer attention matrix with two biases a sparse adaptive mask.

Fig. 1 presents the framework of our proposed Transformer with Sparse Adaptive Mask (TSAM) for network dismantling. In TSAM, the nodes’ representations H are first randomly initialized, which will be aggregated with a learnable degree ranking matrix R as the Transformer input. The TSAM includes a new fusion module into a Transformer architecture. The fusion module first fuses the Transformer attention matrix A with a local attention bias B and a global bias C to obtain a biased attention matrix. It next selects for each node its peers with similar representations to construct a peer mask M^p in each iteration. The sparse adaptive mask M is the bitwise OR of the neighbor mask M^n and the peer mask M^p , which is used to sparsify the biased attention matrix. During each training epoch, the updated nodes’ representations H pass a linear layer to obtain dismantling scores for loss computation and dismantling evaluation.

3.2 Degree Ranking Encoding

Node degree centrality serves as an important indicator for local structures: The higher degree, the higher perturbation of structure stability caused by a single node removal, and such structural perturbation might propagate via neighbors to cause cascaded disconnections. We note that in many networks, node degree distribution is often not continuous, that is, some degrees are not associated with any node. For example, the network in Fig. 1 is without nodes of degree 3.

Encoding the value of degree centrality could lead to computational inefficiency. Moreover, topological heterogeneity across networks results in functional divergence among nodes with identical degrees. Thus, we propose to encode ordinal degree rankings instead of the degree values. For an input network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let d_i denote the degree of a node $v_i \in V$, and $\mathcal{D} = \{d_i\}$ denote the set of nodes' degrees, which is sorted in an increasing order. For example, in Fig. 1, we have $\mathcal{D} = \{1, 2, 4, 5\}$ as the sorted degree set of the input network, where $|\mathcal{D}| = 4$. We encode node degree ranking by a learnable matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{D}| \times d_h}$. Notice that the i -th row \mathbf{R}_i indicates the encoding of the i -th element in \mathcal{D} . For a node v_i with degree d_i , let $\text{index}(\mathcal{D}, d_i)$ be an indexing function to return the index of d_i in the set \mathcal{D} , denoted by $r_i = \text{index}(\mathcal{D}, d_i)$. The degree ranking encoding of v_i is denoted by \mathbf{R}_{r_i} , which is aggregated with the node representation updated in the last epoch as the input to the Transformer. We note that the iterative encoding process is expected to enable the Transformer for learning nodes' representations with the understanding about the network topological information as well as the node competence to the dismantling task. The encoding of a node v_i is randomly initialized using the Xavier [13] method.

3.3 Topological Information as Attention Bias

The attention matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is used to allocate an attention of \mathbf{A}_{ij} by a node v_i for a node v_j . For network dismantling, we propose two attention biases, a local bias \mathbf{B} and a global bias \mathbf{C} to update \mathbf{A} into a biased attention matrix.

Encoding local attention bias When removing a node and its associated edges, its neighbors are the immediate sufferers, as they might be disconnected to the original network. Even worse, such disconnection might be propagated to incur cascaded disconnections of more other nodes. We propose to encode a learnable vector $\mathbf{b} \in \mathbb{R}^{1 \times |\mathcal{D}|}$ for each element in \mathcal{D} . That is, \mathbf{b}_i is a learnable scalar indicating the i -th element in \mathcal{D} . Based on \mathbf{b} , we construct the local bias matrix $\mathbf{B} \in \mathbb{R}^{N \times N}$ as follows: The i -th row of \mathbf{B} indicates the attention of node v_i to other nodes. For v_i with degree d_i , we also use $r_i = \text{index}(\mathcal{D}, d_i)$ to return the index of d_i in \mathcal{D} . Then we set all elements in the i -th row of \mathbf{B}_i as \mathbf{b}_{r_i} . We note that although the degree ranking \mathbf{R} and local bias \mathbf{B} are with the same encoding rationale, they are encoded and used as two different matrices.

Encoding global attention bias The removal of a single node not only impacts on the connection of its neighbors to the input network, it could also impact on the connections of other far away nodes. However, capturing such long range influences may not be easy in the graph structure. We propose to use the shortest-path distance to measure the relation in between two nodes, as the shortest-path is the most efficient way for information dissemination in a network, which, we argue, could also be efficient to disconnect nodes from the network and critical for network integrity. We hence propose to encode shortest-paths in a network to represent the global relation of two nodes.

Let $c_{i,j}$ denote the shortest-path between two nodes v_i and v_j . For example, in Fig. 1, we have $c_{2,5} = 1, c_{2,8} = 3$. For the input network \mathcal{G} , let \mathcal{C} denote the set of all possible shortest-paths among two nodes, which is sorted in an increasing order. We propose to encode a learnable vector $\mathbf{c} \in \mathbb{R}^{1 \times |\mathcal{C}|}$ for each element in \mathcal{C} . That is, \mathbf{c}_i is a learnable scalar indicating the i -th element in \mathcal{C} . Based on \mathbf{c} , we construct the global bias matrix $\mathbf{C} \in \mathbb{R}^{N \times N}$ as follows: For a node v_i , we use an index function $r_{i,j} = \text{index}(\mathcal{C}, v_i, v_j)$ to return the index of $c_{i,j}$ in \mathcal{C} . Then we set the element $\mathbf{C}_{ij} = \mathbf{c}_{r_{i,j}}$. Note that we set $\mathbf{C}_{ii} = 0$.

