

Graph distillation with network symmetry

Feng Lin(林峰) and Jia-Lin He(何嘉林)

Citation: Chin. Phys. B, 2025, 34 (**4**): 040204. DOI: 10.1088/1674-1056/adbeda
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

Self-similarity of multilayer networks

Bing Wang(王冰), Huizhi Yu(于蕙芷), and Daijun Wei(魏代俊)
Chin. Phys. B, 2025, 34 (**1**): 010202. DOI: 10.1088/1674-1056/ad8a47

Opinion consensus incorporating higher-order interactions in individual-collective networks

Shun Ye(叶顺), Li-Lan Tu(涂俐兰), Xian-Jia Wang(王先甲), Jia Hu(胡佳), and Yi-Chao Wang(王薏潮)
Chin. Phys. B, 2024, 33 (**7**): 070201. DOI: 10.1088/1674-1056/ad3b84

Influence of random phase modulation on the imaging quality of computational ghost imaging

Chao Gao(高超), Xiao-Qian Wang(王晓茜), Hong-Ji Cai(蔡宏吉), Jie Ren(任捷), Ji-Yuan Liu(刘籍元), Zhi-Hai Yao(姚治海)
Chin. Phys. B, 2019, 28 (**2**): 020201. DOI: 10.1088/1674-1056/28/2/020201

Distance-based formation tracking control of multi-agent systems with double-integrator dynamics

Zixing Wu(吴梓杏), Jinsheng Sun(孙金生), Ximing Wang(王希铭)
Chin. Phys. B, 2018, 27 (**6**): 060202. DOI: 10.1088/1674-1056/27/6/060202

Ranking important nodes in complex networks by simulated annealing

Yu Sun(孙昱), Pei-Yang Yao(姚佩阳), Lu-Jun Wan(万路军), Jian Shen(申健), Yun Zhong(钟赟)
Chin. Phys. B, 2017, 26 (**2**): 020201. DOI: 10.1088/1674-1056/26/2/020201

Graph distillation with network symmetry

Feng Lin(林峰)[1] and Jia-Lin He(何嘉林)[1,2,3,†]

[1] *China West Normal University, Nanchong* 637000, *China*
[2] *The Internet of Things Perception and Big Data Analysis Key Laboratory of Nanchong City, Nanchong* 637000, *China*
[3] *Institute of Artificial Intelligence, China West Normal University, Nanchong* 637000, *China*

(Received 13 November 2024; revised manuscript received 25 February 2025; accepted manuscript online 11 March 2025)

Graph neural networks (GNNs) have demonstrated excellent performance in graph representation learning. However, as the volume of graph data grows, issues related to cost and efficiency become increasingly prominent. Graph distillation methods address this challenge by extracting a smaller, reduced graph, ensuring that GNNs trained on both the original and reduced graphs show similar performance. Existing methods, however, primarily optimize the feature matrix of the reduced graph and rely on correlation information from GNNs, while neglecting the original graph's structure and redundant nodes. This often results in a loss of critical information within the reduced graph. To overcome this limitation, we propose a graph distillation method guided by network symmetry. Specifically, we identify symmetric nodes with equivalent neighborhood structures and merge them into "super nodes", thereby simplifying the network structure, reducing redundant parameter optimization and enhancing training efficiency. At the same time, instead of relying on the original node features, we employ gradient descent to match optimal features that align with the original features, thus improving downstream task performance. Theoretically, our method guarantees that the reduced graph retains the key information present in the original graph. Extensive experiments demonstrate that our approach achieves significant improvements in graph distillation, exhibiting strong generalization capability and outperforming existing graph reduction methods.

Keywords: graph neural networks, graph distillation, network symmetry, super nodes, feature optimization

PACS: 02.10.Ox, 02.40.Pc, 07.05.Mh, 11.30.–j

DOI: 10.1088/1674-1056/adbeda **CSTR:** 32038.14.CPB.adbeda

1. Introduction

Graph data is widespread and inherently complex in fields such as social networks,[1] recommender systems,[2] and complex network research,[3] necessitating advanced analytical methods. Graph neural networks (GNNs)[4–6] have become pivotal in graph representation learning, leveraging message-passing frameworks where nodes update their embeddings through interactions with their neighbors. This approach effectively captures both structural and feature complexities, driving advancements in tasks such as node classification, link prediction, and graph classification. Notable models include GCN,[7] GAT,[8] and GraphSAGE.[9] However, the growing scale of graph data poses significant efficiency challenges for GNNs. To address this, simplification methods like graph thinning and coarsening have been proposed. These methods, however, often rely on heuristics that may not optimally preserve critical information or downstream performance. Recently, graph distillation[10] has emerged as a promising solution. This approach extracts and optimizes a reduced graph using information such as gradients,[11] feature representations,[12] and training trajectories[13] from both the original and distilled graphs. By doing so, GNNs trained on the distilled graph can achieve performance comparable to those trained on the full original graph.

Despite the remarkable results achieved by graph distillation, current methods applied to graph data often merely adapt techniques originally designed for image data, overlooking the critical role of graph structure. This oversight introduces several limitations: reduced graphs often fail to adequately preserve the structural information of the original graph, as they rely on learned node features to construct the adjacency matrices, which leads to structural loss; moreover, symmetric structures,[14] which are common in networks, introduce redundancy during GNN training. Symmetric nodes — those with identical neighborhood structures — are redundant, and failing to consider the graph structure leaves this redundancy intact, thereby negatively impacting performance. As illustrated in Fig. 4, symmetric nodes cause GNNs to learn similar representations, reducing learning efficiency.

To overcome these limitations, we propose a graph distillation method based on the principles of network symmetry. Our approach leverages the network symmetry theory[14,15] to coarsen the network while retaining the structural information of the original graph, thus mitigating structural loss. In addition, to further enhance performance, we apply a gradient descent algorithm to compute optimal node features, ensuring that the features of the reduced graph remain aligned with those of the original graph. Finally, we construct a complete reduced graph by integrating the adjacency matrix of the re-

[†]Corresponding author. E-mail: hejialin32@126.com

© 2025 Chinese Physical Society and IOP Publishing Ltd. All rights, including for text and data mining, AI training, and similar technologies, are reserved.

duced graph with the optimized synthetic features.

Our contributions are as follows:

(i) Analysis of redundant structures We analyze redundant structures in graph data processed by GNNs, with a particular focus on symmetric structures.

(ii) Proposal of a novel graph distillation method We propose a graph distillation approach that efficiently reduces redundant graph structure information by leveraging approximations of network symmetry.

These contributions are built upon the foundations of network symmetry theory, aiming to bridge the concepts of graph distillation and network symmetry to advance graph data analysis and deep learning research. We validate the state-of-the-art effectiveness of our method through extensive experiments on tasks such as node classification and link prediction.

