SCIENCE CHINA
Information Sciences

• Supplementary File •

Neural algorithmic approach to network dismantling

Peng Zhang[2], Jun Fu[1,2*], Xiaojie Sun[1], Tonglei Cheng[3] & Guanrong Chen[4]

[1]*State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 100819, China*
[2]*School of Future Technology, Northeastern University, Shenyang 100819, China*
[3]*College of Information Science and Engineering, Northeastern University, Shenyang 100819, China*
[4]*Department of Electrical Engineering, City University of Hong Kong, Hong Kong 999077, China*

Appendix A Notation and Preliminaries

The dimensions of the raw feature space, outcome space, and presentation space are donated by m_X, m_Y, and m_H, respectively. Given that the network dismantling (ND) task is regarded as a node regression task in model learning, the number of training samples n is the number of graph nodes. Let $\boldsymbol{A_i} \in \{0,1\}^{k\times k}$, $\boldsymbol{X_i} \in \mathbb{R}^{k\times m_X}$, and $y_i \in \mathbb{R}$ denote the adjacency matrix, feature matrix and label of the k-hop ego-graph of the i-th sample, respectively. The representations learned by models are denoted as $\boldsymbol{H} \in \mathbb{R}^{n\times m_H}$, where $\boldsymbol{H_{:,i}} \in \mathbb{R}^{n\times 1}$ represents the i-th variable in the representation space. Let $\boldsymbol{w} \in \{\boldsymbol{w} \in \mathbb{R}_+^n | \sum_{i=1}^n w_i = n\}$ denote sample weights, u and v represent Random Fourier Features mapping functions, and $\mathcal{N}(\cdot)$ be neighboring nodes of a given node.

Appendix A.1 Algorithmic Alignment

The concept of Algorithmic Alignment [15] is crucial for constructing effective algorithmic reasoners that extrapolate better. Roughly, it means that a neural network aligns with an algorithm when different parts of the algorithm can be "easily" modeled by one of the modules of the neural network. Recent theoretical results explain why algorithmically inspired models improve extrapolation in algorithmic and physics-based tasks [14].Briefly,if we can design architecture components so that the individual parts have to learn nearly linear ground-truth functions, it implies stronger out-of-distribution performance. Expanding on this, category theory and abstract algebra [32] are used to demonstrate an intricate relationship between GNNs and DP, beyond initial observations on algorithms like Bellman-Ford. This revelation greatly broadens the understanding and potential applications of Algorithmic Alignment.

Guided by this principle, novel GNN architectures and training methods have been proposed to facilitate align with a wider range of combinatorial algorithms [18, 33]. These works support the original theoretical findings [15], which are categorized into Algo-level, Step-level, and Unit-level alignments based on algorithm alignment structures [12]. Among them, Algo-level alignment focuses on learning entire algorithms from inputs to outputs. Given that most network dismantling algorithms are based on heuristics and lack precise reasoning structures, we adopt the Algo-level alignment as the foundational concept for our model design.

Appendix A.2 Stable Learning

The Stable Learning framework aims to bridge the gap between the precise modeling of causal inference and the black-box approaches of machine learning, thereby effectively addressing concerns related to stability, explainability, and fairness [34]. From the causality perspective, this framework is developed from the idea of **Directly Confounder Balancing** (DCB) in the potential outcomes framework, which learns global sample weights to make input variables independent [35]. Following regression analysis on the weighted samples, the regression coefficient of each input variable indicates its causal effect on the outcome. Similarly, from the view of statistical learning, the Stable Learning framework provides a theoretical guarantee that the feature variables utilized in the linear regression model to predict the outcome variable are stable [31]. Thus, this framework can be seen as a feature selection method that relies on regression coefficients to differentiate between stable and unstable features in the representation space.

The core of the framework lies in adjusting the weights of samples to eliminate the independence among features. Although various techniques such as Hilbert-Schmidt Independence Criterion (HSIC) and Mutual Information (MI) can be used for independence test, they can hardly be used to optimize deep models at an acceptable cost. Considering the complex correlations among features obtained by deep learning, a novel nonlinear feature decorrelation method based on Random Fourier Features with linear computational complexity is proposed [23]. Equation (A1) measures the independence between any pair of features $\boldsymbol{H_{:,i}}$ and $\boldsymbol{H_{:,j}}$ in the representation space. To simplify the notation, we use A and B represent

* Corresponding author (email: junfu@mail.neu.edu.cn)

one-dimensional random variables instead of $\boldsymbol{H}_{:,i}$ and $\boldsymbol{H}_{:,j}$ and we sample $(A_1, A_2, ..., A_n)$ and $(B_1, B_2, ..., B_n)$ for the distribution of A and B.

$$\hat{\Sigma}_{AB;\boldsymbol{w}} = \frac{1}{n-1}\sum_{i=1}^{n}\left[\left(w_i\boldsymbol{u}\left(A_i\right) - \frac{1}{n}\sum_{j=1}^{n} w_j\boldsymbol{u}\left(A_j\right)\right)^T \left(w_i\boldsymbol{v}\left(B_i\right) - \frac{1}{n}\sum_{j=1}^{n} w_j\boldsymbol{v}\left(B_j\right)\right)\right], \tag{A1}$$

where

$$\begin{aligned}\boldsymbol{u}(A) &= \left(u_1(A), u_2(A), \ldots u_{n_A}(A)\right), u_j(A) \in \mathcal{H}_{\mathrm{RFF}}, \forall j,\\ \boldsymbol{v}(B) &= \left(v_1(B), v_2(B), \ldots v_{n_B}(B)\right), v_j(B) \in \mathcal{H}_{\mathrm{RFF}}, \forall j.\end{aligned} \tag{A2}$$

Here, we sample n_A and n_B functions from $\mathcal{H}_{\mathrm{RFF}}$, where $\mathcal{H}_{\mathrm{RFF}}$ represents the function space of Random Fourier Features characterized by the following form:

$$\begin{aligned}\mathcal{H}_{\mathrm{RFF}} =& \{h : x \to \sqrt{2}\cos(\omega x + \phi) \mid \\ & \omega \sim N(0,1), \phi \sim \text{ Uniform } (0, 2\pi)\},\end{aligned} \tag{A3}$$

where ω is sampled from the standard normal distribution and ϕ is sampled from the uniform distribution.

Appendix B Stable-AGDM Model

Our model takes one network as input, without any node feature information, and produces a scalar value p_n ranging from zero to one for each node n. When dismantling a network, nodes are sorted and removed in descending order of p_n until the target is reached. Then, because the value p_n indicates the likelihood of the node being part of an optimal set, Stable-AGDM employs a node reinsertion strategy [25] to refine the dismantling set after dismantling. And the Stable-AGDM approach is applicable to undirected, unweighted simple graphs. Currently, it cannot be used for the disintegration of weighted networks because the graph neural networks employed do not consider edge weights. However, in the future, we can incorporate graph neural networks that account for edge weights into this framework, enabling its use for weighted network disintegration.