Attention with biases After obtaining \mathbf{B} and \mathbf{C} , we input them into the fusion model to update the Transformer attention matrix \mathbf{A} by

$$\mathbf{A}_{biased} = \mathbf{A} + \mathbf{B} + \mathbf{C},$$

to obtain a biased attention matrix \mathbf{A}_{biased} .

3.4 Fusion with Sparse Adaptive Mask

The biased attention matrix \mathbf{A}_{biased} enables the Transformer to have a network-wide view on comparing and updating nodes' representations, that is, the attention of one node is computed based on the representations of itself and all other nodes. However, this network-wide computation is with prohibitive costs, sometimes causing the so-called over-globalization problem. That is, the attention of a node is unnecessarily allocated and normalized to those nodes without enough importance to network integrity. To address this issue, we propose to construct a sparse mask for each node only caring for its neighbors and peers. The peers of a node are regarded with similar importance to network integrity and so with similar encodings, even they are topologically far apart in the input network. Furthermore, as nodes' encodings are iteratively updated in each training epoch, the peers of a node can also change in different epochs. As such, we construct a sparse adaptive mask for the fusion module.

Neighbor Mask. A node should pay more attention to its neighbors, as they represent its local structure and would be directly impacted due to its removal. We use the adjacency matrix of the input network as the neighbor mask, denoted by $\mathbf{M}^n \in \{0, 1\}^{N \times N}$. $\mathbf{M}_{ij}^n = 1$ indicates the existence of an edge between the node v_i and v_j ; Otherwise, $\mathbf{M}_{ij}^n = 0$. We note that the neighbor mask is not changed during the training epochs.

Peer Mask. The peer mask is used for a node to pay more attention to those nodes with similar importance to network integrity, even they could be topologically far away. By using a peer mask, the global competence to network integrity for two distant nodes can be compared. In order to enhance generalization capability, we propose to utilize the nodes' representations learned by the Transformer for competence evaluation.

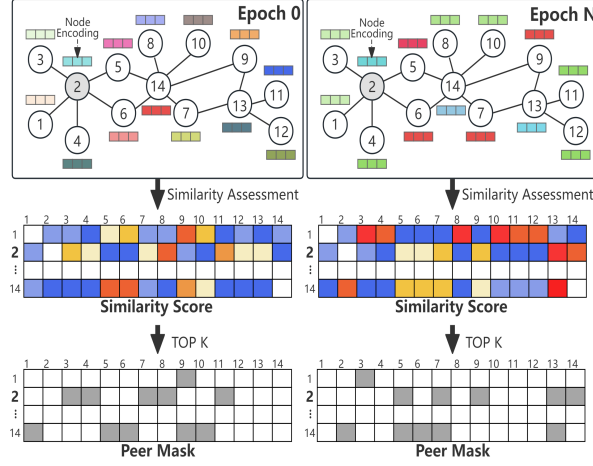


Fig. 2: Illustration of constructing a peer mask.

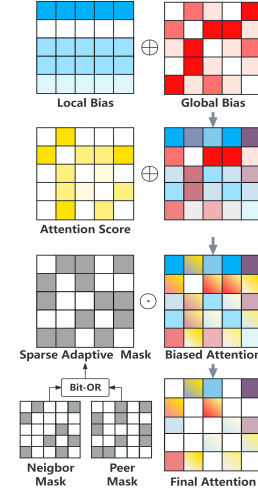


Fig. 3: Illustration of the fusion process.

Let $\mathbf{H} \in \mathbb{R}^{N \times d_h}$ denote the output of a Transformer layer, which will next pass a linear layer for scoring each node, as shown in Fig. 1. The i -th row of \mathbf{H} is the representation of node v_i learned by the Transformer. We compute the similarity between two nodes' representations, and construct a similarity matrix $\mathbf{S} \in \mathbb{R}^{N \times N}$ as follows:

$$\mathbf{S} = \left(\frac{\mathbf{H}}{\|\mathbf{H}\|_{\text{row}}} \right) \left(\frac{\mathbf{H}}{\|\mathbf{H}\|_{\text{row}}} \right)^{\top},$$

where $\|\mathbf{H}\|_{\text{row}}$ denotes the L2 normalization applied to each row of the matrix \mathbf{H} . For a node v_i with degree d_i , we only choose the top- K_i similar nodes, where K_i is computed by

$$K_i = \min(\max(d_i, \alpha_1), \alpha_2) \quad (2)$$

where α_1 and α_2 are hyperparameters to denote the minimum and maximum sampling quantities respectively. We set the values of the top- K_i elements in the i -th row of \mathbf{S} to 1 and the rest to 0, to obtain the peer mask \mathbf{M}^p .

