LSEnet: Lorentz Structural Entropy Neural Network for Deep Graph Clustering

Li Sun [1] **Zhenhao Huang** [1] **Hao Peng** [2] **Yujie Wang** [1] **Chunyang Liu** [3] **Philip S. Yu** [4]

Abstract

Graph clustering is a fundamental problem in machine learning. Deep learning methods achieve the state-of-the-art results in recent years, but they still cannot work without predefined cluster numbers. Such limitation motivates us to pose a more challenging problem of *graph clustering with unknown cluster number*. We propose to address this problem from a fresh perspective of graph information theory (i.e., structural information). In the literature, structural information has not yet been introduced to deep clustering, and its classic definition falls short of discrete formulation and modeling node features. In this work, we first formulate a differentiable structural information (DSI) in the continuous realm, accompanied by several theoretical results. By minimizing DSI, we construct the optimal partitioning tree where densely connected nodes in the graph tend to have the same assignment, revealing the cluster structure. DSI is also theoretically presented as a new graph clustering objective, not requiring the predefined cluster number. Furthermore, we design a neural `LSEnet` in the Lorentz model of hyperbolic space, where we integrate node features to structural information via manifold-valued graph convolution. Extensive empirical results on real graphs show the superiority of our approach.

1. Introduction

Graph clustering aims to group the nodes into several clusters, and routinely finds itself in applications ranging from biochemical analysis to community detection (Jia et al., 2019; Liu et al., 2023c). With the advance of graph neural networks (Kipf & Welling, 2017; Velickovic et al., 2018), deep graph clustering (Devvrit et al., 2022; Wang et al., 2023) achieves remarkable success in recent years.

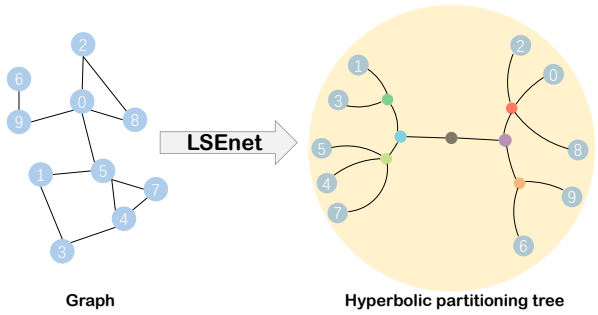


Figure 1. Overview. In hyperbolic space, the proposed `LSEnet` learns a partitioning tree for node clustering without predefined K.

So far, deep graph clustering still cannot work without a predefined cluster number K, and thus one needs to correctly predict the cluster number before clustering, which is often impractical in real cases. Also, estimating cluster numbers is nontrivial. Empirical methods such as Elbow or Bayesian information criterion need to train the deep model repeatedly (Schubert, 2023), and computation cost is much too expensive. For the clustering without graph structures, possible solutions free of K include Bayesian non-parametric methods (Gershman & Blei, 2011), density-based models, e.g., DBSCAN (Ester et al., 1996), and hierarchical clustering (Cohen-addad et al., 2019). However, they cannot be directly applied to graphs owing to the inter-correlation among the nodes. That is, *the problem of graph clustering with unknown cluster number largely remains open.*

In this work, we present a fresh perspective of information theory. Stemming from Shannon entropy, **structural information** (Li & Pan, 2016) is formulated to measure the uncertainty on graphs. Minimizing the structural information, an optimal partitioning tree is constructed to describe the graph's self-organization without the knowledge of cluster number. It sheds light on the targeted problem but presents several significant gaps to deep clustering in the meantime. First and foremost, the clustering ability of structural entropy is still unclear. In the literature, structural information has not yet been introduced to deep clustering, though it has been receiving research attention recently (Liu et al., 2019;

[1]North China Electric Power University, Beijing 102206, China [2]Beihang University, Beijing 100191, China [3]Didi Chuxing, Beijing, China [4]University of Illinois at Chicago, IL, USA. Correspondence to: Li Sun <ccesunli@ncepu.edu>, Hao Peng <penghao@buaa.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Wu et al., 2023; Zou et al., 2023). Second, the discrete formulation prevents the gradient backpropagation, posing a fundamental challenge to train a deep model. Third, the classic definition neglects the node features, which are often equally important to graph clustering.

In light of the aforementioned issues, we present a novel *differentiable structural information* (DSI) in the continuous realm. DSI is formulated with level-wise assignment matrices and is equivalent to the classic formulation under binary assignment. In fact, the conductance for graph clustering is inherently related to DSI. The intuition is that in the partitioning tree $\mathcal{T}$ of DSI minimization, the densely connected nodes in the graph tend to be assigned to the same parent node, minimizing the conductance and presenting the cluster structure. Thus DSI emerges as a new graph clustering objective, not requiring the predefined cluster number. Next, we consider a partitioning tree $\mathcal{T}_{\text{net}}$ learned by a neural network, given that our objective supports gradient backpropagation for the network. We show that the structural entropy of the optimal $\mathcal{T}_{\text{net}}$ well approximates that of Li & Pan (2016) under slight constraint. Furthermore, we design a Lorentz Structural Entropy neural net (`LSEnet`) to learn $\mathcal{T}_{\text{net}}$ (as shown in Figure 1) in the Lorentz model of **hyperbolic space**, where we further integrate node features to structural information via graph convolution net. Specifically, we first embed the leaf nodes of the partitioning tree, and then recursively learn the parent nodes from bottom to top (root node), where the level-wise parent assignment is attentively determined by the proposed Lorentz assigner. Consequently, `LSEnet` combines the advantages of both structural entropy and hyperbolic space for graph clustering. The main contributions are listed as follows.

- We study a challenging yet practical problem of graph clustering with unknown cluster number and, to our best knowledge, make the first attempt to bridge deep graph clustering and structural information.
- We present the differentiable structural information (DSI), generalizing the classic theory to the continuous realm. DSI emerges as a new graph clustering objective, not requiring the cluster number.
- We design a novel hyperbolic `LSEnet` with graph neural network, further integrating the structural information and node features. Extensive empirical results show the superiority of `LSEnet` for graph clustering.

2. Related Work

Deep Graph Clustering. Deep models have achieved state-of-the-art results in node clustering. Classic methods leverage a reconstructive loss to learn node representations, while identifying node clusters with distance-based or model-based algorithms (e.g., k-means, Gaussian mixture model) (Xie et al., 2016; Almahairi et al., 2016; Yang et al., 2017). Other methods learn cluster assignment with generative adversarial nets (Yang et al., 2020; Jia et al., 2019). Contrastive clustering explores the similarity on the graph itself, pulling positive samples together and pushing negative ones apart (Devvrit et al., 2022; Pan & Kang, 2021; Li et al., 2022). Sun et al. (2023b) consider contrastive graph clustering in the product manifold with a loss of Ricci curvature. Normalizing flows have recently been introduced to graph clustering (Wang et al., 2023). Few deep model is built without the predefined cluster number, to the best of our knowledge. Very recently, Liu et al. (2023a) focus on automatically estimating the number of clusters via reinforcement learning, which is orthogonal to our study.

Structural Entropy. Information entropy is a key notation of information theory (Shannon, 1948), measuring the amount of information for unstructured data, and it fails to study the information on graphs. On information measure for graphs, early practices, e.g., Von Neumann entropy (Braunstein et al., 2006) and Gibbs entropy (Bianconi, 2009), are still defined by unstructured probability distributions, and the graph structure is degenerated. Recently, structural entropy is proposed in account of the natural self-organizing in the graphs (Li & Pan, 2016), and has been successfully applied to graph pooling (Wu et al., 2022), adversarial attack (Liu et al., 2019), contrastive learning (Wu et al., 2023), dimension estimation (Yang et al., 2023b) and graph structural learning (Zou et al., 2023). However, structural entropy has not yet been introduced to deep clustering, and the gap roots in the discrete formulation of Li & Pan (2016). Besides, it falls short of considering node features that are important to graph clustering as well.

Riemannian Graph Learning. Euclidean space has been the workhorse of graph learning for decades (Perozzi et al., 2014; Kipf & Welling, 2017; Velickovic et al., 2018; Ye et al., 2024). In recent years, Riemannian manifolds have emerged as an exciting alternative. Hyperbolic models (Nickel & Kiela, 2017; Chami et al., 2019; Sun et al., 2021; 2022a; 2023c; Zhang et al., 2021; Yang et al., 2023a; Fu et al., 2023) achieve remarkable success on the graphs dominated by tree-like structures (Krioukov et al., 2010). In fact, an arbitrary tree can be embedded in hyperbolic space with bounded distortion (Sarkar, 2011). Fu et al. (2021) study the optimal curvature of hyperbolic space for graph embedding. Beyond hyperbolic space, the product manifolds (Sun et al., 2022b; 2024c) show its superiority on generic graph structures. Recently, Ricci curvature of Riemannian geometry is given a differentiable surrogate for graph structural learning (Sun et al., 2023a). Manifold vector fields (ordinary differential equations) (Sun et al., 2024b) are introduced to study information diffusion on the graphs (Sun et al., 2024a). Note that, *the inherent connection between the tree and hyperbolic space supports the construction of our model,* `LSEnet`.

3. Preliminaries and Notations

Herein, different from the typical setting of existing works, we are interested in a more challenging problem of *graph clustering with unknown cluster number.* Some preliminary concepts and notations are introduced here.

Graph and Graph Clustering. A weighted graph G is represented as $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$. $\mathcal{V}$ is the set of N nodes, and the degree of node v_i is denoted as d_i. $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the edge set with a weight function w, and the edge weights are collected in the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. $\mathbf{X}$ is the matrix of node features. For a node subset $\mathcal{U} \subset \mathcal{V}$, its volume $\mathrm{Vol}(\mathcal{U})$ is defined as the sum of degrees of the nodes in $\mathcal{U}$. Graph clustering aims to group nodes into several clusters. In the literature, deep clustering methods typically rely on the predefined cluster number K to build the deep model. In this paper, we consider a more challenging case that *the number of clusters K is unavailable*.

Hyperbolic Space. In Riemannian geometry, unlike the "flat" Euclidean space, hyperbolic space is a curved space with negative curvatures. The notion of curvature κ measures the extent of how the manifold deviates from being flat. We utilize the *Lorentz model* of hyperbolic space. Concretely, a $d-$dimensional Lorentz model $\mathbb{L}^{\kappa,d}$ with curvature κ is defined on the manifold of $\{\boldsymbol{x} \in \mathbb{R}^{d+1} | \langle \boldsymbol{x}, \boldsymbol{x}\rangle_{\mathbb{L}} = \frac{1}{\kappa}\}$, where the Minkowski inner product $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}} = \boldsymbol{x}\mathbf{R}\boldsymbol{y}$ is defined with the matrix of Riemannian metric $\mathbf{R} = diag(-1, 1, 1, \cdots, 1) \in \mathbb{R}^{(d+1)\times(d+1)}$. $\forall \boldsymbol{x}, \boldsymbol{y}$ in the Lorentz, the distance is given as $d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{y}) = arccosh(\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}})$. Lorentz norm is defined as $\|\boldsymbol{u}\|_{\mathbb{L}} = \sqrt{\langle \boldsymbol{u}, \boldsymbol{u}\rangle_{\mathbb{L}}}$, where $\boldsymbol{u}$ is a vector in the tangent space. Please refer to Appendix B for further facts.

Throughout this paper, the lowercase boldfaced $\boldsymbol{x}$ and uppercase $\mathbf{X}$ denote vector and matrix, respectively. The notation table is given in Appendix C.1.

4. Differentiable Structural Information

In this section, we first establish a new formulation of structural information (Li & Pan, 2016), and then give a continuous relaxation to the differentiable realm, yielding a new objective and an optimization approach for graph clustering without the predefined cluster number. We start with the classic formulation as follows.

Definition 4.1 (H-dimensional Structural Entropy (Li & Pan, 2016))**.** Given a weighted graph $G = (\mathcal{V}, \mathcal{E})$ with weight function w and a partitioning tree $\mathcal{T}$ of G with height H, the *structural information* of G with respect to each non-root node α of $\mathcal{T}$ is defined as

$$\mathcal{H}^{\mathcal{T}}(G; \alpha) = -\frac{g_\alpha}{\mathrm{Vol}(G)} \log_2 \frac{V_\alpha}{V_{\alpha^-}}. \tag{1}$$

In the *partitioning tree* $\mathcal{T}$ with root node λ, each tree node α is associated with a subset of $\mathcal{V}$, denoted as module T_α, and the immediate predecessor of it is written as α^-. The module of the leaf node is a singleton of the graph node. The scalar g_α is the total weights of graph edges with exactly one endpoint in module T_α. Then, the H-dimensional structural information of G by $\mathcal{T}$ is given as,

$$\mathcal{H}^{\mathcal{T}}(G) = \sum\nolimits_{\alpha\in\mathcal{T}, \alpha\neq\lambda} \mathcal{H}^{\mathcal{T}}(G; \alpha). \tag{2}$$

Traversing all possible partitioning trees of G with height H, *H-dimensional structural entropy* of G is defined as

$$\mathcal{H}^{H}(G) = \min_{\mathcal{T}} \mathcal{H}^{\mathcal{T}}(G), \quad \mathcal{T}^* = \arg_{\mathcal{T}} \min \mathcal{H}^{\mathcal{T}}(G), \tag{3}$$

where $\mathcal{T}^*$ is the optimal tree of G which encodes the self-organization and minimizes the uncertainty of the graph.

The structural information in Eq. (2) is formulated node-wisely, and cannot be optimized via gradient-based methods.

4.1. A New Formulation

To bridge this gap, we present a new formulation of structural information with the level-wise assignment, which is shown to be equivalent to the classic formulation in Eq. (2).

Definition 4.2 (Level-wise Assignment)**.** For a partitioning tree $\mathcal{T}$ with height H, assuming that the number of tree nodes at the h-th level is N_h, we define a *level-wise parent assignment matrix* $\mathbf{C}^h \in \{0, 1\}^{N_h \times N_{h-1}}$ *from h-th to $(h-1)$-th level*, where $\mathbf{C}^h_{ij} = 1$ means the i-th node of $\mathcal{T}$ at h-th level is the parent node of j-th node at $(h-1)$-th level.

Definition 4.3 (H-dimensional Structural Information)**.** For a graph G and its partitioning tree $\mathcal{T}$ in Definition 4.2, we rewrite the formula of H-dimensional structural information of G with respect to $\mathcal{T}$ **at height** h as

$$\mathcal{H}^{\mathcal{T}}(G; h) = -\frac{1}{V} \sum_{k=1}^{N_h} (V_k^h - \sum_{(i,j)\in\mathcal{E}} S_{ik}^h S_{jk}^h w_{ij}) \log_2 \frac{V_k^h}{V_{k^-}^{h-1}} \tag{4}$$

where $V = \mathrm{Vol}(G)$ is the volume of G. For the k-th node in height h, V_k^h and $V_{k^-}^{h-1}$ are the volume of graph node sets T_k and T_{k^-}, respectively. Thus we have

$$\mathbf{S}^h = \prod\nolimits_{k=H+1}^{h+1} \mathbf{C}^k, \quad \mathbf{C}^{H+1} = \mathbf{I}_N, \tag{5}$$

$$V_k^h = \sum\nolimits_{i=1}^{N} S_{ik}^h d_i, \quad V_{k^-}^{h-1} = \sum\nolimits_{k'=1}^{N_{h-1}} \mathbf{C}_{kk'}^h V_{k'}^{h-1}. \tag{6}$$

Then, the H-dimensional structural information of G is $\mathcal{H}^{\mathcal{T}}(G) = \sum_{h=1}^{H} H^{\mathcal{T}}(G; h)$.

Theorem 4.4 (Equivalence)**.** *The formula $\mathcal{H}^{\mathcal{T}}(G)$ in Definition 4.3 is equivalent to Eq. (2) given in Definition 4.1.*

Proof. Please refer to Appendix A.1. □

4.2. Properties

This part shows some general properties of the new formulation and theoretically demonstrates the inherent connection between structural entropy and graph clustering. We first give an arithmetic property regarding Definition 4.3 to support the following claim on graph clustering. The proofs of the lemma/theorems are detailed in Appendix A.

