SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization

Dongcheng Zou
Beihang University
Beijing, China
zoudongcheng@buaa.edu.cn

Hao Peng*
Beihang University
Beijing, China
penghao@buaa.edu.cn

Xiang Huang
Beihang University
Beijing, China
huang.xiang@buaa.edu.cn

Renyu Yang
Beihang University
Beijing, China
renyu.yang@buaa.edu.cn

Jianxin Li
Beihang University
Beijing, China
lijx@buaa.edu.cn

Jia Wu
Macquarie University
Sydney, Australia
jia.wu@mq.edu.au

Chunyang Liu
Didi Chuxing
Beijing, China
liuchunyang@didiglobal.com

Philip S. Yu
University of Illinois Chicago
Chicago, USA
psyu@uic.edu

ABSTRACT

Graph Neural Networks (GNNs) are de facto solutions to structural data learning. However, it is susceptible to low-quality and unreliable structure, which has been a norm rather than an exception in real-world graphs. Existing graph structure learning (GSL) frameworks still lack robustness and interpretability. This paper proposes a general GSL framework, SE-GSL, through structural entropy and the graph hierarchy abstracted in the encoding tree. Particularly, we exploit the one-dimensional structural entropy to maximize embedded information content when auxiliary neighbourhood attributes is fused to enhance the original graph. A new scheme of constructing optimal encoding trees are proposed to minimize the uncertainty and noises in the graph whilst assuring proper community partition in hierarchical abstraction. We present a novel sample-based mechanism for restoring the graph structure via node structural entropy distribution. It increases the connectivity among nodes with larger uncertainty in lower-level communities. SE-GSL is compatible with various GNN models and enhances the robustness towards noisy and heterophily structures. Extensive experiments show significant improvements in the effectiveness and robustness of structure learning and node representation learning.

*Corresponding author

WWW '23, April 30–May 04, 2023, Austin, TX, USA

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583453

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; • **Mathematics of computing** → *Graph algorithms*; • **Information systems** → *Data mining*.

KEYWORDS

Graph structure learning, structural entropy, graph neural network

ACM Reference Format:
Dongcheng Zou, Hao Peng, Xiang Huang, Renyu Yang, Jianxin Li, Jia Wu, Chunyang Liu, and Philip S. Yu. 2023. SE-GSL: A General and Effective Graph Structure Learning Framework through Structural Entropy Optimization. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30–May 04, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583453

1 INTRODUCTION

Graph Neural Networks (GNNs) [49, 59] have become the cornerstone and de facto solution of structural representation learning. Most of the state-of-the-art GNN models employ message passing [12] and recursive information aggregation from local neighborhoods [17, 31, 41, 52] to learn node representation. These models have been advancing a variety of tasks, including node classification [29, 45, 50], node clustering [2, 32], graph classification [30, 54], and graph generation [55], etc.

GNNs are extremely sensitive to the quality of given graphs and thus require resilient and high-quality graph structures. However, it is increasingly difficult to meet such a requirement in real-world graphs. Their structures tend to be noisy, incomplete, adversarial, and heterophily (i.e., the edges with a higher tendency to connect nodes of different types), which can drastically weaken the representation capability of GNNs [6, 25, 28]. Recent studies also reveal that even a minor perturbation in the graph structure can lead to inferior prediction quality [3, 38, 57]. Additionally, GNNs are vulnerable to attacks since the raw graph topology is decoupled from node features, and attackers can easily fabricate links between entirely different nodes [38, 57].

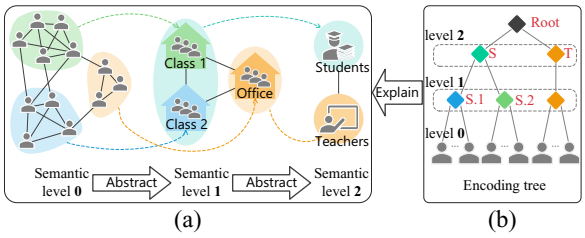

Figure 1: An illustrative example of the hierarchical community (semantics) in a simple social network. (1) Vertices and edges represent the people and their interconnectivity (e.g., common locations, interests, occupations). There are different abstraction levels, and each community can be divided into sub-communities in a finer-grained manner (e.g., students are placed in different classrooms while teachers are allocated different offices). The lowest abstraction will come down to the individuals with own attributes, and the highest abstraction is the social network system. (b) An encoding tree is a natural form to represent and interpret such a multi-level hierarchy.

To this end, Graph Structure Learning (GSL) [5, 16, 33, 39, 44, 61, 62] becomes the recent driving force for learning superior task-relevant graph topology and for enhancing the resilience and robustness of node representation. The existing works focus on jointly optimizing GNN whilst imposing regularization on refined graph structures. Typical methods include metric-based [5, 23, 44], probabilistic sampling [8, 34, 58], and learnable structure approach [16], etc. While promising, GNNs and GSL still have the following issues. *i) robustness to system noises and heterophily graphs.* While many GSL models strive to fuse node features and topological features through edge reconstruction (e.g., add, prune, or reweight) [44, 57, 62], additional noises and disassortative connections will be inevitably involved in the fused structure due to the unreliable priori topology and node embeddings, which would further degrade the GNNs representation capability [22]. *ii) model interpretability.* Fully parameterizing the adjacency matrix will incur a non-negligible cost of parameter storage and updating and is liable to low model interpretability [13]. Although some studies on the improved GNN interpretability [15, 53], few works can effectively explain the topology evolution during graph structure learning. Therefore, fusing the node and topological features in a noisy system environment to obtain GNN-friendly graphs by exploiting inherent graph structures is still an underexplored problem [48].

In this paper, we present SE-GSL, a general and effective graph structure learning framework that can adaptively optimize the topological graph structure in a learning-free manner and can achieve superior node representations, widely applicable to the mainstream GNN models. This study is among the first attempts to marry the structural entropy and encoding tree theory [18] with GSL, which offers an effective measure of the information embedded in an arbitrary graph and structural diversity. The multi-level semantics of a graph can be abstracted and characterized through an encoding tree. Encoding tree [18, 20, 56] represents a multi-grained division of graphs into hierarchical communities and sub-communities, thus providing a pathway to better interpretability. Fig. 1 showcases how such graph semantics are hierarchically abstracted. Specifically, we first enhance the original graph topology by incorporating the vertex similarities and auxiliary neighborhood information via the k-Nearest Neighbors (k-NN) approach, so that noise can be better identified and diminished. This procedure is guided by the k-selector that maximizes the amount of embedded information in the graph structure. We then propose a scheme to establish an optimal encoding tree to minimize the graph uncertainty and edge noises whilst maximizing the knowledge in the encoding tree. To restore the entire graph structure that can be further fed into GNN encoders, we recover edge connectivity between related vertices from the encoding tree taking into account the structural entropy distribution among vertices. The core idea is to weaken the association between vertices in high-level communities whilst establishing dense and extensive connections between vertices in low-level communities. The steps above will be iteratively conducted to co-optimize both graph structure and node embedding learning. SE-GSL[1] is an interpretable GSL framework that effectively exploits the substantive structure of the graph. We conduct extensive experiments and demonstrate significant and consistent improvements in the effectiveness of node representation learning and the robustness of edge perturbations.

Contribution highlights: i) SE-GSL provides a generic GSL solution to improve both the effectiveness and robustness of the mainstream GNN approaches. ii) SE-GSL offers a new perspective of navigating the complexity of attribute noise and edge noise by leveraging structural entropy as an effective measure and encoding tree as the graph hierarchical abstraction. iii) SE-GSL presents a series of optimizations on the encoding tree and graph reconstruction that can not only explicitly interpret the graph hierarchical meanings but also reduce the negative impact of unreliable fusion of node features and structure topology on the performance of GNNs. iv) We present a visualization study to reveal improved interpretability when the graph structure is evolutionary.

2 PRELIMINARIES

This section formally reviews the basic concepts of Graph, Graph Neural Networks (GNNs), Graph Structure Learning (GSL), and Structural Entropy. Important notations are given in Appendix A.1.

2.1 Graph and Graph Structure Learning

Graph and Community. Let $G = \{V, E, X\}$ denote a graph, where V is the set of n vertices[2], $E \subseteq V \times V$ is the edge set, and $X \in \mathbb{R}^{n \times d}$ refers to the vertex attribute set. $\mathrm{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix of G, where A_{ij} is referred to as the weight of the edge between vertex i and vertex j in G. Particularly, if G is unweighted, $\mathrm{A} \in \{0, 1\}^{n \times n}$ and A_{ij} only indicate the existence of the edges. In our work, we only consider the undirected graph, where $\mathrm{A}_{ij} = \mathrm{A}_{ji}$. For any vertex v_i, the degree of v_i is defined as $d(v_i) = \sum_j \mathrm{A}_{ij}$, and $D = \mathrm{diag}(d(v_1), d(v_2), \ldots, d(v_n))$ refers to the degree matrix.