Appendix B.1 Model architecture

Neural Feature Engineering

• **Local Structure Learning (LSL)** This module utilizes the network adjacency matrix to generate local structure vectors for each node. Existing methodologies are dominated by the utilization of feature engineering techniques. To avoid learning failures, feature engineering typically requires features to be heterogeneous. The requirement for heterogeneity arises from the necessity to differentiate nodes situated at various topological positions through the implementation of constructed eigenvectors. Meanwhile, GNNs based on graph isomorphism [19,20] can distinguish more topological structures by constructing structure vectors based on the adjacency matrix of the network, such as GIN. Therefore, we adopt GNNs based on graph isomorphism to learn node structure information and construct structure vectors with strong topological representation ability. To maintain generality, we assume that the raw feature vector of each node is a constant value of one. We use Residual Gated Graph ConvNets [24] (ResGatedG) to learn local structure vectors $\boldsymbol{h_{local}} \in \mathbb{R}^{n\times p}$:

$$\boldsymbol{h'_{local_i}} = \boldsymbol{h_{local_i}}\boldsymbol{W_1} + \sum_{j\in\mathcal{N}(i)} \eta_{i,j} \odot \boldsymbol{h_{local_j}}\boldsymbol{W_2}, \tag{B1}$$

where $\boldsymbol{h'_{local_i}} \in \mathbb{R}^{1\times p}$ denotes the updated local structure vector of the target node i, $\boldsymbol{h_{local_i}} \in \mathbb{R}^{1\times p}$ denotes the local structure vector before the update, $\mathcal{N}(i)$ is the one-hop neighbors of node i and $\eta_{i,j} = \sigma(\boldsymbol{h_{local_i}}\boldsymbol{W_3} + \boldsymbol{h_{local_j}}\boldsymbol{W_4})$ with σ denoting the sigmoid function, and $\boldsymbol{W_1} \in \mathbb{R}^{p\times p}$, $\boldsymbol{W_2} \in \mathbb{R}^{p\times p}$, $\boldsymbol{W_3} \in \mathbb{R}^{p\times 1}$, $\boldsymbol{W_4} \in \mathbb{R}^{p\times 1}$ are learnable parameters. But it should be emphasized that any graph convolutional layer that enhances topological representation ability can be used for this module.

• **Global Structure Learning (GSL)** The module generates global structure vectors for each node, which combines local structure vectors with the graph structure that it has learned. The objective of this module is to assess the global impact of each node. As the dismantling is for the global network, it is imperative to differentiate the importance of two nodes, even if they exhibit similar local structures. To address this issue, we utilize graph pooling to obtain graph feature vector $\boldsymbol{h_{graph}} \in \mathbb{R}^{1\times p}$ containing global structure information and merge it with the node local structure vectors $\boldsymbol{h_{local}} \in \mathbb{R}^{n\times p}$ so as to generate global structure vectors $\boldsymbol{h_{global}} \in \mathbb{R}^{n\times p}$ in the structure presentation space:

$$\boldsymbol{h_{global_i}} = \boldsymbol{W_5}\,\mathrm{ReLU}\left(\boldsymbol{h_{local;i}^{\top}} \cdot \boldsymbol{h_{graph}} \cdot \boldsymbol{W_6}\right), \tag{B2}$$

where $\boldsymbol{h_{global_i}} \in \mathbb{R}^{1\times p}$ denotes the updated global structure vector of the target node i, $\boldsymbol{h_{local_i}} \in \mathbb{R}^{1\times p}$ denotes the local structure vector of the target node i, $\boldsymbol{W_5} \in \mathbb{R}^{1\times p}$ and $\boldsymbol{W_6} \in \mathbb{R}^{p\times p}$ are learnable parameters.

Topological Correlation Learning (TCL)

The module generates representation vectors for each node by discovering the correlation between node global structure vectors. The selection of critical nodes in the ND problem is significantly influenced by the higher-order interactions between nodes. In the preceding modules, the model has learnt the node topology information through GNNs. The focus of this module is to learn the impact of topological interactions on key nodes, namely the correlation between node features. So, GNNs based on graph signal processing are selected to learn the correlation between node features. As shown in Equation (B3), GAT [22] has an additional adaptive edge weight coefficient compared to GCN. This mechanism is analogous, in terms of graph signal processing, to learning a self-adaptive graph shift operator, which corresponds to a self-adaptive filtering effect. So, GAT can focus more on learning the correlation between node features to generate the node representation vectors $\boldsymbol{H} \in \mathbb{R}^{n \times m_H}$ in the representation space:

$$\boldsymbol{H_i'} = \alpha_{i,i}\boldsymbol{H_i}\boldsymbol{W_7} + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\boldsymbol{H_j}\boldsymbol{W_7}, \tag{B3}$$

where $\boldsymbol{H_i} \in \mathbb{R}^{1 \times m_H}$ denotes the updated presentation vector of the target node i, $\boldsymbol{W_7} \in \mathbb{R}^{m_H \times m_H}$ is the learnable parameter and the attention coefficients $\alpha_{i,j}$ is defined as

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\boldsymbol{H_{i,j}}\boldsymbol{W_8}))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\text{LeakyReLU}(\boldsymbol{H_{i,j}}\boldsymbol{W_8}))}, \tag{B4}$$

where $\boldsymbol{H_{i,j}} = [\boldsymbol{H_i}\boldsymbol{W_7} || \boldsymbol{H_i}\boldsymbol{W_7}] \in \mathbb{R}^{1 \times 2m_H}$ and $\boldsymbol{W_8} \in \mathbb{R}^{2m_H \times 1}$ is the learnable parameter.

Feature Seletion (FS)

The module employs the Stable Learning framework to identify stable features within the representation space. The existing node representation vectors comprise local topological information, global information, and topological correlation information of the nodes as a result of the learning processes of the three modules mentioned above. The integration of these information aligns with the empirical cognition of identifying key nodes in current ND problem research. However, the highly flexible structure of the model allows it to learn a wider range of graph information, so there may be an abundance of redundant and irrelevant features in the node representation vectors.

To effectively identify features that have a practical impact on network dismantling, we design a feature selection module to eliminate irrelevant and spurious features. The Stable Learning framework can be statistically understood as a feature selection mechanism based on regression coefficients, which can distinguish stable features from non-stable features in the representation space. Therefore, we adopt a feature selection module based on the Stable Learning framework, which select stable feature variables through a weighted loss function to increase the robustness of the model. For more information, please refer to the "Training algorithm"Appendix B.2 section.