2. Preliminaries

Consider a graph $G = (V, E, \boldsymbol{X})$, where V is the set of nodes with $|V| = N$ representing the number of nodes, E is the set of edges, and $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ is the node feature matrix. The adjacency matrix of G is denoted as $\boldsymbol{A} \in \{0,1\}^{N\times N}$, where $A_{ij} = 1$, if there exists an edge between nodes i and j, and $A_{ij} = 0$ otherwise. The node label matrix is denoted as $\boldsymbol{Y}$. The normalized Laplacian matrix of G is defined as $\boldsymbol{L} = \boldsymbol{I}_N - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{I}_N$ is the identity matrix of size $N \times N$, and $\boldsymbol{D}$ is the degree matrix with diagonal entries $D_{ii} = \sum_j A_{ij}$ and off-diagonal entries $D_{ij} = 0$ for $i \neq j$.

2.1. Computation graph

The message-passing paradigm[16] is a central concept in GNNs and forms the basis of many GNN models, where node embeddings are updated as follows:

$$\boldsymbol{h}_u^{(k+1)} = \sigma\left(\sum_{v\in N(u)} \boldsymbol{W}_k \boldsymbol{h}_v^{(k)} + \boldsymbol{B}_k \boldsymbol{h}_u^{(k)}\right). \tag{1}$$

Here, $N(u)$ denotes the set of first-order neighbors of node u, $\boldsymbol{h}_v^{(k)}$ represents the feature information of node v in the k-th layer, σ is the activation function, and $\boldsymbol{W}_k$ and $\boldsymbol{B}_k$ are the parameter matrices.

The physical meaning of this paradigm is that the representation of a node incorporates the features of all nodes within its l-order neighborhood. This process can be represented using a computational graph.[17] In message passing, the computational graph explicitly shows how the central node aggregates information from its neighbors and updates its own information. For example, the computational graph in Fig. 1 illustrates the embedding process of node 1 in a two-layer GNN.

(i) At the first layer, the information aggregation of node 5's neighbors (nodes 4, 2, and 1) is passed to node 5. The aggregated information is then transformed and updated to generate the embedding of node 5. The same process occurs for node 2.

(ii) At the second layer, the aggregated information from node 1's neighbors (nodes 5 and 2) is passed to node 1. The aggregated information is then updated through transformation to produce node 1's embedding.

Through this iterative process of aggregation and transformation, the node embedding after two iterations of message passing and transformation can serve as the representation of the node, enabling downstream prediction tasks.

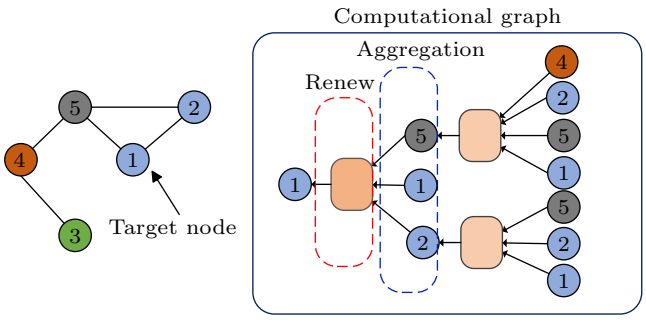


Fig. 1. The computational graph of graph neural network.

2.2. Network symmetry

Network symmetry refers to the properties of a network that remain unchanged under specific transformations, revealing patterns of structural similarity. The study of network symmetry involves analyzing the topological equivalence of nodes and edges, with group theory[18,19] serving as the underlying mathematical foundation. Below are key definitions related to network symmetry in graph theory.

Definition 1 A permutation of a finite set S is a one-to-one correspondence from S to itself. A group composed entirely of permutations is called a permutation group.

Definition 2 For a graph $G(V,E)$, let $S(V)$ represent the set of all permutations on V. A graph automorphism is a permutation in $S(V)$ that preserves the adjacency relationships of the graph. The set $\mathrm{Aut}(G) = \{g \in S(V) | E^g = E\}$ of all automorphisms forms a group under the permutation product operation.

Definition 3 Two nodes x and y in a graph $G(V,E)$ are symmetrically equivalent if there exists an automorphism $g \in \mathrm{Aut}(G)$ such that applying g to x yields y, written as $x^g = y$. This symmetry equivalence defines an equivalence relation, partitioning the vertex set into equivalence classes or orbits. The set of orbits is denoted by $\Delta(G) = \{\Delta_1, \Delta_2, \ldots, \Delta_n\}$, where each Δ_i represents an orbit of $\mathrm{Aut}(G)$.

Orbits[14] in group theory illustrate network symmetry where the local connectivity remains unchanged under vertex permutations. For example, as shown in Fig. 2, swapping node 1 with node 2 does not affect the network's adjacency relations; similarly, replacing node 5 with node 6 or 7 leaves the

network structure intact. Nodes within the same orbit are interchangeable without impacting the network topology, a property that can be visually represented by color-coding nodes of the same orbit. This concept is crucial for classifying nodes into symmetry-based equivalence classes, which has significant implications for understanding and analyzing the structure of complex networks.

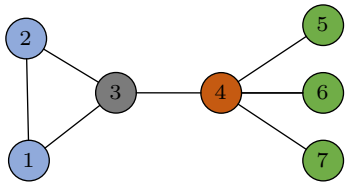


Fig. 2. Illustration of network symmetry and equivalence classes (distinct classes represented by different colors).

2.3. Graph distillation

Graph distillation (GD) is a technique designed to simplify graphs and mitigate issues related to label sparsity and high computational complexity in GNNs. It involves creating a compact reduced graph $G' = (V', E', \boldsymbol{X}')$, where the number of nodes $|V'|$ is significantly smaller than N ($|V'| \ll N$), and $\boldsymbol{X}' \in \mathbb{R}^{N' \times d}$. The objective is for GNNs trained on G' to achieve performance comparable to those trained on the original graph $G = (V, E, \boldsymbol{X})$.