Lemma 4.5 (Additivity-Appendix A.3)**.** *The 1-dimensional structural entropy of G can be decomposed as follows*

$$\mathcal{H}^1(G) = \sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}} \frac{V_j^{h-1}}{V} E([\frac{C_{kj}^h V_k^h}{V_j^{h-1}}]_{k=1,\dots,N_h}), \quad (7)$$

where $E(p_1,...,p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$ is the entropy.

Theorem 4.6 (Connection to Graph Clustering-Appendix A.4)**.** *Given a graph $G = (\mathcal{V}, \mathcal{E})$ with w, the normalized H-structural entropy of graph G is defined as $\tau(G;H) = \mathcal{H}^H(G)/\mathcal{H}^1(G)$, and $\Phi(G)$ is the graph conductance. With the additivity* (Lemma 4.5)*, the following inequality holds,*

$$\tau(G;H) \geq \Phi(G). \quad (8)$$

Proof. We show the key equations here, and further details are given in Appendix A.4. Without loss of generality, we assume $\min\{V_k^h, V - V_k^h\} = V_k^h$. From Definition 4.3,

$$\begin{aligned}\mathcal{H}^{\mathcal{T}}(G) &= -\frac{1}{V}\sum_{h=1}^{H}\sum_{k=1}^{N_h} \phi_{h,k} V_k^h \log_2 \frac{V_k^h}{V_{k^-}^{h-1}} \\ &\geq -\frac{\Phi(G)}{V}\sum_{h=1}^{H}\sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V_{k^-}^{h-1}}, \end{aligned} \quad (9)$$

Let the k-th tree node in height h denoted as α, then $\phi_{h,k}$ is the conductance of graph node subset T_α, and is defined as $\frac{\sum_{i\in T_\alpha, j\in \bar{T}_\alpha} w_{ij}}{\min\{\mathrm{Vol}(T_\alpha), \mathrm{Vol}(\bar{T}_\alpha)\}}$. where $\bar{T}_\alpha$ is the complement set of T_α. Thus, graph conductance is given as $\Phi(G) = \min_{h,k}\{\phi_{h,k}\}$. Next, we utilize Eq. (6) to obtain the following inequality

$$\begin{aligned}\mathcal{H}^{\mathcal{T}}(G) &\geq -\frac{\Phi(G)}{V}\sum_{h=1}^{H}\sum_{k=1}^{N_h}\sum_{j=1}^{N_{h-1}} C_{kj}^h V_k^h \log_2 \frac{V_k^h}{V_{k^-}^{h-1}} \\ &= -\frac{\Phi(G)}{V}\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}} V_j^{h-1}\sum_{k=1}^{N_h}\frac{C_{kj}^h V_k^h}{V_j^{h-1}} \log_2 \frac{C_{kj}^h V_k^h}{V_j^{h-1}} \\ &= \Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}} \frac{V_j^{h-1}}{V} E([\frac{C_{kj}^h V_k^h}{V_j^{h-1}}]_{k=1,\dots,N_h}) \\ &= \Phi(G)\mathcal{H}^1(G). \end{aligned} \quad (10)$$

Since $\tau(G;\mathcal{T}) = \frac{\mathcal{H}^{\mathcal{T}}(G)}{\mathcal{H}^1(G)} \geq \Phi(G)$ holds for every H-height partitioning tree $\mathcal{T}$ of G, $\tau(G) \geq \Phi(G)$ holds. □

The conductance is a well-defined objective for graph clustering (Yin et al., 2017). $\forall G$, the one-dimensional structural entropy $\mathcal{H}^1(G) = -\frac{1}{\mathrm{Vol}(G)}\sum_{i=1}^{N} d_i \log_2 \frac{d_i}{\mathrm{Vol}(G)}$ is yielded as a constant. Minimizing Eq. (4) is thus equivalent to minimizing conductance, grouping nodes into clusters. Also, Eq. (4) needs no knowledge of cluster number in the graph G. *In short, our formulation is capable of serving as an objective of graph clustering without predefined cluster numbers.*

4.3. Differentiablity & Deep Graph Clustering

Here, we elaborate on how to use a deep model to conduct graph clustering with our new objective. Recall Eq. (4), our objective is differentiable over the parent assignment. If the assignment is relaxed to be the likelihood given by a neural net, our formulation supports gradient backpropagation to learn the deep model. For a graph G, we denote the partitioning tree as $\mathcal{T}_{\text{net}}$ where the mapping from learnable embeddings $\mathbf{Z}$ to the assignment is done via a neural net. The differentiable structural information (DSI) is given as $\mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z};\mathbf{\Theta})$. It takes the same form of Eq. (4) where the assignment is derived by a neural net with parameters Θ. Given G, we consider DSI minimization as follows,

$$\mathbf{Z}^*, \mathbf{\Theta}^* = \arg_{\mathbf{Z},\mathbf{\Theta}} \min \mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z};\mathbf{\Theta}). \quad (11)$$

We learn node embeddings $\mathbf{Z}^*$ and parameter $\mathbf{\Theta}^*$ to derive the level-wise parent assignment for G. Then, the optimal partitioning tree $\mathcal{T}_{\text{net}}^*$ of G is constructed by the assignment, so that *densely connected nodes have a higher probability to be assigned to the same parent in $\mathcal{T}_{\text{net}}^*$, minimizing the conductance and presenting the cluster structure.* Meanwhile, node embeddings jointly learned in the optimization provide geometric notions to refine clusters in representation space.

Next, we introduce the theoretical guarantees of $\mathcal{T}_{\text{net}}^*$. We first define the notion of equivalence relationship in partitioning tree $\mathcal{T}$ with height H. If nodes i and j are in the same module at the level h of $\mathcal{T}$ for $h = 1, \cdots, H$, they are said to be in equivalence relationship, denoted as $i \overset{h}{\sim} j$.

Theorem 4.7 (Bound-Appendix A.2)**.** *For any graph G, $\mathcal{T}^*$ is the optimal partitioning tree of Li & Pan (2016), and $\mathcal{T}_{\text{net}}^*$ with height H is the partitioning tree given by Eq. (11). For any pair of leaf embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ of $\mathcal{T}_{\text{net}}^*$, there exists bounded real functions $\{f_h\}$ and constant c, such that for any $0 < \epsilon < 1$, $\tau \leq \mathcal{O}(1/\ln[(1-\epsilon)/\epsilon])$ and $\frac{1}{1+\exp\{-(f_h(\boldsymbol{z}_i,\boldsymbol{z}_j)-c)/\tau\}} \geq 1-\epsilon$ satisfying $i \overset{h}{\sim} j$, we have*

$$|\mathcal{H}^{\mathcal{T}^*}(G) - \mathcal{H}^{\mathcal{T}_{\text{net}}^*}(G)| \leq \mathcal{O}(\epsilon). \quad (12)$$

That is, the structural entropy calculated by our $\mathcal{T}_{\text{net}}^*$ is well approximated to that of Li & Pan (2016), and the difference is bounded under slight constraints. Thus, $\mathcal{T}_{\text{net}}^*$ serves as an alternative to the optimal $\mathcal{T}^*$ for graph clustering.

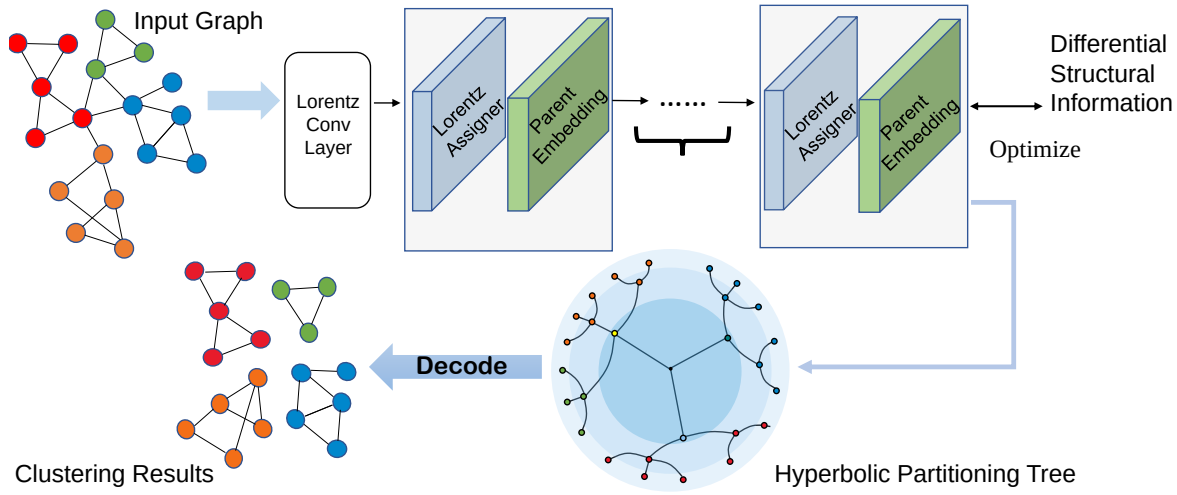


Figure 2. The overall architecture of `LSEnet`. We encode G into Lorentz model $\mathbb{L}^{\kappa,d_\mathcal{T}}$ via a Lorentz convolution layer, and recursively utilize Lorentz assignor and geometric centroid to construct a tree from down to up. Optimizing DSI, we learn the optimal $\mathcal{T}^*_{\text{net}}$ in hyperbolic space, and obtain clustering results correspondingly.

Theorem 4.8 (Flexiblity-Appendix A.6)**.** *$\forall G$, given a partitioning tree $\mathcal{T}$ and adding a node β to get a relaxed $\mathcal{T}'$, the structural information remains unchanged, $H^{\mathcal{T}}(G) = H^{\mathcal{T}'}(G)$, if one of the following conditions holds:*

1. *β as a leaf node, and its module T_β is an empty set.*
2. *β is inserted between node α and its children nodes so that the modules T_β and T_α are equal.*

The theorem above will guide the architecture design of the neural net for $\mathcal{T}^*_{\text{net}}$.

5. `LSEnet`

We propose a novel Lorentz Structural Entropy neural Net (`LSEnet`), which aims to learn the optimal partitioning tree $\mathcal{T}^*_{\text{net}}$ in the Lorentz model of **hyperbolic space**, where we further incorporate node features with structural information by graph convolution net. First, we show the reason we opt for hyperbolic space, rather than Euclidean space.

Theorem 5.1 (Tree $\mathcal{T}$ and Hyperbolic Space (Sarkar, 2011))**.** *$\forall \mathcal{T}$, scaling all edges by a constant so that the edge length is bounded below $\nu\frac{1+\epsilon}{\epsilon}$, there exists an embedding in hyperbolic space that the distortion[1] overall node pairs are bounded by $1+\epsilon$.* (ν is a constant detailed in Appendix B.)

Hyperbolic space is well suited to embed the partitioning tree, and Theorem 5.1 does not hold for Euclidean space.

[1]The distortion is defined as $\frac{1}{|\mathcal{V}|^2}\sum_{ij}\left|\frac{d_G(v_i,v_j)}{d(\mathbf{x}_i,\mathbf{x}_j)}-1\right|$, where each node $v_i \in \mathcal{V}$ is embedded as $\mathbf{x}_i$ in representation space. d_G and d denote the distance in the graph and the space, respectively.

Overall architecture of `LSEnet` is sketched in Figure 2. In hyperbolic space, `LSEnet` first embeds leaf nodes of the tree, and then recursively learns parent nodes, *self-supervised by our new clustering objective Eq. (4).*

5.1. Embedding Leaf Nodes

We design a Lorentz convolution layer to learn leaf embeddings in hyperbolic space, $\text{LConv} : \boldsymbol{x}_i \to \boldsymbol{z}_i^H, \forall v_i$, where $\boldsymbol{x}_i \in \mathbb{L}^{\kappa,d_0}$ and $\boldsymbol{z}_i^H \in \mathbb{L}^{\kappa,d_\mathcal{T}}$ are the node feature and leaf embedding, respectively. In the partitioning tree of height H, the level of nodes is denoted by superscript of h and we have $h = H, \cdots, 1, 0$ from leaf nodes (bottom) to root (top), correspondingly.

We adopt attentional aggregation in LConv and specify the operations as follows. First, dimension transform is done via a *Lorentz linear operator* (Chen et al., 2022). For node $\boldsymbol{x}_i \in \mathbb{L}^{\kappa,d_0}$, the linear operator is given as

$$\text{LLinear}(\boldsymbol{x}) = \begin{bmatrix} \sqrt{\|h(\mathbf{W}\boldsymbol{x},\boldsymbol{v})\|^2 - \frac{1}{\kappa}} \\ h(\mathbf{W}\boldsymbol{x},\boldsymbol{v}) \end{bmatrix} \in \mathbb{L}^{\kappa,d_\mathcal{T}}, \quad (13)$$

where h is a neural network, and $\mathbf{W}$ and $\boldsymbol{v}$ are parameters. Second, we derive attentional weights from the *self-attention mechanism*. Concretely, the attentional weight ω_{ij} between nodes i and j is calculated as

$$\omega_{ij} = \text{LAtt}(\mathbf{Q},\mathbf{K}) = \frac{\exp(-\frac{1}{\sqrt{N}}d^2_\mathbb{L}(\boldsymbol{q}_i,\boldsymbol{k}_j))}{\sum_{l=1}^N \exp(-\frac{1}{\sqrt{N}}d^2_\mathbb{L}(\boldsymbol{q}_i,\boldsymbol{k}_l))}, \quad (14)$$

where $\boldsymbol{q}$ and $\boldsymbol{k}$ are the query and key vector collected in row vector of $\mathbf{Q}$ and $\mathbf{K}$, respectively. The queries and keys

are derived from node feature $\boldsymbol{x}$ via LLinear with different parameters. Third, weighted aggregation is considered as the *arithmetic mean* among manifold-valued vectors. Given a set of points $\{\boldsymbol{x}_i\}_{i=1,\cdots,N}$ in the Lorentz model $\mathbb{L}^{\kappa,d_\mathcal{T}}$,

$$\mathrm{LAgg}(\boldsymbol{\omega}, \mathbf{X}) = \frac{1}{\sqrt{-\kappa}} \sum_{i=1}^{N} \frac{\omega_i}{|\|\sum_{j=1}^{N} \omega_j \boldsymbol{x}_j\|_{\mathbb{L}}|} \boldsymbol{x}_i, \quad (15)$$

where $\boldsymbol{\omega}$ is the weight vector, and $\boldsymbol{x}$ are summarized in $\mathbf{X}$. The augmented form is that for $\boldsymbol{\Omega} = [\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_N]^\top$, we have $\mathrm{LAgg}(\boldsymbol{\Omega}, \mathbf{X}) = [\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_N]^\top$, where the weighted mean is $\boldsymbol{\mu}_i = \mathrm{LAgg}(\boldsymbol{\omega}_i, \mathbf{X})$. Overall, the Lorentz convolution layer is formulated as,

$$\mathrm{LConv}(\mathbf{X}|\mathbf{A}) = \mathrm{LAgg}(\mathrm{LAtt}(\mathbf{Q}, \mathbf{K}) \odot \mathbf{A}, \mathrm{LLinear}(\mathbf{X})), \quad (16)$$

where $\odot$ is the Hadamard product, masking the attentional weight if the corresponding edge does not exist in the graph.

5.2. Learning Parent Nodes

A primary challenge is that, in the partitioning tree, *the node number at each internal level is unknown.* To address this issue, we introduce a simple yet effective method, setting a large enough node number N_h at the h-th level. A large N_h may introduce redundant nodes and result in a relaxed partitioning tree. According to **Theorem 4.8** established in Sec. 4.3, redundant nodes in the partitioning tree do not affect the value of structural entropy, and finally present as empty leaf nodes by optimizing our objective. Theoretically, if an internal level has insufficient nodes, the self-organization of the graph can still be described by multiple levels in the partitioning tree.