Appendix B.2 Training algorithm

Algorithm B1 Training procedure of Stable-AGDM

Require: Training dataset $\boldsymbol{G} = \{(\boldsymbol{A_1}, \boldsymbol{X_1}, y_1), ..., (\boldsymbol{A_N}, \boldsymbol{X_N}, y_N)\}$, Sample size n, Training epochs T, Decorrelation epochs D.

```
for t = 1 to T do
  for 1 to BatchNumber do
    According to the algorithmic framework in the main text (i.e., the Eq. B1, B2, B3 in Appendix B), forward propagation to generate H;
    for 1 to D do
      Optimize sample weights w via Eq. B6;
    end for
    Update f,γ and g via Eq. B5;
  end for
end for
```

Appendix B.2.1 *Training dataset*

We perform supervised training of our model using synthetic networks (25 nodes each) generated by the Barabasi-Albert (BA), the Erdos-Renyi (ER) and the Static Power Law (PL) models respectively. A brute-force search (i.e., attempting every possible combination of nodes) is conducted on each synthetic network to find all the minimum-size solutions that reduce the Largest Connected Component to a specified target size, which in our work is 18%. It is important to note that there may be several optimal dismantling sets for each synthetic network. Hence, the label of each node is computed as the ratio of the number of optimal sets that include the node to the total number of optimal sets. This ratio reflects the probability of the node being part of an optimal set. When there is only one optimal set, the nodes in this set are labeled

1, while the rest are labeled 0. When there are two optimal sets, the nodes that belong to both sets are labeled 1, the nodes that belong to only one set are labeled 0.5, and the nodes that belong to neither set are labeled 0. This approach assigns higher labels to nodes that are more relevant for the optimal dismantling of the network. This is meant to teach the model that some nodes are more essential than others for the optimal dismantling of the network, as they are included in multiple optimal sets.

The imbalanced training data arises from our label generation method, consistent with GDM and other supervised learning-based algorithms: each node's label equals its frequency in optimal dismantling sets divided by the total number of such sets. Since optimal sets contain far fewer nodes than the entire network, most nodes receive zero-valued labels, while only a small fraction have non-zero values.

Appendix B.2.2 *Loss function*

As previously mentioned, the distribution of training samples for the ND task exhibits a significant imbalance, which may lead to biased model performance and hinder its generalization ability. To enhance the model's comprehensive understanding of diverse data information, we choose to employ a widely recognized approach known as sample reweighting for addressing uneven data acquisition. The Stable Learning framework represents an adaptive sample reweighting process, which is highly suitable for supervised learning tasks with abundant samples but imbalanced distributions. Moreover, as discussed in the Feature Seletion module, the Stable Learning framework facilitates the learning of stable features that are relevant to the task. The integration of the Stable Learning framework can enhance the predictive performance of the model, thereby facilitating more comprehensive scientific research. Moreover, to guarantee the efficacy of the node structural information, we integrate graph reconstruction loss to maintain the original network structures within the structure presentation space, as shown in the second term of Equation (B5).

Therefore, Stable-AGDM iteratively optimize sample weights $\boldsymbol{w}$, the neural engineering module f, the topological correlation learning mudule γ and the feature selection module g. The training algorithm implemented is shown in Algorithm B1,with

$$\begin{aligned} & f^{(t+1)}, \gamma^{(t+1)}, g^{(t+1)} \\ & = \arg\min_{f,\gamma,g} \sum_{i=1}^{n} w_i^{(t)} \left(g\left(\gamma\left(f\left(\boldsymbol{A_i}, \boldsymbol{X_i}\right)\right)\right) - y_i\right)^2 \\ & + \alpha \sum_{i,j=1}^{n} a_{i,j} \left\| \boldsymbol{h}_{\boldsymbol{global_i}}^{(\boldsymbol{t})} - \boldsymbol{h}_{\boldsymbol{global_j}}^{(\boldsymbol{t})} \right\|_2^2, \end{aligned} \tag{B5}$$

$$\boldsymbol{w}^{(t+1)} = \arg\min_{\boldsymbol{w} \in \Delta_n} \sum_{1 \leqslant i < j \leqslant m_H} \left\| \hat{\Sigma}_{\boldsymbol{H}_{:,i}^{(t+1)} \boldsymbol{H}_{:,j}^{(t+1)}; \boldsymbol{w}} \right\|_F^2, \tag{B6}$$

where α is a positive hyper-parameter weighing between these two losses, $a_{i,j}$ denotes whether nodes i and j are adjacent, and $\Delta_n = \left\{ \boldsymbol{w} \in \mathbb{R}_+^n \mid \sum_{i=1}^n w_i = n \right\}$.

Appendix B.3 Dismantling Strategy

The existing ND algorithms based on supervised learning regard the prediction values of the model as indicators of node dismantling importance.The top k nodes are removed based on their prediction values until the network dismantling threshold is reached. However, in accordance with the meaning of training sample labels, the node prediction values generated by deep learning models should closely approximate the probabilities of nodes belonging to the optimal solution set. For example, as shown in Supplementary Fig. B1, when the threshold is set to the maximum connected subgraph size of 1, there are two optimal solution sets: 1,3 and 2,4, both with a supervised learning label of 1/2. When the model prediction performance is good, the dismantling order based on model prediction values may be: 1, 2, 3. The direct dismantling of the network solely based on node prediction values may not yield satisfactory results and may deviate from the intended interpretation of model prediction output. Therefore, based on the above considerations, we adopt the following dismantling strategy : After iteratively removing the top k nodes based on their prediction values until reaching the specified threshold, we implement the node reinsertion strategy used in MinSum, which ensures the maximum connected subgraph size after reinserting becomes smaller than the dismantling threshold range.

Appendix C More details of experiments

We train our model using the model configuration and training hyperparameters as shown in Table C1. All experiments are conducted on a shared machine equipped with an Intel (R) Core (TM) i7-9700 CPU, 16GB RAM, and an NVIDIA Quadro P620.

Appendix C.1 Baseline Methods

We compare Stable-AGDM with other state-of-the-art methods that focus on network dismantling. To demonstrate that the dismantling performance of our algorithm is not dependent on the node reinsertion strategy, we incorporate node reinsertion into these benchmark algorithms (except HDA and CI) and add the random_rein algorithm for comparison.

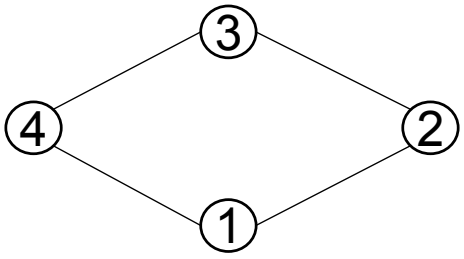

Figure B1 An example of network dismantling.