Suppose that $\mathcal{P} = \{P_1, P_2, \ldots, P_L\}$ is a partition of V. Each P_i is called a *community* (aka. module or cluster), representing a group

[1] code is available at: https://github.com/RingBDStack/SE-GSL
[2] A vertex is defined in the graph and a node in the tree.

of vertices with commonality. Due to the grouping nature of a real-world network, each community of the graph can be hierarchically split into multi-level *sub-communities*. Such *hierarchical community* partition (i.e., hierarchical *semantic*) of a graph can be intrinsically abstracted as the encoding tree [18, 20], and each tree node represents a specific community. Take Fig. 1 as an example: at a high abstraction (semantic) level, the entire graph can be categorized as two coarse-grained communities, i.e., teachers (T) and students (S). Students can be identified as sub-communities like S.1 and S.2, as per the class placement scheme.

Graph Structure Learning (GSL). For any given graph G, the goal of GSL [61] is to simultaneously learn an optimal graph structure G^* optimized for a specific downstream task and the corresponding graph representation Z. In general, the objective of GSL can be summarized as $\mathcal{L}_{gsl} = \mathcal{L}_{task}(Z, Y) + \alpha\mathcal{L}_{reg}(Z, G^*, G)$, where $\mathcal{L}_{task}$ refers to a task-specific objective with respect to the learned representation Z and the ground truth Y. $\mathcal{L}_{reg}$ imposes constraints on the learned graph structure and representations, and α is a hyper-parameter.

2.2 Structural Entropy

Different from information entropy (aka. Shannon entropy) that measures the uncertainty of probability distribution in information theory [37], *structural entropy* [18] measures the structural system diversity, e.g., the uncertainty embedded in a graph.

Encoding Tree. Formally, the encoding tree $\mathcal{T}$ of graph $G = (V, E)$ holds the following properties: **(1)** The root node λ in $\mathcal{T}$ has a label $T_\lambda = V$, V represents the set of all vertices in G. **(2)** Each non-root node α has a label $T_\alpha \subset V$. Furthermore, if α is a leaf node, T_α is a singleton with one vertex in V. **(3)** For each non-root node α, its parent node in T is denoted as α^-. **(4)** For each non-leaf node α, its i-th children node is denoted as $\alpha^{\langle i\rangle}$ ordered from left to right as i increases. **(5)** For each non-leaf node α, assuming the number of children α is N, all vertex subset $T_{\alpha^{\langle i\rangle}}$ form a partition of T_α, written as $T_\alpha = \bigcup_{i=1}^{N} T_{\alpha^{\langle i\rangle}}$ and $\bigcap_{i=1}^{N} T_{\alpha^{\langle i\rangle}} = \emptyset$. If the encoding tree's height is restricted to K, we call it *K-level* encoding tree. Entropy measures can be conducted on different encoding trees.

One-dimensional Structural Entropy. In a single-level encoding tree $\mathcal{T}$, its structural entropy degenerates to the unstructured Shannon entropy, which is formulated as:

$$H^1(G) = -\sum_{v\in V} \frac{d_v}{vol(G)} \log_2 \frac{d_v}{vol(G)}, \tag{1}$$

where d_v is the degree of vertex v, and $vol(G)$ is the sum of the degrees of all vertices in G. According to the fundamental research [18], one-dimensional structural entropy $H^1(G)$ measures the uncertainty of vertex set V in G, which is also the upper bound on the amount of information embedded in G.

High-dimensional Structural Entropy. For the encoding tree $\mathcal{T}$, we define high-dimensional structural entropy of G as:

$$H^K(G) = \min_{\forall \mathcal{T}: height(\mathcal{T}) \le K} \{H^{\mathcal{T}}(G)\}, \tag{2}$$

$$H^{\mathcal{T}}(G) = \sum_{\alpha\in\mathcal{T},\alpha\neq\lambda} H^{\mathcal{T}}(G;\alpha) = -\sum_{\alpha\in\mathcal{T},\alpha\neq\lambda} \frac{g_\alpha}{vol(G)} \log_2 \frac{\mathcal{V}_\alpha}{\mathcal{V}_{\alpha^-}}, \tag{3}$$

where g_α is the sum weights of the cut edge set $[T_\alpha, T_\alpha/T_\lambda]$, i.e., all edges connecting vertices inside T_α with vertices outside T_α. $\mathcal{V}_\alpha$ is the sum of degrees of all vertices in T_α. $H^{\mathcal{T}}(G;\alpha)$ is the structural entropy of node α and $H^{\mathcal{T}}(G)$ is the structural entropy of $\mathcal{T}$. $H^K(G)$ is the K-dimensional structural entropy, with the optimal encoding tree of K-level .

3 OUR APPROACH

This section presents the architecture of SE-GSL, then elaborate on how we enhance the graph structure learning by structural entropy-based optimization of the hierarchical encoding tree.

3.1 Overview of SE-GSL

Fig. 2 depicts the overall pipeline. At the core of SE-GSL is the structure optimization procedure that transforms and enhances the graph structure. More specifically, it encompasses multi-stages: graph structure enhancement, hierarchical encoding tree generation, and sampling-based structure reconstruction before an iterative representation optimization.

First, the original topological information is integrated with vertex attributes and the neighborhood in close proximity. Specifically, we devise a similarity-based edge reweighting mechanism and incorporate k-NN graph structuralization to provide auxiliary edge information. The most suitable k is selected under the guidance of the one-dimensional structural entropy maximization strategy (§ 3.2). Upon the enhanced graph, we present a hierarchical abstraction mechanism to further suppress the edge noise and reveal the high-level hierarchical community structure (encoding tree) (§ 3.3). A novel sampling-based approach is designed to build new graph topology from the encoding tree, particularly by restoring the edge connectivity from the tree hierarchy (§ 3.4). The core idea is to weaken the association between high-level communities whilst establishing dense and extensive connections within low-level communities. To this end, we transform the node structural entropy into probability, rejecting the deterministic threshold. Through multi-iterative stochastic sampling, it is more likely to find favorable graph structures for GNNs. Afterward, the rebuilt graph will be fed into the downstream generic GNN encoders. To constantly improve both the node representation and the graph structure, the optimization pipeline is iterated for multiple epochs. The training procedure of SE-GSL is summarized in Appendix A.4.

3.2 Graph Structure Enhancement

To fully incorporate vertex attributes and neighborhood information in the graph structure, we perform feature fusion and edge reweighting so that the topological structure, together with the informative vertex adjacent similarity, can be passed on to the encoding tree generator. To begin with, we calculate the pair-wise similarity matrix $S \in \mathbb{R}^{|V|\times|V|}$ among vertices in graph G. To better depict the linear correlation between two vertex attributes, we take the Pearson correlation coefficient (PCC) as the similarity measure in the experiments, i.e.,

$$S_{ij} = \mathrm{PCC}(x_i, x_j) = \frac{E((x_i - u_i)(x_j - u_j))}{\sigma_i \sigma_j}, \tag{4}$$