Without loss of generality, we consider the assignment between h-th and $(h-1)$-th levels given node embeddings $\boldsymbol{z}^h \in \mathbb{L}^{\kappa,d_\mathcal{T}}$ at h-th level. Recalling Definition 4.2, the i-th row of assignment $\mathbf{C}^h \in \mathbb{R}^{N_h \times N_{h-1}}$ describes the belonging of i-th node at h-th to the parent nodes at $(h-1)$-th. We design a Lorentz assigner following the intuition that neighborhood nodes in the graph tend to have similar assignments. Concretely, we leverage Multilayer Perceptron MLP to learn the assignment from embeddings, and meanwhile the similarity is parameterized by LAtt defined in Eq. 14. Thus, the Lorentz assigner is formulated as

$$\mathbf{C}^h = \sigma\left(\left(\mathrm{LAtt}(\mathbf{Q}, \mathbf{K}) \odot \mathbf{A}^h\right) \mathrm{MLP}(\mathbf{Z}^h)\right), \quad (17)$$

where $\mathbf{A}^h$ is the graph structure at h-th level of the tree. σ denotes the row normalization via $Softmax$ function.

The remaining task is to infer the node embeddings $\boldsymbol{z}^{h-1} \in \mathbb{L}^{\kappa,d_\mathcal{T}}$ at $(h-1)$-th level. As reported in Chami et al. (2020), a parent node locates at the point that has the shortest path to all the child nodes at h-th level and, correspondingly, the parent node is the Fréchet mean of child nodes in the manifold. The challenge here is that Fréchet mean regarding the canonical distance exists no closed-form solution (Lou et al., 2020). Alternatively, we consider the geometric centroid with respect to the squared Lorentz distance, where the weights are given by the soft assignment $\mathbf{C}^h$,

$$\boldsymbol{z}_j^{h-1} = \arg_{\boldsymbol{z}_j^{h-1}} \min \sum\nolimits_{i=1}^{N} c_{ij} d_{\mathbb{L}}^2(\boldsymbol{z}_j^{h-1}, \boldsymbol{z}_i^h). \quad (18)$$

Solving the optimization constrained in the manifold $\mathbb{L}^{\kappa,d_\mathcal{T}}$, we derive the closed-form solution of parent node embeddings as follows,

$$\mathbf{Z}^{h-1} = \mathrm{LAgg}(\mathbf{C}^h, \mathbf{Z}^h) \in \mathbb{L}^{\kappa,d_\mathcal{T}}, \quad (19)$$

which is the augmented form of Eq. (15).

Theorem 5.2 (Geometric Centroid-Appendix B.4)**.** *In Lorentz model $\mathbb{L}^{\kappa,d_\mathcal{T}}$ of hyperbolic space, for any set of points $\{\boldsymbol{z}_i^h\}$, the arithmetic mean is*

$$\boldsymbol{z}_j^{h-1} = \frac{1}{\sqrt{-\kappa}} \sum_{i=1}^{N} \frac{c_{ji}}{|\|\sum_{l=1}^{n} c_{jl} \boldsymbol{z}_l^h\|_{\mathbb{L}}|} \boldsymbol{z}_i^h \in \mathbb{L}^{\kappa,d_\mathcal{T}}, \quad (20)$$

and is the closed-form solution of the geometric centroid defined in the minimization of Eq. 18.

Remark. In fact, the geometric centroid in Theorem 5.2 is also equivalent to the gyro-midpoint in Poincaré ball model of hyperbolic space, detailed in Appendix B.4

Algorithm 1 Training `LSEnet`

Input: A weighted graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, Height of partitioning tree H, Training iterations L
Output: The partitioning tree $\mathcal{T}$; Tree nodes embeddings $\{\mathbf{Z}^h\}_{h=1,\ldots,H}$ at all the levels;

for *epoch* $= 1$ to L **do**
  Obtain leaf node embeddings $\mathbf{Z}^H = \mathrm{LConv}(\mathbf{X}, \mathbf{A})$.
  for $h = H-1$ to 1 **do**
    Compute $\mathbf{Z}^h$, $\mathbf{A}^h$, $\mathbf{C}^h$ with Eqs. 17, 19 and 21.
    Compute $\mathbf{S}^h$ for $h = 1, \cdots, H$ in Eq. 5.
  end for
  Compute the objective of DSI in Eq. 4.
  Optimize parameters via Riemannian Adam.
end for
Create a root node λ and a queue $\mathcal{Q}$.
while $\mathcal{Q}$ is not empty **do**
  Get first item α in $\mathcal{Q}$.
  Let $h = \alpha.h + 1$ and search subsets P from $\mathbf{S}^h$.
  Create nodes from P and put into the queue $\mathcal{Q}$.
  Add theses nodes into α's children list.
end while
Return the partitioning tree $\mathcal{T} := \lambda$.

Table 1. Clustering results on Cora, Citerseer, AMAP, and Computer datasets in terms of NMI and ARI (%). OOM denotes Out-of-Memory on our hardware. The best results are highlighted in **boldface**, and runner-ups are underlined.

	Cora		Citeseer		AMAP		Computer	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
K-Means	14.98±0.82	8.60±0.40	16.94±0.24	13.43±0.57	19.31±3.75	12.61±3.54	16.64±0.75	2.71±0.82
VGAE (Kipf & Welling, 2016)	43.40±1.62	37.50±2.13	32.70±0.30	33.10±0.55	66.01±0.84	56.24±0.22	37.62±0.24	22.16±0.16
ARGA (Pan et al., 2018)	48.10±0.45	44.10±0.28	35.10±0.58	34.60±0.48	58.36±1.02	44.18±0.85	37.21±0.58	26.28±1.38
MVGRL (Hassani & Ahmadi, 2020)	62.91±0.42	56.96±0.74	46.96±0.25	44.97±0.57	36.89±1.31	18.77±1.54	10.12±2.21	5.53±1.78
Sublime (Liu et al., 2022b)	54.20±0.28	50.32±0.32	44.10±0.27	43.91±0.35	6.37±0.54	5.36±0.24	39.16±1.82	24.15±0.63
ProGCL (Xia et al., 2022)	41.02±1.64	30.71±1.38	39.59±1.58	36.16±2.21	39.56±0.25	34.18±0.58	35.50±2.06	26.08±1.91
GRACE (Yang et al., 2017)	57.30±0.86	52.70±1.20	39.90±2.26	37.70±1.35	53.46±1.32	42.74±1.57	47.90±1.65	36.40±1.56
DEC (Xie et al., 2016)	23.54±1.13	15.13±0.72	28.34±0.42	28.12±0.24	37.35±0.84	18.29±0.64	38.56±0.24	34.76±0.35
MCGC (Pan & Kang, 2021)	44.90±1.56	37.80±1.24	40.14±1.44	38.00±0.85	61.54±0.29	52.10±0.27	53.17±1.29	39.02±0.53
DCRN (Liu et al., 2022a)	48.86±0.85	43.79±0.48	45.86±0.35	47.64±0.30	73.70±1.24	63.69±0.84	OOM	OOM
FT-VGAE (Mrabah et al., 2022)	61.03±0.52	58.22±1.27	44.50±0.13	46.71±0.75	69.76±1.06	59.30±0.81	51.36±0.92	40.07±2.13
gCooL (Li et al., 2022)	58.33±0.24	56.87±1.03	47.29±0.10	46.78±1.51	63.21±0.09	52.40±0.11	47.42±1.76	27.71±2.28
S^3GC (Devvrit et al., 2022)	58.90±1.81	54.40±2.52	44.12±0.90	44.80±0.65	59.78±0.45	56.13±0.58	54.80±1.22	29.93±0.22
Congregate (Sun et al., 2023b)	63.16±0.71	59.27±1.23	50.92±1.58	47.59±1.60	70.99±0.67	60.55±1.36	46.03±0.47	38.57±1.05
DinkNet (Liu et al., 2023b)	62.28±0.24	61.61±0.90	45.87±0.24	46.96±0.30	**74.36**±1.24	**68.40**±1.37	39.54±1.52	33.87±0.12
GC-Flow (Wang et al., 2023)	62.15±1.35	63.14±0.80	40.50±1.32	42.62±1.44	36.45±1.21	37.24±1.26	41.10±1.06	35.60±1.82
RGC (Liu et al., 2023a)	57.60±1.36	50.46±1.72	45.70±0.29	45.47±0.43	47.65±0.91	42.65±1.53	46.24±1.05	36.12±0.30
LSEnet (Ours)	**63.97**±0.67	**63.35**±0.56	**52.26**±1.09	**48.01**±1.25	71.72±1.30	65.08±0.73	**55.03**±0.79	**42.15**±1.02

According to Eq. (17), it requires the graph structure $\mathbf{A}^{h-1}$ among the parent nodes at $(h-1)$-th level to derive the assignment. We give the adjacency matrix of the $(h-1)$-th level graph as follows,

$$\mathbf{A}^{h-1} = (\mathbf{C}^h)^\top \mathbf{A}^h \mathbf{C}^h. \tag{21}$$

We recursively utilize Eqs. (17), (19) and (21) to build the partitioning tree from bottom to top in hyperbolic space.

Complexity Analysis First, DSI loss is computed level-wisely, and for each level h it has the time complexity of $O(N_h|\mathcal{E}|)$. Second, the complexity of constructing the partitioning tree is $O(HN)$ with the breadth-first search in hyperbolic space. Thus, the overall complexity is yielded as $O(|\mathcal{V}||\mathcal{E}|)$ with a small constant H, and $N_h \ll |\mathcal{V}|$ in intermediate layers. Though the predefined K is not required, it still presents similar complexity to most clustering models, i.e., $O(K|\mathcal{V}|^2)$, given real graphs are typically sparse.

5.3. Hyperbolic Partitioning Tree

In the principle of structural information minimization, the optimal partitioning tree is constructed to describe the self-organization of the graph (Li & Pan, 2016; Wu et al., 2023). In the continuous realm, *LSEnet learns the optimal partitioning tree in hyperbolic space* by minimizing the objective in Eq. (4). DSI is applied on all the level-wise assignment $\mathbf{C}$'s, which are parameterized by neural networks in Sec. 5.1 and 5.2. We suggest to place the tree root at the origin of $\mathbb{L}^{\kappa,d_\mathcal{T}}$, so that the learnt tree enjoys the symmetry of Lorentz model. Consequently, the hyperbolic partitioning tree describes the graph's self-organization and clustering structure in light of Theorem 4.6.

The overall procedure of training LSEnet is summarized in Algorithm 1. Specifically, in Lines 1-9, we learn level-wise assignments and node embeddings in a bottom-up manner. In Lines 10-17, the optimal partitioning tree is given via a top-down process in which we avoid empty leaf nodes and fix single chains in the relaxed tree (case one and case two in Theorem 4.8). Further details of the procedure are shown in Algorithm 2 given in Appendix C.2.

In brief, LSEnet combines the advantages of both structural entropy and hyperbolic space, and uncovers the clustering structures without predefined cluster number K.

6. Experiments

We conduct extensive experiments on benchmark datasets to evaluate the effectiveness of the proposed LSEnet.[2] Furthermore, we compare with the classic formulation, evaluate the parameter sensitivity, and visualize the hyperbolic portioning tree. Additional results are shown in Appendix D.

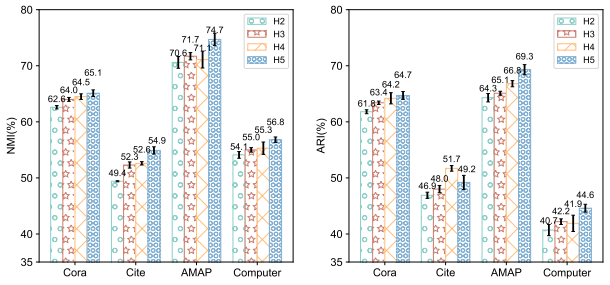
6.1. Graph Clustering

Datasets & Baselines. The evaluations are conducted on 4 datasets, including Cora, Citeseer, and Amazon Photo (AMAP) (Liu et al., 2023b), and a larger Computer dataset (Li et al., 2022). We focus on deep graph clustering in this paper, and thus we primarily compare with the deep models, including 11 graph clustering methods, i.e., GRACE, DEC, MCGC, DCRN , FT-VGAE, gCooL, S^3GC, Congregate, RGC, DinkNet, and GC-Flow, and 5 self-supervised GNNs,

[2]Datasets and codes of LSEnet are available at the link of https://github.com/ZhenhHuang/LSEnet

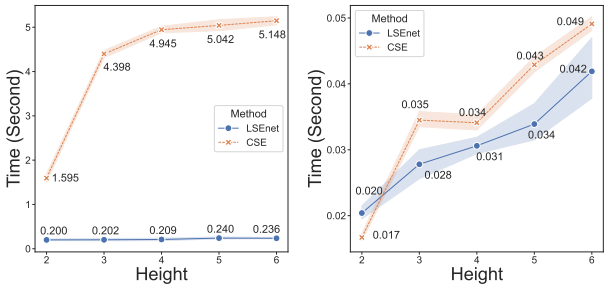
Table 2. Comparison between `LSEnet` and CSE on Cora, Citerseer, Football and Karate datasets in terms of NMI and ARI (%).

	Cora		Citeseer		Football		Karate	
	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
CSE	60.82±0.30	59.36±0.17	47.12±1.01	45.67±0.67	79.23±0.12	54.06±0.05	81.92±0.29	69.77±0.02
LSEnet	62.57±0.59	61.80±0.67	49.35±0.20	46.91±1.12	80.37±1.02	54.72±2.05	82.19±3.16	70.31±0.80
Performance Gain	**1.75**	**2.44**	**2.23**	**1.24**	**1.14**	**0.66**	**0.27**	**0.54**



(a) NMI (b) ARI

Figure 3. Parameter sensitivity on dimension of structural entropy.



(a) Cora (b) Football

Figure 4. Running time to obtain partitioning tree

i.e., VGAE, ARGA, Sublime, ProGCL and MVGRL. Additionally, we provide the results of K-Means as a reference. Datasets and baselines are detailed in Appendix C.3.

Metric & Configuration. Both Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) (Hassani & Ahmadi, 2020; Li et al., 2022) are employed as the evaluation metric for the clustering results. In `LSEnet`, we work in the Lorentz model with standard curvature $\kappa = -1$, given that Lorentz models of different curvatures are mathematically equivalent in essence. The MLP of Lorentz assigner has 3 layers. The dimension of structural entropy is a hyperparameter. The parameters are optimized by Riemannian Adam (Bécigneul & Ganea, 2019) with a 0.003 learning rate. As for the baselines, the hyperparameter setting is the same as the original papers. The cluster number K is set as the number of ground-truth classes, and we report the mean value with standard deviations of 10 independent runs. Further details are in Appendix C.4.

Comparison with State-of-the-art. Clustering results on Cora, Citeseer, AMAP and Computer datasets are collected in Table 1. Concretely, self-supervised GNNs do not generate node clusters themselves, and we apply K-means to obtain the results. In `LSEnet`, node features are projected to Lorentz model via the exponential map (Eq. 60) with respect to the origin point. `LSEnet` learns the hyperbolic partitioning tree, and node clusters are given in a divisive manner with hyperbolic distance, detailed as Algorithm 3 in Appendix C.2. As shown in Table 1, even though all the baselines except RGC have the cluster number K as input parameter, the proposed `LSEnet` achieves the best performance except on AMAP dataset and has more consistent performance, e.g., DinkNet does well on AMAP but worse on Computer and Citeseer. Without K, RGC automatically seeks K via reinforcement learning, while we present another idea of learning a partitioning tree and have better results, e.g., over 20% NMI gain on AMAP dataset. `LSEnet` takes advantage of structural entropy, encoding the self-organization of graphs, and hyperbolic geometry further benefits graph clustering, as shown in the next Sec.

6.2. Discussion on Structural Entropy

Parameter Sensitivity We examine the parameter sensitivity on the dimension of structural entropy in `LSEnet`, i.e., the height of the partitioning tree H. The clustering results in Table 1 is given by $H = 3$. Here, we vary the height H in $\{2, 3, 4, 5\}$, and report the results on Cora, Citeseer, AMAP and Computer datasets in Fig. 3. It shows that `LSEnet` generally receives performance gain when increasing the height. Also, `LSEnet` is able to obtain satisfactory results with small heights.