Table C1 Stabel-AGDM model configuration and hyperparameters

Hyperparameter		Value
Learning Rate		0.01
Weight Decay		1×10^{-5}
Training Epochs		800
Decorrelation Epochs		20
Sample Size		200
Fully Connected Layers		2
RFF Dimension		2
Learning Rate (Stable Learning)		10
Graph Reconstruction Loss Cofficient		0.01
GNN Type	LSL	ResGatedGraph
	TCL	GAT
GNN Layers	LSL	4
	TCL	2
Embedding Dimension	LSL	40
	TCL	40

• FINDER [6]. 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 takes action a that represents the maximum expected rewards given state s.

• CI [2]. CI value is defined as the product of the node reduced degree (original degree minus one) and total reduced degree of neighboring nodes at the surface of a ball of constant radius. The CI method sequentially removes the node with the highest CI value and recalculates the CI values after each node removal.

• MinSum [1]. MinSum, a classical optimal decycling based method, is composed of three stages: (i) use MinSum message passing algorithm to break all the loops in the network; (ii) iteratively remove the root node until the size of the tree component is not larger than the desired threshold C; (iii) reinsert some nodes to close cycles without increasing the size of the largest component.

• GND [3]. In each step, GND uses the spectral cut method to partition the remaining network into two non-overlapping components and removes nodes that cover all of the edges between two components at minimal cost.

• GDM [7]. GDM is the first supervised learning model, which is composed of feature engineering, GNN layers and a regressor. Its dismantling strategy is the same as Stable-AGDM.

• NIRM [10]. NIRM, a supervised learning method, consists of six modules: (1) feature engineering, (2) feature scoring, (3) representation encoding, (4) local scoring, (5) global scoring, and (6) fusion and rank. Its dismantling strategy is the same as Stable-AGDM.

• random_rein. It firstly randomly removes nodes until the biggest size of the remaining components is not larger than the desired threshold C, and then applies the node reinsertion strategy.

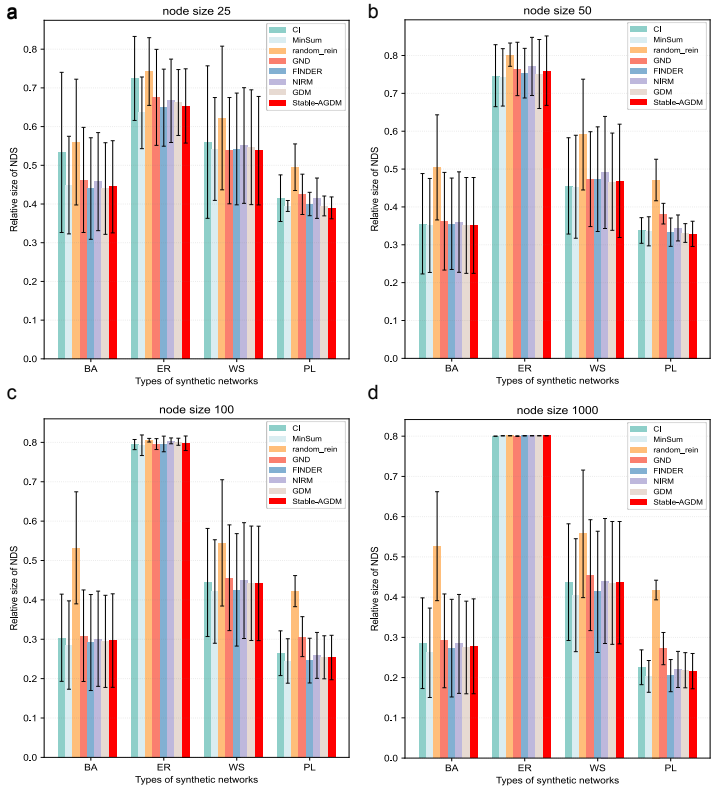

Figure S1 Performance of Stable-AGDM on different types of synthetic networks. To quantify the goodness of each algorithm in the ND problem, the *relative size of NDS* (denoted by $\rho = |\mathcal{N}_{NDS}|/|\mathcal{N}|$), i.e., the fraction of nodes' removal that leads to the 20% residual LCC size, is used as a measure. (a-d) comparison of Stable-AGDM and other baseline algorithms on four types of model networks, i.e., Barabási-Albert scale-free networks (BA), Erdös-Rényi random networks (ER), Watts-Strogatz small-world networks (WS), and static Power Law networks (PL), with network size N =25, 50, 100, 1000.

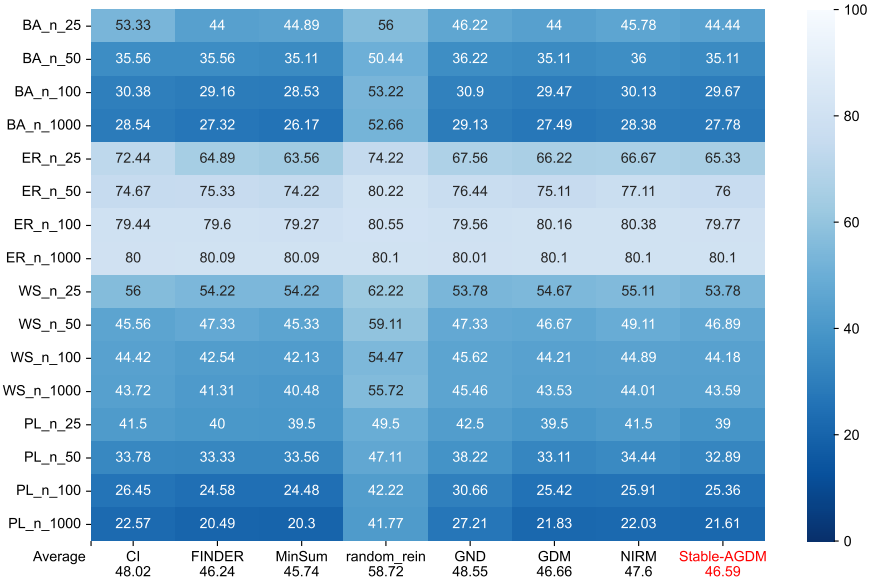

Figure S2 Performance of Stable-AGDM on different types of synthetic networks. To quantify the goodness of each algorithm in the ND problem, the *relative size of NDS* (denoted by $\rho = |\mathcal{N}_{NDS}|/|\mathcal{N}|$), i.e., the fraction of nodes' removal that leads to the 20% residual LCC size, is used as a measure. comparison of Stable-AGDM and other baseline algorithms on four types of model networks, i.e., Barabási-Albert scale-free networks (BA), Erdös-Rényi random networks (ER), Watts-Strogatz small-world networks (WS), and static Power Law networks (PL), with network size N =25, 50, 100, 1000.