It is worth noting that, in contrast to the neighbor mask \mathbf{M}^n , the peer mask \mathbf{M}^p is not fixed but subject to change in each training epoch. This ensures that the peer selection does not favor a specific type of nodes, but instead adaptively chooses more appropriate nodes as the Transformer gradually understands each node importance to network integrity.

Fusion with Sparse Adaptive Mask. The sparse adaptive mask \mathbf{M} is the result of bitwise OR operation on the neighbor mask \mathbf{M}^n and peer mask \mathbf{M}^p :

$$\mathbf{M} = \mathbf{M}^n \vee \mathbf{M}^p \quad (3)$$

Table 1: Statistics of the real-world networks.

Network	Nodes	Edges	Density	$\langle k \rangle$	ClustCoeff.	Diameter	Category
Chicago[8]	12,979	20,627	0.0002	3.18	0.0455	106	Transport
Euroroads(Er)[32]	1,039	1,305	0.0024	2.51	0.0339	62	Transport
AirTraffic(AT)[21]	1,226	2,408	0.0032	3.93	0.0639	17	Airport
Gnutella[24]	8,717	31,525	0.0008	7.23	0.0081	10	Internet
FilmTrust(FT)[16]	874	1,309	0.0034	2.99	0.1916	13	Social
LastFM[30]	7,624	27,806	0.0010	7.29	0.1786	15	Social
RoviraVirgili(RV)[15]	1,133	5,451	0.0085	9.62	0.1663	8	Email
PPI[6]	2,224	6,609	0.0027	5.94	0.1381	11	Protein
Figeys[10]	2,239	6,432	0.0026	5.75	0.0076	10	Protein
Vidal[31]	3,133	6,726	0.0014	4.29	0.0354	13	Protein
Genefusion(Gf)[17]	291	279	0.0066	1.92	0.0017	9	Biology

where \vee denotes the bitwise OR operation. The fusion process applies the \mathbf{M} to the biased attention matrix \mathbf{A}_{biased} to obtain the final attention matrix \mathbf{A}_{final} :

$$\mathbf{A}_{final}(i, j) = \begin{cases} \mathbf{A}_{biased}(i, j) & \text{if } \mathbf{M}_{i,j} = 1, \\ -\infty & \text{if } \mathbf{M}_{i,j} = 0. \end{cases} \quad (4)$$

Note that the (i, j) -element of the final attention matrix represents the attention score of v_i to v_j . In our experiments, we take negative infinity as a very small value (-10^6), so that after the softmax function, the attention score at this position will become 0. Fig. 3 shows the full process of fusion module. By leveraging the two masks, we can achieve a balance between local and global information while significantly reducing computation costs.

3.5 Scoring and Loss Function

Scoring. After obtaining the nodes' representations \mathbf{H} , we evaluate a dismantling score s_i for each node v_i by a linear layer:

$$s_i = \mathbf{W}_s \mathbf{H}_i^\top \quad (5)$$

where \mathbf{W}_s is the learnable matrix of the linear layer. The dismantling score s_i can be understood as the importance of v_i to network integrity. We finally select the top- K nodes with highest dismantling scores to form the TAS \mathcal{V}_t .

Loss Function. To remove the least number of nodes while ensuring the remaining components in the network are small enough, we adopt the loss function defined in [38]:

$$\mathcal{L} = \sum_{v_i \in \mathcal{V}} \prod_{v_j \in \mathcal{N}(v_i)} \frac{1}{1 + s_j} + \sum_{v_i \in \mathcal{V}} s_i, \quad (6)$$

Table 2: Comparison of normalized TAS sizes(%) on real-world networks with 0.01 dismantling threshold. The best is marked in **bold green**, the second best is marked in **bold**, while the third best is marked with underline.