Comparison with Classic Structural Entropy First, the time complexity of `LSEnet` is $O(|\mathcal{V}||\mathcal{E}|)$ regardless of the dimension of structural information, while the classic structural entropy (Li & Pan, 2016), termed as CSE, scales exponentially with respect to its dimension (i.e., tree height). For example, it is $O(|\mathcal{V}|^3 \log^2|\mathcal{V}|)$ for the 3-dimensional case, which is unacceptable for large graphs. Second, on running time, Fig. 4 shows the running time to obtain the partitioning tree of different heights. CSE has a competitive time cost to `LSEnet` on the small dataset (Football), but does badly on larger datasets, e.g., at least 7× time cost to `LSEnet` on Cora. Third, we examine the effectiveness of clustering. CSE itself is unaware of node features, and cannot perform

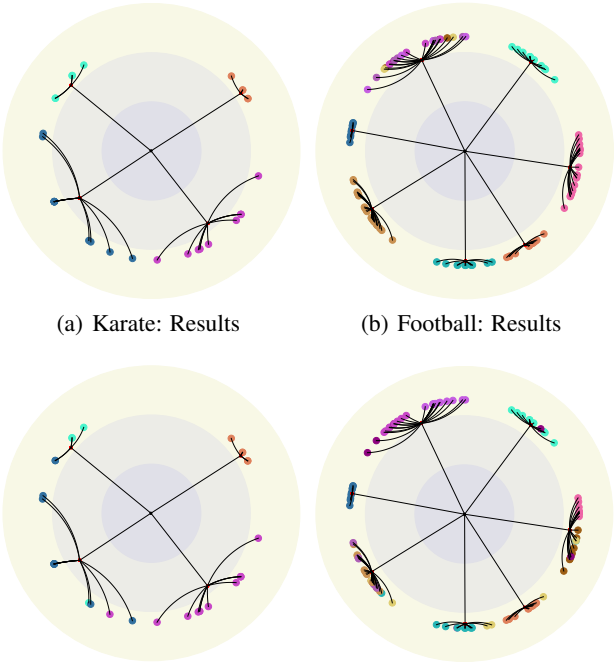
Table 3. Embedding expressiveness regarding link prediction on Cora, Citeseer, AMAP and Computer datasets. The best results are highlighted in **boldface**, and runner-ups are underlined.

	Cora	Citeseer	AMAP	Computer
GCN	91.19±0.51	90.16±0.49	90.12±0.72	93.86±0.36
SAGE	86.02±0.55	88.18±0.22	98.02±0.32	92.82±0.20
GAT	92.55±0.49	89.32±0.36	98.67±0.08	95.93±0.19
HGCN	93.60±0.37	**94.39**±0.42	98.06±0.29	96.88±0.53
LGCN	92.69±0.26	93.49±1.11	97.08±0.08	96.37±0.70
QGCN	95.22±0.29	94.31±0.73	95.17±0.45	95.10±0.03
LSEnet	**95.51**±0.60	94.32±1.51	**98.75**±0.67	**97.06**±1.02

clustering. Instead, we consider the 2-dimensional case and treat the edges between leaves (graph nodes) and layer-one nodes (regarded as clusters) as clustering results. For a fair comparison, same as CSE, we set H as 2 in LSEnet and consider the assignment as the result. We report the results on Cora, Citeseer, Football and Karate in Table 2. LSEnet achieves better results than CSE. It suggests that *hyperbolic space of LSEnet benefits node-cluster assignment.*

Embedding Expressiveness. In addition to the cluster assignment, LSEnet learns node embeddings in the Lorentz model of hyperbolic space. We evaluate the embedding expressiveness regarding *link prediction*. We compare with the popular GCN (Kipf & Welling, 2017), SAGE (Hamilton et al., 2017) and GAT (Velickovic et al., 2018) in Euclidean space, hyperbolic models including HGCN (Chami et al., 2019) and LGCN (Zhang et al., 2021), and a recent QGCN in ultra hyperbolic space (Xiong et al., 2022). Results in terms of AUC are provided in Table 3, where we set H as 3 for LSEnet. In general, hyperbolic embedding achieves superior results to the Euclidean counterpart. Also, it shows that LSEnet even outperforms the recent QGCN. That is, hyperbolic embeddings of LSEnet encode the fruitful structural information for link prediction.

Case Study & Visualization Here, we visualize hyperbolic partitioning trees of LSEnet, and discuss graph clustering on two toy examples of Football and Karate datasets. In LSEnet, the partitioning tree lives in the 2-dimensional Lorentz model, which is represented in the 3-dimensional Euclidean space and hard to visualize. To bridge this gap, we conduct stereographic projection Ψ (Bachmann et al., 2020) on the Lorentz model and obtain the corresponding Poincaré disc $\mathbb{B}^2$, which is 2-dimensional open disc and is more preferable to visualization. Specifically, with the standard curvature $\kappa = -1$, for any $\boldsymbol{x} = [x_1 \ \boldsymbol{x}_s]^\top \in \mathbb{L}^2$, we have $\Psi(\boldsymbol{x}) = \frac{\boldsymbol{x}_s}{x_1+1} \in \mathbb{B}^2$. We plot the learned clusters in Fig. 5 (a) and (b), while ground-truth classes are in Fig. 5 (c) and (d), where the clusters are denoted by different colors. As shown in the figures, the learnt clusters are well separated, and are aligned with the real clusters. The visualization of Cora is in Appendix D.



(a) Karate: Results (b) Football: Results

(c) Karate: Groundtruth (d) Football: Groundtruth

Figure 5. Visualization of hyperbolic partitioning trees.

7. Conclusion

In this paper, we study graph clustering without predefined cluster number from a new perspective of structural entropy. We formulate a new graph clustering objective (DSI) not requiring the cluster number, so that the clusters are revealed in the optimal partitioning tree given by DSI minimization. Furthermore, we present a neural LSEnet to learn the optimal tree in hyperbolic space, where we integrate node features to structural information with Lorentz graph convolution net. Extensive empirical results show the superiority of our approach on real graphs.

Acknowledgements

This work is supported in part by NSFC under grants 62202164, 62322202 and 61932002, CCF-DiDi GAIA Collaborative Research Funds for Young Scholars, and Guangdong Basic and Applied Basic Research Foundation through grant 2023B1515120020. Philip S. Yu is supported in part by NSF under grants III-2106758, and POSE-2346158.

Impact Statement

This paper presents work whose goal is to bridge the structural entropy of information theory and deep learning, and shows a new objective function for deep graph clustering, not requiring the predefined cluster number. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Almahairi, A., Ballas, N., Cooijmans, T., Zheng, Y., Larochelle, H., and Courville, A. Dynamic capacity networks. In *Proceedings of the ICML*, pp. 2549–2558. PMLR, 2016.

Bachmann, G., Bécigneul, G., and Ganea, O. Constant curvature graph convolutional networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119, pp. 486–496. PMLR, 2020.

Bécigneul, G. and Ganea, O. Riemannian adaptive optimization methods. In *Proceedings of 7th International Conference on Learning Representation (ICLR)*, 2019.

Becigneul, G. and Ganea, O.-E. Riemannian Adaptive Optimization Methods. In *International Conference on Learning Representations*, 2018.

Bianconi, G. Entropy of network ensembles. *Physical Review E*, 79(3):036114, 2009.

Braunstein, S. L., Ghosh, S., and Severini, S. The laplacian of a graph as a density matrix: a basic combinatorial approach to separability of mixed states. *Annals of Combinatorics*, 10(3):291–317, 2006.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in the 32nd NeurIPS*, pp. 4869–4880, 2019.

Chami, I., Gu, A., Chatziafratis, V., and Ré, C. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In *Advances in NeurIPS*, volume 33, pp. 15065–15076, 2020.

Chen, W., Han, X., Lin, Y., Zhao, H., Liu, Z., Li, P., Sun, M., and Zhou, J. Fully hyperbolic neural networks. In *Proceedings of the 60th ACL*, pp. 5672–5686, 2022.

Cohen-addad, V., Kanade, V., Mallmann-trenn, F., and Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM*, 66(4):26:1–26:42, 2019.

Devvrit, F., Sinha, A., Dhillon, I., and Jain, P. S^3GC: Scalable self-supervised graph clustering. In *Advances in NeurIPS*, pp. 3248–3261, 2022.

Ester, M., Kriegel, H., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd KDD*, pp. 226–231. AAAI Press, 1996.

Fu, X., Li, J., Wu, J., Sun, Q., Ji, C., Wang, S., Tan, J., Peng, H., and Yu, P. S. ACE-HGNN: adaptive curvature exploration hyperbolic graph neural network. In *Proceedings of the ICDM*, pp. 111–120. IEEE, 2021.

Fu, X., Wei, Y., Sun, Q., Yuan, H., Wu, J., Peng, H., and Li, J. Hyperbolic geometric graph representation learning for hierarchy-imbalance node classification. In *Proceedings of the ACM Web Conference 2023*, pp. 460–468. ACM, 2023.

Gershman, S. J. and Blei, D. M. A Tutorial on Bayesian Nonparametric Models, 2011.

Hamilton, W. L., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in the 30th NeurIPS*, pp. 1024–1034, 2017.

Hassani, K. and Ahmadi, A. H. K. Contrastive multi-view representation learning on graphs. In *Proceedings of the 37th ICML*, volume 119, pp. 4116–4126, 2020.

Jia, Y., Zhang, Q., Zhang, W., and Wang, X. Communitygan: Community detection with generative adversarial nets. In *Proceedings of the WWW*, pp. 784–794. ACM, 2019.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th ICLR*, 2017.

Krioukov, D. V., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguñá, M. Hyperbolic geometry of complex networks. *Physics Review E*, 82(036106), 2010.

Li, A. and Pan, Y. Structural information and dynamical complexity of networks. *IEEE Transactions on Information Theory*, 62(6):3290–3339, 2016.

Li, B., Jing, B., and Tong, H. Graph communal contrastive learning. In *Proceedings of The ACM Web Conference*, pp. 1203–1213. ACM, 2022.

Liu, Y., Liu, J., Zhang, Z., Zhu, L., and Li, A. REM: from structural entropy to community structure deception. In *Advances in the 32nd NeurIPS*, pp. 12918–12928, 2019.

Liu, Y., Tu, W., Zhou, S., Liu, X., Song, L., Yang, X., and Zhu, E. Deep graph clustering via dual correlation reduction. In *Proceedings of the 36th AAAI*, pp. 7603–7611. AAAI Press, 2022a.

Liu, Y., Zheng, Y., Zhang, D., Chen, H., Peng, H., and Pan, S. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pp. 1392–1403, 2022b.

Liu, Y., Liang, K., Xia, J., Yang, X., Zhou, S., Liu, M., Liu, X., and Li, S. Z. Reinforcement graph clustering with unknown cluster number. In *Proceedings of the 31st ACM MM*, pp. 3528–3537. ACM, 2023a.

Liu, Y., Liang, K., Xia, J., Zhou, S., Yang, X., Liu, X., and Li, S. Z. Dink-net: Neural clustering on large graphs. In *Proceedings of the ICML*, volume 202, pp. 21794–21812. PMLR, 2023b.

Liu, Y., Xia, J., Zhou, S., Yang, X., Liang, K., Fan, C., Zhuang, Y., Li, S. Z., Liu, X., and He, K. A Survey of Deep Graph Clustering: Taxonomy, Challenge, Application, and Open Resource, 2023c.

Lou, A., Katsman, I., Jiang, Q., Belongie, S. J., Lim, S., and Sa, C. D. Differentiating through the fréchet mean. In *Proceedings of the 37th ICML*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6393–6403. PMLR, 2020.

Mrabah, N., Bouguessa, M., and Ksantini, R. Escaping feature twist: A variational graph auto-encoder for node clustering. In *Proceedings of the 31st IJCAI*, pp. 3351–3357. ijcai.org, 2022.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in 30th NeurIPS*, pp. 6338–6347, 2017.

Pan, E. and Kang, Z. Multi-view contrastive graph clustering. In *Advances in NeurIPS*, pp. 2148–2159, 2021.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th IJCAI*, pp. 2609–2615. ijcai.org, 2018.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th KDD*, pp. 701–710. ACM, 2014.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *Proceedings of the 19th International Symposium on Graph Drawing (Eindhoven, The Netherlands)*, volume 7034, pp. 355–366. Springer, 2011.

Schubert, E. Stop using the elbow criterion for k-means and how to choose the number of clusters instead. *SIGKDD Explor.*, 25(1):36–42, 2023.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

Sun, L., Zhang, Z., Zhang, J., Wang, F., Peng, H., Su, S., and Yu, P. S. Hyperbolic variational graph neural network for modeling dynamic graphs. In *Proceedings of the 35th AAAI*, pp. 4375–4383, 2021.

Sun, L., Ye, J., Peng, H., and Yu, P. S. A self-supervised riemannian GNN with time varying curvature for temporal graph learning. In *Proceedings of the 31st ACM CIKM*, pp. 1827–1836, 2022a.

Sun, L., Zhang, Z., Ye, J., Peng, H., Zhang, J., Su, S., and Yu, P. S. A self-supervised mixed-curvature graph neural network. In *Proceedings of the 36th AAAI*, pp. 4146–4155, 2022b.

Sun, L., Huang, Z., Wu, H., Ye, J., Peng, H., Yu, Z., and Yu, P. S. Deepricci: Self-supervised graph structure-feature co-refinement for alleviating over-squashing. In *Proceedings of the 23rd ICDM*, pp. 558–567, 2023a.

Sun, L., Wang, F., Ye, J., Peng, H., and Yu, P. S. CONGREGATE: contrastive graph clustering in curvature spaces. In *Proceedings of the 32nd IJCAI*, pp. 2296–2305. ijcai.org, 2023b.

Sun, L., Ye, J., Peng, H., Wang, F., and Yu, P. S. Self-supervised continual graph learning in adaptive riemannian spaces. In *Proceedings of the 37th AAAI*, pp. 4633–4642, 2023c.

Sun, L., Hu, J., Li, M., and Peng, H. R-ode: Ricci curvature tells when you will be informed. In *Proceedings of the ACM SIGIR*, 2024a.

Sun, L., Hu, J., Zhou, S., Huang, Z., Ye, J., Peng, H., Yu, Z., and Yu, P. S. Riccinet: Deep clustering via a riemannian generative model. In *Proceedings of the ACM Web Conference 2024 (WWW'24)*, pp. 4071–4082, 2024b.

Sun, L., Huang, Z., Wang, Z., Wang, F., Peng, H., and Yu, P. S. Motif-aware riemannian graph neural network with generative-contrastive learning. In *Proceedings of the 38th AAAI*, pp. 9044–9052, 2024c.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *Proceedings of the 6th ICLR*, 2018.

Wang, T., Mirzazadeh, F., Zhang, X., and Chen, J. Gc-flow: A graph-based flow network for effective clustering. In *Proceedings of the ICML*, volume 202, pp. 36157–36173. PMLR, 2023.

Wu, J., Chen, X., Xu, K., and Li, S. Structural entropy guided graph hierarchical pooling. In *Proceedings of the 39th ICML*, pp. 24017–24030. PMLR, 2022.

Wu, J., Chen, X., Shi, B., Li, S., and Xu, K. SEGA: structural entropy guided anchor view for graph contrastive learning. In *Proceedings of the 40th ICML*, volume 202, pp. 37293–37312. PMLR, 2023.

Xia, J., Wu, L., Wang, G., Chen, J., and Li, S. Z. Progcl: Rethinking hard negative mining in graph contrastive learning. In *Proceedings of the ICML*, volume 162, pp. 24332–24346. PMLR, 2022.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised deep embedding for clustering analysis. In *Proceedings of the ICML*, pp. 478–487. PMLR, 2016.