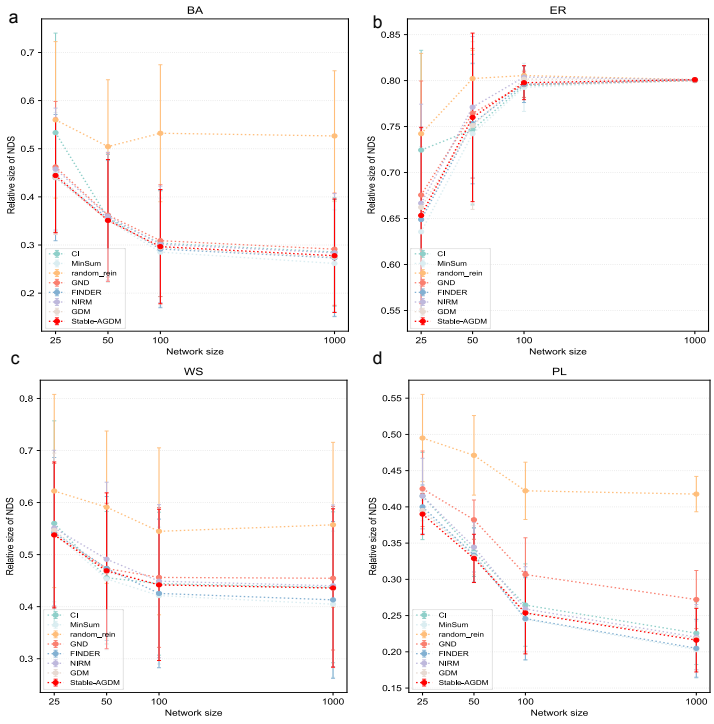

Figure S3 Performance trend of the algorithm as a function of scale on each synthetic network. (a-d) represent different kinds of synthetic network models, i.e., Barabási-Albert scale-free networks (BA), Erdös-Rényi random networks (ER), Watts-Strogatz small-world networks (WS), and static Power Law networks (PL).

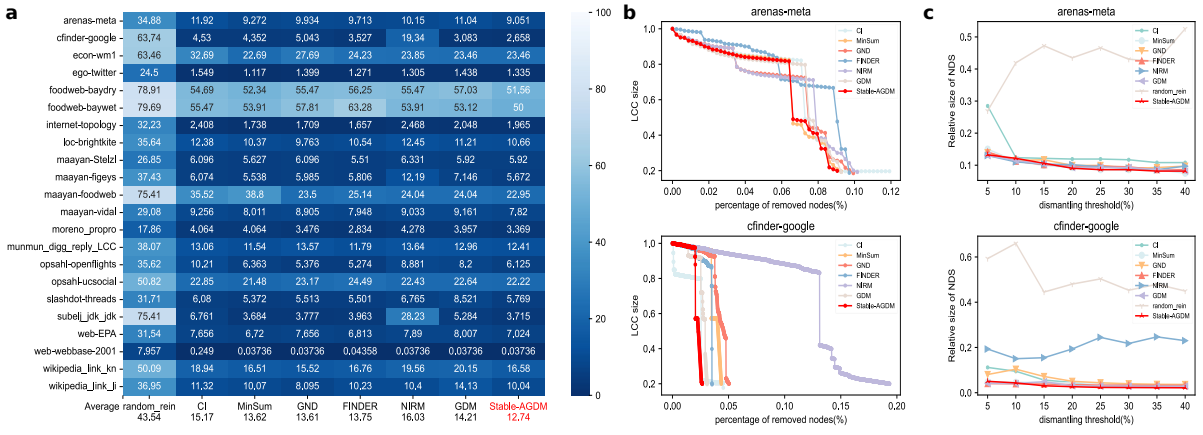

Figure S4 Performance of Stable-AGDM on real-world networks. (a) To quantify the goodness of each algorithm in ND problem, we consider the *relative size of NDS* (denoted by $\rho = |\mathcal{N}_{NDS}|/|\mathcal{N}|$), i.e., the fraction of nodes' removal that leads to the 20% residual LCC size, is used as a measure. The smaller the relative size of NDS ρ, the darker the color. (b) The Area Under the Curve (AUC) represents changes in the size of the Largest Connected Component (LCC) during the attacks. A lower area under the curve indicates more efficient network teardown. (c) comparison of the relative sizes of NDS at different dismantling thresholds.

Appendix C.2 Dataset

Appendix C.2.1 *Synthetic Networks.*

Our synthetic data are composed of Barabási-Albert scale-free networks, Erdös-Rényi random networks, Watts-Strogatz small-world networks and static Power Law networks, which are implemented in igraph [36] and Networkx [37]. 10 instances are generated with 25, 50, 100 and 1K nodes each and the results are averaged.

All networks were generated with the following configurations:

- Barabasi-Albert (BA) scale-free networks The number of outgoing edges per vertex (m) ranges from 2 to 10.
- Erdos-Renyi (ER) networks The average degree ranges from 4 to 20.
- Static Power-law generational models The power-law exponent (γ) ranges from 2.1 to 2.9, and the average degree ranges from 4 to 20.