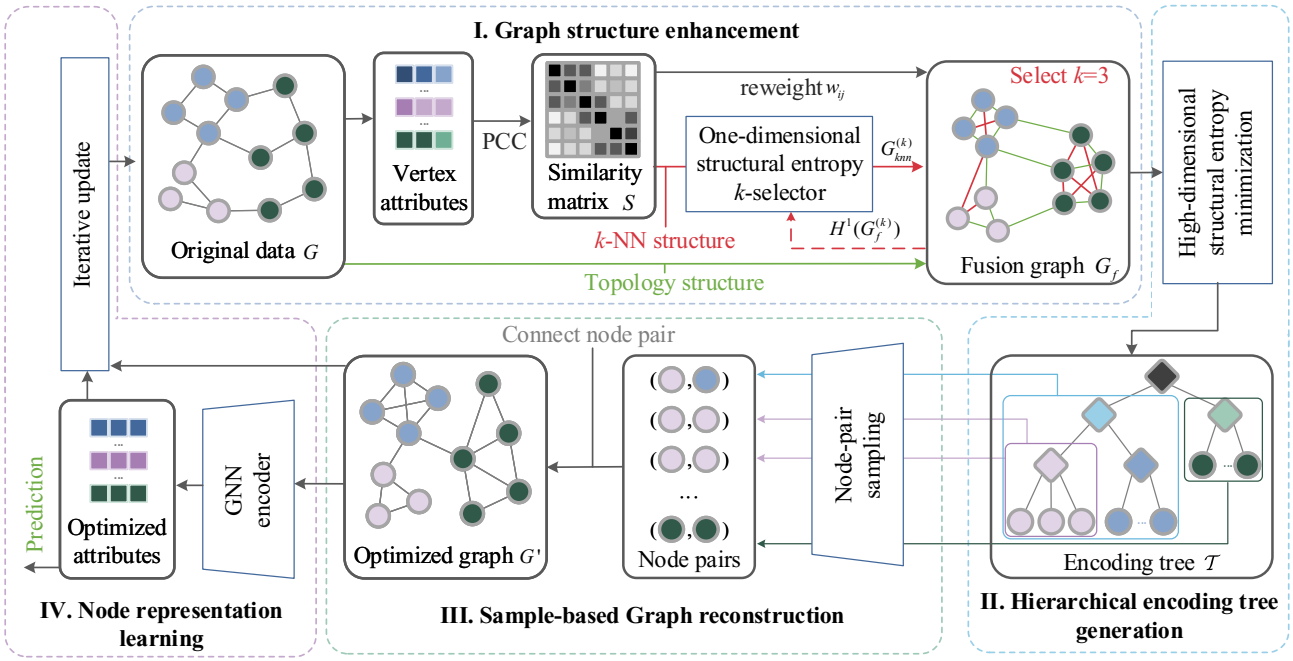

Figure 2: The overall architecture of SE-GSL.

where x_i and $x_j \in \mathbb{R}^{1\times d}$ are the attribute vectors of vertices i and j, respectively. u_i and σ_i denote the mean value and variance of x_i, and $E(\cdot)$ is the dot product function. Based on S, we can intrinsically construct the k-NN graph $G_{knn} = \{V, E_{knn}\}$ where each edge in E_{knn} represents a vertex and its k nearest neighbors (e.g., the edges in red in Fig 2). We fuse G_{knn} and the original G to $G_f = \{V, E_f = E \cup E_{knn}\}$.

A key follow-up step is pinpointing the most suitable number k of nearest neighbors. An excessive value of k would make G_f over-noisy and computationally inefficient, while a small k would result in insufficient information and difficulties in hierarchy extraction. As outlined in § 2.2, a larger one-dimensional structural entropy indicates more information that G_f can potentially hold. Hence, we aim to maximize the one-dimensional structural entropy $H^1(G_f)$ to guide the selection of k for larger encoding information. In practice, we gradually increase the integer parameter k, generate the corresponding $G_f^{(k)}$ and compute $H^1(G_f^{(k)})$. Observably, when k reaches a threshold k_m, $H^1(G_f^{(k)})$ comes into a plateau without noticeable increase. This motivates us to regard this critical point k_m as the target parameter. The k-selector algorithm is depicted in Appendix A.5. In addition, the edge e_{ij} between v_i and v_j is reweighted as:

$$\omega_{ij} = S_{ij} + M, \quad M = \frac{1}{2|V|} \cdot \frac{1}{|E|} \sum_{1<i,j<n} S_{ij}, \tag{5}$$

where M is a modification factor that amplifies the trivial edge weights and thus makes the k-selector more sensitive to noises.

3.3 Hierarchical Encoding Tree Generation

Our methodology of abstracting the fused graph into a hierarchy is inspired by the structural entropy theory [18, 20]. We intend to minimize the graph uncertainty (i.e., edge noises) and maximize the knowledge embedded (e.g., optimal partition) in the high-dimensional hierarchy. Correspondingly, the objective of structural entropy minimization is to find out an encoding tree $\mathcal{T}$ that minimizes $H^{\mathcal{T}}(G_f)$ defined in Eq. 3. Due to the difficulty in graph semantic complexity quantification, we restrict the optimization objective to the K-level tree with a hyperparameter K. The optimal K-dimensional encoding tree is represented as:

$$\mathcal{T}^* = \underset{\forall \mathcal{T}: height(\mathcal{T}) \le K}{\arg\min} (H^{\mathcal{T}}(G)). \tag{6}$$

To address this optimization problem, we design a greedy-based heuristic algorithm to approximate $H^K(G)$. To assist the greedy heuristic, we define two basic operators:

Definition 1. Combining operator: Given an encoding tree $\mathcal{T}$ for $G = (V, E)$, let α and β be two nodes in $\mathcal{T}$ sharing the same parent γ. The combining operator $\mathrm{CB}_{\mathcal{T}}(\alpha, \beta)$ updates the encoding tree as: $\gamma \leftarrow \delta^-; \delta \leftarrow \alpha^-; \delta \leftarrow \beta^-$. A new node δ is inserted between γ and its children α, β.

Definition 2. Lifting operator: Given an encoding tree $\mathcal{T}$ for $G = (V, E)$, let α, β and γ be the nodes in $\mathcal{T}$, satisfying $\beta^- = \gamma$ and $\alpha^- = \beta$. The lifting operator $\mathrm{LF}_{\mathcal{T}}(\alpha, \beta)$ updates the encoding tree as: $\gamma \leftarrow \alpha^-$; IF :$T_\beta = \varnothing$, THEN :drop($\beta$). The subtree rooted at α is lifted by placing itself as γ's child. If no more children exist after lifting, β will be deleted from $\mathcal{T}$.

In light of the high-dimensional structural entropy minimization principle [20], we first build a full-height binary encoding tree by greedily performing the combining operations. Two children of the root are combined to form a new partition iteratively until the structural entropy is no longer reduced. To satisfy the height restriction, we further squeeze the encoding tree by lifting subtrees to higher levels. To do so, we select and conduct lifting operations between a non-root node and its parent node that can reduce the structural entropy to the maximum. This will be repeated until the encoding tree height is less than K and the structural entropy can no longer be decreased. Eventually, we obtain an encoding tree with a specific height K with minimal structural entropy. The pseudo-code is detailed in Appendix A.6.

3.4 Sample-based Graph Reconstruction

The subsequent step is to restore the topological structure from the hierarchy whilst guaranteeing the established hierarchical semantics in optimal encoding tree $\mathcal{T}^*$. The key to graph reconstruction is determining which edges to augment or weaken. Intuitively, the nodes in real-life graphs in different communities tend to have different labels. The work [63] has demonstrated the effectiveness of strengthening intra-cluster edges and reducing inter-cluster edges in a cluster-awareness approach to refine the graph topology. However, for hierarchical communities, simply removing cross-community edges will undermine the integrity of the higher-level community. Adding edges within communities could also incur additional edge noises to lower-level partitioning.

We optimize the graph structure with community preservation by investigating the structural entropy of *deduction* between two interrelated nodes as the criterion of edge reconstruction:

Definition 3. Structural entropy of deduction: Let $\mathcal{T}$ be an encoding tree of G. We define the structural entropy of the deduction from non-leaf node λ to its descendant α as:

$$H^{\mathcal{T}}(G;(\lambda,\alpha]) = \sum_{\beta, T_\alpha \subseteq T_\beta \subset T_\lambda} H^{\mathcal{T}}(G;\beta). \tag{7}$$

This node structure entropy definition exploits the hierarchical structure of the encoding tree and offers a generic measure of top-down deduction to determine a community or vertex in the graph.

From the viewpoint of message passing, vertices with higher structural entropy of deduction produce more diversity and uncertainty and thus require more supervisory information. Therefore, such vertices need expanded connection fields during the graph reconstruction to aggregate more information via extensive edges. To achieve this, we propose an analog sampling-based graph reconstruction method. The idea is to explore the node pairs at the leaf node level (the lowest semantic level) and stochastically generate an edge for a given pair of nodes with a certain probability associated with the deduction structural entropy.