Xiong, B., Zhu, S., Potyka, N., Pan, S., Zhou, C., and Staab, S. Pseudo-riemannian graph convolutional networks. In *Advances in NeurIPS*, 2022.

Yang, C., Liu, M., Wang, Z., Liu, L., and Han, J. Graph clustering with dynamic embedding. *arXiv preprint arXiv:1712.08249*, 2017.

Yang, L., Wang, Y., Gu, J., Wang, C., Cao, X., and Guo, Y. JANE: jointly adversarial network embedding. In *Proceedings of the 29th IJCAI*, pp. 1381–1387. ijcai.org, 2020.

Yang, M., Zhou, M., Ying, R., Chen, Y., and King, I. Hyperbolic representation learning: Revisiting and advancing. In *Proceedings of the 40th ICML*. PMLR, 2023a.

Yang, Z., Zhang, G., Wu, J., Yang, J., Sheng, Q. Z., Peng, H., Li, A., Xue, S., and Su, J. Minimum entropy principle guided graph neural networks. In *Proceedings of the 16th ACM WSDM*, pp. 114–122, 2023b.

Ye, Y., Lian, X., and Chen, M. Efficient exact subgraph matching via gnn-based path dominance embedding. *Proceedings of the VLDB*, 17(7):1628–1641, 2024.

Yin, H., Benson, A. R., Leskovec, J., and Gleich, D. F. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD*, pp. 555–564. ACM, 2017.

Zhang, Y., Wang, X., Shi, C., Liu, N., and Song, G. Lorentzian graph convolutional networks. In *Proceedings of the Web Conference 2021*, pp. 1249–1261. ACM / IW3C2, 2021.

Zou, D., Peng, H., Huang, X., Yang, R., Li, J., Wu, J., Liu, C., and Yu, P. S. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference*, pp. 499–510, 2023.

Appendix

The appendix is structured in four sections. **A. Proofs** on differential structural information, **B. Hyperbolic Space**, including important notions and facts in hyperbolic geometry, **C. Technical Details** on algorithms, datasets, baselines, etc., and **D. Additional Results** on real datasets.

A. Proofs

Here, we detail lemmas and theorems on the proposed **Differential Structural Information**. In particular, we prove the *equivalence, additivity, flexibility, bound* and *the connection to graph clustering*, and show the supporting lemmas.

A.1. Proof of Theorem 4.4

Theorem 4.4 (Equivalence) *The formula $\mathcal{H}^{\mathcal{T}}$ in Definition 4.3 is equivalent to the Eq. (2) given in Definition 4.1.*

Proof. From Eq. (1), we can rewrite structural information of G w.r.t all nodes of $\mathcal{T}$ at height h as follows:

$$\begin{aligned}\mathcal{H}^{\mathcal{T}}(G;h) &= -\frac{1}{V}\sum_{k=1}^{N_h} g_k^h \log_2 \frac{V_k^h}{V_{k^-}^{h-1}} \\ &= -\frac{1}{V}[\sum_{k=1}^{N_h} g_k^h \log_2 V_k^h - \sum_{k=1}^{N_h} g_k^h \log_2 V_{k^-}^{h-1}]. \end{aligned} \tag{1}$$

Then, the H-dimensional structural information of G can be written as

$$\mathcal{H}^{\mathcal{T}}(G) = -\frac{1}{V}[\sum_{h=1}^{H}\sum_{k=1}^{N_h} g_k^h \log_2 V_k^h - \sum_{h=1}^{H}\sum_{k=1}^{N_h} g_k^h \log_2 V_{k^-}^{h-1}]. \tag{2}$$

Since at height h,

$$\begin{aligned} g_k^h &= V_k^h - \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \in T_{\alpha_k^h})\mathbf{I}(j \in T_{\alpha_k^h}) w_{ij} \\ &= V_k^h - \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} S_{ik}^h S_{jk}^h w_{ij}, \end{aligned} \tag{3}$$

where we use S_{ik}^h to represent $\mathbf{I}(i \in T_{\alpha_k^h})$, and $T_{\alpha_k^h}$ is the subset of $\mathcal{V}$ corresponds the k-th node in height h.

$$V_k^h = \sum_{i=1}^{N} \mathbf{I}(i \in T_{\alpha_k^h}) d_i = \sum_{i=1}^{N} S_{ik}^h d_i. \tag{4}$$

We then have the following equation:

$$\sum_{h=1}^{H}\sum_{k=1}^{N_h} g_k^h \log_2 V_k^h = \sum_{h=1}^{H}\sum_{k=1}^{N_h} V_k^h \log_2 V_k^h - \sum_{h=1}^{H}\sum_{k=1}^{N_h} (\log_2 V_k^h) \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} S_{ik}^h S_{jk}^h w_{ij}. \tag{5}$$

Similarly,

$$\sum_{h=1}^{H}\sum_{k=1}^{N_h} g_k^h \log_2 V_{k^-}^{h-1} = \sum_{h=1}^{H}\sum_{k=1}^{N_h} V_k^h \log_2 V_{k^-}^{h-1} - \sum_{h=1}^{H}\sum_{k=1}^{N_h} (\log_2 V_{k^-}^{h-1}) \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} S_{ik}^h S_{jk}^h w_{ij}. \tag{6}$$

To find the ancestor of node k in height $h-1$, we utilize the assignment matrix C^h as follows:

$$V_{k^-}^{h-1} = \sum_{k'=1}^{N_{h-1}} \mathbf{C}_{kk'}^h V_{k'}^{h-1}. \tag{7}$$

We can easily verify that

$$S^h_{ik} = \mathbf{I}(i \in T_{\alpha^h_k}) = \sum_{\alpha^{h-1}_k}^{N_{H-1}} ... \sum_{\alpha^{h+1}_k}^{N_{h+1}} \mathbf{I}(\{i\} \subseteq T_{\alpha^{H-1}_k}) \mathbf{I}(T_{\alpha^{H-1}_k} \subseteq T_{\alpha^{H-2}_k}) ... \mathbf{I}(T_{\alpha^{h+1}_k} \subseteq T_{\alpha^h_k})$$
$$= \sum_{j_{H-1}}^{N_{H-1}} ... \sum_{j_{h+1}}^{N_{h+1}} \mathbf{C}^H_{ij_{H-1}} \mathbf{C}^{H-1}_{j_{H-1}j_{H-2}} ... \mathbf{C}^{h+1}_{j_{h+1}k}. \tag{8}$$

Since $\mathbf{C}^{H+1} = \mathbf{I}_N$, we have $\mathbf{S}^h = \prod_{h=H+1}^{h+1} \mathbf{C}^h$.

Utilizing the equations above, we have completed the proof. □

A.2. Proof of Theorem 4.7

Theorem 4.7 (Bound) *For any graph G, $\mathcal{T}^*$ is the optimal partitioning tree given by Li & Pan (2016), and $\mathcal{T}^*_{\text{net}}$ with height H is the partitioning tree given from Eq. (11). For any pair of leaf embeddings $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ of $\mathcal{T}^*_{\text{net}}$, there exists bounded real functions $\{f_h\}$ and constant c, such that for any $0 < \epsilon < 1$, $\tau \leq \mathcal{O}(1/\ln[(1-\epsilon)/\epsilon])$ and $\frac{1}{1+\exp\{-(f_h(\boldsymbol{z}_i,\boldsymbol{z}_j)-c)/\tau\}} \geq 1-\epsilon$ satisfying $i \overset{h}{\sim} j$, we have*

$$|\mathcal{H}^{\mathcal{T}^*}(G) - \mathcal{H}^{\mathcal{T}^*_{\text{net}}}(G)| \leq \mathcal{O}(\epsilon). \tag{9}$$

We first prove the existence of $\{f_h\}$ by giving a construction process. Fix $0 < \epsilon < 1$, we have

$$\sigma^h_{ij} = \frac{1}{1+\exp\{-(f_h(\boldsymbol{z}_i,\boldsymbol{z}_j)-c)/\tau\}} \geq 1-\epsilon \tag{10}$$
$$\Rightarrow f_h(\boldsymbol{z}_i,\boldsymbol{z}_j) \geq c + \tau \ln\frac{1-\epsilon}{\epsilon} \leq 1 + c, \tag{11}$$

since $\tau \leq \mathcal{O}(1/\ln[(1-\epsilon)/\epsilon])$. Similarly, when $i \overset{h}{\sim} j$ fails, we have $\sigma^h_{ij} \leq \epsilon \Rightarrow f_h(\boldsymbol{z}_i,\boldsymbol{z}_j) \leq c - \tau \ln\frac{1-\epsilon}{\epsilon} \geq c-1$. Without loss of generality, we let $c = 0$ and find that if $i \overset{h}{\sim} j$ holds, $f_h \geq 1$, otherwise, $f_h \leq -1$. If we rewrite f_h as a compose of scalar function $f_h = f(g_h(\boldsymbol{z}_i,\boldsymbol{z}_j)) + u(g_h(\boldsymbol{z}_i,\boldsymbol{z}_j)) : \mathbb{R} \to \mathbb{R}$, where g_h is a scalar function and u is a step function as $u(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases}$. If f has a discontinuity at 0 then the construction is easy. At level h, we find a node embedding $\boldsymbol{a}^h$ that is closest to the root $\boldsymbol{z}_o$ and denote the lowest common ancestor of $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ in a level less than h as $\boldsymbol{a}^h_{ij}$. Then set $g_h(\boldsymbol{z}_i,\boldsymbol{z}_j) = d_{\mathbb{L}}(\boldsymbol{z}_o,\boldsymbol{a}^h_{ij}) - d_{\mathbb{L}}(\boldsymbol{z}_o,\boldsymbol{a}^h)$. Clearly $g_h(\boldsymbol{z}_i,\boldsymbol{z}_j) \geq 0$ if $i \overset{h}{\sim} j$ holds. Then we set $f(\cdot) = \sinh(\cdot)$, i.e. $f_h(\boldsymbol{z}_i,\boldsymbol{z}_j) = \sinh(g_h(\boldsymbol{z}_i,\boldsymbol{z}_j)) + u(g_h(\boldsymbol{z}_i,\boldsymbol{z}_j))$.

If $i \overset{h}{\sim} j$ holds, clearly $f_h \geq 1$, otherwise, $f_h \leq -1$. Since the distance between all pairs of nodes is bounded, g_h is also bounded. All the properties are verified, so such $\{f_h\}$ exist.

Then we have the following lemma.

Lemma A.1. *For a weighted Graph $G = (V, E)$ with a weight function w, and a partitioning tree $\mathcal{T}$ of G with height H, we can rewrite the formula of H-dimensional structural information of G w.r.t $\mathcal{T}$ as follows:*

$$\mathcal{H}^{\mathcal{T}}(G) = -\frac{1}{V}\sum_{h=1}^{H}\sum_{i=1}^{N}(d_i - \sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij}) \cdot \log_2(\frac{\sum_l^N \mathbf{I}(i \overset{h}{\sim} l) d_l}{\sum_l^N \mathbf{I}(i \overset{h-1}{\sim} l) d_l}), \tag{12}$$

where $\mathbf{I}(\cdot)$ is the indicator function.

Proof. We leave the proof of Lemma A.1 in Appendix A.5. □

Now we start our proof of Theorem 4.7.

Proof. We then focus on the absolute difference between $H^{\mathcal{T}^*}(G)$ and $H^{\mathcal{T}^*_{\text{net}}}(G)$

$$\begin{aligned}|\mathcal{H}^{\mathcal{T}^*}(G) - H^{\mathcal{T}^*_{\text{net}}}(G)| &\leq |\mathcal{H}^{\mathcal{T}^*}(G) - \mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z}^*;\Theta^*)| + |\mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z}^*;\Theta^*) - H^{\mathcal{T}^*_{\text{net}}}(G)| \\ &\leq 2\sup_{\mathbf{Z},\mathbf{\Theta}}|\mathcal{H}^{\mathcal{T}}(G) - \mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z};\Theta)|, \end{aligned} \tag{13}$$

where $\mathcal{T} = \text{decode}(\mathbf{Z})$. Recall the Eq. (12), we can divide them into four corresponding parts, denoted as A, B, C, and D respectively. Then, we have

$$\begin{aligned}|\mathcal{H}^{\mathcal{T}}(G) - \mathcal{H}^{\mathcal{T}_{\text{net}}}(G;\mathbf{Z};\Theta)| &= -\frac{1}{\text{Vol}(G)}[A+B+C+D] \\ &\leq -\frac{1}{\text{Vol}(G)}[|A|+|B|+|C|+|D|]. \end{aligned} \tag{14}$$

For part A, give fixed i and k, we assume $i \overset{h}{\sim} l$ holds for some $l_1, ..., l_{m_i}$.

$$\begin{aligned}|A| &= |\sum_{h=1}^{H}\sum_{i}^{N} d_i \log_2(\sum_{l}^{N}\mathbf{I}(i \overset{h}{\sim} l)d_l) - \sum_{h=1}^{H}\sum_{i}^{N} d_i \log_2(\sum_{l}^{N}\sigma_{il}^h d_l)| \\ &\leq |\sum_{h=1}^{H}\sum_{i}^{N} d_i \log_2 \frac{\sum_{c=1}^{m_i} d_{l_c}}{\sum_{c=1}^{m_i}(1-\epsilon)d_{l_c} + \sum_{\bar{c}}\epsilon d_{l_{\bar{c}}}}| \\ &\leq |\sum_{h=1}^{H}\sum_{i}^{N} d_i \log_2 \frac{\sum_{c=1}^{m_i} d_{l_c}}{\sum_{c=1}^{m_i}(1-\epsilon)d_{l_c}}| \\ &= |\sum_{h=1}^{H}\sum_{i}^{N} d_i \log_2 \frac{1}{1-\epsilon}| \\ &= H\,\text{Vol}(G)\log_2\frac{1}{1-\epsilon}. \end{aligned} \tag{15}$$

For part B, we still follow the assumption above and assume $i \overset{h}{\sim} j$ holds for some $j_e, ..., j_{n_i}$ in the neighborhood of i.

$$\begin{aligned}|B| &= |-\sum_{h=1}^{H}\sum_{i}^{N}[\log_2(\sum_{l}^{N}\mathbf{I}(i \overset{h}{\sim} l)d_l)\cdot\sum_{j\in\mathcal{N}(i)}\mathbf{I}(i \overset{h}{\sim} j)w_{ij}] + \sum_{h=1}^{H}\sum_{i}^{N}[\log_2(\sum_{l}^{N}\sigma_{il}^h d_l)\cdot\sum_{j\in\mathcal{N}(i)}\sigma_{ij}^h w_{ij}]| \\ &\leq |\sum_{h=1}^{H}\sum_{i}^{N}\log_2(\sum_{c=1}^{m_i}d_{l_c})\cdot\sum_{e=1}^{n_i}w_{ij_e} - \log_2(\sum_{c=1}^{m_i}(1-\epsilon)d_{l_c} + \sum_{\bar{c}}\epsilon d_{l_{\bar{c}}})\cdot[\sum_{e=1}^{n_i}(1-\epsilon)w_{ij_e} + \sum_{\bar{e}}\epsilon w_{ij_{\bar{e}}}]| \\ &\leq |\sum_{h=1}^{H}\sum_{i}^{N}\log_2(\sum_{c=1}^{m_i}d_{l_c})\cdot\sum_{e=1}^{n_i}w_{ij_e} - \log_2(\sum_{c=1}^{m_i}(1-\epsilon)d_{l_c})\cdot\sum_{e=1}^{n_i}(1-\epsilon)w_{ij_e}| \\ &= |\sum_{h=1}^{H}\sum_{i}^{N}(\sum_{e=1}^{n_i}w_{ij_e})[\log_2(\sum_{c=1}^{m_i}d_{l_c} - (1-\epsilon)\log_2(1-\epsilon) - (1-\epsilon)\log_2(\sum_{c=1}^{m_i}d_{l_c}]| \\ &= |\sum_{h=1}^{H}\sum_{i}^{N}(\sum_{e=1}^{n_i}w_{ij_e})[\epsilon\log_2(\sum_{c=1}^{m_i}d_{l_c} - (1-\epsilon)\log_2(1-\epsilon)]| \\ &\leq (1-\epsilon)\log_2\frac{1}{1-\epsilon}\sum_{h=1}^{H}\sum_{i}^{N}\sum_{e=1}^{n_i}w_{ij_e} \\ &\leq (1-\epsilon)\log_2\frac{1}{1-\epsilon}\sum_{h=1}^{H}\sum_{i}^{N}d_i \\ &= (1-\epsilon)HV\log_2\frac{1}{1-\epsilon}. \end{aligned} \tag{16}$$