Datasets	DC	BC	CI	PR	NV	GAT	GCN	NIRM	NEES	DCRS	TSAM
Chi.	50.8	55.58	60.42	51.56	70.74	77.73	76.65	46.58	<u>42.15</u>	35.3	33.03 _{+6.43%}
Er	41.29	48.99	82.19	33.69	46.2	87.78	89.51	38.11	<u>28.1</u>	23.39	22.42 _{+4.12%}
AT	32.79	50.98	68.03	28.14	73.82	76.59	98.04	<u>25.61</u>	26.43	23.57	21.70 _{+7.64%}
Gnu.	36.64	38.35	40.35	35.07	95.23	63.78	98.82	<u>35.03</u>	43.32	32.45	32.83 _{-1.17%}
FT	22.77	33.75	42.68	<u>22.20</u>	64.87	81.69	98.74	44.39	25.97	19.11	14.53 _{+24.0%}
Las.	31.69	40.11	48.6	<u>27.96</u>	79.55	61.23	98.95	70.96	31.47	26.02	25.47 _{+2.17%}
RV	48.46	54.55	59.31	<u>44.40</u>	94.88	86.41	98.94	45.01	52.34	43.07	41.31 _{+4.10%}
PPI	27.34	35.7	34.67	<u>24.19</u>	41.32	61.38	98.88	25.49	24.82	21.67	21.04 _{+3.53%}
Fig.	18.89	18.89	25.9	16.03	86.02	54.27	30.64	39.53	<u>10.05</u>	8.93	8.84 _{+1.00%}
Vid.	20.84	23.46	31.6	20.14	89.59	54.23	98.82	31.38	<u>17.3</u>	16.02	14.20 _{+11.35%}
Gf	19.24	21.65	82.82	13.4	78.69	76.63	98.28	<u>13.75</u>	19.59	11.34	11.34 _{+0.00%}

Table 3: Average TAS and standard deviation results from twenty instances generated by four synthetic models.

Datasets	DC	BC	CI	PR	NEES	DCRS	TSAM
PLC	35.16 \pm 2.42	40.99 \pm 2.65	54.70 \pm 5.92	33.69 \pm 2.27	42.01 \pm 3.21	32.19 \pm 1.71	31.78 \pm 1.49
BA	45.30 \pm 1.76	47.60 \pm 2.32	64.17 \pm 3.58	<u>43.23</u> \pm 1.88	53.40 \pm 2.78	41.47 \pm 1.60	40.56 \pm 1.79
ER	65.95 \pm 2.99	65.66 \pm 1.90	72.74 \pm 3.84	<u>59.00</u> \pm 1.54	63.66 \pm 4.35	54.61 \pm 2.20	50.51 \pm 1.66
WS	77.13 \pm 2.53	76.92 \pm 2.36	82.04 \pm 3.18	<u>73.01</u> \pm 3.66	76.65 \pm 3.30	63.98 \pm 1.30	63.80 \pm 1.34

where $\mathcal{N}(v_i)$ stands for the neighbor set of node v_i . The first term represents the expected number of unaffected nodes after the removal of a node. $\frac{1}{1+s_j}$ is inversely proportional to the importance of v_j for network integrity. The number of unaffected nodes is estimated by evaluating the collective effect within the egonet after the removal of v_i . To minimize the number of targeted attack nodes, the sum of decomposition scores is used as a regularization term.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate the effectiveness of our TSAM on both real-world networks and synthetic networks.

Real-world Networks. We selected 11 real-world networks of diverse network scales and different topological structures spanning multiple domains, including society, internet, biology, collaboration and etc. Table 1 summarizes the statistics of these networks.

Synthetic Networks. We employ four standard generative models to get synthetic network data: the WS (Watts-Strogatz) [35], BA (Barabási-Albert) [2],