Specifically, for a given $\mathcal{T}$, assume the node δ has a set of child nodes $\{\delta^{\langle 1\rangle}, \delta^{\langle 2\rangle}, \ldots, \delta^{\langle n\rangle}\}$. The probability of the child $\delta^{\langle i\rangle}$ is defined as: $P(\delta^{\langle i\rangle}) = \sigma_\delta(H^{\mathcal{T}}(G_f;(\lambda,\delta^{\langle i\rangle}])))$, where λ is the root of $\mathcal{T}$ and $\sigma_\delta(\cdot)$ represents a distribution function. Take softmax function as an example, the probability of $\delta^{\langle i\rangle}$ can be calculated as:

$$P(\delta^{\langle i\rangle}) = \frac{\exp(H^{\mathcal{T}}(G_f;(\lambda,\delta^{\langle i\rangle}])))}{\sum_{j=1}^{n}\exp(H^{\mathcal{T}}(G_f;(\lambda,\delta^{\langle j\rangle}])))}. \tag{8}$$

The probability of a non-root node can be acquired recursively. To generate new edges, we sample leaf node pairs by a top-down approach with a single sampling flow as follows:
(1) For the encoding tree (or subtree) with root node δ, two different child nodes $\delta^{\langle i\rangle}$ and $\delta^{\langle j\rangle}$ are selected by sampling according to $P(\delta^{\langle i\rangle})$ and $P(\delta^{\langle j\rangle})$. Let $\delta_1 \leftarrow \delta^{\langle i\rangle}$ and $\delta_2 \leftarrow \delta^{\langle j\rangle}$ **(2)** If δ_1 is a non-leaf node, we perform sampling once on the subtree rooted at δ_1 to get $\delta_1^{\langle i\rangle}$, then update $\delta_1 \leftarrow \delta_1^{\langle i\rangle}$. The same is operated on δ_2. **(3)** After recursively performing step (2), we sample two leaf nodes δ_1 and δ_2, while adding the edge connecting vertex $v_1 = T_{\delta_1}$ and $v2 = T_{\delta_2}$ into the edge set E' of graph G'. To establish extensive connections at all levels, multiple samplings are performed on all encoding subtrees. For each subtree rooted at δ, we conduct independent samplings for $\theta \times n$ times, where n is the number of δ's children, and θ is a hyperparameter that positively correlated with the density of reconstructed graph. For simplicity, we adopt a uniform θ for all subtrees. Separately setting and tuning θ of each semantic level for precise control is also feasible.

3.5 Time Complexity of SE-GSL

The overall time complexity is $O(n^2 + n + n\log^2 n)$, in which n is the number of nodes. Separately, in § 3.2, the time complexity of calculating similarity matrix is $O(n^2)$ and of k-selector is $O(n)$. According to [18], the optimization of a high-dimensional encoding tree in § 3.3 costs the time complexity of $O(n\log^2 n)$. As for the sampling process in § 3.4, the time complexity can be proved as $O(2n)$. We report the time cost of SE-GSL in Appendix A.3.

4 EXPERIMENTAL SETUP

Software and Hardware. All experiments are conducted on a Linux server with GPU (NVIDIA RTX A6000) and CPU (Intel i9-10980XE), using PyTorch 1.12 and Python 3.9.
Datasets. We experiment on nine open graph benchmark datasets, including three citation networks (i.e., Cora, Citeseer, and Pubmed), two social networks (i.e., PT and TW), three website networks from WebKB (i.e., Cornell, Texas, and Wisconsin), and a co-occurrence network. Their statistics are summarized in Appendix A.2.
Baseline and backbone models. We compare SE-GSL with baselines including general GNNs (i.e., GCN, GAT, GCNII, Grand) and graph learning/high-order neighborhood awareness methods (i.e. MixHop, Dropedge, Geom-GCN, GDC, GEN, H_2GCN). Four classic GNNs (GCN, GAT, GraphSAGE, APPNP) are chosen as the backbone encoder that SE-GSL works upon. Details are in Appendix A.3.
Parameter settings. For SE-GSL with various backbones, we uniformly adopt two-layer GNN encoders. To avoid over-fitting, We adopt ReLU (ELU for GAT) as the activation function and apply a dropout layer with a dropout rate of 0.5. The training iteration is set to 10. The embedding dimension d is chosen from $\{8, 16, 32, 48, 64\}$, while the height of the encoding tree K is searched among $\{2, 3, 4\}$, and the hyperparameter θ in 3.4 is tuned among

Table 1: Classification Accuracy (%) comparison, with improvement range of SE-GSL against the baselines. The best results are bolded and the second-bests are underlined. Green denotes the outperformance percentage, while yellow denotes underperformance.

Method	Cora	Citeseer	Pubmed	PT	TW	Actor	Cornell	Texas	Wisconsin
GCN	$87.26_{\pm 0.63}$	$76.22_{\pm 0.71}$	$87.46_{\pm 0.12}$	$67.62_{\pm 0.21}$	$62.46_{\pm 1.94}$	$27.65_{\pm 0.55}$	$49.19_{\pm 1.80}$	$57.30_{\pm 2.86}$	$48.57_{\pm 4.08}$
GAT	$87.52_{\pm 0.69}$	$76.04_{\pm 0.78}$	$86.61_{\pm 0.15}$	$68.76_{\pm 1.01}$	$61.68_{\pm 1.20}$	$27.77_{\pm 0.59}$	$57.09_{\pm 6.32}$	$58.10_{\pm 4.14}$	$51.34_{\pm 4.78}$
GCNII	$87.57_{\pm 0.87}$	$75.47_{\pm 1.01}$	$\underline{88.64_{\pm 0.23}}$	$68.93_{\pm 0.93}$	$65.17_{\pm 0.47}$	$30.66_{\pm 0.66}$	$58.76_{\pm 7.11}$	$55.36_{\pm 6.45}$	$51.96_{\pm 4.36}$
Grand	$\mathbf{87.93}_{\pm 0.71}$	$77.59_{\pm 0.85}$	$86.14_{\pm 0.98}$	$69.80_{\pm 0.75}$	$\underline{66.79_{\pm 0.22}}$	$29.80_{\pm 0.60}$	$57.21_{\pm 2.48}$	$56.56_{\pm 1.53}$	$52.94_{\pm 3.36}$
Mixhop	$85.71_{\pm 0.85}$	$75.94_{\pm 1.00}$	$87.31_{\pm 0.44}$	$69.48_{\pm 0.30}$	$66.34_{\pm 0.22}$	$33.72_{\pm 0.76}$	$64.47_{\pm 4.78}$	$63.16_{\pm 6.28}$	$72.12_{\pm 3.34}$
Dropedge	$86.32_{\pm 1.09}$	$76.12_{\pm 1.32}$	$87.58_{\pm 0.34}$	$68.49_{\pm 0.91}$	$65.24_{\pm 1.45}$	$30.10_{\pm 0.71}$	$58.94_{\pm 5.95}$	$59.20_{\pm 5.43}$	$60.45_{\pm 4.48}$
Geom-GCN-P	84.93	75.14	88.09	-	-	31.63	60.81	67.57	64.12
Geom-GCN-S	85.27	74.71	84.75	-	-	30.30	55.68	59.73	56.67
GDC	$87.17_{\pm 0.36}$	$76.13_{\pm 0.53}$	$88.08_{\pm 0.16}$	$66.14_{\pm 0.54}$	$64.14_{\pm 1.40}$	$28.74_{\pm 0.76}$	$59.46_{\pm 4.35}$	$56.42_{\pm 3.99}$	$48.30_{\pm 4.29}$
GEN	$\underline{87.84_{\pm 0.69}}$	$\mathbf{78.77}_{\pm 0.88}$	$86.13_{\pm 0.41}$	$\underline{71.62_{\pm 0.78}}$	$65.16_{\pm 0.77}$	$\mathbf{36.69}_{\pm 1.02}$	$65.57_{\pm 6.74}$	$73.38_{\pm 6.65}$	$54.90_{\pm 4.73}$
H_2GCN-2	$87.81_{\pm 1.35}$	$76.88_{\pm 1.77}$	$\mathbf{89.59}_{\pm 0.33}$	$68.15_{\pm 0.30}$	$63.33_{\pm 0.77}$	$35.62_{\pm 1.30}$	$\mathbf{82.16}_{\pm 6.00}$	$\underline{82.16_{\pm 5.28}}$	$\underline{85.88_{\pm 4.22}}$
SE-GSL	$\mathbf{87.93}_{\pm 1.24}$	$\underline{77.63_{\pm 1.65}}$	$88.16_{\pm 0.76}$	$\mathbf{71.91}_{\pm 0.66}$	$\mathbf{66.99}_{\pm 0.93}$	$\underline{36.34_{\pm 2.07}}$	$\underline{75.21_{\pm 5.54}}$	$\mathbf{82.49}_{\pm 4.80}$	$\mathbf{86.27}_{\pm 4.32}$
Improvement	0.00∼3.00	-1.14∼2.92	-1.43∼3.41	0.29∼5.77	0.20∼5.31	-0.35∼8.69	-6.95∼26.02	0.33∼27.13	0.39∼37.97

$\{0.5, 1, 3, 5, 10, 30\}$. We adopt the scheme of data split in Geom-GCN [28] and H_2GCN [60] for all experiments – nodes are randomly divided into the train, validation, and test sets, which take up 48%, 32%, 20%, respectively. In each iteration, the GNN encoder optimization is carried out for 200 epochs, using the Adam optimizer, with an initial learning rate of 0.01 and a weight decay of $5e-4$. The model with the highest accuracy on validation sets is used for further testing and reporting.