Gradient matching in graph distillation is a technique that optimizes reduced graphs by aligning their gradient information with that of the original graphs. This alignment ensures that reduced graphs resemble the original ones, thereby preserving performance on downstream tasks while simplifying the graph structure. Consider a distillation model consisting of a single-layer GNN with a mean squared error (MSE) loss function, defined as

$$\mathscr{L} = \frac{1}{2}\|\boldsymbol{AXW} - \boldsymbol{Y}\|_F^2, \tag{2}$$

where $\boldsymbol{Y}$ is the node label matrix and $\boldsymbol{W}$ is the matrix of model parameters. By differentiating the MSE loss, we obtain the gradient expressions for the original and reduced graphs

$$\nabla_{\boldsymbol{W}} = (\boldsymbol{AX})^{\mathrm{T}}(\boldsymbol{AXW} - \boldsymbol{Y}), \tag{3}$$

$$\nabla'_{\boldsymbol{W}} = (\boldsymbol{A'X'})^{\mathrm{T}}(\boldsymbol{A'X'W} - \boldsymbol{Y'}). \tag{4}$$

The objective for gradient matching in the distillation process is then

$$\mathscr{L}_{\mathrm{GM}} = \|\nabla_{\boldsymbol{W}} - \nabla'_{\boldsymbol{W}}\|_F^2. \tag{5}$$

An upper bound for $\mathscr{L}_{\mathrm{GM}}$ can be established as

$$\begin{aligned}\mathscr{L}_{\mathrm{GM}} \leq\ & \|\boldsymbol{W}\|_F^2 \|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{X} - \boldsymbol{X'}^{\mathrm{T}}\boldsymbol{A'}^2\boldsymbol{X'}\|_F^2 \\ & + \|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{AY} - \boldsymbol{X'}^{\mathrm{T}}\boldsymbol{A'Y'}\|_F^2.\end{aligned} \tag{6}$$

In this inequality, $\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{X} - \boldsymbol{X'}^{\mathrm{T}}\boldsymbol{A'}^2\boldsymbol{X'}\|_F^2$ measures the discrepancy in node feature representations between the original and reduced graphs. $\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{AY} - \boldsymbol{X'}^{\mathrm{T}}\boldsymbol{A'Y'}\|_F^2$ quantifies the difference in label information.

These terms are crucial as they guide the optimization updates for the reduced graph, ensuring it retains essential characteristics of the original graph while being more computationally efficient.

3. The proposed method: GDNS

In this section, we introduce the proposed GDNS method, as illustrated in Fig. 3. This method improves fidelity compared to previous graph distillation approaches and reduces the computational complexity of the iterative process, thereby enhancing distillation efficiency. The GDNS method consists of two main steps:

(i) Simplifying the network structure Leveraging network symmetry theory to simplify the graph by reducing redundant structures while preserving critical structural information.

(ii) Matching node features Aligning the node features of the original and reduced graphs through gradient descent optimization to ensure fidelity in feature information transfer.

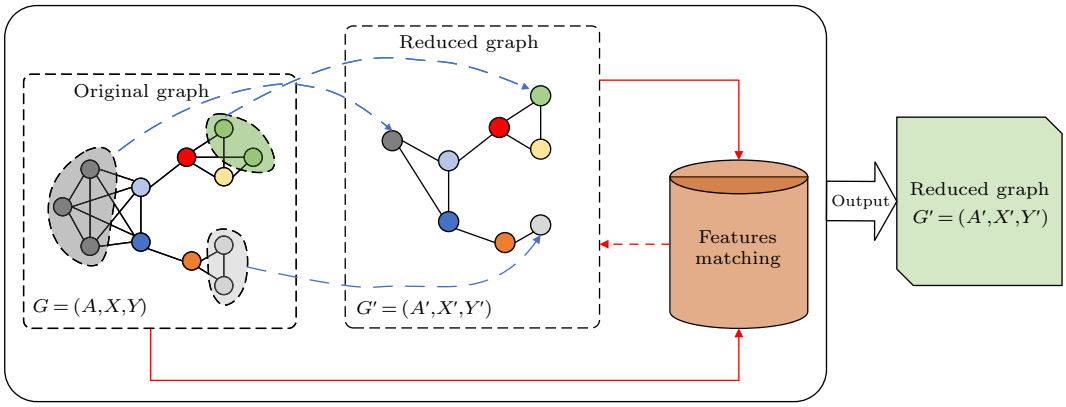


Fig. 3. Schematic overview of the GDNS architecture.

3.1. Structure redundancy

In this section, we analyze the symmetric structures in the graph and their impact on GNNs training, which motivates our approach. In the GNNs model, each node is associated with a computational graph, as described in Subsection 2.1. When two nodes have different computational graphs, their representations will also differ. Conversely, if two nodes share similar computational graphs, their representations will be similar. In a graph $G = (V, E)$, nodes can be partitioned into equivalence classes $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_k\}$ based on the network symmetry described in Subsection 2.2. Nodes within the same equivalence class are referred to as symmetric nodes. For these symmetric nodes, due to their similar neighborhood structures, their computational graphs are identical, leading to similar representations. For instance, in the network illustrated in Fig. 4, nodes 3, 4, and 5 are symmetric. They share the same computational graph, as shown in Fig. 4(b), resulting in identical representations. This redundancy leads to repetitive optimization of the GNNs' parameters, unnecessarily increasing computational complexity.

To validate our analysis, we applied the GCN model to link prediction tasks using three real-world datasets — Cora, CiteSeer, and PubMed (see Subsection 4.1 for details). We initially created a training set T by randomly selecting a subset of nodes from each network and then refined it by retaining only one representative node from each equivalence class Δ_i, resulting in a smaller training set T' to assess the impact of symmetric nodes on model efficiency. We use these two training sets to train the GNNs separately and test them using the test set of the original graph. After completing the experiments, we obtained significant results, as shown in Fig. 5. Specifically, figure 5(a) compares the AUC scores,[20] revealing consistent performance levels; in some instances, T' even slightly outperforms T. Figure 5(b) illustrates the training times, showing a significant reduction in computational requirements for T' compared to T. These findings highlight the impact of symmetric nodes on the computational complexity of GNN models and suggest that effectively managing these nodes can substantially enhance model efficiency without compromising performance.

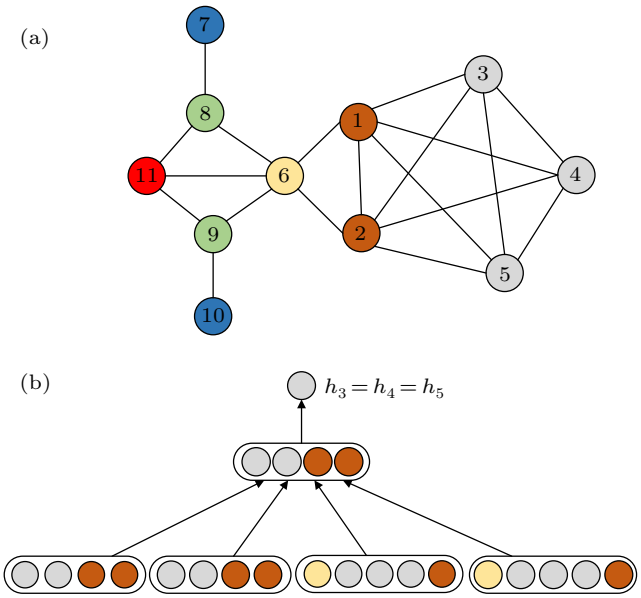


Fig. 4. (a) Illustration of equivalence classes, and (b) the embeddings of nodes 3, 4, and 5.