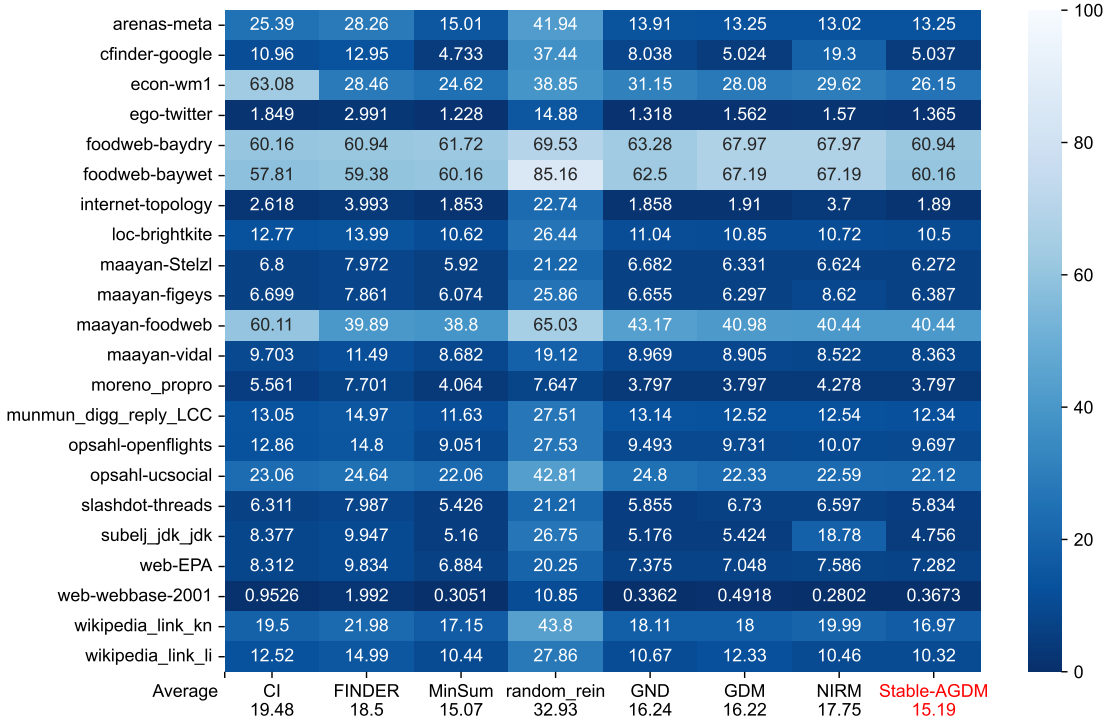

Figure S5 Performance of Stable-AGDM on real-world networks. To further validate the model's effectiveness, we test a smaller target (5% residual LCC) on real networks.

• Watts-Strogatz (WS) small-world networks The average degree ranges from 4 to 20, and the rewiring probability (p) ranges from 0.01 to 0.15.

Appendix C.2.2 *Real-world Networks.*

We choose 22 real-world networks, as shown in Table C2. We consider all the networks as undirected and unweighted.

Appendix C.3 Experiments results and analyses

In our experiments, we compare Stable-AGDM with other state-of-the-art methods, such as CI [2], MinSum [1], GND [3], FINDER [6], GDM [7], NIRM [10]. As mentioned earlier, Stable-AGDM performs a node reinsertion strategy during dismantling. To demonstrate that the dismantling performance of our algorithm is not dependent on the node reinsertion strategy, we incorporate the node reinsertion strategy into the above-mentioned benchmark algorithms (except FINDER and CI) and add random_rein algorithm, which randomly removes nodes until the target is reached and then applies the node reinsertion strategy.

In the ND problem, the size of *largest connected component* (LCC) is frequently used as a measure for quantifying normal system functions, and when the size of LCC decreases below a predefined threshold C, it is regarded as a collapse in system functionality [26–28]. Thus, a widely accepted and rigorous definition of the ND problem is to determine the minimum *network dismantling node set* (NDS), whose removal is able to dismantle the network into isolated sub-components of a specific small size [1], defined by.

$$\mathcal{N}^*_{dns} \equiv \{\mathcal{N}_{dns} \subseteq \mathcal{N} : |\mathcal{G}_{lcc}(\mathcal{N}_{dns})|/|\mathcal{N}| \leqslant C\} \tag{C1}$$

where $|\mathcal{G}_{lcc}(\mathcal{N}_{dns})|$ denotes the number of nodes in LCC $\mathcal{G}_{lcc}$ after the removal of NDS $\mathcal{N}_{dns}$ and $|\mathcal{N}|$ is the number of nodes in $\mathcal{G}$. To quantify the goodness of each algorithm in the ND problem, we consider the *relative size of NDS* (denoted by ρ, and $\rho = |\mathcal{N}_{dns}|/|\mathcal{N}|$), e.g., the fraction of nodes' removal that leads to a 20% residual LCC size in this paper. The *relative size of NDS* is adopted because we are more concerned with the situation where the network dismantles to the critical point of collapse rather than the process itself. Unless otherwise specified, the performance evaluation metric used in our experiments is the *relative size of NDS*.

Additionally, we chose a 20% residual LCC size as the evaluation criterion because it aligns with the method used to generate node labels for supervised learning. We emphasize that this choice of training labels is arbitrary, and other labels might be more effective for different datasets or objectives. To ensure a fair comparison with models like GDM, we used the same training dataset and label generation approach as GDM. Specifically, we identified all minimal solutions through brute-force that reduce the Largest Connected Component (LCC) to approximately 18% of its original size. The label for each node was then calculated as the proportion of optimal sets it is part of relative to the total number of optimal solutions. Since the model essentially learns the optimal dismantling sets that reduce the LCC to around 18%, we selected a similar