Table 2: Classification accuracy(%) of SE-GSL and corresponding backbones. Wisc. is short for Wisconsin.

Method	Actor	TW	Texas	Wisc.	Improvement
SE-GSL$_{GCN}$	35.03	66.88	75.68	79.61	↑ 5.20∼31.04
SE-GSL$_{SAGE}$	36.20	66.92	**82.49**	**86.27**	↑ 0.25∼6.79
SE-GSL$_{GAT}$	32.46	63.57	74.59	78.82	↑ 4.69∼27.48
SE-GSL$_{APPNP}$	**36.34**	**66.99**	81.28	83.14	↑ 2.01∼12.16

5 RESULTS AND ANALYSIS

In this section, we demonstrate the efficacy of SE-GSL on semi-supervised node classification (§ 5.1, followed by micro-benchmarks that investigate the detailed effect of the submodules on the overall performance and validate the robustness of SE-GSL when tackling random perturbations (§ 5.2). For better interpretation, we visualize the change of structural entropy and graph topology (§ 5.3).

5.1 Node Classification

5.1.1 Comparison with baselines. We compare the node classification performance of SE-GSL with ten baseline methods on nine benchmark datasets. Table 1 shows the average accuracy and the standard deviation. Note that the results of H_2GCN (except PT and TW) and Geom-GCN are from the reported value in original papers (- for not reported), while the rest are obtained based on the execution of the source code provided by their authors under our experimental settings. Our observations are three-fold:

(1) SE-GSL achieves optimal results on 5 datasets, runner-up results on 8 datasets, and advanced results on all datasets. The accuracy can be improved up to 3.41% on Pubmed, 3.00% on Cora, and 2.92% on Citeseer compared to the baselines. This indicates that our design can effectively capture the inherent and deep structure of the graph and hence the classification improvement.

(2) SE-GSL shows significant improvement on the datasets with heteropily graphs, e.g., up to 37.97% and 27.13% improvement against Wisconsin and Texas datasets, respectively. This demonstrates the importance of the graph structure enhancement that can contribute to a more informative and robust node representation.

(3) While all GNN methods can achieve satisfactory results on citation networks, the graph learning/high-order neighborhood awareness frameworks substantially outperform others on the WebKB datasets and the actor co-occurrence networks, which is highly disassortative. This is because these methods optimize local neighborhoods for better information aggregation. Our method is one of the top performers among them due to the explicit exploitation of the global structure information in the graph hierarchical semantics.

5.1.2 Comparison base on different backbones. Table 2 shows the mean classification accuracy of SE-GSL with different backbone encoders. Observably, SE-GSL upon GCN and GAT overwhelmingly outperforms its backbone model, with an accuracy improvement of up to 31.04% and 27.48%, respectively. This indicates the iterative mechanism in the SE-GSL pipeline can alternately optimize the node representation and graph structure. We also notice that despite the lower improvement, SE-GSL variants based on GraphSAGE and APPNP perform relatively better compared to those on GCN and GAT. This is most likely due to the backbone model itself being more adapted to handle disassortative settings on graphs.

5.2 Micro-benchmarking

5.2.1 Effectiveness of k-selector. This subsection evaluates how the one-dimensional structural entropy guides the k-selector in § 3.2. Table 3 showcases the selected parameter k in each iteration with

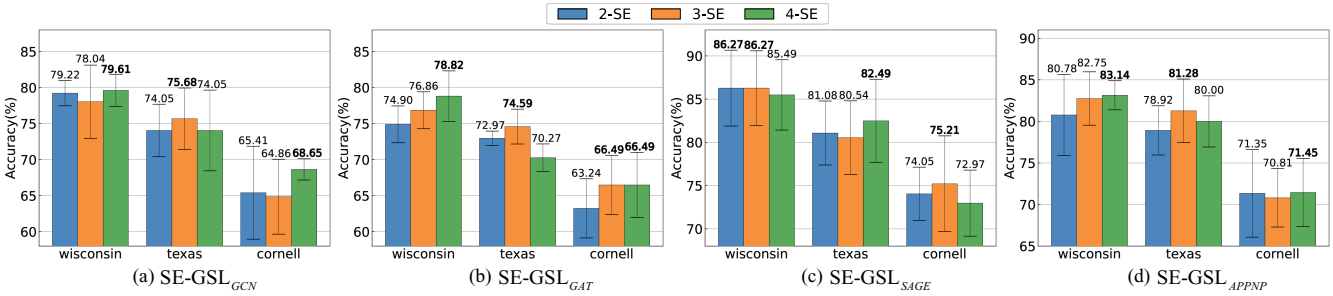

Figure 3: Results of SE-GSL with different encoding tree heights.

Table 3: The k selection for each iteration in structural optimization. Bolds represent the k selection when the accuracy reaches maximum.

Iteration	1	2	3	4	5	6	7	8	9
Cora	22	**22**	19	22	21	22	20	21	20
Actor	23	15	15	15	14	15	14	**14**	15
TW	50	16	16	**17**	15	17	27	16	16
Wisconsin	21	16	**11**	16	14	13	16	13	11
Texas	21	13	13	**13**	13	10	14	10	14

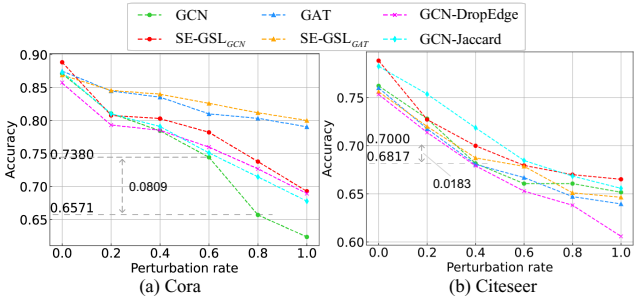

Figure 4: Robustness of SE-GSL against random noises.

SE-GSL$_{GCN}$. Noticeably, as the iterative optimization proceeds, the optimal parameter k converges to a certain range, indicating the gradual stabilization of the graph structure and node representation. The disparity of parameter k among different datasets also demonstrates the necessity of customizing k in different cases rather than using k as a static hyperparameter.

5.2.2 Impact of the encoding tree's height K. We evaluate all four variants of SE-GSL on the website network datasets, and the encoding tree height K involved in § 3.3 varies from 2 to 4. As shown in Fig. 3, there is a huge variation in the optimal tree heights among different datasets. For example, in the variants based on GAT, GCN, and APPNP, the best results can be targeted at $K = 3$ in Texas and at $K = 4$ in Cornell and Wisconsin. By contrast, in SE-GSL$_{SAGE}$, $K = 2$ can enable the best accuracy of 86.27%. This weak correlation between the best K and the model performance is worth investigating further, which will be left as future work.

5.2.3 Sensitivity to perturbations. We introduce random edge noises into Cora and Citeseer, with perturbation rates of 0.2, 0.4, 0.6, 0.8, and 1. As shown in Fig. 4(a), SE-GSL outperforms baselines in both GCN and GAT cases under most noise settings. For instance, SE-GSL$_{GCN}$ achieves up to 8.09% improvement against the native GCN when the perturbation rate is 0.8; by contrast, improvements by GCN-Jaccard and GCN-DropEdge are merely 6.99% and 5.77%, respectively. A similar phenomenon is observed for most cases in the Citeseer dataset (Fig. 4(b)), despite an exception when compared against GCN-Jaccard. Nevertheless, our approach is still competitive and even better than GCN-Jaccard at a high perturbation rate.

5.3 Interpretation of Structure Evolution

5.3.1 Structural entropy variations analysis. We evaluate how the structural entropy changes during the training of SE-GSL$_{GAT}$ with 2-dimensional structural entropy on WebKB datasets. For comparison, we visualize the entropy changes on Wisconsin without the structure learning. In the experiment setting, both the graph structure and the encoding tree are updated once at each iteration (i.e., 200 GNN epochs), and within one iteration, the structural entropy is only affected by edge weights determined by the similarity matrix. For comparison, we normalize the structural entropy by $\frac{H^{\mathcal{T}}(G)}{H^{1}(G)}$.