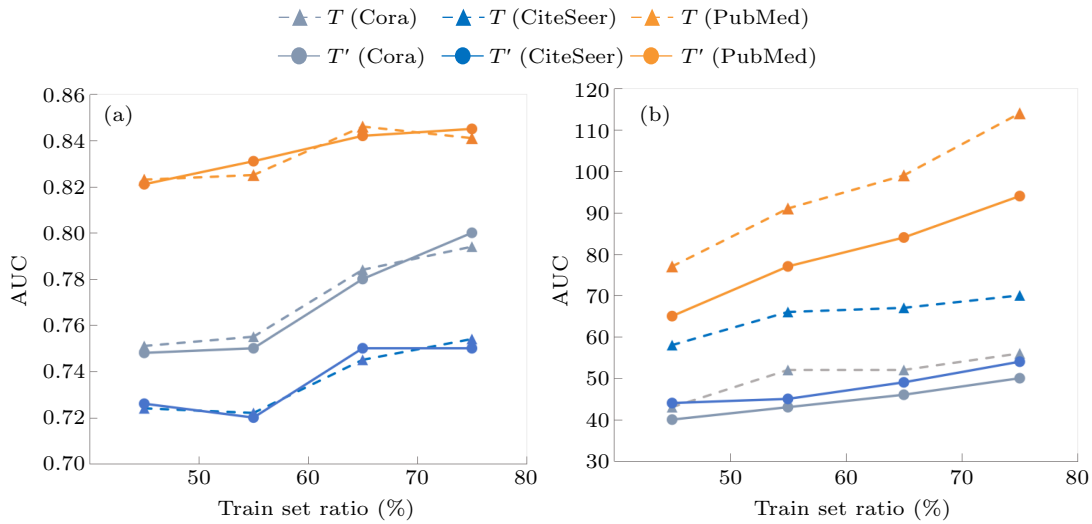


Fig. 5. (a) AUC scores and (b) training times for training sets T and T' across three real networks.

3.2. Reduced network

As discussed in Subsection 3.1, symmetric structures in networks reduce both the efficiency and effectiveness of GNNs training, indicating that retaining all symmetric structures is unnecessary. Therefore, we propose a symmetry removal strategy based on network symmetry theory. This strategy involves representing each equivalence class as a single super node, preserving the connectivity between equivalence

classes, as illustrated in Figs. 6(a) and 6(b). The specific theory is described as follows.

Let $\boldsymbol{A}$ be the $n \times n$ adjacency matrix of an original network, $\boldsymbol{S}$ the $n \times m$ characteristic matrix of the partition, where $[\boldsymbol{S}]_{ik} = 1$ if $i \in \Delta_k$ and 0 otherwise. Define $\boldsymbol{\Lambda} = \mathrm{diag}(n_1, \ldots, n_m)$, where $n_k = |\Delta_k|$ represents the size of each equivalence class. Then, we obtain $\boldsymbol{A}'$ by

$$\boldsymbol{A}' = \boldsymbol{\Lambda}^{-1}\boldsymbol{S}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{S}. \tag{7}$$

The $\boldsymbol{A}'$ is the $m \times m$ adjacency matrix of the reduced network, obtained by partitioning the vertex set into equivalence classes $\Delta = \Delta_1 \cup \cdots \Delta_m$. It is theoretically proven that the reduced graph $\boldsymbol{A}'$ preserves the key structural information of the original graph $\boldsymbol{A}$: The equivalence classes satisfy a regularity condition known as equitability,[14] which can be expressed in matrix form as

$$\boldsymbol{A}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{A}'. \tag{8}$$

If $(\lambda, \boldsymbol{v})$ is an eigenpair of $\boldsymbol{A}'$, then $(\lambda, \boldsymbol{S}\boldsymbol{v})$ is an eigenpair of $\boldsymbol{A}$, since

$$\boldsymbol{A}(\boldsymbol{S}\boldsymbol{v}) = \boldsymbol{S}\boldsymbol{A}'\boldsymbol{v} = \lambda(\boldsymbol{S}\boldsymbol{v}). \tag{9}$$

Thus, the spectrum of $\boldsymbol{A}'$ is a subset of the spectrum of $\boldsymbol{A}$, and the eigenvectors of $\boldsymbol{A}$ are lifted from the reduced graph by repeating entries on equivalence classes. Specifically, $\boldsymbol{A}$ has an eigenbasis of the form[14]

$$\{\boldsymbol{S}\boldsymbol{v}_1, \ldots, \boldsymbol{S}\boldsymbol{v}_m, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n-m}\}, \tag{10}$$

where $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ is any eigenbasis of $\boldsymbol{A}'$, and $\boldsymbol{S}^{\mathrm{T}}\boldsymbol{w}_i = 0$ for all i. This implies that $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n-m}\}$ are redundant eigenvectors, reflecting the network's symmetries. Their entries sum to zero within each equivalence class. In contrast, $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$ are non-redundant eigenvectors that preserve the essential structural information of the original graph.

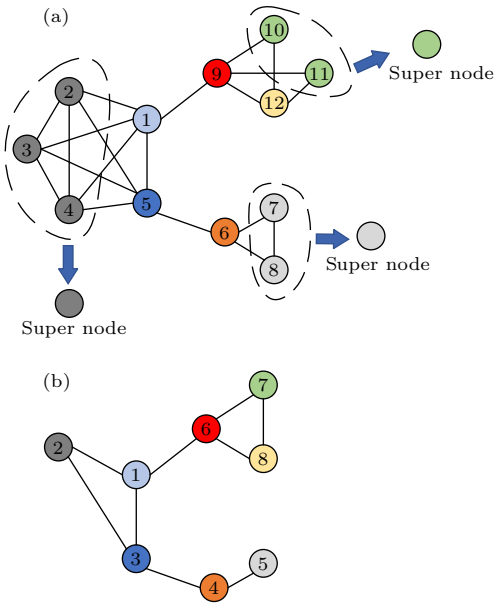


Fig. 6. (a) An illustrative graph with symmetric nodes, (b) the reduced graph.