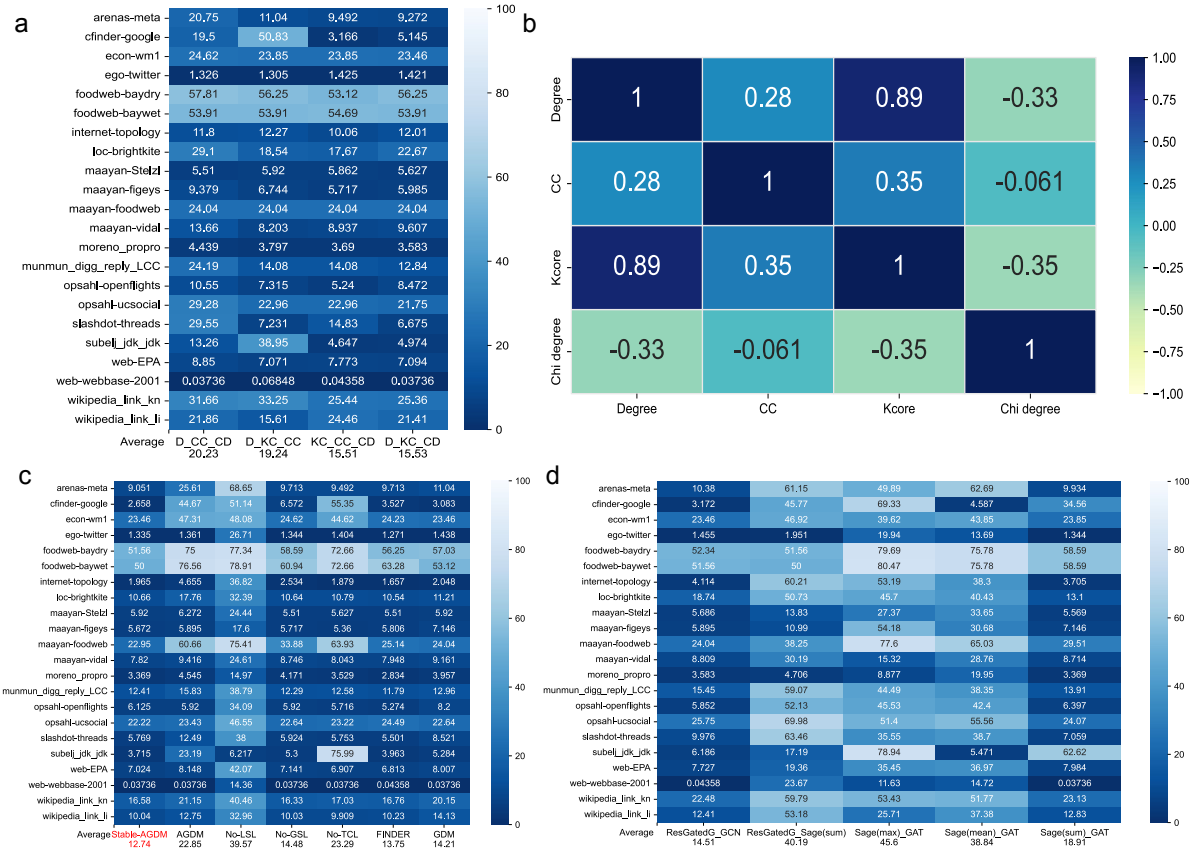

Figure S6 The relationship between model structure and reasoning ability. (a) The dismantling performance of models with different combinations of features: Degree (D), Clustering Coefficient (CC), K-Core (KC), χ^2 of degree (CD); (b) The average correlation matrix for the four features in manual feature engineering over 22 real-world networks. Degree, χ^2 of degree (Chi degree), Clustering Coefficient (CC), and K-Core (Kcore) are the four features given here. Details of the 22 networks are shown in Supplementary Table C2 in Supplementary Information. Each element is the averaged value of the Kendall correlation coefficient τ between the two indices corresponding to its position over the 22 networks, and the value is visualized by the color. (c) The AGDM algorithm represents the model without the Stable Learning framework, No-LSL represents the model without Local Structure Learning, No-GSL indicates the model without Global Structure Learning, and No-TCL denotes the model without Topological Correlation Learning; (d) Ranking of GNN's ability to learn graph structure information [19, 20, 29]: ResGatedG is the most effective, followed by Sage(sum), Sage(mean), and Sage(max) being the least;Ranking of GNN's ability to topological correlation information [20–22]: GAT is the most effective, followed by GCN, and Sage(sum) being the least.

target, 20% residual LCC, for performance evaluation based on the influence of the supervised labels. While alternative thresholds (e.g., $\sqrt{N}$) may suit specific applications, our approach prioritizes a consistent evaluation framework aligned with existing methods.

Fig. S1 and S2 show that the performance of Stable-AGDM is always competitive across network scales and types, especially outperforming other supervised learning algorithms. The error bars indicate that the robustness of node removal from Stable-AGDM is similar to other baseline algorithms, which suggests that the algorithm's dismantling ability is relatively reliable rather than claiming dataset-specific superiority. Also, as shown in Supplementary Fig. S3, the trend of dismantling efficiency changing for each algorithm in BA and WS networks differs from that of ER networks as network scale changes. This could be explained by the fact that dismantling algorithms perform better on network models with scale-free or small-world properties. This becomes more prominent for BA and WS networks as network size increases, while ER networks become more disordered. This indirectly suggests that the Stable-AGDM algorithm learns network structural properties and uses them to improve performance.

Fig. S4 shows the performance of each dismantling method on every real-world network considered in this study, allowing for an overall comparison. As shown in Fig. S4 **a**, Stable-AGDM has the best performance on 11 networks and also achieves top three performance on the rest networks. On average, Stable-AGDM outperforms the other algorithms. Specifically, it only needs to remove 12.74% nodes to dismantle a network on average. This remains noteworthy since the proposed Stable-AGDM is a one-pass method without manual feature engineering while the others need recompute the nodes' structural importance or consider manual features during the dismantling process, validating our approach. To further validate the model's effectiveness, we test a smaller target (5% residual LCC) on real networks. As shown in Fig. S5, Stable-AGDM maintains competitive dismantling performance even at this stricter threshold, especially outperforming other supervised learning algorithms.

To conduct a thorough performance analysis, we utilize the arenas-meta network and cfinder-google network as repre-

Table C2 Statistical properties of real-world networks.

Network	Category	Nodes Number	Edges Number	Modularity
arenas-meta [38]	Social	6.5K	43.3K	0.417
cfinder-google [38]	Hyperlink	15.8K	149.5K	0.491
econ-wm1 [38]	Economic	260	2.6K	0.2623
ego-twitter [38]	Social	23.4K	32.8K	0.8771
foodweb-baydry [38]	Trophic	128	2.1K	0.1459
foodweb-baywet [38]	Trophic	128	2.1K	0.1472
Internet-topology [38]	Infrastructure	34.8K	107.7K	0.5646
loc-brightkite [38]	Social	58.2K	214.1K	0.6166
maayan-figeys [38]	Metabolic	2.2K	6.4K	0.472
maayan-foodweb [38]	Trophic	183	2.5K	0.3054
maayan-Stelzl [38]	Metabolic	1.7K	3.2K	0.6149
maayan-vidal [38]	Metabolic	3.1K	6.7K	0.6531
moreno_propro [38]	Metabolic	1.9K	2.3K	0.8492
munmun_digg_rely_LCC [38]	Communication	29.7K	84.8K	0.4069
opsahl-openflights [38]	Infrastructure	2.9K	15.7K	0.6065
opsahl-ucsocial [38]	Communication	1.9K	13.8K	0.253
slashdot-threads [38]	Communication	51.1K	117.4K	0.4607
subelj_jdk_jdk [38]	Software	6.4K	53.7K	0.4317
web-EPA [39]	Hyperlink	4.3K	8.9K	0.6219
web-webbase-2001 [39]	Hyperlink	16.1K	25.6K	0.9248
wikipedia_link_kn [38]	Hyperlink	29.5K	278.7K	0.5881
wikipedia_link_li [38]	Hyperlink	49.1K	294.3K	0.5997

sentative examples. We initially examine the Area Under the Curve (AUC) encoding changes in the size of the Largest Connected Component (LCC) throughout the attacks. A lower area under the curve indicates a more effective network dismantling. Fig. S4 **b** shows that Stable-AGDM dismantles the network faster than the other algorithms. To further verify whether the proposed algorithm has good dismantling performance at different dismantling thresholds, Fig. S4 **c** shows that Stable-AGDM maintains competitive performance at different thresholds.

To explicitly understand the impact of each module on the performance of Stable-AGDM, we have performed a module ablation experiment. As shown in Fig. S6**c**, these modules help improve the performance of our algorithm. Among the four modules, the Local Structure Learning module contributes the most to the overall performance, followed by the Stable Learning framework and the Topological Correlation Learning module, while the Global Structure Learning module contributes the least. Although the No-GSL model already has competitive performance, learning the global graph structure information is more conducive to improving the network dismantling performance. This aligns with our design goal of employing global projection to consider the global influence of each node. More importantly, the Local Structure Learning module greatly affects the algorithm's performance, verifying the analysis for the ND task.