As shown in Fig. 5(a)-(c), as the accuracy goes up, the normalized structural entropy constantly decreases during the iterative graph reconstruction, reaching the minimums of 0.7408 in Texas, 0.7245 in Cornell, and 0.7344 in Wisconsin. This means the increasing determinism of the overall graph structure and the reduced amount of information required to determine a vertex. Interestingly, if our graph reconstruction mechanism is disabled (as shown in Fig. 5(d)), the normalized structural entropy keeps rising from 0.7878, compared with Fig. 5(c). Accordingly, the final accuracy will even converge to 55.34%, a much lower level.

Such a comparison also provides a feasible explanation for the rising trend of the normalized structural entropy within every single iteration. This stems from the smoothing effect during the GNN training. As the node representation tends to be homogenized, the graph structure will be gradually smoothed, leading to a decrease in the one-dimensional structural entropy thus the normalized structural entropy increases.

5.3.2 Visualization. Fig. 6 visualizes the topology of the original Cora and Citeseer graph and of the 2nd and 5th iterations. The vertex color indicates the class it belongs to, and the layout denotes connecting relations. Edges are hidden for clarity. As the iteration

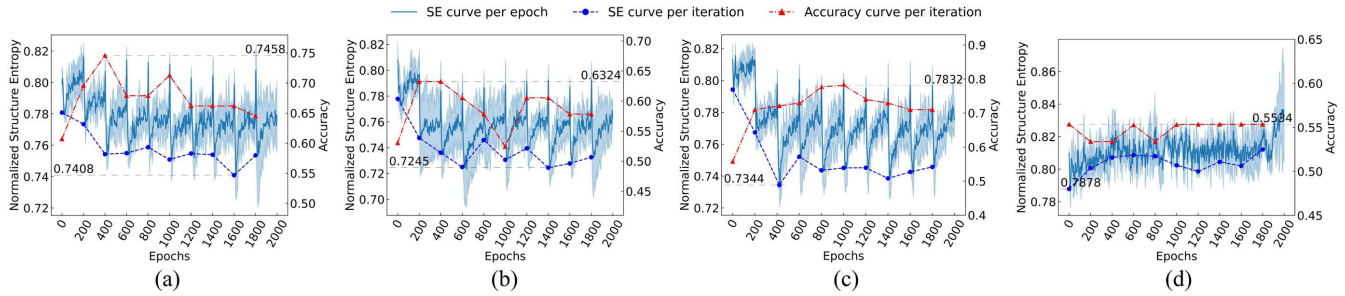

Figure 5: The normalized structural entropy changes during the training of SE-GSL$_{GAT}$ with 2-dimensional structural entropy on (a) Texas, (b) Cornell, and (c) Wisconsin. The structure is iterated every 200 epochs. By comparison, (d) shows the entropy changes on Wisconsin without the graph reconstruction strategy.

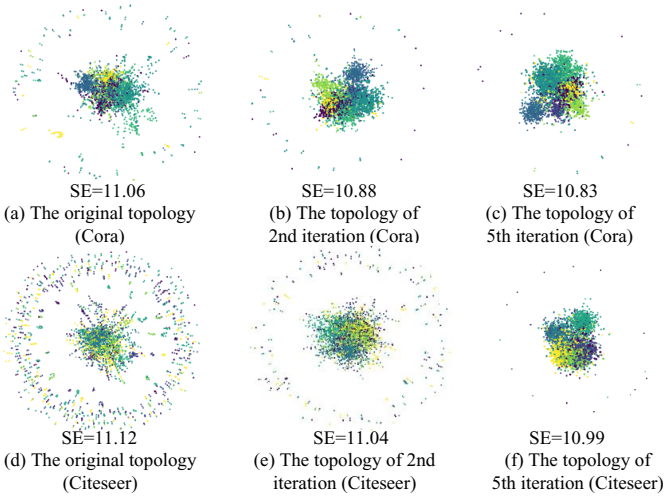

Figure 6: The visualized evolution of the graph structure on Cora (a,b,c) and Citeseer (d,e,f). The corresponding Structural Entropy (SE) is also shown.

continues, much clearer clustering manifests – few outliers and more concentrated clusters. Vertices with the same label are more tightly connected due to the iterative graph reconstruction scheme. This improvement hugely facilitates the interpretability of the GSL and the node representation models.

pan2021information

6 RELATED WORK

Graph structure learning and neighborhood optimization. The performance of GNNs is heavily dependent on task-relevant links and neighborhoods. Graph structure learning explicitly learns and adjusts the graph topology, and our SE-GSL is one of them. GDC [10] reconnects graphs through graph diffusion to aggregate from multi-hop neighborhoods. Dropedge [34], Neuralsparse [58] contribute to graph denoising via edge-dropping strategy while failing to renew overall structures. LDS-GNN [8] models edges by sampling graphs from the Bernoulli distribution. Meanwhile, we consider linking the structural entropy, which is more expressive of graph topology, to the sampling probability. GEN [43], IDGL [5] explore the structure from the node attribute space by the k-NN method. Differently, instead of directly using attribute similarity, we regenerate edges from the hierarchical abstraction of graphs to avoid inappropriate metrics. Besides adjusting the graph structure, methods to optimize aggregation are proposed with results on heterophily graphs. MixHop [1] learns the aggregation parameters for neighborhoods of different hops through a mixing network, while H_2GCN [60] identifies higher-order neighbor-embedding separation and intermediate representation combination, for adapting to heterophily graphs. Geom-GCN [28] aggregates messages over both the original graph and latent space by a designed geometric scheme.

Structural entropy with neural networks. Structural information principles [18], defining encoding trees and structural entropy, were first used in bioinformatic structure analysis [19, 20]. Existing work mainly focuses on network analysis, node clustering and community detection[21, 24, 27]. As an advanced theory on graphs and hierarchical structure, structural information theory has great potential in combination with neural networks. SR-MARL [56] applies structural information principles to hierarchical role discovery in multi-agent reinforcement learning, thereby boosting agent network optimization. SEP [47] provides a graph pooling scheme based on optimal encoding trees to address local structure damage and suboptimal problem. It essentially uses structural entropy minimization for a multiple-layer coarsened graph. MinGE [26] and MEDE [52] estimate the node embedding dimension of GNNs via structural entropy, which introduces both attribute entropy and structure entropy as objective. Although these works exploit structural entropy to mine the latent settings of neural networks and GNNs, how to incorporate this theory in the optimization process is still understudied, and we are among the first attempts.

7 CONCLUSION

To cope with edge perturbations in graphs with heterophily, this paper proposes a novel graph structure learning framework SE-GSL that considers the structural entropy theory in graph structure optimization. We design a structure enhancement module guided by the one-dimensional structural entropy maximization strategy to extend the information embedded in the topology. To capture the hierarchical semantics of graphs, high-dimensional structural entropy minimization is performed for optimal encoding trees. We propose

a node sampling technique on the encoding tree to restore the most appropriate edge connections at different community levels, taking into account the deduction structural entropy distribution. In the future, we plan to combine delicate loss functions with structural entropy so that the knowledge in encoding trees can be converted into gradient information, which will further allow for end-to-end structure optimization.

ACKNOWLEDGMENTS

This paper was supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grant 62002007, S&T Program of Hebei through grant 20310101D, Natural Science Foundation of Beijing Municipality through grant 4222030, CCF-DiDi GAIA Collaborative Research Funds for Young Scholars, the Fundamental Research Funds for the Central Universities, Xiaomi Young Scholar Funds for Beihang University, and in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.

[2] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the International Conference on Machine Learning*. PMLR, 874–883.

[3] Aleksandar Bojchevski and Stephan Günnemann. 2019. Certifiable robustness to graph perturbations. *Advances in Neural Information Processing Systems* 32 (2019).

[4] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1725–1735.

[5] Yu Chen, Lingfei Wu, and Mohammed Zaki. 2020. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in Neural Information Processing Systems* 33 (2020), 19314–19326.

[6] Enyan Dai, Charu Aggarwal, and Suhang Wang. 2021. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 227–236.

[7] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in Neural Information Processing Systems* 33 (2020), 22092–22103.

[8] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning discrete structures for graph neural networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1972–1982.

[9] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proceedings of the International Conference on Learning Representations*.

[10] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*. 13366–13378.