3.3. Feature matching

While the coarsened adjacency matrix $\boldsymbol{A}'$ enhances computational efficiency in training GNNs, it may lead to a decrease in accuracy due to the loss of detailed structural and feature information. To address this issue, we propose relearning the node feature matrix $\boldsymbol{X}'$ to compensate for the accuracy degradation. Inspired by the upper bound established in Eq. (6), we design a loss function that directly optimizes $\boldsymbol{X}'$ by minimizing the discrepancies between the original graph G and the reduced graph G':

$$\begin{aligned} f(\boldsymbol{X}') = {} & \|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{X} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'^2\boldsymbol{X}'\|_F^2 \\ & + \|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'\boldsymbol{Y}'\|_F^2. \end{aligned} \tag{11}$$

This loss function comprises two critical components that together ensure the reduced graph retains essential characteristics of the original graph. The first term $\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{X} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'^2\boldsymbol{X}'\|_F^2$ measures the difference in node features influenced by second-order proximities between the original and reduced graphs. By considering $\boldsymbol{A}^2$ and $\boldsymbol{A}'^2$, we capture structural information up to two hops, ensuring that $\boldsymbol{X}'$ reflects similar structural patterns as $\boldsymbol{X}$. The second term $\|\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'\boldsymbol{Y}'\|_F^2$ quantifies the divergence in label information propagation, ensuring that the relationship between node features and labels in the reduced graph aligns with that of the original graph.

To optimize $\boldsymbol{X}'$, we compute the gradient of the loss function $f(\boldsymbol{X}')$ with respect to $\boldsymbol{X}'$. Since $f(\boldsymbol{X}')$ consists of two squared Frobenius norms, we can compute the gradients accordingly.

The gradient of the first term is

$$\nabla_{\boldsymbol{X}'} f_1(\boldsymbol{X}') = -2\boldsymbol{A}'^2\boldsymbol{X}'(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}^2\boldsymbol{X} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'^2\boldsymbol{X}')^{\mathrm{T}}. \tag{12}$$

The gradient of the second term is

$$\nabla_{\boldsymbol{X}'} f_2(\boldsymbol{X}') = -2\boldsymbol{A}'\boldsymbol{Y}'(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Y} - \boldsymbol{X}'^{\mathrm{T}}\boldsymbol{A}'\boldsymbol{Y}')^{\mathrm{T}}. \tag{13}$$

Therefore, the total gradient is

$$\nabla_{\boldsymbol{X}'} f(\boldsymbol{X}') = \nabla_{\boldsymbol{X}'} f_1(\boldsymbol{X}') + \nabla_{\boldsymbol{X}'} f_2(\boldsymbol{X}'). \tag{14}$$

To minimize $f(\boldsymbol{X}')$, we use gradient descent or other optimization algorithms. The update rule for $\boldsymbol{X}'$ is

$$\boldsymbol{X}' \leftarrow \boldsymbol{X}' - \eta \nabla_{\boldsymbol{X}'} f(\boldsymbol{X}'), \tag{15}$$

where η is the learning rate. By minimizing $f(\boldsymbol{X}')$ we adjust $\boldsymbol{X}'$ to closely match the structural and label information of the original graph. This optimization allows the reduced graph to retain essential properties of G, thereby compensating for the loss of accuracy introduced during the coarsening

process. The refined feature matrix $\boldsymbol{X}'$ enhances the performance of GNNs trained on G', achieving a balance between computational efficiency and model accuracy.

Algorithm 1 presents the pseudocode for the GDNS method. Theoretically, our reduced graph preserves key information and structural similarities of the original graph while aligning node features. This results in improved performance when used to train GNNs.

Algorithm 1: GDNS.

Input: Original graph $G = (\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y})$.
Output: Reduced graph $G = (\boldsymbol{A}', \boldsymbol{X}', \boldsymbol{Y}')$.

```
Identify equivalence classes Δ_1, Δ_2, ..., Δ_k based on network symmetry.
for each equivalence Δ_i do
    Create super node V_i representing Δ_i.
    Update X'_{v_i} ← (1/|Δ_i|) Σ_{v∈Δ_i} X_v.
    Remove symmetric nodes and connected edges.
end for
Update A', Y'.
for t to T do
    Compute f(X') via Eq. (11).
    Update X' ← X' − η∇_{X'} f(X').
end for
return Reduced graph G = (A', X', Y')
```

4. Experimental results

In this section, we conduct extensive experiments, including node classification and link prediction on various datasets, to validate the effectiveness and generality of GDNS.

4.1. Datasets

For the node classification task, we used four widely studied citation network datasets commonly employed in graph neural network research: Cora,[7] CiteSeer,[7] PubMed,[7] and Squirrel.[21] For the link prediction task, we utilized four datasets: Cora, CiteSeer, PubMed, and DBLP.[22] Among these datasets, DBLP is a heterogeneous graph, which includes three types of nodes (paper (P), author (A), conference (C)) and four types of edges (PA, AP, PC, CP). The other datasets (Cora, CiteSeer, PubMed, and Squirrel) are homogeneous graphs, containing only one type of node. In the homogeneous citation networks, such as Cora, CiteSeer, and PubMed, the nodes represent papers, while in Squirrel, the nodes represent articles. Table 1 summarizes each network's characteristics, including the number of nodes ($|V|$), edges ($|E|$), classes ($|C|$), the dimension of initial node features (d_{f}), the number of equivalence classes ($|\Delta|$), and the ratio of symmetric nodes (R_{SN}).

Table 1. Detailed topological structures of five networks.

Datasets	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert C\rvert$	d_{f}	$\lvert\Delta\rvert$	R_{SN}
Cora	2708	5429	7	1433	2365	0.13
CiteSeer	3327	4732	6	3703	2091	0.37
PubMed	19717	44338	3	500	12998	0.34
Squirrel	5201	396846	5	2089	3301	0.36
DBLP	18405	67946	–	334	7546	0.41

4.2. Benchmark methods

To evaluate the effectiveness of GDNS, we compared it with eight benchmark graph simplification methods. These methods include:

(I) Four traditional methods:

(i) Random sampling (selecting a subset of nodes or edges at random);

(ii) Herding[23] (iteratively selecting representative nodes or edges that best describe the data distribution);

(iii) K-center[24] (choosing nodes to minimize the maximum distance to any unselected node);

(iv) A graph coarsening method[25] (reducing graph size by merging similar nodes or edges).

(II) Four graph distillation methods:

(i) Three gradient matching methods (GCOND,[11] SGDD,[26] GDC[11]);

(ii) One trajectory matching method (SFGC[13]).

4.3. Experimental setup

To effectively evaluate the quality of reduced graphs generated by GDNS, we followed a two-step process for all reduced graphs obtained from different methods. First, a GNN was trained on the reduced graph to obtain model parameters. Second, the performance of the GNN was tested on the original graph's test set to assess the quality of the reduced graphs.

In the graph distillation methods, specific configurations were applied to ensure consistency and comparability. GCOND was distilled using a 2-layer SGC and evaluated on 2-layer GCNs with 256 hidden units. Similarly, SGDD used a 2-layer SGC for distillation and was evaluated on 2-layer GCNs with 256 hidden units. In contrast, SFGC was both distilled and evaluated using 2-layer GCNs with the same hidden unit configuration.