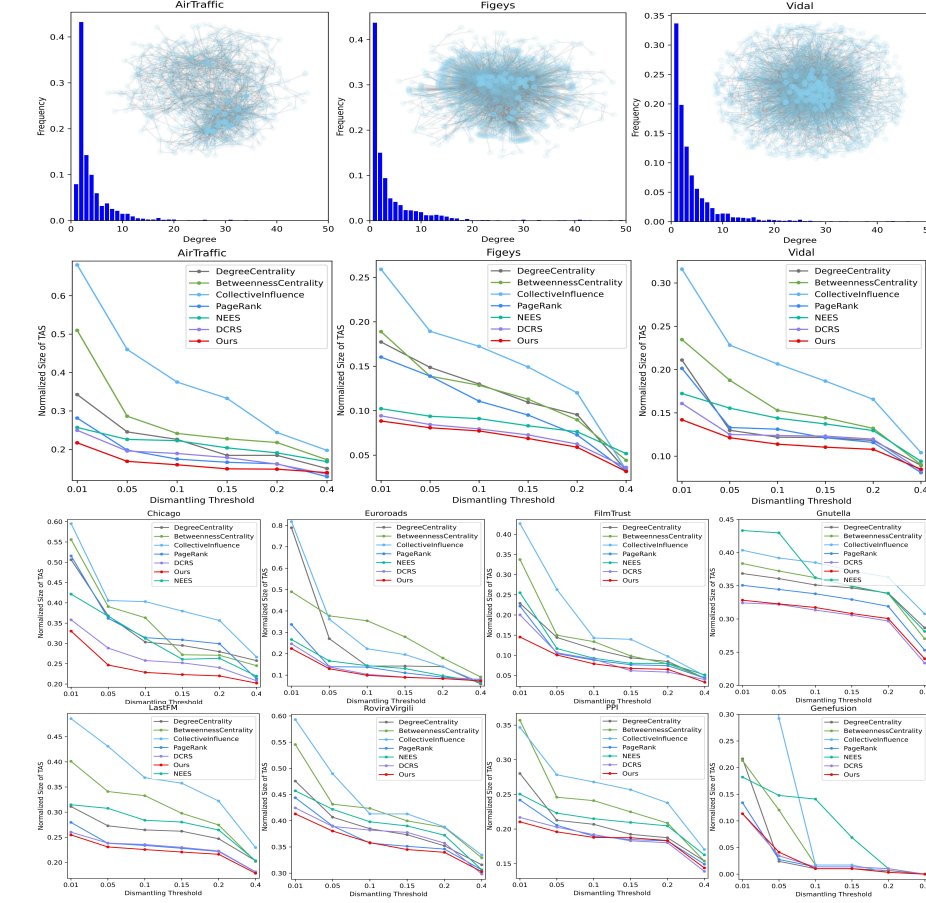


Fig. 4: Results of three real-world networks. Top: Network topology and degree distribution visualizations. Bottom: Dismantling performance of different models across varying dismantling thresholds on each network.

PLC (Powerlaw-Cluster) [18], and ER (Erdős-Rényi) [9]. We use each model to generate 20 synthetic instances and average the results.

Competitors. We compare the TSAM with the two types of competitors: (1) four centrality-based schemes: DC (Degree Centrality) [1], BC (Betweenness Centrality) [12], CI (Collective Centrality) [25], and PR (PageRank) [26]; (2) six neural model-based schemes: the NV (Node2Vec) [14], GCN [20], GAT [33], NIRM [37], NEES [22], and DCRS [38].

Implementation Details. In our experiments, we use one Transformer layer and one attention head. All experiments are implemented in the Linux operating system using an NVIDIA GeForce RTX 4090 with 24 GB memory, based on the Pytorch version 2.4.0, cuda 11.8, and Python 3.9.

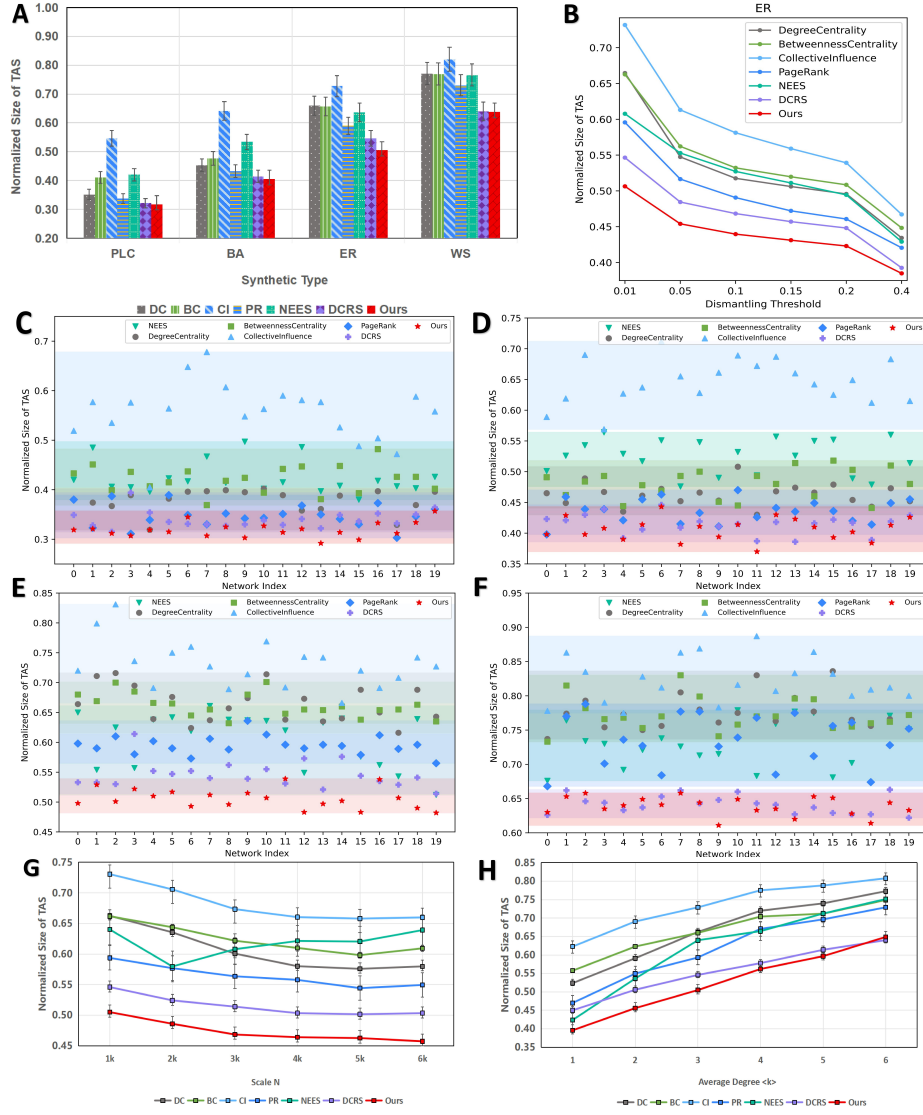


Fig. 5: Comparison of dismantling Performance on synthetic networks. (A): TAS comparison over four synthetic networks (network size $N = 1000$, dismantling threshold $\theta = 0.01$). (B): TAS against different dismantling thresholds on ER networks. (C,D,E,F): the result of the normalized TAS on PLC, BA, ER and WS models for the 20 experimented instances, respectively. (G): Dismantling performance on ER networks with fixed average degree $\langle k \rangle = 6$ and varying network sizes N from 1000 to 6000. (H): TAS comparison on ER networks with fixed $N = 1000$ and varying $\langle k \rangle$ from 4 to 9.