It is noted that processing graph structure and topological correlation information have distinct requirements for GNNs [20]. GNNs based on graph isomorphism (e.g., GIN, k-GNN) are good at processing graph structure information, whereas those based on graph signal processing (e.g., GCN, GAT) are better at processing topological correlation information. To verify this analysis, in Fig. S6 **d**, we analyze the performance of the model using GNNs with varying learning abilities in these two modules. We observe that the performance of the model using different GNNs shows notable differences, verifying that graph structure learning and topological correlation learning are essential to improving the performance of the model. It is stressed that the two GNNs employed in this study are replaceable, provided that the GNN structures to be chosen can enhance the learning ability of these two modules.

References

1 Braunstein A, Dall'Asta L, Semerjian G, et al. Network dismantling. Proc. Natl. Acad. Sci. U.S.A, 2016, 113: 12368-12373
2 Morone F and Makse HA. Influence maximization in complex networks through optimal percolation. Nature, 2015, 524: 65–68
3 Ren XL, Gleinig N, Helbing D, et al. Generalized network dismantling. Proc. Natl. Acad. Sci. U.S.A, 2019, 116: 6554-6559
4 Clusella P, Grassberger P, Pérez-Reche FJ, et al. Immunization and targeted destruction of networks using explosive percolation. Phys. Rev. Lett, 2016, 117: 208301
5 Kitsak M, Gallos LK, Havlin S, et al. Identification of influential spreaders in complex networks. Nat. Phys, 2010, 6: 888-893
6 Fan C, Zeng L, Sun Y, et al. Finding key players in complex networks through deep reinforcement learning. Nat. Mach. Intell, 2020, 2: 317-324
7 Grassia M, De Domenico M and Mangioni G. Machine learning dismantling and early-warning signals of disintegration in complex systems. Nat. Commun, 2021, 12: 5190
8 Liu Q and Wang B. Neural extraction of multiscale essential structure for network dismantling. Neural Netw, 2022, 154: 99-108

9 Tian M, Dong Z and Wang X. Reinforcement learning approach for robustness analysis of complex networks with incomplete information. Chaos Solitons Fract, 2021, 144: 110643
10 Zhang J and Wang B. Dismantling complex networks by a neural model trained from tiny networks. In: Proceedings of ACM International Conference on Information & Knowledge Management (CIKM), Atlanta GA, 2022. 2559-2568
11 Bengio Y, Lodi A and Prouvost A. Machine learning for combinatorial optimization: a methodological tour d'Horizon. Eur. J. Oper. Res, 2021, 290: 405-421
12 Cappart Q, Chételat D, Khalil E, et al. Combinatorial optimization and reasoning with graph Neural Networks. J. Mach. Learn. Res, 2023, 1-61
13 Garmendia AI, Ceberio J and Mendiburu A. Neural combinatorial optimization: a new player in the field. ArXiv:2205.01356
14 Xu K, Zhang M, Li J, et al. How neural networks extrapolate: from feedforward to graph neural networks. In: Proceedings of International Conference on Learning Representations (ICLR), Vienna, 2021.
15 Xu K, Li J, Zhang M, et al. What can neural networks reason about? In: Proceedings of International Conference on Learning Representations (ICLR), Addis Ababa, 2020.
16 Numeroso D, Bacciu D and Veličković P. Dual algorithmic reasoning. In: Proceedings of International Conference on Learning Representations (ICLR), 2023.
17 Ibarz B, Kurin V, Papamakarios G, et al. A Generalist Neural Algorithmic Learner. In: Proceedings of the First Learning on Graphs Conference (LoG), 2022. 2-1
18 Tang H, Huang Z, Gu J, et al. Towards scale-invariant graph-related problem solving by iterative homogeneous GNNs. In: Proceedings of International Conference on Neural Information Processing Systems (NeurIPS), 2020. 15811-15822
19 Xu K, Hu W, Leskovec J, et al. How powerful are graph neural networks? In: Proceedings of International Conference on Learning Representations (ICLR), 2019.
20 Hamilton WL. Graph representation learning. Morgan & Claypool Publishers, 2020. 60-95
21 Kipf TN and Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference on Learning Representations (ICLR), 2017.
22 Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of International Conference on Learning Representations, ICLR, 2018.
23 Zhang X, Cui P, Xu R *et al.* Deep stable learning for out-of-distribution generalization. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2021. 5372-5382
24 Bresson X and Laurent T. Residual gated graph convnets. ArXiv:1711.07553
25 Qin SM, Ren XL and Lü LY. Efficient network dismantling via node explosive percolation. Commun. Theor. Phys, 2019, 71: 764
26 Bellingeri M, Cassi D and Vincenzi S. Efficiency of attack strategies on complex model and real-world networks. Sci. A, 2014, 414: 174-180
27 Albert R and Barabási AL. Statistical mechanics of complex networks. Rev. Mod. Phys, 2002, 74: 47-97
28 Callaway DS, Newman MEJ, Strogatz SH, et al. Network robustness and fragility: percolation on random graphs. Phys. Rev. Lett, 2000, 85: 5468-5471
29 Dwivedi VP, Joshi CK, Luu AT, et al. Benchmarking graph neural networks. J. Mach. Learn. Res, 2023, 24: 1-48
30 Lü L, Chen D, Ren XL, et al. Vital nodes identification in complex networks. Phys. Rep, 2016, 650: 1-63
31 Xu R, Zhang X, Shen Z, et al. A theoretical analysis on independence-driven importance Weighting for covariate-shift generalization. In: Proceedings of International Conference on Machine Learning (ICML), Baltimore, 2022.24803-24829
32 Dudzik AJ and Veličković P. Graph neural networks are dynamic programmers. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS), New Orleans, 2022. 20635-20647
33 Bevilacqua B, Nikiforou K, Ibarz B, et al. Neural algorithmic reasoning with causal regularisation. In: Proceedings of International Conference on Machine Learning (ICML), Hawai, 2023. 2272-2288
34 Cui P and Athey S. Stable learning establishes some common ground between causal inference and machine learning. Nat. Mach. Intell, 2022, 4: 110-115.
35 Shen Z, Cui P, Zhang T, et al. Stable learning via sample reweighting. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), New York, 2020. 5692-5699.
36 Nepusz GCAT and Csárdi G. The igraph software package for complex network research. Complex Syst, 2006, 1695: 1-9.
37 Hagberg AA, Schult DA and Swart PJ. Exploring network structure, dynamics, and function using networkX. Los Alamos National Laboratory (LANL), Los Alamos. 2008.
38 Kunegis J. KONECT: the koblenz network collection. In: Proceedings of the 22nd international conference on world wide web (IW3C2), io de Janeiro, 2013. 1343-1350
39 Rossi RA and Ahmed NK. The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), 2015. 29(1)