For dataset splitting, we adopted standard practices. For node classification tasks, we used the common split widely applied across datasets, while for link prediction tasks, the datasets were divided following an 8:1:1 ratio (training/validation/test). To validate the generalization ability of GDNS, we conducted extensive experiments using different GNN architectures, all implemented within the CogDL0.6 [27] deep learning framework.

The training process was optimized using the Adam optimizer with a learning rate of 0.01. All models were trained for up to 2000 iterations, integrating an early stopping mechanism (PATIENCE) during the validation phase to minimize the risk of overfitting. To ensure the reliability of the results, the final model performance was reported as the average of

over 100 runs, providing robust and statistically meaningful performance metrics.

4.4. Node classification

Table 2 shows ACC and F1 scores across four datasets. Traditional graph simplification methods perform the worst due to their exclusive focus on structural reduction without considering downstream GNNs task performance. While graph distillation methods (GCond, SFGC, and SGDD) attempt to address this by optimizing node features and graph structure, their aggressive reduction strategies result in the loss of critical structural information that cannot be recovered during optimization. Our proposed GDNS method, grounded in network symmetry theory, selectively eliminates redundant information while preserving essential structural properties. Results in Table 2 demonstrate that GDNS-generated reduced graphs consistently outperform all baseline methods across all datasets, validating our hypothesis about the importance of structural information preservation for GNNs' performance.

4.5. Link prediction

On DBLP, we propose a straightforward heterogeneous graph processing approach, as illustrated in Fig. 7 using P-type nodes. The methodology comprises three key steps: (i) initializing an empty graph containing only P-type nodes; (ii) constructing edges between P-nodes through meta-path connections from the original graph to form a homogeneous subgraph; and (iii) applying graph reduction techniques followed by GNNs training for node representation learning. This process is replicated for all node types, with the resulting representations ultimately being used for link prediction tasks.

Table 3 presents the link prediction performance metrics (AUC) across four benchmark datasets. Consistent with the node classification results where traditional graph simplification methods underperform compared to graph distillation approaches, our method demonstrates superior performance. Specifically, GDNS achieves state-of-the-art results on three datasets, showing improvements of 1.9% (Cora) and 3.1% (CiteSeer) over the baseline. These enhancements stem from our method's effective preservation of original graph structures coupled with downstream task optimization. Notably, the PubMed dataset exhibits minimal performance gains, which we hypothesize may be attributed to its higher average node degree characteristics.

Table 2. The ACC values and F1 scores of eight methods on four networks.

Dataset	Cora		CiteSeer		PubMed		Squirrel	
Metric	ACC	F1	ACC	F1	ACC	F1	ACC	F1
Random	0.768	0.779	0.642	0.625	0.733	0.731	0.250	0.241
Coarsening	0.706	0.697	0.590	0.582	0.287	0.266	0.211	0.196
Herding	0.769	0.780	0.667	0.67	0.762	0.774	0.214	0.203
K-center	0.768	0.775	0.643	0.625	0.694	0.687	0.228	0.216
Gcond	0.798	0.805	0.707	0.691	0.771	0.769	0.257	0.238
SFGC	0.806	0.812	0.724	0.718	0.775	0.77	0.269	0.252
SGDD	0.801	0.804	0.702	0.696	0.780	0.78	0.247	0.227
Whole	0.810	0.826	0.717	0.71	0.793	0.789	0.330	0.327
GDNS	0.822	0.840	0.744	0.732	0.829	0.825	0.349	0.338

Table 3. The AUC scores of eight methods on four networks.

Dataset	Cora	CiteSeer	PubMed	DBLP
Random	0.587	0.548	0.623	0.675
Coarsening	0.675	0.642	0.726	0.753
Herding	0.609	0.577	0.656	0.738
K-center	0.676	0.643	0.731	0.756
GDC	0.703	0.667	0.758	0.778
Gcond	0.710	0.673	0.769	0.803
SGDD	0.739	0.708	0.797	0.822
Whole	0.784	0.745	0.846	0.837
GDNS	0.803	0.776	0.848	0.798

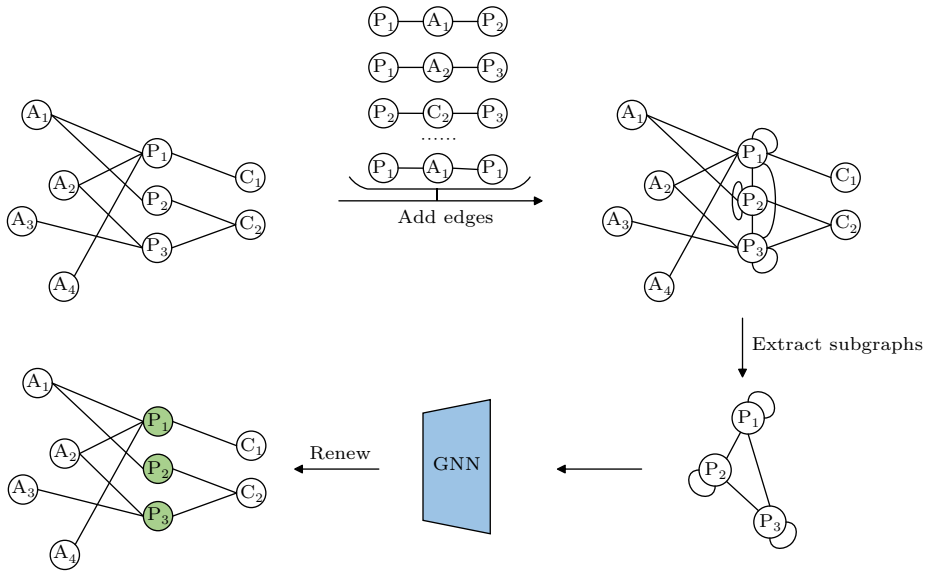


Fig. 7. Heterogeneous graph transformation process (take the type P node as an example).

4.6. Generalization ability

We evaluated GDNS's generalization capability through node classification experiments using six GNNs architectures (GCN, SGC,[28] PPNP,[29] GAT, GRAND,[30] and MixHop[31]) and compared against three graph distillation baselines (GCOND, SFGC, and SGDD). Results in Table 4 demonstrate GDNS's superior performance across all GNNs models, achieving average accuracy improvements of 4.6%, 3.8%, and 6.6% on the three datasets, respectively, compared to GCOND. This consistent outperformance across diverse GNNs architectures and datasets validates GDNS's robust generalization capability.

Table 4. The ACC values of six GNNs methods on three networks.