Similarly, we can get the same results for parts C and D as A and B respectively. Then,

$$|\mathcal{H}^{\mathcal{T}}(G) - \mathcal{H}^{\mathcal{T}_{\text{net}}}(G; \mathbf{Z}; \Theta)| \leq 2H(2-\epsilon)\log_2 \frac{1}{1-\epsilon}. \tag{17}$$

Substituting into Equation.13, we obtain

$$|\mathcal{H}^{\mathcal{T}^*}(G) - H^{\mathcal{T}^*_{\text{net}}}(G)| \leq 4H(2-\epsilon)\log_2 \frac{1}{1-\epsilon}. \tag{18}$$

Since the limitation is a constant number,

$$\lim_{\epsilon \to 0} \frac{(2-\epsilon)\log_2 \frac{1}{1-\epsilon}}{\epsilon} = 2, \tag{19}$$

we have $|H^{\mathcal{T}^*}(G) - H^{\mathcal{T}^*_{\text{net}}}(G)| \leq \mathcal{O}(\epsilon)$, which completes the proof. □ □

A.3. Proof of Lemma 4.5

Lemma 4.5 (Additive) *The one-dimensional structural entropy of G can be decomposed as follows*

$$\mathcal{H}^1(G) = \sum_{h=1}^{H} \sum_{j=1}^{N_{h-1}} \frac{V_j^{h-1}}{V} E([\frac{C_{kj}^h V_k^h}{V_j^{h-1}}]_{k=1,\ldots,N_h}), \tag{20}$$

where $E(p_1, ..., p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$ *is the entropy.*

Proof. To verify the equivalence to the one-dimensional structural entropy, we expand the above formula.

$$\mathcal{H}(G) = -\sum_{h=1}^{H} \sum_{j=1}^{N_{h-1}} \frac{V_j^{h-1}}{V} \sum_{k=1}^{N_h} \frac{C_{kj}^h V_k^h}{V_j^{h-1}} \log_2 \frac{C_{kj}^h V_k^h}{V_j^{h-1}} \tag{21}$$

$$= -\frac{1}{V} \sum_{h=1}^{H} \sum_{k=1}^{N_h} \sum_{j=1}^{N_{h-1}} C_{kj}^h V_k^h \log_2 \frac{C_{kj}^h V_k^h}{V_j^{h-1}} \tag{22}$$

$$= -\frac{1}{V} \sum_{h=1}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V_{k^-}^{h-1}}. \tag{23}$$

We omit some j that make $C_{kj}^h = 0$ so that the term V_j^{h-1} only exists in the terms where $C_{kj}^h = 1$, meaning that V_j^{h-1} exists if and only if $V_j^{h-1} = V_{k^-}^{h-1}$ since we summing over all k. Continue the process that

$$\mathcal{H}(G) = -\frac{1}{V} \sum_{h=1}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V} + \frac{1}{V} \sum_{h=1}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_{k^-}^{h-1}}{V} \tag{24}$$

$$= -\frac{1}{V} \sum_{i=1}^{N} d_i \log_2 \frac{d_i}{V} - \frac{1}{V} \sum_{h=1}^{H-1} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V} + \frac{1}{V} \sum_{h=1}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_{k^-}^{h-1}}{V} \tag{25}$$

$$= -\frac{1}{V} \sum_{i=1}^{N} d_i \log_2 \frac{d_i}{V} - \frac{1}{V} \sum_{h=1}^{H-1} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V} + \frac{1}{V} \sum_{h=2}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_{k^-}^{h-1}}{V}. \tag{26}$$

From the first line to the second line, we separate from the term when $h = H$. For the third line, we eliminate the summation term in $h = 1$, since when $h = 1$, in the last term, $\log_2 \frac{V_{k^-}^0}{V} = \log_2 \frac{V}{V} = 0$.

Then let us respectively denote $-\sum_{h=1}^{H-1} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_k^h}{V}$ and $\sum_{h=2}^{H} \sum_{k=1}^{N_h} V_k^h \log_2 \frac{V_{k^-}^{h-1}}{V}$ as A and B, using the trick

about C_{kj}^h like above, we have

$$A+B=\sum_{h=2}^{H}[(\sum_{k=1}^{N_h}V_k^h\log_2\frac{V_{k^-}^{h-1}}{V})-(\sum_{j=1}^{N_{h-1}}V_j^{h-1}\log_2\frac{V_j^{h-1}}{V})] \tag{27}$$

$$=\sum_{h=2}^{H}[\sum_{k=1}^{N_h}V_k^h\log_2\frac{V_{k^-}^{h-1}}{V}-\sum_{j=1}^{N_{h-1}}\sum_{k=1}^{N_h}C_{kj}^hV_k^h\log_2\frac{V_j^{h-1}}{V})] \tag{28}$$

$$=\sum_{h=2}^{H}[\sum_{k=1}^{N_h}V_k^h\log_2\frac{V_{k^-}^{h-1}}{V}-\sum_{k=1}^{N_h}V_k^h\log_2\frac{V_{k^-}^{h-1}}{V}] \tag{29}$$

$$=0.$$

Thus

$$\mathcal{H}(G)=-\frac{1}{V}\sum_{i=1}^{N}d_i\log_2\frac{d_i}{V}, \tag{30}$$

which is exactly the one-dimensional structural entropy $\mathcal{H}^1(G)$ of G. The proof is completed. □

A.4. Proof of Theorem 4.6

Theorem 4.6 (Connection to Graph Clustering) *Given a graph $G=(\mathcal{V},\mathcal{E})$ with w, the normalized H-structural entropy of graph G is defined as $\tau(G;H)=\mathcal{H}^H(G)/\mathcal{H}^1(G)$, and $\Phi(G)$ is the graph conductance. With the additivity of DSE* (Lemma 4.5)*, the following inequality holds,*

$$\tau(G;H)\geq\Phi(G). \tag{31}$$

Proof. Recall the formula of H-dimensional structural information of G for a partitioning tree $\mathcal{T}$ in Definition 4.3. For the conductance $\phi_{h,k}$ of k-th node in $\mathcal{T}$ as height h, following the definition of graph conductance, we have

$$\phi_{h,k}=\frac{\sum_{i\in T_{h,k},j\notin T_{h,k}}w_{ij}}{\min\{V_k^h,V-V_k^h\}}=\frac{V_k^h-\sum_{(i,j)\in\mathcal{E}}s_{ik}^hs_{jk}^hw_{ij}}{V_k^h}. \tag{32}$$

Without loss of generality, we assume that $V_k^h\leq\frac{1}{2}V$ such that $\min\{V_k^h,V-V_k^h\}=V_k^h$. From Definition 4.3, we have

$$\mathcal{H}^{\mathcal{T}}(G)=-\frac{1}{V}\sum_{h=1}^{H}\sum_{k=1}^{N_h}\phi_{h,k}V_k^h\log_2\frac{V_k^h}{V_{k^-}^{h-1}} \tag{33}$$

$$\geq-\Phi(G)\sum_{h=1}^{H}\sum_{k=1}^{N_h}\frac{V_k^h}{V}\log_2\frac{V_k^h}{V_{k^-}^{h-1}} \tag{34}$$

$$=-\Phi(G)\sum_{h=1}^{H}\sum_{k=1}^{N_h}\frac{\sum_{j=1}^{N_{h-1}}C_{kj}^hV_k^h}{V}\log_2\frac{V_k^h}{V_{k^-}^{h-1}} \tag{35}$$

$$=-\Phi(G)\sum_{h=1}^{H}\sum_{k=1}^{N_h}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\frac{C_{kj}^hV_k^h}{V_j^{h-1}}\log_2\frac{V_k^h}{V_{k^-}^{h-1}} \tag{36}$$

$$=-\Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\sum_{k=1}^{N_h}\frac{C_{kj}^hV_k^h}{V_j^{h-1}}\log_2\frac{V_k^h}{\sum_{m=1}^{N_{h-1}}C_{km}^hV_m^{h-1}} \tag{37}$$

$$=-\Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\sum_{k=1}^{N_h}\frac{C_{kj}^hV_k^h}{V_j^{h-1}}\log_2\frac{C_{kj}^hV_k^h}{C_{kj}^h\sum_{m=1}^{N_{h-1}}C_{km}^hV_m^{h-1}} \tag{38}$$

$$-\Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\sum_{k=1}^{N_h}\frac{C_{kj}^h V_k^h}{V_j^{h-1}}\log_2\frac{C_{kj}^h V_k^h}{C_{kj}^h\sum_{m=1}^{N_{h-1}}C_{km}^h V_m^{h-1}} \tag{39}$$

$$=-\Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\sum_{k=1}^{N_h}\frac{C_{kj}^h V_k^h}{V_j^{h-1}}\log_2\frac{C_{kj}^h V_k^h}{V_j^{h-1}} \tag{40}$$

$$=\Phi(G)\sum_{h=1}^{H}\sum_{j=1}^{N_{h-1}}\frac{V_j^{h-1}}{V}\mathrm{Ent}([\frac{C_{kj}^h V_k^h}{V_j^{h-1}}]_{k=1,\dots,N_h}) \tag{41}$$

$$=\Phi(G)\mathcal{H}^1(G). \tag{42}$$

To get the final results, we use the inverse trick about C_{kj}^h mentioned in Appendix A.3. So the normalized H-dimensional structural information of G w.r.t. $\mathcal{T}$ satisfies

$$\tau(G;\mathcal{T})=\frac{\mathcal{H}^{\mathcal{T}}(G)}{\mathcal{H}^1(G)}\geq\Phi(G). \tag{43}$$

Since this inequality holds for every H-height partitioning tree $\mathcal{T}$ of G, $\tau(G;H)\geq\Phi(G)$ holds. □

A.5. Proof of Lemma A.1

Lemma A.1 *For a weighted Graph $G=(V,E)$ with a weight function w, and a partitioning tree $\mathcal{T}$ of G with height H, we can rewrite the formula of H-dimensional structural information of G w.r.t $\mathcal{T}$ as follows:*

$$\mathcal{H}^{\mathcal{T}}(G;H)=-\frac{1}{V}\sum_{h=1}^{H}\sum_{i=1}^{N}(d_i-\sum_{j\in\mathcal{N}(i)}\mathbf{I}(i\overset{h}{\sim}j)w_{ij})\cdot\log_2(\frac{\sum_l^N\mathbf{I}(i\overset{h}{\sim}l)d_l}{\sum_l^N\mathbf{I}(i\overset{h-1}{\sim}l)d_l}), \tag{44}$$

where $\mathbf{I}(\cdot)$ is the indicator function.

Proof. From Equation.1, we can rewrite structural information of G w.r.t all nodes of $\mathcal{T}$ at height h as follows:

$$\begin{aligned}H^{\mathcal{T}}(G;h)&=-\frac{1}{V}\sum_{k=1}^{N_h}g_k^h\log_2\frac{V_k^h}{V_{k^-}^{h-1}}\\&=-\frac{1}{V}[\sum_{k=1}^{N_h}g_k^h\log_2 V_k^h-\sum_{k=1}^{N_h}g_k^h\log_2 V_{k^-}^{h-1}].\end{aligned} \tag{45}$$

Then the H-dimensional structural information of G can be written as

$$\mathcal{H}^{\mathcal{T}}(G)=-\frac{1}{V}[\sum_{h=1}^{H}\sum_{k=1}^{N_h}g_k^h\log_2 V_k^h-\sum_{h=1}^{H}\sum_{k=1}^{N_h}g_k^h\log_2 V_{k^-}^{h-1}]. \tag{46}$$

Since

$$g_k^h=V_k^h-\sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)}\mathbf{I}(i\in T_{\alpha_k^h})\mathbf{I}(j\in T_{\alpha_k^h})w_{ij}, \tag{47}$$

$$V_k^h=\sum_{i}^{N}\mathbf{I}(i\in T_{\alpha_k^h})d_i, \tag{48}$$

we have the following equation with the help of the equivalence relationship $i \overset{h}{\sim} j$:

$$\sum_{h=1}^{H}\sum_{k=1}^{N_h} g_k^h \log_2 V_k^h = \sum_{h=1}^{H}\sum_{k=1}^{N_h} V_k^h \log_2 V_k^h - \sum_{h=1}^{H}\sum_{k=1}^{N_h} (\log_2 V_k^h) \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij}. \tag{49}$$

Then we focus on the above equation term by term. If we fix the height of h and the node i of G, then the node α to which i belongs and its ancestors α^- in $\mathcal{T}$ are also fixed. Let's denote V_k^h as V_α where $h(\alpha) = h$.

$$\begin{aligned}\sum_{h=1}^{H}\sum_{h(\alpha)=h} V_\alpha \log_2 V_\alpha &= \sum_{h=1}^{H}\sum_{h(\alpha)=h}\sum_{i=1}^{N} \mathbf{I}(i \in T_\alpha) d_i \log_2 (\sum_{l=1}^{N} \mathbf{I}(l \in T_\alpha) d_l) \\ &= \sum_{h=1}^{H}\sum_{i=1}^{N} d_i \log_2 (\sum_{l=1}^{N} \mathbf{I}(i \overset{h}{\sim} l) d_l),\end{aligned} \tag{50}$$

$$\sum_{h=1}^{H}\sum_{h(\alpha)=h} (\log_2 V_\alpha) \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij} = \sum_{h=1}^{H}\sum_{i=1}^{N} [\log_2 (\sum_{l=1}^{N} \mathbf{I}(i \overset{h}{\sim} l) d_l)] \cdot \sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij}. \tag{51}$$

Similarly,

$$\sum_{h=1}^{H}\sum_{h(\alpha)=h} g_\alpha \log_2 V_{\alpha^-} = \sum_{h=1}^{H}\sum_{h(\alpha)=h} V_\alpha \log_2 V_{\alpha^-} - \sum_{h=1}^{H}\sum_{h(\alpha)=h} (\log_2 V_{\alpha^-}) \sum_{i}^{N}\sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij}, \tag{52}$$

$$\begin{aligned}\sum_{h=1}^{H}\sum_{h(\alpha)=h} V_\alpha \log_2 V_\alpha &= \sum_{h=1}^{H}\sum_{h(\alpha)=h}\sum_{i=1}^{N} \mathbf{I}(i \in T_\alpha) d_i \log_2 (\sum_{l=1}^{N} \mathbf{I}(l \in T_{\alpha^-}) d_l) \\ &= \sum_{h=1}^{H}\sum_{i=1}^{N} d_i \log_2 (\sum_{l=1}^{N} \mathbf{I}(i \overset{h-1}{\sim} l) d_l),\end{aligned} \tag{53}$$

$$\sum_{h=1}^{H}\sum_{h(\alpha)=h} (\log_2 V_{\alpha^-}) \sum_{i=1}^{N}\sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij} = \sum_{h=1}^{H}\sum_{i=1}^{N} [\log_2 (\sum_{l=1}^{N} \mathbf{I}(i \overset{h-1}{\sim} l) d_l)] \cdot \sum_{j\in\mathcal{N}(i)} \mathbf{I}(i \overset{h}{\sim} j) w_{ij}. \tag{54}$$

Utilizing the equations above, we have completed the proof. □

A.6. Proof of Theorem 4.8

Theorem 4.8 (Flexible) *$\forall G$, given a partitioning tree $\mathcal{T}$ and adding a node β to get a relaxed partitioning tree $\mathcal{T}'$, the structural information remains unchanged, $H^{\mathcal{T}}(G) = H^{\mathcal{T}'}(G)$, if one of the following conditions holds:*

1. *β as a leaf node, and the corresponding node subset T_β is an empty set.*
2. *β is inserted between node α and its children nodes so that the corresponding node subsets $T_\beta = T_\alpha$.*

Proof. For the first case, since β is a leaf node corresponds an empty subset of $\mathcal{V}$, $g_\beta = 0$, then the structural information of β is $\mathcal{H}^{\mathcal{T}}(G;\beta) = 0$. For the second case, we notice the term $\log_2 \frac{V_\alpha}{V_{\alpha^-}}$ in Equation 1. Since the volume of T_α equals the one of T_β, the term vanishes to 0. □

B. Hyperbolic Space

B.1. Riemannian Manifold

Riemannian manifold is a real and smooth manifold $\mathbb{M}$ equipped with Riemannian metric tensor $g_{\boldsymbol{x}}$ on the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathbb{M}$ at point $\boldsymbol{x}$. A Riemannian metric assigns to each $\boldsymbol{x} \in \mathbb{M}$ a positive-definite inner product $g_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathbb{M} \times \mathcal{T}_{\boldsymbol{x}}\mathbb{M} \to \mathbb{R}$, which induces a norm defined as $|\cdot| : \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{M} \mapsto \sqrt{g_{\boldsymbol{x}}(\boldsymbol{v}, \boldsymbol{v})} \in \mathbb{R}$.