[11] Lise Getoor. 2005. Link-based classification. In *Advanced methods for knowledge discovery from complex data*. Springer, 189–207.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1263–1272.

[13] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 80–89.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems* 30 (2017).

[15] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[16] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[17] Xu Keyulu, Hu Weihua, Leskovec Jure, and Stefanie Jegelka. 2019. How powerful are graph neural networks. *Proceedings of the International Conference on Learning Representations*.

[18] Angsheng Li and Yicheng Pan. 2016. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory* 62, 6 (2016), 3290–3339.

[19] Angsheng Li, Xianchen Yin, and Yicheng Pan. 2016. Three-dimensional gene map of cancer cell types: Structural entropy minimisation principle for defining tumour subtypes. *Scientific reports* 6, 1 (2016), 1–26.

[20] Angsheng Li, Xianchen Yin, Bingxiang Xu, Danyang Wang, Jimin Han, Yi Wei, Yun Deng, Ying Xiong, and Zhihua Zhang. 2018. Decoding topologically associating domains with ultra-low resolution Hi-C data by graph structural entropy. *Nature communications* 9, 1 (2018), 1–12.

[21] Angsheng Li, Xiaohui Zhang, and Yicheng Pan. 2017. Resistance maximization principle for defending networks against virus attack. *Physica A: Statistical Mechanics and its Applications* 466 (2017), 211–223.

[22] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable Representations Make A Stronger Defender: Unsupervised Structure Refinement for Robust GNN. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 925–935.

[23] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. 2018. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[24] Yiwei Liu, Jiamou Liu, Zijian Zhang, Liehuang Zhu, and Angsheng Li. 2019. REM: From structural entropy to community structure deception. *Advances in Neural Information Processing Systems* 32 (2019).

[25] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*. 779–787.

[26] Gongxu Luo, Jianxin Li, Jianlin Su, Hao Peng, Carl Yang, Lichao Sun, Philip S Yu, and Lifang He. 2021. Graph entropy guided node embedding dimension selection for graph neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. 2767–2774.

[27] Yicheng Pan, Feng Zheng, and Bingchen Fan. 2021. An Information-theoretic Perspective of Hierarchical Clustering. *arXiv preprint arXiv:2108.06036* (2021).

[28] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

[29] Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S. Yu. 2019. Fine-Grained Event Categorization with Heterogeneous Graph Convolutional Networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 3238–3245.

[30] Hao Peng, Jianxin Li, Qiran Gong, Senzhang Wang, Yuanxin Ning, and Lifang He. 2020. Motif-Matching Based Subgraph-Level Attentional Convolutional Network for Graph Classification. In *Proceedings of 34th AAAI Conference on Artificial Intelligence*.

[31] Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. 2021. Reinforced Neighborhood Selection Guided Multi-Relational Graph Neural Networks. *ACM Trans. Inf. Syst.* 40, 4 (2021), 1–46.

[32] Hao Peng, Ruitong Zhang, Shaoning Li, Yuwei Cao, Shirui Pan, and Philip S. Yu. 2023. Reinforced, Incremental and Cross-Lingual Event Detection From Social Messages. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 1 (2023), 980–998.

[33] Sun Qingyun, Li Jianxin, Peng Hao, Wu Jia, Fu Xingcheng, Ji Cheng, and Yu Philip S. 2022. Graph Structure Learning with Variational Information Bottleneck. In *Proceedings of 36th AAAI Conference on Artificial Intelligence*.

[34] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *Proceedings of the International Conference on Learning Representations*.

[35] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.

[36] Benedek Rozemberczki and Rik Sarkar. 2021. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *arXiv preprint arXiv:2101.03091* (2021).

[37] Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[38] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S Yu Philip, Lifang He, and Bo Li. 2022. Adversarial Attack and Defense on Graph Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering* (2022), 1–20.

[39] Qingyun Sun, Jianxin Li, Haonan Yuan, Xingcheng Fu, Hao Peng, Cheng Ji, Qian Li, and Philip S. Yu. 2022. Position-Aware Structure Learning for Graph Topology-Imbalance by Relieving Under-Reaching and Over-Squashing. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. Association for Computing Machinery, 1848–1857.

[40] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 807–816.

[41] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *Proceedings of the International Conference on Learning Representations* (2017).

[42] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315* (2019).

[43] Ruijia Wang, Shuai Mou, Xiao Wang, Wanpeng Xiao, Qi Ju, Chuan Shi, and Xing Xie. 2021. Graph structure estimation neural networks. In *Proceedings of the Web Conference 2021*. 342–353.

[44] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 1243–1253.

[45] Max Welling and Thomas N Kipf. 2016. Semi-supervised classification with graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[46] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 4816–4823.

[47] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. 2022. Structural entropy guided graph hierarchical pooling. In *Proceedings of the International Conference on Machine Learning*. PMLR, 24017–24030.

[48] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph information bottleneck. *Advances in Neural Information Processing Systems* 33 (2020), 20437–20448.

[49] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2020), 4–24.

[50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *Proceedings of the International Conference on Learning Representations*.

[51] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the International Conference on Machine Learning*. PMLR, 40–48.

[52] Zhengyu Yang, Ge Zhang, Jia Wu, Hao Peng, Xue Shan, Jian Yang, Li Angsheng, Jianlin Su, and Quan Z. Sheng. 2023. Minimum Entropy Principle Guided Graph Neural Networks. In *Proceedings of the ACM International WSDM Conference*.

[53] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[54] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems* 31 (2018).

[55] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5708–5717.

[56] Xianghua Zeng, Hao Peng, and Angsheng Li. 2023. Effective and Stable Role-based Multi-Agent Collaboration by Structural Information Principles. In *Proceedings of 37th AAAI Conference on Artificial Intelligence*.

[57] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems* 33 (2020), 9263–9275.

[58] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *Proceedings of the International Conference on Machine Learning*. PMLR, 11458–11468.

[59] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81.

[60] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.

[61] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036* (2021).

[62] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*. 2069–2080.

[63] Yanqiao Zhu, Yichen Xu, Feng Yu, Shu Wu, and Liang Wang. 2020. Cagnn: Cluster-aware graph neural networks for unsupervised graph representation learning. *arXiv preprint arXiv:2009.01674* (2020).

A APPENDIX

A.1 Glossary of notations

In Table 4, we summarize the notations used in our work.

Table 4: Glossary of Notations.

Notation	Description
$G; A; S$	Graph; Adjacency matrix; Similarity matrix.
$v; e; x$	Vertex; Edge; Vertex attribute.
$V; E; X$	Vertex set; Edge set; Attribute set.
$\|V\|; \|E\|$	The number of vertices and edges.
$\mathcal{P}; P_i$	The partition of V; A community.
$D; d(v_i)$	The degree matrix; The degree of vertex v_i.
e_{ij}	The edge between v_i and v_j.
w_{ij}	The weight of edge e_{ij}.
$vol(G)$	The volume of graph G, i.e., degree sum in G.
$G_{knn}^{(k)}$	The k-NN graph with parameter k.
G_f	Fusion graph.
$G_f^{(k)}$	The fusion graph with parameter k.
$\mathcal{T}$	Encoding tree.
$\mathcal{T}^*$	The optimal encoding tree.
λ	The root node of an encoding tree.
α	A non-root node of an encoding tree.
α^-	The parent node of α.
$\alpha^{\langle i \rangle}$	the i-th child of α.
T_λ	The label of λ, i.e., node set V.
T_α	The label of α, i.e., a subset of V.
$\mathcal{V}_\alpha$	Volume of graph G.
g_a	the sum weights of cut edge set $[T_\alpha, T_\alpha/T_\lambda]$.
$N(\mathcal{T})$	The number of non-root node in $\mathcal{T}$.
$H^{\mathcal{T}}(G)$	Structural entropy of G under $\mathcal{T}$.
$H^K(G)$	K-dimensional structural entropy.
$H^1(G)$	One-dimensional structural entropy.
$H^{\mathcal{T}}(G; \alpha)$	Structural entropy of node α in $\mathcal{T}$.
$H^{\mathcal{T}}(G; (\lambda, \alpha])$	Structural entropy of a deduction from λ to α.

A.2 Dataset and Time Costs of SE-GSL

Our framework SE-GSL is evaluated on nine graph datasets. the statistics of these datasets are shown in Table 5. The time costs of SE-GSL on all datasets are shown in Table 6.

Table 5: Statistics of benchmark datasets.