Dataset	Methods	GCN	SGC	PPNP	GAT	GRAND	MixHop	Avg.	Impro (%)
Cora	GCOND	0.798	0.795	0.793	0.662	0.761	0.772	0.764	↑4.60
	SFGC	0.806	0.810	0.808	0.799	0.763	0.789	0.796	↑1.40
	SGDD	0.801	0.804	0.796	0.784	0.754	0.778	0.786	↑2.40
	GDNS	0.826	0.822	0.820	0.822	0.778	0.793	0.810	–
CiteSeer	GCOND	0.707	0.703	0.696	0.691	0.714	0.708	0.703	↑3.80
	SFGC	0.724	0.718	0.705	0.713	0.731	0.728	0.72	↑2.10
	SGDD	0.702	0.713	0.692	0.706	0.709	0.703	0.704	↑3.70
	GDNS	0.744	0.718	0.738	0.749	0.747	0.749	0.741	–
PubMed	GCOND	0.777	0.776	0.773	0.708	0.781	0.795	0.768	↑6.60
	SFGC	0.775	0.774	0.776	0.769	0.797	0.801	0.782	↑5.20
	SGDD	0.780	0.766	0.787	0.791	0.804	0.807	0.789	↑4.50
	GDNS	0.829	0.815	0.840	0.821	0.852	0.845	0.834	–

4.7. Ablation study

To validate the effectiveness of the two components in GDNS, we conducted an ablation study on three citation network datasets (Cora, CiteSeer, and PubMed). We performed link prediction experiments on these datasets using different training set ratios.

Figure 8 presents comparative results between reduced and original graphs, with Fig. 8(a) showing performance without feature matching and Fig. 8(b) with feature matching. The reduced graph demonstrates enhanced GNNs training effectiveness, validating our analysis in Section 3. Feature matching further improves the reduced graph's performance, as evidenced in Fig. 8(b). Notably, the reduced graph maintains consistent superior performance across varying training set ratios, confirming GDNS's stability under diverse training set ratios.

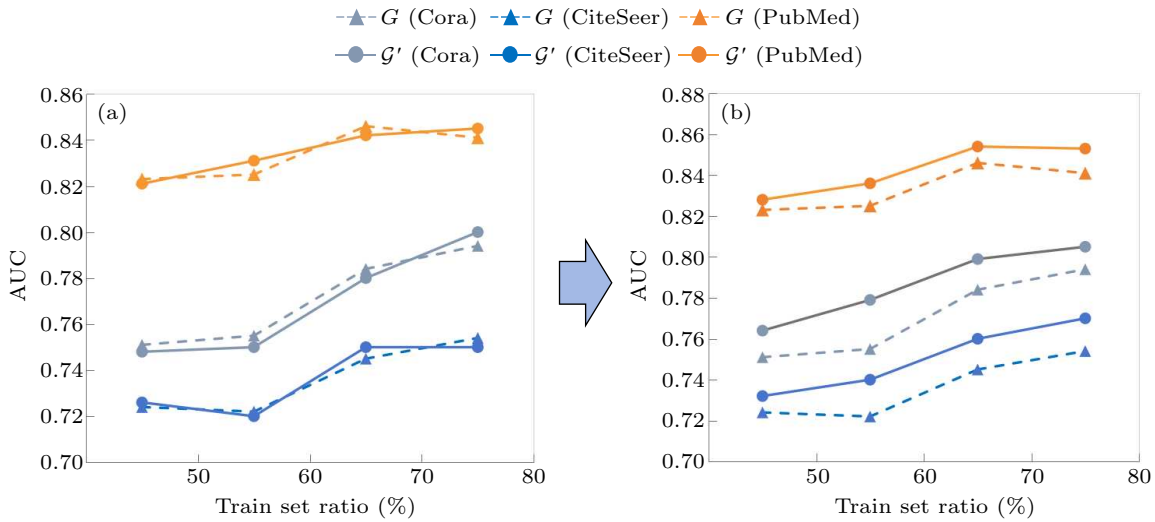


Fig. 8. AUC scores across three networks for (a) training sets G and $G' = (\boldsymbol{A}', \boldsymbol{X}, \boldsymbol{Y})$, and (b) G and $G' = (\boldsymbol{A}', \boldsymbol{X}', \boldsymbol{Y}')$.

4.8. Computational efficiency

Figure 9 compares the GNNs training time between reduced and original graphs across four datasets. The results demonstrate substantially reduced training times for reduced graphs, with the disparity increasing with epoch count, particularly evident in DBLP, PubMed, and CiteSeer datasets. By eliminating redundant connections and information, the reduced graphs exhibit a notably slower growth in computational overhead across training epochs, effectively mitigating the exponential increase in training time characteristic of the

original graphs. This optimization proves especially beneficial for large-scale graph datasets.

Figure 10 shows GNNs memory consumption during training across four datasets. GDNS-generated reduced graphs achieve significant memory reduction, with CiteSeer and PubMed datasets showing 30%–40% decreased usage. The Cora dataset exhibits minimal memory differential between original and reduced graphs due to its limited symmetrical components. These results demonstrate GDNS's effectiveness in reducing GNNs memory overhead through redundant structure elimination.

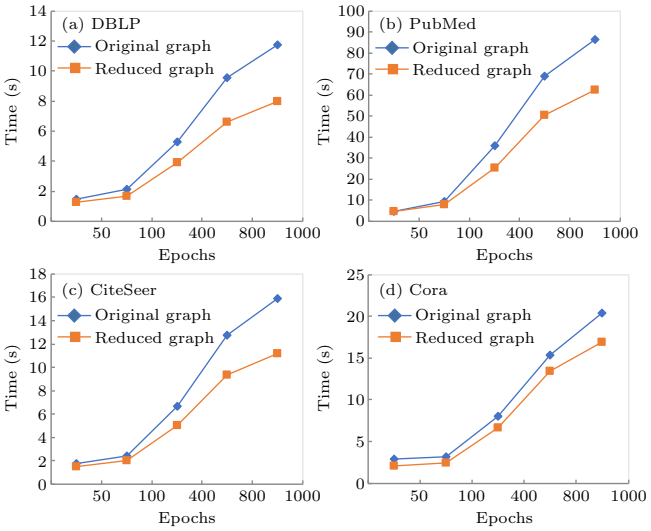


Fig. 9. Training time of GNNs on the synthetic and original graphs across four datasets.

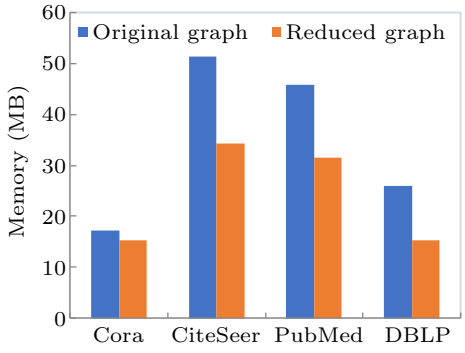


Fig. 10. Memory usage of GNNs on the synthetic and original graphs across four datasets.