An exponential map at point $\boldsymbol{x} \in \mathbb{M}$ is denoted as $\mathrm{Exp}_{\boldsymbol{x}}(\cdot) : \boldsymbol{u} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{M} \mapsto \mathrm{Exp}_{\boldsymbol{x}}(\boldsymbol{u}) \in \mathbb{M}$. It takes a tangent vector $\boldsymbol{u}$ in the tangent space at $\boldsymbol{x}$ and transforms $\boldsymbol{x}$ along the geodesic starting at $\boldsymbol{x}$ in the direction of $\boldsymbol{u}$.

A logarithmic map at point $\boldsymbol{x}$ is the inverse of the exponential map at point $\boldsymbol{x}$, which maps the point $\boldsymbol{y} \in \mathbb{M}$ to a vector $\boldsymbol{v}$ in the tangent space at $\boldsymbol{x}$.

B.2. Models of Hyperbolic Space

Poincaré ball model The d-dimensional Poincaré ball model with constant negative curvature κ, is defined within a d-dimensional hypersphere, formally denoted as $\mathbb{B}_\kappa^d = \{\boldsymbol{x} \in \mathbb{R}^d | \|\boldsymbol{x}\|^2 = -\frac{1}{\kappa}\}$. The Riemannian metric tensor at $\boldsymbol{x}$ is $g_{\boldsymbol{x}}^\kappa = (\lambda_{\boldsymbol{x}}^\kappa)^2 g_{\mathbb{E}} = \frac{4}{(1+\kappa\|\boldsymbol{x}\|^2)^2} g_{\mathbb{E}}$, where $g_{\mathbb{E}} = \mathbf{I}$ is the Euclidean metric. The distance function is given by

$$d_{\mathbb{B}}^\kappa(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\sqrt{-\kappa}} \tanh^{-1}(\|(-\boldsymbol{x}) \oplus_\kappa \boldsymbol{y}\|), \tag{55}$$

where $\oplus_\kappa$ is the Möbius addition

$$\boldsymbol{x} \oplus_\kappa \boldsymbol{y} = \frac{(1 - 2\kappa\boldsymbol{x}^\top\boldsymbol{y} - \kappa\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 + \kappa\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 - 2\kappa\boldsymbol{x}^\top\boldsymbol{y} + \kappa^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2}. \tag{56}$$

The exponential and logarithmic maps of Pincaré ball model are defined as

$$\mathrm{Exp}_{\boldsymbol{x}}^\kappa(\boldsymbol{v}) = \boldsymbol{x} \oplus_\kappa \left(\frac{1}{\sqrt{-\kappa}} \tanh\left(\sqrt{-\kappa}\frac{\lambda_{\boldsymbol{x}}^\kappa\|\boldsymbol{v}\|}{2}\right)\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\right) \tag{57}$$

$$\log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) = \frac{2}{\sqrt{-\kappa}\lambda_{\boldsymbol{x}}^\kappa} \tanh^{-1}(\sqrt{-\kappa}\|(-\boldsymbol{x}) \oplus_\kappa \boldsymbol{y}\|)\frac{(-\boldsymbol{x}) \oplus_\kappa \boldsymbol{y}}{\|(-\boldsymbol{x}) \oplus_\kappa \boldsymbol{y}\|}. \tag{58}$$

Lorentz model The d-dimensional Lorentz model equipped with constant curvature $\kappa < 0$ and Riemannian metric tensor $\mathbf{R} = \mathrm{Diag}(\text{-}1, 1, \ldots, 1)$, is defined in $(d+1)$-dimensional Minkowski space whose origin is $(\sqrt{-1/\kappa}, 0, ..., 0)$, i.e., $\mathbb{L}_\kappa^d = \{\boldsymbol{x} \in \mathbb{R}^{d+1} | \langle \boldsymbol{x}, \boldsymbol{x}\rangle_{\mathbb{L}} = \frac{1}{\kappa}\}$, where the inner product is $\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}} = -x_0y_0 + \sum_{i=1}^d x_iy_i = \boldsymbol{x}^T\mathbf{R}\boldsymbol{y}$. The distance between two points is given by

$$d_{\mathbb{L}}(\boldsymbol{x}, \boldsymbol{y}) = \cosh^{-1}(-\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}}). \tag{59}$$

The tangent space at $\boldsymbol{x} \in \mathbb{L}_\kappa^d$ is the set of $\boldsymbol{y} \in \mathbb{L}_\kappa^d$ such that orthogonal to $\boldsymbol{x}$ w.r.t. Lorentzian inner product, denotes as $\mathcal{T}_{\boldsymbol{x}}\mathbb{L}_\kappa^d = \{\boldsymbol{y} \in \mathbb{R}^{d+1} | \langle \boldsymbol{y}, \boldsymbol{x}\rangle_{\mathbb{L}} = 0\}$. The exponential and logarithmic maps at $\boldsymbol{x}$ are defined as

$$\mathrm{Exp}_{\boldsymbol{x}}^\kappa(\boldsymbol{u}) = \cosh(\sqrt{-\kappa}\|\boldsymbol{u}\|_{\mathbb{L}})\boldsymbol{x} + \sinh(\sqrt{-\kappa}\|\boldsymbol{u}\|_{\mathbb{L}})\frac{\boldsymbol{u}}{\sqrt{-\kappa}\|\boldsymbol{u}\|_{\mathbb{L}}} \tag{60}$$

$$\log_{\boldsymbol{x}}^\kappa(\boldsymbol{y}) = \frac{\cosh^{-1}(\kappa\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}})}{\sqrt{(\kappa\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}})^2 - 1}}(\boldsymbol{y} - \kappa\langle \boldsymbol{x}, \boldsymbol{y}\rangle_{\mathbb{L}}\boldsymbol{x}). \tag{61}$$

B.3. Tree and Hyperbolic Space

We denote the embedding of node i of graph G in $\mathbb{H}$ is ϕ_i. The following definitions and theorem are from (Sarkar, 2011).

Definition B.1 (Delaunay Graph)**.** Given a set of vertices in $\mathbb{H}$ their Delaunay graph is one where a pair of vertices are neighbors if their Voronoi cells intersect.

Definition B.2 (Delaunay Embedding of Graphs)**.** Given a graph G, its Delaunay embedding in $\mathbb{H}$ is an embedding of the vertices such that their Delaunay graph is G.

Definition B.3 (β separated cones)**.** Suppose cones $C(\overrightarrow{\phi_j\phi_i}, \alpha)$ and $C(\overrightarrow{\phi_j\phi_x}, \gamma)$ are adjacent with the same root φj. Then the cones are β separated if the two cones are an angle 2β apart. That is, for arbitrary points $p \in C(\overrightarrow{\phi_j\phi_i}, \alpha)$ and $q \in C(\overrightarrow{\phi_j\phi_x}, \gamma)$, the angle $\angle p\phi_j q > 2\beta$.

Lemma B.4. *If cones $C(\overrightarrow{\phi_j\phi_i}, \alpha)$ and $C(\overrightarrow{\phi_j\phi_x}, \gamma)$ are β separated and $\phi_r \in C(\overrightarrow{\phi_j\phi_i}, \alpha)$ and $\phi_s \in C(\overrightarrow{\phi_j\phi_x}, \gamma)$ then there is a constant ν depending only on β such that $|\phi_r\phi_j|_{\mathbb{H}} + |\phi_s\phi_j|_{\mathbb{H}} > |\phi_r\phi_s|_{\mathbb{H}} > |\phi_r\phi_j|_{\mathbb{H}} + |\phi_s\phi_j|_{\mathbb{H}} - \nu$.*

Theorem B.5 ((Sarkar, 2011))**.** *If all edges of $\mathcal{T}$ are scaled by a constant factor $\tau \geq \eta_{\max}$ such that each edge is longer than $\nu\frac{(1+\epsilon)}{\epsilon}$ and the Delaunay embedding of $\mathcal{T}$ is β separated, then the distortion over all vertex pairs is bounded by $1+\epsilon$.*

Following the above Theorem, we know that a tree can be embedded into hyperbolic space with an arbitrarily low distortion.

B.4. Midpoint in the Manifold

Let $(\mathbb{M}, g)$ be a Riemannian manifold, and $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$ are points on the manifold. The Fréchet variance at point $\boldsymbol{\mu} \in \mathbb{M}$ of these points are given by

$$\Psi(\boldsymbol{\mu}) = \sum_i d^2(\boldsymbol{\mu}, \boldsymbol{x}_i). \tag{62}$$

If there is a point $\boldsymbol{p}$ locally minimizes the variance, it is called Fréchet mean. Generally, if we assign each $\boldsymbol{x}_i$ a weight w_i, the Fréchet mean can be formulated as

$$\boldsymbol{p} = \arg\min_{\boldsymbol{\mu}\in\mathbb{M}} \sum_i w_i d^2(\boldsymbol{\mu}, \boldsymbol{x}_i). \tag{63}$$

Note that, d is the canonical distance in the manifold.

Theorem 5.2 (Arithmetic Mean as Geometric Centroid) *In hyperbolic space $\mathbb{L}^{\kappa,d_\mathcal{T}}$, for any set of $\{\boldsymbol{z}_i^h\}$, the arithmetic mean of $\boldsymbol{z}_j^{h-1} = \frac{1}{\sqrt{-\kappa}} \sum_{i=1}^n \frac{c_{ij}}{|\|\sum_{l=1}^n c_{lj}\boldsymbol{z}_l^h\|_{\mathbb{L}}|} \boldsymbol{z}_i^h$ is the manifold $\boldsymbol{z}_j^{h-1} \in \mathbb{L}^{\kappa,d_\mathcal{T}}$, and is the close-form solution of the geometric centroid specified in the minimization of Eq. (18).*

Proof. Since $d_{\mathbb{L}}^2(\boldsymbol{z}_i^h, \boldsymbol{z}_j^{h-1}) = \|\boldsymbol{z}_i^h - \boldsymbol{z}_j^{h-1}\|_{\mathbb{L}}^2 = \|\boldsymbol{z}_i^h\|_{\mathbb{L}}^2 + \|\boldsymbol{z}_j^{h-1}\|_{\mathbb{L}}^2 - 2\langle \boldsymbol{z}_i^h, \boldsymbol{z}_j^{h-1}\rangle_{\mathbb{L}} = 2/\kappa - 2\langle \boldsymbol{z}_i^h, \boldsymbol{z}_j^{h-1}\rangle_{\mathbb{L}}$, minimizing Eq. (18) is equivalent to maximizing $\sum_i c_{ij}\langle \boldsymbol{z}_i^h, \boldsymbol{z}_j^{h-1}\rangle_{\mathbb{L}} = \langle \sum_i c_{ij}\boldsymbol{z}_i^h, \boldsymbol{z}_j^{h-1}\rangle_{\mathbb{L}}$. To maximize the Lorentz inner product, $\boldsymbol{z}_j^{h-1}$ must be $\eta \sum_i c_{ij}\boldsymbol{z}_i^h$ for some positive constant η. Then the inner product will be

$$\langle \boldsymbol{z}_j^{h-1}, \boldsymbol{z}_j^{h-1}\rangle_{\mathbb{L}} = \eta^2 \langle \sum_i c_{ij}\boldsymbol{z}_i^h, \sum_i c_{ij}\boldsymbol{z}_i^h\rangle_{\mathbb{L}} = \frac{1}{\kappa}. \tag{64}$$

So η will be $\frac{1}{\sqrt{-\kappa}|\|\sum_i c_{ij}\boldsymbol{z}_i^h\|_{\mathbb{L}}|}$, the proof is completed. □

Remark. In fact, the geometric centroid in Theorem 5.2 is also equivalent to the gyro-midpoint in Poincaré ball model of hyperbolic space.

Proof. Let $\boldsymbol{z}_i^h = [t_i^h, (\boldsymbol{s}_i^h)^T]^T$, substitute into Theorem 5.2, we have

$$\boldsymbol{z}_j^{h-1} = \frac{1}{\sqrt{-\kappa}} \sum_{i=1}^n \frac{c_{ij}}{|\|\sum_{l=1}^n c_{lj}\boldsymbol{z}_l^h\|_{\mathbb{L}}|} \boldsymbol{z}_i^h \tag{65}$$

$$= \frac{1}{\sqrt{-\kappa}} \frac{[\sum_i c_{ij}t_i^h, \sum_i c_{ij}\boldsymbol{s}_i^h]^T}{\sqrt{(\sum_i c_{ij}t_i^h)^2 - \|\sum_i c_{ij}\boldsymbol{s}_i^h\|_2^2}}. \tag{66}$$

Recall the stereographic projection $\mathcal{S} : \mathbb{L}^{\kappa,d_\mathcal{T}} \to \mathbb{B}^{\kappa,d_\mathcal{T}}$:

$$\mathcal{S}([t_i^h, (\boldsymbol{s}_i^h)^T]) = \frac{\boldsymbol{s}_i^h}{1 + \sqrt{-\kappa}t_i^h}. \tag{67}$$

Then we apply $\mathcal{S}$ to both sides of Eq. (66) to obtain

$$\boldsymbol{b}_j^{h-1} = \frac{1}{\sqrt{-\kappa}} \frac{\sum_i c_{ij} \boldsymbol{s}_i^h}{\sqrt{(\sum_i c_{ij} t_i^h)^2 - \|\sum_i c_{ij} \boldsymbol{s}_i^h\|_2^2} + \sum_i c_{ij} t_i^h} \tag{68}$$

$$= \frac{1}{\sqrt{-\kappa}} \frac{\frac{\sum_i c_{ij} \boldsymbol{s}_i^h}{\sum_i c_{ij} t_i^h}}{1 + \sqrt{1 - \frac{\|\sum_i c_{ij} \boldsymbol{s}_i^h\|_2^2}{(\sum_i c_{ij} t_i^h)^2}}} \tag{69}$$

$$= \frac{\bar{\boldsymbol{b}}^h}{1 + \sqrt{1 + \kappa \|\bar{\boldsymbol{b}}^h\|_2^2}}, \tag{70}$$

where $\boldsymbol{b}_j^{h-1} = \mathcal{S}(\boldsymbol{z}_j^{h-1})$, and $\bar{\boldsymbol{b}}^h = \frac{1}{\sqrt{-\kappa}} \frac{\sum_i c_{ij} \boldsymbol{s}_i^h}{\sum_i c_{ij} t_i^h}$. Let $\boldsymbol{b}_i^h = \mathcal{S}(\boldsymbol{z}_i^h)$, then

$$\bar{\boldsymbol{b}}^h = 2 \frac{\sum_i c_{ij} \frac{\boldsymbol{b}_i^h}{1+\kappa\|\boldsymbol{b}_i^h\|_2^2}}{\sum_i c_{ij} \frac{1-\kappa\|\boldsymbol{b}_i^h\|_2^2}{1+\kappa\|\boldsymbol{b}_i^h\|_2^2}} = \frac{\sum_i c_{ij} \lambda^\kappa_{\boldsymbol{b}_i^h} \boldsymbol{b}_i^h}{\sum_i c_{ij} (\lambda^\kappa_{\boldsymbol{b}_i^h} - 1)}, \tag{71}$$

which is the gyro-midpoint in Poincaré ball model. The proof is completed. □

C. Technical Details

C.1. Notations

The mathematical notations are described in Table 4.