4.2 Experimental Results

Real-world Networks. Table 2 presents the normalized TAS over network size for the real-world networks, when setting the dismantling threshold to 0.01. The smaller the normalized TAS, the more effective the network dismantling is, as it requires fewer target nodes to be removed. From the table, we can observe that our TSAM achieves the best performance in terms of the smallest normalized TAS in 10 out of 11 real-world networks, showing its superiority over the competitors. On the Gnu. real-world network, our model plays the second best, with a slightly more nodes than the best one.

Taking the real-world networks of the AirTraffic, Figeys and Vidal as examples, we visualize the network nodes and degree distribution in Fig. 4 to demonstrate the differences in their structures and characteristics. Facing these structurally heterogeneous real-world networks, achieving superior performance on all of them with a single model is highly challenging. The results demonstrate its generalization capability and resilience to topological differences. Fig. 4 also plots the dismantling performance in terms of the normalized TAS under different dismantling threshold θ for the eleven real-world networks. It can be observed that our TSAM also outperforms the competitors in most cases.

Synthetic networks. We select and compare six competitive methods with our TSAM on synthetic networks, as presented in Table 3. Fig 5 (A) plots the mean normalized TAS size and the standard deviation for the seven schemes, where each result is averaged over 20 generated network instances. Instances generated from the same synthetic model share similar network characteristics, while differences can exist across instances even using a same synthetic model with the same parameters. For example, the WS model generates networks with high clustering coefficients and small-world properties, whereas the PLC model produces networks exhibiting both scale-free degree distributions and hierarchical clustering structures. Results show that our TSAM achieves the best performance in synthetic networks.

Fig. 5 (B) plots the normalized TAS size under different θ in ER networks as well, where our TSAM performs the best in terms of the smallest average results. Fig. 5 (C,D,E,F) plot the normalized TAS for each experimented instances, where our TSAM achieves the smallest value in most instances and with a smaller variance. Note that although our TSAM outperforms the others in terms of averaged normalized TAS size, it may not be the best one in every network instance. We further investigate the dismantling performance of our TSAM on the ER networks generated with different parameters. Fig. 5 (G) and (H) respectively plot the results for fixing the network average degree $\langle k \rangle = 6$ and increasing the network size N , and for fixed $N = 1000$ and varying $\langle k \rangle$ from 4 to 9. Our TSAM demonstrates robust performance in both scenarios, achieving competitive results comparable to state-of-the-art schemes.

4.3 Ablation Experiment

To verify the effectiveness of individual modules in TSAM, we conduct ablation experiments on the six key components, including the three structural encodings (w/o DRE: degree ranking encoding; w/o LB: local bias; w/o GB: global bias), two sparse masks (w/o NM: neighbor mask; w/o PM: peer mask), and the w/o IE: iterative encoding update mechanism. Table 4 presents the results of ablation experiments. It can be observed that removing any component degrades the model performance across all evaluated networks. This performance degradation validates the essential role of each module in the network dismantling task.

Table 4: Ablation study results on the real-world networks with $\theta = 0.01$.

Methods	Chi.	Er	AT	Gnu.	FT	Las.	RV	PPI	Fig.	Vid.	Gf
w/o DRE	49.65	56.02	69.17	79.86	46.68	60.99	77.05	57.06	52.30	40.82	49.83
w/o LB	33.31	22.51	22.43	33.04	14.53	25.47	42.81	21.17	9.20	16.69	11.34
w/o GB	35.52	23.10	22.84	36.07	19.45	33.85	41.40	22.71	13.04	17.30	11.68
w/o NM	43.58	35.51	37.93	57.80	28.15	67.92	77.67	41.95	53.37	43.47	19.59
w/o PM	33.28	26.37	23.90	32.93	18.88	31.22	41.57	23.56	14.92	17.34	11.34
w/o IE	34.86	29.45	22.43	35.31	16.13	29.11	43.51	24.37	9.78	15.19	11.34
TSAM	33.03	22.42	21.7	32.83	14.53	25.47	41.31	21.04	8.84	14.2	11.34

For the nodes' representations, the effect of eliminating the DRE (degree ranking encoding) part is generally lower than eliminating only the IE part. This may be due to the fact that by removing the DRE component, only global information is used for representations without taking into account local information, while local structural information usually plays a significant role in the ND problem. For similar reasons, the impact of eliminating the NM (neighbor mask) part has a greater effect than removing other bias or mask part in the fusion module. Although the local structure information is important, we note that the best performance is achieved when including global topology information (the global bias) and comparing node importance (the peer mask) in the TSAM scheme.