5. Related work

5.1. Graph neural networks

Graph neural networks (GNNs) are neural network models designed for graph-structured data, integrating the strengths of graph computation and neural networks to capture both structural information and abstract node features. GNNs are generally categorized into spectral-based and spatial-based approaches. Spectral-based GNNs, such as ChebyNet,[32] BernNet,[33] and GWNN,[34] utilize spectral graph theory to design filtering operations in the spectral domain. In contrast, spatial-based GNNs, including GAT, GraphSAGE, and spatial GCN,[35] explicitly leverage the graph's structure to perform feature extraction in the spatial domain.

5.2. Dataset distillation & graph coarsening

Dataset distillation[36,37] aims to reduce the time and memory required to train deep networks on large datasets by creating a small synthetic dataset that retains the generalization properties of the full dataset. This technique has proven highly useful in applications such as continual learning and neural architecture search. Graph coarsening[38,39] operates on the principle of aggregating similar nodes into super-nodes to produce a smaller graph, thereby making large-scale graph processing more efficient. It is widely used in graph neural networks, graph embedding, and graph partitioning.

6. Conclusion

In this paper, we identified a substantial number of symmetric structures within network graphs that are ineffective for training GNNs. To address this, we proposed a graph distillation method based on network symmetry (GDNS). Our method first coarsens the network by removing symmetric structures according to symmetry theory, then aligns the coarsened graph with the original node features to construct a reduced graph. Theoretically, GDNS eliminates redundant structural information while preserving the essential information of the original graph. Our experiments confirm these theoretical conclusions.

Despite its notable success, GDNS does lose some informational properties during compression, especially in heterogeneous graphs. Therefore, future work will focus on extending GDNS to heterogeneous graphs, providing a promising avenue for further research in network simplification.

Acknowledgments

Project supported by the National Natural Science Foundation of China (Grant No. 62176217), the Program from the Sichuan Provincial Science and Technology, China (Grant No. 2018RZ0081), and the Fundamental Research Funds of China West Normal University (Grant No. 17E063).

References

[1] Mitchell J C 1974 *Annual Review of Anthropology* **3** 279
[2] Lü L, Medo M, Yeung C H, Zhang Y C, Zhang Z K and Zhou T 2012 *Physics Reports* **519** 1
[3] Cui D X, He J L, Xiao Z F and Ren W P 2023 *Chin. Phys. B* **32** 098904
[4] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C and Sun M 2020 *AI open* **1** 57
[5] Lin B and Wang Y X 2024 *Chin. Phys. B* **33** 084401
[6] Kong J G, Li Q X, Li J, Lin Y and Zhu J J 2022 *Chin. Phys. Lett.* **39** 067503
[7] Kipf T N and Welling M 2016 arXiv:1609.02907[cs.LG]

[8] Veličković P, Cucurull G, Casanova A, Romero A, Lio P and Bengio Y 2017 arXiv:1710.10903[stat.ML]
[9] Hamilton W, Ying Z and Leskovec J 2017 arXiv:1706.02216 [cs.SI]
[10] Gao X, Yu J, Jiang W, Chen T, Zhang W and Yin H 2024 arxiv:2401.11720[cs.LG]
[11] Jin W, Zhao L, Zhang S, Liu Y, Tang J and Shah N 2021 arxiv:2110.07580[cs.LG]
[12] Liu M, Li S, Chen X and Song L 2022 arxiv:2206.13697[cs.LG]
[13] Zheng X, Zhang M, Chen C, Nguyen Q V H, Zhu X and Pan S 2023 arXiv:2306.02664 [cs.LG]
[14] Sánchez-García R J 2020 *Communications Physics* **3** 87
[15] McKay B D and Piperno A 2014 *Journal of Symbolic Computation* **60** 94
[16] Gilmer J, Schoenholz S S, Riley P F, Vinyals O and Dahl G E 2017 arXiv:1704.01212 [cs.LG]
[17] You J, Gomes-Selman J M, Ying R and Leskovec J 2021 *Proceedings of the AAAI Conference on Artificial Intelligence* **35** 10737
[18] Xiao Y, Macarthur B D, Wang H and Xiong M 2008 *Phys. Rev. E* **78** 046102
[19] Slansky R 1981 *Physics Reports* **79** 1
[20] Bradley A P 1997 *Pattern Recognition* **30** 1145
[21] Rozemberczki B, Allen C and Sarkar R 2021 *Journal of Complex Networks* **9** cnab014
[22] Yun S, Jeong M, Kim R, Kang J and Kim H J 2019 arXiv:1911.06455[cs.LG]
[23] Welling M 2009 ICML '09, June 14–18, Montreal Quebec, Canada, p.1121
[24] Sener O and Savarese S 2017 arXiv:1708.00489[stat.ML]
[25] Huang Z, Zhang S, Xi C, Liu T and Zhou M 2021 KDD '21, August 14-18, Singapore, p. 675
[26] Lim D, Hohne F, Li X, Huang S L, Gupta V, Bhalerao O and Lim S 2021 arXiv:2110.14446 [cs.LG]
[27] Cen Y, Hou Z, Wang Y, *et al.* 2023 WWW '23, April 30-May 4, Austin TX, USA, p. 747
[28] Wu F, Souza A, Zhang T, Fifty C, Yu T and Weinberger K 2019 arXiv:1902.07153 [cs.LG]
[29] Hassani K and Khasahmadi A H 2020 arXiv:2006.05582 [cs.LG]
[30] Feng W, Zhang J, Dong Y, Han Y, Luan H, Xu Q, Yang Q, Kharlamov E and Tang J 2020 NeurIPS2020, December 6–12, Online, p. 22092
[31] Abu-El-Haija S, Perozzi B, Kapoor A, Alipourfard N, Lerman K, Harutyunyan H, Steeg G V and Galstyan A 2019 arXiv:1905.00067 [cs.LG]
[32] Tang S, Li B and Yu H 2024 *Communications in Mathematics and Statistics* **2024** 1
[33] He M, Wei Z and Xu H 2021 NeurIPS2021, December 6–14, Online, p. 14239
[34] Xu B, Shen H, Cao Q, Qiu Y and Cheng X 2019 arXiv:1904.07785[cs.LG]
[35] Danel T, Spurek P, Tabor J, Smieja M, Struski L, Slowik A and Maziarka L 2020 ICONIP 2020, November 23–27, Bangkok, Thailand, p. 668
[36] Wang T, Zhu J Y, Torralba A and Efros A A 2018 arXiv:1811.10959[cs.LG]
[37] Zhao B, Mopuri K R and Bilen H 2020 arXiv:2006.05929[cs.CV]
[38] Loukas A 2019 arXiv:1808.10650 [cs.DS]
[39] Loukas A and Vandergheynst P 2018 arXiv:1802.07510 [cs.LG]