C.2. Algorithm

The Algorithm 2 is the decoding algorithm to recover a partitioning tree from hyperbolic embeddings and level-wise assignment matrices. we perform this process in a Breadth First Search manner. As we create a tree node object, we put it into a simple queue. Then we search the next level of the tree nodes and create children nodes according to the level-wise assignment matrices and hyperbolic embeddings. Finally, we add these children nodes to the first item in the queue and put them into the queue. For convenience, we can add some attributes into the tree node objects, such as node height, node number, and so on.

In Algorithm 3, we give the approach to obtain node clusters of a predefined cluster number from a partitioning tree $\mathcal{T}$. The key idea of this algorithm is that we first search the first level of the tree if the number of nodes is more than the predefined numbers, we merge the nodes that are farthest away from the root node, and if the number of nodes is less than the predefined numbers, we search the next level of the tree, split the nodes that closest to the root. As we perform this iteratively, we will finally get the ideal clustering results. The way we merge or split node sets is that: the node far away from the root tends to contain fewer points, and vice versa.

C.3. Datasets & Baselines

We evaluate our model on a variety of datasets, i.e., KarateClub, FootBall, Cora, Citeseer, Amazon-Photo (AMAP), and a larger Computer. All of them are publicly available. We give the statistics of the datasets in Table 1 as follows.

In this paper, we compare with deep graph clustering methods and self-supervised GNNs introduced as follows,

- **RGC** (Liu et al., 2023a) enables the deep graph clustering algorithms to work without the guidance of the predefined cluster number.
- **DinkNet** (Liu et al., 2023b) optimizes the clustering distribution via the designed dilation and shrink loss functions in an adversarial manner.
- **Congregate** (Sun et al., 2023b) approaches geometric graph clustering with the re-weighting contrastive loss in the proposed heterogeneous curvature space.

Algorithm 2 Decoding a partitioning tree from hyperbolic embeddings in BFS manner.

Input: Hyperbolic embeddings $\mathbf{Z}^h = \{\mathbf{z}_1^h, \mathbf{z}_2^h, ..., \mathbf{z}_N^h\}$ for $h = 1, ..., H$; The height of tree H; The matrix $\mathbf{S}^h$ computed in Equation 5; List of node attributions $node\_attr$, including $node\_set$, $children$, $height$, $coordinates$; Abstract class Node;

Function `ConstructTree`(**Z**, H, $node\_attr$) **:**
- $h = 0$;
- # First create a root node.
- $root = \text{Node}(node\_attr[h])$;
- # Create a simple queue.
- $que = \text{Queue}()$;
- $que.\text{put}(root)$;
- **while** *que is not empty* **do**
  - $node = que.\text{get}()$;
  - $node\_set = node.node\_set$
  - $k = node.height + 1$;
  - # If in the last level of the tree
  - **if** $h == H$ **then**
    - **for** i *in* $node\_set$ **do**
      - $child = \text{Node}(node\_attr[h][i])$
      - $child.coordinates = \mathbf{Z}_i^h$
      - $node.children.\text{append}(child)$
    - **end**
  - **else**
    - **for** $k = 1$ *to* N_h **do**
      - L_child = [i in where $S_{ik}^h == 1$]
      - **if** $\text{len}(L\_child) > 0$ **then**
        - $child = \text{Node}(node\_attr[h][L\_child])$
        - $child.coordinates = \mathbf{Z}_{[L\_child]}^h$
        - $node.children.\text{append}(child)$
        - $que.\text{put}(child)$
      - **end**
    - **end**
  - **end**
- **end**
- **return** $root$;

Output: `ConstructTree`(**Z**, H, $node\_attr$)

- **GC-Flow** (Wang et al., 2023) models the representation space by using Gaussian mixture, leading to an anticipated high quality of node clustering.
- **S^{3}GC** (Devvrit et al., 2022) introduces a salable contrastive loss in which the nodes are clustered by the idea of walktrap, i.e., random walk trends to trap in a cluster.
- **MCGC** (Pan & Kang, 2021) learns a new consensus graph by exploring the holistic information among attributes and graphs rather than the initial graph.
- **DCRN** (Liu et al., 2022a) designs the dual correlation reduction strategy to alleviate the representation collapse problem.
- **Sublime** (Liu et al., 2022b) guides structure optimization by maximizing the agreement between the learned structure and a self-enhanced learning target with contrastive learning.
- **gCooL** (Li et al., 2022) clusters nodes with a refined modularity and jointly train the cluster centroids with a bi-level contrastive loss.

- **FT-VGAE** (Mrabah et al., 2022) makes effort to eliminate the feature twist issue in the autoencoder clustering architecture by a solid three-step method.
- **ProGCL** (Xia et al., 2022) devises two schemes (ProGCL-weight and ProGCLmix) for further improvement of negatives-based GCL methods.
- **MVGRL** (Hassani & Ahmadi, 2020) contrasts across the multiple views, augmented from the original graph to learn discriminative encodings.
- **ARGA** (Pan et al., 2018) regulates the graph autoencoder with a novel adversarial mechanism to learn informative node embeddings.
- **GRACE** (Yang et al., 2017) has multiples nonlinear layers of deep denoise autoencoder with an embedding loss, and is devised to learn the intrinsic distributed representation of sparse noisy contents.
- **DEC** (Xie et al., 2016) uses stochastic gradient descent (SGD) via backpropagation on a clustering objective to learn the mapping.
- **VGAE** (Kipf & Welling, 2016) makess use of latent variables and is capable of learning interpretable latent representations for undirected graphs.

C.4. Implementation Notes

`LSEnet` is implemented upon PyTorch 2.0 [3], Geoopt [4], PyG [5] and NetworkX [6]. The dimension of structural information is a hyperparameter, which is equal to the height of partitioning tree. The hyperbolic partitioning tree is learned in a $2-$dimensional hyperbolic space of Lorentz model by default. For the training phase, we use Riemannian Adam (Becigneul & Ganea, 2018) and set the learning rate to 0.003.

The loss function of `LSEnet` is the differentiable structural information (DSI) formulated in Sec. 4.2 of our paper. We suggest to pretrain LConv of our model with an auxiliary link prediction loss, given as follows,

$$\mathcal{L}_{link} = \frac{1}{\#(i,j)} \sum_{(i,j)} \mathbf{I}[(i,j) \in \mathcal{E}] \log Pr((i,j) \in \mathcal{E}) + (1 - \mathbf{I}[(i,j) \in \mathcal{E}]) \log[1 - Pr((i,j) \in \mathcal{E})] \tag{72}$$

where $\#(i,j)$ is the number of the sampling nodes pairs. The probability is defined as $Pr((i,j) \in \mathcal{E}) = \frac{1}{1+\exp\left(-\frac{s - d_{\mathbb{L}}(\boldsymbol{z}_i, \boldsymbol{z}_j)}{\tau}\right)}$, and $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ are the leaf node embedding of the hyperbolic partitioning tree.

D. Additional Results

The hyperbolic partitioning trees of Cora is visualized in Fig. 6, where different clusters are distinguished by colors.

[3] https://pytorch.org/
[4] https://geoopt.readthedocs.io/en/latest/index.html
[5] https://pytorch-geometric.readthedocs.io/en/latest/
[6] https://networkx.org/

Notation	Description
G	Graph
$\mathcal{V}$	Graph nodes set
$\mathcal{E}$	Graph edges set
$\mathbf{X}$	Node attributes matrix
$\mathbf{A}$	Graph adjacency matrix
N	Graph node number
w	Edge weight function
v_i	A graph node
d_i	The degree of graph node v_i
$\mathrm{Vol}(\cdot)$	The volume of graph or subset of graph
$\mathcal{U}$	A subset of $\mathcal{V}$
K	Number of clusters
$\mathbb{L}^{\kappa,d}$	d-dimensional Lorentz model with curvature κ
$\langle\cdot,\cdot\rangle_{\mathbb{L}}$	Lorentz inner product
$\mathbf{R}$	Riemannian metric tensor
$\boldsymbol{x}$	Vector or point on manifold
$d_{\mathbb{L}}(\cdot,\cdot)$	Lorentz distance
$\|\cdot\|_{\mathbb{L}}$	Lorentz norm
$\|\cdot\|$	Euclidean norm
H	Dimension of structural information
$\mathcal{T}$	A partitioning tree
α	A Non-root node of $\mathcal{T}$
α^-	The immediate predecessor of α
λ	Root node of $\mathcal{T}$
T_α	The module of α, a subset of $\mathcal{V}$ associated with α
g_α	The total weights of graph edges with exactly one endpoint in module T_α
$\mathcal{H}^{\mathcal{T}}(G)$	Structural information of G w.r.t. $\mathcal{T}$
$\mathcal{H}^{\mathcal{T}}(G;\alpha)$	Structural information of tree node α
$\mathcal{H}^{H}(G)$	H-dimensional structural entropy of G
$\mathcal{H}^{\mathcal{T}}(G;h)$	Structural information of all tree node at level h
$\mathbf{C}^h$	The level-wise assignment matrix between h and $h-1$ level
N_h	The maximal node numbers in h-level of $\mathcal{T}$
V_k^h	The volume of graph node sets T_k
$E(p_1,...,p_n)$	Entropy function w.r.t. distribution $(p_1,...,p_n)$
$\Phi(G)$	Graph conductance
$\mathbf{Z}$	Tree node embeddings matrix
$\mathbf{\Theta}$	Learnable parameters of `LSEnet`
$\mathcal{H}^{\mathcal{T}_{\mathrm{net}}}(G;\mathbf{Z};\mathbf{\Theta})$	The differentiable structural information (DSI)
$\mathcal{T}_{\mathrm{net}}$	The partitioning tree decode from `LSEnet`
$\mathcal{O}(\cdot)$	Equivalent infinitesimal
$O(\cdot)$	The time complexity

Table 4. The notation descriptions.

Datasets	# Nodes	# Features	# Edges	# Classes
KarateClub	34	34	156	4
FootBall	115	115	1226	12
Cora	2708	1433	5278	7
Citeseer	3327	3703	4552	6
AMAP	7650	745	119081	8
AMAC	13752	767	245861	10

Table 5. The statistics of the datasets.

Algorithm 3 Obtaining objective cluster numbers from a partitioning tree.

Input: A partitioning tree $\mathcal{T}$; Tree node embeddings $\mathbf{Z}$ in hyperbolic space; Objective cluster numbers K.

Let λ be the root of the tree.
for u in $\mathcal{T}.nodes$ / λ **do**
  Compute distance to λ, i.e., $d_{\mathbb{L}}(\boldsymbol{z}_\lambda, \boldsymbol{z}_u)$.
end for
Sorted non-root tree nodes by distance to λ in ascending order, output a sorted list L.
Let $h = 1$ and count the number M of nodes at height h.
while $M > K$ **do**
  Merge two nodes u and v that are farthest away from root λ.
  Compute midpoint p of u and v, and compute $d_{\mathbb{L}}(\boldsymbol{z}_\lambda, \boldsymbol{z}_p)$.
  Add p into L and sort the list again in ascending order.
  $M = M - 1$.
end while
while $M < K$ **do**
  for v in L **do**
    Let $h = h + 1$.
    Search children nodes of v in h-level as sub-level list S and count the number of them as m.
    $M = M + m - 1$.
    if $M > K$ **then**
      Perform merge operation to S the same as it to L.
      Delete v from L, and add S to L.
      Break the for-loop.
    else if $M = K$ **then**
      Delete v from L, and add S to L.
      Break the for-loop.
    else
      Delete v from L, and add S to L.
    end if
  end for
end while
Result set $R = \{\}$
for $i = 0$ to $K - 1$ **do**
  Get graph node subset Q from $L[i]$.
  Assign each element of Q a clustering category i.
  Add results into R.
end for

Output: R

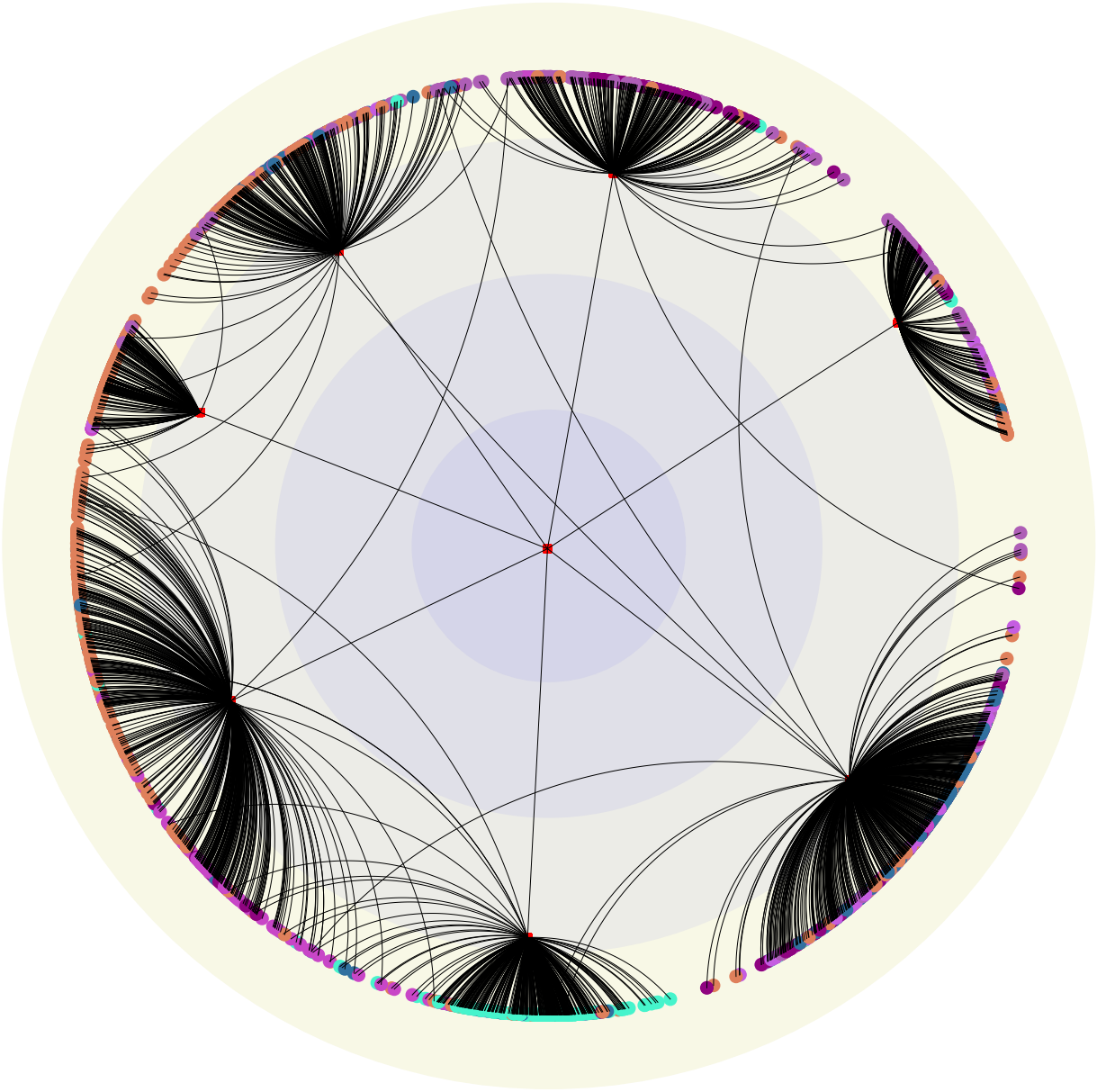


Figure 6. Visualization of hyperbolic partitioning trees of Cora.