4.4 Efficiency Analysis

To validate the efficiency of our mask mechanism, we compare it with full global attention on several real-world networks and record the average runtime per epoch over 50 training epochs. Table 5 presents the detailed results. The results demonstrate that our mask mechanism achieves significant runtime reductions and operational efficiency improvements.

Table 5: Average runtime consumption per epoch (seconds).

Methods	Er	AT	FT	RV	PPI	Fig.	Vid.	Gf
Full Att.	93.06	137.14	32.78	107.51	433.52	424.68	663.71	4.83
Masked Att.	0.82	1.24	0.61	1.95	3.81	3.16	5.13	0.14

4.5 Hyperparameters Analysis

Hyperparameter configurations. Our model incorporates two critical hyperparameters, α_1 and α_2 , representing the minimum and maximum sample sizes respectively. These hyperparameters influence network dismantling performance to some extent, with their optimal values depending on network scale, average degree, and other topological characteristics. For reference, our hyperparameter configurations are provided in Table 6.

Table 6: Configurations of α_1 and α_2 in both real-world and sythetic networks.

Datasets	Chi.	Er	AT	Gnu.	FT	Las.	RV	PPI	Fig.	Vid.	Gf	PLC	BA	ER	WS
α_1	1	2	1	2	1	3	3	5	1	1	1	5	5	3	1
α_2	10	10	10	20	5	15	10	20	10	10	3	10	10	10	1

Sensitivity analysis. To validate the sensitivity of our TSAM to hyperparameters, we configure α_1 and α_2 as four distinct pairs: $(1, \infty)$, $(5, 5)$, $(10, 10)$, and $(20, 20)$. These configurations represent sampling sizes equivalent to node degree, and fixed sizes of 5, 10, and 20 respectively. The comparative results on several real-world networks are presented in Table 7.

Table 7: Dismantling performance with different hyperparameters($\theta = 0.01$).

Configurations	Er	AT	FT	RV	PPI	Fig.	Vid.	Gf
$(1, \infty)$	23.29	23.33	15.45	44.31	22.48	12.59	14.65	11.68
$(5, 5)$	22.23	23.74	14.3	42.54	22.30	9.11	16.69	11.34
$(10, 10)$	25.51	23.49	14.87	41.39	20.82	9.74	17.3	11.68
$(20, 20)$	34.26	22.51	14.87	43.95	22.03	12.95	18.7	11.68
TSAM	22.42	21.7	14.53	41.31	21.04	8.84	14.2	11.34

The analysis reveals that the model performance exhibits measurable variations under different hyperparameter configurations. Nevertheless, our TSAM maintains robust dismantling efficacy across multiple networks, demonstrating competitive results relative to benchmark requirements.

5 Conclusion

This paper studies the potentials of using large models for the classic network dismantling problem. We have proposed the TSAM, a Transformer-based framework with a sparse adaptive mask, and experimented its superiority over the state-of-the-art competitors in real-world networks and synthetic networks. The performance superiority can be attributed to the large model encoding capabilities, but more important is from our design: The degree ranking encoding is to capture local structural biases, the shortest-path encoding is to encode global relations in terms of long-range dependencies, and a sparse mask combining static neighbor mask and adaptive peer mask. The adaptive mask helps to

identify nodes of comparable importance to network integrity, enabling global comparisons while reducing computational overhead.

Despite the new state-of-the-art results, we acknowledge some limitations of our model. The TSAM does not fully exploit higher-order structural patterns (e.g., multi-hop egonet dynamics). While the peer mask reduces computation, scalability to billion-scale networks requires further optimization. Future work will explore hierarchical attention mechanisms for higher-order structures and large-scale networks.

Acknowledgments. This work is supported in part by National Natural Science Foundation of China (Grant No: 62172167).