Dataset	Nodes	Edges	Classes	Features	homophily
Cora	2708	5429	7	1433	0.83
Citeseer	3327	4732	6	3703	0.71
Pubmed	19717	44338	3	500	0.79
PT	1912	31299	2	3169	0.59
TW	2772	63462	2	3169	0.55
Actor	7600	33544	5	931	0.24
Cornell	183	295	5	1703	0.30
Texas	183	309	5	1703	0.11
Wisconsin	251	499	5	1703	0.21

- **Citation networks** [45, 51]. Cora, Citeseer, and Pubmed are benchmark datasets of citation networks. Nodes represent paper, and edges represent citation relationships in these networks. The features are bag-of-words representations of papers, and labels denote their academic fields.
- **Social networks** [35]. TW and PT are two subsets of Twitch Gamers dataset [36], designed for binary node classification tasks, where nodes correspond to users and links to mutual friendships. The features are liked games, location, and streaming habits of the users. The labels denote whether a streamer uses explicit language (Taiwanese and Portuguese).
- **WebKB networks** [11]. Cornell, Texas, and Wisconsin are three subsets of WebKB, where nodes are web pages, and edges are hyperlinks. The features are the bag-of-words representation of pages. The labels denote categories of pages, including student, project, course, staff, and faculty.
- **Actor co-occurrence network** [40]. This dataset is a subgraph of the film-director-actor-writer network, in which nodes represent actors, edges represent co-occurrence relation, node features are keywords of the actor, and labels are the types of actors.

A.3 Baselines

Baselines are briefly described as follows[3]:

- **GCN** [45] is the most popular GNN, which defines the first-order approximation of a localized spectral filter on graphs.
- **GAT** [41] introduces a self-attention mechanism to important scores for different neighbor nodes.
- **GraphSAGE** [14] is an inductive framework that leverages node features to generate embeddings by sampling and aggregating features from the local neighborhood.
- **APPNP** [9] combines GCN with personalized PageRank.
- **GCNII**[4] [4] employs residual connection and identity mapping.
- **Grand**[5] [7] purposes a random propagation strategy for data augmentation, and uses consistency regularization to optimize.
- **Mixhop**[6] [1] aggregates mixing neighborhood information.
- **Geom-GCN**[7] [28] exploits geometric relationships to capture long-range dependencies within structural neighborhoods. Three variant of Geom-GCN is used for comparison.
- **GDC**[8] [10] refines graph structure based on diffusion kernels.
- **GEN**[9] [43] estimates underlying meaningful graph structures.
- **H_2GCN**[10] [60] combine multi-hop neighbor-embeddings for adapting to both heterophily and homophily graph settings.
- **DropEdge**[11] [34] randomly removes edges from the input graph for over-fitting prevention.
- **Jaccard**[12] [46] prunes the edges connecting nodes with small Jaccard similarity.

[3]For GCN, GAT, GraphSAGE, and APPNP layers, we adopt implementation from DGL library [42]:https://github.com/dmlc/dgl
[4]https://github.com/chennnM/GCNII
[5]https://github.com/THUDM/GRAND
[6]https://github.com/samihaija/mixhop
[7]https://github.com/graphdml-uiuc-jlu/geom-gcn
[8]https://github.com/gasteigerjo/gdc
[9]https://github.com/BUPT-GAMMA/Graph-Structure-Estimation-Neural-Networks
[10]https://github.com/GemsLab/H2GCN
[11]https://github.com/DropEdge/DropEdge
[12]https://github.com/DSE-MSU/DeepRobust

Table 6: Comparison of training time(hr.) of achieving the best performance based on GPU.

Method	Cora	Citeseer	Pubmed	PT	TW	Actor	Cornell	Texas	Wisconsin
SE-GSL$_{GCN}$	0.071	0.213	4.574	0.178	0.374	1.482	0.006	0.008	0.009
SE-GSL$_{SAGE}$	0.074	0.076	4.852	0.169	0.214	0.817	0.006	0.007	0.009
SE-GSL$_{GAT}$	0.071	0.180	4.602	0.172	0.329	1.273	0.006	0.008	0.009
SE-GSL$_{APPNP}$	0.073	0.215	4.854	0.138	0.379	1.367	0.010	0.011	0.013

A.4 Overall algorithm of SE-GSL

The overall algorithm of SE-GSL is shown in Algorithm 1. Note that, if choose to retain the connection from the previous iteration, to ensure that the number of edges remains stable during the training, a percentage of edges in the reconstructed graph with low similarity will be dropped in each iteration.

Algorithm 1: Model training for SE-GSL

Input: a graph $G = (V, E)$, features X, labels Y_L, mode $\in True, False$
iterations η, encoding tree height K, hyperparameter θ
Output: optimized graph $G' = (V, E')$, prediction Y_P, GNN parameters Θ

Initialize Θ;
for $i = 1$ *to* η **do**
  Update Θ by classification loss $\mathcal{L}_{cls}(Y_L, Y_P)$;
  Getting node representation $X' = \text{GNN}(X)$;
  Initialize $k = 1$ for k-NN structuralization;
  Create fusion map G_f according to Algorithm 2;
  Create K-dimensional encoding tree $\mathcal{T}^*$ according to Algorithm 3;
  for *each non-root node* α *in* $\mathcal{T}^*$ **do**
    Calculate $H^{\mathcal{T}^*}(G_f; (\lambda, \alpha])$ through Eq. 7;
    Assign probability $P(\alpha)$ to α through Eq. 8;
  for *each subtree rooted at* α *in* $\mathcal{T}^*$ **do**
    Assuming α has n children, set $t = \theta \times n$;
    for $j = 1$ *to* t **do**
      Sample a node pair (v_m, v_n) according to § 3.4;
      Adding edge e_{mn} to G';
  if *mode* **then**
    Let $E' = E \cup E'$, where E' and E are the edge set of G' and G, respectively;
    Drop a percentage of edges in G';
  Update graph structure $G \leftarrow G'$; Update node representation: $X \leftarrow X'$;
Get prediction Y_P;
Return G', Y_P and Θ;

A.5 Algorithm of one-dimensional structural entropy guided graph enhancement

The k-selector is designed for choosing an optimal k for k-NN structuralization under the guidance of one-dimensional structural entropy maximization. The algorithm of k-selector and fusion graph construction is shown in Algorithm 2.

Algorithm 2: k-selector and fusion graph construction

Input: a graph $G = (V, E)$, node representation X
Output: fusion graph G_f

Calculate $S \in \mathbb{R}^{|V| \times |V|}$ via Eq. 4;
for $k = 2$ *to* $|V| - 1$ **do**
  Generate G_{knn} by S;
  Generate $G_f^{(k)} = \{V, E_f = E \cup E_{knn}\}$;
  Reweight $G_f^{(k)}$ via Eq. 5;
  Calculate $H^1(G_f^{(k)})$ via Eq. 1;
  if $H^1(G_f^{(k)})$ *reaches the maximal optima* **then**
    $G_f \leftarrow G_f^{(k)}$;
    Return G_f;

A.6 Algorithm of high-dimensional structural entropy minimization

The pseudo-code of the high-dimensional structural entropy minimization algorithm is shown in Algorithm 3.

Algorithm 3: K-dimensional structural entropy minimization

Input: a graph $G = (V, E)$, the height of encoding tree $k > 1$
Output: Optimal high-dimensional encoding tree $\mathcal{T}^*$

//Initialize an encoding tree $\mathcal{T}$ with height 1 and root λ
Create root node λ;
for $v_i \in V$ **do**
  Create node α_i. Let $T_{\alpha_i} = v_i$;
  $v_i^- = \lambda$;
//Generation of binary encoding tree
while λ *has more than* 2 *children* **do**
  Select α_i and α_j in $\mathcal{T}$, condition on $\alpha_i^- = \alpha_j^- = \lambda$ and $\arg\max_{\alpha_i, \alpha_j}(H^{\mathcal{T}}(G) - H^{\mathcal{T}}_{\text{CB}(\alpha_i, \alpha_j)}(G))$;
  $\text{CB}(\alpha_i, \alpha_j)$ according to Definition 1;
//Squeezing of encoding tree
while height$(\mathcal{T}) > K$ **do**
  Select non-root node α and β in $\mathcal{T}$, condition on $\alpha^- = \beta$ and $\arg\max_{\alpha, \beta}(H^{\mathcal{T}}(G) - H^{\mathcal{T}}_{\text{LF}(\alpha, \beta)}(G))$;
  $\text{LF}(\alpha, \beta)$ according to Definition 2;
Return $\mathcal{T}^* \leftarrow \mathcal{T}$;