References

1. Albert, R., Jeong, H., Barabási, A.L.: Error and attack tolerance of complex networks. *nature* **406**(6794), 378–382 (2000)
2. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. *science* **286**(5439), 509–512 (1999)
3. Bavelas, A.: Communication patterns in task-oriented groups. *The journal of the acoustical society of America* **22**(6), 725–730 (1950)
4. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150* (2020)
5. Braunstein, A., Dall’Asta, L., Semerjian, G., Zdeborová, L.: Network dismantling. *Proceedings of the National Academy of Sciences* **113**(44), 12368–12373 (2016)
6. Bu, D., Zhao, Y., Cai, L., Xue, H., Zhu, X., Lu, H., Zhang, J., Sun, S., Ling, L., Zhang, N., et al.: Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic acids research* **31**(9), 2443–2450 (2003)
7. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020)
8. Eash, R., Chon, K., Lee, Y., Boyce, D.: Equilibrium traffic assignment on an aggregated highway network for sketch planning. *Transportation Research* **13**, 243–257 (1979)
9. Erdos, P., Rényi, A., et al.: On the evolution of random graphs. *Publ. math. inst. hung. acad. sci* **5**(1), 17–60 (1960)
10. Ewing, R.M., Chu, P., Elisma, F., Li, H., Taylor, P., Climie, S., McBroom-Cerajewski, L., Robinson, M.D., O’Connor, L., Li, M., et al.: Large-scale mapping of human protein–protein interactions by mass spectrometry. *Molecular systems biology* **3**(1), 89 (2007)
11. Fan, C., Zeng, L., Sun, Y., Liu, Y.Y.: Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence* **2**(6), 317–324 (2020)
12. Freeman, L.: A set of measures of centrality based on betweenness. *Sociometry* (1977)
13. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. pp. 249–256. *JMLR Workshop and Conference Proceedings* (2010)

14. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
15. Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. *Physical review E* **68**(6), 065103 (2003)
16. Guo, G., Zhang, J., Yorke-Smith, N.: A novel evidence-based bayesian similarity measure for recommender systems. *ACM Transactions on the Web (TWEB)* **10**(2), 1–30 (2016)
17. Höglund, M., Frigyesi, A., Mitelman, F.: A gene fusion network in human neoplasia. *Oncogene* **25**(18), 2674–2678 (2006)
18. Holme, P., Kim, B.J.: Growing scale-free networks with tunable clustering. *Physical review E* **65**(2), 026107 (2002)
19. Jazmin, G.: Major internet outage: Dozens of websites and apps were down (2020)
20. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016)
21. Kunegis, J.: Konect: the koblenz network collection. In: Proceedings of the 22nd international conference on world wide web. pp. 1343–1350 (2013)
22. Liu, Q., Wang, B.: Neural extraction of multiscale essential structure for network dismantling. *Neural Networks* **154**, 99–108 (2022)
23. Ma, S., Zeng, W., Xiao, W., Zhao, X.: Dismantling complex networks with graph contrastive learning and multi-hop aggregation. *Information Sciences* p. 120780 (2024)
24. Matei, R., Iamnitchi, A., Foster, P.: Mapping the gnutella network. *IEEE Internet Computing* **6**(1), 50–57 (2002)
25. Morone, F., Makse, H.A.: Influence maximization in complex networks through optimal percolation. *Nature* **524**(7563), 65–68 (2015)
26. Page, L.: The pagerank citation ranking: Bringing order to the web. *Tech. rep., Technical Report* (1999)
27. Park, J., Yun, S., Park, H., Kang, J., Jeong, J., Kim, K.M., Ha, J.W., Kim, H.J.: Deformable graph transformer. *arXiv preprint arXiv:2206.14337* (2022)
28. Radicchi, F.: Percolation in real interdependent networks. *Nature Physics* **11**(7), 597–602 (2015)
29. Ren, X.L., Gleinig, N., Helbing, D., Antulov-Fantulin, N.: Generalized network dismantling. *Proceedings of the national academy of sciences* **116**(14), 6554–6559 (2019)
30. Rozemberczki, B., Sarkar, R.: Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In: Proceedings of the 29th ACM international conference on information & knowledge management. pp. 1325–1334 (2020)
31. Rual, J.F., Venkatesan, K., Hao, T., Hirozane-Kishikawa, T., Dricot, A., Li, N., Berriz, G.F., Gibbons, F.D., Dreze, M., Ayivi-Guedehoussou, N., et al.: Towards a proteome-scale map of the human protein–protein interaction network. *Nature* **437**(7062), 1173–1178 (2005)
32. Šubelj, L., Bajec, M.: Robust network community detection using balanced propagation. *The European Physical Journal B* **81**, 353–362 (2011)
33. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. *stat* **1050**(20), 10–48550 (2017)
34. Wandelt, S., Sun, X., Feng, D., Zanin, M., Havlin, S.: A comparative analysis of approaches to network-dismantling. *Scientific reports* **8**(1), 13513 (2018)

35. Watts, D.J., Strogatz, S.H.: Collective dynamics of ‘small-world’ networks. *nature* **393**(6684), 440–442 (1998)
36. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.Y.: Do transformers really perform badly for graph representation? *Advances in neural information processing systems* **34**, 28877–28888 (2021)
37. Zhang, J., Wang, B.: Dismantling complex networks by a neural model trained from tiny networks. In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. pp. 2559–2568 (2022)
38. Zhang, J., Wang, B.: Encoding node diffusion competence and role significance for network dismantling. In: *Proceedings of the ACM Web Conference 2023*. pp. 111–121 (2023)
39. Zhao, J., Li, C., Wen, Q., Wang, Y., Liu, Y., Sun, H., Xie, X., Ye, Y.: Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094* (2021)